التعلم العميق

عن طريق الامثلة

٥٠ مشروع تعلم عميق تم حلها وشرحها باستخدام بايثون

ترجمة واعداد: د. علاء طعيمة

بـــمه تعالى

التعلم العميق: عن طريق الامثلة

50 مشروع تعلم عميق تم حلها وشرحها باستخدام بايثون

ترجمة واعداد:

د. علاء طعيمة

مقدمة المؤلف

يعمل التعلم العميق على إحداث ثورة تدريجيةفي كل مجال من مجالات الذكاء الاصطناعي، مما يجعل تطوير التطبيقات أسهل.

في هذا الكتاب، تنقل مشاريع التعلم العميق باستخدام بايثون كل المعرفة اللازمة لتنفيذ مشاريع التعلم العميق المعقدةفي مختلف مجالات التعلم العميق والرؤية الحاسوبية. كل مشروع من هذه المشاريع فريد من نوعه، مما يساعدك على إتقان الموضوع تدريجيًا. ستتعلم كيفية تشخيص مجموعة متنوعة من الامراض باستخدام نموذج الشبكة العصبية التلافيفية (CNN) وكذلك ستتعلم الكثير من المشاريع الجذابة الاخرى الي ستساعدكفي اكتساب المعرفة لتطوير أنظمة التعلم العميق الخاصة بك بطريقة مباشرة وفعالة.

لقد حاولت قدر المستطاع ان اترجم المشاريع الأكثر طرحاًفي مجال التعلم العميق مع الشرح المناسب والكافي، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذاكان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدنا الالكتروني alaa.taima@qu.edu.iq .

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخلفي مجالات التعلم الآلي والتعلم العميق وعلم البيانات ومساعدة القارئ العربي على تعلم هذا المجالات. اسأل الله التوفيقفي هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصينفي مجال التعلم الآلي والتعلم العميق وعلم البيانات. ونرجو لك الاستمتاع مع الكتاب ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية

العراق

المحتويات

1) تصنيف الصور الطبية باستخدام CNN Disease Classification with CNN

الهدف من هذه الدراسة هو تصنيف الصور الطبية (Medical MNIST) باستخدام نموذج الشبكة العصبية التلافيفية (CNN).

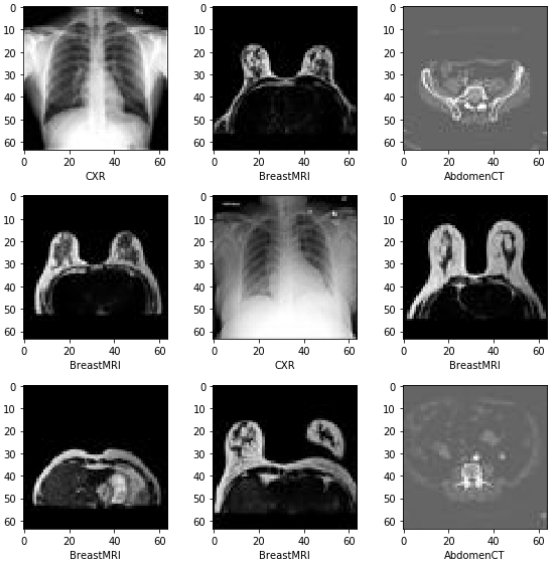
هنا، قمت بتدريب نموذج CNN باستخدام مجموعة بيانات جيدة المعالجة للصور الطبية. يمكن استخدام هذا النموذج لتصنيف الصور الطبية بناءً على الفئات المقدمة وفقًا لمجموعة بيانات التدريب.

الكود هنا ...

مجموعة البيانات

تم تطوير مجموعة البيانات هذهفي عام 2017 بواسطة Arturo Polanco Lozano. تُعرف أيضًا باسم مجموعة بيانات MedNIST للأشعة والتصوير الطبي. لإعداد مجموعة البيانات هذه، تم جمع الصور من عدة مجموعات بيانات، وهي TCIA و RSNA Bone Age Challange وNIH Chest Chest X-ray dataset.

تحتوي مجموعة البيانات هذه على 58954 صورة طبية تنتمي إلى 6 فئات: AbdomenCT (10000 صورة)، HeadCT (10000 صورة)، Hand (10000 صورة)، CXR (10000 صورة)، CXR (10000 صورة)، BreastMRI (8954 صورة)، ChestCT (10000 صورة).



حجم مجموعة البيانات 75.98 ميجا بايت.

استيراد مجموعة البيانات

```
!wget -N "https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-admin/MedNIST.zip"
!unzip -qo "MedNIST.zip"
!rm "MedNIST.zip"
```

تخصيص أدلة الاختبار والتدريب

```
test_dir = "Medical/Medical_test"
train_dir = "Medical/Medical_train"
```

استيراد المكتبات

```
import os
import numpy as np
import pandas as pd
import random, datetime, os, shutil, math
```

تحضير البيانات

في مجموعة البيانات هذه، تم تزويدنا بستة أدلة (مجلدات) مختلفة يتكون كل منها من عشرات الآلاف من الصور. ومع ذلك، من أجل تغذية هذه البياناتفي نموذج CNN، نحتاج أولاً إلى استخلاصهافي مجموعة التدريب والاختبار منه.

هنا قمنا بالفعل بإنشاء دليل فارغ باسم "test_dir" مع جميع المجلدات الفرعية فيه كما هو الحالفي "train_dir" و "train_dir" يتكون من جميع مجلدات الصور.

```
def prep_test_data(med, train_dir, test_dir):
  pop = os.listdir(train_dir+'/'+med)
  test_data=random.sample(pop, 2000)
  #print(test_data)
  for f in test_data:
    shutil.copy(train_dir+'/'+med+'/'+f, test_dir+'/'+med+'/')
```

```
for medi in os.listdir(train_dir):
  prep_test_data(medi, train_dir, test_dir)
```

بعد إنشاء مجموعة الاختبار، يمكننا التحقق مما إذا كان كلا المجلدين، أي "train_dir" و "test_dir" لهما نفس عدد الفئات.

```
#for train
target_classes = os.listdir(train_dir)
num_classes = len(target_classes)
print('Number of target classes:', num_classes)
print(list(enumerate(target_classes)))
```

```
Number of target classes: 6
[(0, 'AbdomenCT'), (1, 'ChestCT'), (2, 'Hand'), (3, 'HeadCT'), (4, 'CXR'), (5, 'BreastMRI')]
```

```
#for test
target_classes = os.listdir(test_dir)
num_classes = len(target_classes)
print('Number of target classes:', num_classes)
print(list(enumerate(target_classes)))
```

```
Number of target classes: 6
[(0, 'AbdomenCT'), (1, 'ChestCT'), (2, 'Hand'), (3, 'HeadCT'), (4, 'CXR'),
(5, 'BreastMRI')]
```

تعيين توزيعات التدريب والاختبار

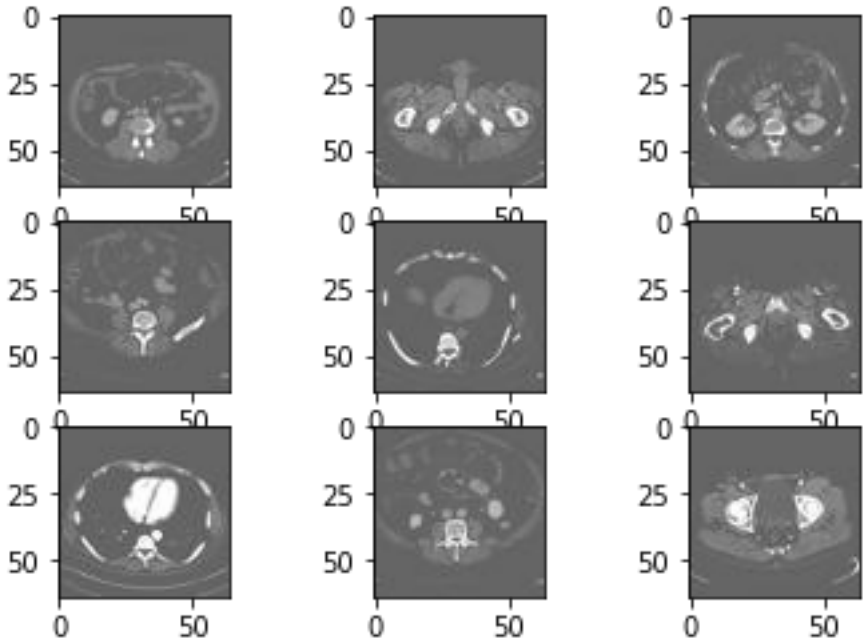
```
training_set_distribution = [len(os.listdir(os.path.join(train_dir,
dir))) for dir in os.listdir(train_dir)]
testing_set_distribution = [len(os.listdir(os.path.join(test_dir,
dir))) for dir in os.listdir(test_dir)]
```

عرض صورة كعينة

```
def show_mri(med):
  num = len(med)
  if num == 0:
    return None
  rows = int(math.sqrt(num))
  cols = (num+1)//rows
  f, axs = plt.subplots(rows, cols)
  fig = 0
  for b in med:
    img = image.load_img(b)
    row = fig // cols
    col = fig % cols
    axs[row, col].imshow(img)
    fig += 1
  plt.show()
```

```
import matplotlib.pyplot as plt
from matplotlib.image import imread
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
```

```
dir_name = os.path.join(train_dir,"AbdomenCT")
all_images = [os.path.join(dir_name, fname) for fname in
os.listdir(dir_name)]
show_mri(all_images[:9])
```



المعالجة المسبقة للصور

- image_size(height, width) كـ (32 ، 32).في مجموعة البيانات هذه، كان لدينا ما يقرب من 10000 صورة لكل فئة. علاوة على ذلك، كانت جميع الصور أنواعًا من الصور الطبية مثل الأشعة السينية أو الأشعة المقطعية أو التصوير بالرنين المغناطيسي. ومعظم أجزاء هذه الأنواع من الصور ضبابية. ومن ثم، فإن الحجم الأكبر بكثير لن يضيف قيمة أكبر بالمقارنة مع زيادة وقت التدريب بسبب زيادة حجم الصورة.
- batch_size() ، هو معلمة فائقة تحدد ببساطة عدد العينات التي يجب العمل من خلالها قبل تحديث معلمات النموذج الداخلي.
- class_mode()، هنا استخدمت class_mode كـ "categorical". نظرًا لأنه يحدد نوع مصفوفات التسميات التي يتم إرجاعها، هنافي حالة "categorical"، ستكون تسميات مشفرات واحد ساخن ثنائية الأبعاد.

```
image_size = (32, 32, 3)
datagen=ImageDataGenerator(rescale = 1./255,
                           shear_range=0.2,
                           zoom_range=0.2,
                           horizontal_flip=True,
                           )
```

```
training_set=datagen.flow_from_directory(train_dir,
                                         target_size=image_size[:2],
                                         batch_size=32,
```

```
                                        class_mode='categorical',
                                        shuffle=False
                                        #color_mode='rgb'
                                        )
```

```
validation_set=datagen.flow_from_directory(test_dir,
                                           target_size=image_size[:2],
                                           batch_size=32,
                                           class_mode='categorical',
                                           shuffle=False
                                           )
```

استيراد واستخدام Callbacks

استخدمنا هنا عدة Callbacks، Callbacks هو ببساطة كائن يمكنه تنفيذ إجراءاتفي مراحل مختلفة من التدريب (على سبيل المثال،في بداية أو نهاية حقبة (فترة) ما، قبل أو بعد دفعة واحدة، إلخ).

- Early Stopping: يستخدم لإيقاف التدريب عندما يتوقف القياس المرصود عن التحسن.
- Reduce LR On Plateau: يستخدم لتقليل معدل التعلم عندما يتوقف أحد المقاييس عن التحسن.
- Model Checkpoin: يستخدم لحفظ نموذج Keras أو أوزان النموذج عند بعض التردد.

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import plot_model
```

```
es = EarlyStopping(monitor='val_acc', mode='max', verbose=1, patience=7)
filepath = "modelMedicalMNIST.h5"
ckpt = ModelCheckpoint(filepath, monitor='acc', verbose=1,
save_best_only=True, mode='max')
rlp = ReduceLROnPlateau(monitor='acc', patience=3, verbose=1)
```

بناء النموذج

بالنسبة لنموذج تصنيف الصور هذا، قمنا بتعريف نموذج الشبكة العصبية التلافيفية CNN بسبع طبقات. هنا، طبقتان بهما ReLU (وحدة خطية مصححة) والأخيرة لها دالة تنشيط "Softmax".

```
def cnn(image_size, num_classes):
```

```
    classifier = Sequential()
    classifier.add(Conv2D(64, (5, 5), input_shape=image_size,
activation='relu', padding='same'))
    classifier.add(MaxPooling2D(pool_size = (2, 2)))
    classifier.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    classifier.add(MaxPooling2D(pool_size = (2, 2)))
    classifier.add(Flatten())
    classifier.add(Dense(num_classes, activation = 'softmax'))
    classifier.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['acc'])
    return classifier

neuralnetwork_cnn = cnn(image_size, num_classes)
neuralnetwork_cnn.summary()
#plot_model(neuralnetwork_cnn, show_shapes=True)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 64)        4864
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 6)                 49158
=================================================================
Total params: 127,878
Trainable params: 127,878
Non-trainable params: 0
_________________________________________________________________
```

استخدمنا هنا "softmax"، يتم استخدام دالة softmax كدالة تنشيطفي طبقة الإخراج لنماذج الشبكة العصبية التي تتنبأ بتوزيع احتمالي متعدد الحدود. هذا، كمافي هذه الحالة، كان لدينا فئات متعددة من الصور. علاوة على ذلك، يُعد softmax مفيدًا جدًا لأنه يحول الدرجات إلى توزيع احتمالي عادي، والذي يمكن عرضه لاحقًا للمستخدم أو يمكن استخدامه كمدخل لأنظمة أخرى.

علاوة على ذلك، تم تجميع النموذج باستخدام دالة خطأ الانتروبيا الفئوية categorical cross-entropy.

يتم استخدام "Adam" كمحسِّن، نظرًا لقدرته على ضبط معدل التعلم للنموذج من تلقاء نفسه وفقًا للموقف.

تطبيق النموذج

```
history = neuralnetwork_cnn.fit_generator(
    generator=training_set, validation_data=validation_set,
    callbacks=[es, ckpt, rlp], epochs = 5,
)
```

```
Epoch 1/5
1841/1843 [============================>.] - ETA: 0s - loss: 0.2316 - acc: 0.9245
Epoch 00001: acc improved from -inf to 0.92455, saving model to modelMedicalMNIST.h5
1843/1843 [==============================] - 49s 27ms/step - loss: 0.2314 - acc: 0.9246 - val_loss: 0.0340 - val_acc: 0.9930
Epoch 2/5
1841/1843 [============================>.] - ETA: 0s - loss: 0.0330 - acc: 0.9917
Epoch 00002: acc improved from 0.92455 to 0.99167, saving model to modelMedicalMNIST.h5
1843/1843 [==============================] - 49s 26ms/step - loss: 0.0329 - acc: 0.9917 - val_loss: 0.0234 - val_acc: 0.9949
Epoch 3/5
1841/1843 [============================>.] - ETA: 0s - loss: 0.0412 - acc: 0.9886
Epoch 00003: acc did not improve from 0.99167
1843/1843 [==============================] - 49s 26ms/step - loss: 0.0412 - acc: 0.9886 - val_loss: 0.0179 - val_acc: 0.9958
Epoch 4/5
1841/1843 [============================>.] - ETA: 0s - loss: 0.0159 - acc: 0.9963
Epoch 00004: acc improved from 0.99167 to 0.99632, saving model to modelMedicalMNIST.h5
1843/1843 [==============================] - 49s 26ms/step - loss: 0.0159 - acc: 0.9963 - val_loss: 0.0081 - val_acc: 0.9977
Epoch 5/5
1841/1843 [============================>.] - ETA: 0s - loss: 0.0308 - acc: 0.9947
Epoch 00005: acc did not improve from 0.99632
1843/1843 [==============================] - 49s 26ms/step - loss: 0.0307 - acc: 0.9947 - val_loss: 0.0082 - val_acc: 0.9978
```

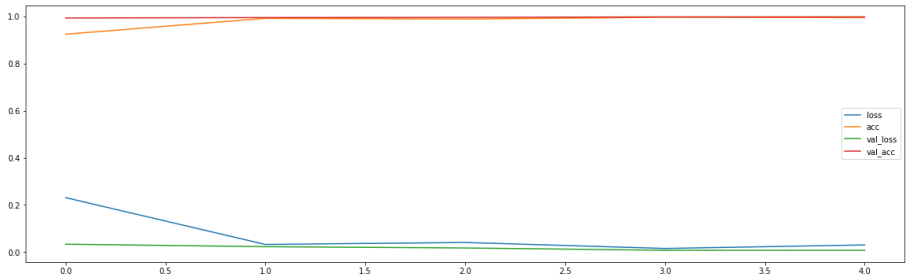
حقق النموذج دقة استثنائية بلغت 99٪في مجموعة التحقق و99٪ من الدقةفي مجموعة الاختبار.

رسم النتائج

الرسم البياني بين الخطأ والدقة وخطأ التحقق من الصحة ودقة التحقق من الصحة.

- تمثل "loss " خطأ التدريب.
- يمثل "acc" دقة التدريب.
- يمثل "val_loss" خطأ التحقق من الصحة.
- يمثل "val_acc" دقة التحقق من الصحة.

```
fig, ax = plt.subplots(figsize=(20, 6))
pd.DataFrame(history.history).iloc[:, :-1].plot(ax=ax)
```



التنبؤ

التنبؤ على مجموعة التحقق من الصحة:

```
batch_size=32
pred=neuralnetwork_cnn.predict_generator(validation_set,steps=306/batch_size)
predicted_class_indices=np.argmax(pred,axis=1)
```

```
labels = (validation_set.class_indices)
labels = dict((v,k) for k,v in labels.items())
predictions = [labels[k] for k in predicted_class_indices]
```

عرض فئة الصورة والصورة المتوقعة:

```
filenames=validation_set.filenames[0]
results=pd.DataFrame({"Filename":filenames,
                      "Predictions":predictions})
```

```
display(results.head(15))
```

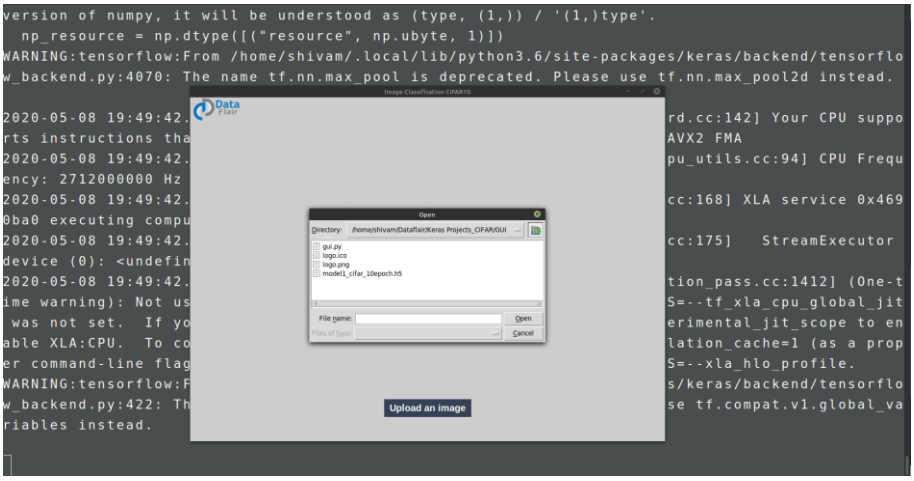
	Filename	Predictions
0	AbdomenCT/000001.jpeg	AbdomenCT
1	AbdomenCT/000001.jpeg	AbdomenCT
2	AbdomenCT/000001.jpeg	AbdomenCT
3	AbdomenCT/000001.jpeg	AbdomenCT
4	AbdomenCT/000001.jpeg	AbdomenCT
5	AbdomenCT/000001.jpeg	AbdomenCT
6	AbdomenCT/000001.jpeg	ChestCT
7	AbdomenCT/000001.jpeg	AbdomenCT
8	AbdomenCT/000001.jpeg	AbdomenCT
9	AbdomenCT/000001.jpeg	AbdomenCT
10	AbdomenCT/000001.jpeg	AbdomenCT
11	AbdomenCT/000001.jpeg	AbdomenCT
12	AbdomenCT/000001.jpeg	ChestCT
13	AbdomenCT/000001.jpeg	AbdomenCT
14	AbdomenCT/000001.jpeg	AbdomenCT
15	AbdomenCT/000001.jpeg	AbdomenCT

رابط الكود : انقر هنا.

2) تصنيف الصور: مشروع التعلم العميق في بايثون مع Keras

تصنيف الصور Image classification هو مشروع تعلم عميق رائع. على وجه التحديد، يندرج تصنيف الصور ضمن فئة مشروع الرؤية الحاسوبية.

في هذا المشروع، سنبني شبكة عصبية التفافية CNN في Keras مع بايثون على مجموعة بيانات CIFAR-10. أولاً، سوف نستكشف مجموعة البيانات الخاصة بنا، ثم سنقوم بتدريب شبكتنا العصبية باستخدام بايثون و Keras.



ما هو تصنيف الصور؟

- تتمثل مشكلة التصنيف في تصنيف جميع وحدات البكسل للصورة الرقمية في إحدى الفئات المحددة.
- تصنيف الصور هو حالة الاستخدام الأكثر أهمية في تحليل الصور الرقمية.
- تصنيف الصور هو تطبيق لكل من التصنيف الخاضع للإشراف والتصنيف غير الخاضع للإشراف.
  - في التصنيف الخاضع للإشراف ، نختار عينات لكل فئة مستهدفة. نقوم بتدريب شبكتنا العصبية على عينات الفئة المستهدفة هذه ثم نصنف عينات جديدة.
  - في التصنيف غير الخاضع للإشراف، نقوم بتجميع عينات الصور في مجموعات من الصور لها خصائص متشابهة. ثم نصنف كل مجموعة في فئاتنا المقصودة.

حول مجموعة بيانات تصنيف الصور

CIFAR-10 هي مجموعة بيانات رؤية حاسوبية مشهورة جداً. تمت دراسة مجموعة البيانات هذه جيداًفي العديد من أنواع أبحاث التعلم العميق للتعرف على الأشياء.

تتكون مجموعة البيانات هذه من 60.000 صورة مقسمة إلى 10 فئات مستهدفة، وتحتوي كل فئة على 6000 صورة ذات شكل 32 * 32. تحتوي مجموعة البيانات هذه على صور ذات دقة منخفضة (32 * 32)، مما يسمح للباحثين بتجربة خوارزميات جديدة. الفئات العشر المختلفة لمجموعة البيانات هذه هي:

1. Airplane
2. Car
3. Bird
4. Cat
5. Deer
6. Dog
7. Frog
8. Horse
9. Ship
10. Truck

تتوفر مجموعة بيانات CIFAR-10 بالفعل في وحدة مجموعات البيانات في Keras. لا نحتاج إلى تنزيله؛ يمكننا استيراده مباشرة من keras.datasets.

متطلبات المشروع

الشرط الأساسي لتطوير وتنفيذ مشروع تصنيف الصور هو تثبيت Keras و Tensorflow.

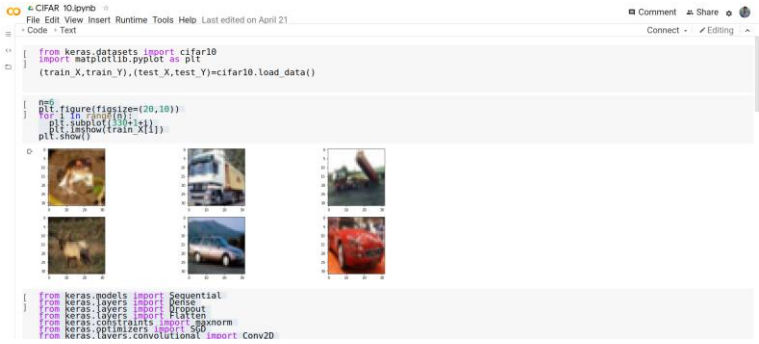
خطوات تصنيف الصور على CIFAR-10:

الخطوة 1: تحميل مجموعة البيانات من وحدة مجموعات بيانات keras

```
from keras.datasets import cifar10
import matplotlib.pyplot as plt

(train_X,train_Y),(test_X,test_Y)=cifar10.load_data()
```

الخطوة 2: رسم بعض الصور من مجموعة البيانات للتمثيل المرئي لمجموعة البيانات

```
n=6
plt.figure(figsize=(20,10))
for i in range(n):
plt.subplot(330+1+i)
plt.imshow(train_X[i])
plt.show()
```



الخطوة 3: استيراد الطبقات والوحدات المطلوبة لإنشاء بُنية الشبكة العصبية الالتفافية الخاصة بنا

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
```

الخطوة 4: تحويل قيم البكسل لمجموعة البيانات إلى نوع float ثم تسوية مجموعة البيانات

```
train_x=train_X.astype('float32')
test_X=test_X.astype('float32')

train_X=train_X/255.0
test_X=test_X/255.0
```

الخطوة 5: إجراء ترميز واحد ساخن للفئات المستهدفة

```
train_Y=np_utils.to_categorical(train_Y)
test_Y=np_utils.to_categorical(test_Y)

num_classes=test_Y.shape[1]
```

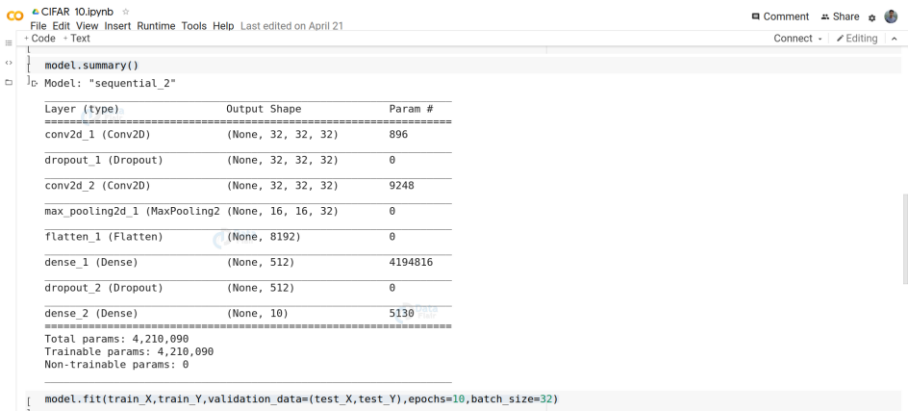
الخطوة 6: إنشاء النموذج المتسلسل وإضافة الطبقات

```
model=Sequential()
model.add(Conv2D(32,(3,3),input_shape=(32,32,3),
    padding='same',activation='relu',
    kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32,(3,3),activation='relu',padding='same',kernel_cons
traint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(512,activation='relu',kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

الخطوة 7: تكوين المحسن وتجميع النموذج

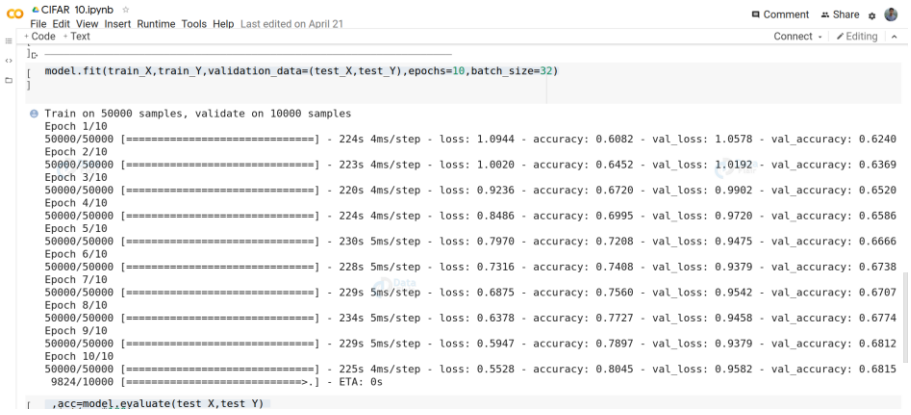
```
sgd=SGD(lr=0.01,momentum=0.9,decay=(0.01/25),nesterov=Fals)

model.compile(loss='categorical_crossentropy',
  optimizer=sgd,
  metrics=['accuracy'])
```

الخطوة 8: عرض ملخص النموذج من أجل فهم أفضل لبُنية النموذج



```
model.summary()
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               4194816
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 4,210,090
Trainable params: 4,210,090
Non-trainable params: 0
_________________________________________________________________
model.fit(train_X,train_Y,validation_data=(test_X,test_Y),epochs=10,batch_size=32)
```

الخطوة 9: تدريب النموذج



```
model.fit(train_X,train_Y,validation_data=(test_X,test_Y),epochs=10,batch_size=32)

Train on 50000 samples, validate on 10000 samples
Epoch 1/10
50000/50000 [==============================] - 224s 4ms/step - loss: 1.0944 - accuracy: 0.6082 - val_loss: 1.0578 - val_accuracy: 0.6240
Epoch 2/10
50000/50000 [==============================] - 223s 4ms/step - loss: 1.0020 - accuracy: 0.6452 - val_loss: 1.0192 - val_accuracy: 0.6369
Epoch 3/10
50000/50000 [==============================] - 220s 4ms/step - loss: 0.9236 - accuracy: 0.6720 - val_loss: 0.9902 - val_accuracy: 0.6520
Epoch 4/10
50000/50000 [==============================] - 224s 4ms/step - loss: 0.8486 - accuracy: 0.6995 - val_loss: 0.9720 - val_accuracy: 0.6586
Epoch 5/10
50000/50000 [==============================] - 230s 5ms/step - loss: 0.7970 - accuracy: 0.7208 - val_loss: 0.9475 - val_accuracy: 0.6666
Epoch 6/10
50000/50000 [==============================] - 228s 5ms/step - loss: 0.7316 - accuracy: 0.7408 - val_loss: 0.9379 - val_accuracy: 0.6738
Epoch 7/10
50000/50000 [==============================] - 229s 5ms/step - loss: 0.6875 - accuracy: 0.7560 - val_loss: 0.9542 - val_accuracy: 0.6707
Epoch 8/10
50000/50000 [==============================] - 234s 5ms/step - loss: 0.6378 - accuracy: 0.7727 - val_loss: 0.9458 - val_accuracy: 0.6774
Epoch 9/10
50000/50000 [==============================] - 229s 5ms/step - loss: 0.5947 - accuracy: 0.7897 - val_loss: 0.9379 - val_accuracy: 0.6812
Epoch 10/10
50000/50000 [==============================] - 225s 4ms/step - loss: 0.5528 - accuracy: 0.8045 - val_loss: 0.9582 - val_accuracy: 0.6815
 9824/10000 [============================>.] - ETA: 0s
,acc=model.evaluate(test_X,test_Y)
```

الخطوة 10: حساب الدقة في اختبار البيانات

```
acc=model.evaluate(test_X,test_Y)
print(acc*100)
```

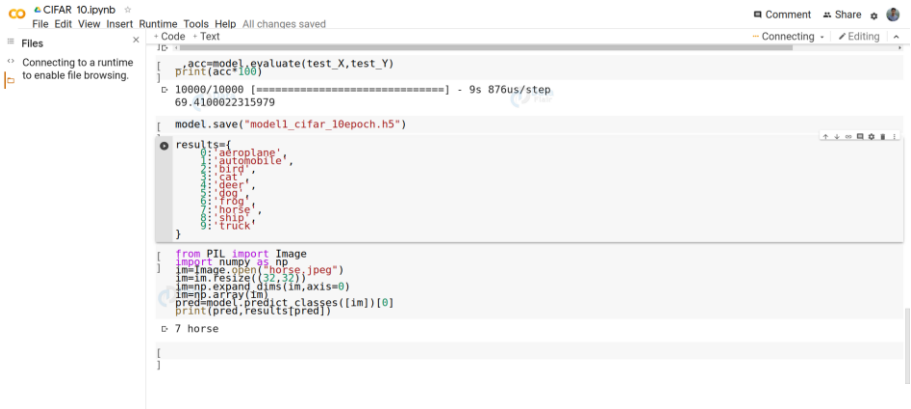
الخطوة 11: حفظ النموذج

```
model.save("model1_cifar_10epoch.h5")
```

الخطوة 12: عمل قاموس لتعيين فئات المخرجات وعمل تنبؤات من النموذج

```
results={
    0:'aeroplane',
    1:'automobile',
    2:'bird',
    3:'cat',
```

```
    4:'deer',
    5:'dog',
    6:'frog',
    7:'horse',
    8:'ship',
    9:'truck'
}
from PIL import Image
import numpy as np
im=Image.open("__image_path__")
# the input image is required to be in the shape of dataset, i.e
(32,32,3)

im=im.resize((32,32))
im=np.expand_dims(im,axis=0)
im=np.array(im)
pred=model.predict_classes([im])[0]
print(pred,results[pred])
```



يمكنك اختبار النتيجة على إدخال الصورة المخصص الخاص بك. لتحسين الدقة، حاول زيادة عدد الحقب (الفترات) إلى 25 للتدريب.

مشروع تصنيف الصور GUI

هنا، سنقوم ببناء واجهة مستخدم رسومية GUI لمصنف الصور الخاص بنا. سنقوم ببناء واجهة المستخدم الرسومية هذه باستخدام مكتبة بايثون Tkinter. لتثبيت Tkinker:

```
sudo apt-get install python3-tk
```

لجعل واجهة المستخدم الرسومية GUI تقوم بإنشاء ملف جديد gui.py ونسخ نموذجنا ("model1_cifar_10epoch.h5") إلى هذا الدليل.

الآن قم بلصق الكود أدناهفي ملف gui.py:

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import numpy
```

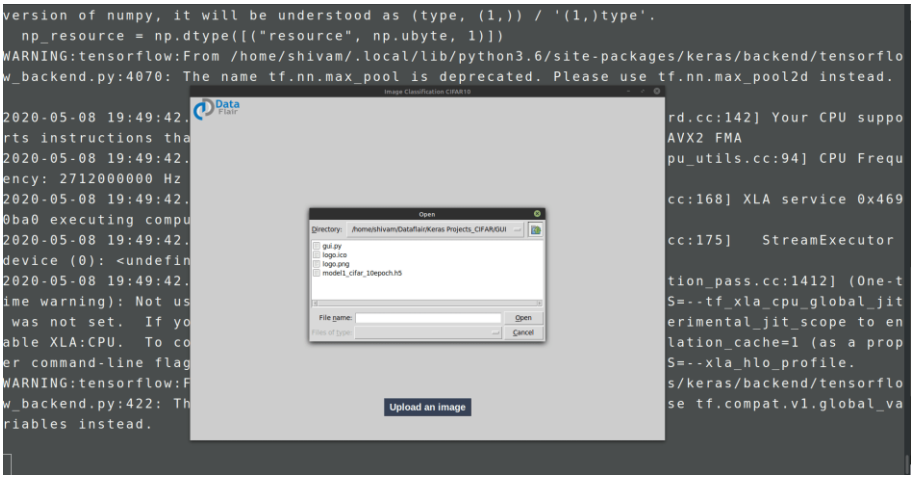
```
#load the trained model to classify the images

from keras.models import load_model
model = load_model('model1_cifar_10epoch.h5')

#dictionary to label all the CIFAR-10 dataset classes.

classes = {
    0:'aeroplane',
    1:'automobile',
    2:'bird',
    3:'cat',
    4:'deer',
    5:'dog',
    6:'frog',
    7:'horse',
    8:'ship',
    9:'truck'
}
#initialise GUI

top=tk.Tk()
top.geometry('800x600')
top.title('Image Classification CIFAR10')
top.configure(background='#CDCDCD')
label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)

def classify(file_path):
    global label_packed
    image = Image.open(file path)
    image = image.resize((32,32))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    pred = model.predict_classes([image])[0]
    sign = classes[pred]
    print(sign)
    label.configure(foreground='#011638', text=sign)

def show_classify_button(file_path):
    classify_b=Button(top,text="Classify Image",
   command=lambda: classify(file path),padx=10,pady=5)
    classify_b.configure(background='#364156', foreground='white',
font=('arial',10,'bold'))
    classify_b.place(relx=0.79,rely=0.46)

def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)
  uploaded.thumbnail(((top.winfo_width()/2.25),
    (top.winfo_height()/2.25)))
        im=ImageTk.PhotoImage(uploaded)
        sign image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass
```

```
upload=Button(top,text="Upload an image",command=upload_image,
  padx=10,pady=5)
```

```
upload.configure(background='#364156', foreground='white',
    font=('arial',10,'bold'))

upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Image Classification CIFAR10",pady=20,
font=('arial',20,'bold'))

heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

الآن قم بتشغيل ملف python gui.py لتنفيذ مشروع تصنيف الصور:

```
python3 gui.py
```



الملخص

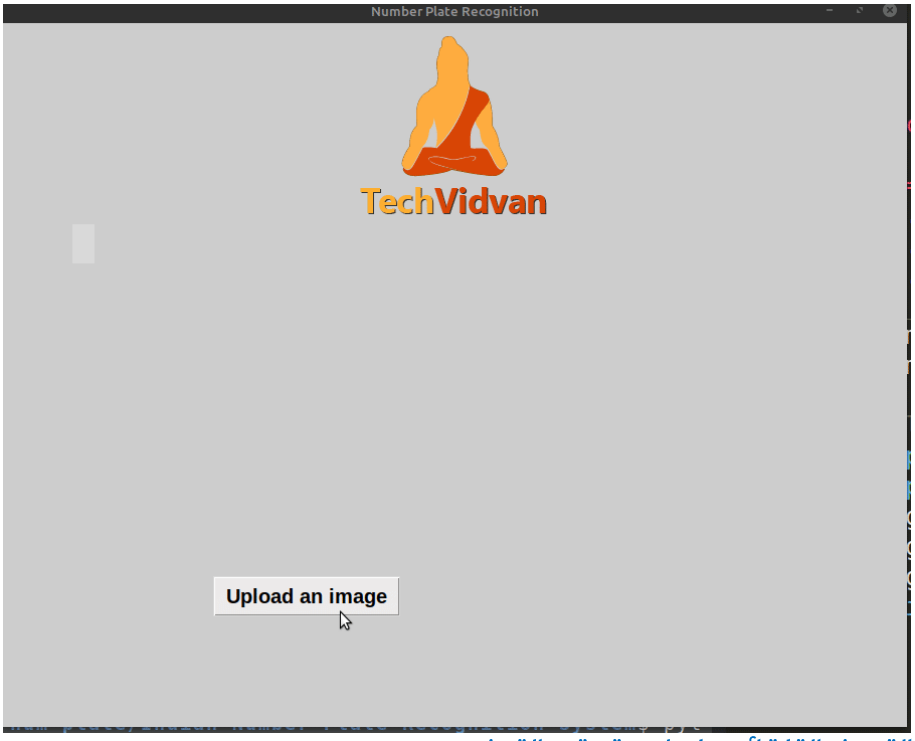
كان الهدف من مشروع تصنيف الصور هو تمكين المبتدئين من بدء العمل مع Keras لحل مشاكل التعلم العميق في الوقت الفعلي.

في مشروع keras للتعليم العميق، تحدثنا عن نموذج تصنيف الصور لتحليل الصور الرقمية. ناقشنا تصنيفات الصور الخاضعة للإشراف وغير الخاضعة للإشراف.

ثم شرحنا مجموعة بيانات CIFAR-10 وفئاتها. أخيرًا، رأينا كيفية بناء شبكة عصبية التفافية CNN لتصنيف الصور على مجموعة بيانات CIFAR-10.

3) التعرف التلقائي على لوحات ارقام الترخيص للسيارات باستخدام التعلم العميق Automatic License Number Plate Recognition using Deep Learning

يهدف هذا المشروع إلى التعرف على لوحات أرقام السيارات. من أجل الكشف عن لوحات أرقام الترخيص، سنستخدم OpenCV لتحديد لوحات الأرقام وبايثون pytesseract لاستخراج الأحرف والأرقام من لوحات الأرقام.



التعرف التلقائي على لوحة رقم الترخيص

OpenCV هي مكتبة مفتوحة المصدر للتعلم الآلي وتوفر بُنية تحتية مشتركة للرؤية الحاسوبية. في حين أن Pytesseract هو محرك Tesseract-OCR لقراءة أنواع الصور واستخراج المعلومات الموجودةفي الصورة.

تثبيت حزمة OpenCV و Pytesseract:

```
pip3 install opencv-python
pip3 install pytesseract
```

في مشروع بايثون هذا، لتحديد لوحة الأرقامفي صورة الإدخال، سنستخدم الميزات التالية لـ openCV:

- **Gaussian Blur**: هنا نستخدم نواة غاوسية لتنعيم الصورة. هذه التقنية فعالة للغاية لإزالة الضوضاء الغاوسية. يوفر OpenCV دالة cv2.GaussianBlur() لهذه المهمة.
- **Sobel**: هنا نحسب المشتقات من الصورة. هذه الميزة مهمة للعديد من مهام الرؤية الحاسوبية. باستخدام المشتقات نحسب التدرجات، ويشير التغيير الكبير في التدرج إلى تغيير كبير في الصورة. يوفر OpenCV دالة cv2.Sobel()لحساب عوامل تشغيل Sobel.
- **Morphological Transformation**: هذه هي العمليات التي تعتمد على أشكال الصور ويتم إجراؤها على الصور الثنائية. العمليات المورفولوجية الأساسية هي التآكل، التمدد ، الفتح ، الإغلاق. الدوال المختلفة المتوفرة في OpenCV هي:
  - cv2.erode()
  - cv2.dilate()
  - cv2.morphologyEx()
- **Contours**: الكنتورات هي المنحنيات التي تحتوي على جميع النقاط المتصلة بنفس الشدة. هذه أدوات مفيدة للغاية للتعرف على الأشياء. يوفر OpenCV دوال cv2.findContours() لهذه الميزة.

تنزيل الكود المصدري للمشروع

قبل المتابعة، يرجى تنزيل الكود المصدري: **Automatic Number Plate Recognition**

الآن، دعنا نتعمقفي رمز التعرف على لوحة الأرقام. اتبع الخطوات التالية:

الخطوة 1: استيراد المكتبات:

بالنسبة لهذا المشروع، نحتاج إلى مكتبات بايثون numpy وpillow مع openCV وpytesseract

```
import numpy as np
import cv2
from PIL import Image
import pytesseract as tess
```

الخطوة 2: تعريف ثلاث دوال

سنقوم الآن بتعريف ثلاث دوال، للعثور على المعالم غير الضرورية التي قد تحددها openCV ولكن ليس لديها احتمال أن تكون لوحة أرقام.

الدالة الأولى للتحقق من نطاق المنطقة ونسبة العرض إلى الارتفاع:

```
def ratioCheck(area, width, height):
    ratio = float(width) / float(height)
    if ratio < 1:
```

```
        ratio = 1 / ratio
    if (area < 1063.62 or area > 73862.5) or (ratio < 3 or ratio > 6):
        return False
    return True
```

الدالة الثانية للتحقق من متوسط مصفوفة الصورة:

```
def isMaxWhite(plate):
    avg = np.mean(plate)
    if(avg>=115):
        return True
    else:
         return False
```

الدالة الثالثة للتحقق من دوران الكونتور:

```
def ratio_and_rotation(rect):
    (x, y), (width, height), rect_angle = rect

    if(width>height):
        angle = -rect_angle
    else:
        angle = 90 + rect_angle

    if angle>15:
         return False

    if height == 0 or width == 0:
        return False

    area = height*width
    if not ratioCheck(area,width,height):
        return False
    else:
        return True
```

الخطوة 3: كتابة دالة لتنظيف لوحة الأرقام المحددة للمعالجة المسبقة قبل التغذية إلى pytesseract:

```
def clean2_plate(plate):
    gray_img = cv2.cvtColor(plate, cv2.COLOR_BGR2GRAY)

     , thresh = cv2.threshold(gray img, 110, 255, cv2.THRESH BINARY)
    if cv2.waitKey(0) & 0xff == ord('q'):
        pass
    num_contours,hierarchy =
cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL,
cv2.CHAIN APPROX NONE)

    if num_contours:
        contour_area = [cv2.contourArea(c) for c in num_contours]
        max_cntr_index = np.argmax(contour_area)

        max_cnt = num_contours[max_cntr_index]
        max_cntArea = contour_area[max_cntr_index]
        x,y,w,h = cv2.boundingRect(max_cnt)

        if not ratioCheck(max_cntArea,w,h):
            return plate,None

        final_img = thresh[y:y+h, x:x+w]
        return final_img,[x,y,w,h]
```

```
    else:
        return plate, None
```

الخطوة 4: في هذه الخطوة، سوف نأخذ إدخال صورة. سنقوم بإجراء عمليات Gaussian Blur وSobel والعمليات المورفولوجية.

بعد أن نجد الخطوط العريضةفي الصورة ونجري حلقة عبر كل كونتور لتحديد لوحة الأرقام. سنقوم بعد ذلك بتنظيف كونتور الصورة وتغذيتها على pytesseract للتعرف على الرقم والأحرف.

```
img = cv2.imread("testData/sample15.jpg")
print("Number  input image...",)
cv2.imshow("input",img)

if cv2.waitKey(0) & 0xff == ord('q'):
    pass
img2 = cv2.GaussianBlur(img, (3,3), 0)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

img2 = cv2.Sobel(img2,cv2.CV_8U,1,0,ksize=3)
_,img2 = cv2.threshold(img2,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

element = cv2.getStructuringElement(shape=cv2.MORPH_RECT, ksize=(17,
3))
morph_img_threshold = img2.copy()
cv2.morphologyEx(src=img2, op=cv2.MORPH_CLOSE, kernel=element,
dst=morph_img_threshold)
num_contours, hierarchy=
cv2.findContours(morph_img_threshold,mode=cv2.RETR_EXTERNAL,method=cv2
.CHAIN_APPROX_NONE)
cv2.drawContours(img2, num_contours, -1, (0,255,0), 1)


for i,cnt in enumerate(num_contours):

    min_rect = cv2.minAreaRect(cnt)

    if ratio_and_rotation(min_rect):

        x,y,w,h = cv2.boundingRect(cnt)
        plate_img = img[y:y+h,x:x+w]
        print("Number  identified number plate...")
        cv2.imshow("num plate image",plate_img)
        if cv2.waitKey(0) & 0xff == ord('q'):
            pass

        if(isMaxWhite(plate_img)):
            clean_plate, rect = clean2_plate(plate_img)
            if rect:
                fg=0
                x1,y1,w1,h1 = rect
                x,y,w,h = x+x1,y+y1,w1,h1
                # cv2.imwrite("clena.png",clean_plate)
                plate_im = Image.fromarray(clean_plate)
                text = tess.image_to_string(plate_im, lang='eng')
                print("Number  Detected Plate Text : ",text)
```

كود مشروع GUI

أنشئ ملفًا جديدًا gui.py وألصق الكود أدناه:

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
from tkinter import PhotoImage
import numpy as np
import cv2
import pytesseract as tess
def clean2_plate(plate):
    gray_img = cv2.cvtColor(plate, cv2.COLOR_BGR2GRAY)

    _, thresh = cv2.threshold(gray_img, 110, 255, cv2.THRESH_BINARY)
    num contours,hierarchy =
cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)

    if num_contours:
        contour_area = [cv2.contourArea(c) for c in num_contours]
        max_cntr_index = np.argmax(contour_area)

        max_cnt = num_contours[max_cntr_index]
        max_cntArea = contour_area[max_cntr_index]
        x,y,w,h = cv2.boundingRect(max_cnt)

        if not ratioCheck(max cntArea,w,h):
            return plate,None

        final_img = thresh[y:y+h, x:x+w]
        return final_img,[x,y,w,h]

    else:
        return plate,None

def ratioCheck(area, width, height):
    ratio = float(width) / float(height)
    if ratio < 1:
        ratio = 1 / ratio
    if (area < 1063.62 or area > 73862.5) or (ratio < 3 or ratio > 6):
        return False
    return True

def isMaxWhite(plate):
    avg = np.mean(plate)
    if(avg>=115):
        return True
    else:
        return False

def ratio_and_rotation(rect):
    (x, y), (width, height), rect_angle = rect

    if(width>height):
        angle = -rect_angle
    else:
        angle = 90 + rect_angle

    if angle>15:
        return False

    if height == 0 or width == 0:
        return False

    area = height*width
```

```
    if not ratioCheck(area,width,height):
        return False
    else:
        return True

top=tk.Tk()
top.geometry('900x700')
top.title('Number Plate Recognition')
top.iconphoto(True, PhotoImage(file="/home/shivam/Dataflair/Keras
Projects_CIFAR/GUI/logo.png"))
img = ImageTk.PhotoImage(Image.open("logo.png"))
top.configure(background='#CDCDCD')
label=Label(top,background='#CDCDCD', font=('arial',35,'bold'))
# label.grid(row=0,column=1)
sign_image = Label(top,bd=10)
plate_image=Label(top,bd=10)
def classify(file_path):
    res_text=[0]
    res_img=[0]
    img = cv2.imread(file_path)
    img2 = cv2.GaussianBlur(img, (3,3), 0)
    img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

    img2 = cv2.Sobel(img2,cv2.CV 8U,1,0,ksize=3)
    _,img2 =
cv2.threshold(img2,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

    element = cv2.getStructuringElement(shape=cv2.MORPH_RECT,
ksize=(17, 3))
    morph_img_threshold = img2.copy()
    cv2.morphologyEx(src=img2, op=cv2.MORPH_CLOSE, kernel=element,
dst=morph_img_threshold)
    num_contours, hierarchy=
cv2.findContours(morph_img_threshold,mode=cv2.RETR_EXTERNAL,method=cv2
.CHAIN_APPROX_NONE)
    cv2.drawContours(img2, num_contours, -1, (0,255,0), 1)


    for i,cnt in enumerate(num_contours):

        min_rect = cv2.minAreaRect(cnt)

        if ratio_and_rotation(min_rect):

            x,y,w,h = cv2.boundingRect(cnt)
            plate_img = img[y:y+h,x:x+w]
            print("Number  identified number plate...")
            res_img[0]=plate_img
            cv2.imwrite("result.png",plate_img)
            if(isMaxWhite(plate_img)):
                clean_plate, rect = clean2_plate(plate_img)

                if rect:
                    fg=0
                    x1,y1,w1,h1 = rect
                    x,y,w,h = x+x1,y+y1,w1,h1
                    plate_im = Image.fromarray(clean_plate)
                    text = tess.image_to_string(plate_im, lang='eng')
                    res_text[0]=text
                    if text:
                        break
    label.configure(foreground='#011638', text=res_text[0])
```

```
    uploaded=Image.open("result.png")
    im=ImageTk.PhotoImage(uploaded)
    plate_image.configure(image=im)
    plate_image.image=im
    plate_image.pack()
    plate_image.place(x=560,y=320)
def show_classify_button(file_path):
    classify_b=Button(top,text="Classify Image",command=lambda:
classify(file_path),padx=10,pady=5)
    classify_b.configure(background='#364156',
foreground='white',font=('arial',15,'bold'))
    classify_b.place(x=490,y=550)
def upload image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)

uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)
))
        im=ImageTk.PhotoImage(uploaded)
        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass
upload=Button(top,text="Upload an
image",command=upload_image,padx=10,pady=5)
upload.configure(background='#364156',
foreground='white',font=('arial',15,'bold'))
upload.pack()
upload.place(x=210,y=550)

sign_image.pack()
sign_image.place(x=70,y=200)


label.pack()
label.place(x=500,y=220)
heading = Label(top,image=img)
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

الملخص

في هذه المقالة، قمنا بتطوير مشروع التعلم العميق للتعرف على رقم لوحة الترخيص. ناقشنا بعض الميزات المهمة لـ openCV مثل Gaussian blur ومشغلي Sobel والتحولات المورفولوجية. يكتشف التطبيق نص لوحة الأرقام من صورة. لقد حددنا ونظفنا لوحة الأرقام باستخدام OpenCV. لتحديد أرقام وأحرف لوحة الأرقام استخدمنا pytesseract.

4) المراقبة العميقة مع التعلم العميق: مشروع المراقبة الذكية بالفيديو Deep Surveillance with Deep Learning: Intelligent Video Surveillance Project

أمنية المراقبة Surveillance security هو عمل شاق للغاية ويستغرق وقتًا طويلاً.في هذا البرنامج التعليمي، سنبني نظامًا لأتمتة مهمة تحليل المراقبة بالفيديو. سنقوم بتحليل تدفق الفيديوفي الوقت الفعلي وتحديد أي أنشطة غير طبيعية مثل العنف أو السرقة.

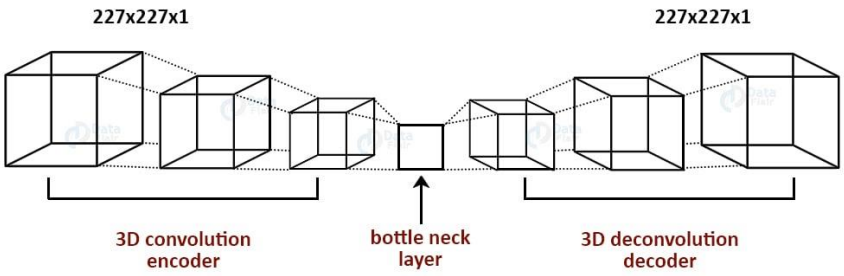
هناك الكثير من الأبحاث الجاريةفي الصناعة حول المراقبة بالفيديو فيما بينها؛ لقد تضخم دور مقاطع فيديو الدوائر التلفزيونية المغلقة CCTV. يتم وضع كاميرات CCTVفي جميع أنحاء الأماكن للمراقبة والأمن.

في العقد الماضي كانت هناك تطوراتفي خوارزميات التعلم العميق للمراقبة العميقة. لقد أظهرت هذه التطورات اتجاهًا أساسيًافي المراقبة العميقة وتعد بتحقيق مكاسب كبيرةفي الكفاءة. التطبيقات النموذجية للمراقبة العميقة هي تحديد السرقة واكتشاف العنف واكتشاف فرص الانفجار.

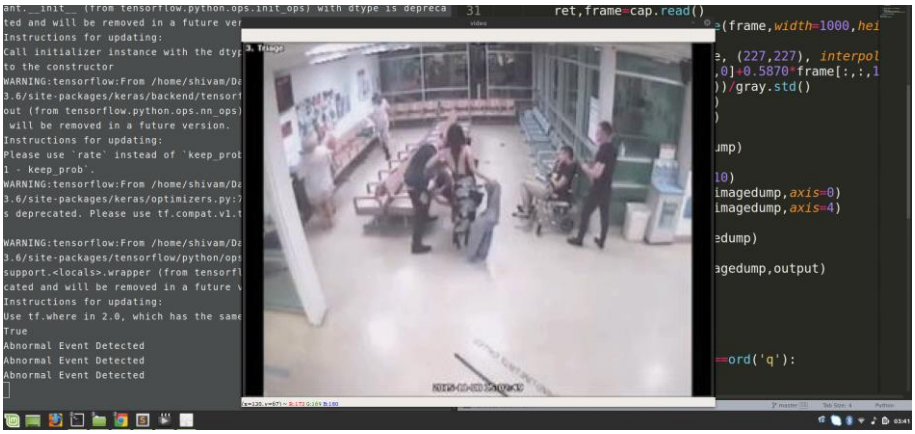
معمارية الشبكات:

لقد رأينا بشكل عام شبكات عصبية عميقة لمهام الرؤية الحاسوبية وتصنيف الصور واكتشاف الكائنات.في هذا المشروع، يتعين علينا توسيع الشبكات العصبية العميقة إلى ثلاثية الأبعاد لتعلم الميزات المكانية والزمانية لتغذية الفيديو.

بالنسبة لمشروع المراقبة بالفيديو هذا، سنقدم المشفر التلقائي الزماني المكاني spatio temporal autoencoder، والذي يعتمد على شبكة التفافية ثلاثية الأبعاد. يستخرج جزء المشفر المعلومات المكانية والزمانية، ثم يعيد مفكك الشفرة بناء الإطارات. يتم تحديد الأحداث غير الطبيعية عن طريق حساب خسارة (خطأ) إعادة الإعمار باستخدام المسافة الإقليدية بين الدُفعة الأصلية والمُعاد بناؤها.



المراقبة الذكية بالفيديو مع التعلم العميق



سنستخدم المشفر التلقائي الزماني المكاني لتحديد الأنشطة غير الطبيعية.

مجموعة البيانات لاكتشاف الأحداث غير الطبيعية في المراقبة بالفيديو:

فيما يلي مجموعات البيانات الشاملة التي تُستخدم لتدريب النماذج على مهام اكتشاف العيوب.

مجموعة بيانات CUHK Avenue:

تحتوي مجموعة البيانات هذه على 16 تدريبًا و21 مقطع فيديو تجريبيًا. يحتوي الفيديو على 30652 لقطةفي المجموع.

تحتوي مقاطع الفيديو التدريبية على فيديو بمواقف عادية. تحتوي مقاطع فيديو الاختبار على مقاطع فيديو بها أحداث قياسية وغير طبيعية.

رابط تنزيل مجموعة البيانات:Avenue Dataset for Abnormal Event Detection

مجموعة بيانات المشاة UCSD:

تحتوي مجموعة البيانات هذه على مقاطع فيديو للمشاة. وهي تشمل مجموعات من الأشخاص يسيرون باتجاه الكاميرا وبعيداً عنها وبالتوازي معها. يشمل الحدث غير الطبيعي:

- الكيانات غير المشاة.
- أنماط حركة المشاة الشاذة.

رابط تنزيل مجموعة البيانات: UCSD Anomaly Detection Dataset

الكود المصدري المشروع

قبل المتابعة، يرجى تنزيل الكود المصدري الذي استخدمناهفي مشروع التعلم العميق هذا: Video Surveillance Project Code

المراقبة بالفيديو - كود كشف الشذوذ الزوجي:

أولاً، قم بتنزيل أي مجموعة من مجموعات البيانات المذكورة أعلاه ووضعهافي دليل باسم "train".

قم بإنشاء ملف بايثون جديد train.py وألصق الكود الموضحفي الخطوات التالية:

1. استيراد المكتبات:

```
from keras.preprocessing.image import img_to_array,load_img
import numpy as np
import glob
import os
import cv2

from keras.layers import Conv3D,ConvLSTM2D,Conv3DTranspose
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint, EarlyStopping
import imutils
```

2. تهيئة متغير مسار الدليل ووصف دالة لمعالجة إطارات الفيديو وتخزينها:

```
store_image=[]
train_path='./train'
fps=5
train_videos=os.listdir('train_path')
train_images_path=train_path+'/frames'
os.makedirs(train_images_path)

def store_inarray(image_path):
    image=load_img(image_path)
    image=img_to_array(image)
    image=cv2.resize(image, (227,227), interpolation = cv2.INTER_AREA)
    gray=0.2989*image[:,:,0]+0.5870*image[:,:,1]+0.1140*image[:,:,2]
    store_image.append(gray)
```

3. استخراج الإطارات من الفيديو واستدعاء دالة الخزن:

```
for video in train_videos:
    os.system( 'ffmpeg -i {}/{} -r 1/{}
{}/frames/%03d.jpg'.format(train_path,video,fps,train_path))
    images=os.listdir(train_images_path)
    for image in images:
        image path=train image path + '/' + image
        store_inarray(image_path)
```

4. تخزين قائمة store_imageفي ملف فارغ "training.npy":

```
store_image=np.array(store_image)
a,b,c=store_image.shape

store_image.resize(b,c,a)
store_image=(store_image-store_image.mean())/(store_image.std())
store_image=np.clip(store_image,0,1)
np.save('training.npy',store_image)
```

5. إنشاء معمارية للمشفر التلقائي المكاني:

```
stae_model=Sequential()
```

```
stae_model.add(Conv3D(filters=128,kernel_size=(11,11,1),strides=(4,4,1
),padding='valid',input_shape=(227,227,10,1),activation='tanh'))
stae_model.add(Conv3D(filters=64,kernel_size=(5,5,1),strides=(2,2,1),p
adding='valid',activation='tanh'))
stae_model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,paddi
ng='same',dropout=0.4,recurrent_dropout=0.3,return_sequences=True))
stae_model.add(ConvLSTM2D(filters=32,kernel_size=(3,3),strides=1,paddi
ng='same',dropout=0.3,return_sequences=True))
stae_model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,retur
n_sequences=True, padding='same',dropout=0.5))
stae_model.add(Conv3DTranspose(filters=128,kernel_size=(5,5,1),strides
=(2,2,1),padding='valid',activation='tanh'))
stae model.add(Conv3DTranspose(filters=1,kernel size=(11,11,1),strides
=(4,4,1),padding='valid',activation='tanh'))

stae_model.compile(optimizer='adam',loss='mean_squared_error',metrics=
['accuracy'])
```

6. تدريب المشفر التلقائي على ملف "training.npy" وحفظ النموذج باسم "save_model.h5":

```
training_data=np.load('training.npy')
frames=training_data.shape[2]
frames=frames-frames%10

training_data=training_data[:,:,:frames]
training_data=training_data.reshape(-1,227,227,10)
training_data=np.expand_dims(training_data,axis=4)
target_data=training_data.copy()

epochs=5
batch_size=1

callback_save = ModelCheckpoint("saved_model.h5",
monitor="mean_squared_error", save_best_only=True)

callback_early_stopping = EarlyStopping(monitor='val_loss',
patience=3)

stae_model.fit(training_data,target_data, batch_size=batch_size,
epochs=epochs, callbacks = [callback_save,callback_early_stopping])
stae_model.save("saved_model.h5")
```

قم بتشغيل هذا السكريبت لتدريب نموذج المشفر التلقائي وحفظه.

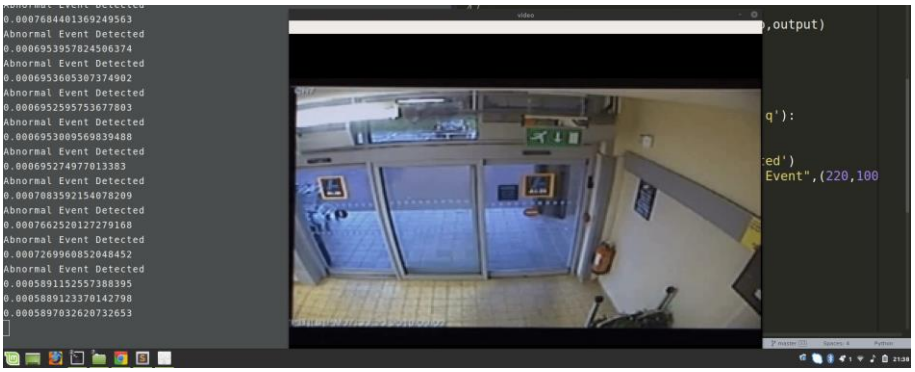
الآن قم بإنشاء ملف python آخر "test.py" ولاحظ نتائج اكتشاف الأحداث غير الطبيعيةفي أي مقطع فيديو مخصص.

الصق الكود أدناهفي "test.py":

```
import cv2
import numpy as np
from keras.models import load_model
import argparse
from PIL import Image
import imutils


def mean_squared_loss(x1,x2):
    difference=x1-x2
```

```
    a,b,c,d,e=difference.shape
    n_samples=a*b*c*d*e
    sq_difference=difference**2
    Sum=sq_difference.sum()
    distance=np.sqrt(Sum)
    mean_distance=distance/n_samples

    return mean_distance


model=load_model("saved_model.h5")

cap = cv2.VideoCapture("__path_to_custom_test_video")
print(cap.isOpened())

while cap.isOpened():
    imagedump=[]
    ret,frame=cap.read()

    for i in range(10):
        ret,frame=cap.read()
        image = imutils.resize(frame,width=700,height=600)

        frame=cv2.resize(frame, (227,227), interpolation =
cv2.INTER_AREA)

gray=0.2989*frame[:,:,0]+0.5870*frame[:,:,1]+0.1140*frame[:,:,2]
        gray=(gray-gray.mean())/gray.std()
        gray=np.clip(gray,0,1)
        imagedump.append(gray)

    imagedump=np.array(imagedump)

    imagedump.resize(227,227,10)
    imagedump=np.expand_dims(imagedump,axis=0)
    imagedump=np.expand_dims(imagedump,axis=4)

    output=model.predict(imagedump)

    loss=mean_squared_loss(imagedump,output)

    if frame.any()==None:
        print("none")

    if cv2.waitKey(10) & 0xFF==ord('q'):
        break
    if loss>0.00068:
        print('Abnormal Event Detected')
        cv2.putText(image,"Abnormal
Event",(100,80),cv2.FONT_HERSHEY_SIMPLEX,2,(0,0,255),4)

    cv2.imshow("video",image)

cap.release()
cv2.destroyAllWindows()
```

الآن، قم بتشغيل هذا السكريبت ولاحظ نتائج المراقبة بالفيديو، وسوف يسلط الضوء على الأحداث غير الطبيعية.



الملخص

في مشروع التعلم العميق هذا، نقوم بتدريب المشفر التلقائي لاكتشاف الأحداث غير الطبيعية. نقوم بتدريب المشفر التلقائي على مقاطع الفيديو العادية. نحدد الأحداث غير الطبيعية بناءً على المسافة الإقليدية لتغذية الفيديو المخصصة والإطارات التي تنبأ بها المشفر التلقائي.

وضعنا قيمة حدية للأحداث غير الطبيعية.في هذا المشروع، تبلغ القيمة 0.0068؛ يمكنك تغيير هذا الحد لتجربة الحصول على نتائج أفضل.

5) التعرف على الارقام العربية باستخدام التعلم العميق Recognise Arabic Digits using Deep Learning

دعونا نبني نموذج التعلم العميق الذي يمكنه قراءة وفهم واكتشاف أكثر من 28 رقمًا باللغة العربية المكتوبة بخط اليد.

كونها واحدة من اللغات الست المعترف بها من قبل الأمم المتحدة، اللغة العربية هي اللغة الرسمية لحوالي 1.8 مليون شخصفي جميع أنحاء العالم. بالحديث عن انتشار اللغة، انتشرت مع نمو الإسلام حول العالم. اكتسبت مفردات من اللغة الفارسية الوسطى والتركية.في العالم الإسلامي خلال القرن الثامن، أصبحت معرفة اللغة العربية إلزامية للجميع.

يمكن للمرء أن يستمرفي الحديث عن تاريخ اللغة العربية وتطورها، إلا أنه لن يخدم الغرض من هذا المقال. اليوم، سوف نتعلم كيفية استخدام معرفتنافي التعلم الآلي والتعلم العميق للتعرف على الأرقام العربية.

استيراد المكتبات اللازمة

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Model
from tensorflow.keras.layers import Dense, Dropout, Conv2D, MaxPooling2D,
Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import Sequential
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
import pandas as pd
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

فك ضغط مجموعة البيانات

```
!wget 'https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-
admin/arabic.zip'
```

تحميل تسميات التدريب والاختبار

الخطوة الأولى والأهم لإعداد نموذجنا هي الوصول إلى قاعدة البيانات. للقيام بذلك، يمكنك اتباع هذا الرابط. نجد أن مجموعة البيانات الخاصة بنا بها 4 ملفات csv:

1. صور التدريب.
2. تسميات التدريب.
3. صور الاختبار.

4. تسميات الاختبار

```
train=pd.read_csv('csvTrainImages 13440x1024.csv')
test=pd.read_csv('csvTestImages 3360x1024.csv')
train_label=pd.read_csv('csvTrainLabel 13440x1.csv')
test_label=pd.read_csv('csvTestLabel 3360x1.csv')
```

عرض أشكال تسميات التدريب والاختبار

```
print("Training Images Shape = ", train.shape)
print("Testing Images Shape = ",test.shape)
print("Training Labels Shape = ",train_label.shape)
print("Testing Labels Shape = ",test_label.shape)
Training Images Shape =  (13439, 1024)
Testing Images Shape =  (3359, 1024)
Training Labels Shape =  (13439, 1)
Testing Labels Shape =  (3359, 1)
```

ترميز التسميات باستخدام Label Binarizer

سنبدأ بتحميل جميع الملفات باستخدام مكتبة pandas ونلاحظ شكل مجموعة البيانات.

قراءة وحدات بكسل الصورة من ملف csv هي خطوتنا التالية متبوعة بتحويلها إلى مصفوفة عددية.

```
train=np.array(train)
test=np.array(test)
train_label=np.array(train_label)
test_label=np.array(test_label)

X = train
y0 = train_label
```

```
binencoder = LabelBinarizer()
y = binencoder.fit_transform(y0)

X_images = X.reshape(-1,32,32)
test_images = test.reshape(-1,32,32)


print(X_images.shape)
print(test_images.shape)
```

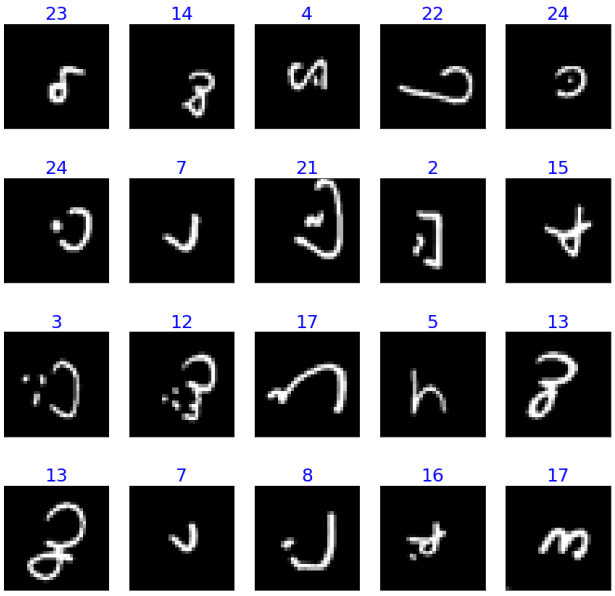
من أجل تحويل تسميات التدريب الخاصة بنا إلى نموذج يمكن الوصول إليه بواسطة نموذجنا، سنستخدم Label Binarizer (). LabelBinarizaer هي فئة أدوات مساعدة للمساعدةفي إنشاء مصفوفة مؤشر التسمية من قائمة التسميات متعددة الفئات. بعد ترميز تسميات التدريب، فإن الخطوة التالية هي تغيير حجم بيانات الصورة من المصفوفة المعقدة.

رسم بعض الصور

بالنسبة لنموذجنا، سنستخدم حجم إدخال 32 × 32. أي أن كل صورة من صورنا سيكون لها ارتفاع وعرض 32 بكسل لكل منها.

```
def visualize_images(df, img_size, number_of_images, name):
    plt.figure(figsize=(8,8))

    n_rows = df.shape[0]
    f = plt.figure(figsize=(15,15)) # defining a figure
    reshaped_df = df.reshape(df.shape[0], img_size, img_size)
    number_of_rows = number_of_images/5 if number_of_images%5 == 0 else
(number_of_images/5) +1
    for i in range(number_of_images):
        f.add_subplot(number_of_rows, 5, i+1, xticks=[], yticks=[])
        #plt.figure(figsize = (7,7))
        plt.title(np.argmax(name[i]), color = 'blue', fontdict = {'size' :
'25'})
        plt.imshow(reshaped_df[i], cmap='gray')
```

```
visualize_images(X_train, 32, 20, y_train)
```

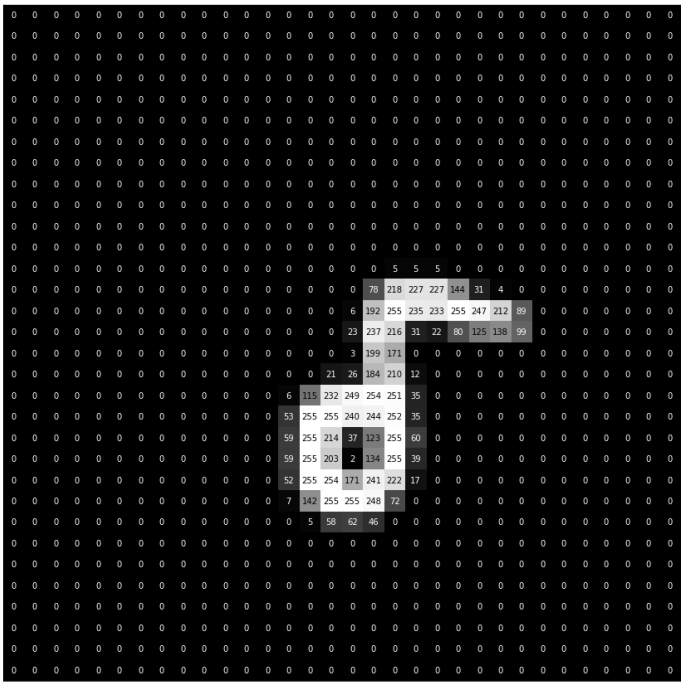


رسم بكسلات صورة واحدة

لتصور صورة بشكل فعال من خلال بيانات البكسل، نستخدم الدالة التالية ونمرر كإدخال أول صورة لبيانات التدريب ونرى توزيع القيم للحرف العربي المكتوب بخط اليد.

```
def visualize_input(img, ax):
    img = img.reshape(32, 32)
    ax.imshow(img, cmap='gray')
    width, height = img.shape
    thresh = img.max()/2.5
    for x in range(width):
        for y in range(height):
            ax.annotate(str(round(img[x][y],2)), xy=(y,x),
                        horizontalalignment='center',
                        verticalalignment='center',
                        color='white' if img[x][y]<thresh else 'black')
```

```
fig = plt.figure(figsize = (15,15))
ax = fig.add_subplot(111, xticks=[], yticks=[])


visualize_input(X_train[0], ax)
```



تحجيم وتشكيل الصور

```
# Scaling and shaping the images
X_train = X_train/255
X_test = X_test/255

X_train = X_train.reshape(-1,32,32,1).astype('float32')
X_test = X_test.reshape(-1,32,32,1).astype('float32')
```

معمارية النموذج

بعد هذه اللمحة القصيرة للصور، فإن الخطوة التالية هي تقسيم صورنا للتدريب واختبار الأجزاء باستخدام دالة train_test_split() من مكتبة ScikitLearn. لغرض نموذجنا، نستخدم قسم التدريب ـ الاختبار بنسبة 80٪ و20٪ على التوالي.

من الممارسات الجيدة أثناء إعداد نماذج التعلم العميق استخدام ميزة callbacks في Keras. باستخدام EarlyStopping() ومراقبة خطأ التحقق من الصحة، يتوقف نموذجنا عن التدريب إذا لم يتم تقليل خطأ التحقق لمدة 5 فترات. بهذه الطريقة، نضمن حداً أدنى لقيمة الخطأ ونمنع أيضًا فرط التعلم overfitting إلى حد ما.

الخطوة التالية هي تحديد معمارية نموذجنا المتسلسل. نظرًا لأننا نتعامل مع بيانات الصورة بحجم 32 × 32، فقد أنشأنا الطبقة الأولى كطبقة التفاف ثنائية الأبعاد متبوعة بطبقة تجميع بحد أقصى لها حجم فلتر 2 × 2. يتبع هذه "الكتلة" كتلتان متشابهتان، وبعد ذلك نضيف طبقة Dropout. الدالة الرئيسية لإضافة طبقات Dropout هي منع فرط التعلم. إنه يسقط أو يتجاهل عدة مخرجات للطبقة بشكل عشوائي ويقللها إلى 0. يشجع الشبكة العصبية على تعلم شبكة متفرقة. بعد ذلك، قمنا بتسوية متغيرات الإدخال متبوعة بعدة طبقات كثيفة ولدينا طبقة إخراج كثيفة مع تعيين دالة التنشيط sigmoid.

```
# Defiining Early Stopping function to monitor Validation Loss
es = EarlyStopping(monitor='val_loss', patience=5)

# Defining Model Architecture
model = Sequential()
```

```
model.add(Conv2D(32,(3,3),input_shape=(32,32,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.2))
model.add(Flatten())

model.add(Dense(36,activation='relu'))
model.add(Dense(36, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(28, activation='sigmoid'))
model.summary()
```

عندما نطبع ملخص النموذج، نلاحظ أن نموذجنا يحتوي على حوالي 25000 معلمة قابلة للتدريب.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 32)          9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 32)          0
_________________________________________________________________
dropout (Dropout)            (None, 2, 2, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 36)                4644
_________________________________________________________________
dense_1 (Dense)              (None, 36)                1332
_________________________________________________________________
dropout_1 (Dropout)          (None, 36)                0
_________________________________________________________________
dense_2 (Dense)              (None, 28)                1036
=================================================================
Total params: 25,828
Trainable params: 25,828
Non-trainable params: 0
_________________________________________________________________
```

تجميع النموذج

```
# Compile the model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accu
racy'])
```

بتجميع نموذجنا باستخدام مُحسِّن آدم والتقاطع الفئوي كدالة خطأ، نبدأ التدريب عن طريق تعيين حجم دفعة من 50 وتهيئة التدريب لـ 100 حقبة.

بعد انتهاء تدريبنا، نلاحظ أننا حققنا دقة تحقق تبلغ 90٪ وقيمة خطأ منخفضة تصل إلى 0.2.

تدريب النموذج

```
# Run the model for a batch size of 50 for 100 epochs
history = model.fit(X_train,
                    y_train,
                    validation_data = (X_test, y_test),
                    batch_size = 50,
                    epochs = 100,
                    callbacks = [es]
                   )
```

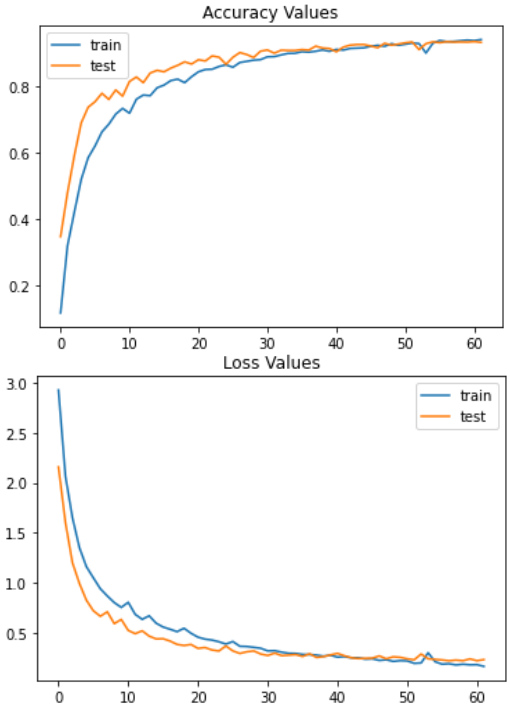
```
Epoch 59/100
216/216 [==============================] - 10s 48ms/step - loss: 0.1887 - accuracy: 0.9379 - val_loss: 0.2235 - val_accuracy: 0.9342
Epoch 60/100
216/216 [==============================] - 10s 48ms/step - loss: 0.1826 - accuracy: 0.9397 - val_loss: 0.2414 - val_accuracy: 0.9338
Epoch 61/100
216/216 [==============================] - 10s 48ms/step - loss: 0.1841 - accuracy: 0.9382 - val_loss: 0.2235 - val_accuracy: 0.9353
Epoch 62/100
216/216 [==============================] - 10s 48ms/step - loss: 0.1671 - accuracy: 0.9413 - val_loss: 0.2344 - val_accuracy: 0.9330
```

رسم النتائج

من خلال التخطيط لدقة نموذجنا وخطأه، نلاحظ أن تدريبنا انتهى قبل أن يتمكن من إكمال دورة من 100 حقبة، مما يقلل الدقة ويضمن الحد الأدنى من قيمة الخطأ.

```
# Function to plot "accuracy vs epoch" graphs and "Loss vs epoch" graphs for
training and validation data
def plot_metrics(model_name, metric = 'accuracy'):
    if metric == 'loss':
        plt.title("Loss Values")
        plt.plot(model_name.history['loss'], label = 'train')
        plt.plot(model_name.history['val_loss'], label = 'test')
        plt.legend()
        plt.show()
    else:
        plt.title("Accuracy Values")
        plt.plot(model_name.history['accuracy'], label='train')
        plt.plot(model_name.history['val_accuracy'], label='test')
        plt.legend()
        plt.show()
```

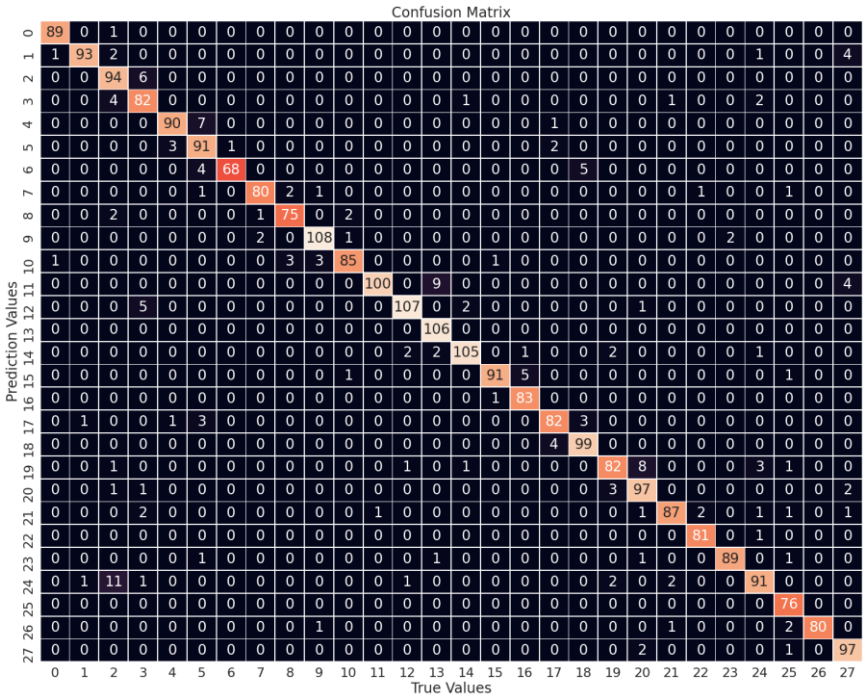
```
plot_metrics(history, 'accuracy')
plot_metrics(history, 'loss')
```



تقييم أداء النموذج

أخيرًا، للتحقق من أداء نموذجنا، قمنا برسم مصفوفة الارتباك على بيانات الاختبار والتحقق من أن نموذجنا دقيق بالفعل.

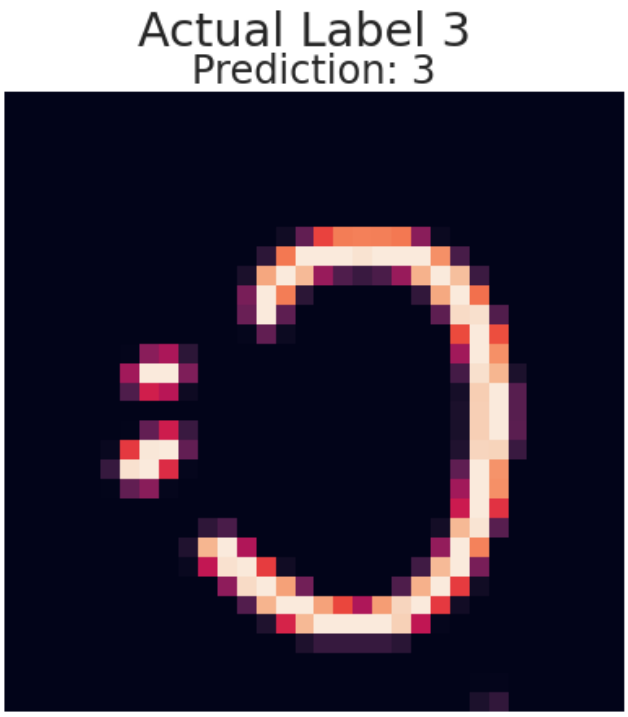
```
#Plotting a confusion matrix for checking the performance of our model
Y_pred = np.argmax(model.predict(X_test), axis = 1)
cnf = confusion_matrix(y_test.argmax(axis = 1), Y_pred)


df_cnf = pd.DataFrame(cnf, range(28), range(28))
sns.set(font_scale = 2)
plt.figure(figsize = (25, 20))
sns.heatmap(df_cnf, annot = True, linewidths = 0.8, fmt = '0.3g', cbar =
False)
plt.title("Confusion Matrix")
plt.xlabel("True Values")
plt.ylabel("Prediction Values")
plt.show()
```



عمل تنبؤات على تسمية واحدة

```
pred = np.argmax(model.predict(np.expand_dims(X_test[7], axis = 0)))
preds = "Prediction: " + str(pred)
plt.figure(figsize = (7,7))
actual_label = np.argmax(y_test[7]) + 1
plt.imshow(X_test[7])

plt.grid(False)
plt.axis("off")
plt.title(preds)
plt.suptitle("Actual Label " + str(actual_label))
```

```
Text(0.5, 0.98, 'Actual Label 3')
```



حفظ النموذج

```
# Saving our trained model
from tensorflow.keras.models import save_model
if os.path.isfile('best_model.h5') is False:
    model.save('best_model.h5')
```

الملخص

نرى أن نموذجنا ناجح ويمكنه التنبؤ بالحروف العربية المكتوبة بخط اليد بدقة كبيرة. لقد استخدمنا التعلم الآلي ومعرفة التعلم العميق بشكل جيد عندما أعددنا هذا النموذج. من أجل الوصول إلى الكود الكامل، يرجى اتباع هذا الرابط.

6) الكشف عن كوفيد -19 بالأشعة السينية للصدر باستخدام CNN
Detecting Covid-19 with Chest X-ray

تعد جائحة COVID-19 أحد أكبر التحديات التي تواجه نظام الرعاية الصحيةفي الوقت الحالي. وهو مرض تنفسي يصيب رئتينا ويمكن أن يتسببفي تلف دائم للرئتين أدى إلى ظهور أعراض مثل صعوبة التنفس وفي بعض الحالات الالتهاب الرئوي وفشل الجهاز التنفسي.في هذه المقالة، سوف نستخدم بيانات الأشعة السينية للرئتين الطبيعية والمرضى المصابين بفيروس COVID وندرب نموذجًا للتمييز بينهما.

مجموعة البيانات والنماذج المستخدمة:

مجموعة البيانات المستخدمةفي هذا المنشور هي الفائزة بجائزة مجتمع Kaggle. تم جمع مجموعة البيانات من قبل باحثين من قطر وبنغلاديش. تحتوي مجموعة البيانات هذه على 3 أنواع من الصور:

- COVID-19 إيجابي (219 صورة).
- الالتهاب الرئوي الفيروسي (1341 صورة).
- أشعة سينية عادية (1345 صورة).

لذلك، يتعين علينا التصنيف بين هذه الفئات الثلاث المختلفة وسنستخدم طبقة softmax للتصنيف.

هذه الصور لها حجم (1024، 1024) وثلاث قنوات ملونة. قام مؤلفو مجموعة البيانات أيضًا بتدريب نموذج ResNet -34 وحققوا دقة بلغت 98.5٪.

التنفيذ

- في هذه المقالة، سوف نستخدم نموذج Xception بمساعدة Keras API. حصل هذا النموذج على دقة ImageNet top-1 بنسبة 79٪ ودقة أعلى 5 بنسبة 95٪.
- أولاً ، نحتاج إلى استيراد الوحدات اللازمة.

```
import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras import Sequential
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import InceptionResNetV2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.layers import Dense,Flatten,  Input, Dropout
```

- الآن، سوف نستخدم Kaggle API لتنزيل مجموعة البيانات على النظام. أولاً، سنطلب مفتاح API ، للقيام بذلك، انتقل فقط إلى قسم الملف الشخصي على Kaggle وقم بتنزيل ملف JSON يحتوي على تفاصيل API الخاصة بنا ، وبعد ذلك فقط قم بتحميل هذا إلى colab أو تحديد موقعه في بيئة jupyter المحلية.

```
# code
"""
Kaggle API setup
Credits: https://www.kaggle.com/general/74235
"""
# Install Kaggle module
!pip install kaggle


# Upload API details json file to colab
from google.colab import files
files.upload()
# create a Kaggle directory and move json files to there
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
# change permissions of kaggle json file
! chmod 600 ~/.kaggle/kaggle.json
# Now we download our dataset with following command format :
"""
! kaggle datasets download -d user/dataset
 or
! kaggle competitions download -c 'name-of-competition'
"""
! kaggle datasets download -d tawsifurrahman/covid19-radiography-
database
```

- الآن، نقوم بفك ضغط مجموعة البيانات في المجلد المطلوب.

```
! unzip covid19-radiography-database.zip -d /content/data
```

- الآن قمنا بمعالجة مجموعة البيانات مسبقًا، قمنا بتقليل حجم الصورة من (1024 ، 1024) إلى (299،299) [الحد الأقصى للحجم المقبول بواسطة نموذج Xception] ، وقمنا بتقسيمها إلى حجم دفعة من 16.

```
# Load Xception model
base = Xception(weights="imagenet", input_shape
=(299,299,3),include_top= False)
# set base model trainable to false
for layers in base.layers:
    layers.trainable=False

base.summary()
```

```
Downloading data from

https://storage.googleapis.com/tensorflow/keras-applications/xception/

xception_weights_tf_dim_ordering_tf_kernels_notop.h5

83689472/83683744 [==============================] - 1s 0us/step

Model: "xception"

__________________________________________________________________________
_________________________
Layer (type)                    Output Shape         Param #     Connected
to

==========================================================================
=======================
input_1 (InputLayer)            [(None, 299, 299, 3) 0

__________________________________________________________________________
_________________________
block1_conv1 (Conv2D)           (None, 149, 149, 32) 864
input_1[0][0]

__________________________________________________________________________
_________________________
block1_conv1_bn (BatchNormaliza (None, 149, 149, 32) 128
block1_conv1[0][0]

__________________________________________________________________________
_________________________
block1_conv1_act (Activation)   (None, 149, 149, 32) 0
block1_conv1_bn[0][0]

__________________________________________________________________________
_________________________
block1_conv2 (Conv2D)           (None, 147, 147, 64) 18432
block1_conv1_act[0][0]

__________________________________________________________________________
_________________________
```

```
block1_conv2_bn (BatchNormaliza (None, 147, 147, 64) 256
block1_conv2[0][0]

__________________________________________________________________________________
__________________________

block1_conv2_act (Activation)   (None, 147, 147, 64) 0
block1_conv2_bn[0][0]

__________________________________________________________________________________
__________________________

block2_sepconv1 (SeparableConv2 (None, 147, 147, 128 8768
block1_conv2_act[0][0]

__________________________________________________________________________________
__________________________

block2_sepconv1_bn (BatchNormal (None, 147, 147, 128 512
block2_sepconv1[0][0]

__________________________________________________________________________________
__________________________

block2_sepconv2_act (Activation (None, 147, 147, 128 0
block2_sepconv1_bn[0][0]

__________________________________________________________________________________
__________________________

block2_sepconv2 (SeparableConv2 (None, 147, 147, 128 17536
block2_sepconv2_act[0][0]

__________________________________________________________________________________
__________________________

block2_sepconv2_bn (BatchNormal (None, 147, 147, 128 512
block2_sepconv2[0][0]

__________________________________________________________________________________
__________________________

conv2d (Conv2D)                 (None, 74, 74, 128)  8192
block1_conv2_act[0][0]

__________________________________________________________________________________
__________________________

block2_pool (MaxPooling2D)      (None, 74, 74, 128)  0
block2_sepconv2_bn[0][0]

__________________________________________________________________________________
__________________________

batch_normalization (BatchNorma (None, 74, 74, 128)  512
conv2d[0][0]

__________________________________________________________________________________
__________________________

add (Add)                       (None, 74, 74, 128)  0
block2_pool[0][0]


batch_normalization[0][0]
```

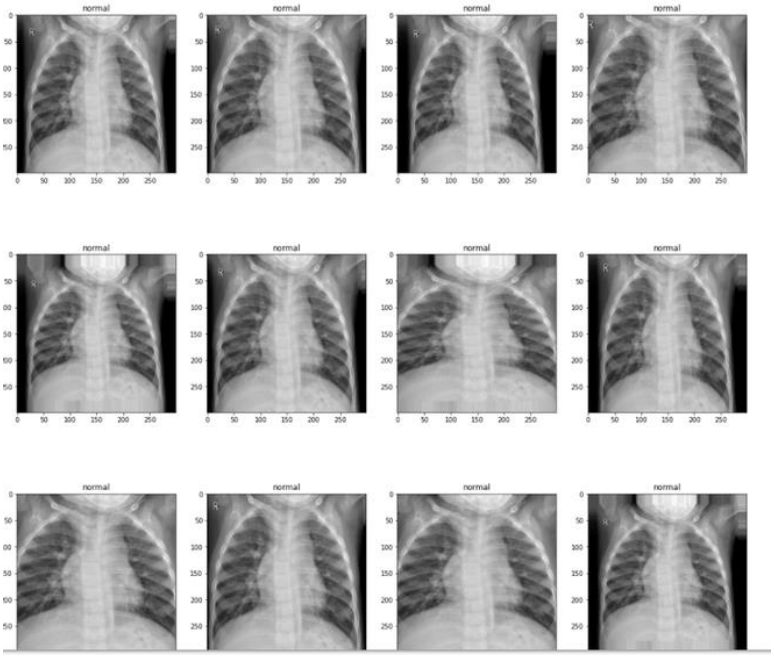
```
_____
_____
block3_sepconv1_act (Activation (None, 74, 74, 128)  0           add[0][0]

_____
_____
block3_sepconv1 (SeparableConv2 (None, 74, 74, 256)  33920
block3_sepconv1_act[0][0]

_____
_____
block3_sepconv1_bn (BatchNormal (None, 74, 74, 256)  1024
block3_sepconv1[0][0]

_____
_____
block3_sepconv2_act (Activation (None, 74, 74, 256)  0
block3_sepconv1_bn[0][0]

_____
_____
block3_sepconv2 (SeparableConv2 (None, 74, 74, 256)  67840
block3_sepconv2_act[0][0]

_____
_____
block3_sepconv2_bn (BatchNormal (None, 74, 74, 256)  1024
block3_sepconv2[0][0]

_____
_____
conv2d_1 (Conv2D)               (None, 37, 37, 256)  32768       add[0][0]

_____
_____
block3_pool (MaxPooling2D)      (None, 37, 37, 256)  0
block3_sepconv2_bn[0][0]

_____
_____
batch_normalization_1 (BatchNor (None, 37, 37, 256)  1024
conv2d_1[0][0]

_____
_____
.............
```

(Trimmed model Summary)

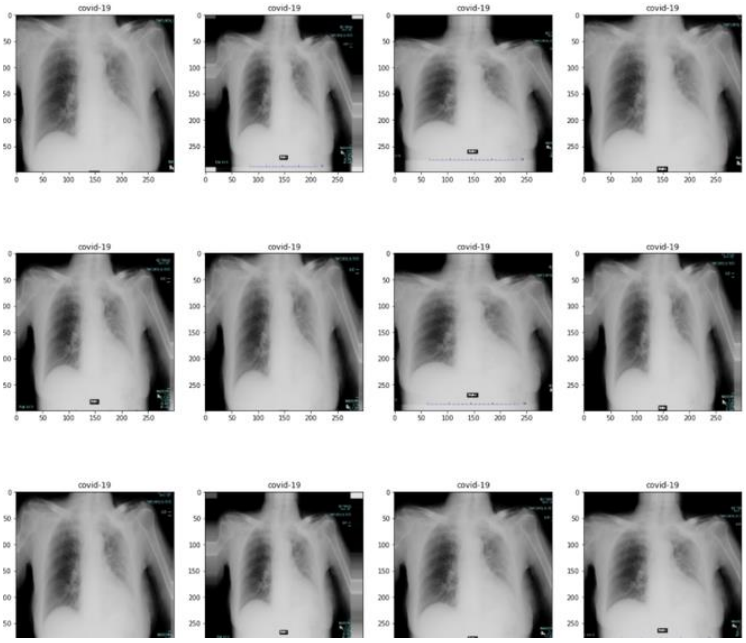
```
==================================================================================
======================
Total params: 20,861,480
Trainable params: 0
Non-trainable params: 20,861,480
_____
```

- الآن، نطبق بعض زيادة البيانات على مجموعة البيانات ونجهزها للتدريب. بعد ذلك، نرسم بعض الصور التدريبية. سنقسم مجموعة البيانات بحيث يكون لدينا 75٪ بيانات للتدريب و 25٪ للاختبار / التحقق من الصحة.

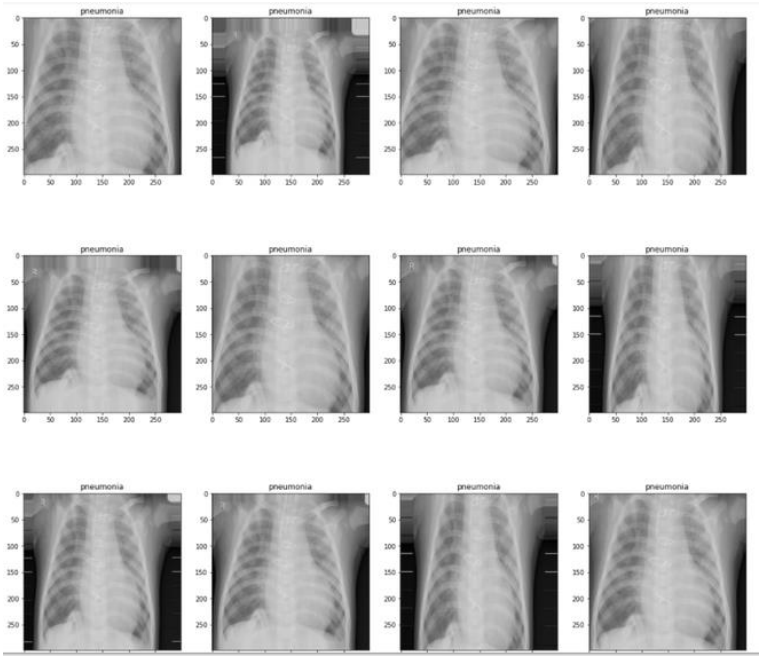
```
# Define augmentation
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom range=0.2,
        validation_split=0.25,
        horizontal_flip =True
        )

# apply augmentations on dataset
train =train_datagen.flow_from_directory(
    "data/",
    target_size=(299, 299),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training')
val =train_datagen.flow_from_directory(
    "data/",
    target_size=(299, 299),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
class_names=['covid-19','normal','pneumonia']

# code to plot images
def plotImages(images_arr, labels):
    fig, axes = plt.subplots(12, 4, figsize=(20,80))
    axes = axes.flatten()
    label=0
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.set_title(class_names[np.argmax(labels[label])])
        label=label+1
    plt.show()

# append a batch of images from each category (COVID-19, Normal,
Viral_Pneumonia)
images = [train[34][0][0] for i in range(16)]
images = images + [train[5][0][0] for i in range(16)]
images = images + [train[0][0][0] for i in range(16)]

# append the batch of labels
labels=[]
labels = [train[34][1][0] for i in range(16)]
labels=  labels + [train[5][1][0] for i in range(16)]
labels=  labels + [train[0][1][0] for i in range(16)]

# plot images with labels
plotImages(images,labels)
```



الأشعة السينية للرئتين الطبيعية (Normal Lungs X-ray)



كوفيد ـ19 (+) أشعة سينية للرئتين (Covid -19 (+) Lungs X-ray)



الأشعة السينية للرئة ذات الالتهاب الرئوي الفيروسي (Viral Pneumonia Lungs X-ray)

- الآن ، نحدد نموذجنا، أولاً، سنقوم باستيراد نموذجنا الأساسي ، أي Xception (نستخدم أوزان imagenet مسبقة التدريب) في نموذجنا المتسلسل، ونقوم بتسوية الطبقة العليا وتطبيق طبقة كثيفة dense layer (طبقة متصلة بالكامل fully connected layer) وطبقة تصنيف softmax لتصنيفها بين 3 فئات. لمنع النموذج من الضبط الزائد (فرط التعلم) overfitting ، سنضيف أيضًا بعض طبقات dropout .

```
# Define our complete models
model = Sequential()
model.add(Input(shape =(299,299,3)))
model.add(base)
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(16))
model.add(Dense(3,activation='softmax'))
model.summary()
```

```
Model: "sequential"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================
```

```
xception (Functional)        (None, 10, 10, 2048)      20861480
_________________________________________________________________
dropout (Dropout)            (None, 10, 10, 2048)      0
_________________________________________________________________
flatten (Flatten)            (None, 204800)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 204800)            0
_________________________________________________________________
dense (Dense)                (None, 16)                3276816
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 51
=================================================================
Total params: 24,138,347
Trainable params: 3,276,867
Non-trainable params: 20,861,480
_________________________________________________________________
```

- سنقوم الآن بتجميع النموذج وتدريبه، نستخدم Adam Optimizer بمعدل تعلم 0.001. سنقوم بتدريب النموذج لمدة 30 حقبة.

```
# import adam optimizer
from tensorflow.keras.optimizers import Adam
# compile model(define metrics and loss)
model.compile(
    optimizer=Adam(learning_rate=1e-3),
    loss="categorical_crossentropy",
    metrics=["accuracy"],
)
# train model for 30 epoch
model.fit_generator(train,epochs=30,validation_data=val)

# save model
model.save('epoch_30.h5')
```

```
Epoch 1/30
137/137 [==============================] - 121s 886ms/step -
loss: 5.7757 - accuracy: 0.8528 - val_loss: 3.4022 - val_accuracy: 0.8966
Epoch 2/30
137/137 [==============================] - 119s 867ms/step -
loss: 3.3137 - accuracy: 0.9028 - val_loss: 2.0748 - val_accuracy: 0.9228
Epoch 3/30
137/137 [==============================] - 119s 866ms/step -
```

```
loss: 2.2811 - accuracy: 0.9161 - val_loss: 2.2661 - val_accuracy: 0.9186
Epoch 4/30
137/137 [==============================] - 119s 867ms/step -
loss: 1.6122 - accuracy: 0.9339 - val_loss: 3.8654 - val_accuracy: 0.8648
Epoch 5/30
137/137 [==============================] - 120s 877ms/step -
loss: 1.0704 - accuracy: 0.9440 - val_loss: 1.6559 - val_accuracy: 0.9214
Epoch 6/30
137/137 [==============================] - 120s 875ms/step -
loss: 0.7675 - accuracy: 0.9509 - val_loss: 1.3920 - val_accuracy: 0.9255
Epoch 7/30
137/137 [==============================] - 120s 872ms/step -
loss: 0.5744 - accuracy: 0.9509 - val_loss: 1.2669 - val_accuracy: 0.9021
Epoch 8/30
137/137 [==============================] - 119s 872ms/step -
loss: 0.4065 - accuracy: 0.9528 - val_loss: 1.1800 - val_accuracy: 0.9145
Epoch 9/30
137/137 [==============================] - 118s 864ms/step -
loss: 0.2160 - accuracy: 0.9638 - val_loss: 0.7624 - val_accuracy: 0.9379
Epoch 10/30
137/137 [==============================] - 119s 865ms/step -
loss: 0.2552 - accuracy: 0.9606 - val_loss: 0.4897 - val_accuracy: 0.9421
Epoch 11/30
137/137 [==============================] - 118s 864ms/step -
loss: 0.2015 - accuracy: 0.9651 - val_loss: 0.4510 - val_accuracy: 0.9476
Epoch 12/30
137/137 [==============================] - 121s 880ms/step -
loss: 0.1473 - accuracy: 0.9725 - val_loss: 0.3458 - val_accuracy: 0.9352
Epoch 13/30
137/137 [==============================] - 121s 880ms/step -
loss: 0.1534 - accuracy: 0.9656 - val_loss: 0.5945 - val_accuracy: 0.9297
Epoch 14/30
137/137 [==============================] - 120s 876ms/step -
loss: 0.1315 - accuracy: 0.9734 - val_loss: 0.4655 - val_accuracy: 0.9407
```

```
Epoch 15/30
137/137 [==============================] - 121s 882ms/step -
loss: 0.1127 - accuracy: 0.9661 - val_loss: 0.3728 - val_accuracy: 0.9186
Epoch 16/30
137/137 [==============================] - 121s 882ms/step -
loss: 0.1198 - accuracy: 0.9716 - val_loss: 0.4312 - val_accuracy: 0.9476
Epoch 17/30
137/137 [==============================] - 120s 875ms/step -
loss: 0.1046 - accuracy: 0.9771 - val_loss: 0.4035 - val_accuracy: 0.9393
Epoch 18/30
137/137 [==============================] - 119s 870ms/step -
loss: 0.0872 - accuracy: 0.9761 - val_loss: 0.8248 - val_accuracy: 0.9145
Epoch 19/30
137/137 [==============================] - 120s 874ms/step -
loss: 0.1116 - accuracy: 0.9752 - val_loss: 0.3309 - val_accuracy: 0.9393
Epoch 20/30
137/137 [==============================] - 120s 877ms/step -
loss: 0.1261 - accuracy: 0.9729 - val_loss: 0.5384 - val_accuracy: 0.8924
Epoch 21/30
137/137 [==============================] - 119s 869ms/step -
loss: 0.0840 - accuracy: 0.9748 - val_loss: 0.5690 - val_accuracy: 0.9366
Epoch 22/30
137/137 [==============================] - 119s 868ms/step -
loss: 0.0942 - accuracy: 0.9761 - val_loss: 0.3517 - val_accuracy: 0.9448
Epoch 23/30
137/137 [==============================] - 120s 876ms/step -
loss: 0.1207 - accuracy: 0.9656 - val_loss: 0.2871 - val_accuracy: 0.9434
Epoch 24/30
137/137 [==============================] - 118s 864ms/step -
loss: 0.0959 - accuracy: 0.9729 - val_loss: 0.4589 - val_accuracy: 0.9366
Epoch 25/30
137/137 [==============================] - 119s 867ms/step -
loss: 0.0945 - accuracy: 0.9748 - val_loss: 0.3964 - val_accuracy: 0.9490
Epoch 26/30
```

```
137/137 [==============================] - 119s 871ms/step -
loss: 0.1039 - accuracy: 0.9761 - val_loss: 0.3048 - val_accuracy: 0.9393
Epoch 27/30
137/137 [==============================] - 119s 866ms/step -
loss: 0.0905 - accuracy: 0.9739 - val_loss: 0.3308 - val_accuracy: 0.9407
Epoch 28/30
137/137 [==============================] - 120s 873ms/step -
loss: 0.0757 - accuracy: 0.9766 - val_loss: 0.1871 - val_accuracy: 0.9517
Epoch 29/30
137/137 [==============================] - 119s 871ms/step -
loss: 0.1012 - accuracy: 0.9688 - val_loss: 0.7361 - val_accuracy: 0.9297
Epoch 30/30
137/137 [==============================] - 120s 874ms/step -
loss: 0.0713 - accuracy: 0.9780 - val_loss: 0.3497 - val_accuracy: 0.9434
```

الملخص

لقد حصلنا على دقة 97.8٪في مجموعة التدريب و94.3٪في مجموعة التحقق من الصحةفي 30 حقبة فقط على نموذج Xception، وهي قريبة من دقة 98.3٪ كما أفاد مؤلفو الورقة.

7) الكشف عن سرطان الجلد باستخدام التعلم العميق Detecting Skin Cancer using Deep Learning

سريعًا، فكرفي خمسة أشخاص تحبهم. أمك، أبيك، أختك أو أخيك، ربما أفضل صديق وجدتك؟

إحصائيًا، سيصاب واحد من هؤلاء الأشخاص الخمسة بسرطان الجلدفي مرحلة ما من حياتهم. مع الحفاظ على سمعة السرطان المميتة، تنتشر معظم سرطانات الجلد إلى أجزاء أخرى من الجسم و / أو تكون قاتلة ما لم يتم اكتشافها وعلاجها مبكرًا. وبسبب عدم كفاءة أنظمة الرعاية الصحية لدينا اليوم، سيبدأ التشخيص والعلاج على مراحلفي وقت متأخر كثيرًا عما يمكن (ويجب) أن يبدأ.

ومع ذلك، لا تزال التشخيصات عملية بصرية، تعتمد على إجراء طويل الأمد للفحوصات السريرية، يليه تحليل تنظير الجلد، ثم خزعة وأخيراً فحص الأنسجة المرضية. تستغرق هذه العملية شهورًا بسهولة وتحتاج إلى العديد من المهنيين الطبيين ولا تزال دقيقة بنسبة 77٪ فقط.

إن الكم الهائل من الوقت والتقنيات التي يستغرقها تشخيص المريض (ناهيك عن بدء العلاج) والفرص العديدة للخطأ البشري، تترك الآلاف من القتلى سنويًا.

ولكن مع تطور الذكاء الاصطناعي وقدرات التعلم الآلي، هناك إمكانية ساطعة لتوفير الوقت وتخفيف الأخطاء ــ مما يؤدي إلى إنقاذ ملايين الأرواح على المدى الطويل.

على وجه الخصوص، يمكن للشبكات العصبية التلافيفية (CNNs) أتمتة معظم عملية التشخيص بدقة مساوية أو أكثر من الطرق الحالية. لتجديد المعلومات حول الشبكات العصبية.

لكي أرى بنفسي، قمت بتكرار شبكة CNN باستخدام 10000 صورة تدريبية لمقارنة النتائج بالخبراء البشريين وتحليل النتائج لمعرفة مدى تباينها. تتناول هذه المقالة العملية وتشرح كيف يمكن تحسين شبكة CNN الأساسية بسهولة تامة ومواءمة القدرات البشرية أو تجاوزها.

في 14 خطوة بسيطة نسبيًا، سأوضح كيف صممت وضبطت هذا النموذج، بالإضافة إلى النتائج النهائية وكيفية مقارنتها.

تتضمن مجموعة البيانات التي استخدمتها 7 فئات رئيسية من سرطانات الجلد:

- Melanocytic nevi, Melanoma,
- Benign keratosis-like lesions,
- Basal cell carcinoma,
- Actinic keratoses,

- Vascular lesions, and
- Dermatofibroma

يستخدم هذا النموذج مجموعة بيانات HAM10000 وتتم معالجته على وحدة معالجة الرسومات GPU.

الخطوة 1: استيراد المكتبات الأساسية

In [1]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
from glob import glob
import seaborn as sns
from PIL import Image
np.random.seed(123)
from sklearn.preprocessing import label_binarize
from sklearn.metrics import confusion_matrix
import itertools

import keras
from keras.utils.np_utils import to_categorical # used for converting labels to one-hot-e
ncoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras import backend as K
import itertools
from keras.layers.normalization import BatchNormalization
from keras.utils.np_utils import to_categorical # convert to one-hot-encoding

from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from sklearn.model_selection import train_test_split
```

تشمل هذه المكتبات Matplotlib و Numpy و Pandas و Sklearn و Keras.

```
#1. Function to plot model's validation loss and validation accuracy
def plot_model_history(model_history):
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    # summarize history for accuracy
    axs[0].plot(range(1,len(model_history.history['acc'])+1),model_history.history['acc'
])
    axs[0].plot(range(1,len(model_history.history['val_acc'])+1),model_history.history[
'val_acc'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')
    axs[0].set_xticks(np.arange(1,len(model_history.history['acc'])+1),len(model_history
.history['acc'])/10)
    axs[0].legend(['train', 'val'], loc='best')
```

```
    # summarize history for loss
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['los
s'])
    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history[
'val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')
    axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_histor
y.history['loss'])/10)
    axs[1].legend(['train', 'val'], loc='best')
    plt.show()
```

هناك أيضًا دالة لرسم خطأ / دقة التحقق من الصحة. (خطأ التحقق من الصحة هو الخطأ بعد تشغيل مجموعة التحقق من البيانات من خلال الشبكة المدربة ـ مدى دقة النموذج بشكل أساسي).

الخطوة الثانية: عمل قاموس للصور والتسميات

```
base_skin_dir = os.path.join('..', 'input')

# Merging images from both folders HAM10000_images_part1.zip and HAM10000_images_part2.zip
 into one dictionary

imageid_path_dict = {os.path.splitext(os.path.basename(x))[0]: x
                     for x in glob(os.path.join(base_skin_dir, '*', '*.jpg'))}

# This dictionary is useful for displaying more human-friendly labels later on

lesion_type_dict = {
    'nv': 'Melanocytic nevi',
    'mel': 'Melanoma',
    'bkl': 'Benign keratosis-like lesions ',
    'bcc': 'Basal cell carcinoma',
    'akiec': 'Actinic keratoses',
    'vasc': 'Vascular lesions',
    'df': 'Dermatofibroma'
}
```

للحصول على مصدر واحد للبيانات، يتم دمج مجلدي مجموعة البيانات معًا ويتم إنشاء قاموس مناسب من التسميات البسيطة للأنواع المختلفة.

الخطوة الثالثة: قراءة البيانات ومعالجتها

```
skin_df = pd.read_csv(os.path.join(base_skin_dir, 'HAM10000_metadata.csv'))

# Creating New Columns for better readability

skin_df['path'] = skin_df['image_id'].map(imageid_path_dict.get)
skin_df['cell_type'] = skin_df['dx'].map(lesion_type_dict.get)
skin_df['cell_type_idx'] = pd.Categorical(skin_df['cell_type']).codes
```

يتم إنشاء مسار للانضمام إلى مجلد الصورة (المجلد الأساسي) بمجلد واضح وتتم إضافة أعمدة جديدة لتنظيم البيانات بشكل أفضل.

```
# Now lets see the sample of tile_df to look on newly made columns
skin_df.head()
```

	lesion_id	image_id	dx	dx_type	age	sex	localization	path
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	../input/ham10000_images_part_1/ISIC_0027419.jpg
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	../input/ham10000_images_part_1/ISIC_0025030.jpg
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	../input/ham10000_images_part_1/ISIC_0026769.jpg
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	../input/ham10000_images_part_1/ISIC_0025661.jpg
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	../input/ham10000_images_part_2/ISIC_0031633.jpg

الخطوة 4: تنظيف البيانات

```
skin_df.isnull().sum()
```

```
lesion_id         0
image_id          0
dx                0
dx_type           0
age              57
sex               0
localization      0
path              0
cell_type         0
cell_type_idx     0
dtype: int64
```

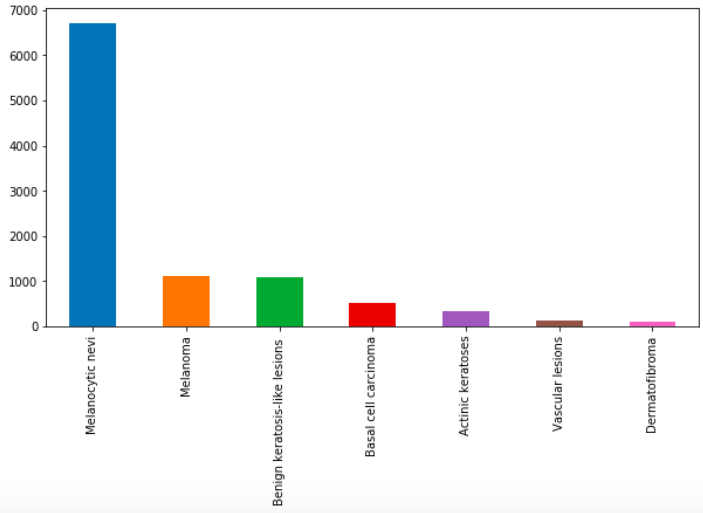
يتم التحقق من مجموعة البيانات بحثًا عن القيم أو أنواع البيانات المفقودة لكل حقل، ويتم تعبئتها بقيم خالية (بالنسبة للعمر،في هذه الحالة).

الخطوة 5: تحليل البيانات الاستكشافية

هذه الخطوة هي رسم البيانات وفهمها بشكل أفضل لأنفسنا ـ رؤية الميزات المختلفة لمجموعة البيانات، وكيفية توزيعها، وبعض الأرقام الفعلية.

```
fig, ax1 = plt.subplots(1, 1, figsize= (10, 5))
skin_df['cell_type'].value_counts().plot(kind='bar', ax=ax1)
```

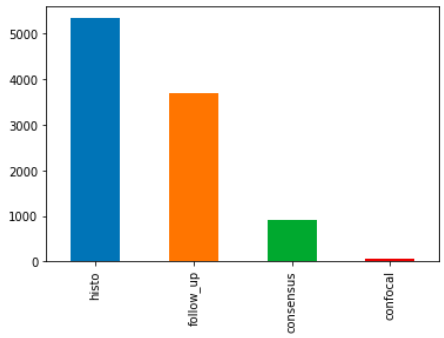
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7ae85ee48>
```



يتم توزيع مجموعة البيانات أيضًا إلى 4 فئات رئيسية من الجروح lesions: Histopathology, Confocal, follow up, and consensus.

```
skin_df['dx_type'].value_counts().plot(kind='bar')
```

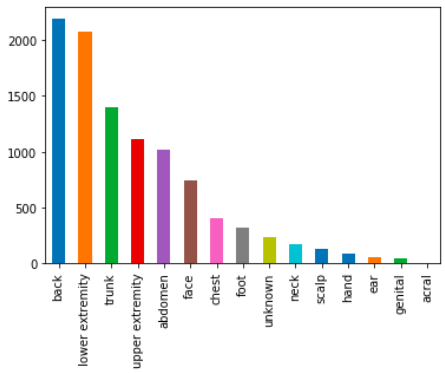
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7ae83fd30>
```



يمكن أيضًا تصور ميزات أخرى مثل موقع الآفات location of lesions وعمر / جنس المريض.

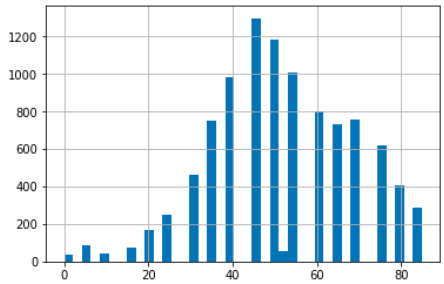
```
skin_df['localization'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7adf89668>
```




```
skin_df['age'].hist(bins=40)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7adea9588>
```



الخطوة 6: تحميل الصور وتغيير حجمها

يتم تحميل الصورفي عمود الصورة وتغيير حجمها حتى يتمكن Tensorflow من التعامل مع الحجم.

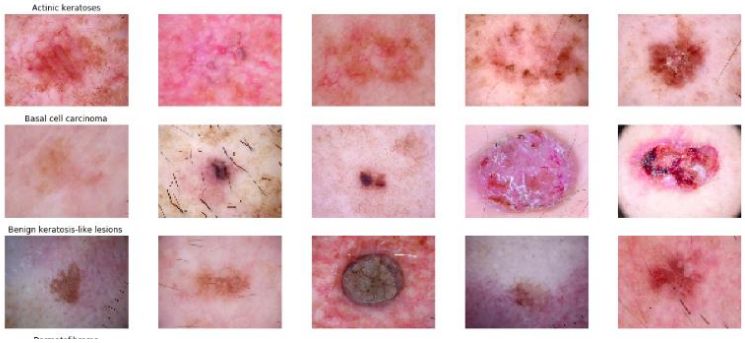
```
skin_df['image'] = skin_df['path'].map(lambda x: np.asarray(Image.open(x).resize((100,75
))))
```

```
skin_df.head()
```

	lesion_id	image_id	dx	dx_type	age	sex	localization	path
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	../input/ham10000_images_part_1/ISIC_0027419.jpg
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	../input/ham10000_images_part_1/ISIC_0025030.jpg
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	../input/ham10000_images_part_1/ISIC_0026769.jpg

الآن، يمكنك أيضًا عرض الصورفي مجموعة البيانات المنظمة حسب نوع سرطان الجلد.

```
n_samples = 5
fig, m_axs = plt.subplots(7, n_samples, figsize = (4*n_samples, 3*7))
for n_axs, (type_name, type_rows) in zip(m_axs,
                                         skin_df.sort_values(['cell_type']).groupby('cel
l_type')):
    n_axs[0].set_title(type_name)
    for c_ax, (_, c_row) in zip(n_axs, type_rows.sample(n_samples, random_state=1234).it
errows()):
        c_ax.imshow(c_row['image'])
        c_ax.axis('off')
fig.savefig('category_samples.png', dpi=300)
```



الخطوة 7: تقسيم مجموعات التدريب والاختبار

```
x_train_o, x_test_o, y_train_o, y_test_o = train_test_split(features, target, test_size=
0.20,random_state=1234)
```

يتم تقسيم البيانات إلى مجموعة تدريب واختبار مع قسم 80 20 على التوالي.

الخطوة 8: التسوية Normalization

```
x_train = np.asarray(x_train_o['image'].tolist())
x_test = np.asarray(x_test_o['image'].tolist())

x_train_mean = np.mean(x_train)
x_train_std = np.std(x_train)

x_test_mean = np.mean(x_test)
x_test_std = np.std(x_test)

x_train = (x_train - x_train_mean)/x_train_std
x_test = (x_test - x_test_mean)/x_test_std
```

يتم تسوية x_train، x_test بالطرح من القيم المتوسطة ثم القسمة على انحرافها.

الخطوة 9: ترميز التسمية Label Encoding

تحتوي التسمياتفي هذه المرحلة على الفئات السبع المختلفة لأنواع سرطان الجلد من الأرقام

```
# Perform one-hot encoding on the labels
y_train = to_categorical(y_train_o, num_classes = 7)
y_test = to_categorical(y_test_o, num_classes = 7)
```

من 0 إلى 6. ويتم ترميز هذه التسمياتفي متجهات واحد ساخن.

الخطوة 10: تقسيم التدريب والتحقق من صحة

```
x_train, x_validate, y_train, y_validate = train_test_split(x_train, y_train, test_size
= 0.1, random_state = 2)
```

```
# Reshape image in 3 dimensions (height = 75px, width = 100px , canal = 3)
x_train = x_train.reshape(x_train.shape[0], *(75, 100, 3))
x_test = x_test.reshape(x_test.shape[0], *(75, 100, 3))
x_validate = x_validate.reshape(x_validate.shape[0], *(75, 100, 3))
```

يتم تقسيم التدريب بشكل أكبر مع جزء صغير ـ 10٪، يتم الاحتفاظ به لمجموعة التحقق من الصحة (التي يتم تقييم النموذج عليها) ويتم استخدام الـ 90٪ المتبقية لتدريب النموذج.

تذكير سريع بالخطوات الأربع لشبكات CNN:

Convolution -> pooling -> flatten -> full connection

```
# Set the CNN model
# my CNN architechture is In -> [[Conv2D->relu]*2 -> MaxPool2D -> Dropout]*2 -> Flatten ->
 Dense -> Dropout -> Out
input_shape = (75, 100, 3)
num_classes = 7

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',padding = 'Same',input_shape=i
nput_shape))
model.add(Conv2D(32,kernel_size=(3, 3), activation='relu',padding = 'Same',))
model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu',padding = 'Same'))
model.add(Conv2D(64, (3, 3), activation='relu',padding = 'Same'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.40))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

بشكل اساسي...

- ابدأ بصورة إدخال.
- قم بتطبيق الفلاتر أو خرائط المعالم على الصورة، مما يعطي طبقة تلافيفية.
- قم بتقسيم خطية الصورة باستخدام دالة المعدل Relu.
- الصورة جاهزة لخطوة التجميع pooling (وهي من أجل "الثبات المكاني")
- بعد الانتهاء من التجميع، ينتهي الأمر بخريطة ميزات مجمعة.
- يتم تسوية خريطة المعالم المجمعة قبل إدخالهافي شبكة عصبية اصطناعية.

الخطوة 12: ضبط المحسن

```
# Define the optimizer
optimizer = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=Fa
lse)
```

```
# Compile the model
model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accur
acy"])
```

```
# Set a learning rate annealer
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
```

بمجرد إضافة طبقاتنا إلى النموذج، نحتاج إلى إعداد:

- دالة score.
- دالة الخطأ (معدل الخطأ بين التسميات الملحوظة والتوقعات) ـ تساعدفي قياس مدى ضعف أداء نموذجنا على الصور ذات التسميات المعروفة.
- خوارزمية التحسين ـ لتكرار وتحسين المعلمات (قيم الفلاتر والأوزان وانحياز الخلايا العصبية) لتقليل الخطأ.

مفاجئة!!! خطوة المكافأة 🎉

زيادة البيانات Data Augmentation ـ إنه اختياري. ولكن لتجنب مشكلة فرط التعلم overfitting، يمكننا توسيع مجموعة بيانات HAM 10000 بشكل مصطنع لتصبح أكبر. تكمن الفكرةفي تعديل بيانات التدريب بتحويلات صغيرة لإعادة إنتاج الأشكال المختلفة للصورة نفسها مع إضافات مثل التدرجات الرمادية والتقلبات الأفقية والتقلبات الرأسية والقص العشوائية والترجمات والدورات والمزيد.

```
# With data augmentation to prevent overfitting

datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.1, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally (fraction of total w
idth)
        height_shift_range=0.1,  # randomly shift images vertically (fraction of total he
ight)
        horizontal_flip=False,  # randomly flip images
        vertical_flip=False)  # randomly flip images

datagen.fit(x_train)
```

الخطوة 13: تدريب النموذج

يُدرب النموذج مع x_train، y_train. يتم اختيار حجم دفعة من 10 و50 حقبة ـ من المهم أن تضعفي اعتبارك أنه كلما كان حجم الدُفعة أصغر، كلما تم تدريب النموذج بشكل أكثر كفاءة.

```
# Fit the model
epochs = 50
batch_size = 10
history = model.fit_generator(datagen.flow(x_train,y_train, batch_size=batch_size),
                              epochs = epochs, validation_data = (x_validate,y_validate
),
                              verbose = 1, steps_per_epoch=x_train.shape[0] // batch_siz
e
                              , callbacks=[learning_rate_reduction])
```

```
Epoch 00029: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 30/50
721/721 [==============================] - 22s 31ms/step - loss: 0.5773 - acc: 0.7849 -
val_loss: 0.5816 - val_acc: 0.7893
Epoch 31/50
721/721 [==============================] - 22s 31ms/step - loss: 0.5632 - acc: 0.7842 -
val_loss: 0.5867 - val_acc: 0.7905
Epoch 32/50
721/721 [==============================] - 22s 30ms/step - loss: 0.5694 - acc: 0.7781 -
val_loss: 0.5915 - val_acc: 0.7830
Epoch 33/50
721/721 [==============================] - 23s 32ms/step - loss: 0.5659 - acc: 0.7829 -
val_loss: 0.6040 - val_acc: 0.7818
Epoch 34/50
171/721 [======>.......................] - ETA: 15s - loss: 0.5379 - acc: 0.7936
```

الخطوة 14: تقييم النموذج

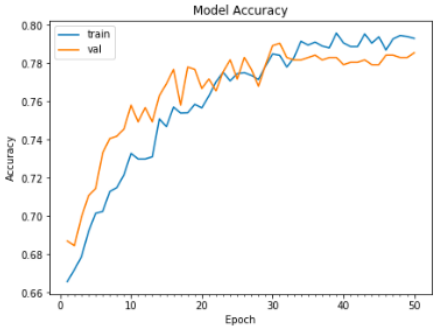
(دقة الاختبار والتحقق من الصحة، مصفوفة الارتباك، تحليل الحالات المصنفة بشكل خاطئ)

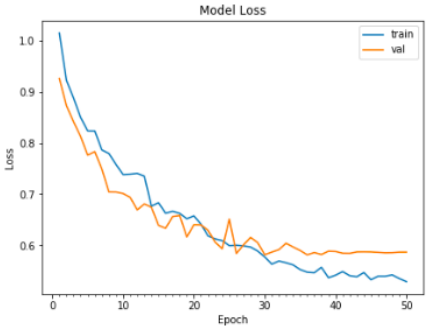
```
loss, accuracy = model.evaluate(x_test, y_test, verbose=1)
loss_v, accuracy_v = model.evaluate(x_validate, y_validate, verbose=1)
print("Validation: accuracy = %f  ;  loss_v = %f" % (accuracy_v, loss_v))
print("Test: accuracy = %f  ;  loss = %f" % (accuracy, loss))
model.save("model.h5")
```

```
2003/2003 [==============================] - 1s 694us/step
802/802 [==============================] - 0s 527us/step
Validation: accuracy = 0.785536  ;  loss_v = 0.586728
Test: accuracy = 0.764853  ;  loss = 0.616134
```

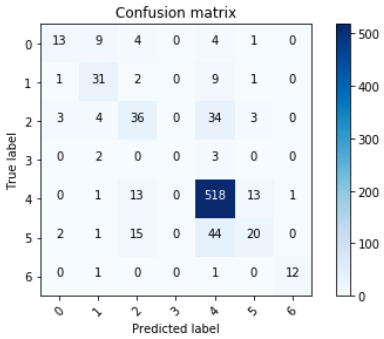
يتم فحص دقة الاختبار والتحقق من صحة نموذجنا، ويتم رسم مصفوفة الارتباك. يتم أيضًا تحديد عدد الصور المصنفة بشكل خاطئ لكل نوع.

```
plot_model_history(history)
```






```
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(7))
```



في النهاية، بلغت دقة النموذج حوالي 78٪ ويمكن تدريبه بسهولة على تجاوز 80٪. مقارنةً بالعين البشرية التي تبلغ دقتها 77.0344٪، والوقت والجهد اللذين تستغرقهما، من الآمن أن نقول إن نموذجنا هو الفائز من حيث الكفاءة.

8) تحديد سلالة الكلاب باستخدام التعلم العميق Dog's Breed Identification using Deep Learning

ذات مرة ذهبت إلى الحديقة ووجدت كلبًا لطيفًا جدًا. لم أكن على دراية بسلالتها وكان لدي فضول شديد لمعرفة أن الكلب ينتمي إلى أي عائلة؟ هل أنت أيضًا متشوق لمعرفة أنواع سلالات الكلاب dog breeds؟ إذا أنتفي المكان المناسب. فلنقم معًا بإنشاء نموذج التعلم العميق للتعرف على سلالة الكلاب بمجرد النظر إلى صورتها.

حول مشروع تحديد سلالة الكلاب

في مشروع تعلم الآلة بايثون هذا، سنبني نموذجًا لتحديد سلالة الكلاب. سنحول جميع صور الكلاب إلى تنسيق أرقام باستخدام "OpenCV" ثم نقوم بإدخالهافي Resnet50V2، وهو نوع خاص من الشبكات العصبية بموجب تقنية نقل التعلم Transfer Learning، والتي ستساعدنا في تحديد سلالة الكلاب.

مجموعة بيانات لمشروع تحديد سلالة الكلاب

يمكنك تنزيل مجموعة البيانات لهذا المشروع من هنا: **Dog's Breed Identification Dataset**

معمارية النموذج

ما هو نقل التعلم؟

في عملية نقل التعلم Transfer Learning، يتم استخدام طبقة نموذج مدرب مسبقًا بها أوزان ومعلمات لتدريب نموذج آخر. يتم تدريب طبقة النموذج المحددة مسبقًا على ملايين البيانات من الفئات المختلفة. هذه العملية مفيدة للغاية لأنها تقلل من وقت تدريب الشبكات العصبية وتؤدي أيضًا إلى انخفاض خطأ التعميم.

في هذا المشروع، سنستخدم الشبكة المتبقية (ResNet) التي تحتوي على طبقة شبكة مُدرَّبة مسبقًا. لذلك دعونا نرى ما هو Resnet.

ما هو ResNet؟

الشبكة المتبقية (ResNet) هي نوع معين من الشبكات العصبية التي تُستخدم للعديد من مشاكل الرؤية الحاسوبية. يحتوي ResNet على طبقات تلافيفية وتجميع وتنشيط وطبقات متصلة بالكامل مكدسة بإحدى الطبقات الأخرى. الشبكة العصبية التلافيفية CNN هي نوع من الشبكات العصبية العميقة، والتي تُستخدم لمعالجة الصور وتصنيفها. كما يوحي الاسم، تساعد الشبكة التلافيفيةفي تصنيف الصور المعقدة بضرب قيمة البكسل بالأوزان ثم جمعها.

تم تدريب طبقات ResNet هذه مسبقًا على أكثر من مليون صورة من قاعدة بيانات ImageNet. نظرًا للعديد من الطبقات، تحل شبكة ResNet المشكلات المعقدة وتزيد من دقة النموذج وأدائه.

تستخدم كل شبكة ResNet فلتراً أوليًا أو نواة بحجم 3 × 3 و 7 × 7 بخطوة 2. وهناك إصدارات عديدة من ResNet. في هذا المشروع، سنستخدم Resnet50V2 (الإصدار 2) الذي يبلغ عمقه 50 طبقة ويطبق خاصية Batch Normalization، ودالة تنشيط RELU قبل مضاعفة الإدخال بالعمليات التلافيفية (مصفوفة الوزن).

متطلبات المشروع

يمكنك تثبيت جميع الوحدات النمطية لهذا المشروع باستخدام الأمر التالي:

```
pip install numpy, pandas, opencv-python , tensorflow, matplotlib , sklearn
```

الإصدارات المستخدمةفي هذا المشروع لبايثون والوحدات النمطية المقابلة لها هي كما يلي:

1) python: 3.8.5 (or 3.x)
2) tensorflow: 2.3.1
3) opencv: 4.1.2
4) sklearn: 0.24.2
5) numpy: 1.19.5
6) pandas: 1.1.5
7) matplotlib : 3.2.2

ملاحظة: يجب أن يكون إصدار TensorFlow 2.2 أو أعلى من أجل استخدام keras أو تثبيت keras مباشرة.

سأستخدم منصة Google Colab من Google للتدريب على النموذج لأنه يوفر أداة GPU مجانية.

هيكل المشروع

- train.csv: يحتوي هذا المجلد على صور سنستخدمها لتدريب نموذجنا. يوجد 10،222 صورة في هذا المجلد.
- test.csv: يحتوي هذا المجلد على صور سنستخدمها لاختبار نموذجنا المُدرب. يوجد 10357 صورة في هذا المجلد.
- labels.csv: يحتوي هذا الملف على صور تسمى عمود "id" وعمود "breed" الذي يحتوي على أسماء السلالات المعنية.

- model/: يحتوي هذا الدليل على المُحسِّن والمقاييس والأوزان الخاصة بنموذجنا المُدرَّب.
- dogbreed_identification.py: هذا هو الملف حيث سنكتب الكود الخاص بنا لتدريب نموذجنا والتنبؤ.

خطوات مشروع تحديد سلالة الكلاب:

الخطوة 1: استيراد المكتبات

أولاً، سننشئ ملفًا يسمى "dogbreed_identification.py" ونستورد جميع المكتبات التي تمت مشاركتهافي قسم المتطلبات الأساسية.

الكود:

```
#TechVidvan
# load all required libraries for Dog's Breed Identification Project
import cv2
import numpy as np
import pandas as pd
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import load model, Model
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D,
Dropout, BatchNormalization
from tensorflow.keras.applications.resnet_v2 import ResNet50V2,
preprocess_input
```

الخطوة 2: تحليل ملف مجموعة البيانات.

سوف نتخطى "labels.csv" ونحولها إلى إطار بيانات Dataframe. سيخزن متغير "train_file" موقع مجلد التدريب الذي يحتوي على صور التدريب وسيخزن متغير "test_file" موقع مجلد الاختبار الذي يحتوي على صور الاختبار.

الكود:

```
#read the csv file
df_labels = pd.read_csv("labels.csv")
#store training and testing images folder location
train_file = 'train/'
test_file = 'test/'
```

ملف مجموعة البيانات:

	A	B	C
1	id	breed	
2	000bec180eb18c7604dcecc8fe0dba07	boston_bull	
3	001513dfcb2ffafc82cccf4d8bbaba97	dingo	
4	001cdf01b096e06d78e9e5112d419397	pekinese	
5	00214f311d5d2247d5dfe4fe24b2303d	bluetick	
6	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever	
7	002211c81b498ef88e1b40b9abf84e1d	bedlington_terrier	
8	00290d3e1fdd27226ba27a8ce248ce85	bedlington_terrier	
9	002a283a315af96eaea0e28e7163b21b	borzoi	
10	003df8b8a8b05244b1d920bb6cf451f9	basenji	
11	0042188c895a2f14ef64a918ed9c7b64	scottish_deerhound	
12	004396df1acd0f1247b740ca2b14616e	shetland_sheepdog	
13	0067dc3eab0b3c3ef0439477624d85d6	walker_hound	
14	00693b8bc2470375cc744a6391d397ec	maltese_dog	
15	006cc3ddb9dc1bd827479569fcdc52dc	bluetick	
16	0075dc49dab4024d12fafe67074d8a81	norfolk_terrier	
17	00792e341f3c6eb33663e415d0715370	african_hunting_dog	
18	007b5a16db9d9ff9d7ad39982703e429	wire-haired_fox_terrier	
19	007b8a07882822475a4ce6581e70b1f8	redbone	
20	007ff9a78eba2aebb558afea3a51c469	lakeland_terrier	
21	008887054b18ba3c7601792b6a453cc3	boxer	

دعونا نتحقق من عدد أنواع سلالات الكلاب الموجودةفي مجموعة البيانات الخاصة بنا.

الكود:

```
#check the total number of unique breed in our dataset file
print("Total number of unique Dog Breeds
:",len(df_labels.breed.unique()))
```

المخرجات:

```
Total number of unique Dog Breeds : 120
```

كما ترون، هناك 120 نوعًا فريدًا من سلالات الكلاب.في هذا المشروع، سنستخدم 60 نوعًا من سلالات الكلاب. وفقًا لمواصفات نظامك، يمكنك زيادة هذه القيمة أو تقليلها.

الكود:

```
#specify number
num_breeds = 60
im_size = 224
batch_size = 64
encoder = LabelEncoder()
```

المخرجات:

```
60
yorkshire_terrier , whippet , welsh_springer_spaniel , walker_hound , toy_terrier

tibetan_terrier , sussex_spaniel , standard_poodle , soft-coated_wheaten_terrier , siberian_husky

shetland_sheepdog , scottish_deerhound , schipperke , saluki , rottweiler

redbone , pomeranian , pekinese , otterhound , norwich_terrier

norfolk_terrier , miniature_schnauzer , miniature_pinscher , maltese_dog , malamute

leonberg , labrador_retriever , komondor , kelpie , japanese_spaniel

irish_wolfhound , irish_terrier , ibizan_hound , greater_swiss_mountain_dog , great_dane

golden_retriever , german_short-haired_pointer , french_bulldog , eskimo_dog , english_springer

english_foxhound , dingo , dandie_dinmont , collie , clumber

chihuahua , cardigan , bull_mastiff , briard , boxer

boston_bull , border_terrier , bluetick , blenheim_spaniel , bernese_mountain_dog

beagle , basenji , appenzeller , airedale , afghan_hound
```

الخطوة 3: المعالجة المسبقة للبيانات

كما ناقشنا سابقًا، سنستخدم 60 نوعًا مختلفًا من سلالات الكلاب. لذلك سنختار تلك السلالات الأكثر عددًا ولديها المزيد من الصور. لهذا، سنأخذ المساعدة من دالة 'value_counts()' للـpandas. بعد ذلك، قم بتغيير إطار البيانات الذي يحتوي على سجلات لتلك السلالات الستين فقط.

الكود:

```
#get only 60 unique breeds record
breed_dict = list(df_labels['breed'].value_counts().keys())
new_list = sorted(breed_dict,reverse=True)[:num_breeds*2+1:2]
#change the dataset to have only those 60 unique breed records
df_labels = df_labels.query('breed in @new_list')
```

بمساعدة عمود المعرف id، سننشئ عمودًا آخر "img_file" والذي سيكون له اسم صورة بامتدادات. سيكون هذا مفيدًا جدًا، وإلا يتعين علينا إلحاق الامتداد بعد كل تكرار.

الكود:

```
#create new column which will contain image name with the image
extension
df_labels['img_file'] = df_labels['id'].apply(lambda x: x + ".jpg")
```

المخرجات:

	id	breed	img_file
0	000bec180eb18c7604dcecc8fe0dba07	boston_bull	000bec180eb18c7604dcecc8fe0dba07.jpg
1	001513dfcb2ffafc82cccf4d8bbaba97	dingo	001513dfcb2ffafc82cccf4d8bbaba97.jpg
2	001cdf01b096e06d78e9e5112d419397	pekinese	001cdf01b096e06d78e9e5112d419397.jpg
3	00214f311d5d2247d5dfe4fe24b2303d	bluetick	00214f311d5d2247d5dfe4fe24b2303d.jpg
4	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever	0021f9ceb3235effd7fcde7f7538ed62.jpg
8	003df8b8a8b05244b1d920bb6cf451f9	basenji	003df8b8a8b05244b1d920bb6cf451f9.jpg
9	0042188c895a2f14ef64a918ed9c7b64	scottish_deerhound	0042188c895a2f14ef64a918ed9c7b64.jpg
10	004396df1acd0f1247b740ca2b14616e	shetland_sheepdog	004396df1acd0f1247b740ca2b14616e.jpg
11	0067dc3eab0b3c3ef0439477624d85d6	walker_hound	0067dc3eab0b3c3ef0439477624d85d6.jpg
12	00693b8bc2470375cc744a6391d397ec	maltese_dog	00693b8bc2470375cc744a6391d397ec.jpg

كما ترى، لدينا عمود جديد 'img_file' والذي تم تمييزه.

الخطوة 4: ترميز البيانات وقياسها

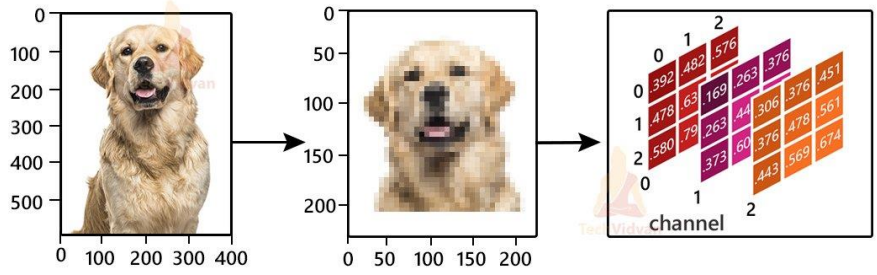
الآن أصبحت مجموعة البيانات الخاصة بنا جاهزة، سنقوم بإجراء عمليات على السجلات لأغراض التدريب والاختبار.

يمكننا نحن البشر رؤية الصورة ومعرفة ما بداخلها بسهولة، لكن الآلة تتطلب بيانات رقمية للتعرف على كل شيء.

لهذا، سنقوم بتحويل صورتنا إلى التنسيق العددي. لذا دعونا نفهم كيف يتم ذلك.

الصور عبارة عن مجموعات من وحدات البكسل الصغيرة (أو عناصر الصورة) وهي أصغر المعلومات حول الصورة. كما تعلم، جميع الصور الملونة عبارة عن مزيج من ثلاثة ألوان أساسية وهي "أحمر" و "أخضر" و "أزرق". لذلك، بالنسبة للصور الملونة، هناك ثلاث مصفوفات أو قنوات.

يسمى كل عنصر من عناصر هذه المصفوفة بالبكسل. يعتمد حجم المصفوفة على عدد البكسل. تشير قيمة البكسل إلى شدة أو سطوع البكسل وتتراوح من 0–255. تمثل قيمة البكسل الأصغر الأقرب للصفر اللون الأسود بينما تمثل القيمة الأكبر الأقرب من 255 اللون الأبيض.



فيما يلي تنسيقات مختلفة مثل Grayscale وRGB وHSV وCMYK التي يتم تخزين الصور بها. RGB هي واحدة من أكثرها شعبية وسوف نستخدمهافي مشروعنا.

بمساعدة مكتبة "opencv" سنقرأ صورنا باستخدام دالة 'imread()'.

سيعود Numpy Array بتنسيق الارتفاع والعرض والقناة. جميع الصورفي مجموعة البيانات لدينا بأشكال مختلفة، لذا قم بتغيير حجم كل الصور بنفس العرض والارتفاع، أيفي حالتنا 224 × 224. سنقوم أيضًا بقياس جميع قيم المصفوفة الخاصة بنافي النطاق 1– و 1 باستخدام دالة 'preprocess_input()'.

الكود:

```
#create a numpy array of the shape
#(number of dataset records, image size , image size, 3 for rgb
channel ayer)
#this will be input for model
train_x = np.zeros((len(df_labels), im_size, im_size, 3),
dtype='float32')

#iterate over img_file column of our dataset
for i, img_id in enumerate(df_labels['img_file']):
  #read the image file and convert into numeric format
  #resize all images to one dimension i.e. 224x224
  #we will get array with the shape of
  # (224,224,3) where 3 is the RGB channels layers
  img =
cv2.resize(cv2.imread(train_file+img_id,cv2.IMREAD_COLOR),((im_size,im
_size)))
  #scale array into the range of -1 to 1.
  #preprocess the array and expand its dimension on the axis 0
  img_array = preprocess_input(np.expand_dims(np.array(img[...,::-
1].astype(np.float32)).copy(), axis=0))
  #update the train_x variable with new element
  train_x[i] = img_array
```

نحتاج أيضًا إلى تشفير عمود Breed الخاص بنا لأنه بتنسيق النص. بعد الترميز، سيكون لعمود السلالة قيمة من 0 إلى العدد الإجمالي لطول العمود ناقص 1. يتم تعيين هذه الأرقام أبجديًا.

الكود:

```
#This will be the target for the model.
#convert breed names into numerical format
train_y = encoder.fit_transform(df_labels["breed"].values)
```

الخطوة 5: مجموعات التدريب والاختبار

بعد أن تصبح مجموعات المدخلات والأهداف جاهزة، سنقسم نموذجنا إلى مجموعات تدريب واختبار بنسبة 20:80، حيث سيتم استخدام 80٪ من إجمالي البيانات للتدريب و 20٪ المتبقية لأغراض الاختبار.

الكود:

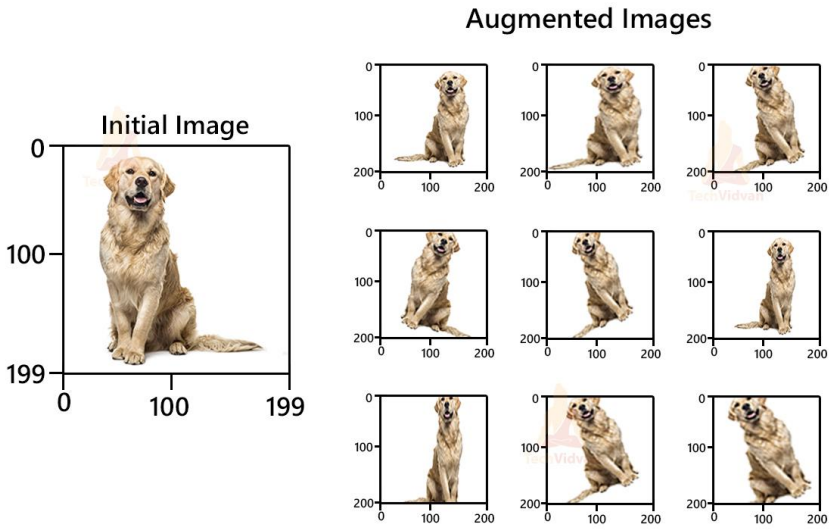
```
#split the dataset in the ratio of 80:20.
#80% for training and 20% for testing purpose
x_train, x_test, y_train, y_test =
train_test_split(train_x,train_y,test_size=0.2,random_state=42)
```

الخطوة 6: زيادة الصور

الزيادة Augmentation هوفي الأساس تقنية يمكن استخدامها لتوسيع حجم الصور بشكل مصطنعفي الوقت الفعلي عن طريق إنشاء إصدارات مختلفة معدلة. سيساعد هذا النموذج على التعميم وأيضًا تحسين الأداء.

بعض تقنيات الزيادة الأكثر استخدامًا للصور هي:

- **زيادة الموضع Position Augmentation**: يغير موضع البكسل (العناصر الموجودة في المصفوفة الخاصة بنا) باستخدام تقنيات القياس والترجمة والدوران والتقليب والقص.
- **زيادة اللون Color Augmentation**: يغير قيمة البكسل (العناصر الموجودة في المصفوفة الخاصة بنا) عن طريق تغيير مستويات السطوع والتباين والتدرج والتشبع.



سنحصل على مساعدة 'ImageDataGenerator()' لإنشاء أنواع مختلفة من التدريب واختبار الصور عن طريق تعديل الصور الأصلية.

الكود:

```
#Image augmentation using ImageDataGenerator class
train_datagen = ImageDataGenerator(rotation_range=45,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.25,
                                   horizontal_flip=True,
                                   fill_mode='nearest')

#generate images for training sets
train_generator = train_datagen.flow(x_train,
                                     y_train,
                                     batch_size=batch_size)

#same process for Testing sets also by declaring the instance
test_datagen = ImageDataGenerator()

test_generator = test_datagen.flow(x_test,
                                   y_test,
                                   batch_size=batch_size)
```

الخطوة 7: بناء النموذج

كما ناقشنا سابقًا، سنستخدم ResNet50V2 مع وجود معلمات مدربة من مجموعة بيانات Imagenet لبناء نموذجنا. لن نقوم بتضمين الطبقة العليا التي هي ناتج الشبكة سابقة التدريب، وبدلاً من ذلك، سنستبدلها بمدخلات لها شكل أبعاد mxnx3. تتوقع الطبقة شكل إدخال 224 × 224.

الكود:

```
#building the model using ResNet50V2 with input shape of our image
array
#weights for our network will be from of imagenet dataset
#we will not include the first Dense layer
resnet = ResNet50V2(input_shape = [im_size,im_size,3],
weights='imagenet', include_top=False)
#freeze all trainable layers and train only top layers
for layer in resnet.layers:
    layer.trainable = False

#add global average pooling layer and Batch Normalization layer
x = resnet.output
x = BatchNormalization()(x)
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
#add fully connected layer
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
```

وأخيرًا، قم بإنشاء طبقة كثيفة (متصلة بالكامل) للإخراج بالشكل الذي يساوي عدد السلالات ودالة التنشيط "softmax". ثم تهيئة فئة النموذج بإدخال الشبكة والطبقة الكثيفة كإخراج.

الكود:

```
#add output layer having the shape equal to number of breeds
predictions = Dense(num_breeds, activation='softmax')(x)

#create model class with inputs and outputs
model = Model(inputs=resnet.input, outputs=predictions)
#model.summary()
```

المخرجات:

تنزيل البيانات من:

https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50v2_weights_tf_dim_ordering_tf_kernels_notop.h5

94674944/94668760 [==============================] – 1s ous/step

الخطوة 7: تدريب النموذج

لتدريب البيانات، سنستخدم مُحسِّن "RMSprop" بمعدل تعلم e-31 (0.001) على حجم دفعة 64 (مجموعة من 64 صورة لكل تكرار) لمدة 20 حقبة.

احفظ النموذج من أجل عملية التنبؤ.

الكود:

```
#epochs for model training and learning rate for optimizer
epochs = 20
learning_rate = 1e-3

#using RMSprop optimizer to compile or build the model
optimizer = RMSprop(learning_rate=learning_rate,rho=0.9)
model.compile(optimizer=optimizer,
              loss='sparse_categorical_crossentropy',
              metrics=["accuracy"])

#fit the training generator data and train the model
hist = model.fit(train_generator,
                 steps_per_epoch= x_train.shape[0] // batch_size,
                 epochs= epochs,
                 validation_data= test_generator,
                 validation_steps= x_test.shape[0] // batch_size)

#Save the model for prediction
model.save("model")
```

المخرجات:

```
Epoch 14/20
64/64 [==============================] - 57s 888ms/step - loss: 0.7355 - accuracy: 0.7760 - val_loss: 0.7189 - val_accuracy: 0.7979
Epoch 15/20
64/64 [==============================] - 57s 890ms/step - loss: 0.7000 - accuracy: 0.7821 - val_loss: 0.7163 - val_accuracy: 0.7998
Epoch 16/20
64/64 [==============================] - 58s 902ms/step - loss: 0.6872 - accuracy: 0.7868 - val_loss: 0.7308 - val_accuracy: 0.8027
Epoch 17/20
64/64 [==============================] - 58s 898ms/step - loss: 0.6798 - accuracy: 0.7937 - val_loss: 0.7465 - val_accuracy: 0.8037
Epoch 18/20
64/64 [==============================] - 58s 897ms/step - loss: 0.6778 - accuracy: 0.7961 - val_loss: 0.7468 - val_accuracy: 0.7920
Epoch 19/20
64/64 [==============================] - 57s 888ms/step - loss: 0.7084 - accuracy: 0.7888 - val_loss: 0.7390 - val_accuracy: 0.7979
Epoch 20/20
64/64 [==============================] - 57s 894ms/step - loss: 0.6405 - accuracy: 0.8050 - val_loss: 0.7505 - val_accuracy: 0.7920
```

كما ترى حصلنا على دقة 80.50٪ وهو أمر جيد لأننا نأخذ 60 سلالة فقط من الكلاب.

الخطوة 8: التنبؤ

أخيرًا، سوف نتنبأ بسلالة صورتنا باستخدام النموذج المدرب. للتنبؤ، أستخدم إحدى صور الكلاب الخاصة بصديقي، وهي سلالة 'Rottweiler' تم التقاطها من هاتفه.

الكود:

```
#load the model
model = load_model("model")

#get the image of the dog for prediction
pred_img_path = 'rottweiler.jpg'
#read the image file and convert into numeric format
#resize all images to one dimension i.e. 224x224
pred_img_array =
cv2.resize(cv2.imread(pred_img_path,cv2.IMREAD_COLOR),((im_size,im_siz
e)))
#scale array into the range of -1 to 1.
#expand the dimension on the axis 0 and normalize the array values
```

```
pred_img_array =
preprocess_input(np.expand_dims(np.array(pred_img_array[...,::-
1].astype(np.float32)).copy(), axis=0))

#feed the model with the image array for prediction
pred_val = model.predict(np.array(pred_img_array,dtype="float32"))

#display the image of dog
cv2.imshow("TechVidvan",cv2.resize(cv2.imread(pred_img_path,cv2.IMREAD
_COLOR),((im_size,im_size))))

#display the predicted breed of dog
pred breed = sorted(new list)[np.argmax(pred val)]
print("Predicted Breed for this Dog is :",pred_breed)
```

مخرجات تحديد سلالة الكلاب

الملخص

في هذا المشروع، قمنا بتطوير نموذج يتنبأ بالسلالة من صورة الكلب باستخدام الشبكة العصبية المتبقية، ResNet50V2. لقد حصلنا على دقة 80.50٪ وهو أمر جيد لأننا أخذنا 60 فئة فريدة فقط من السلالات للتدريب ومجموعات الاختبار و20 حقبة فقط. يمكنك زيادة عدد فئات السلالات والحقب لزيادة دقة النموذج.

9) الكشف عن الأخبار الزائفة باستخدام لغة بايثون والتعلم الآلي
Detecting Fake News with Python and Machine Learning

هل تثق في كل الأخبار التي تسمعها من وسائل التواصل الاجتماعي؟

كل الأخبار ليست حقيقية، أليس كذلك؟

كيف ستكتشف الأخبار الكاذبة؟

الجواب هو بايثون. من خلال ممارسة مشروع بايثون المتقدم هذا للكشف عن الأخبار المزيفة، ستفرق بسهولة بين الأخبار الحقيقية والمزيفة.

قبل المضي قدمًا في مشروع التعلم الآلي هذا، كن على دراية بالمصطلحات المتعلقة به مثل الأخبار المزيفة، وtfidfvectorizer، وPassiveAggressive Classifier.

ما هي الأخبار الكاذبة؟

نوع من الصحافة الصفراء، الأخبار المزيفة fake news تحتوي على أجزاء من الأخبار قد تكون خدعة وتنتشر بشكل عام عبر وسائل التواصل الاجتماعي ووسائل الإعلام الأخرى عبر الإنترنت. غالبًا ما يتم ذلك لتعزيز أو فرض أفكار معينة وغالبًا ما يتم تحقيقه من خلال جداول الأعمال السياسية. قد تحتوي مثل هذه العناصر الإخبارية على ادعاءات كاذبة و / أو مبالغ فيها، وقد ينتهي بها الأمر إلى أن تصبح فيروسية بواسطة الخوارزميات، وقد ينتهي الأمر بالمستخدمين في فقاعة تصفية.

ما هو TfidfVectorizer؟

TF (Term Frequency): عدد المرات التي تظهر فيها كلمة في مستند ما هو مصطلح "Frequency ". تعني القيمة الأعلى أن المصطلح يظهر في كثير من الأحيان أكثر من غيره، وبالتالي، يعد المستند مطابقًا جيدًا عندما يكون المصطلح جزءًا من مصطلحات البحث.

IDF (Inverse Document Frequency): قد تكون الكلمات التي تظهر في المستند عدة مرات ، ولكنها تظهر أيضًا مرات عديدة في العديد من المستندات الأخرى ، غير ذات صلة. IDF هو مقياس لمدى أهمية المصطلح في المجموعة بأكملها.

يقوم TfidfVectorizer بتحويل مجموعة من المستندات الأولية إلى مصفوفة من ميزات TF-IDF.

ما هو PassiveAggressiveClassifier ؟

PassiveAggressiveClassifier هي خوارزميات التعلم عبر الإنترنت. تظل هذه الخوارزمية سلبية للحصول على نتيجة تصنيف صحيحة، وتتحول إلى عدوانية aggressiveفي حالة سوء التقدير والتحديث والتعديل. على عكس معظم الخوارزميات الأخرى، فهي لا تتقارب. والغرض منه هو إجراء تحديثات تصحح الخطأ، مما يؤدي إلى تغيير طفيف للغايةفي معيار متجه الوزن.

كشف الأخبار الكاذبة في بايثون

لبناء نموذج لتصنيف الخبر بدقة على أنه حقيقي أو مزيف.

حول اكتشاف الأخبار الكاذبة باستخدام بايثون

مشروع بايثون المتقدم هذا لاكتشاف صفقات الأخبار الوهمية مع الأخبار الكاذبة والحقيقية. باستخدام sklearn، نقوم ببناء TfidfVectorizer على مجموعة البيانات الخاصة بنا. بعد ذلك، نقوم بتهيئة مصنف PassiveAggressive ونلائم النموذج.في النهاية، تخبرنا درجة الدقة ومصفوفة الارتباك عن مدى جودة نموذجنا.

مجموعة بيانات الأخبار المزيفة

مجموعة البيانات التي سنستخدمها لمشروع بايثون هذا ـ سنسميها news.csv. مجموعة البيانات هذه لها شكل 4×7796. يحدد العمود الأول الأخبار news، والثاني والثالث هما العنوان title والنص text، ويحتوي العمود الرابع على تسميات تشير إلى ما إذا كانت الأخبار حقيقية أم مزيفة. تشغل مجموعة البيانات 29.2 ميغابايت من المساحة ويمكنك تنزيلها من هنا.

متطلبات المشروع

ستحتاج إلى تثبيت المكتبات التالية بـ pip:

```
pip install numpy pandas sklearn
```

ستحتاج إلى تثبيت Jupyter Lab لتشغيل الكود البرمجي الخاص بك. انتقل إلى موجه الأوامر الخاص بك وقم بتشغيل الأمر التالي:

```
C:\Users\DataFlair>jupyter lab
```

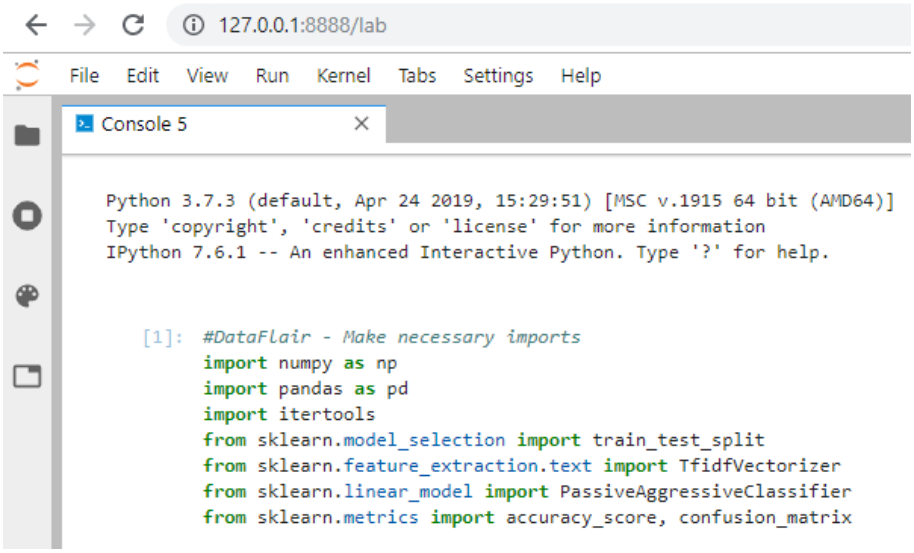
سترى نافذة متصفح جديدة تفتح؛ إنشاء وحدة تحكم جديدة واستخدامها لتشغيل التعليمات البرمجية الخاصة بك. لتشغيل أسطر متعددة من التعليمات البرمجية مرة واحدة، اضغط على Shift + Enter.

خطوات الكشف عن الأخبار الكاذبة باستخدام بايثون

اتبع الخطوات أدناه لاكتشاف الأخبار المزيفة وأكمل مشروع بايثون المتقدم الأول:

الخطوة 1: قم بإجراء عمليات الاستيراد الضرورية:

```
import numpy as np
import pandas as pd
import itertools
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```



الخطوة 2: الآن، دعنا نقرأ البياناتفي DataFrame، ونحصل على شكل البيانات وأول 5 سجلات.

```
#Read the data
df=pd.read_csv('D:\\DataFlair\\news.csv')

#Get shape and head
df.shape
df.head()
```

```
[2]: #Read the data
df=pd.read_csv('D:\\DataFlair\\news.csv')

#Get shape and head
df.shape
df.head()
```

[2]:

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

الخطوة 3: احصل على التسميات من DataFrame.

```
#DataFlair - Get the labels
labels=df.label
```

```
labels.head()
```

```
[3]: #DataFlair - Get the labels
     labels=df.label
     labels.head()

[3]: 0    FAKE
     1    FAKE
     2    REAL
     3    FAKE
     4    REAL
     Name: label, dtype: object
```

الخطوة 4: قسّم مجموعة البيانات إلى مجموعات تدريب واختبار.

```
#DataFlair - Split the dataset
x_train,x_test,y_train,y_test=train_test_split(df['text'], labels,
test_size=0.2, random_state=7)
```

```
[4]: #DataFlair - Split the dataset
     x_train,x_test,y_train,y_test=train_test_split(df['text'], labels, test_size=0.2, random_state=7)
```

الخطوة 5: دعونا نهيئ TfidfVectorizer بكلمات توقف stop words من اللغة الإنجليزية وحد أقصى لتكرار المستندات يبلغ 0.7 (سيتم تجاهل المصطلحات ذات التردد العالي للمستند). كلمات التوقف هي الكلمات الأكثر شيوعًافي اللغة التي يجب تصفيتها قبل معالجة بيانات اللغة الطبيعية. وتحول TfidfVectorizer مجموعة من المستندات الأولية إلى مصفوفة من ميزات TF-IDF.

الآن، قم بتدريب fit وتحويل vectorizerفي مجموعة التدريب، وقم بتحويل vectorizerفي مجموعة الاختبار.

```
#DataFlair - Initialize a TfidfVectorizer
tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)

#DataFlair - Fit and transform train set, transform test set
tfidf_train=tfidf_vectorizer.fit_transform(x_train)
tfidf_test=tfidf_vectorizer.transform(x_test)
```

```
[5]: #DataFlair - Initialize a TfidfVectorizer
     tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)

     #DataFlair - Fit and transform train set, transform test set
     tfidf_train=tfidf_vectorizer.fit_transform(x_train)
     tfidf_test=tfidf_vectorizer.transform(x_test)
```

الخطوة 6: بعد ذلك، سنقوم بتهيئة PassiveAggressiveClassifier.. سنلائم هذا على tfidf_train و y_train.

بعد ذلك، سنتنبأ بمجموعة الاختبار من TfidfVectorizer ونحسب الدقة مع accuracy_score() من sklearn.metrics.

```
#DataFlair - Initialize a PassiveAggressiveClassifier
pac=PassiveAggressiveClassifier(max_iter=50)
pac.fit(tfidf_train,y_train)

#DataFlair - Predict on the test set and calculate accuracy
y_pred=pac.predict(tfidf_test)
score=accuracy_score(y_test,y_pred)
print(f'Accuracy: {round(score*100,2)}%')
```

```
[6]: #DataFlair - Initialize a PassiveAggressiveClassifier
     pac=PassiveAggressiveClassifier(max_iter=50)
     pac.fit(tfidf_train,y_train)

     #DataFlair - Predict on the test set and calculate accuracy
     y_pred=pac.predict(tfidf_test)
     score=accuracy_score(y_test,y_pred)
     print(f'Accuracy: {round(score*100,2)}%')

     Accuracy: 92.82%
```

الخطوة 7: حصلنا على دقة 92.82٪ مع هذا النموذج. أخيرًا، دعنا نطبع مصفوفة الارتباك للحصول على نظرة ثاقبة لعدد السلبيات والإيجابيات الخاطئة والصحيحة.

```
#DataFlair - Build confusion matrix
confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])
```

```
[7]: #DataFlair - Build confusion matrix
     confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])

[7]: array([[589,  49],
            [ 42, 587]], dtype=int64)
```

```
[ ]:
```

لذلك من خلال هذا النموذج، لدينا 589 قيمة موجبة حقيقية، و587 قيمة سلبية حقيقية، و42 إيجابية كاذبة، و49 قيمة سلبية كاذبة.

الملخص

اليوم، تعلمنا اكتشاف الأخبار المزيفة باستخدام بايثون. لقد أخذنا مجموعة بيانات سياسية، وقمنا بتطبيق TfidfVectorizer، وقمنا بتهيئة PassiveAggressiveClassifier، وتناسب نموذجنا. انتهى بنا الأمر بالحصول على دقة مقدارها 92.82٪.

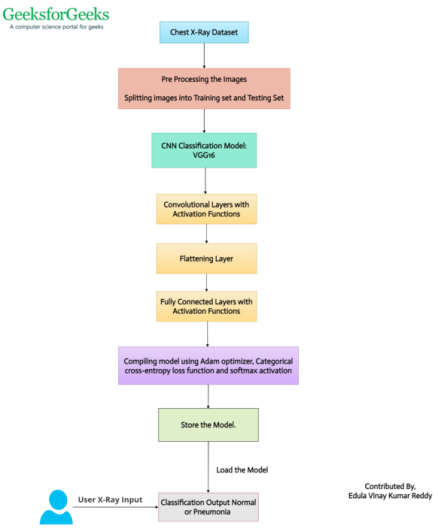
10) كشف الالتهاب الرئوي باستخدام التعلم العميق Pneumonia Detection using Deep Learning

في هذه المقالة سنناقش حل مشكلة طبية مثل الالتهاب الرئوي Pneumonia وهو مرض خطير قد يحدثفي إحدى الرئتين أو كليهما وينتج عادة عن فيروسات أو فطريات أو بكتيريا. سوف نكتشف مرض الرئة هذا بناءً على صور الأشعة السينية التي لدينا. تم أخذ مجموعة بيانات الأشعة السينية للصدر من Kaggle والتي تحتوي على صور أشعة سينية مختلفة متمايزة بفئتين "التهاب رئوي Pneumonia " و "عادي Normal ". سنقوم بإنشاء نموذج تعليمي عميق يخبرنافي الواقع ما إذا كان الشخص مصابًا بمرض الالتهاب الرئوي أو لا يعاني من الالتهاب الرئوي.

الأدوات والتقنيات:

- **VGG16**: إنها بُنية شبكة عصبية تلافيفية (CNN) سهلة الاستخدام على نطاق واسع مستخدمةفي ImageNet وهي مهمة قاعدة بيانات مرئية ضخمة تُستخدمفي أبحاث برامج التعرف على الأشياء المرئية.
- **نقل التعلم Transfer learning (TL)**: هو أسلوبفي التعلم العميق يركز على أخذ شبكة عصبية مُدربة مسبقًا وتخزين المعرفة المكتسبة أثناء حل مشكلة واحدة وتطبيقها على مجموعات بيانات مختلفة جديدة.في هذه المقالة، يمكن تطبيق المعرفة المكتسبة أثناء تعلم التعرف على 1000 فئة مختلفةفي ImageNet عند محاولة التعرف على المرض.

معمارية النموذج:



الوحدات المطلوبة:

- **Keras**: إنها وحدة بايثون للتعلم العميق تعملفي الجزء العلوي من مكتبة TensorFlow. تم إنشاؤه لجعل تنفيذ نماذج التعلم العميق أسهل وأسرع ما يمكن للبحث والتطوير. نظرًا لحقيقة أن Keras تعمل على قمة TensorFlow، يتعين علينا تثبيت TensorFlow أولاً. لتثبيت هذه المكتبة، اكتب الأوامر التاليةفي IDE / Terminal.

```
pip install tensorflow

pip install keras
```

- **SciPy**: SciPy هي وحدة بايثون مجانية ومفتوحة المصدر تُستخدمفي الحوسبة التقنية والعلمية. نظرًا لأننا نطلب تحويلات الصورفي هذه المقالة، يتعين علينا تثبيت وحدة SciPy. لتثبيت هذه المكتبة، اكتب الأمر التاليفي IDE / Terminal.

```
pip install scipy
```

- **glob**:في بايثون ، تُستخدم وحدة glob لاسترداد الملفات / أسماء المسار المطابقة لنمط محدد. لمعرفة عدد الفئات الموجودةفي مجلد مجموعة بيانات التدريب الخاص بنا، نستخدم هذه الوحدةفي هذه المقالة.

```
pip install glob2
```

خطوات التنفيذ:

الخطوة 1: قم بتنزيل مجموعة البيانات من عنوان url هذا. تحتوي مجموعة البيانات على مجلدات الاختبار والتدريب والتحقق. سنستخدم مجموعات بيانات الاختبار والتدريب لتدريب نموذجنا. ثم سنتحقق من نموذجنا باستخدام مجموعة بيانات التحقق.

الخطوة 2: قم باستيراد جميع الوحدات الضرورية المتوفرةفي keras مثل ImageDataGenerator وModel وDense وFlatten والبقية. سننشئ كوداً عامًا مما يعني أنه يتعين علينا فقط تغيير اسم المكتبة، ثم سيعمل الكود تلقائيًا فيما يتعلق بـ VGG16 وVGG19 وresnet50.

```
from keras.models import Model
from keras.layers import Flatten,Dense
from keras.applications.vgg16 import VGG16
import matplotlib.pyplot as plot
from glob import glob
```

الخطوة 3: بعد ذلك، سنقدم حجم صورتنا، أي 224 × 224، وهذا حجم ثابت لمعمارية VGG16. 3 يدل على أننا نعمل مع نوع RGB من الصور. ثم سنوفر مسار بيانات التدريب والاختبار.

```
IMAGESHAPE = [224, 224, 3]
training_data = 'chest_xray/train'
testing_data = 'chest_xray/test'
```

الخطوة 4: الآن، سنقوم باستيراد نموذج VGG16 الخاص بنا. أثناء الاستيراد، سنستخدم أوزان imageNet & include_top = False تشير إلى أننا لا نريد تصنيف 1000 فئة مختلفة موجودةفي imageNet، فإن مشكلتنا تدور حول فئتين من الالتهاب الرئوي وعادي، ولهذا السبب نحن فقط نتخلى عن الطبقتين الأولى والأخيرة. سنقوم فقط بتصميم طبقاتنا الخاصة وإضافتها إلى VGG16.

```
vgg_model = VGG16(input_shape=IMAGESHAPE, weights='imagenet', include_top=False)
```

الخطوة 5: بعد استيراد نموذج VGG16، يتعين علينا إجراء هذا التغيير المهم. باستخدام التكرار الحلقي for على جميع الطبقات وتعيين trainable = False، بحيث لا يتم تدريب جميع الطبقات.

```
for each_layer in vgg_model.layers:
    each_layer.trainable = False
```

الخطوة 6: سنحاول معرفة عدد الفئات الموجودةفي مجموعة بيانات التدريب الخاصة بنا لفهم عدد تسميات الإخراج التي يجب أن تكون لدينا.

```
classes = glob('chest_xray/train/*')
```

الخطوة 7: نظرًا لأننا قمنا بحذف العمودين الأول والأخيرفي الخطوة السابقة، سنقوم فقط بإنشاء طبقة مسطحة flattened layer وأخيرًا نضيف الطبقة الأخيرة مع دالة تنشيط softmax. تشير len(classes) إلى عدد الفئات الموجودةفي طبقة الإخراج الخاصة بنا.

```
flatten_layer = Flatten()(vgg_model.output)
prediction = Dense(len(classes), activation='softmax')(flatten_layer)
```

الخطوة 8: سنقوم الآن بدمج ناتج VGG والتنبؤ، كل هذا معًا سينشئ نموذجًا. عندما نتحقق من ملخص النموذج، يمكننا أن نلاحظ أن الطبقة الأخيرة بها فئتان فقط.

```
final_model = Model(inputs=vgg_model.input, outputs=prediction)
final_model.summary()
```

الخطوة 9: الآن سنقوم بتجميع compile نموذجنا باستخدام مُحسِّن آدم ومقياس التحسين كدقة.

```
final_model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

الخطوة 10: بعد تجميع النموذج، يتعين علينا استيراد مجموعة البيانات الخاصة بنا إلى Keras باستخدام ImageDataGeneratorفي Keras. لإنشاء ميزات إضافية، نستخدم مقاييس مثل

إعادة القياس rescale، القص shear_range، التكبير zoom_range، هذه ستساعدنافي مرحلتي التدريب والاختبار.

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
testing_datagen = ImageDataGenerator(rescale =1. / 255)
```

الخطوة 11: الآن سنقوم بإدخال الصور باستخدام دالة flow_from_directory(). تأكد من أنه يتعين علينا هنا تمرير نفس حجم الصورة كما بدأنا سابقًا. يشير حجم الدفعة 4 إلى أنه سيتم تقديم 4 صور مرة واحدة للتدريب. Class_mode فئوية أي إما ذات الرئة Pneumonia أو غير ذات الرئة Not Pneumonia.

```
training_set = train_datagen.flow_from_directory('chest_xray/train',
                                                 target_size = (224,
224),
                                                 batch_size = 4,
                                                 class_mode =
'categorical')
```

الخطوة 12: وبالمثل، سنفعل نفس الشيء لمجموعة بيانات الاختبار ما فعلناه لمجموعة بيانات التدريب.

```
test_set = testing_datagen.flow_from_directory('chest_xray/test',
                                               target_size = (224,
224),
                                               batch_size = 4,
                                               class_mode =
'categorical')
```

الخطوة 13: أخيرًا، نقوم بتركيب النموذج باستخدام دالة fit_generator() وتمرير جميع التفاصيل اللازمة فيما يتعلق بمجموعة بيانات التدريب والاختبار لدينا كوسيطات. سيستغرق هذا بعض الوقت للتنفيذ.

```
fitted_model = final_model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=5,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

الخطوة 14: إنشاء ملف نموذج وتخزين هذا النموذج. حتى لا نحتاج إلى تدريب النموذجفي كل مرة نقدم فيها مدخلات.

```
final_model.save('our_model.h5')
```

الخطوة 15: قم بتحميل النموذج الذي أنشأناه. الآن اقرأ الصورة والمعالجة المسبقة للصورة أخيرًا نتحقق من المخرجات التي يعطيها نموذجنا باستخدام دالة() model.predict .

```
from keras_preprocessing import image
```

```
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
import numpy as np
model=load_model('our_model.h5') #Loading our model
img=image.load img('D:/Semester -
6/PneumoniaGFG/chest_xray/val/PNEUMONIA/person1954_bacteria_4886.jpeg
',target_size=(224,224))
imagee=image.img_to_array(img) #Converting the X-Ray into pixels
imagee=np.expand_dims(imagee, axis=0)
img_data=preprocess_input(imagee)
prediction=model.predict(img_data)
if prediction[0][0]>prediction[0][1]:  #Printing the prediction of
model.
    print('Person is safe.')
else:
    print('Person is affected with Pneumonia.')
print(f'Predictions: {prediction}')
```

التنفيذ الكامل

Pneumonia.py:

```
from keras.models import Model
from keras.layers import Flatten,Dense
from keras.applications.vgg16 import VGG16  #Import all the necessary
modules
import matplotlib.pyplot as plot
from glob import glob

IMAGESHAPE = [224, 224, 3] #Provide image size as 224 x 224 this is a
fixed-size for VGG16 architecture
vgg_model = VGG16(input_shape=IMAGESHAPE, weights='imagenet',
include_top=False)
#3 signifies that we are working with RGB type of images.
training_data = 'chest_xray/train'
testing_data = 'chest_xray/test' #Give our training and testing path

for each_layer in vgg_model.layers:
    each_layer.trainable = False #Set the trainable as False, So that
all the layers would not be trained.
classes = glob('chest_xray/train/*') #Finding how many classes
present in our train dataset.
flatten layer = Flatten()(vgg model.output)
prediction = Dense(len(classes), activation='softmax')(flatten_layer)
final_model = Model(inputs=vgg_model.input, outputs=prediction)
#Combine the VGG output and prediction , this all together will
create a model.
final model.summary() #Displaying the summary
final_model.compile( #Compiling our model using adam optimizer and
optimization metric as accuracy.
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255, #importing our
dataset to keras using ImageDataGenerator in keras.
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal flip = True)
testing_datagen = ImageDataGenerator(rescale =1. / 255)
training_set = train_datagen.flow_from_directory('chest_xray/train',
#inserting the images.
```

```
                                                    target_size = (224,
224),
                                                    batch_size = 4,
                                                    class_mode =
'categorical')
test_set = testing_datagen.flow_from_directory('chest_xray/test',
                                                   target_size = (224,
224),
                                                   batch_size = 4,
                                                   class_mode =
'categorical')
fitted_model = final_model.fit_generator( #Fitting the model.
  training set,
  validation_data=test_set,
  epochs=5,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
plot.plot(fitted_model.history['loss'], label='training loss')
#Plotting the accuracies
plot.plot(fitted_model.history['val_loss'], label='validation loss')
plot.legend()
plot.show()
plot.savefig('LossVal_loss')
plot.plot(fitted model.history['acc'], label='training accuracy')
plot.plot(fitted_model.history['val_acc'], label='validation
accuracy')
plot.legend()
plot.show()
plot.savefig('AccVal_acc')
final_model.save('our_model.h5') #Saving the model file.
```

Test.py:

```
from keras_preprocessing import image
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
import numpy as np
model=load_model('our_model.h5') #Loading our model
img=image.load_img('D:/Semester -
6/PneumoniaGFG/chest_xray/val/PNEUMONIA/person1954_bacteria_4886.jpeg
',target_size=(224,224))
imagee=image.img to array(img) #Converting the X-Ray into pixels
imagee=np.expand_dims(imagee, axis=0)
img_data=preprocess_input(imagee)
prediction=model.predict(img_data)
if prediction[0][0]>prediction[0][1]:  #Printing the prediction of
model.
    print('Person is safe.')
else:
    print('Person is affected with Pneumonia.')
print(f'Predictions: {prediction}')
```

المخرجات:

يتم عرض النتائج في الفيديو أدناه:

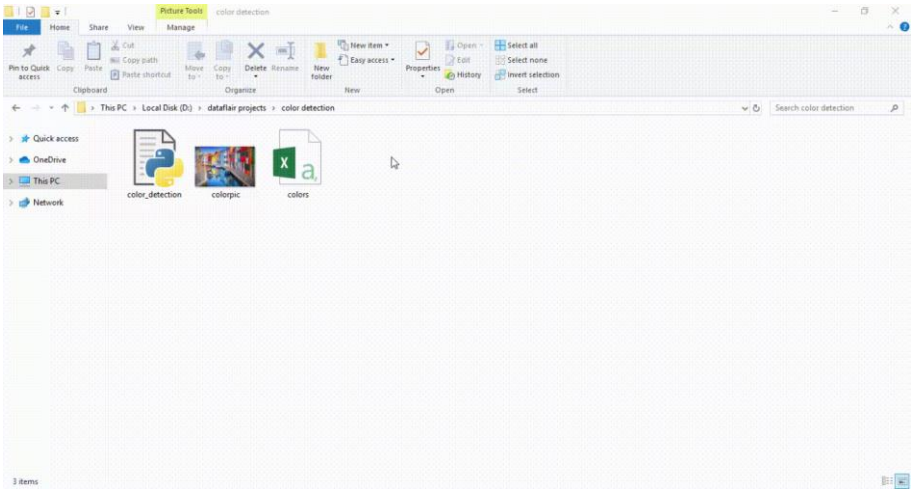
https://media.geeksforgeeks.org/wp-content/uploads/20220222002658/Output.mp4?_=1

11) اكتشاف اللون باستخدام OpenCV وPandas Color Detection using Pandas & OpenCV

سيكون مشروع اليوم مثيرًا وممتعًافي بنائه. سنعمل مع الألوان وستتعرف على العديد من المفاهيم خلال هذا المشروع. يُعد اكتشاف الألوان Color detection ضروريًا للتعرف على الكائنات، كما يتم استخدامه كأداةفي العديد من تطبيقات تحرير الصور والرسم.

ما هو اكتشاف اللون؟

اكتشاف اللون هو عملية الكشف عن اسم أي لون. بسيط أليس كذلك؟ حسنًا، هذه مهمة سهلة للغاية بالنسبة للبشر ولكن بالنسبة لأجهزة الكمبيوتر، فهي ليست مباشرة. تعمل عيون وأدمغة الإنسان معًا لترجمة الضوء إلى لون. تنقل مستقبلات الضوء الموجودةفي أعيننا الإشارة إلى الدماغ. ثم يتعرف دماغنا على اللون. منذ الطفولة، قمنا بتعيين أضواء معينة بأسماء ألوانها. سنستخدم نفس الإستراتيجية إلى حد ما لاكتشاف أسماء الألوان.



حول مشروع بايثون

في مشروع بايثون لاكتشاف الألوان هذا، سننشئ تطبيقًا يمكنك من خلاله الحصول تلقائيًا على اسم اللون بالنقر فوقها. لهذا، سيكون لدينا ملف بيانات يحتوي على اسم اللون وقيمه. ثم نحسب المسافة من كل لون ونجد الأقصر.

مجموعة البيانات

تتكون الألوان من 3 ألوان أساسية؛ الأحمر والأخضر والأزرق.في أجهزة الكمبيوتر، نحدد كل قيمة لونفي نطاق من 0 إلى 255. إذن ما هو عدد الطرق التي يمكننا من خلالها تحديد اللون؟ الجواب

هو 256 * 256 * 256 = 16.581.375. هناك ما يقرب من 16.5 مليون طريقة مختلفة لتمثيل اللون.في مجموعة البيانات الخاصة بنا، نحتاج إلى تعيين قيم كل لون بأسمائها المقابلة. لكن لا تقلق، لسنا بحاجة إلى تعيين جميع القيم. سنستخدم مجموعة بيانات تحتوي على قيم RGB مع أسمائها المقابلة. تم أخذ ملف CSV لمجموعة البيانات الخاصة بنا من هذا الرابط:

Colors Dataset

يشتمل ملف colours.csv على 865 اسمًا لونيًا مع قيم RGB وhex.

المتطلبات الأساسية

قبل البدءفي مشروع بايثون هذا باستخدام الكود المصدري، يجب أن تكون على دراية بمكتبة الرؤية الحاسوبيةفي بايثون وهي OpenCV و Pandas.

OpenCV و Pandas و numpy هي حزم بايثون الضرورية لهذا المشروعفي بايثون. لتثبيتها، ما عليك سوى تشغيل أمر pip هذافي جهازك الطرفي:

```
pip install opencv-python numpy pandas
```

خطوات بناء مشروع في بايثون - اكتشاف اللون

فيما يلي خطوات إنشاء تطبيقفي بايثون يمكنه اكتشاف الألوان:

الخطوة 1:تنزيل وفك ضغط الملف المضغوط

Color Detection Zip File

يحتوي مجلد المشروع على 3 ملفات:

- Color_detection.py ـ كود المصدر الرئيسي لمشروعنا.
- Colorpic.jpg ـ صورة عينة للتجربة.
- Colors.csv ـ ملف يحتوي على مجموعة البيانات الخاصة بنا.

الخطوة 2: التقاط صورة من المستخدم

نحن نستخدم مكتبة argparse لإنشاء argument parser. يمكننا إعطاء مسار صورة مباشرة من موجه الأوامر:

```
import argparse

ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required=True, help="Image Path")
args = vars(ap.parse_args())
img_path = args['image']
#Reading image with opencv
img = cv2.imread(img_path)
```

الخطوة 3: قراءة ملف CSV مع pandas

تعد مكتبة pandas مفيدة للغاية عندما نحتاج إلى إجراء عمليات مختلفة على ملفات البيانات مثل CSV. يقرأ pd.read_csv() ملف CSV ويحملهفي pandas DataFrame. لقد قمنا بتعيين اسم لكل عمود للوصول إليه بسهولة.

```
#Reading csv file with pandas and giving names to each column
index=["color","color_name","hex","R","G","B"]
csv = pd.read_csv('colors.csv', names=index, header=None)
```

الخطوة 4: تعيين حدث mouse callback على النافذة

أولاً، أنشأنا نافذة تُعرض فيها صورة الإدخال. بعد ذلك، قمنا بتعيين دالة callback والتي سيتم استدعاؤها عند حدوث حدث الماوس.

```
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_function)
```

باستخدام هذه الأسطر، أطلقنا على نافذتنا اسم "image" وقمنا بتعيين دالة callback والتي ستستدعي draw_function() كلما حدث حدث الماوس.

الخطوة 5: إنشاء دالة draw_function

سيحسب قيم rgb للبكسل الذي نضغط عليه مرتين. معلمات الدالة لها اسم الحدث، إحداثيات (x,y) موضع الماوس، إلخ.في الدالة، نتحقق مما إذا كان الحدث قد تم النقر عليه مرتين ثم نحسب ونضبط قيم r,g,b جنبًا إلى جنب مع (x,y) مواضع الماوس.

```
def draw_function(event, x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b,g,r,xpos,ypos, clicked
        clicked = True
        xpos = x
        ypos = y
        b,g,r = img[y,x]
        b = int(b)
        g = int(g)
        r = int(r)
```

الخطوة 6: حساب المسافة للحصول على اسم اللون

لدينا قيم r,g,b. الآن، نحتاج إلى دالة أخرى ستعيد لنا اسم اللون من قيم RGB. للحصول على اسم اللون، نحسب المسافة (d) التي تخبرنا عن مدى قربنا من اللون واختيار واحد له مسافة دنيا.

يتم حساب مسافتنا بهذه الصيغة:

```
d = abs(Red – ithRedColor) + (Green – ithGreenColor) + (Blue – ithBlueColor)
def getColorName(R,G,B):
    minimum = 10000
    for i in range(len(csv)):
        d = abs(R- int(csv.loc[i,"R"])) + abs(G- int(csv.loc[i,"G"]))+
abs(B- int(csv.loc[i,"B"]))
        if(d<=minimum):
```

```
            minimum = d
            cname = csv.loc[i,"color_name"]
    return cname
```

الخطوة 7: عرض الصورة على النافذة

عند حدوث حدث نقر مزدوج، سيتم تحديث اسم اللون وقيم RGBفي النافذة.

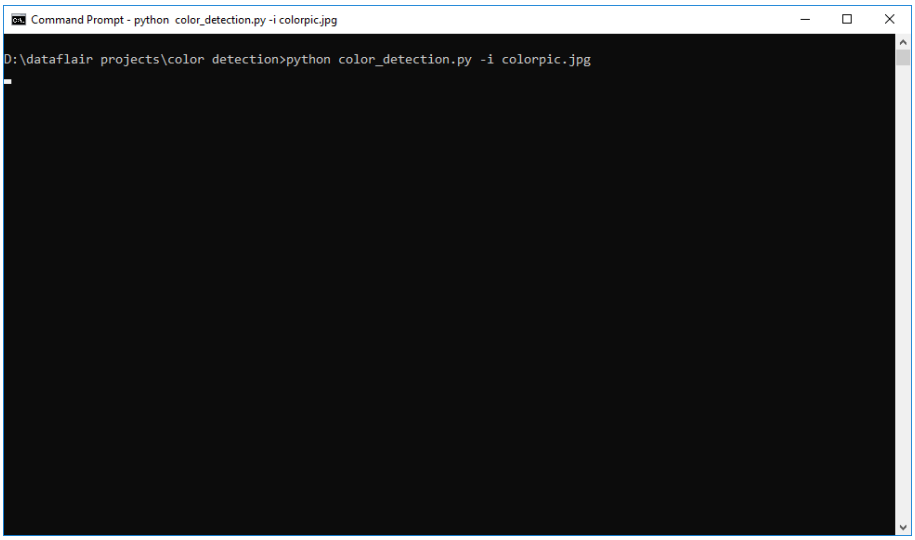
باستخدام دالة ()cv2.imshow، نرسم الصورة على النافذة. عندما ينقر المستخدم نقرًا مزدوجًا على النافذة، نرسم مستطيلًا ونحصل على اسم اللون لرسم نص على النافذة باستخدام دوال cv2.rectangle و ()cv2.putText.

```
while(1):
    cv2.imshow("image",img)
    if (clicked):
        #cv2.rectangle(image, startpoint, endpoint, color, thickness)
-1 thickness fills rectangle entirely
        cv2.rectangle(img,(20,20), (750,60), (b,g,r), -1)

        #Creating text string to display ( Color name and RGB values )
        text = getColorName(r,g,b) + ' R='+ str(r) + ' G='+ str(g) + '
B='+ str(b)

        #cv2.putText(img,text,start,font(0-7), fontScale, color,
thickness, lineType, (optional bottomLeft bool) )
        cv2.putText(img,
text,(50,50),2,0.8,(255,255,255),2,cv2.LINE_AA)
  #For very light colours we will display text in black colour
        if(r+g+b>=600):
            cv2.putText(img, text,(50,50),2,0.8,(0,0,0),2,cv2.LINE_AA)

        clicked=False

    #Break the loop when user hits 'esc' key
    if cv2.waitKey(20) & 0xFF ==27:
        break

cv2.destroyAllWindows()
```

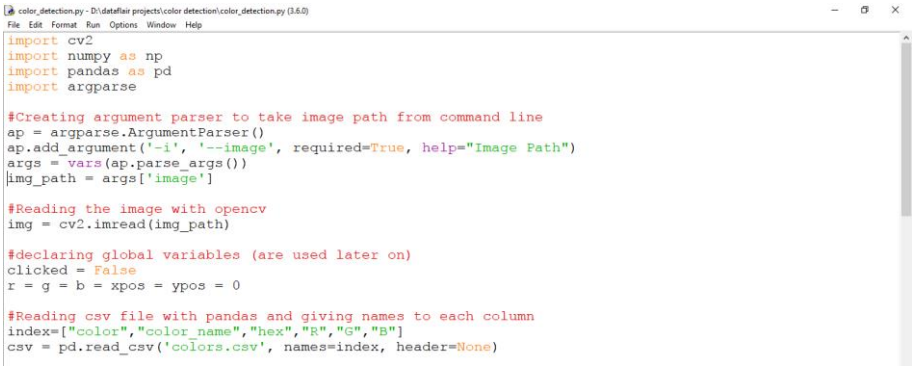
الخطوة 8: تشغيل ملف بايثون

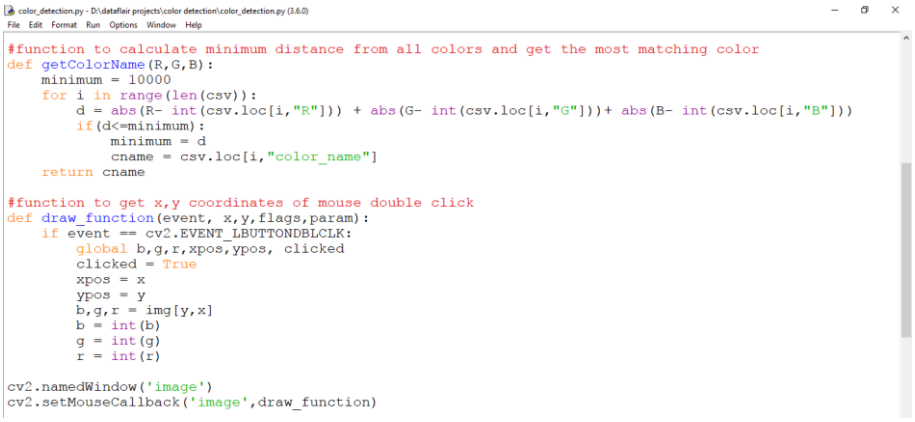
اكتمل الآن مشروع بايثون المبتدئ، يمكنك تشغيل ملف بايثون من موجه الأوامر. تأكد من إعطاء مسار صورة باستخدام الوسيطة "i-". إذا كانت الصورةفي دليل آخر، فأنت بحاجة إلى إعطاء المسار الكامل للصورة:

```
python color_detection.py -i <add your image path here>
```



لقطات الشاشة:





```
while(1):

    cv2.imshow("image",img)
    if (clicked):

        #cv2.rectangle(image, startpoint, endpoint, color, thickness)-1 fills entire rectangle
        cv2.rectangle(img,(20,20), (750,60), (b,g,r), -1)

        #Creating text string to display( Color name and RGB values )
        text = getColorName(r,g,b) + ' R='+ str(r) +  ' G='+ str(g) +  ' B='+ str(b)

        #cv2.putText(img,text,start,font(0-7),fontScale,color,thickness,lineType )
        cv2.putText(img, text,(50,50),2,0.8,(255,255,255),2,cv2.LINE_AA)

        #For very light colours we will display text in black colour
        if(r+g+b>=600):
            cv2.putText(img, text,(50,50),2,0.8,(0,0,0),2,cv2.LINE_AA)

        clicked=False

    #Break the loop when user hits 'esc' key
    if cv2.waitKey(20) & 0xFF ==27:
        break

cv2.destroyAllWindows()
```

المخرجات:

انقر نقرًا مزدوجًا فوق النافذة لمعرفة اسم لون البكسل.

الملخص

في مشروع بايثون هذا مع الكود المصدري، تعلمنا عن الألوان وكيف يمكننا استخراج قيم RGB الملونة واسم لون البكسل. لقد تعلمنا كيفية التعامل مع الأحداث مثل النقر المزدوج على النافذة وشاهدنا كيفية قراءة ملفات CSV مع pandas وتنفيذ العمليات على البيانات. يستخدم هذافي العديد من تطبيقات تحرير الصور والرسم.

12) كشف تصدعات الطريق باستخدام التعلم العميق Road Crack Detection using Deep Learning

أضرار التصدعات في الطريق

تحدث معظم حوادث السيارات بسبب اتخاذ السائقين قرارات سيئة، بمافي ذلك السرعة، وعدم الانصياع، والقيادة المشتتة للانتباه. ومع ذلك، فإن بعض الحوادث تحدث بدون خطأ من السائق. يمكن أن تؤدي حالة الطريق السيئة إلى وقوع حوادث تصيب السائقين والركاب والمشاة.

يمكن أن تؤدي ظروف الطريق السيئة إلى أكثر من مجرد ركوب وعر. يمكن أن تتدهور الطرق إلى المستوى الذي تصبح فيه خطرة. وهذا يشمل الحفر، وأسطح الطرق غير المستوية، والخرسانة المكسورة، وحديد التسليح المكشوف، والحفر، وشقوق الطرق. إذا اصطدم سائق بحفرة كبيرة، فقد ينفجر الإطار مما يتسببفي انحراف السيارة إلى حارة أخرى، والتصادم مع مركبة أخرى. يمكن أن تتسبب أسطح الطرق غير المستويةفي فقدان السائق السيطرة على سيارته، مما يؤدي إلى وقوع حادث أو حادث انقلاب يؤدي إلى إصابة السائق والركاب والمشاة.

هدف المشروع

من أجل زيادة سرعة الإصلاح، قمنا بإنشاء جهاز كشف شقوق الطريق والذي يمكن تثبيته على الطائرات بدون طيار. سيسمح ذلك بفحص الطريق بسرعة أعلى مما يسمح للسلطات المسؤولة بالحصول على البيانات بشكل أسرع وبالتالي زيادة الدقة.

استيراد المكتبات

```
from matplotlib import pyplot as plt
import cv2
import numpy as np
from PIL import Image
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, callbacks, optimizers
from sklearn.metrics import confusion_matrix, f1_score
from tensorflow import keras
import os, shutil
import random
from PIL import Image
```

مجموعة البيانات

تتكون مجموعة البيانات من 40 ألف صورة تم تصنيفها على أنها سلبية وإيجابية.

```
!wget "https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-
admin/Road_Crack_DCSV9HG.zip"
!unzip -qo "Road_Crack.zip"
```

تقسيم البيانات

```
data_dir = 'Road Crack/'

batch_size = 64
# image_size = (32, 32)
image_size = (28, 28)

print("Training set")
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
                data_dir,
                validation_split=0.2,
                subset="training",
                color_mode="grayscale",
                image_size=image_size,
                seed=113,
                shuffle=True,
                batch_size=batch_size
            )

print("Validation set")
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
                data_dir,
                validation_split=0.2,
                subset="validation",
                color_mode="grayscale",
                image_size=image_size,
                seed=113,
                shuffle=True,
                batch_size=batch_size
            )
```

المخرجات:

```
Training set
Found 40000 files belonging to 2 classes.
Using 32000 files for training.
Validation set
Found 40000 files belonging to 2 classes.
Using 8000 files for validation.
```

كما ترى فإن مجموعة البيانات تحتوي على 40000 صورة مقطعة على موجب وسلبي مع سلبي ليس به تشققات وموجب يحتوي على صدع.

```
Xtrain = np.empty((0,*image_size,1))
ytrain = np.empty((0,1))

for x in train_ds.enumerate():
    for y in x[1][0]:
        Xtrain = np.append(Xtrain, np.expand_dims(np.array(y),0), axis = 0)
    #print(Xtrain.shape)
    ytrain = np.append(ytrain, np.array(x[1][1]))
    #print(ytrain.shape)

Xtrain.shape, ytrain.shape
```

```
((32000, 28, 28, 1), (32000,))
```

```
class_names = train_ds.class_names
print(class_names)
```

```
['Negative', 'Positive']
```

```
Xval = np.empty((0,*image_size,1))
yval = np.empty((0,1))

for x in val_ds.enumerate():
    for y in x[1][0]:
        Xval = np.append(Xval, np.expand_dims(np.array(y),0), axis = 0)
    #print(Xtrain.shape)
    yval = np.append(yval, np.array(x[1][1]))
    #print(ytrain.shape)

Xval.shape, yval.shape
```

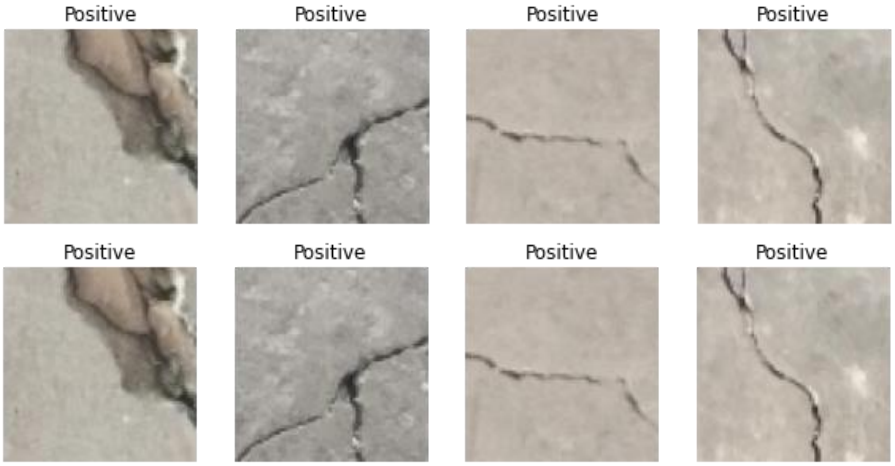
```
((8000, 28, 28, 1), (8000,))
```

```
print("Number of samples - ")
for i in range(len(class_names)):
    print(class_names[i], "-", yval.tolist().count(float(i)))
```

```
Number of samples -
Negative - 3999
Positive - 4001
```

رسم البيانات

```
num_samples = 4    # the number of samples to be displayed in each class

for x in class_names:
    plt.figure(figsize=(10, 10))

    filenames = os.listdir(data_dir + x)

    for i in range(num_samples):
        ax = plt.subplot(1, num_samples, i + 1)
        img = Image.open(os.path.join(data_dir, x, filenames[i]))
        plt.imshow(img)
        plt.title(x)
        plt.axis("off")
```



```
Xtrain = Xtrain/255
Xval = Xval/255
```

بناء النموذج

```
model = keras.models.Sequential([
    layers.Conv2D(8, 3, activation='relu', input_shape=Xtrain[0].shape),
    layers.MaxPool2D(pool_size=(2, 2)),

    layers.Conv2D(16, 3, activation='relu'),
    layers.MaxPool2D(pool_size=(2, 2)),

    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPool2D(pool_size=(2, 2)),

    layers.Flatten(),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

cb = [callbacks.EarlyStopping(monitor = 'val_loss', patience = 5,
restore_best_weights = True)]
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 8)         80
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 8)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 16)        1168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 16)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 32)          4640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 32)                1056
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 6,977
Trainable params: 6,977
Non-trainable params: 0
_________________________________________________________________
```

يحتوي النموذج على معلمات 7 الاف فقط وهي منخفضة جداً ولكنها لا تزال تقوم بعمل جيد للغايةفي تحديد الشقوق على الطريق.

تطبيق النموذج وتجميعه

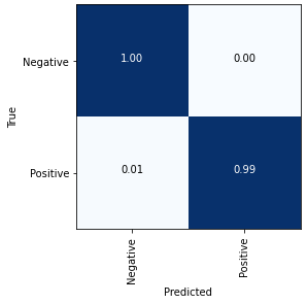
```
model.compile(loss=keras.losses.BinaryCrossentropy(),
optimizer=optimizers.Adam(0.0001), metrics=['accuracy'])

history = model.fit(Xtrain, ytrain, validation_data=(Xval, yval), epochs=300,
callbacks=cb)
```

```
Epoch 95/300
1000/1000 [==============================] - 2s 2ms/step - loss: 0.0212 - accuracy: 0.9942 - val_loss: 0.0368 - val_accuracy: 0.9901
Epoch 96/300
1000/1000 [==============================] - 2s 2ms/step - loss: 0.0214 - accuracy: 0.9941 - val_loss: 0.0265 - val_accuracy: 0.9930
Epoch 97/300
1000/1000 [==============================] - 2s 2ms/step - loss: 0.0214 - accuracy: 0.9940 - val_loss: 0.0269 - val_accuracy: 0.9929
Epoch 98/300
1000/1000 [==============================] - 2s 2ms/step - loss: 0.0210 - accuracy: 0.9941 - val_loss: 0.0284 - val_accuracy: 0.9918
Epoch 99/300
1000/1000 [==============================] - 2s 2ms/step - loss: 0.0210 - accuracy: 0.9944 - val_loss: 0.0260 - val_accuracy: 0.9930
```

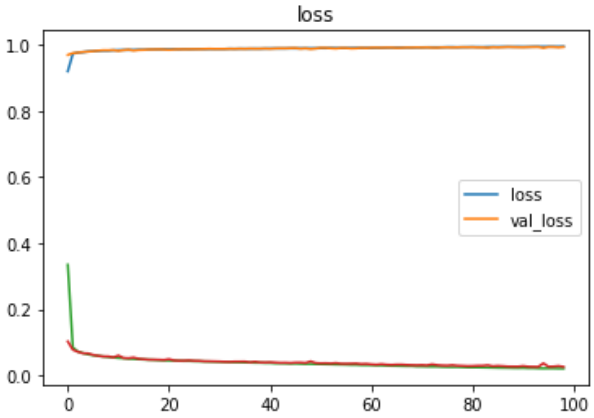
النتائج

ما هي أفضل طريقة لإلقاء نظرة على النتيجة باستخدام الرسوم البيانية. تسهيل الأرقام وتحويلها إلى رسوم بيانية.

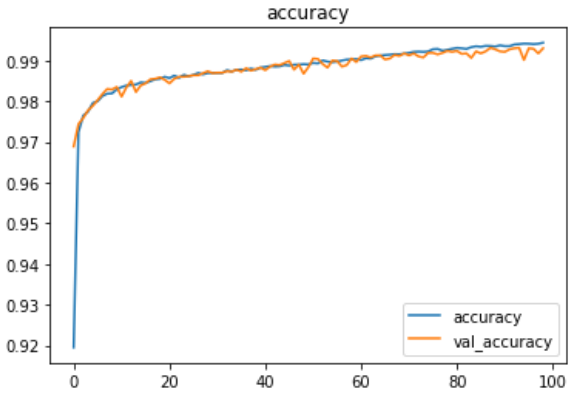
```
model.evaluate(Xval, yval)
ypred = (model.predict(Xval)>0.5).astype('int')
cm = confusion_matrix(yval, ypred)

cm = cm.astype('int') / cm.sum(axis=1)[:, np.newaxis]

fig = plt.figure(figsize = (4, 4))
ax = fig.add_subplot(111)

for i in range(cm.shape[1]):
    for j in range(cm.shape[0]):
        if cm[i,j] > 0.8:
            clr = "white"
        else:
            clr = "black"
        ax.text(j, i, format(cm[i, j], '.2f'), horizontalalignment="center",
color=clr)

_ = ax.imshow(cm, cmap=plt.cm.Blues)
ax.set_xticks(range(len(class_names)))
ax.set_yticks(range(len(class_names)))
ax.set_xticklabels(class_names, rotation = 90)
ax.set_yticklabels(class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```




```
def plot(history, variable1, variable2):
    plt.plot(range(len(history[variable1])), history[variable1])
    plt.plot(range(len(history[variable2])), history[variable2])
    plt.legend([variable1, variable2])
    plt.title(variable1)


plot(history.history, "loss", 'val_loss')
```




```
plot(history.history, "accuracy", 'val_accuracy')
```

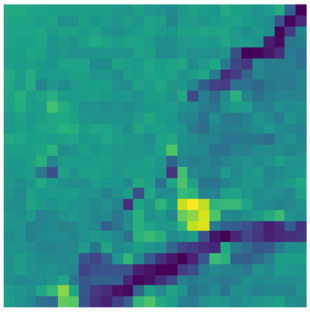


التنبؤ

كما ترى، تم توقع الطريق الذي لا يوجد به صدع على أنه سلبي.

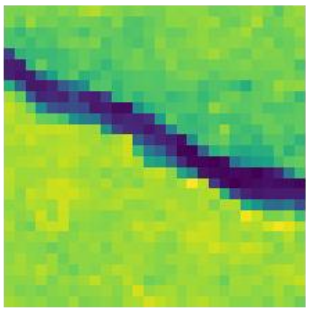
```
# pick random test data sample from one batch
x = random.randint(0, 32 - 1) # default batch size is 32

for i in val_ds.as_numpy_iterator():
    img, label = i
    plt.axis('off')   # remove axes
    plt.imshow(img[x])    # shape from (64, 64, 64, 1) --> (64, 64, 1)
    output = model.predict(np.expand_dims(img[x],0))[0][0]    # getting
output; input shape (64, 64, 3) --> (1, 64, 64, 1)
    pred = (output > 0.5).astype('int')
    print("Predicted: ", class_names[pred], '(', output, '-->', pred, ')')
# Picking the label from class_names base don the model output
    print("True: ", class_names[label[x]])
    break
```

```
Predicted:  Positive ( 1.0 --> 1 )
True:  Positive
```



```
x = random.randint(0, 32 - 1) # default batch size is 32

for i in val_ds.as_numpy_iterator():
    img, label = i
    plt.axis('off')   # remove axes
    plt.imshow(img[x])    # shape from (64, 64, 64, 1) --> (64, 64, 1)
    output = model.predict(np.expand_dims(img[x],0))[0][0]    # getting
output; input shape (64, 64, 3) --> (1, 64, 64, 1)
    pred = (output > 0.5).astype('int')
    print("Predicted: ", class_names[pred], '(', output, '-->', pred, ')')
# Picking the label from class_names base don the model output
    print("True: ", class_names[label[x]])
    break
```

```
Predicted:  Positive ( 1.0 --> 1 )
True:  Positive
```



حفظ النموذج

```
model.save("road_crack.h5")
```

رابط الكود: انقر هنا

13) التنبؤ بأمراض القلب والأوعية الدموية باستخدام التعلم العميق Cardiovascular Diseases Prediction using Deep Learning

امراض القلب والاوعية الدموية Cardiovascular Diseases(CVD) هي السبب الأكثر شيوعًا للوفيات على مستوى العالم، حيث تودي بحياة ما يقدر بنحو 17.9 مليون شخص كل عام، وهو ما يمثل 31 ٪ من جميع الوفياتفي جميع أنحاء العالم. فشل القلب Heart failure هو حدث شائع تسببه أمراض القلب والأوعية الدموية.

يتميز بعدم قدرة القلب على ضخ كمية كافية من الدم إلى الجسم. بدون تدفق الدم الكافي، تتعطل جميع وظائف الجسم الرئيسية. فشل القلب هو حالة أو مجموعة من الأعراض التي تضعف القلب.

استيراد المكتبات:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import seaborn as sns
from tensorflow.keras.layers import Dense, BatchNormalization, Dropout, LSTM
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import callbacks
from sklearn.metrics import precision_score, recall_score, confusion_matrix,
classification_report, accuracy_score, f1_score
```

تحميل البيانات

```
#Loading data
data = pd.read_csv("https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-
admin/heart_failure_clinical_records_dataset_lsgYy2P.csv")
data.head()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	t
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	

```
data.info()
```

حول البيانات:

Age: عمر المريض.

anaemia: إذا كان مستوى الهيموجلوبين لدى المريض أقل من المعدل الطبيعي.

creatinine phosphokinase: مستوى فوسفوكيناز الكرياتينفي الدمفي ميكروغرام / لتر.

diabetes: إذا كان المريض يعاني من مرض السكري.

ejection fraction: الكسر القذفي هو قياس كمية الدم التي يضخها البطين الأيسر مع كل انقباض

high_blood_pressure: إذا كان المريض يعاني من ارتفاع ضغط الدم.

platelets: تعداد الصفيحاتفي الدم بالكيلوغرام / مل.

serum_creatinine: مستوى الكرياتينينفي الدم بالملجم / ديسيلتر.

serum_sodium: مستوى الصوديومفي الدم بالملي مكافئ / لتر.

sex: جنس المريض.

smoking: إذا كان المريض يدخن بنشاط أو كان يدخنفي الماضي.

time: هو موعد زيارة المريض لمتابعة المرضفي شهور.

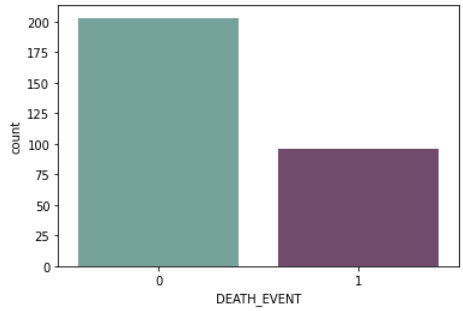
DEATH_EVENT: إذا توفي المريض خلال فترة المتابعة.

تحليل البيانات

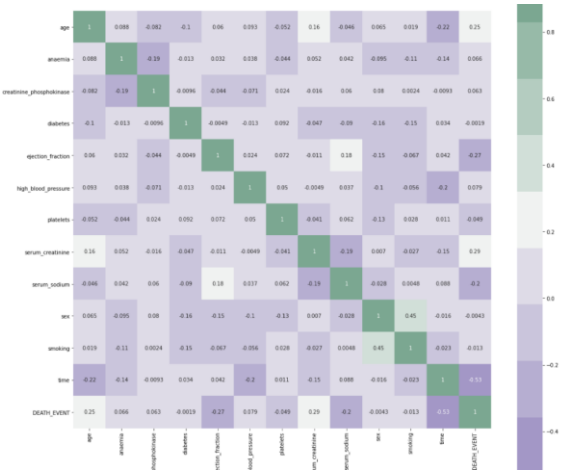
خطواتفي تحليل البيانات والتصوير:

نبدأ تحليلنا من خلال رسم مخطط عد لسمة targer. مصفوفة الارتباط corelation matrix من السمات المختلفة لفحص أهمية الميزة.

```
#first of all let us evaluate the target and find out if our data is
imbalanced or not
cols= ["#6daa9f","#774571"]
sns.countplot(x= data["DEATH_EVENT"], palette= cols)
```




```
#Examaning a corelation matrix of all the features
cmap = sns.diverging_palette(275,150,  s=40, l=65, n=9)
corrmat = data.corr()
plt.subplots(figsize=(18,18))
sns.heatmap(corrmat,cmap= cmap,annot=True, square=True);
```



النقاط المهمة:

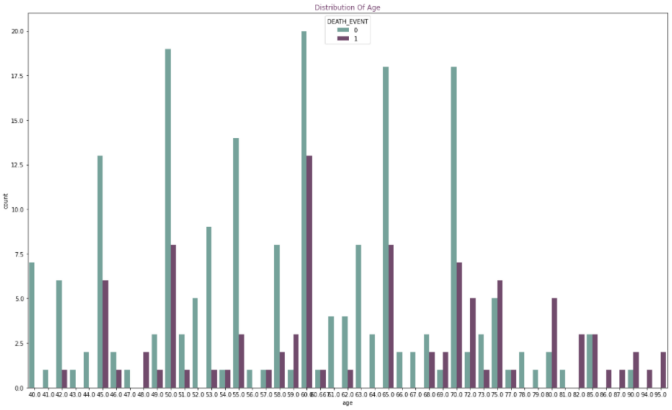
يعد وقت زيارة متابعة المريض للمرض أمرًا بالغ الأهمية لأن التشخيص الأولي لمشكلة القلب والأوعية الدموية والعلاج يقلل من فرص حدوث أي وفاة. إنها تحمل وعلاقة عكسية.

الكسر القذفي هو ثاني أهم ميزة. إنه متوقع تمامًا لأنهفي الأساس كفاءة القلب.

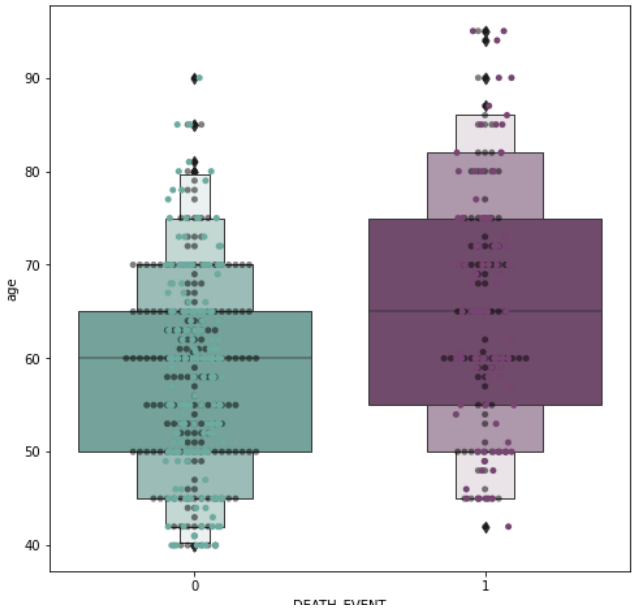
عمر المريض هو ثالث أكثر السمات ارتباطًا. من الواضح أن أداء القلب يتدهور مع تقدم العمر

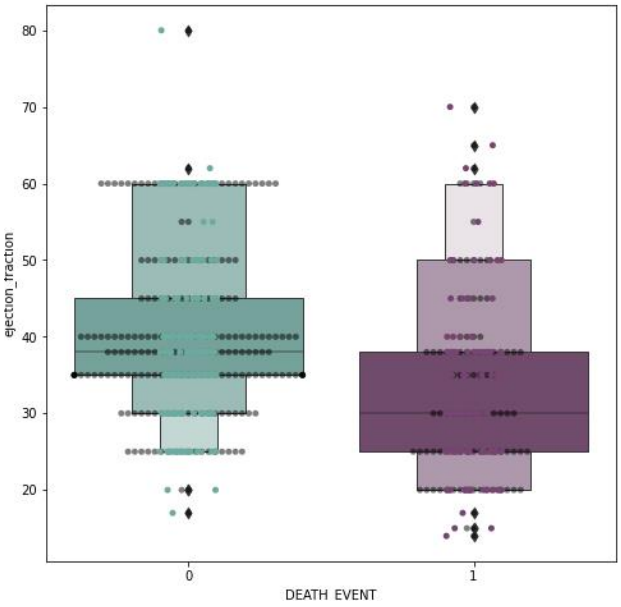
بعد ذلك، سوف نفحص رسم عد للعمر.

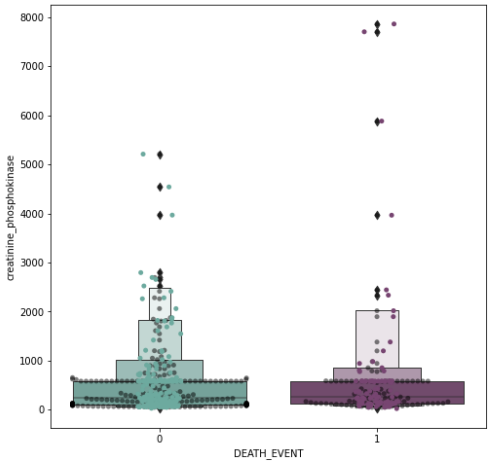
```
#Evauating age distrivution
plt.figure(figsize=(20,12))
#colours =["#774571","#b398af","#f1f1f1" ,"#afcdc7", "#6daa9f"]
Days_of_week=sns.countplot(x=data['age'],data=data, hue
="DEATH_EVENT",palette = cols)
Days_of_week.set_title("Distribution Of Age", color="#774571")
```

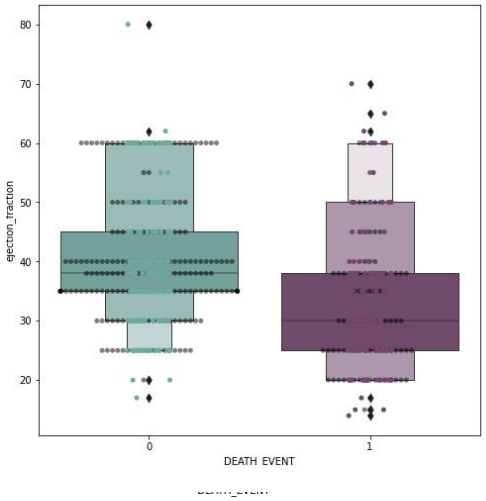


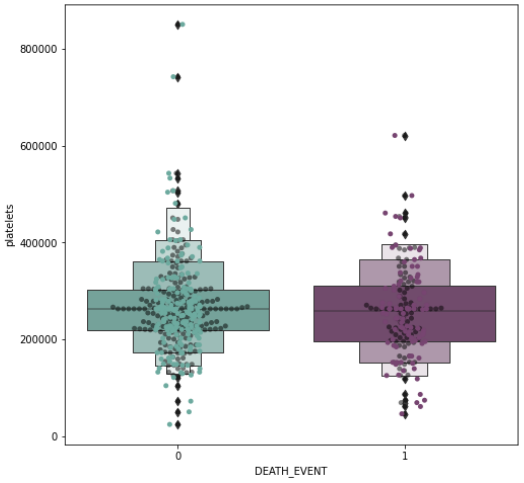
```
# Boxen and swarm plot of some non binary features.
feature =
["age","creatinine_phosphokinase","ejection_fraction","platelets","serum_crea
tinine","serum_sodium", "time"]
for i in feature:
    plt.figure(figsize=(8,8))
    sns.swarmplot(x=data["DEATH_EVENT"], y=data[i], color="black", alpha=0.5)
    sns.boxenplot(x=data["DEATH_EVENT"], y=data[i], palette=cols)
    sns.stripplot(x=data["DEATH_EVENT"], y=data[i], palette=cols)
    plt.show()
```

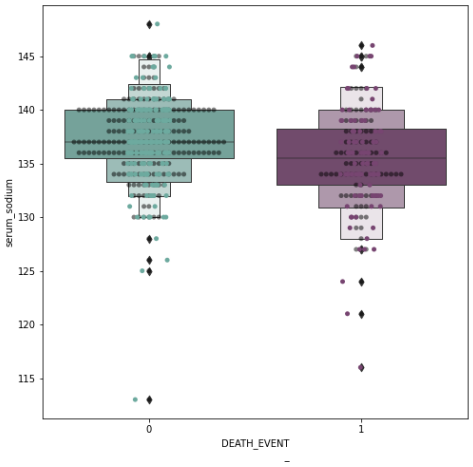


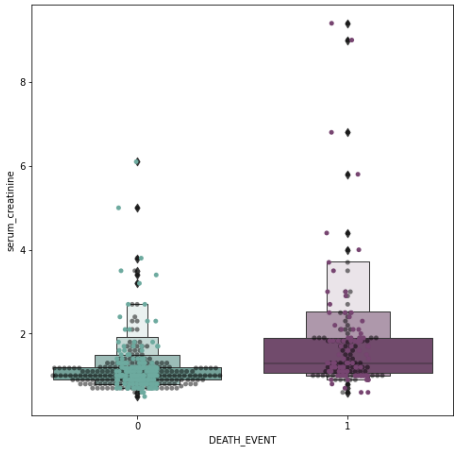



creatinine_phosphokinase
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
DEATH_EVENT


ejection_fraction
80
70
60
50
40
30
20
0
1
DEATH_EVENT


platelets
800000
600000
400000
200000
0
0
1
DEATH_EVENT

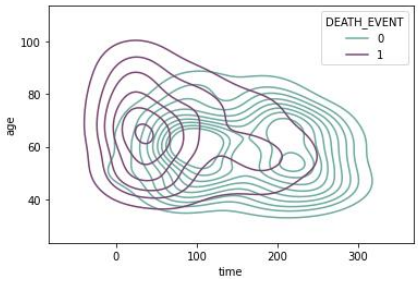




لقد رصدت القيم المتطرفة outliersفي مجموعة البيانات الخاصة بنا. لم أقم بإزالتها بعد لأنها قد تؤدي إلى فرط التعلم overfitting. على الرغم من أننا قد ننتهي بإحصائيات أفضل.في هذه الحالة، مع البيانات الطبية، قد تكون القيم المتطرفة عاملاً حاسمًا مهمًا.

بعد ذلك، نفحص مخطط kde للوقت والعمر لأن كلاهما ميزات مهمة.

```
sns.kdeplot(x=data["time"], y=data["age"], hue =data["DEATH_EVENT"],
palette=cols)
```



```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	299.0	60.833893	11.894809	40.0	51.0	60.0	70.0	95.0
anaemia	299.0	0.431438	0.496107	0.0	0.0	0.0	1.0	1.0
creatinine_phosphokinase	299.0	581.839465	970.287881	23.0	116.5	250.0	582.0	7861.0
diabetes	299.0	0.418060	0.494067	0.0	0.0	0.0	1.0	1.0
ejection_fraction	299.0	38.083612	11.834841	14.0	30.0	38.0	45.0	80.0
high_blood_pressure	299.0	0.351171	0.478136	0.0	0.0	0.0	1.0	1.0
platelets	299.0	263358.029264	97804.236869	25100.0	212500.0	262000.0	303500.0	850000.0
serum_creatinine	299.0	1.393880	1.034510	0.5	0.9	1.1	1.4	9.4
serum_sodium	299.0	136.625418	4.412477	113.0	134.0	137.0	140.0	148.0
sex	299.0	0.648829	0.478136	0.0	0.0	1.0	1.0	1.0
smoking	299.0	0.321070	0.467670	0.0	0.0	0.0	1.0	1.0
time	299.0	130.260870	77.614208	4.0	73.0	115.0	203.0	285.0
DEATH_EVENT	299.0	0.321070	0.467670	0.0	0.0	0.0	1.0	1.0

معالجة البيانات

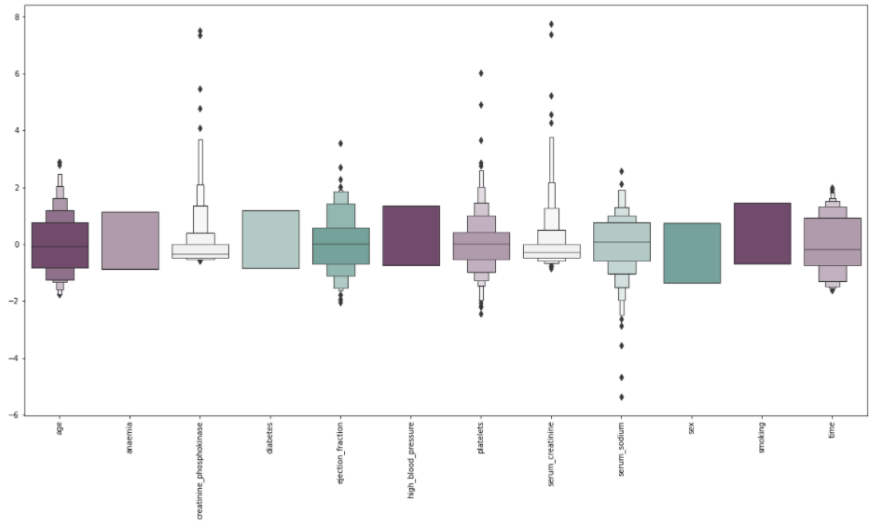
الخطوات المتبعةفي المعالجة المسبقة للبيانات:

- إسقاط القيم المتطرفة بناءً على تحليل البيانات.
- تعيين قيم للمعالم كـ X والهدف كـ y.
- إجراء تحجيم الميزات.
- تقسيم الى مجموعات تدريب واختبار.

```
#assigning values to features as X and target as y
X=data.drop(["DEATH_EVENT"],axis=1)
y=data["DEATH_EVENT"]

#Set up a standard scaler for the features
col_names = list(X.columns)
s_scaler = preprocessing.StandardScaler()
X_df= s_scaler.fit_transform(X)
X_df = pd.DataFrame(X_df, columns=col_names)
X_df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	299.0	5.703353e-16	1.001676	-1.754448	-0.828124	-0.070223	0.771889	2.877170
anaemia	299.0	1.009969e-16	1.001676	-0.871105	-0.871105	-0.871105	1.147968	1.147968
creatinine_phosphokinase	299.0	0.000000e+00	1.001676	-0.576918	-0.480393	-0.342574	0.000166	7.514640
diabetes	299.0	9.060014e-17	1.001676	-0.847579	-0.847579	-0.847579	1.179830	1.179830
ejection_fraction	299.0	-3.267546e-17	1.001676	-2.038387	-0.684180	-0.007077	0.585389	3.547716
high_blood_pressure	299.0	0.000000e+00	1.001676	-0.735688	-0.735688	-0.735688	1.359272	1.359272
platelets	299.0	7.723291e-17	1.001676	-2.440155	-0.520870	-0.013908	0.411120	6.008180
serum_creatinine	299.0	1.425838e-16	1.001676	-0.865509	-0.478205	-0.284552	0.005926	7.752020
serum_sodium	299.0	-8.673849e-16	1.001676	-5.363206	-0.595996	0.085034	0.766064	2.582144
sex	299.0	-8.911489e-18	1.001676	-1.359272	-1.359272	0.735688	0.735688	0.735688
smoking	299.0	-1.188199e-17	1.001676	-0.687682	-0.687682	-0.687682	1.454161	1.454161
time	299.0	-1.901118e-16	1.001676	-1.629502	-0.739000	-0.196954	0.938759	1.997038

```
#Looking at the scaled features
colours =["#774571","#b398af","#f1f1f1" ,"#afcdc7", "#6daa9f"]
plt.figure(figsize=(20,10))
sns.boxenplot(data = X_df,palette = colours)
plt.xticks(rotation=90)
plt.show()
```




```
#spliting test and training sets
X_train, X_test, y_train,y_test =
train_test_split(X_df,y,test_size=0.25,random_state=7)
```

بناء النموذج

في هذا المشروع، نبني شبكة عصبية اصطناعية ANN.

يتم تضمين الخطوات التاليةفي بناء النموذج:

- بدء تشغيل ANN.
- التعريف بإضافة طبقات.
- تجميع ANN.
- تدريب ANN.

```
early_stopping = callbacks.EarlyStopping(
    min_delta=0.001, # minimium amount of change to count as an improvement
    patience=20, # how many epochs to wait before stopping
    restore_best_weights=True)

# Initialising the NN
model = Sequential()
```

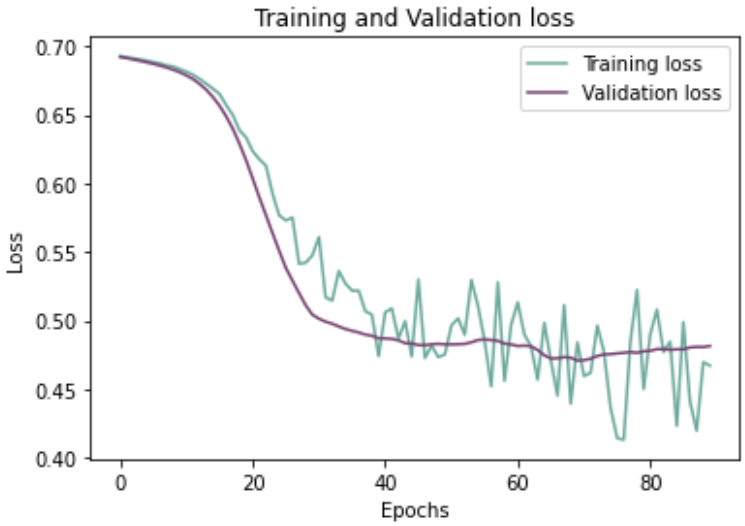
```
# Layers
model.add(Dense(units = 16, kernel_initializer = 'uniform', activation =
'relu', input_dim = 12))
model.add(Dense(units = 8, kernel_initializer = 'uniform', activation =
'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 4, kernel_initializer = 'uniform', activation =
'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 1, kernel_initializer = 'uniform', activation =
'sigmoid'))
from tensorflow.keras.optimizers import SGD
# Compiling the ANN
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy'])

# Train the ANN
history = model.fit(X_train, y_train, batch_size = 32, epochs = 500,
validation_split=0.2)
```

```
6/6 [==============================] - 0s 5ms/step - loss: 0.2721 - accuracy: 0.8547 - val_loss: 1.0377 - val_accuracy: 0.8000
Epoch 496/500
6/6 [==============================] - 0s 4ms/step - loss: 0.2154 - accuracy: 0.8827 - val_loss: 1.0369 - val_accuracy: 0.8000
Epoch 497/500
6/6 [==============================] - 0s 4ms/step - loss: 0.2815 - accuracy: 0.8268 - val_loss: 1.0419 - val_accuracy: 0.8000
Epoch 498/500
6/6 [==============================] - 0s 4ms/step - loss: 0.2632 - accuracy: 0.8436 - val_loss: 1.0445 - val_accuracy: 0.8000
Epoch 499/500
6/6 [==============================] - 0s 4ms/step - loss: 0.2937 - accuracy: 0.8380 - val_loss: 1.0470 - val_accuracy: 0.8000
Epoch 500/500
6/6 [==============================] - 0s 4ms/step - loss: 0.2742 - accuracy: 0.8603 - val_loss: 1.0506 - val_accuracy: 0.8000
```

```
val_accuracy = np.mean(history.history['val_accuracy'])
print("\n%s: %.2f%%" % ('val_accuracy', val_accuracy*100))
```

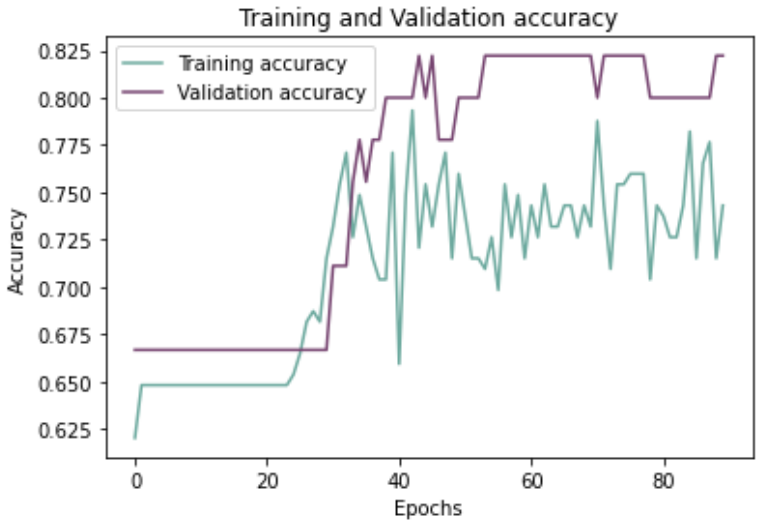
```
val_accuracy: 79.81%
```

```
history_df = pd.DataFrame(history.history)

plt.plot(history_df.loc[:, ['loss']], "#6daa9f", label='Training loss')
plt.plot(history_df.loc[:, ['val_loss']],"#774571", label='Validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc="best")

plt.show()
```



رسم دقة التدريب والتحقق من الصحة على مر الفترات

```
history_df = pd.DataFrame(history.history)

plt.plot(history_df.loc[:, ['accuracy']], "#6daa9f", label='Training
accuracy')
plt.plot(history_df.loc[:, ['val_accuracy']], "#774571", label='Validation
accuracy')

plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



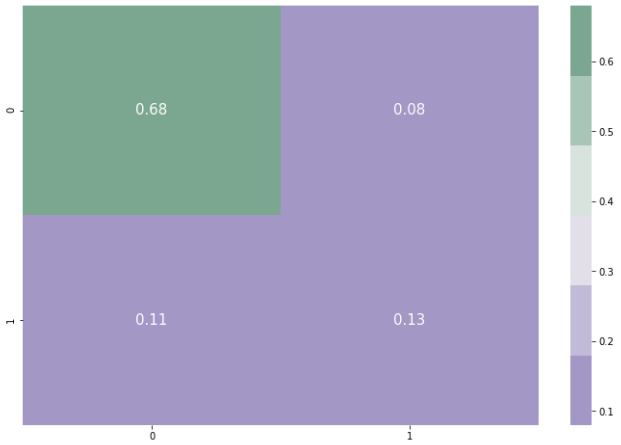
الاستنتاجات

اختتام النموذج بـ:

- الاختبار على مجموعة الاختبار.
- تقييم مصفوفة الارتباك.
- تقييم تقرير التصنيف.

```
# Predicting the test set results
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)
np.set_printoptions()
```

```
# confusion matrix
cmap1 = sns.diverging_palette(275,150,  s=40, l=65, n=6)
plt.subplots(figsize=(12,8))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), cmap = cmap1, annot = True,
annot_kws = {'size':15})
```




```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.83	0.88	0.85	57
1	0.53	0.44	0.48	18
accuracy			0.77	75
macro avg	0.68	0.66	0.67	75
weighted avg	0.76	0.77	0.77	75

حفظ النموذج

```
model.save('heart.h5')
```

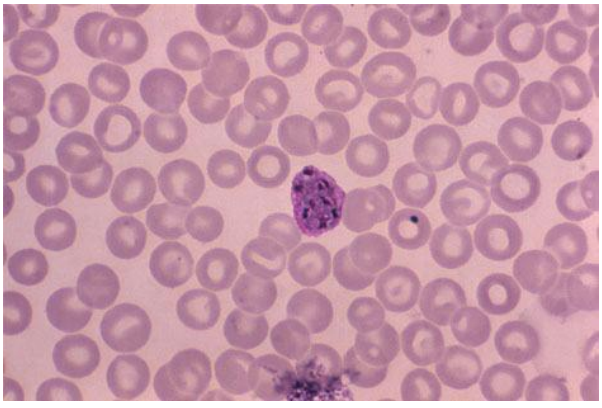
14) كشف الملاريا باستخدام التعلم العميق Malaria Detection using Deep Learning

مقدمة

مع تزايد عدد الجدل الدائر حول انتشار "الذكاء الاصطناعي"، من الحكمة عرض أمثلة على كيف يمكن للتعلم الآلي أن يكون له تطبيقات بعيدة المدى وإيجابيةفي العالم الواقعي. ودعماً لذلك، سنقوم بمعالجة أحد أخطر الأمراض التي تصيب شبه القارة الأفريقية اليوم، وهو الملاريا Malaria.

الملاريا مرض قاتل تسببه طفيليات البلازموديوم التي تنتقل إلى الناس من خلال لدغات بعوضة الأنوفيلة المصابة، والمعروفة باسم ناقلات الملاريا malaria vectors. لقد وجدت منذ 30 مليون سنة، وتم تحديدها على أنها سبب رئيسي للوفاةفي الحضارات القديمةفي جميع أنحاء العالم. اليوم، لا تزال الملاريا مرضًا خطيرًا، حيث يتعرض ما يقرب من نصف سكان العالم للخطر، على الرغم من أن منظمة الصحة العالمية قد حددت المنطقة الأفريقية على أنها تحمل نصيبًا كبيرًا بشكل غير متناسب من عبء الملاريا العالمي، حيث تضم المنطقة 92٪ من سكان العالم. حالات الملاريا و93٪ من وفيات الملاريافي عام 2017.

يمكن التعرف على طفيليات الملاريا عن طريق فحص عينة من دم المريض المصاب تحت المجهر الضوئي. قبل الفحص، يتم نشر العينة عبر شريحة مجهرية، ويتم صبغها بخليط صبغ يعزز تباين طفيلي البلازموديومفي خلايا الدم الحمراء للمريض. يتم قبول هذه التقنية كمعيار من قبل السلطات الصحيةفي جميع أنحاء العالم، وتتميز بدقة مقبولة وفعالية من حيث التكلفة، ولكنها أيضًا تتطلب الكثير من الوقت والعمالة، فضلاً عن أنها تعتمد بشكل كبير على خبرة الفني.



طفيليات الملاريا النشيطة التي تصيب خلايا الدم الحمراء.

لتسريع هذه العملية، سنقدم أتمتة جزئية لاكتشاف الملاريا من خلال بناء مصنف الملاريا على شبكة CNN باستخدام Keras. مجموعة البيانات التي سنستخدمها اليوم هي مجموعة بيانات Malaria Cell Images من Kaggle، والتي تحتوي على أكثر من 13000 صورة RGB لكل من الخلايا غير المصابة والمتطفلة.

لمنع وقت التدريب المفرط، نوصي بأن يقوم المستخدم بتشغيل هذا الكود فقط عند ربطه بموارد وحدة معالجة الرسومات GPU. كمافي البرنامج التعليمي السابق، نفترض أن القارئ على دراية بعناصر التعلم العميق، لا سيما مع تصنيف الصور الأساسي. تمت تغطية النظرية الكامنة وراء CNN على نطاق واسعفي دورات ودروس متعددة، وبالتالي لن تتكرر هنا.

التنفيذ

للبدء، دعنا نستورد حزم بايثون os و shutil لمعالجة البيانات، وتحديد المسارات إلى بياناتنا جنبًا إلى جنب مع المسار إلى أدلة العمل لدينا. نقوم بفصل بياناتنا إلى عينات إيجابية وأخرى سلبية: يشير الحرف "A" إلى العينات المصابة / المتطفلة، بينما يشير الحرف "B" إلى العينات غير المصابة.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


import shutil

import os
print(os.listdir("../input/cell_images/cell_images/"))base_dir='../input/c
ell_images/cell_images/'
work_dir=  "work/"
os.mkdir(work dir)base dir A
='../input/cell_images/cell_images/Parasitized/'
base_dir_B ='../input/cell_images/cell_images/Uninfected/'

work_dir_A = "work/A/"
os.mkdir(work_dir_A)
work_dir_B = "work/B/"
os.mkdir(work_dir_B)
```

بعد ذلك، قمنا بتقسيم بياناتنا ـ لأغراض التدريب والتحقق والاختبار. تحت كل مجلد مقسم، سننشئ مجلدين فرعيين لفئات الإخراج لدينا، يُطلق عليهما الإيجابي (المصاب infected) والسالب (غير مصاب uninfected). سنقوم بنسخ الصور من كلا دليلي المصدر إلى هذه لاحقًاـ وهذا يتحايل على بعض قيود القراءة فقط التي تؤثر على بيئات معينة.

```
train_dir = os.path.join(work_dir, 'train')
os.mkdir(train_dir)

validation_dir = os.path.join(work_dir, 'validation')
os.mkdir(validation_dir)

test_dir = os.path.join(work_dir, 'test')
```

```
os.mkdir(test_dir)

print("New directories for train, validation, and test
created")train_pos_dir = os.path.join(train_dir, 'pos')
os.mkdir(train pos dir)
train_neg_dir = os.path.join(train_dir, 'neg')
os.mkdir(train_neg_dir)

validation_pos_dir = os.path.join(validation_dir, 'pos')
os.mkdir(validation_pos_dir)
validation_neg_dir = os.path.join(validation_dir, 'neg')
os.mkdir(validation_neg_dir)

test_pos_dir = os.path.join(test_dir, 'pos')
os.mkdir(test_pos_dir)
test_neg_dir = os.path.join(test_dir, 'neg')
os.mkdir(test_neg_dir)

print("Train, Validation, and Test folders made for both A and B
datasets")
```

لتبسيط الأمور لعرضها وتحليلها لاحقًا، دعنا نعيد تسمية جميع صورنا لتتوافق مع الفئة المستهدفة، سواء كانت موجبة (A) أو سلبية (B).

```
i = 0

for filename in os.listdir(base_dir_A):
dst ="pos" + str(i) + ".jpg"
src =base_dir_A + filename
dst =work_dir_A + dst

# rename() function will
# rename all the files
shutil.copy(src, dst)
i += 1



j = 0

for filename in os.listdir(base_dir_B):
dst ="neg" + str(j) + ".jpg"
src =base_dir_B + filename
dst =work_dir_B + dst

# rename() function will
# rename all the files
shutil.copy(src, dst)
j += 1

print("Images for both categories have been copied to working directories,
renamed to A & B + num")
```

الآن وقد تم إعداد جميع المجلدات، فلنقم بإجراء تقسيم يدوي تدريب/ اختبار ونسخ صور المصدر الخاصة بنا إلى الدلائل الخاصة بكل منها.في مثالنا، ستحتوي كل فئة على 3000 صورة تدريب و1000 صورة تحقق و500 صورة اختبار. تُستخدم صور التدريب لتلائم النموذج باستخدام معلمات الشبكة، بينما تُستخدم صور التحقق لضبط المعلمات المذكورة للحصول

على قدرة تعميم أفضل ودقة معززة. يتقارب دور مجموعات البيانات الثلاث تقريبًا مع أسئلة الممارسة والامتحانات التدريبية والامتحانات النهائية، على التوالي.

```
fnames = ['pos{}.jpg'.format(i) for i in range(3000)]
for fname in fnames:
    src = os.path.join(work_dir_A, fname)
    dst = os.path.join(train_pos_dir, fname)
    shutil.copyfile(src, dst)

fnames = ['pos{}.jpg'.format(i) for i in range(3000, 4000)]
for fname in fnames:
    src = os.path.join(work_dir_A, fname)
    dst = os.path.join(validation_pos_dir, fname)
    shutil.copyfile(src, dst)

fnames = ['pos{}.jpg'.format(i) for i in range(4000, 4500)]
for fname in fnames:
    src = os.path.join(work_dir_A, fname)
    dst = os.path.join(test_pos_dir, fname)
    shutil.copyfile(src, dst)



fnames = ['neg{}.jpg'.format(i) for i in range(3000)]
for fname in fnames:
    src = os.path.join(work_dir_B, fname)
    dst = os.path.join(train_neg_dir, fname)
    shutil.copyfile(src, dst)

fnames = ['neg{}.jpg'.format(i) for i in range(3000, 4000)]
for fname in fnames:
    src = os.path.join(work_dir_B, fname)
    dst = os.path.join(validation_neg_dir, fname)
    shutil.copyfile(src, dst)

fnames = ['neg{}.jpg'.format(i) for i in range(4000, 4500)]
for fname in fnames:
    src = os.path.join(work_dir_B, fname)
    dst = os.path.join(test_neg_dir, fname)
    shutil.copyfile(src, dst)

print("Train, validation, and test datasets split and ready for
use")print('total training pos images:', len(os.listdir(train_pos_dir)))
print('total training neg images:', len(os.listdir(train_neg_dir)))
print('total validation pos images:', len(os.listdir(validation_pos_dir)))
print('total validation neg images:', len(os.listdir(validation_neg_dir)))
print('total test pos images:', len(os.listdir(test_pos_dir)))
print('total test meg images:', len(os.listdir(test_neg_dir)))
```

بعد ذلك، نقوم بإعداد وتطبيع مدخلات البيانات الخاصة بنا للشبكة. يمكننا استخدام فئة ImageDataGenerator من Keras لأتمتة هذه الخطوة، مع إنتاج دفعات إدخالفي نفس الوقت لتدريب التدرج الاشتقاقي الدفعي batch gradient descent. للتلخيص، يقوم ImageDataGenerator بتعديل شدة البكسل إلى ما بين 0 و 1، ويحول محتوى JPEG إلى خرائط موتر فاصلة عائمة لكل صورة، ويغير حجمها إلى 150 × 150 بكسل. أثناء قيامنا بتقسيم بياناتنا إلى فئتين من مجلدات الإخراج، نحدد class_mode على أنها "binary "، وسوف يتعلم ImageDataGenerator تلقائيًا ربط محتويات كل مجلد بالتسمية الصحيحة.

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)train_generator =
train_datagen.flow_from_directory(
        train dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
validation_generator = test_datagen.flow_from_directory(
        validation_dir,target_size=(150, 150),
        batch_size=20,
        class_mode='binary')

print("Image preprocessing complete")
```

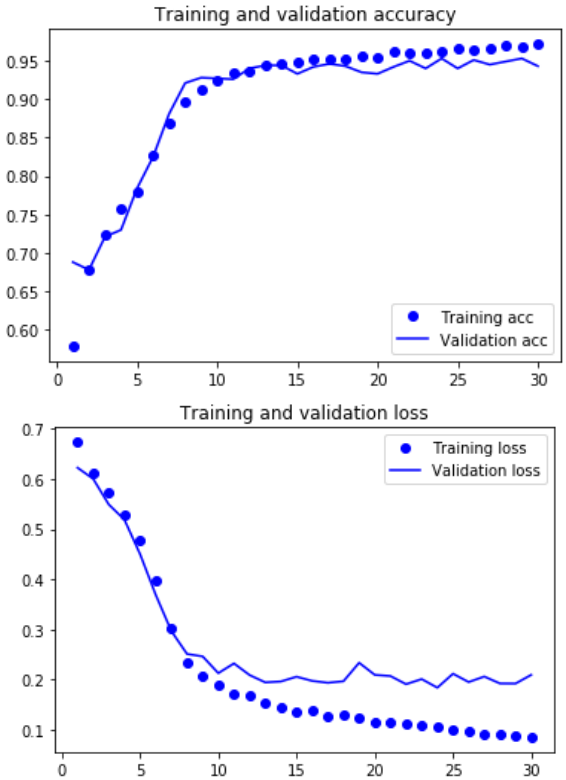
بعد الانتهاء من إعداد جميع البيانات، يمكننا الآن بناء شبكتنا. تتكون بُنية CNN الخاصة بنا من عدة طبقات تلافيفية convolutional layers لتمثيل الميزات وطبقات متصلة بالكامل fully-connected layers متبوعة بمصنف sigmoid للتصنيف الثنائي.

باختصار، يتمثل دور طبقة الالتفاففي تعلم ميزات تمييزية منخفضة المستوى وعالية المستوى من خلال تحديد أنماط البكسل المهمةفي مناطق الصورة. يعمل دور طبقات التجميع القصوى max pooling layers على تقليل العبء الحسابي عن طريق تحديد الحد الأقصى لقيمة الإخراج للالتفاف لخريطة تنشيط المخرجات. ـ طبقة متصلة، وأخيرًا طبقة تصنيف تتميز بتنشيط sigmoid لإنتاج نتيجة تصنيف.في هذا المشروع، نستخدم مُحسِّن RMSprop ولكن يتم تشجيع القارئ على تجربة أدوات تحسين أخرى داخل مكتبة Keras، مثل SGD وADAM.

```
from keras import layers
from keras import modelsmodel = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.summary()

from keras import optimizers
model.compile(loss='binary_crossentropy',
optimizer=optimizers.RMSprop(lr=1e-5),
metrics=['acc'])

print("Model created")
```

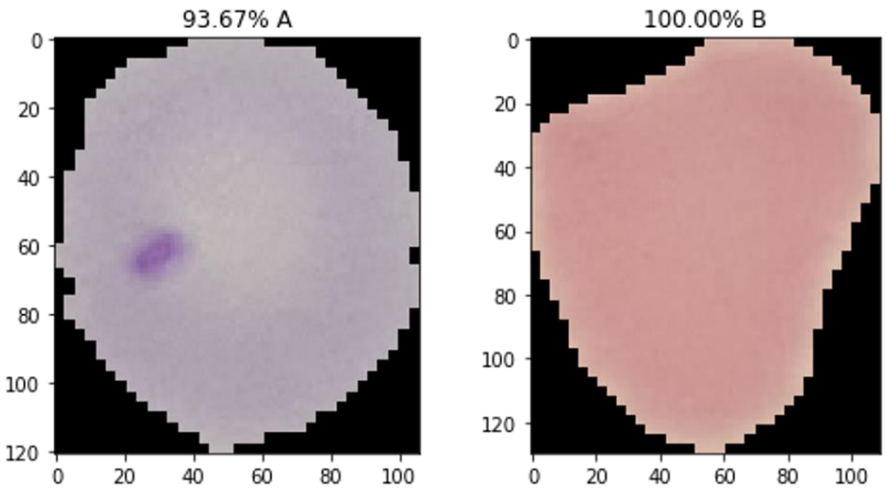
فلنبدأ الآنفي تدريب نموذجنا. سوف نتدرب لمدة 30 حقبة لتحقيق التوازن بين الدقة والكفاءة الحسابية. ثم يتم حفظ نموذجنا المُدربفي متغير السجل، حيث يمكننا من خلاله رسم مقاييس الخطأ والدقة.

```
history = model.fit_generator(
        train_generator,
        steps_per_epoch=100,
        epochs=30,
        validation_data=validation_generator,
        validation_steps=200)
model.save('basic_malaria_pos_neg_v1.h5')import matplotlib.pyplot as
pltacc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```



تبدو جيدة! تزيد دقة التحقق الخاصة بنا عن 92٪، على الرغم من أن مخططات التحقق تشير إلى أننا بدأنافي الاستفادة من بيانات التدريب الخاصة بنا. ومع ذلك، ليس سيئًا بالنسبة لنموذج أولي سريع. أخيرًا، دعنا نستخدم نموذجنا المُدرب على مجموعة بيانات الاختبار الخاصة بنا، ونفحص النتائج بصريًا.

```
eval_datagen = ImageDataGenerator(rescale=1./255)
eval_generator = eval_datagen.flow_from_directory(
        test_dir,target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
eval_generator.reset()
pred = model.predict_generator(eval_generator,1000,verbose=1)
print("Predictions finished")

import matplotlib.image as mpimgfor index, probability in enumerate(pred):
    image_path = test_dir + "/" +eval_generator.filenames[index]
    img = mpimg.imread(image_path)

    plt.imshow(img)
    print(eval_generator.filenames[index])
    if probability > 0.5:
        plt.title("%.2f" % (probability[0]*100) + "% B")
    else:
        plt.title("%.2f" % ((1-probability[0])*100) + "% A")
    plt.show()
```



صورتان اختباريتان بفواصل ثقة متوقعة (يسار: مصاب، يمين: غير مصاب)

لقد قدمنا لك الآن المخططات لتصنيف الملاريا. العب حولك واكتشف ما إذا كان بإمكانك تحسينه، وصنع أداة قابلة للتطبيق لإنقاذ الأرواح!

15) التعرف على المشاة باستخدام CNN Pedestrian Recognition using CNN

لتوضيح كيف يمكن استخدام الشبكة العصبية التلافيفية CNNفي تطوير السيارات ذاتية القيادة self-driving cars، سننتقل خلال عملية تدريب CNN لاكتشاف ما إذا كانت الصورة بها أحد المشاة pedestrian أم لا. يعد التعرف على الصور جزءًا لا يتجزأ من قدرات القيادة الذاتية، ويجب أن تتمتع المركبات بقدرة قوية على التمييز عندما يكون أحد المشاة بالقرب من أحد تدابير السلامة الهامة.

في هذا العرض التوضيحي، نستخدم Google Colab ونخزن الملفات على Google Drive.

استيراد المكتبات الضرورية

نبدأ باستيراد المكتبات المطلوبة.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as npimg
import os

## Keras
import keras
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Convolution2D, MaxPooling2D, Dropout, Flatten,
Dense
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam

import pandas as pd
import random
import ntpath
from matplotlib.image import imread

## Sklearn
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
```

بعد ذلك، نقوم باستيراد البيانات من Google Drive. تتكون البيانات من مجموعات بيانات التدريب والتحقق من الصحة، كل منها يحتوي على صور للمشاة وصور بدون مشاة. بدلاً من ذلك، يمكن أيضًا تخزين البيانات محليًا وتحميلها.

```
data_directory = '/content/drive/MyDrive/Colab Notebooks/MGSC 673 Group
Project/data'

os.listdir(data_directory)
```

```
# Set a path for the Training and Validation directories
train_path=data_directory+'/train/'
validation_path=data_directory+'/validation/'

os.listdir(train_path)

os.listdir(validation_path)
```

عرض الأمثلة

أولاً، دعنا نطبع صورة نموذجية تتضمن أحد المشاة.

```
os.listdir(train_path+'pedestrian')[9]

# Print a picture with a pedestrian:
pedestrian=train_path+'pedestrian/'+'pic_160.jpg'
imread(pedestrian).shape
plt.imshow(imread(pedestrian))
```

بعد ذلك، دعنا نطبع صورة نموذجية بدون وجود مشاة.

```
os.listdir(train_path+'no pedestrian')[9]

# Print a picture without a pedestrian:
os.listdir(train_path+'no pedestrian')[5]
no_pedestrian=train_path+'no pedestrian/'+'train (399).jpg'
plt.imshow(imread(no_pedestrian))
```

معالجة الصور

لإعداد الصور للشبكة العصبية، نقوم أولاً بمعالجتها مسبقًا باستخدام أداة تولد العديد من الاختلافات الطفيفة لكل صورة. ينتج عن هذا نموذج أكثر دقة مع انخفاض طفيف نسبيًافي الكفاءة.

نستخدم دالة ImageDataGenerator لزيادة augment الصورفي مجموعة التدريب الخاصة بنا. زيادة الصورة هي عملية إنشاء بيانات تدريب إضافية عن طريق إنشاء المزيد من الصور التي تمثل اختلافات طفيفة (دوران، انعكاسات، إلخ) للصور التي لدينا بالفعل. توفر هذه العملية المزيد من الصور لنموذجنا للتدريب عليها.

```
image_gen=ImageDataGenerator(rescale=1/255,shear_range=0.1,zoom_range=0.1,
fill_mode='nearest')

image_gen.flow_from_directory(train_path)

image_gen.flow_from_directory(validation_path)

# specify the shape of the input images:
image_shape=(200,200,3)
```

بناء نموذج CNN

يمكننا الآن بناء النموذج بالطبقات التي اخترناها. تم اختيار البُنية بعد اختبار مجموعات مختلفة من الطبقات والمعلمات الفائقة.في النهاية، تضمن النموذج الأفضل أداءً ما يلي:

1. طبقة الالتفاف Convolution layer وطبقة التجميع pooling layer (تتكرر 3 مرات).
2. طبقة التسطيح Flattening layer.
3. طبقة التسرب Dropout layer بمعدل تسرب 0.5 وتنشيط ReLu.
4. الطبقة النهائية بدالة التنشيط sigmoid
5. دالة خطأ الانتروبيا الثنائية Binary cross-entropy مع مُحسِّن آدم Adam optimizer.

```
model=Sequential()
# Add convolution and pooling layer:
model.add(Convolution2D(filters=32,kernel_size=(3,3),input_shape=image_sha
pe,activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
# Add convolution and pooling layer:
model.add(Convolution2D(filters=64,kernel_size=(3,3),input_shape=image_sha
pe,activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
# Add convolution and pooling layer:
model.add(Convolution2D(filters=64,kernel_size=(3,3),input_shape=image_sha
pe,activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2,2)))
# Add a flattening layer:
model.add(Flatten())
# Use dropout hyperparameter:
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1,activation='sigmoid'))
# Compile the model:
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accura
cy'])

model.summary()
```

الاستجابة هو ملخص للنموذج:

```
Model: "sequential"
_________________________________________________________________ Layer
(type) Output Shape Param #
================================================================= conv2d
(Conv2D) (None, 198, 198, 32) 896 max_pooling2d (MaxPooling2D (None, 99,
99, 32) 0 ) conv2d_1 (Conv2D) (None, 97, 97, 64) 18496 max_pooling2d_1
(MaxPooling (None, 48, 48, 64) 0 2D) conv2d_2 (Conv2D) (None, 46, 46, 64)
36928 max_pooling2d_2 (MaxPooling (None, 23, 23, 64) 0 2D) flatten
(Flatten) (None, 33856) 0 dense (Dense) (None, 128) 4333696 dropout
(Dropout) (None, 128) 0 dense_1 (Dense) (None, 1) 129
================================================================= Total
params: 4,390,145 Trainable params: 4,390,145 Non-trainable params: 0
```

تدريب نموذج CNN

لقد وضعنا قاعدة التوقف المبكر early stopping بحيث يتوقف النموذج عن التدريب عندما تتوقف خطأ التحقق من الصحة عن التحسن، ونحدد حجم دفعة من 16 لتحديد عدد الصور التي يتم تمريرها عبر الشبكةفي المرة الواحدة.

```
early_stop=EarlyStopping(monitor='val_loss',patience=5)
batch_size=16
```

نستخدم الدالة flow_from_directory() لسحب صور التدريب والتحقق من الصحة التي تم إنشاؤهافي خطوة المعالجة المسبقة بواسطة دالة ImageDataGenerator من مسارات ملف التدريب والتحقق من الصحة على التوالي.

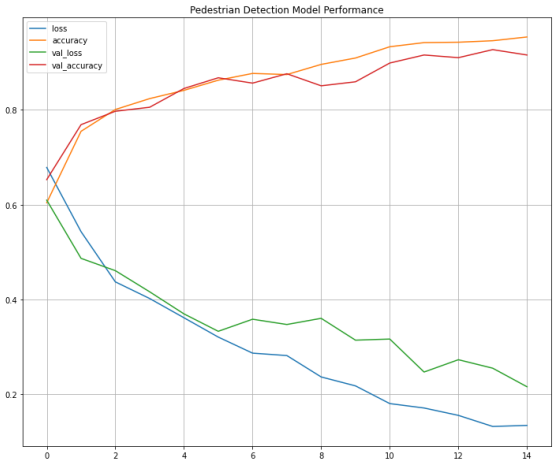
```
train_image_gen =
image_gen.flow_from_directory(train_path,target_size=image_shape[:2],color
_mode='rgb',
batch_size=batch_size,class_mode='binary')

val_image_gen =
image_gen.flow_from_directory(validation_path,target_size=image_shape[:2],
color_mode='rgb',
batch_size=batch_size,class_mode='binary',shuffle=False)

results=model.fit_generator(train_image_gen,epochs=30,validation_data=val_
image_gen,callbacks=[early_stop])
```

عند تدريب النموذج، نرى الاستجابة التالية التي توضح المقاييس مع كل فترة تدريب إضافية:

```
Epoch 1/15
79/79 [==============================] - 230s 3s/step - loss: 0.6781 - accuracy: 0.6038 - val_loss: 0.6092 - val_accuracy: 0.6525
Epoch 2/15
79/79 [==============================] - 23s 296ms/step - loss: 0.5428 - accuracy: 0.7544 - val_loss: 0.4865 - val_accuracy: 0.7684
Epoch 3/15
79/79 [==============================] - 21s 270ms/step - loss: 0.4370 - accuracy: 0.8003 - val_loss: 0.4606 - val_accuracy: 0.7966
Epoch 4/15
79/79 [==============================] - 22s 273ms/step - loss: 0.4018 - accuracy: 0.8233 - val_loss: 0.4160 - val_accuracy: 0.8051
Epoch 5/15
79/79 [==============================] - 21s 272ms/step - loss: 0.3613 - accuracy: 0.8407 - val_loss: 0.3695 - val_accuracy: 0.8446
Epoch 6/15
79/79 [==============================] - 22s 277ms/step - loss: 0.3205 - accuracy: 0.8621 - val_loss: 0.3327 - val_accuracy: 0.8672
Epoch 7/15
79/79 [==============================] - 21s 268ms/step - loss: 0.2867 - accuracy: 0.8764 - val_loss: 0.3582 - val_accuracy: 0.8559
Epoch 8/15
79/79 [==============================] - 21s 271ms/step - loss: 0.2816 - accuracy: 0.8740 - val_loss: 0.3471 - val_accuracy: 0.8757
Epoch 9/15
79/79 [==============================] - 21s 271ms/step - loss: 0.2365 - accuracy: 0.8954 - val_loss: 0.3601 - val_accuracy: 0.8503
Epoch 10/15
79/79 [==============================] - 22s 280ms/step - loss: 0.2177 - accuracy: 0.9089 - val_loss: 0.3140 - val_accuracy: 0.8588
Epoch 11/15
79/79 [==============================] - 22s 279ms/step - loss: 0.1803 - accuracy: 0.9326 - val_loss: 0.3164 - val_accuracy: 0.8983
Epoch 12/15
79/79 [==============================] - 22s 277ms/step - loss: 0.1710 - accuracy: 0.9414 - val_loss: 0.2468 - val_accuracy: 0.9153
Epoch 13/15
79/79 [==============================] - 22s 276ms/step - loss: 0.1555 - accuracy: 0.9422 - val_loss: 0.2728 - val_accuracy: 0.9096
Epoch 14/15
79/79 [==============================] - 22s 274ms/step - loss: 0.1321 - accuracy: 0.9453 - val_loss: 0.2549 - val_accuracy: 0.9266
Epoch 15/15
79/79 [==============================] - 22s 274ms/step - loss: 0.1343 - accuracy: 0.9532 - val_loss: 0.2158 - val_accuracy: 0.9153
```

مقاييس أداء الرسم

للحصول على تمثيل مرئي لأداء النموذج، نرسم الخطأ والدقة على مر الفترات.



تقييم نتائج النموذج

خطوتنا الأخيرة هي تقييم جودة النموذج من خلال النظرفي تقرير التصنيف classification report ومصفوفة الارتباك confusion matrix.

علينا أولاً تحويل التنبؤات إلى قيم صواب أو خطأ:

```
pred=model.predict_generator(val_image_gen)

pred[:5]

predictions=pred>0.5
predictions[:5]
```

يمكننا بعد ذلك طباعة تقرير التصنيف، والذي يوضح قيم precision وrecall وF1 score وsupport values.في حالة اكتشاف المشاة، من المهم أن يتمتع النموذج بدقة عالية للقيم السالبة. هذا مهم لأن التنبؤ السلبي يعني أن السيارة ستعتقد أنه لا يوجد أي مشاة، وهو أمر غير آمن للغاية إذا كان هناك بالفعل أحد المشاة.

في نموذجنا، نرى أن دقة القيم السالبة تبلغ 90٪، مما يعني أن النموذج سيفشلفي التعرف على ما يقرب من 10٪ من المشاة، وهو مقياس يجب تتبعه وتحسينه بمرور الوقت.

```
print(classification_report(val_image_gen.classes,predictions))
```

	precision	recall	f1-score	support
0	0.90	0.92	0.91	177
1	0.92	0.90	0.91	177
accuracy			0.91	354
macro avg	0.91	0.91	0.91	354
weighted avg	0.91	0.91	0.91	354

أخيرًا، نطبع مصفوفة الارتباك كخطوتنا الأخيرة.

```
confusion_matrix(val_image_gen.classes, predictions)
```

يعد هيكل CNN نهجًا قويًا مع تطبيقات واسعة النطاق، لا سيما تلك الموجودةفي مساحة السيارة المستقلة autonomous vehicle. على مدى السنوات القادمة، من المرجح أن يتم تنفيذ CNNفي المزيد من الأنظمة ذات الفوائد الفريدة.

16) التعرف على العاطفة من الكلام مع librosa Speech Emotion Recognition with librosa

التعرف على عاطفة الكلام Speech emotion recognition، أفضل مشروع بايثون صغير على الإطلاق. يمكن رؤية أفضل مثال على ذلكفي مراكز الاتصال. إذا لاحظت أن موظفي مراكز الاتصال لا يتحدثون أبدًا بالطريقة نفسها، فإن طريقتهمفي الترويج / التحدث إلى العملاء تتغير مع العملاء. الآن، يحدث هذا أيضًا مع عامة الناس، ولكن ما مدى صلة ذلك بمراكز الاتصال؟ إليك إجابتك، يتعرف الموظفون على مشاعر العملاء من الكلام، حتى يتمكنوا من تحسين خدماتهم وتحويل المزيد من الأشخاص. بهذه الطريقة، يستخدمون التعرف على عاطفة الكلام. لذا، دعونا نناقش هذا المشروع بالتفصيل.

ما هو التعرف على عاطفة الكلام؟

التعرف على عاطفة الكلام Speech Emotion Recognition، والمختصر بـ SER، هو فعل محاولة التعرف على المشاعر الإنسانية والحالات العاطفية من الكلام. هذا هو الاستفادة من حقيقة أن الصوت غالبًا ما يعكس العاطفة الأساسية من خلال النغمة والنبرة. هذه أيضًا هي الظاهرة التي تستخدمها الحيوانات مثل الكلاب والخيول لتتمكن من فهم المشاعر البشرية.

SER صعب لأن المشاعر ذاتية والتعليق التوضيحي يمثل تحديًا.

ما هي librosa؟

librosa هي مكتبة بايثون لتحليل الصوت والموسيقى. يحتوي على تخطيط حزمة مسطح، ويوحد الواجهات والأسماء، والتوافق مع الإصدارات السابقة، والدوال المعيارية، والرمز القابل للقراءة. علاوة على ذلك،في مشروع بايثون الصغير هذا، نوضح كيفية تثبيته (وبعض الحزم الأخرى) باستخدام pip.

ما هو JupyterLab؟

JupyterLab عبارة عن واجهة مستخدم مفتوحة المصدر ومستندة إلى الويب لـ Project Jupyter ولديها جميع الوظائف الأساسية لـ Jupyter Notebook ، مثل أجهزة الكمبيوتر المحمولة والمحطات الطرفية ومحررات النصوص ومتصفحات الملفات والمخرجات الغنية والمزيد. ومع ذلك، فإنه يوفر أيضًا دعمًا محسنًا لملحقات الجهات الخارجية.

لتشغيل التعليمات البرمجيةفي JupyterLab، ستحتاج أولاً إلى تشغيلها باستخدام موجه الأوامر:

```
C:\Users\DataFlair>jupyter lab
```

هذا سيفتح لك جلسة جديدةفي متصفحك. قم بإنشاء وحدة تحكم جديدة وابدأفي كتابة التعليمات البرمجية الخاصة بك. يمكن لـ JupyterLab تنفيذ عدة أسطر من التعليمات البرمجيةفي وقت

واحد؛ الضغط على مفتاح الإدخال لن يؤدي إلى تنفيذ الكود الخاص بك، ستحتاج إلى الضغط على Shift + Enter لنفسه.

التعرف على عاطفة الكلام - الهدف

لبناء نموذج للتعرف على المشاعر من الكلام باستخدام مكتبات librosa و sklearn ومجموعة بيانات RAVDESS.

التعرف على عاطفة الكلام - حول مشروع Python Mini

في مشروع بايثون المصغر هذا، سنستخدم مكتبات librosa و soundfile و sklearn (من بين أمور أخرى) لبناء نموذج باستخدام مصنف MLPClassifier. سيكون هذا قادرًا على التعرف على المشاعر من ملفات الصوت. سنقوم بتحميل البيانات، واستخراج الميزات منها، ثم تقسيم مجموعة البيانات إلى مجموعات تدريب واختبار. بعد ذلك، سنهيئ MLPClassifier وندرب النموذج. أخيرًا، سنحسب دقة نموذجنا.

مجموعة البيانات

بالنسبة لمشروع بايثون المصغر هذا، سنستخدم مجموعة بيانات RAVDESS؛ هذه هي قاعدة بيانات Ryerson الصوتية المرئية لمجموعة بيانات الكلام والأغاني العاطفية، وهي مجانية التنزيل. تحتوي مجموعة البيانات هذه على 7356 ملفًا تم تصنيفها من قبل 247 فردًا 10 مرات على الصحة العاطفية وكثافتها وصدقها. يبلغ حجم مجموعة البيانات بالكامل 24.8 غيغابايت من 24 ممثلًا، لكننا خفضنا معدل العينةفي جميع الملفات، ويمكنك تنزيلها من هنا.

المتطلبات الأساسية

ستحتاج إلى تثبيت المكتبات التالية بـ pip:

```
pip install librosa soundfile numpy sklearn pyaudio
```

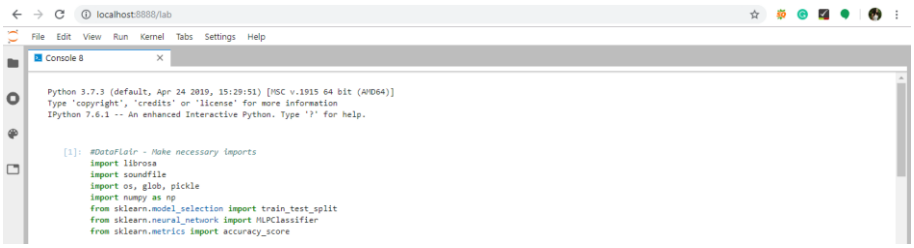
إذا واجهت مشكلاتفي تثبيت librosa مع pip، فيمكنك تجربته باستخدام conda.

خطوات لمشاريع بايثون التعرف على عاطفة الكلام

الخطوة 1: استيراد المكتبات الضرورية:

```
import librosa
import soundfile
import os, glob, pickle
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

لقطات الشاشة:



الخطوة 2: تحديد خاصية extract_feature لاستخراج ميزات mfcc وchroma وmel من ملف صوتي. تأخذ هذه الدالة 4 معلمات ـ اسم الملف وثلاث معلمات منطقية للميزات الثلاث:

- **mfcc**: Mel Frequency Cepstral Coefficient ، يمثل طيف القدرة قصير المدى للصوت.
- **chroma**: تتعلق بفئات الصوت الـ 12 المختلفة.
- **mel**: Mel Spectrogram Frequency.

افتح الملف الصوتي باستخدام soundfile.SoundFile باستخدام_as بحيث يتم إغلاقه تلقائيًا بمجرد الانتهاء. اقرأ منه واسمه X. أيضًا، احصل على معدل العينة. إذا كانت chroma صحيحة، احصل على Short-Time Fourier Transform لـ X.

دع النتيجة تكون مصفوفة numpy فارغة. الآن، لكل ميزة من الميزات الثلاثة، إذا كانت موجودة، قم بإجراء استدعاء للدالة المقابلة من librosa.feature (على سبيل المثال_ librosa.feature.mfcc لـ mfcc) ، واحصل على القيمة المتوسطة. استدعاء الدالة ()hstack من numpy مع النتيجة وقيمة الميزة ، وتخزين هذافي النتيجة. ()hstack يكدس المصفوفات بالتسلسل الأفقي (بطريقة عمودية). ثم أعد النتيجة.

```
#DataFlair - Extract features (mfcc, chroma, mel) from a sound file
def extract_feature(file_name, mfcc, chroma, mel):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        if chroma:
            stft=np.abs(librosa.stft(X))
        result=np.array([])
        if mfcc:
            mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate,
n_mfcc=40).T, axis=0)
            result=np.hstack((result, mfccs))
        if chroma:
            chroma=np.mean(librosa.feature.chroma_stft(S=stft,
sr=sample_rate).T,axis=0)
```

```
            result=np.hstack((result, chroma))
if mel:
            mel=np.mean(librosa.feature.melspectrogram(X,
sr=sample_rate).T,axis=0)
            result=np.hstack((result, mel))
return result
```

لقطة الشاشة:

```
[2]: #DataFlair - Extract features (mfcc, chroma, mel) from a sound file
     def extract_feature(file_name, mfcc, chroma, mel):
         with soundfile.SoundFile(file_name) as sound_file:
             X = sound_file.read(dtype="float32")
             sample_rate=sound_file.samplerate
             if chroma:
                 stft=np.abs(librosa.stft(X))
             result=np.array([])
             if mfcc:
                 mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
                 result=np.hstack((result, mfccs))
             if chroma:
                 chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
                 result=np.hstack((result, chroma))
             if mel:
                 mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
                 result=np.hstack((result, mel))
         return result
```

الخطوة 3: الآن، دعنا نحدد قاموسًا يحتوي على الأرقام والعواطف المتوفرةفي مجموعة بيانات RAVDESS، وقائمة تضم من نريدهم ـ الهدوء الآن، دعنا نحدد قاموسًا يحتوي على الأرقام والعواطف المتوفرةفي مجموعة بيانات RAVDESS، وقائمة تضم من نريدهم ـ الهدوء والسعادة والخوف والاشمئزاز. والسعادة happy والخوف fearful والاشمئزاز disgust.

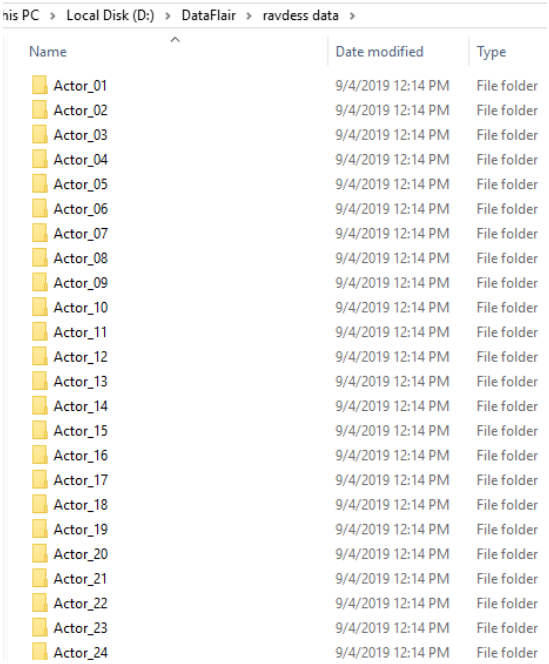
```
#DataFlair - Emotions in the RAVDESS dataset
emotions={
  '01':'neutral',
  '02':'calm',
  '03':'happy',
  '04':'sad',
  '05':'angry',
  '06':'fearful',
  '07':'disgust',
  '08':'surprised'
}

#DataFlair - Emotions to observe
observed_emotions=['calm', 'happy', 'fearful', 'disgust']
```

لقطة الشاشة:

```
[3]: #DataFlair - Emotions in the RAVDESS dataset
     emotions={
         '01':'neutral',
         '02':'calm',
         '03':'happy',
         '04':'sad',
         '05':'angry',
         '06':'fearful',
         '07':'disgust',
         '08':'surprised'
     }

     #DataFlair - Emotions to observe
     observed_emotions=['calm', 'happy', 'fearful', 'disgust']
```

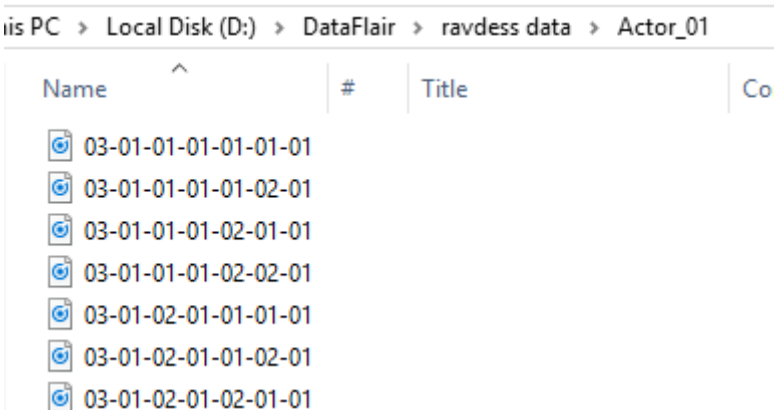
الخطوة 4: الآن، دعنا نحمل البيانات بدالة load_data()ـ يأخذ هذا الحجم النسبي لمجموعة الاختبار كمعامل. x و y قوائم فارغة ؛ سنستخدم الدالة glob() من وحدة glob للحصول على جميع أسماء المسار لملفات الصوتفي مجموعة البيانات الخاصة بنا. النمط الذي نستخدمه لهذا هو: "D:\\DataFlair\\ravdess data\\Actor_**.wav". هذا لأن مجموعة البيانات الخاصة بنا تبدو كما يلي:

لقطة الشاشة:



لذلك، لكل مسار من هذا القبيل، احصل على الاسم الأساسي للملف، العاطفة عن طريق تقسيم الاسم حول "-" واستخراج القيمة الثالثة:

لقطة الشاشة:



باستخدام قاموس المشاعر لدينا، يتحول هذا الرقم إلى عاطفة، وتتحقق دالتنا مما إذا كانت هذه المشاعر مدرجةفي قائمة المشاعر المرصودة لدينا؛ إذا لم يكن كذلك، فإنه يستمرفي الملف التالي. يقوم بإجراء استدعاء لاستخراج الميزة extract_feature وتخزين ما يتم إرجاعهفي "feature". ثم يقوم بإلحاق الميزة بـ x والعاطفة بـ y. لذلك، تحتوي القائمة x على الميزات وy تحمل العواطف. نستدعي الدالة train_test_split بها، وحجم الاختبار، وقيمة الحالة العشوائية، ونعيد ذلك.

```
#DataFlair - Load the data and extract features for each sound file
def load_data(test_size=0.2):
    x,y=[],[]
    for file in glob.glob("D:\\DataFlair\\ravdess
data\\Actor_*\\*.wav"):
        file name=os.path.basename(file)
        emotion=emotions[file_name.split("-")[2]]
        if emotion not in observed_emotions:
            continue
        feature=extract_feature(file, mfcc=True, chroma=True,
mel=True)
        x.append(feature)
        y.append(emotion)
    return train_test_split(np.array(x), y, test_size=test_size,
random_state=9)
```

لقطة الشاشة:

```
[4]: #DataFlair - Load the data and extract features for each sound file
     def load_data(test_size=0.2):
         x,y=[],[]
         for file in glob.glob("D:\\DataFlair\\ravdess data\\Actor_*\\*.wav"):
             file_name=os.path.basename(file)
             emotion=emotions[file_name.split("-")[2]]
             if emotion not in observed_emotions:
                 continue
             feature=extract_feature(file, mfcc=True, chroma=True, mel=True)
             x.append(feature)
             y.append(emotion)
         return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
```

الخطوة 5: حان الوقت لتقسيم مجموعة البيانات إلى مجموعات تدريب واختبار! دعونا نحتفظ بمجموعة الاختبار 25٪ من كل شيء ونستخدم دالة load_data لهذا الغرض.

```
#DataFlair - Split the dataset
x_train,x_test,y_train,y_test=load_data(test_size=0.25)
```

لقطة شاشة:

```
[5]: #DataFlair - Split the dataset
     x_train,x_test,y_train,y_test=load_data(test_size=0.25)
```

الخطوة 6: مراقبة شكل مجموعات بيانات التدريب والاختبار:

```
#DataFlair - Get the shape of the training and testing datasets
print((x_train.shape[0], x_test.shape[0]))
```

لقطة الشاشة:

```
[6]: #DataFlair - Get the shape of the training and testing datasets
     print((x_train.shape[0], x_test.shape[0]))

     (576, 192)
```

الخطوة 7: الحصول على عدد الميزات المستخرجة.

لقطة شاشة الإخراج:

```
[7]: #DataFlair - Get the number of features extracted
     print(f'Features extracted: {x_train.shape[1]}')

     Features extracted: 180
```

الخطوة 8: الآن، دعنا نبدأ تصنيف MLPC. هذا هو مصنف متعدد الطبقات Perceptron؛ يقوم بتحسين دالة خطأ السجل باستخدام LBFGS أو التدرج الاشتقاقي العشوائي SGD. على

عكس SVM أو Naive Bayes، فإن MLPClassifier لديه شبكة عصبية داخلية لغرض التصنيف. هذا هو نموذج feedforward ANN.

```
#DataFlair - Initialize the Multi Layer Perceptron Classifier
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08,
hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
```

لقطة الشاشة:

```
[8]: #DataFlair - Initialize the Multi Layer Perceptron Classifier
     model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
```

الخطوة 9: تطبيق / تدريب النموذج Fit/train the model.

```
#DataFlair - Train the model
model.fit(x_train,y_train)
```

لقطة شاشة الإخراج:

```
[9]: #DataFlair - Train the model
     model.fit(x_train,y_train)

[9]: MLPClassifier(activation='relu', alpha=0.01, batch_size=256, beta_1=0.9,
                   beta_2=0.999, early_stopping=False, epsilon=1e-08,
                   hidden_layer_sizes=(300,), learning_rate='adaptive',
                   learning_rate_init=0.001, max_iter=500, momentum=0.9,
                   n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
                   random_state=None, shuffle=True, solver='adam', tol=0.0001,
                   validation_fraction=0.1, verbose=False, warm_start=False)
```

الخطوة 10: دعونا نتوقع قيم مجموعة الاختبار. هذا يعطينا y_pred (المشاعر المتوقعة للميزات الموجودةفي مجموعة الاختبار).

```
#DataFlair - Predict for the test set
y_pred=model.predict(x_test)
```

لقطة الشاشة:

```
[10]: #DataFlair - Predict for the test set
      y_pred=model.predict(x_test)
```

الخطوة 11: لحساب دقة نموذجنا، سنقوم باستدعاء دالة accuracy_score() التي قمنا باستيرادها من sklearn. أخيرًا، سنقرب الدقة إلى منزلتين عشريتين ونطبعها.

```
#DataFlair - Calculate the accuracy of our model
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

#DataFlair - Print the accuracy
print("Accuracy: {:.2f}%".format(accuracy*100))
```

لقطة الشاشة:

```
[11]: #DataFlair - Calculate the accuracy of our model
      accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

      #DataFlair - Print the accuracy
      print("Accuracy: {:.2f}%".format(accuracy*100))

      Accuracy: 72.40%
```

```
[ ]:
```

الملخص

في مشروع بايثون المصغر هذا، تعلمنا التعرف على المشاعر من الكلام. استخدمنا مصنف MLPC لهذا الغرض واستفدنا من مكتبة librosa لاستخراج الميزات منه. كما سترى، قدم النموذج دقة 72.4٪. هذا جيد بما يكفي بالنسبة لنا حتى الآن.

17) كشف الاحتيال على بطاقة الائتمان باستخدام التعلم العميق
Credit Card Fraud Detection using Deep Learning

من المهم أن تكون شركات بطاقات الائتمان قادرة على التعرف على معاملات بطاقات الائتمان الاحتيالية حتى لا يتم تحصيل رسوم من العملاء مقابل العناصر التي لم يشتروها.

يحدث الاحتيال fraud عادةً عندما يحصل شخص ما على أرقام بطاقة الائتمان أو الخصم الخاصة بك من خلال مواقع الويب غير المحمية أو من خلال مخطط لسرقة الهوية من أجل الحصول على أموال أو ممتلكات بطريقة احتيالية. نظرًا لتكرار حدوثها والضرر المحتمل الذي قد تسببه لكل من الأفراد والمؤسسات المالية، فمن الأهمية بمكان اتخاذ خطوات وقائية بالإضافة إلى التعرف على متى تكون المعاملة احتيالية.

استيراد المكتبات الضرورية

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout,
BatchNormalization
from tensorflow.keras.layers import Conv1D, MaxPool1D
from tensorflow.keras.optimizers import Adam

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler    # in order to scale data
from sklearn.metrics import classification_report,accuracy_score


import warnings as wr
wr.filterwarnings("ignore")
```

قراءة مجموعة البيانات

- 492 عملية احتيال من أصل 284807 صفقة.
- الميزات V1 - V28 هي نتيجة لتحويل PCA وهي ببساطة تمثيلات رقمية.
- "Amount" هو قيمة المعاملة بالدولار.
- متغير "Time" هو مقدار الوقت المنقضي من وقت حدوث المعاملة الأولى.
- الاحتيال = 1، ليس الاحتيال = 0.

يمكن تنزيل مجموعة البيانات من الرابط.

يمكن استيراد مجموعة البيانات من خلال سطور التعليمات البرمجية التالية:

```
!wget -N "https://cainvas-
static.s3.amazonaws.com/media/user_data/DheerajPeru/creditcard.csv.zip
"
!unzip -o "creditcard.csv.zip"
!rm "creditcard.csv.zip"
```

التعرف على البيانات

```
data = pd.read_csv("https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-admin/creditcard.csv")
data.head(5)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.

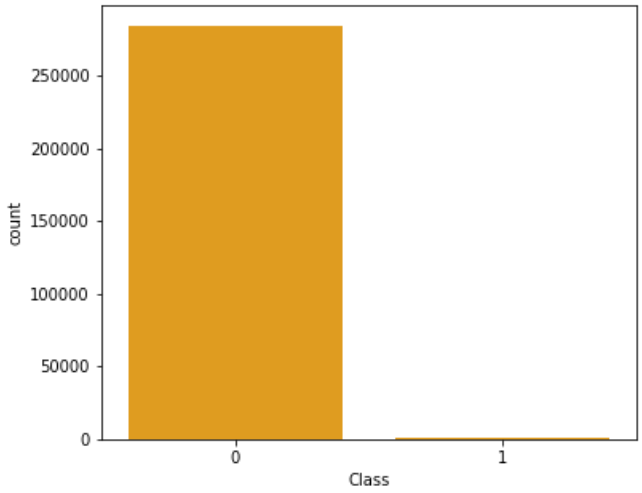
```
data.shape
```

```
(284807, 31)
```

التمثيل المرئي للبيانات

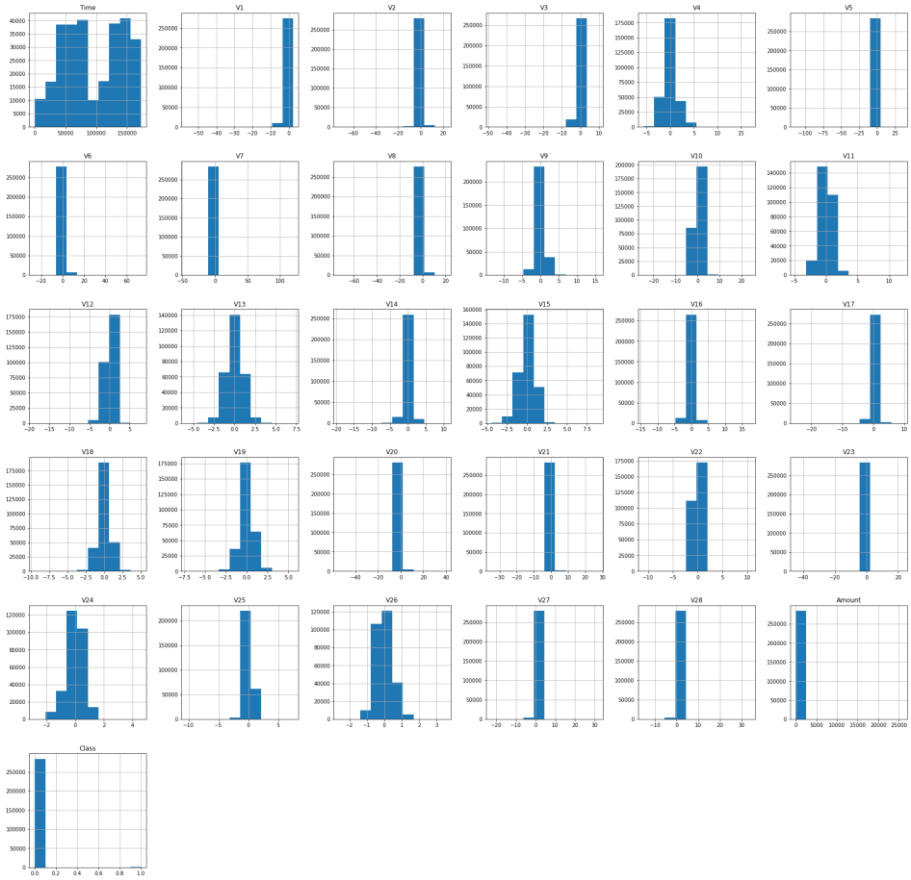
يتم عرض تصنيف العد لكل فئة على النحو التالي:

```
plt.figure(figsize = (6,5))
sns.countplot(data.Class, color = "orange")
plt.show()
```



يتم عرض الهيستوكرام لقيم كل عمود أدناه:

```
data.hist(figsize=(30,30))
plt.show()
```



المعالجة المسبقة للبيانات

```
fraud = data[data.Class == 1]
fraud
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
541	406.0	-2.312227	1.951992	-1.609851	3.997906	-0.522188	-1.426545	-2.537387	1.391657	-2.770089	...	0.517232	-0.035049	-0.465211	0.32
623	472.0	-3.043541	-3.157307	1.088463	2.288644	1.359805	-1.064823	0.325574	-0.067794	-0.270953	...	0.661696	0.435477	1.375966	-0.29
4920	4462.0	-2.303350	1.759247	-0.359745	2.330243	-0.821628	-0.075788	0.562320	-0.399147	-0.238253	...	-0.294166	-0.932391	0.172726	-0.08
6108	6986.0	-4.397974	1.358367	-2.592844	2.679787	-1.128131	-1.706536	-3.496197	-0.248778	-0.247768	...	0.573574	0.176968	-0.436207	-0.05
6329	7519.0	1.234235	3.019740	-4.304597	4.732795	3.624201	-1.357746	1.713445	-0.496358	-1.282858	...	-0.379068	-0.704181	-0.656805	-1.63
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	...	0.778584	-0.319189	0.639419	-0.29
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	...	0.370612	0.028234	-0.145640	-0.08
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	...	0.751826	0.834108	0.190944	0.03
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	...	0.583276	-0.269209	-0.456108	-0.18
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	...	-0.164350	-0.295135	-0.072173	-0.45

492 rows × 31 columns

```
non_fraud = data[data.Class == 0]
non_fraud
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.

284315 rows × 31 columns

```
print("Shape of fraud data:", fraud.shape)
print("Shape of non-fraus data:", non_fraud.shape)
```

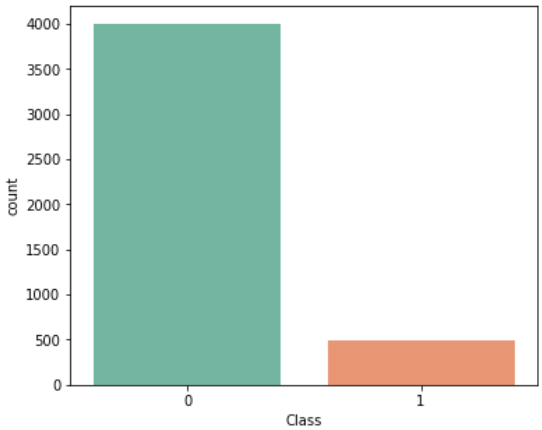
```
Shape of fraud data: (492, 31)
Shape of non-fraus data: (284315, 31)
```

موازنة مجموعة البيانات

نظرًا لأن مجموعة البيانات غير متوازنة إلى حد كبير، فهناك حاجة إلى موازنة ذلك، من أجل جعل الفئات قريبة جداً.

```
fraud = data[data.Class == 1]
non_fraud = data[data.Class == 0]
non_fraud_balanced = non_fraud.sample(4000)
balanced_data = fraud.append(non_fraud_balanced, ignore_index = True)
```

```
plt.figure(figsize = (6,5))
sns.countplot(y, palette="Set2")
plt.show()
```



جزء التدريب والاختبار

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2,
random_state = 42)
xtrain.shape
```

```
(3593, 30)
```

```
xtest.shape
```

```
(899, 30)
```

توحيد البيانات

نحتاج إلى توحيد مجموعة بيانات الإدخال نظرًا لوجود اختلافات كبيرة بين نطاقات كل ميزة.

```
scaled_xtrain = scaler.fit_transform(xtrain)
scaled_xtest = scaler.fit_transform(xtest)
```

إعادة تشكيل البيانات

```
scaled_xtrain3d =
scaled_xtrain.reshape(scaled_xtrain.shape[0],scaled_xtrain.shape[1],1)
scaled_xtest3d =
scaled_xtest.reshape(scaled_xtest.shape[0],scaled_xtest.shape[1],1)

scaled_xtrain3d.shape, scaled_xtest3d.shape
```

```
((3593, 30, 1), (899, 30, 1))
```

بناء النموذج

تم بناء شبكة CNN العصبية على النحو التالي.

```
# First Layer:

cnn = Sequential()
cnn.add(Conv1D(32, 2, activation = "relu", input_shape = (30,1)))
cnn.add(Dropout(0.1))

# Second Layer:

cnn.add(BatchNormalization()) # Batch normalization is a technique for
training very deep neural networks
                              # that standardizes the inputs to a layer for
each mini-batch. This
                              # has the effect of stabilizing the learning
process and dramatically
                              # reducing the number of training epochs
required to train deep networks

cnn.add(Conv1D(64, 2, activation = "relu"))
cnn.add(Dropout(0.2))         # prevents over-fitting (randomly remove some
neurons)
```

```
# Flattening Layer:

cnn.add(Flatten())
cnn.add(Dropout(0.4))
cnn.add(Dense(64, activation = "relu"))
cnn.add(Dropout(0.5))
# Last Layer:

cnn.add(Dense(1, activation = "sigmoid"))

cnn.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 29, 32)            96
_________________________________________________________________
dropout (Dropout)            (None, 29, 32)            0
_________________________________________________________________
batch_normalization (BatchNo (None, 29, 32)            128
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 28, 64)            4160
_________________________________________________________________
dropout_1 (Dropout)          (None, 28, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 1792)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1792)              0
_________________________________________________________________
dense (Dense)                (None, 64)                114752
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 119,201
Trainable params: 119,137
Non-trainable params: 64
_________________________________________________________________
```

تجميع وتدريب النموذج

```
cnn.compile(optimizer = Adam(lr=0.0001), loss = "binary_crossentropy",
metrics = ["accuracy"])
history = cnn.fit(scaled_xtrain3d, ytrain, epochs = 20,
validation_data=(scaled_xtest3d, ytest), verbose=1)
```

```
113/113 [==============================] - 0s 3ms/step - loss: 0.0774 - accuracy: 0.9775 - val_loss: 0.0556 - val_accuracy: 0.9878
Epoch 16/20
113/113 [==============================] - 0s 3ms/step - loss: 0.0789 - accuracy: 0.9786 - val_loss: 0.0568 - val_accuracy: 0.9878
Epoch 17/20
113/113 [==============================] - 0s 3ms/step - loss: 0.0781 - accuracy: 0.9786 - val_loss: 0.0556 - val_accuracy: 0.9867
Epoch 18/20
113/113 [==============================] - 0s 3ms/step - loss: 0.0742 - accuracy: 0.9800 - val_loss: 0.0553 - val_accuracy: 0.9855
Epoch 19/20
113/113 [==============================] - 0s 3ms/step - loss: 0.0750 - accuracy: 0.9786 - val_loss: 0.0550 - val_accuracy: 0.9867
Epoch 20/20
113/113 [==============================] - 0s 3ms/step - loss: 0.0684 - accuracy: 0.9802 - val_loss: 0.0545 - val_accuracy: 0.9867
```

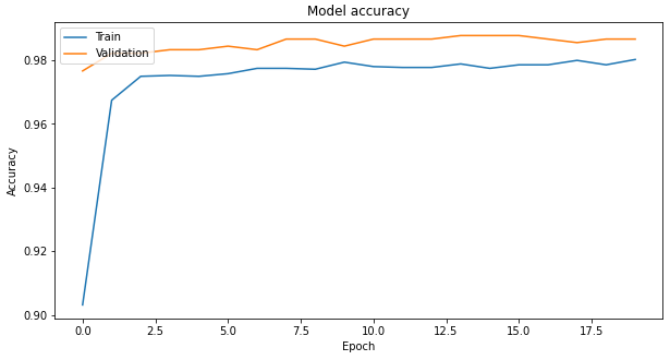
تم تدريب النموذج ببيانات مقاسة ومعاد تشكيلها، مع فترات = 20.

يتم تحقيق دقة تدريب تصل إلى ~ 98٪ من خلال الشبكة المبنية.

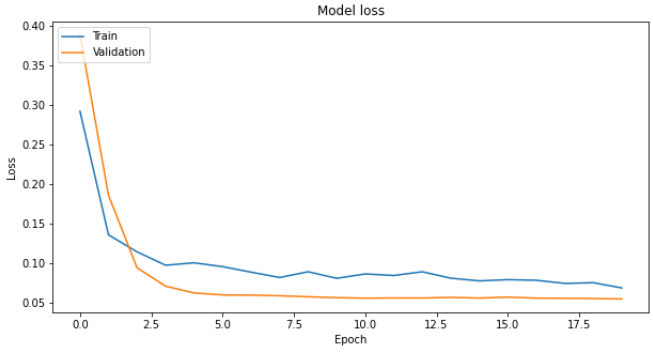
رسم النتائج

الرسم البياني للفترات مقابل الدقة كما هو موضح أدناه للنموذج:

```
fig, ax1 = plt.subplots(figsize= (10, 5))
plt.plot(history.history["accuracy"])
plt.plot(history.history["val_accuracy"])
plt.title("Model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Train", "Validation"], loc = "upper left")
plt.show()
```

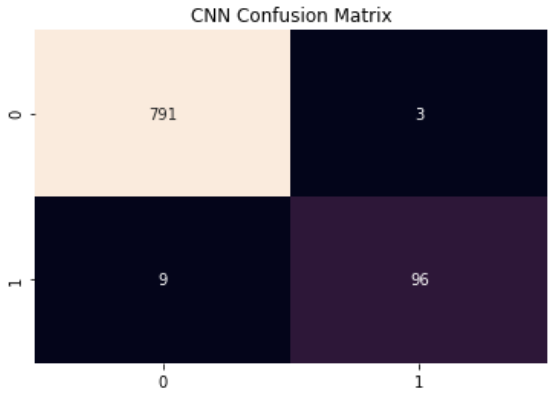



```
fig, ax1 = plt.subplots(figsize= (10, 5))
plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.title("Model loss")
plt.ylabel("Loss")
plt.xlabel("Epoch")
plt.legend(["Train", "Validation"], loc = "upper left")
plt.show()
```



تقييم النموذج

```
from sklearn.metrics import confusion_matrix
cnn_predictions = cnn.predict_classes(scaled_xtest3d)
confusion_matrix = confusion_matrix(ytest, cnn_predictions)
sns.heatmap(confusion_matrix, annot=True, fmt="d", cbar = False)
plt.title("CNN Confusion Matrix")
plt.show()
```




```
accuracy_score(ytest, cnn_predictions)
```

```
0.9866518353726362
```

```
from sklearn.metrics import precision_recall_fscore_support as score
precision, recall, fscore, support = score(ytest, cnn_predictions)
print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))
print('support: {}'.format(support))
```

```
precision: [0.98875    0.96969697]
recall: [0.99622166 0.91428571]
fscore: [0.99247177 0.94117647]
support: [794 105]
```

حفظ النموذج

```
cnn.save('fraud_detection_model.h5')
```

الملخص

تم تصميم نموذج التعلم العميق للكشف عن الاحتيال على بطاقات الائتمان بدقة تصل إلى 98٪.

18) تصنيف سرطان الثدي مع التعلم العميق Breast Cancer Classification with Deep Learning

ما هو التعلم العميق؟

نهج مكثف للتعلم الآلي، التعلم العميق مستوحى من عمل الدماغ البشري وشبكاته العصبية البيولوجية. تتكون البنى مثل الشبكات العصبية العميقة والشبكات العصبية المتكررة والشبكات العصبية التلافيفية وشبكات الاعتقاد العميق من طبقات متعددة لتمرير البيانات قبل إنتاج المخرجاتفي النهاية. يعمل التعلم العميق على تحسين الذكاء الاصطناعي وجعل العديد من تطبيقاته ممكنة؛ يتم تطبيقه على العديد من مجالات الرؤية الحاسوبية، والتعرف على الكلام، ومعالجة اللغة الطبيعية، والتعرف على الصوت، وتصميم الأدوية.

ما هو Keras؟

Keras هي مكتبة شبكة عصبية مفتوحة المصدر مكتوبة بلغة بايثون. إنها واجهة برمجة تطبيقات API عالية المستوى ويمكن تشغيلها فوق TensorFlow وCNTK وTheano. تهدف Keras إلى تمكين التجارب السريعة والنماذج الأولية أثناء التشغيل بسلاسة على وحدة المعالجة المركزية ووحدة معالجة الرسومات. إنه سهل الاستخدام وقابل للتوسيع.

تصنيف سرطان الثدي - الهدف

لبناء مُصنِّف سرطان الثدي على مجموعة بيانات IDC يمكنها تصنيف صورة الأنسجة بدقة على أنها حميدة benign أو خبيثة malignant.

تصنيف سرطان الثدي - حول مشروع بايثون

في هذا المشروعفي بايثون، سننشئ مصنفًا للتدريب على 80٪ من مجموعة بيانات صور أنسجة سرطان الثدي. من هذا، سنحتفظ بـ 10٪ من البيانات للتحقق من صحتها. باستخدام Keras، سنحدد CNN (الشبكة العصبية التلافيفية)، ونسميها CancerNet، وندربها على صورنا. سنشتق بعد ذلك مصفوفة ارتباك confusion matrix لتحليل أداء النموذج.

IDC هو Invasive Ductal Carcinoma سرطان القنوات الغازية. السرطان الذي يتطورفي قناة الحليب ويغزو أنسجة الثدي الليفية أو الدهنية خارج القناة؛ هو الشكل الأكثر شيوعًا لسرطان الثدي ويشكل 80٪ من جميع تشخيصات سرطان الثدي. وعلم الأنسجة هو دراسة التركيب المجهري للأنسجة.

مجموعة البيانات

سنستخدم مجموعة البيانات IDC_regular (مجموعة بيانات صورة أنسجة سرطان الثدي) من Kaggle. تحتوي مجموعة البيانات هذه على 2،77،524 رقعة بحجم 50 × 50 مستخرجة من 162 صورة شريحة كاملة لعينات سرطان الثدي الممسوحة ضوئيًا عند x40. من بين هؤلاء، 1،98،738 نتيجة اختبار سلبية و78،786 نتيجة اختبار IDC إيجابية. مجموعة البيانات متاحة

في المجال العام ويمكنك تنزيلها من هنا. ستحتاج إلى 3.02 غيغابايت على الأقل من مساحة القرص لهذا الغرض.

تبدو أسماء الملفاتفي مجموعة البيانات هذه كما يلي:

8863_idx5_x451_y1451_class0

هنا، 8863_idx5 هو معرف المريض patient ID، و451 و1451 هما إحداثيات x و y للقطع، و0 هو تسمية الفئة (يشير 0 إلى عدم وجود IDC).

المتطلبات الأساسية

ستحتاج إلى تثبيت بعض حزم بايثون لتتمكن من تشغيل مشروع بايثون المتقدم هذا. يمكنك القيام بذلك باستخدام pip-

```
pip install numpy opencv-python pillow tensorflow keras imutils scikit-
learn matplotlib
```

خطوات لمشروع متقدم في بايثون - تصنيف سرطان الثدي

الخطوة 1: قم بتنزيل هذا الملف المضغوط. قم بفك ضغطهفي الموقع المفضل لديك، وانطلق إلى هناك.

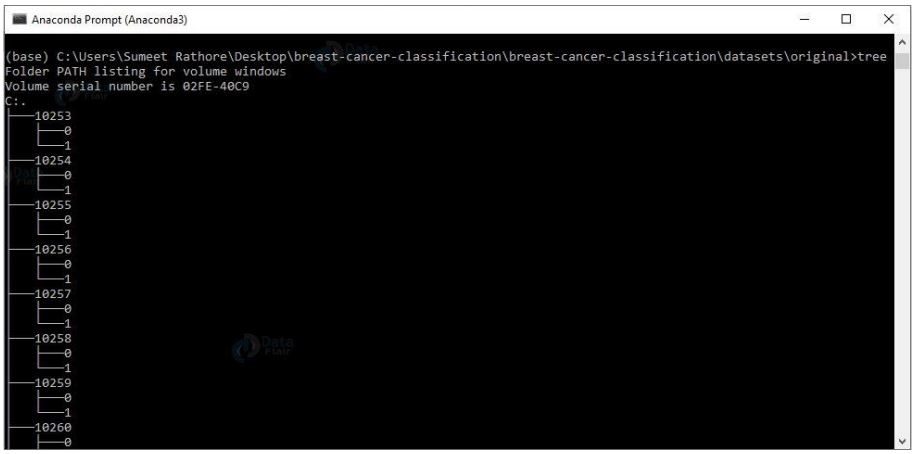
الخطوة 2: الآن، داخل دليل تصنيف سرطان الثدي الداخلي، أنشئ مجموعات بيانات الدليل- بداخله، أنشئ الدليل الأصلي:

```
mkdir datasets
mkdir datasets\original
```

الخطوة 3: قم بتنزيل مجموعة البيانات.

الخطوة 4: قم بفك ضغط مجموعة البياناتفي الدليل الأصلي. لمراقبة بُنية هذا الدليل، سنستخدم أمر الشجرة:

```
cd breast-cancer-classification\breast-cancer-
classification\datasets\original
tree
```

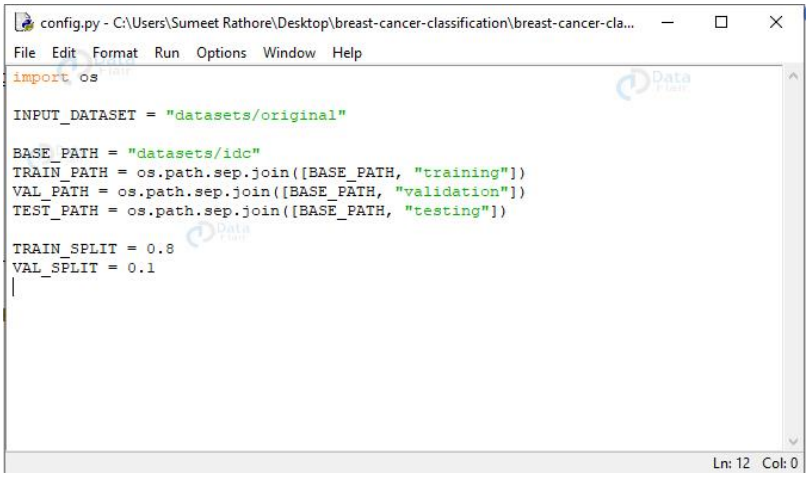


لدينا دليل لكل مريض معرف patient ID. وفي كل دليل من هذا القبيل، لدينا دليلين 0 و1 للصور ذات المحتوى الحميد والخبيث.

config.py:

يتضمن هذا بعض التهيئة التي سنحتاجها لبناء مجموعة البيانات وتدريب النموذج. ستجد هذا في دليل cancernet.

```
import os

INPUT_DATASET = "datasets/original"

BASE_PATH = "datasets/idc"
TRAIN_PATH = os.path.sep.join([BASE_PATH, "training"])
VAL_PATH = os.path.sep.join([BASE_PATH, "validation"])
TEST_PATH = os.path.sep.join([BASE_PATH, "testing"])

TRAIN_SPLIT = 0.8
VAL_SPLIT = 0.1
```



هنا، نعلن المسار إلى مجموعة بيانات الإدخال (datasets/original)، والمسار الخاص بالدليل الجديد (datasets/idc)، ومسارات أدلة التدريب والتحقق من الصحة والاختبار باستخدام المسار الأساسي. نعلن أيضًا أنه سيتم استخدام 80٪ من مجموعة البيانات بأكملها للتدريب، وسيتم استخدام 10٪ منها للتحقق.

build_dataset.py:

سيؤدي هذا إلى تقسيم مجموعة البيانات الخاصة بنا إلى مجموعات التدريب والتحقق والاختبار في النسبة المذكورة أعلاه – 80٪ للتدريب (منها 10٪ للتحقق من الصحة) و20٪ للاختبار. باستخدام ImageDataGenerator من Keras، سنقوم باستخراج مجموعات من الصور لتجنب توفير مساحة لمجموعة البيانات بأكملهافي الذاكرة مرة واحدة.

```
from cancernet import config
from imutils import paths
import random, shutil, os

originalPaths=list(paths.list_images(config.INPUT_DATASET))
random.seed(7)
random.shuffle(originalPaths)

index=int(len(originalPaths)*config.TRAIN_SPLIT)
trainPaths=originalPaths[:index]
testPaths=originalPaths[index:]

index=int(len(trainPaths)*config.VAL_SPLIT)
valPaths=trainPaths[:index]
trainPaths=trainPaths[index:]

datasets=[("training", trainPaths, config.TRAIN_PATH),
          ("validation", valPaths, config.VAL_PATH),
          ("testing", testPaths, config.TEST_PATH)
]

for (setType, originalPaths, basePath) in datasets:
        print(f'Building {setType} set')

        if not os.path.exists(basePath):
                print(f'Building directory {base_path}')
                os.makedirs(basePath)

        for path in originalPaths:
                file=path.split(os.path.sep)[-1]
                label=file[-5:-4]

                labelPath=os.path.sep.join([basePath,label])
                if not os.path.exists(labelPath):
                        print(f'Building directory {labelPath}')
                        os.makedirs(labelPath)

                newPath=os.path.sep.join([labelPath, file])
                shutil.copy2(inputPath, newPath)
```

build_dataset.py - C:\Users\Sumeet Rathore\Desktop\breast-cancer-classification\breast-can...

File Edit Format Run Options Window Help

```
from cancernet import config
from imutils import paths
import random, shutil, os

originalPaths=list(paths.list_images(config.INPUT_DATASET))
random.seed(7)
random.shuffle(originalPaths)

index=int(len(originalPaths)*config.TRAIN_SPLIT)
trainPaths=originalPaths[:index]
testPaths=originalPaths[index:]

index=int(len(trainPaths)*config.VAL_SPLIT)
valPaths=trainPaths[:index]
trainPaths=trainPaths[index:]

datasets=[("training", trainPaths, config.TRAIN_PATH),
          ("validation", valPaths, config.VAL_PATH),
          ("testing", testPaths, config.TEST_PATH)
]

for (setType, originalPaths, basePath) in datasets:
        print(f'Building {setType} set')

        if not os.path.exists(basePath):
                print(f'Building directory {basePath}')
                os.makedirs(basePath)

        for path in originalPaths:
                file=path.split(os.path.sep)[-1]
                label=file[-5:-4]

                labelPath=os.path.sep.join([basePath,label])
                if not os.path.exists(labelPath):
                        print(f'Building directory {labelPath}')
                        os.makedirs(labelPath)

                newPath=os.path.sep.join([labelPath, file])
                shutil.copy2(path, newPath)
```

Ln: 40 Col: 0

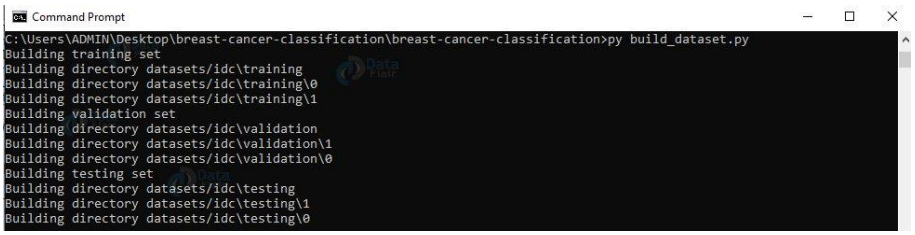
في هذا، سنستورد من config وimutils وrandom وshutil وos. سننشئ قائمة بالمسارات الأصلية للصور، ثم نغير القائمة. بعد ذلك، نحسب فهرسًا بضرب طول هذه القائمةفي 0.8 حتى نتمكن من تقسيم هذه القائمة للحصول على قوائم فرعية لمجموعات بيانات التدريب والاختبار. بعد ذلك، نحسب أيضًا فهرسًا يوفر 10٪ من القائمة لمجموعة بيانات التدريب للتحقق من الصحة والاحتفاظ بالباقي للتدريب نفسه.

الآن، مجموعات البيانات عبارة عن صفوف tuples تحتوي على مجموعات للحصول على معلومات حول مجموعات التدريب والتحقق من الصحة والاختبار. هذه تحمل المسارات والمسار الأساسي لكل منها. لكل setType ومسار ومسار أساسيفي هذه القائمة، سنطبع، على سبيل المثال، "Building testing set ". إذا كان المسار الأساسي غير موجود، فسننشئ الدليل. ولكل مسارفي originalPaths، سنستخرج اسم الملف وتسمية الفئة. سننشئ المسار إلى دليل

التصنيف (0 أو 1) ـ إذا لم يكن موجودًا بعد، فسننشئ هذا الدليل صراحةً. الآن، سنبني المسار إلى الصورة الناتجة ونسخ الصورة هنا ـ حيث تنتمي.

الخطوة 5: قم بتشغيل السكريبت build_dataset.py:

```
py build_dataset.py
```



cancernet.py:

الشبكة التي سنبنيها ستكون CNN (شبكة عصبية تلافيفية) ونطلق عليها اسم CancerNet. تقوم هذه الشبكة بالعمليات التالية:

- استخدم فلاتر CONV 3 × 3.
- ضع هذه الفلاتر فوق بعضها البعض.
- استخدم تجميع حد الاقصى max-pooling.
- استخدم التفاف عميق قابل للفصل depthwise separable convolution (أكثر كفاءة ، يشغل ذاكرة أقل)

```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import SeparableConv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras import backend as K

class CancerNet:
  @staticmethod
  def build(width,height,depth,classes):
    model=Sequential()
    shape=(height,width,depth)
    channelDim=-1

    if K.image_data_format()=="channels_first":
      shape=(depth,height,width)
      channelDim=1

    model.add(SeparableConv2D(32, (3,3),
padding="same",input_shape=shape))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=channelDim))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))
```

```
    model.add(SeparableConv2D(64, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=channelDim))
    model.add(SeparableConv2D(64, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=channelDim))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))

    model.add(SeparableConv2D(128, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=channelDim))
    model.add(SeparableConv2D(128, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=channelDim))
    model.add(SeparableConv2D(128, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=channelDim))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(256))
    model.add(Activation("relu"))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))

    model.add(Dense(classes))
    model.add(Activation("softmax"))

    return model
```

cancernet.py - C:\Users\Sumeet Rathore\Desktop\breast-cancer-classification\breast-cancer-classification\cancernet\canc...

File Edit Format Run Options Window Help

```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import SeparableConv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras import backend as K

class CancerNet:
        @staticmethod
        def build(width,height,depth,classes):
                model=Sequential()
                shape=(height,width,depth)
                channelDim=-1

                if K.image_data_format()=="channels_first":
                        shape=(depth,height,width)
                        channelDim=1

                model.add(SeparableConv2D(32, (3,3), padding="same",input_shape=shape))
                model.add(Activation("relu"))
                model.add(BatchNormalization(axis=channelDim))
                model.add(MaxPooling2D(pool_size=(2,2)))
                model.add(Dropout(0.25))

                model.add(SeparableConv2D(64, (3,3), padding="same"))
                model.add(Activation("relu"))
                model.add(BatchNormalization(axis=channelDim))
                model.add(SeparableConv2D(64, (3,3), padding="same"))
                model.add(Activation("relu"))
                model.add(BatchNormalization(axis=channelDim))
                model.add(MaxPooling2D(pool_size=(2,2)))
                model.add(Dropout(0.25))

                model.add(SeparableConv2D(128, (3,3), padding="same"))
                model.add(Activation("relu"))
                model.add(BatchNormalization(axis=channelDim))
                model.add(SeparableConv2D(128, (3,3), padding="same"))
                model.add(Activation("relu"))
```

```
model.add(BatchNormalization(axis=channelDim))
model.add(SeparableConv2D(128, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=channelDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(classes))
model.add(Activation("softmax"))

return model
```

Ln: 40 Col: 36

نحن نستخدم API التسلسلي لبناء CancerNet وSeparableConv2D لتنفيذ التلافيفات العميقة. تحتوي فئة CancerNet على طريقة ثابتة لبناء يأخذ أربع معلمات ـ عرض الصورة وارتفاعها، وعمقها (عدد قنوات الألوانفي كل صورة)، وعدد الفئات التي ستتنبأ بها الشبكة، والتي بالنسبة لنا، 2 (0 و 1).

في هذه الطريقة، نقوم بتهيئة النموذج والشكل. عند استخدام channel_first، نقوم بتحديث الشكل وبُعد القناة.

الآن، سنقوم بتحديد ثلاثة طبقات DEPTHWISE_CONV => RELU => POOL ؛ كل مع تكديس أعلى وعدد أكبر من الفلاتر. ينتج المصنف softmax النسب المئوية للتنبؤ لكل فئة.في النهاية، نعيد النموذج.

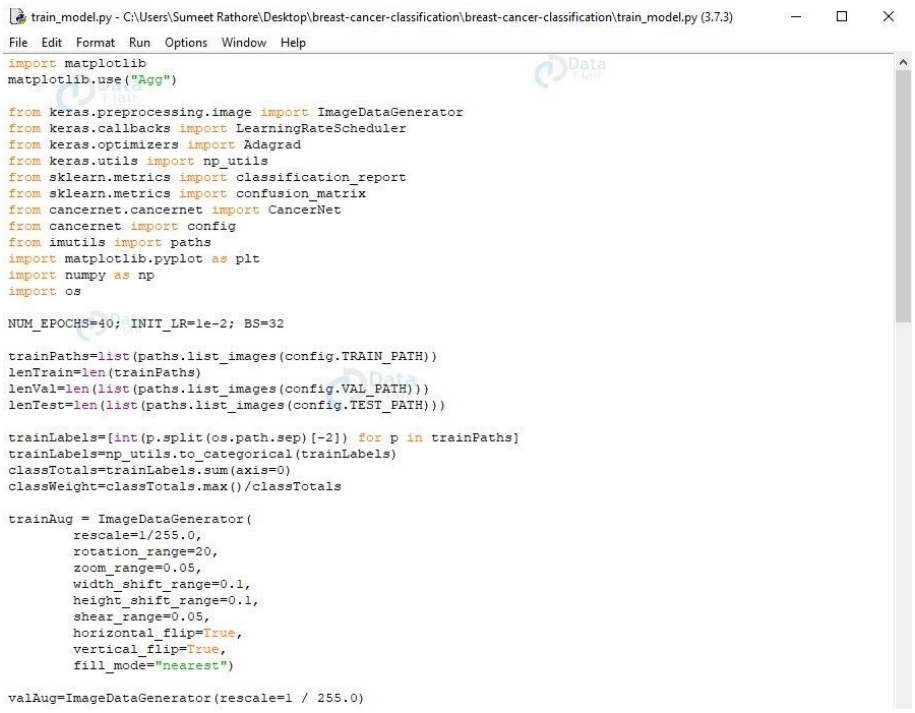
train_model.py:

هذا تدريب وتقييم نموذجنا. هنا، سنستورد من keras و sklearn و cancernet و config و imutils و matplotlib و numpy و os.

```
import matplotlib
matplotlib.use("Agg")

from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import LearningRateScheduler
from keras.optimizers import Adagrad
from keras.utils import np_utils
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from cancernet.cancernet import CancerNet
from cancernet import config
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

NUM_EPOCHS=40; INIT_LR=1e-2; BS=32

trainPaths=list(paths.list_images(config.TRAIN_PATH))
lenTrain=len(trainPaths)
lenVal=len(list(paths.list_images(config.VAL_PATH)))
lenTest=len(list(paths.list_images(config.TEST_PATH)))
```

```
trainLabels=[int(p.split(os.path.sep)[-2]) for p in trainPaths]
trainLabels=np_utils.to_categorical(trainLabels)
classTotals=trainLabels.sum(axis=0)
classWeight=classTotals.max()/classTotals

trainAug = ImageDataGenerator(
  rescale=1/255.0,
  rotation_range=20,
  zoom_range=0.05,
  width_shift_range=0.1,
  height_shift_range=0.1,
  shear_range=0.05,
  horizontal_flip=True,
  vertical_flip=True,
  fill_mode="nearest")

valAug=ImageDataGenerator(rescale=1 / 255.0)

trainGen = trainAug.flow_from_directory(
  config.TRAIN_PATH,
  class_mode="categorical",
  target_size=(48,48),
  color_mode="rgb",
  shuffle=True,
  batch_size=BS)
valGen = valAug.flow_from_directory(
  config.VAL_PATH,
  class_mode="categorical",
  target_size=(48,48),
  color_mode="rgb",
  shuffle=False,
  batch_size=BS)
testGen = valAug.flow_from_directory(
  config.TEST_PATH,
  class_mode="categorical",
  target_size=(48,48),
  color_mode="rgb",
  shuffle=False,
  batch_size=BS)

model=CancerNet.build(width=48,height=48,depth=3,classes=2)
opt=Adagrad(lr=INIT_LR,decay=INIT_LR/NUM_EPOCHS)
model.compile(loss="binary_crossentropy",optimizer=opt,metrics=["accur
acy"])


M=model.fit_generator(
  trainGen,
  steps_per_epoch=lenTrain//BS,
  validation_data=valGen,
  validation_steps=lenVal//BS,
  class_weight=classWeight,
  epochs=NUM_EPOCHS)

print("Now evaluating the model")
testGen.reset()
pred_indices=model.predict_generator(testGen,steps=(lenTest//BS)+1)

pred_indices=np.argmax(pred_indices,axis=1)

print(classification_report(testGen.classes, pred_indices,
target_names=testGen.class_indices.keys()))
```

```
cm=confusion_matrix(testGen.classes,pred_indices)
total=sum(sum(cm))
accuracy=(cm[0,0]+cm[1,1])/total
specificity=cm[1,1]/(cm[1,0]+cm[1,1])
sensitivity=cm[0,0]/(cm[0,0]+cm[0,1])
print(cm)
print(f'Accuracy: {accuracy}')
print(f'Specificity: {specificity}')
print(f'Sensitivity: {sensitivity}')

N = NUM_EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0,N), M.history["loss"], label="train_loss")
plt.plot(np.arange(0,N), M.history["val_loss"], label="val_loss")
plt.plot(np.arange(0,N), M.history["acc"], label="train_acc")
plt.plot(np.arange(0,N), M.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on the IDC Dataset")
plt.xlabel("Epoch No.")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig('plot.png')
```


train_model.py - C:\Users\Sumeet Rathore\Desktop\breast-cancer-classification\breast-cancer-classification\train_model.py (3.7.3)

File Edit Format Run Options Window Help

```
import matplotlib
matplotlib.use("Agg")

from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import LearningRateScheduler
from keras.optimizers import Adagrad
from keras.utils import np_utils
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from cancernet.cancernet import CancerNet
from cancernet import config
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

NUM_EPOCHS=40; INIT_LR=1e-2; BS=32

trainPaths=list(paths.list_images(config.TRAIN_PATH))
lenTrain=len(trainPaths)
lenVal=len(list(paths.list_images(config.VAL_PATH)))
lenTest=len(list(paths.list_images(config.TEST_PATH)))

trainLabels=[int(p.split(os.path.sep)[-2]) for p in trainPaths]
trainLabels=np_utils.to_categorical(trainLabels)
classTotals=trainLabels.sum(axis=0)
classWeight=classTotals.max()/classTotals

trainAug = ImageDataGenerator(
        rescale=1/255.0,
        rotation_range=20,
        zoom_range=0.05,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.05,
        horizontal_flip=True,
        vertical_flip=True,
        fill_mode="nearest")

valAug=ImageDataGenerator(rescale=1 / 255.0)
```

```
trainGen = trainAug.flow_from_directory(
        config.TRAIN_PATH,
        class_mode="categorical",
        target_size=(48,48),
        color_mode="rgb",
        shuffle=True,
        batch_size=BS)
valGen = valAug.flow_from_directory(
        config.VAL_PATH,
        class_mode="categorical",
        target_size=(48,48),
        color_mode="rgb",
        shuffle=False,
        batch_size=BS)
testGen = valAug.flow_from_directory(
        config.TEST_PATH,
        class_mode="categorical",
        target_size=(48,48),
        color_mode="rgb",
        shuffle=False,
        batch_size=BS)

model=CancerNet.build(width=48,height=48,depth=3,classes=2)
opt=Adagrad(lr=INIT_LR,decay=INIT_LR/NUM_EPOCHS)
model.compile(loss="binary_crossentropy",optimizer=opt,metrics=["accuracy"])


M=model.fit_generator(
        trainGen,
        steps_per_epoch=lenTrain//BS,
        validation_data=valGen,
        validation_steps=lenVal//BS,
        class_weight=classWeight,
        epochs=NUM_EPOCHS)

print("Now evaluating the model")
testGen.reset()
pred_indices=model.predict_generator(testGen,steps=(lenTest//BS)+1)

pred_indices=np.argmax(pred_indices,axis=1)
```

```
M=model.fit_generator(
        trainGen,
        steps_per_epoch=lenTrain//BS,
        validation_data=valGen,
        validation_steps=lenVal//BS,
        class_weight=classWeight,
        epochs=NUM_EPOCHS)

print("Now evaluating the model")
testGen.reset()
pred_indices=model.predict_generator(testGen,steps=(lenTest//BS)+1)

pred_indices=np.argmax(pred_indices,axis=1)

print(classification_report(testGen.classes, pred_indices, target_names=testGen.class_indices.keys()))

cm=confusion_matrix(testGen.classes,pred_indices)
total=sum(sum(cm))
accuracy=(cm[0,0]+cm[1,1])/total
specificity=cm[1,1]/(cm[1,0]+cm[1,1])
sensitivity=cm[0,0]/(cm[0,0]+cm[0,1])
print(cm)
print(f'Accuracy: {accuracy}')
print(f'Specificity: {specificity}')
print(f'Sensitivity: {sensitivity}')

N = NUM_EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0,N), M.history["loss"], label="train_loss")
plt.plot(np.arange(0,N), M.history["val_loss"], label="val_loss")
plt.plot(np.arange(0,N), M.history["acc"], label="train_acc")
plt.plot(np.arange(0,N), M.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on the IDC Dataset")
plt.xlabel("Epoch No.")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig('plot.png')
```

Ln: 81 Col: 43

في هذا السكريبت، أولاً، نقوم بتعيين القيم الأولية لعدد الفترات number of epochs ومعدل التعلم learning rate وحجم الدُفعة batch size. سنحصل على عدد المسارات في الأدلة الثلاثة

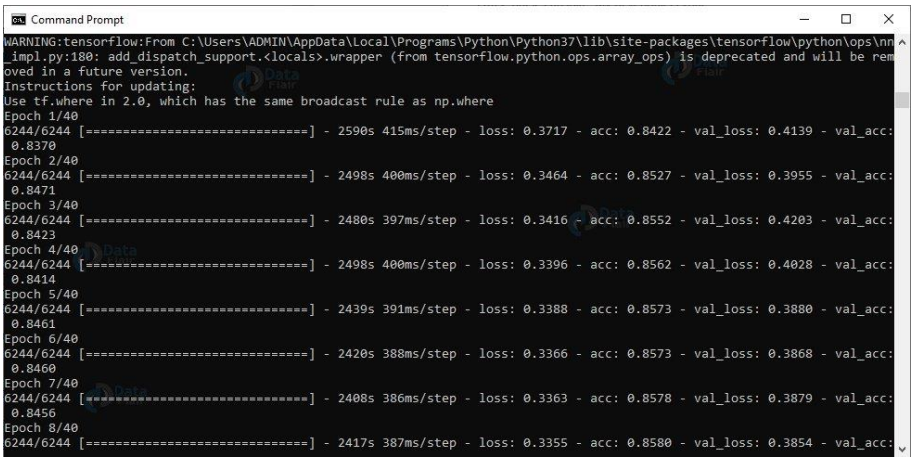
للتدريب والتحقق من الصحة والاختبار. بعد ذلك، سنحصل على وزن الفئة لبيانات التدريب حتى نتمكن من التعامل مع عدم التوازن.

الآن، نقوم بتهيئة كائن زيادة بيانات التدريب training data augmentation. هذه عملية تنظيم تساعد على تعميم generalize النموذج. هذا هو المكان الذي نقوم فيه بتعديل أمثلة التدريب بشكل طفيف لتجنب الحاجة إلى المزيد من بيانات التدريب. سنبدأ عملية التحقق من صحة واختبار كائنات زيادة البيانات.

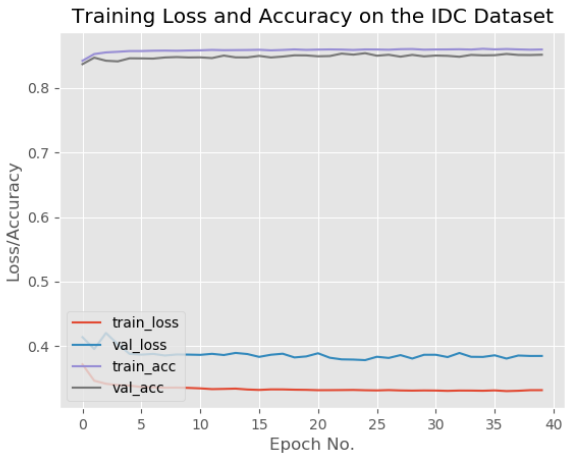
سنقوم بتهيئة مولدات التدريب والتحقق والاختبار حتى يتمكنوا من إنشاء مجموعات من الصور بحجم batch_size. بعد ذلك، سنقوم بتهيئة النموذج باستخدام مُحسِّن Adagrad وتجميعه باستخدام دالة خطأ binary_crossentropy. الآن، للتدريب fit النموذج، نقوم بإجراء استدعاء لـ fit_generator().

لقد نجحنا في تدريب نموذجنا. الآن، دعنا نقيم النموذج بناءً على بيانات الاختبار الخاصة بنا. سنقوم بإعادة تعيين المولد وإجراء تنبؤات بشأن البيانات. بعد ذلك، بالنسبة للصور من مجموعة الاختبار، نحصل على مؤشرات التسميات ذات الاحتمال الأكبر المتنبأ به المقابل. وسنقوم بعرض تقرير التصنيف classification report.

الآن، سنحسب مصفوفة الارتباك ونحصل على accuracy وspecificity وsensitivity، ونعرض جميع القيم. أخيرًا، سنرسم لخطأ التدريب ودقته.



```
Command Prompt
6244/6244 [==============================] - 2382s 381ms/step - loss: 0.3303 - acc: 0.8605 - val_loss: 0.3807 - val_acc:
 0.8529
Epoch 38/40
6244/6244 [==============================] - 2384s 382ms/step - loss: 0.3308 - acc: 0.8599 - val_loss: 0.3854 - val_acc:
 0.8513
Epoch 39/40
6244/6244 [==============================] - 2676s 429ms/step - loss: 0.3317 - acc: 0.8594 - val_loss: 0.3847 - val_acc:
 0.8511
Epoch 40/40
6244/6244 [==============================] - 2379s 381ms/step - loss: 0.3318 - acc: 0.8598 - val_loss: 0.3848 - val_acc:
 0.8516
Now evaluating the model
              precision    recall  f1-score   support

           0       0.91      0.87      0.89     39736
           1       0.72      0.79      0.75     15769

    accuracy                           0.85     55505
   macro avg       0.81      0.83      0.82     55505
weighted avg       0.86      0.85      0.85     55505

[[34757  4979]
 [ 3271 12498]]
Accuracy: 0.8513647419151428
Specificity: 0.792567696112626
Sensitivity: 0.8746980068451782

C:\Users\ADMIN\Desktop\breast-cancer-classification\breast-cancer-classification>
```



الملخص

في هذا المشروعفي بايثون، تعلمنا بناء مصنف لسرطان الثدي على مجموعة بيانات IDC (مع صور الأنسجة لسرطان الأقنية الغازية) وأنشأنا شبكة CancerNet لنفسه. استخدمنا Keras لتنفيذ نفس الشيء. آمل أن تكون قد استمتعت بمشروع بايثون هذا.

19) التنبؤ بأمراض القلب باستخدام التعلم العميق Heart Disease Prediction using Deep Learning

تغطي أمراض القلب Heart disease مجموعة من الحالات المختلفة التي يمكن أن تؤثر على قلبك. إنها واحدة من أكثر الأمراض تعقيداً للتنبؤ بها نظرًا لعدد العوامل المحتملةفي جسمك التي يمكن أن تؤدي إليها. يشكل تحديد أمراض القلب والتنبؤ بها تحديًا كبيرًا للأطباء والباحثين على حد سواء. سأحاول حل هذه المشكلة باستخدام التعلم العميق (DL) مع مجموعة البيانات العامة المتاحة هنافي UCI Machine Learning Repository. هيا بنا نبدأ.

يوجد 303 سجلاتفي مجموعة البيانات وتحتوي على 14 سمة مستمرة. الهدف هو توقع وجود مرض قلبي لدى المريض.

احتوت مجموعة البيانات على مجموعة أصلية من 76 ميزة تم تضييقها الآن إلى إجمالي 14 ميزة على النحو التالي:

- **age**: عمر الشخص بالسنوات.
- **sex**: جنس الشخص (1 = ذكر ، 0 = أنثى).
- **cp**: ألم الصدر الذي يعاني منه (القيمة 1: الذبحة الصدرية النموذجية ، القيمة 2: الذبحة الصدرية غير النمطية ، القيمة 3: الألم غير الزاوي ، القيمة 4: بدون أعراض)
- **trestbps**: ضغط دم الشخص أثناء الراحة.
- **chol**: قياس الكوليسترول لدى الشخص بالملجم / ديسيلتر.
- **fbs**: سكر دم الشخص أثناء الصيام (> 120 مجم / ديسيلتر ، 1 = صحيح ؛ 0 = خطأ)
- **restecg**: استراحة قياس تخطيط القلب الكهربائي (0 = طبيعي ، 1 = وجود شذوذفي موجة ST-T، 2 = إظهار تضخم البطين الأيسر المحتمل أو المحدد وفقًا لمعايير Estes).
- **thalach**: أقصى معدل لضربات القلب تم تحقيقه.
- **exang**: الذبحة الصدرية الناتجة عن التمرين (1 = نعم ؛ 0 = لا).
- **oldpeak**: اكتئاب ST الناجم عن التمرين بالنسبة للراحة (يتعلق "ST" بالمواقع على مخطط ECG).
- **slope**: منحدر الجزء ST من تمرين الذروة (القيمة 1: uploping ، القيمة 2: مسطح، القيمة 3: downsloping)
- **ca**: عدد الاوعية الرئيسية (0 ـ 3)

- **thal**: اضطرابفي الدم يسمى الثلاسيميا (3 = طبيعي ؛ 6 = خلل ثابت ؛ 7 = عيب قابل للعكس)
- **الهدف target**: أمراض القلب (0 = لا، 1 = نعم)

التحليل الاستكشافي

قبل أن نبدأفي تحليل البيانات التفصيلي، فلنبدأ بالتحليل الاستكشافي لفهم كيفية توزيع البيانات واستخراج المعرفة الأولية.

أول الأشياء، قم بتنزيل البيانات من الرابط المقدم أعلاه واستورد مجموعة البيانات إلى pandas DataFrame.

قم بتنزيل ملف csv من الرابط المذكور أعلاه وقم بتحميل ملف مجموعة بيانات csv.

```
from google.colab import files
uploaded = files.upload()
```

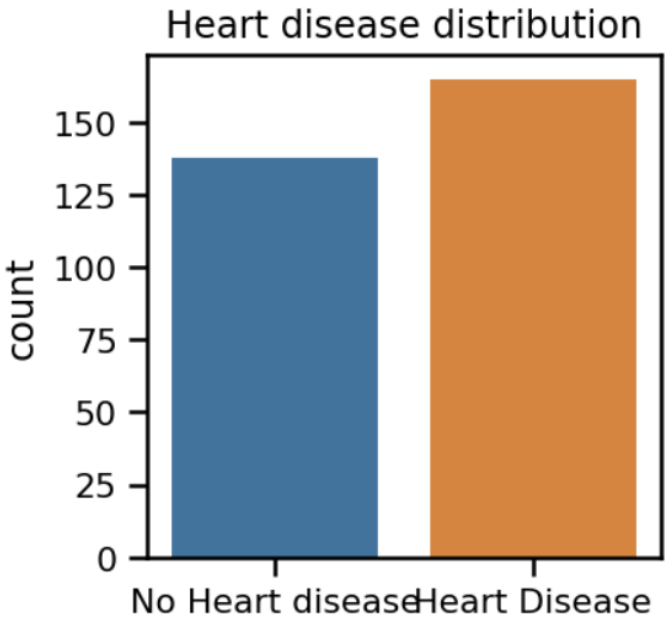
قم باستيراد مجموعة البيانات إلى pandas DataFrame وطباعة أول 20 سجلًا.

```
import io
Data = pd.read_csv(io.BytesIO(uploaded['heart.csv']))
print(Data.head(20))
```

```
    age  sex  cp  trestbps  chol  fbs  ...  exang  oldpeak  slope  ca  thal  target
0    63    1   3       145   233    1  ...      0      2.3      0   0     1       1
1    37    1   2       130   250    0  ...      0      3.5      0   0     2       1
2    41    0   1       130   204    0  ...      0      1.4      2   0     2       1
3    56    1   1       120   236    0  ...      0      0.8      2   0     2       1
4    57    0   0       120   354    0  ...      1      0.6      2   0     2       1
5    57    1   0       140   192    0  ...      0      0.4      1   0     1       1
6    56    0   1       140   294    0  ...      0      1.3      1   0     2       1
7    44    1   1       120   263    0  ...      0      0.0      2   0     3       1
8    52    1   2       172   199    1  ...      0      0.5      2   0     3       1
9    57    1   2       150   168    0  ...      0      1.6      2   0     2       1
10   54    1   0       140   239    0  ...      0      1.2      2   0     2       1
11   48    0   2       130   275    0  ...      0      0.2      2   0     2       1
12   49    1   1       130   266    0  ...      0      0.6      2   0     2       1
13   64    1   3       110   211    0  ...      1      1.8      1   0     2       1
14   58    0   3       150   283    1  ...      0      1.0      2   0     2       1
15   50    0   2       120   219    0  ...      0      1.6      1   0     2       1
16   58    0   2       120   340    0  ...      0      0.0      2   0     2       1
17   66    0   3       150   226    0  ...      0      2.6      0   0     2       1
18   43    1   0       150   247    0  ...      0      1.5      2   0     2       1
19   69    0   3       140   239    0  ...      0      1.8      2   2     2       1

[20 rows x 14 columns]
```

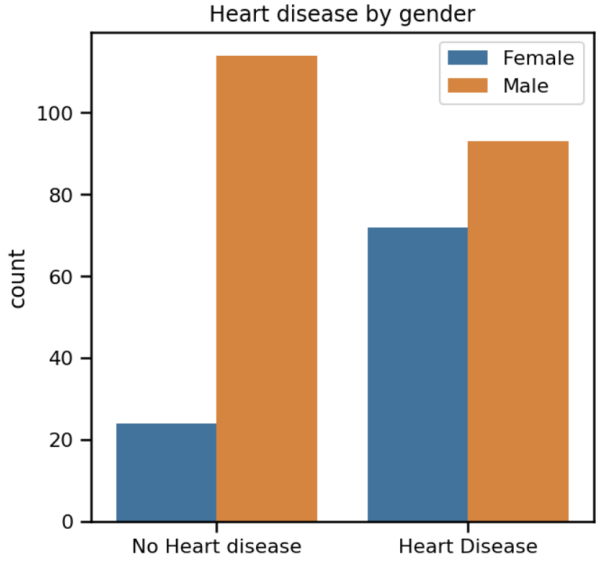
مع استيراد البيانات بنجاح، دعنا نرسم التوزيع بين أمراض القلب وغيابها، المشار إليها بواسطة الميزة الهدف.

```
f = sns.countplot(x='target', data=Data)
f.set_title("Heart disease distribution")
f.set_xticklabels(['No Heart disease', 'Heart Disease'])
plt.xlabel("");
```



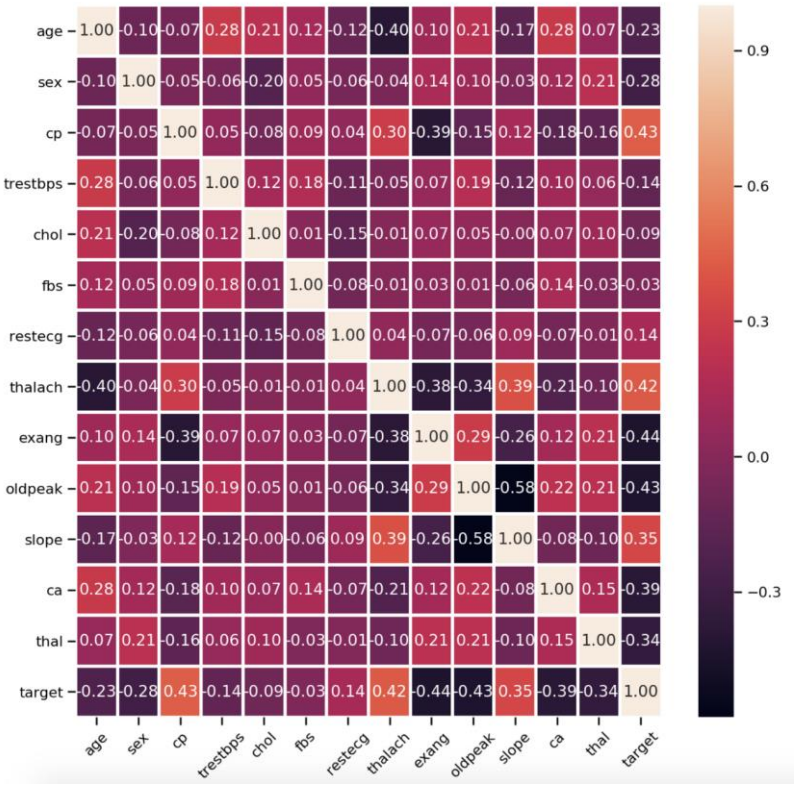
يمكننا أن نرى من الرسم البياني أعلاه أن التوزيع بين أمراض القلب الإيجابية والسلبية هو نفسه تقريبًا. إعداد جيد لمشكلة DL للتصنيف الثنائي.

```
f = sns.countplot(x='target', data=Data, hue='sex')
plt.legend(['Female', 'Male'])
f.set_title("Heart disease by gender")
f.set_xticklabels(['No Heart disease', 'Heart Disease'])
plt.xlabel("");
```



من الرسم البياني أعلاه يمكننا أن نرى أن توزيع "عدم وجود أمراض القلب" بين الذكور والإناث منحرف. لسنا متأكدينفي هذه المرحلة من تأثير ذلكفي النموذج النهائي، إن وجد، أو العلاقة بين الاثنينفي هذه المرحلة. إن فهم العلاقات بين العوامل المختلفة التي تؤثر على النتيجة النهائية لأمراض القلب هو مفتاح هذا التحليل.في هذه المرحلة، حاولنا تحديد النمط من خلال رسم مخططات فردية بين عوامل مختلفة. هناك طريقة بديلة للقيام بذلك: مصفوفة الارتباط correlation matrix أو خريطة الحرارة heat map. من المفيد جدًا إبراز المتغيرات الأكثر ارتباطًافي جدول البيانات.في هذا الرسم، يتم تلوين معاملات الارتباط وفقًا للقيمة.

```
heat_map = sns.heatmap(Data.corr(method='pearson'), annot=True,
fmt='.2f', linewidths=2)
heat_map.set_xticklabels(heat_map.get_xticklabels(), rotation=45);
plt.rcParams["figure.figsize"] = (50,50)
```



الخريطة الحرارية بين جميع السمات الـ 14

كما يمكنك ملاحظة أنه لا يوجد ارتباط قوي بين أي من السمات الأربعة عشر.

بناء مصنف Keras الثنائي

بعد استكشاف البيانات، حان الوقت لبناء مصنف Keras للتنبؤ بأمراض القلب. قمنا بتقسيم مجموعة البيانات إلى مجموعتين: مجموعة التدريب ومجموعة الاختبار. لتقسيم البيانات، استخدمنا مكتبة scikit-Learn، وبشكل أكثر تحديداً، استفدنا من دالة sklearn.model_selection.train_test_split().

```
from sklearn.model_selection import train_test_split
Input_train, Input_test, Target_train, Target_test =
train_test_split(InputScaled, Target, test_size = 0.30, random_state =
5)
print(Input_train.shape)
print(Input test.shape)
print(Target_train.shape)
print(Target_test.shape)
```

فيما يلي حجم كل مجموعة أعلاه على التوالي:

```
(212, 13)
(91, 13)
(212, 1)
(91, 1)
```

سنستخدم نموذج Keras Sequential.

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(30, input_dim=13, activation='tanh'))
model.add(Dense(20, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
```

في السطر الأول، قمنا بتعيين النموذج على أنه متسلسل sequential. ثم نضيف الطبقات الثلاث الكثيفة المتصلة بالكامل: طبقتان مخفيتان ومخرج واحد. يتم تعريف هذه باستخدام فئة كثيفة. المستوى الأول له بعد 13 وهو ما يتوافق مع 13 سمة عمود.

نستخدم tanh لضبط دالة التنشيط. الطبقة الثانية تحتوي على 20 خلية عصبية ودالة تنشيط tanh. تحتوي طبقة الإخراج على خلية عصبية واحدة (ناتج) ودالة تنشيط sigmoid المناسبة لمشاكل التصنيف الثنائي.

دعونا نجمع (compile) وندرب (fit) النموذج:

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['ac
curacy'])
model.fit(Input_train, Target_train, epochs=100, verbose=1)
```

تحتوي دالة compile على ثلاث وسيطات:

- مُحسِّن آدم **adam**: خوارزمية للتحسين القائم على التدرج من الدرجة الأولى.

- دالة الخطأ **binary_crossentropy**: الخطأ اللوغاريتمي، والتي يتم تعريفها لمشكلة تصنيف ثنائي في Keras على أنها binary_crossentropy.
- مقياس الدقة **accuracy**: لتقييم أداء النموذج الخاص بك أثناء التدريب والاختبار.

أخيرًا، قمنا بتعيين الحقب = 100 ودع النموذج يتدرب.

لن يستغرق هذا أكثر من بضع ثوانٍ إذا كنت تعمل على إعداد Google colab. يمكننا طباعة ملخص النموذج وتقييم النموذج مقابل بيانات الاختبار التي احتفظنا بها جانبًا من قبل.

```
model.summary()

score = model.evaluate(Input_test, Target_test, verbose=0)

print('Model Accuracy = ',score[1])
```

هذا هو الناتج الذي حصلت عليه:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 30)                420
_________________________________________________________________
dense_5 (Dense)              (None, 20)                620
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 21
=================================================================
Total params: 1,061 Trainable params: 1,061 Non-trainable params: 0
_________________________________________________________________
Model Accuracy =  0.9010988965139284
```

النموذج عند تقييمه على بيانات الاختبار، يكون دقيقًا بنسبة 90.10 بالمائة.

الملخص

لقد قمنا بتدريب نموذج Keras لتصنيف أمراض القلب بناءً على مجموعة البيانات مفتوحة المصدر. على الرغم من أننا حققنا هذه النتيجة على مجموعة بيانات أصغر، ستتمكن من تطبيق نفس مفاهيم استكشاف البيانات وهندسة الميزات وبناء النماذج على مجموعات بيانات أكبر. لقد جعلت الكود متاحًا هنا في github repo، فلا تترددفي تنزيله وتجربته.

20) تحديد الجنس والعمر باستخدام التعلم العميق Gender and Age Detection with Deep Learning

لطالما كان العمر والجنس سمة مهمة لهويتنا. إنه أيضًا عامل مهمفي حياتنا الاجتماعية. يمكن تطبيق تنبؤات العمر والجنس التي يتم إجراؤها باستخدام الذكاء الاصطناعي على العديد من المجالات مثل تطوير الواجهة الذكية بين الإنسان والآلة، والأمن، ومستحضرات التجميل، والتجارة الإلكترونية.

استيراد مجموعة البيانات

أثناء العمل على cAInvas ، فإن إحدى ميزاته الرئيسية هي UseCases Gallary. عند العمل على أي من حالات الاستخدام الخاصة به، لن تضطر إلى البحث عن البيانات يدويًا، البيانات موجودةفي الجدول وهي موجودة بتنسيق csv. سنقوم بتحميل مجموعة البيانات من خلال pandas كإطار بياناتفي مساحة العمل لدينا.

```
!wget -N "https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-admin/age_gender.zip"
!unzip -qo age_gender.zip
!rm age_gender.zip
```

استيراد المكتبات اللازمة

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.layers import Conv2D,InputLayer, Dropout, BatchNormalization, Flatten, Dense, MaxPooling2D
from tensorflow.keras import utils
from tensorflow.keras.models import Sequential
from keras.callbacks import ModelCheckpoint
```

تحليل البيانات

تحتوي مجموعة البيانات على العمر، والعرق، والجنس، واسم الصورة، وقيم البكسل كأعمدة، ويمكننا الحصول على وصف لمجموعة البيانات عن طريق تشغيل دالة وصف pandas، ثم يتعين علينا الحصول على قيم البكسل من عمود البكسل وتخزينهافي numpy المصفوفة، وسنقوم بإنشاء فئات عمرية بدلاً من وجود جميع الأعمار المنفصلة، ويمكننا الحصول على نظرة ثاقبة حول الأعمار المختلفة لمجموعة البيانات من خلال رسم مخطط بياني موضح أدناه: المكتبات الضرورية.

```
Dataset = pd.read_csv('age_gender.csv')
```

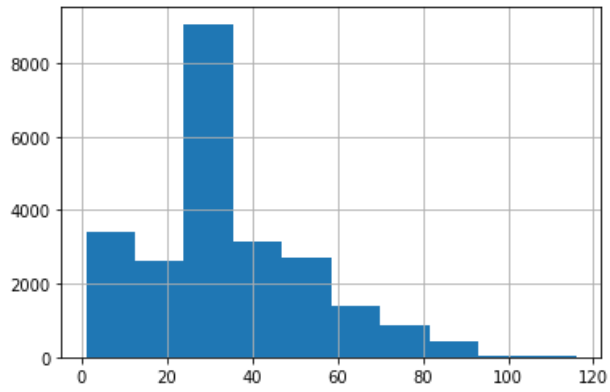
```
Dataset.head(5)
```

	age	ethnicity	gender	img_name	pixels
0	1	2	0	20161219203650636.jpg.chip.jpg	129 128 128 126 127 130 133 135 139 142 145 14...
1	1	2	0	20161219222752047.jpg.chip.jpg	164 74 111 168 169 171 175 182 184 188 193 199...
2	1	2	0	20161219222832191.jpg.chip.jpg	67 70 71 70 69 67 70 79 90 103 116 132 145 155...
3	1	2	0	20161220144911423.jpg.chip.jpg	193 197 198 200 199 200 202 203 204 205 208 21...
4	1	2	0	20161220144914327.jpg.chip.jpg	202 205 209 210 209 209 210 211 212 214 218 21...

```
Dataset.describe()
```

	age	ethnicity	gender
count	23705.000000	23705.000000	23705.000000
mean	33.300907	1.269226	0.477283
std	19.885708	1.345638	0.499494
min	1.000000	0.000000	0.000000
25%	23.000000	0.000000	0.000000
50%	29.000000	1.000000	0.000000
75%	45.000000	2.000000	1.000000
max	116.000000	4.000000	1.000000

```
# Transforming pixels which is in string format to numpy array
Dataset['pixels'] = Dataset['pixels'].map(lambda x: np.array(x.split(' '),
dtype=np.float32).reshape(48, 48))
# Plotting the data according to age
Dataset['age'].hist()
```




```
# Putting the age into a category
Dataset["age_cat"] = pd.cut(Dataset["age"],
                            bins=[0., 20., 40.0, 60., 80., np.inf],
                            labels=[1, 2, 3, 4, 5])
```

```
# Counting the category of data
Dataset["age_cat"].value_counts()
```

```
2    12122
1     4877
3     4311
4     1855
5      540
Name: age_cat, dtype: int64
```

إنشاء مجموعتي بيانات لـ AgeModel و GenderModel

بمجرد استخراج قيم البكسل وتخزينهافي مصفوفة عددية، سنستخرج متغيرين منفصلين للتسمية: أحدهما للعمر والآخر للجنس، لذلك باختصار سننشئ نموذجين للتنبؤ بالعمر والجنس.

```
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(Dataset, Dataset["age_cat"]):
    strat_train_set = Dataset.loc[train_index]
    strat_test_set = Dataset.loc[test_index]
```

```
def age_cat_proportions(data):
    return data["age_cat"].value_counts() / len(data)

train_set, test_set = train_test_split(Dataset, test_size=0.2,
random_state=42)

compare_props = pd.DataFrame({
    "Overall": age_cat_proportions(Dataset),
    "Stratified": age_cat_proportions(strat_test_set),
    "Random": age_cat_proportions(test_set),
}).sort_index()
compare_props["Rand. %error"] = 100 * compare_props["Random"] /
compare_props["Overall"] - 100
compare_props["Strat. %error"] = 100 * compare_props["Stratified"] /
compare_props["Overall"] - 100
compare_props
```

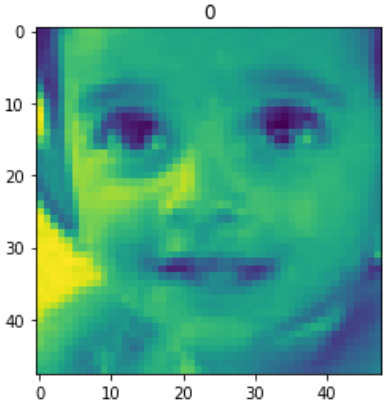
	Overall	Stratified	Random	Rand. %error	Strat. %error
1	0.205737	0.205864	0.201223	-2.193972	0.061513
2	0.511369	0.511285	0.517401	1.179673	-0.016499
3	0.181860	0.181818	0.178865	-1.646950	-0.023196
4	0.078254	0.078254	0.079730	1.886792	0.000000
5	0.022780	0.022780	0.022780	0.000000	0.000000

```
for set_ in (strat_train_set, strat_test_set):
    set_.drop("age_cat", axis=1, inplace=True)
```

إنشاء مجموعة TrainSet-TestSet

```
# split the data into train ad test
np.random.seed(42)
y_age = np.array(full_dataset['age'])
y_gender = np.array(full_dataset['gender'])
print('X',X.shape)
print('y_age',y_age.shape)
print('y_gender',y_gender.shape)

X_train, X_test, y_age_train, y_age_test, y_gender_train, y_gender_test =
train_test_split(X,y_age, y_gender, test_size=0.2, random_state=42)
X (23705, 48, 48, 1)
y_age (23705,)
y_gender (23705,)
```

```
def plot(X,y):
        plt.title(y)
        plt.imshow(X.reshape(48,48))
        plt.show()
plot(full_dataset['pixels'][50],full_dataset['gender'][10])
```



بُنية النموذج AgeModel

بعد إنشاء مجموعة البيانات، فإن الخطوة التالية هي تمرير بيانات التدريب الخاصة بنا إلى نموذج التعلم العميق الخاص بنا لتعلم كيفية تصنيف عمر وجنس بيانات الصورة. بُنية النموذج المستخدمة لنموذج AgeModel:

```
import tensorflow.keras.layers as L

tf.keras.backend.clear_session()

AgeModel = tf.keras.Sequential([
    L.InputLayer(input_shape=(48,48,1)),
    L.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    L.BatchNormalization(),
    L.MaxPooling2D((2, 2)),
    L.Conv2D(64, (3, 3), activation='relu'),
```

```
    L.MaxPooling2D((2, 2)),
    L.Flatten(),
    L.Dense(64, activation='relu'),
    L.Dropout(rate=0.5),
    L.Dense(1)
])

AgeModel.compile(optimizer='adam',
                 loss='mean_squared_error')
AgeModel.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
batch_normalization (BatchNo (None, 46, 46, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 23, 23, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 21, 21, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 6400)              0
_________________________________________________________________
dense (Dense)                (None, 64)                409664
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 428,673
Trainable params: 428,609
Non-trainable params: 64
_________________________________________________________________
```

دالة الخطأ المستخدمة هي "متوسط الخطأ التربيعي" ومحسِّن "Adam" للتنبؤ بالعمر و"binary_crossentropy" ومحسِّن "Adam" للتنبؤ بالجنس.

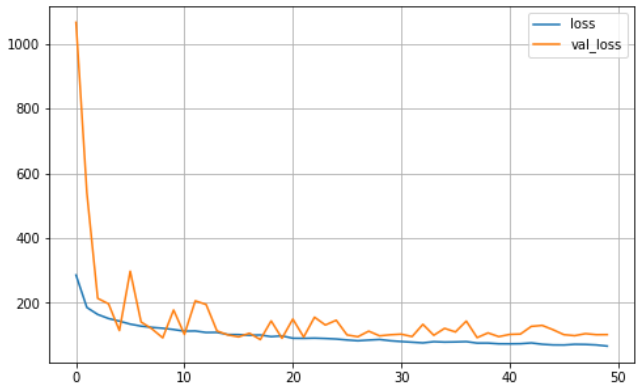
تدريب النموذج AgeModel

```
checkpointer = ModelCheckpoint('ageModel.h5', monitor='val_loss', mode='min',
verbose=2, save_best_only=True)
history = AgeModel.fit(X_train, y_age_train, epochs=50, validation_split=0.2,
batch_size=64,callbacks=[checkpointer])
```

```
232/238 [============================>.] - ETA: 0s - loss: 71.3997
Epoch 00048: val_loss did not improve from 86.29923
238/238 [==============================] - 1s 5ms/step - loss: 71.3280 - val_loss: 104.6958
Epoch 49/50
231/238 [============================>.] - ETA: 0s - loss: 69.8919
Epoch 00049: val_loss did not improve from 86.29923
238/238 [==============================] - 1s 5ms/step - loss: 69.7740 - val_loss: 101.4516
Epoch 50/50
232/238 [============================>.] - ETA: 0s - loss: 66.0722
Epoch 00050: val_loss did not improve from 86.29923
238/238 [==============================] - 1s 5ms/step - loss: 66.1853 - val_loss: 101.6719
```

مخطط التدريب للنموذج AgeModel

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.show()
```



تقييم النموذج AgeModel

```
AgeModel.evaluate(X_test,y_age_test)
```

```
149/149 [==============================] - 0s 2ms/step - loss: 99.3864
```

```
y_age_test[:10]
```

```
array([90, 37, 35, 24, 40, 24, 21, 40, 54, 57])
```

```
y_age_pred = AgeModel.predict(X_test[:10])
np.round(y_age_pred)
```

```
array([[105.],
       [ 26.],
       [ 37.],
       [ 29.],
       [ 29.],
       [ 36.],
       [ 24.],
       [ 27.],
       [ 54.],
       [ 46.]], dtype=float32)
```

بُنية النموذج GenderModle

```
##Gender Model
tf.keras.backend.clear_session()
GenderModel = tf.keras.Sequential([
    L.InputLayer(input_shape=(48,48,1)),
    L.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    L.BatchNormalization(),
    L.MaxPooling2D((2, 2)),
    L.Conv2D(64, (3, 3), activation='relu'),
    L.MaxPooling2D((2, 2)),
    L.Flatten(),
    L.Dense(64, activation='relu'),
    L.Dropout(rate=0.5),
    L.Dense(1, activation='sigmoid')
])

GenderModel.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])
GenderModel.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
batch_normalization (BatchNo (None, 46, 46, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 23, 23, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 21, 21, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 6400)              0
_________________________________________________________________
dense (Dense)                (None, 64)                409664
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 428,673
Trainable params: 428,609
Non-trainable params: 64
_________________________________________________________________
```

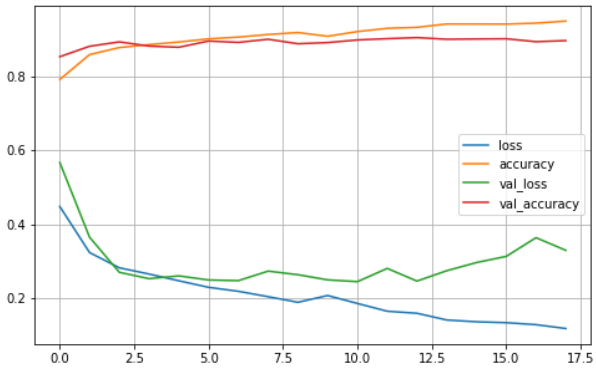
تدريب النموذج GenderModel

```
Gender_history = GenderModel.fit(
    X_train, y_gender_train, epochs=18, validation_split=0.2, batch_size=64)
```

```
Epoch 14/18
238/238 [==============================] - 1s 5ms/step - loss: 0.1408 - accuracy: 0.9411 - val_loss: 0.2740 - val_accuracy: 0.8998
Epoch 15/18
238/238 [==============================] - 1s 5ms/step - loss: 0.1360 - accuracy: 0.9410 - val_loss: 0.2962 - val_accuracy: 0.9006
Epoch 16/18
238/238 [==============================] - 1s 6ms/step - loss: 0.1335 - accuracy: 0.9409 - val_loss: 0.3129 - val_accuracy: 0.9011
Epoch 17/18
238/238 [==============================] - 1s 6ms/step - loss: 0.1282 - accuracy: 0.9434 - val_loss: 0.3634 - val_accuracy: 0.8932
Epoch 18/18
238/238 [==============================] - 1s 6ms/step - loss: 0.1178 - accuracy: 0.9490 - val_loss: 0.3291 - val_accuracy: 0.8961
```

مخطط التدريب للنموذج GenderModel

```
GenderModel.save("GenderModel.h5")
pd.DataFrame(Gender_history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.show()
```



تقييم النموذج GenderModel

```
loss, acc = GenderModel.evaluate(X_test,y_gender_test,verbose=0)
print('Test loss: {}'.format(loss))
print('Test Accuracy: {}'.format(acc))
```

```
Test loss: 0.3629770576953888
Test Accuracy: 0.8873655200004578
```

```
y_gender_test[:10]
```

```
array([1, 1, 0, 1, 1, 1, 1, 1, 0, 0])
```

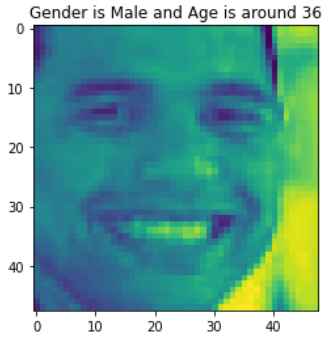
```
y_gender_pred = GenderModel.predict(X_test)
np.transpose(np.round(y_gender_pred))
```

```
array([[1., 1., 0., ..., 0., 0., 1.]], dtype=float32)
```

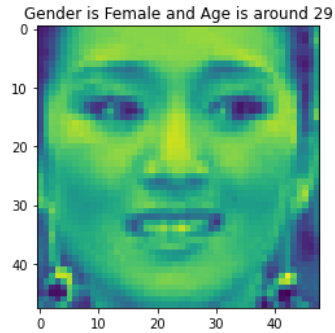
الوصول إلى أداء النموذج

```
def plot(X,y_age,y_gender):
    if y_gender<=0.5:
        plt.title('Gender is Male and Age is around ' +str(y_age))

    else:
        plt.title('Gender is Female and Age is around ' +str(y_age))
    plt.imshow(X.reshape(48,48))
    plt.show()
```

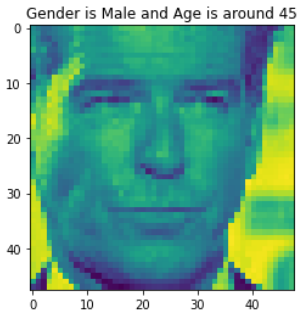
```
n=2
plot(X_test[n],int(y_age_pred[n]),y_gender_pred[n])
```




```
n=4
plot(X_test[n],int(y_age_pred[n]),y_gender_pred[n])
```




```
n=9
plot(X_test[n],int(y_age_pred[n]),y_gender_pred[n])
```



21) تصنيف الصوت الحضري باستخدام الشبكات العصبية التلافيفية Urban Sound Classification using Convolutional Neural Networks

مقدمة

على مدى السنوات الخمس الماضية، انتقلت التطوراتفي الذكاء الاصطناعي إلى وسط الصوت، سواء كان ذلكفي إنشاء أشكال جديدة من الموسيقى (بدرجات متفاوتة من النجاح)، أو تحديد أدوات معينة من مقطع فيديو. تم بالفعل إطلاق بعض هذه المشاريع، مثل Watson Beat من شركة IBM، للاستخدام التجاري ـ في الواقع، يدعي منشئها أن الشبكة التي تقف وراء Watson Beat قد تعلمت الاستجابة العاطفية المرتبطة بعناصر موسيقية محددة، وهو أمر شخصي للغاية وكان سابقًا المجال الحصري من الملحنين البشر.

على الرغم من أن الشبكات العصبية للذاكرة طويلة المدى (LSTMs) عادة ما تكون مرتبطة بمشاريع التعلم العميق القائمة على الصوت، إلا أنه يمكن أيضًا معالجة عناصر تحديد الصوت كمهمة تقليدية لتصنيف الصور متعددة الفئات باستخدام الشبكات العصبية التلافيفية CNN. في هذا البرنامج التعليمي، سوف نوضح كيف يمكن لشبكة عصبية بسيطة تم إنشاؤهافي Keras، جنبًا إلى جنب مع بعض مكتبات تحليل الصوت المفيدة، التمييز بين 10 أصوات مختلفة بدقة عالية، باستخدام مجموعة بيانات UrbanSound المتاحة على Kaggle.

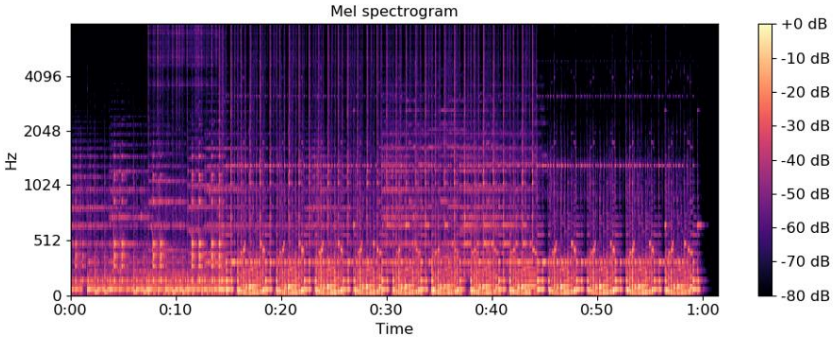
قد يبدو من غير البديهي استخدام الشبكات العصبية التلافيفية CNN لإجراء تصنيف الصوت، ولكن النظريةفي الواقع بسيطة للغاية ـ يمكن تمثيل كل الصوت بصورة مخطط طيفي spectrogram، تصور التغيراتفي التردد (هرتز) والشدة / جهارة الصوت (ديسيبل) بمرور الوقت. إذا كانت مجموعة الأصوات التي تم النظر فيها مميزة بشكل فردي، فستظهر مخططات الطيف مختلفة بما يكفي للسماح بالتمييز باستخدام CNN. بطبيعة الحال، لن يكون نهجًا كهذا بالضرورة قادرًا على التمييز بين الكلمات الفرديةفي الجملة (وهو ما تشتهر به LSTM)، ولكنه سيكون مناسبًا لفصل الكلب الذي ينبح عن المحادثات العادية، على سبيل المثال.

سنفترض أن القارئ على دراية بعناصر التعلم العميق والمبادئ الكامنة وراء الشبكات العصبية التلافيفية CNN.

التنفيذ

تتكون مجموعة بيانات UrbanSound من 8732 مقتطفات صوتية قصيرة (أقل من 4 ثوانٍ) لأصوات حضرية من 10 فئات مختلفة. تتكون كل من مجموعات بيانات التدريب والاختبار لدينا من ملفات wav. وجدول بيانات csv. المصاحب الذي يوضح بالتفصيل معرفاتهم وفي

حالة بيانات التدريب، فإن الأوصاف الصحيحة الخاصة بهم، والتي تكون بمثابة تسمياتهم. من أجل تحويل بياناتنا إلى تمثيلات طيفية، سنستخدم LibROSA، حزمة بايثون مفتوحة المصدر لتحليل الموسيقى والصوت.



مخطط طيفي LibROSA لعينة صوت مدتها دقيقة واحدة

تم تنفيذنا على Kaggle، ولكن يجب أن يكون أي مثيل بايثون ممكّن بواسطة GPU قادرًا على تحقيق نفس النتائج. تم تكييف الكود الخاص بنا من محاولة تم إجراؤها باستخدام نقل التعلم باستخدام FAST.AI.في هذا البرنامج التعليمي، سننشئ شبكة من البداية باستخدام Keras، لفهم أفضل لكيفية تأثير بنية الشبكة على تصنيف الصور المتشابهة للغاية.

بادئ ذي بدء، نقوم باستيراد حزم Pandas وNumpy، جنبًا إلى جنب مع ملف تعريف الذاكرة لمراقبة استخدام الذاكرة نظرًا للكم الهائل من تحويل البيانات الذي نحتاج إلى تنفيذه على خوادم Kaggle.

```
%matplotlib inline
from memory_profiler import memory_usage
import os
import pandas as pd
from glob import glob
import numpy as np
```

نقوم بعد ذلك بتثبيت libavtools، وهو إطار عمل مفتوح المصدر لمعالجة الفيديو والصوت من أجل الوصول إلى LibROSA. إذا كنت تفعل ذلك محليًا، فيجب أن تفعل الشيء نفسه عبر وحدة التحكم.

```
%%capture
!apt-get install libav-tools -y
```

بعد ذلك، دعنا نستورد مكتبات Keras الضرورية لبناء شبكتنا بالإضافة إلى الحزم الإضافية الضرورية الأخرى. وتجدر الإشارة بشكل خاص إلى حزمة أداة تجميع البيانات المهملة garbage collector package، والتي تتيح لنا تنظيف ذاكرة الوصول العشوائي أثناء عملية

تحويل البيانات. أخيرًا، نبني أيضًا أدلة عمل في مثال Kaggle الخاص بنا من أجل تخزين صورنا المحولة.

```
from keras import layers
from keras import models
from keras.layers.advanced_activations import LeakyReLU
from keras.optimizers import Adam
import keras.backend as K
import librosa
import librosa.display
import pylab
import matplotlib.pyplot as plt
from matplotlib import figure
import gc
from path import Path!mkdir /kaggle/working/train
!mkdir /kaggle/working/test
```

بعد ذلك، نبدأ عملية تحويل البيانات الخاصة بنا عن طريق تحديد الدوال التي ستحول ملفات wav. الخاصة بنا إلى صور، بتنسيق jpg. باختصار، نقوم باستخراج السلاسل الزمنية الصوتية ومعدل أخذ العينات لكل ملف wav. باستخدام LibROSA ، قبل إنشاء مخطط طيفي للبيانات وتخطيطه وحفظه كصورة مقابلة.

```
def create_spectrogram(filename,name):
plt.interactive(False)
clip, sample_rate = librosa.load(filename, sr=None)
fig = plt.figure(figsize=[0.72,0.72])
ax = fig.add_subplot(111)
ax.axes.get_xaxis().set_visible(False)
ax.axes.get_yaxis().set_visible(False)
ax.set_frame_on(False)
S = librosa.feature.melspectrogram(y=clip, sr=sample_rate)
librosa.display.specshow(librosa.power_to_db(S, ref=np.max))
filename = '/kaggle/working/train/' + name + '.jpg'
plt.savefig(filename, dpi=400, bbox_inches='tight',pad_inches=0)
plt.close()
fig.clf()
plt.close(fig)
plt.close('all')
del filename,name,clip,sample_rate,fig,ax,S
```

وبالمثل، بالنسبة إلى دليل بيانات الاختبار لدينا:

```
def create_spectrogram_test(filename,name):
plt.interactive(False)
clip, sample_rate = librosa.load(filename, sr=None)
fig = plt.figure(figsize=[0.72,0.72])
ax = fig.add_subplot(111)
ax.axes.get_xaxis().set_visible(False)
ax.axes.get_yaxis().set_visible(False)
ax.set_frame_on(False)
S = librosa.feature.melspectrogram(y=clip, sr=sample_rate)
librosa.display.specshow(librosa.power_to_db(S, ref=np.max))
filename = Path('/kaggle/working/test/' + name + '.jpg')
fig.savefig(filename, dpi=400, bbox_inches='tight',pad_inches=0)
plt.close()
fig.clf()
```

```
plt.close(fig)
plt.close('all')
del filename,name,clip,sample_rate,fig,ax,S
```

مع هذا التحديد، فلنبدأفي تحويل بيانات التدريب الخاصة بنا. سنفعل ذلك على دفعات من 2000 صورةفي المرة الواحدة، باستخدام حزمة جامع البيانات المهملة لتحسين استخدام الذاكرة بين الدُفعات.

```
Data_dir=np.array(glob("../input/train/Train/*"))%load_ext
memory_profiler%%memit
i=0
for file in Data dir[i:i+2000]:
    #Define the filename as is, "name" refers to the JPG, and is split off
into the number itself.
    filename,name = file,file.split('/')[-1].split('.')[0]
    create_spectrogram(filename,name)gc.collect()%%memit
i=2000
for file in Data_dir[i:i+2000]:
    filename,name = file,file.split('/')[-1].split('.')[0]
    create_spectrogram(filename,name)gc.collect()%%memit
i=4000
for file in Data_dir[i:]:
    filename,name = file,file.split('/')[-1].split('.')[0]
    create_spectrogram(filename,name)gc.collect()
```

القيام بالشيء نفسه لمجموعة بيانات الاختبار الخاصة بنا:

```
Test_dir=np.array(glob("../input/test/Test/*"))%%memit
i=0
for file in Test_dir[i:i+1500]:
    filename,name = file,file.split('/')[-1].split('.')[0]
    create_spectrogram_test(filename,name)gc.collect()%%memit
i=1500
for file in Test_dir[i:]:
    filename,name = file,file.split('/')[-1].split('.')[0]
    create_spectrogram_test(filename,name)gc.collect()
```

على عكس البرنامج التعليمي السابق، لا توجد تسميات البيانات الخاصة بنا مباشرةًفي أسماء ملفات الصورفي هذا المشروع، ولكن بدلاً من ذلكفي جدول بيانات csv. المصاحب. يمكنك تصفح جدول البيانات وقضاء بعض الوقتفي إعادة تسمية كل ملف وفقًا لذلك، ولكن Keras توفر مولد بيانات من خطوة واحدة يقرأ جداول البيانات بسهولة للحصول على التسميات الصحيحة أثناء إعداد البياناتفي دفعات محددة للتدريب والتحقق من الصحة.

دعونا نستخدمه الآن. لاحظ أننا نلحق امتداد jpg بعمود المعرفّفي جدول بيانات التدريب الخاص بنا لضمان اقتران الملف الصحيح.

```
from keras_preprocessing.image import ImageDataGenerator

def append_ext(fn):
return fn+".jpg"

traindf=pd.read_csv('../input/train.csv',dtype=str)
testdf=pd.read_csv('../input/test.csv',dtype=str)
traindf["ID"]=traindf["ID"].apply(append_ext)
```

```
testdf["ID"]=testdf["ID"].apply(append_ext)

datagen=ImageDataGenerator(rescale=1./255.,validation_split=0.25)


train_generator=datagen.flow_from_dataframe(
dataframe=traindf,
directory="/kaggle/working/train/",
x_col="ID",
y_col="Class",
subset="training",
batch_size=32,
seed=42,
shuffle=True,
class_mode="categorical",
target_size=(64,64))

valid_generator=datagen.flow_from_dataframe(
dataframe=traindf,
directory="/kaggle/working/train/",
x_col="ID",
y_col="Class",
subset="validation",
batch_size=32,
seed=42,
shuffle=True,
class_mode="categorical",
target_size=(64,64))
```

الآن بالنسبة للجزء الممتع، دعنا نبني نموذج الشبكة العصبية المتسلسلة الخاص بنا باستخدام مُحسِّن RMSProp (جرب ADAM أو المحسِّنين الآخرين ومعرفة ما إذا كان بإمكانك الحصول على مزيد من الدقة!). تتكون بُنية شبكتنا من 6 طبقات تلافيفية مع كثافة فلتر متزايدة من أجل استخراج أفضل ميزات كل صورة مع كل طبقة متتالية. تعمل طبقات التجميع والتسرب على زيادة الكفاءة الحسابية ومنع فرط التعلم، على التوالي.

في محاولتنا، لوحظ أن إضافة طبقات ذات رقم فلتر مرتفعفي نهاية الشبكة عزز الدقة بنسبة تصل إلى 3٪. بشكل بديهي، يمكن فهم ذلك على أنه فلاتر الطبقة السابقة تستهدف الميزات المشتركة بشكل أساسي بواسطة جميع مخططات الطيف (الخطوط والمنحنيات).في المقابل، بمجرد أن نصل إلى الطبقات اللاحقة مع خرائط الإخراج الملتفة الخاصة بهم، فإن مخططات الطيف تكون متشابهة بدرجة كافية بحيث يساعد عدد أكبر من الفلاتر المعقدةفي التمييز بينها بشكل أفضل. لا ترددفي تعديل بُنية الشبكة لمراقبة ذلك بنفسك.

```
from keras.layers import Dense, Activation, Flatten, Dropout,
BatchNormalization
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPooling2D
from keras import regularizers, optimizers
import pandas as pd
import numpy as np
```

```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
input_shape=(64,64,3)))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(128, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
model.compile(optimizers.rmsprop(lr=0.0005, decay=1e-
6),loss="categorical_crossentropy",metrics=["accuracy"])
model.summary()
```

الآن بعد أن أصبح كل شيء جاهزًا، فلنقم بتقييم نموذجنا!

```
#Fitting keras model, no test gen for now
STEP SIZE TRAIN=train generator.n//train generator.batch size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
#STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation data=valid generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=150
)
model.evaluate_generator(generator=valid_generator, steps=STEP_SIZE_VALID
)
```

```
[0.3850516012475052, 0.9457013571963591]
```

قيم الخطأ والدقة من نموذجنا، مدربة على أكثر من 150 حقبة بمعدل تعلم 0.0005

يبدو أن نموذجنا يناسب البيانات جيدًا، بدقة تقترب من 95٪. يوضح هذا المشروع القوة الكامنة في البيانات الجيدة والنظيفة والمعنونة ـ تشغيل الشبكة على جزء صغير فقط من بيانات التدريب، لنقل 2000 صورة أولية، سوف يمنحك دقة تبلغ 83٪ فقط. هذا مرتبط بتشابه مخططاتنا الطيفية ـ لا تقلق، فلن تحتاج إلى 6000 صورة لتمييز القطط والكلاب عن بعضها!

دعونا نلاحظ أدائها من خلال توقعهافي مجموعة الاختبار. نحتاج إلى إنشاء مولد اختبار لتغذية بيانات الاختبار لشبكتنا.

```
test_datagen=ImageDataGenerator(rescale=1./255.)
test_generator=test_datagen.flow_from_dataframe(
    dataframe=testdf,
    directory="/kaggle/working/test/",
    x_col="ID",
    y_col=None,
    batch_size=32,
    seed=42,
    shuffle=False,
    class_mode=None,

target_size=(64,64))STEP_SIZE_TEST=test_generator.n//test_generator.batch_
size
```

دعنا الآن نتوقع أول 7 مقاطع صوتيةفي بيانات الاختبار الخاصة بنا:

```
test_generator.reset()
pred=model.predict_generator(test_generator,
steps=STEP_SIZE_TEST,
verbose=1)predicted_class_indices=np.argmax(pred,axis=1)

#Fetch labels from train gen for testing
labels = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels.items())
predictions = [labels[k] for k in predicted_class_indices]
print(predictions[0:6])
```

يجب أن تبدو مخرجاتك كما يلي:

```
['jackhammer', 'children_playing', 'drilling', 'dog_bark', 'street_music',
'jackhammer', 'air_conditioner']
```

الملخص

- يمكن للشبكات العصبية التلافيفية CNN تصنيف مقاطع الصوت إلى درجة عالية من الدقة من خلال استخدام تمثيلات الصور.
- تتفوق الشبكات المعقدة التي تحتوي على المزيد من الفلاترفي الطبقات اللاحقة على الشبكات الأبسط عند العمل مع صور مماثلة.
- كمية البيانات هي المفتاح لتحسين دقة التصنيف، لا سيما مع الصور المماثلة.

22) الكشف عن COVID-19 من صور الأشعة السينية للصدر باستخدام نقل التعلم Detecting COVID-19 From Chest X-Ray Images using Transfer Learning

تطبيق ويب قائم على Django تم إنشاؤه لغرض اكتشاف وجود COVID-19 من صور Chest X-Ray مع نماذج متعددة للتعلم الآلي تم تدريبها على البنى المبنية مسبقًا. تم استخدام ثلاثة نماذج مختلفة للتعلم الآلي لبناء هذا المشروع وهي Xception وResNet50 وVGG16. تم تدريب نموذج التعلم العميق على مجموعة بيانات متاحة للجمهور، مجموعة بيانات SARS-COV-2-Ct-Scan. الغرض من هذا المشروع هو تطبيق بُنى الشبكة العصبية التلافيفية (CNN)في حل مشاكل الوباءفي مرحلة أولية.

الأدوات والتقنيات المستخدمة

بعض المكتبات والتقنيات الهامة المستخدمة مذكورة أدناه:

- لغة البرمجة: بايثون
- إطار عمل الويب: Django
- إطار تعلم الآلة والتعلم العميق: Tensorflow
- مطور الواجهة الأمامية: HTML، CSS (BootStrap)
- المكتبات الأساسية: keras و sklearn و venv و seaborn و matplotlib

يمكن العثور على قائمة مفصلة بجميع المكتبات هنا.

التنفيذ خطوة بخطوة

جزء التعلم العميق

الخطوة 1: تحويل مجموعة البيانات Dataset إلى إطار البيانات Dataframe

- قم بتنزيل مجموعة البيانات من هنا.
- تحويل البيانات إلى pandas dataframe مع الأعمدة المقابلة.
  - File [Image File]
  - DiseaseID [Serial Number]
  - DiseaseType [COVID, non-COVID]

```
train_dir = 'path/to/dataset'
train_data = []

for defects_id, sp in enumerate(disease_types):
    for file in os.listdir(os.path.join(train_dir, sp)):
        train_data.append(['{}/{}'.format(sp, file), defects_id, sp])
```

```
train = pd.DataFrame(train data, columns=['File', 'DiseaseID',
'Disease Type'])
```

الخطوة 2: القراءة والمعالجة المسبقة لإطار البيانات

- قراءة الصور.
- تحويل الصور الى الحجم القياسي (64 × 64)
- إنشاء مصفوفات عددية للإدخال / الإخراج X_Train و Y_Train
- تسوية قيم RGB بالقسمة على 255.

```
IMAGE_SIZE = 64

def read_image(filepath):
    return cv2.imread(os.path.join(data_dir, filepath))

def resize_image(image, image_size):
    return cv2.resize(image.copy(), image_size,
                      interpolation=cv2.INTER_AREA)

X_train = np.zeros((train.shape[0], IMAGE_SIZE, IMAGE_SIZE, 3))

for i, file in tqdm(enumerate(train['File'].values)):
    image = read_image(file)
    if image is not None:
        X_train[i] = resize_image(image, (IMAGE_SIZE, IMAGE_SIZE))

X Train = X train / 255.

Y_train = train['DiseaseID'].values
Y_train = to_categorical(Y_train, num_classes=2)
```

الخطوة 3: تقسيم مجموعة البيانات إلى تدريب/التحقق من الصحة

- تقسيم إلى مجموعات بيانات التحقق والتدريب.
- تحديد تقسيم النسبة المئوية والحالة العشوائية وفقًا لذلك.

```
X_train, X_val, Y_train, Y_val = train_test_split(
  X_Train, Y_train, test_size=0.2, random_state = 42)
```

الخطوة 4: تحديد معمارية النموذج

- سنقوم باستيراد ثلاث معماريات مختلفة مذكورة أدناه:
  - VGG16
  - ResNet50
  - Xception
- هيكل معمارية النموذج:
  - Conv2Dلشكل الإدخال(3،3)
  - معمارية ResNet50 / Xception / VGG16
  - إضافة GlobalAveragePooling2D ()
  - اضافة طبقة Dropout

- إضافة طبقة DenseNet النهائية مع تنشيطrelu
- بالنسبة للإخراج المتعدد، اضافة طبقة Softmax

- استخدام مُحسِّن "adam"، يمكن ضبط المعلمات الفائقة وفقًا لذلك.
- يقترح الكود التالي كود لبناء النموذج:

```
def build_model():

    # Use Any One of the Following Lines
    resnet50 = ResNet50(weights='imagenet', include_top=False)
    xception = Xception(weights='imagenet', include_top=False)
    vgg16 = VGG16(weights='imagenet', include_top=False)

    input = Input(shape=(SIZE, SIZE, N_ch))
    x = Conv2D(3, (3, 3), padding='same')(input)

    # Use Any One of the Following Lines
    x = resnet50(x)
    x = xception(x)
    x = vgg16(x)

    x = GlobalAveragePooling2D()(x)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)
    x = Dense(256, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)

    # multi output
    output = Dense(2, activation='softmax', name='root')(x)

    # model
    model = Model(input, output)

    optimizer = Adam(lr=0.003, beta_1=0.9, beta_2=0.999,
                     epsilon=0.1, decay=0.0)

    model.compile(loss='categorical_crossentropy',
                  optimizer=optimizer, metrics=['accuracy'])

    model.summary()

    return model
```

الخطوة 5: تدريب النموذج

- استدعاء دالة build_model()
- استخدم annealer، رد اتصال callback يراقب الكمية وإذا لم يتم ملاحظة أي تحسن لعدد "patience " من الحقب، يتم تقليل معدل التعلم.
- استخدام ImageDataGenerator لتنفيذ زيادة بيانات الصورةفي الوقت الفعلي.
- تدريب النموذج على x_train، y_train.
- حفظ أوزان النموذج بتنسيق hdf5. ورسم بياني للنموذج بتنسيق json.

```
# Use Any one of the Lines Below
hdf5_save = 'ResNet50_Model.hdf5'
hdf5_save = 'Xception_Model.hdf5'
hdf5_save = 'VGG16_Model.hdf5'

model = build_model()
annealer = ReduceLROnPlateau(
    monitor='val_accuracy', factor=0.70, patience=5,
    verbose=1, min_lr=1e-4)

checkpoint = ModelCheckpoint(h5f5_save, verbose=1,
save_best_only=True)

datagen = ImageDataGenerator(rotation_range=360,
                             width_shift_range=0.2,
                             height_shift_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True,
                             vertical_flip=True)

datagen.fit(X_train)

# Use Any one of the lines Below
model_graph = 'ResNet50.json'
model_graph = 'Xception.json'
model_graph = 'VGG16.json'

model_json = model.to_json()
with open(model_graph, "w") as json_file:
    json_file.write(model_json)
```

بناء تطبيق الويب

- قم بإنشاء مشروع Django مع تطبيق مهيأ بداخله والذي سيستخدم أوزان النموذج المحفوظة للتنبؤ بصور الصدر التي تم تحميلها بالأشعة السينية.
- قم بإنشاء صفحة ثابتة أساسية باستخدام نموذج لإرسال ملف الصورة إلى الواجهة الخلفية.

HTML

```
<form method="post" id="imageForm" enctype="multipart/form-data">
   {% csrf_token %}
   <label for="ImgFile">Upload Image</label>
   <input type="file" name="ImgFile" class="form-control"/>
   <input type="submit" id="submitButton" class="btn" name="submit"
value="Solve"/>
</form>
```

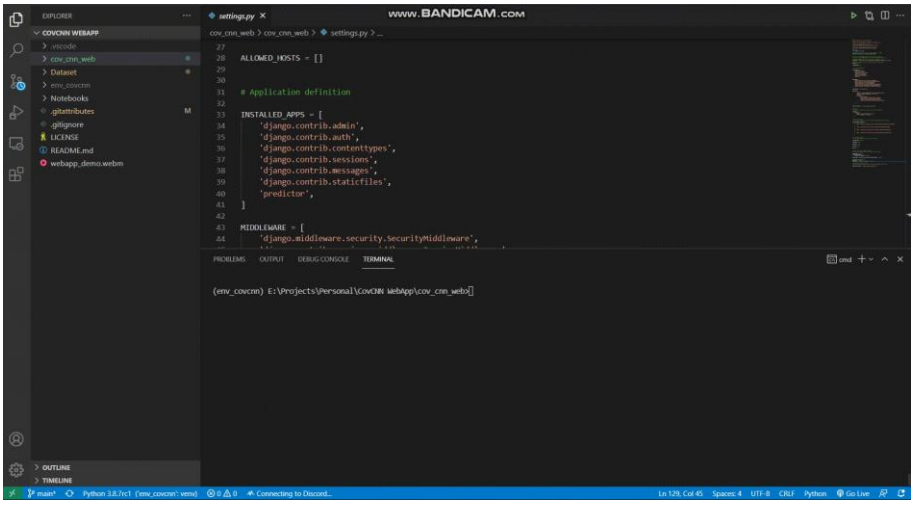
- داخل مجلد views.py، تعامل مع الصورة التي تم تحميلها. قم بتحميل ملفات النموذج وأرسل الرد مرة أخرى إلى الواجهة الأمامية.
- سيحتوي الرد على التفاصيل التالية:
  - التنبؤ النموذجي Model Prediction
  - درجات الثقة Confidence Score

- مدة التنبؤ (بالثواني) Prediction Duration
- أضف نمطًا إلى الواجهة الأمامية باستخدام (Bootstrap) CSS وفقًا لذلك.

ملاحظة: يستغرق تحميل نماذج متعددة واستخدام model.predict() الكثير من الوقت وسيكون أكثر من ذلك بكثيرفي حالة عدم وجود خدمات GPUفي مثيل Cloud. لتوسيع هذا التطبيق إلى حمل خادم أعلى، ضعفي اعتبارك استخدام خدمة TensorFlow.

DEMO

يتم عرض نسخة تجريبية من المشروع تم بناؤه واختباره على المضيف المحليفي الفيديو أدناه



23) تشخيص مرض الزهايمر باستخدام CNN Alzheimer Diagnosis using CNN

مرض الزهايمر Alzheimer's disease هو حالة تنكسية تظهر فيها أعراض الخرف بمرور الوقت. يكون فقدان الذاكرة ضئيلًافي المراحل المبكرة من داء الزهايمر، لكن الأشخاص المصابين بداء الزهايمرفي المرحلة المتأخرة يفقدون قدرتهم على التحدث والاستجابة لما يحيط بهم.

هناك 7 مراحل مختلفة من مرض الزهايمر:

- المرحلة 1: السلوك الخارجي الطبيعي Normal Outward Behavior.
- المرحلة الثانية: تغييرات طفيفة جداً Very Mild Changes.
- المرحلة 3: انحدار طفيف Mild Decline.
- المرحلة 4: انحدار معتدل Moderate Decline.
- المرحلة الخامسة: انحدار حاد بشكل معتدل Moderately Severe Decline.
- المرحلة 6: الانحدار الشديد Severe Decline.
- المرحلة السابعة: هبوط شديد للغاية Very Severe Decline.

في هذه المقالة، أوضحنا كيف يمكن اكتشاف مرض الزهايمرفي سن مبكرة باستخدام الشبكات العصبية التلافيفية CNN.

استيراد المكتبات الضرورية:

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
%matplotlib inline
import tensorflow.keras.backend as K
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from pylab import imread,subplot,imshow,show
import cv2
import os
```

مجموعة البيانات:

تتوفر مجموعة البيانات على Kaggle التي قدمها Sarvesh Dubey.

في الأساس، يتم تقسيم البيانات إلى نوعين، اختبار البيانات وتدريبها. كان هناك 5121 صورةفي مجموعة التدريب و1279في مجموعة الاختبار. كانت الصور المقدمةفي المجموعة واضحة تمامًا ومنسقة جيدًا.

يوجدفي جميع التصنيفات الأربعة التي نحتاج إلى توقعها.

1. Mild Dementia,
2. Moderate Dementia
3. Non-Dementia
4. Very Mild Dementia

تحضير البيانات:

كانت مجموعة التدريب والاختبار موجودة بالفعلفي مجموعة البيانات. ومع ذلك، كانت مجموعة التحقق من الصحة مفقودة. لذلك كان علينا تقسيم مجموعة بيانات التدريب إلى نسبة 20:80 أي (4097 للتدريب و1024 للتحقق).

معالجة الصورة:

بحكم التعريف، معالجة الصور هي طريقة لإجراء بعض العمليات على صورة، من أجل الحصول على صورة محسنة أو لاستخراج بعض المعلومات المفيدة منها. إنه نوع من معالجة الإشارات يكون فيه الإدخال صورة وقد يكون الإخراج صورة أو خصائص / ميزات مرتبطة بتلك الصورة.

إعادة القياس:

نظرًا لأن حجم الصورة يحتوي على حد أقصى يبلغ 255 بكسل، أي أن النطاق يبلغ [0،255]، ولكن يصبح من الصعب على النموذج معالجة مثل هذا البكسل العالي، لذلك نحتاج إلى إعادة قياسه قبل التغذية بالنموذج.

```
train = ImageDataGenerator(rescale=1./255)
test =  ImageDataGenerator(rescale=1./255)
val =  ImageDataGenerator(rescale=1./255)
```

```
train='Alzheimer_s Dataset/train/'
```

```
train_data = tf.keras.preprocessing.image_dataset_from_directory(
    train,
    validation_split=0.2,
    image_size=(224,224),
    batch_size=32,
    subset='training',
    seed=1000 )
```

```
val='Alzheimer_s Dataset/train/'
```

```
val_data = tf.keras.preprocessing.image_dataset_from_directory(
    val,
    validation_split=0.2,
    image_size=(224,224),
```

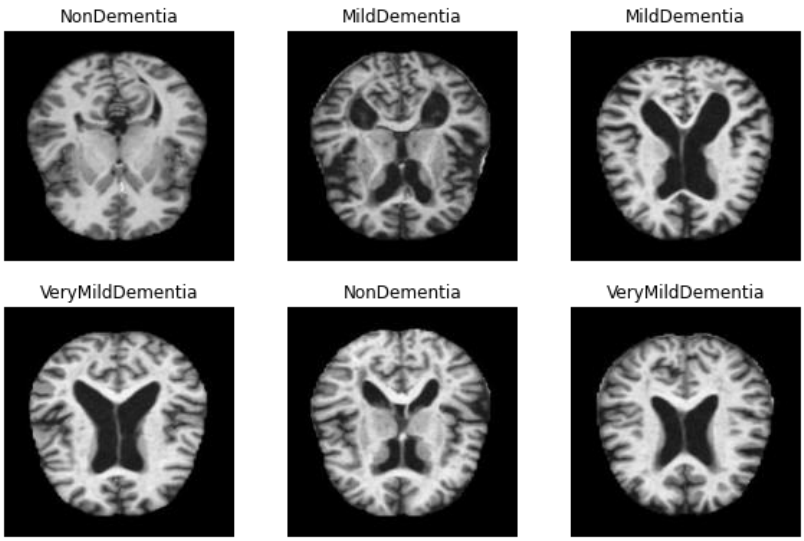
```
    batch_size=32,
    subset='validation',
    seed=1000
    )
```

```
test='Alzheimer_s Dataset/test/'
```

```
test_data=tf.keras.preprocessing.image_dataset_from_directory(
    test,
    image_size=(224,224),
    batch_size=32,
    seed=1000
    )
class_names = ['MildDementia', 'ModerateDementia', 'NonDementia',
'VeryMildDementia']
```

```
train_data.class_names = class_names
val_data.class_names = class_names
```

```
print(val_data)
plt.figure(figsize=(10, 10))
for images, labels in train_data.take(1):
    for i in range(6):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(train_data.class_names[labels[i]])
        plt.axis("off")
```



بناء النموذج

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D,
MaxPooling2D,Input
from tensorflow.keras.layers import Dense
```

```
model=Sequential()
```

```
model.add(Conv2D(16,(3,3), activation='relu', input_shape=(224,224,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32,(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32,(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(10,activation='relu'))
model.add(Dense(5,activation='relu'))
model.add(Dense(12,activation='relu'))
model.add(Dense(30,activation='relu'))
model.add(Dense(10,activation='relu'))
model.add(Dense(100,activation='relu'))
model.add(Dense(133,activation='relu'))
model.add(Dense(4,activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 109, 109, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 54, 54, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 52, 52, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 26, 26, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 21632)             0
_________________________________________________________________
dense (Dense)                (None, 10)                216330
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 55
_________________________________________________________________
dense_2 (Dense)              (None, 12)                72
_________________________________________________________________
dense_3 (Dense)              (None, 30)                390
_________________________________________________________________
dense_4 (Dense)              (None, 10)                310
_________________________________________________________________
dense_5 (Dense)              (None, 100)               1100
```

```
_________________________________________________________________
dense_6 (Dense)              (None, 133)               13433
_________________________________________________________________
dense_7 (Dense)              (None, 4)                 536
=================================================================
Total params: 246,562
Trainable params: 246,562
Non-trainable params: 0
```

تجميع وتدريب النموذج

```
model.compile(optimizer = tf.keras.optimizers.Adam(1e-4),
loss="sparse_categorical_crossentropy", metrics=["accuracy"])
```

```
history = model.fit(train_data, validation_data=val_data, epochs=10)
```

```
Epoch 1/10
129/129 [==============================] - 7s 51ms/step - loss: 1.0227 -
accuracy: 0.5116 - val_loss: 0.8785 - val_accuracy: 0.5742
Epoch 2/10
129/129 [==============================] - 6s 49ms/step - loss: 0.8658 -
accuracy: 0.5880 - val_loss: 1.0934 - val_accuracy: 0.5332
Epoch 3/10
129/129 [==============================] - 6s 49ms/step - loss: 0.8157 -
accuracy: 0.6117 - val_loss: 0.7731 - val_accuracy: 0.6182
Epoch 4/10
129/129 [==============================] - 6s 49ms/step - loss: 0.7660 -
accuracy: 0.6412 - val_loss: 0.7004 - val_accuracy: 0.6729
Epoch 5/10
129/129 [==============================] - 6s 48ms/step - loss: 0.6692 -
accuracy: 0.6966 - val_loss: 0.6423 - val_accuracy: 0.7129
Epoch 6/10
129/129 [==============================] - 6s 48ms/step - loss: 0.5324 -
accuracy: 0.7718 - val_loss: 0.8265 - val_accuracy: 0.6572
Epoch 7/10
129/129 [==============================] - 6s 49ms/step - loss: 0.4833 -
accuracy: 0.8018 - val_loss: 0.4976 - val_accuracy: 0.7930
Epoch 8/10
129/129 [==============================] - 6s 48ms/step - loss: 0.3579 -
accuracy: 0.8619 - val_loss: 0.4487 - val_accuracy: 0.8203
Epoch 9/10
129/129 [==============================] - 6s 48ms/step - loss: 0.2537 -
accuracy: 0.9112 - val_loss: 0.3463 - val_accuracy: 0.8662
Epoch 10/10
129/129 [==============================] - 6s 49ms/step - loss: 0.2310 -
accuracy: 0.9151 - val_loss: 0.3526 - val_accuracy: 0.8584
```

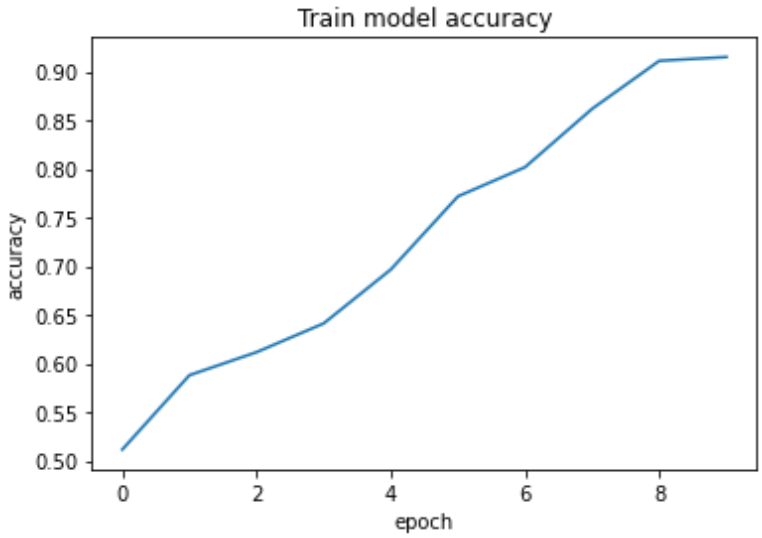
حفظ النموذج:

```
model.save("alz_model1.h5")
```

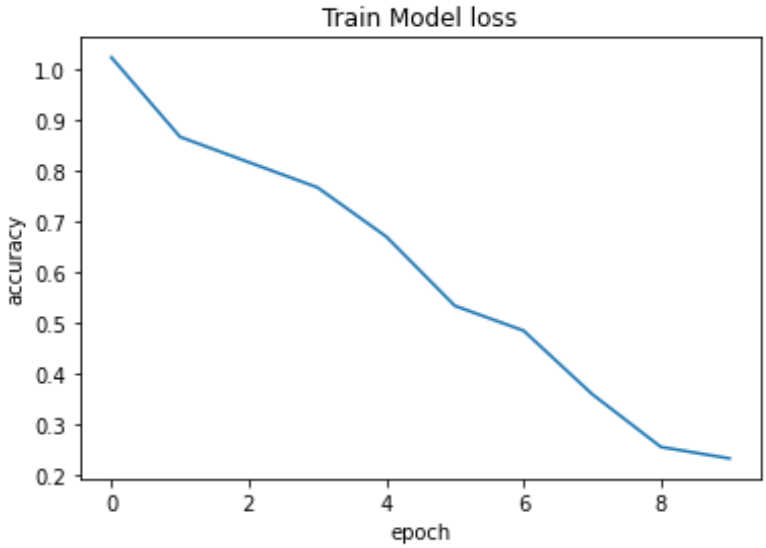
```
model.evaluate(val_data)
```

رسم النتائج:

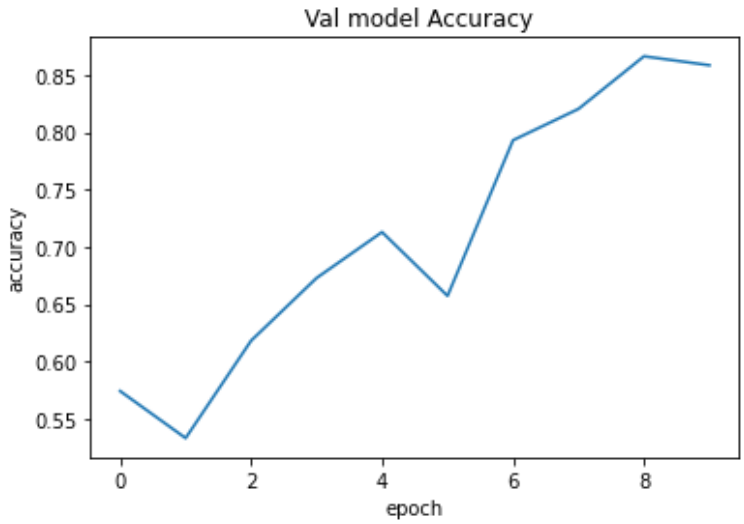
```
plt.plot(history.history['accuracy'])
plt.title('Train model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



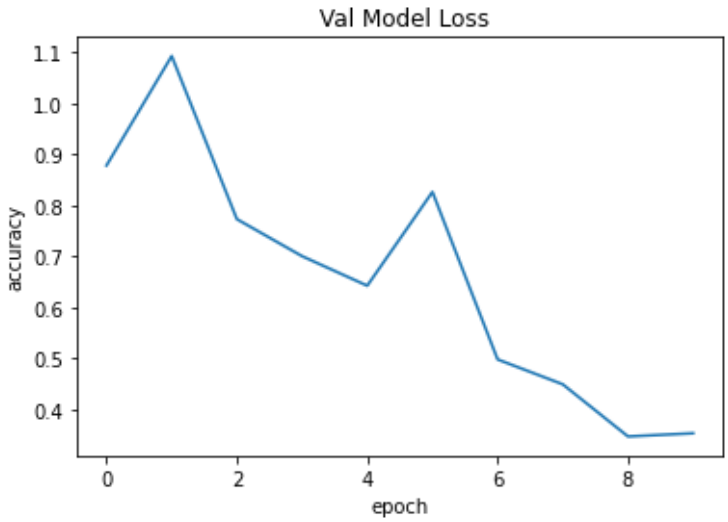

```
plt.plot(history.history['loss'])
plt.title('Train Model loss')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



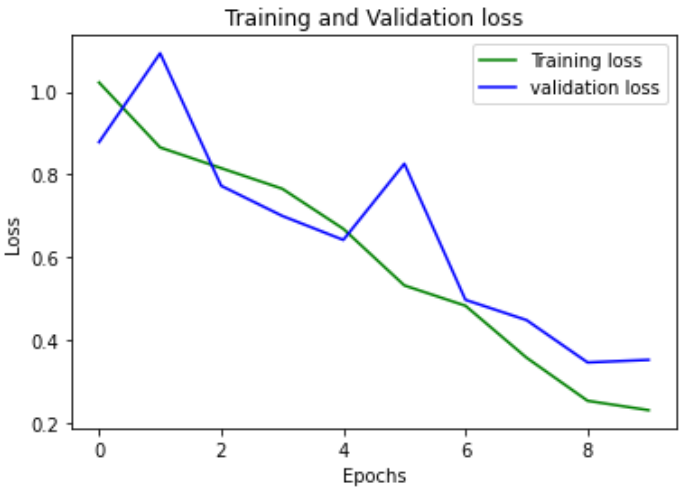
```
plt.plot(history.history['val_accuracy'])
plt.title(' Val model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```




```
plt.plot(history.history['val_loss'])
plt.title(' Val Model Loss')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



```
loss_train = history.history['loss']
loss_val = history.history['val_loss']
plt.plot(loss_train, 'g', label='Training loss')
plt.plot(loss_val, 'b', label='validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
accuracy_train = history.history['accuracy']
accuracy_val = history.history['val_accuracy']
plt.plot(accuracy_train, 'g', label='Training accuracy')
plt.plot(accuracy_val, 'b', label='Validation accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

التنبؤ:

```
class_names={0:"MildDementia", 1:"ModerateDementia", 2:"NonDementia",
3:"VeryMildDementia"}
```

```
for images, labels in val_data.take(1):
    for i in range(6):
        print("True_class:",val_data.class_names[labels[i]])
        x = image.img_to_array(images[i])
        x = np.expand_dims(x, axis=0)
        p=np.argmax(model.predict(x))
        if p==0:
            print("Predicted Image: Mild Dementia")
        elif p==1:
            print("Predicted Image: Moderate Dementia")
        elif p==2:
            print("Predicted Image: Non Dementia")
        else:
            print("Predicted Image: Very Mild Dementia")
```

```
        print("Predicted class:",p)
print("All the predicted images are correct!!!!!")
```

```
True_class: VeryMildDementia
Predicted Image: Very Mild Dementia
Predicted class: 3
True_class: VeryMildDementia
Predicted Image: Non Dementia
Predicted class: 2
True_class: NonDementia
Predicted Image: Non Dementia
Predicted class: 2
True_class: VeryMildDementia
Predicted Image: Very Mild Dementia
Predicted class: 3
True_class: NonDementia
Predicted Image: Non Dementia
Predicted class: 2
True_class: VeryMildDementia
Predicted Image: Very Mild Dementia
Predicted class: 3
All the predicted images are correct!!!!!
```

24) نظام الكشف عن الأشكال ثلاثية الأبعاد باستخدام التعلم العميق 3-D Shape Detection System using Deep learning

تعد تقنيات اكتشاف الأشكال Shape detection techniques جانبًا مهمًا من الرؤية الحاسوبية وتُستخدم لتحويل بيانات الصورة الأولية إلى تمثيلات رمزية مطلوبة للتعرف على الكائن والموقع.

في هذه المقالة، يتم تقديم نوتبوك يحتوي على تطوير نظام يكتشف أربعة أنواع من الأشكال ثلاثية الأبعاد: مكعب Cube، واسطوانة Cylinder، وشبه كروي Spheroid، وكروي Sphere.

النموذج المستخدم مبني على mobilenet v1، مع الاستفادة من فوائد نقل التعلم transfer learning من أجل بناء نموذج خفيف الوزن ولكن دقيق لشبكة CNN.

يتم تنفيذه على منصة Cainvas، التي توفر تنفيذًا سلسًا لأجهزة الكمبيوتر المحمولة من نوع بايثون لبناء أنظمة ذكاء اصطناعي يمكن نشرهافي النهاية على edge (أي نظام مضمن مثل MCUs المدمجة).

يمكن العثور على النوتبوك هنا.

Mobilenet v1 - النموذج الأساسي

تعتمد شبكات MobileNets على بُنية مبسطة تستخدم تلافيف قابلة للفصل بعمق لبناء شبكات عصبية عميقة خفيفة الوزن.

الغرض من استخدام mobilenet لحالة الاستخدام هذه هو أن هذا المشروع يهدف إلى نشره على الأجهزة المحمولة على edge، ومن ثم يكون منطقيًا تمامًا لبناء نموذج يعتمد على فئة من النماذج الفعالة (MobileNets) التي تم تدريبها مسبقًا على نشر مجموعة من نماذج DNN دقيقة الضبط لتطبيقات الرؤية المتنقلة والمدمجة.

تحميل MobileNet:

```
base_model=MobileNet(input_shape=(IMAGE_SIZE, IMAGE_SIZE,3), alpha =
ALPHA,
                     depth_multiplier = 1, dropout = 0.001,
include_top = False,
                     weights = "imagenet", classes = 4,
backend=keras.backend,

layers=keras.layers,models=keras.models,utils=keras.utils)
```

هنا يتم تعيين معلمات النموذج على النحو التالي:

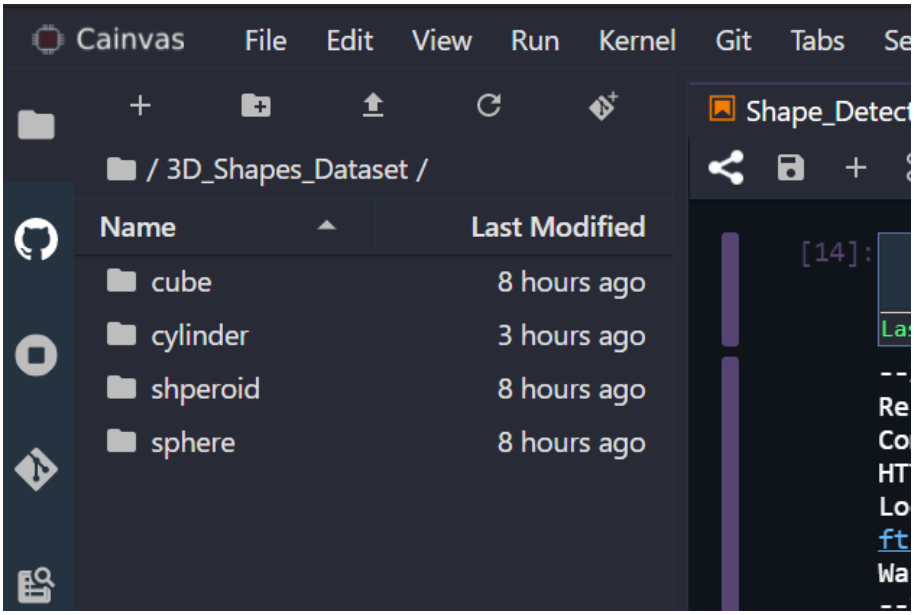
IMAGE_SIZE = 224

ALPHA = 0.75

EPOCHS=20

مجموعة بيانات الأشكال ثلاثية الأبعاد

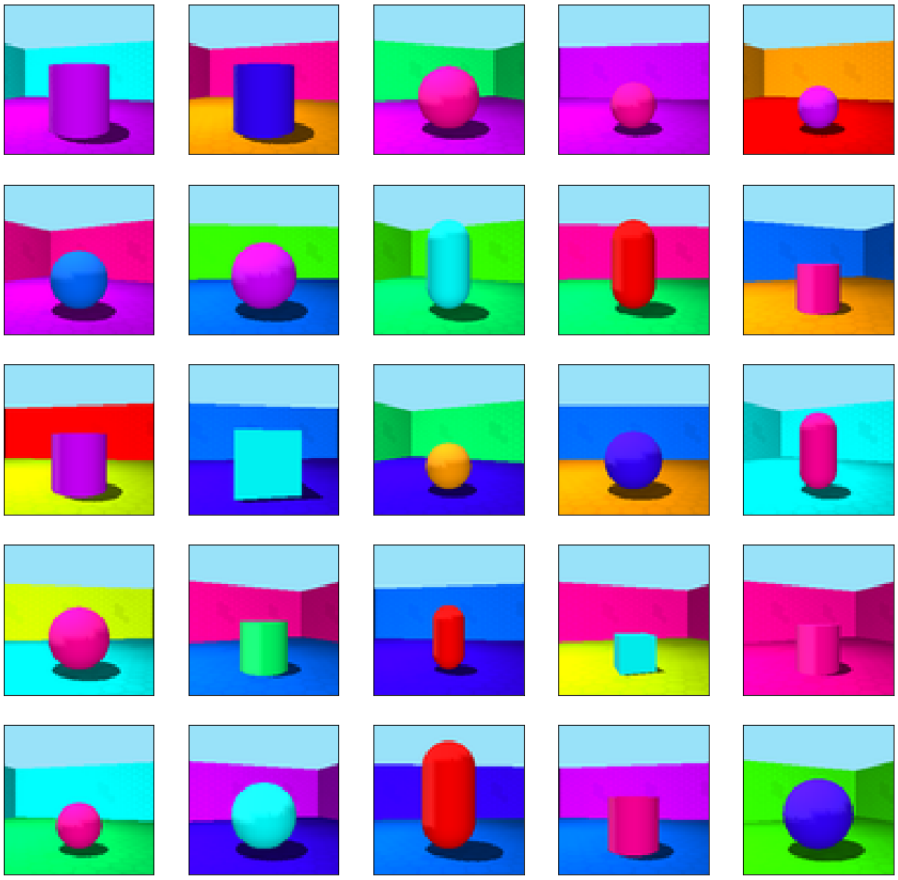
مجموعة البيانات المستخدمة هنا عبارة عن مجموعة بيانات مستخرجة مخصصة مع صور بحجم (224،224). يتكون من 4 أدلة تحتوي على الصور المقابلة لفئات الأشكال الأربعة.



تتم معالجة جميع الصور المستخدمةفي التدريب والاختبار مسبقًا على النحو التالي:

```
def prepare_image(file):
    img = image.load_img(img_path + file, target_size=(IMAGE_SIZE,
IMAGE SIZE))
    img_array = image.img_to_array(img)
    img_array_expanded_dims = np.expand_dims(img_array, axis=0)
    return
keras.applications.mobilenet.preprocess_input(img_array_expanded_dims)
```

رسم عينة من مجموعة بيانات التدريب:



بناء النموذج - نقل التعلم

```
def build_finetune_model(base_model, dropout, fc_layers, num_classes):
    for layer in base_model.layers:
        layer.trainable = False

    x = base_model.output
    x = GlobalAveragePooling2D()(x)

    for fc in fc_layers:
        # New FC layer, random init
        x = Dense(fc, activation='relu')(x)
        x = Dropout(dropout)(x)

    # New softmax layer
    predictions = Dense(num_classes, activation='softmax')(x)

    finetune_model = Model(inputs=base_model.input,
outputs=predictions)
```

```
    return finetune_model

FC_LAYERS = [100, 50]
dropout = 0.5

finetune_model = build_finetune_model(base_model,
                                      dropout=dropout,
                                      fc_layers=FC_LAYERS,
                                      num_classes=4)
```

النموذج المراد يضبط بشكل دقيق fine-tune مبني عن طريق إضافة بضع طبقات إضافية إلى نموذج mobilenet الأساسي.

هنا، نضيف طبقتين كثيفتين متصلتين بالكامل من 100 و 50 خلية عصبية على التوالي مع دالة تنشيط "relu" ونسبة تسرب 0.5، إلى آخر طبقة من mobileenet، وطبقة إخراج نهائية للتنبؤات_ وهي طبقة كثيفة أخرى بها 4 خلايا عصبية ناتجة ودالة تنشيط "softmax". (كل خلية عصبية تتوافق مع فئة إخراج من الأشكال).

تدريب النموذج - الضبط الدقيق

الآن بعد أن تم بناء نموذج تعلم النقل الخاص بنا، يمكننا تدريبه (fine-tune) على مجموعة البيانات المذكورة سابقًا باستخدام keras ImageDataGenerator لمعالجة الصور بشكل أكبر حتى تكون مناسبة لنموذج mobienet الخاص بنا، وبالتالي إنشاء مولد تدريب. (الكود الموضح أدناه):

```
train_datagen=ImageDataGenerator(preprocessing_function=preprocess_inp
ut)

train_generator=train_datagen.flow_from_directory('3D_Shapes_Dataset',

target_size=(IMAGE_SIZE,IMAGE_SIZE),
                                                  color_mode='rgb',
                                                  batch_size=32,

class_mode='categorical', shuffle=True)
```

يتم الآن تجميع نموذج CNN الذي تم إنشاؤه مسبقًا باستخدام مُحسِّن آدم، وخطأ categorical crossentropy ومقاييسفي الاعتبار أثناء التدريب هو دقة النموذج.

يتم بعد ذلك إدخال مولد التدريب المحددفي النموذج الذي تم تجميعه كما هو موضحفي الكود أدناه.

```
finetune_model.summary()
finetune_model.compile(optimizer='Adam',loss='categorical_crossentropy
',metrics=['accuracy'])
step_size_train=train_generator.n//train_generator.batch_size
history =
finetune_model.fit_generator(generator=train_generator,steps_per_epoch
=step_size_train,epochs=EPOCHS, shuffle=True)

finetune_model.save('shape_model.h5')
```

يمكن عرض ملخص النموذج على أنه الإخراج قبل بدء التدريب داخل النوتبوك.

أخيرًا، يتم حفظ النموذج بعد اكتمال التدريب كنموذج keras (.h5).

اختبار النموذج

يحقق النموذج دقة تصل إلى 99٪، ومنذ تصنيفه للشكل الهندسي فقط للكائن، فإنه لا يصلح حتىفي مثل هذه المستويات العالية من الدقة.

تم اختبار النموذج على كائنات من العالم الحقيقي بالإضافة إلى صور الإنترنت لفهم قدراته بشكل أفضل.

فيما يلي النتائج:

```
In [55]: # Real-Life example
         img_path="sample1.jpg"
         predict_shape(img_path)
```

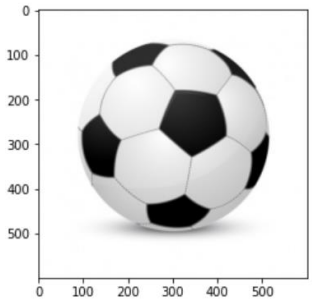
```
Shape Detected:  Cube
```


```
In [56]: # Real-Life example 2
         img_path="sample2.jpg"
         predict_shape(img_path)
```

```
Shape Detected:  Spheroid
```

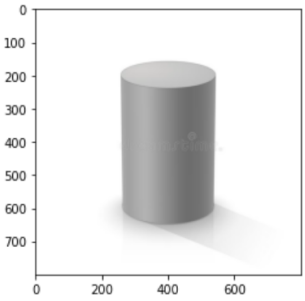
```
In [59]: # Real-Life example 3
         img_path="sample3.jpg"
         predict_shape(img_path)
```

```
Shape Detected:  Sphere
```




```
In [64]: # Real-Life example 4
         img_path="sample4.jpg"
         predict_shape(img_path)
```

```
Shape Detected:  Cylinder
```



الملخص

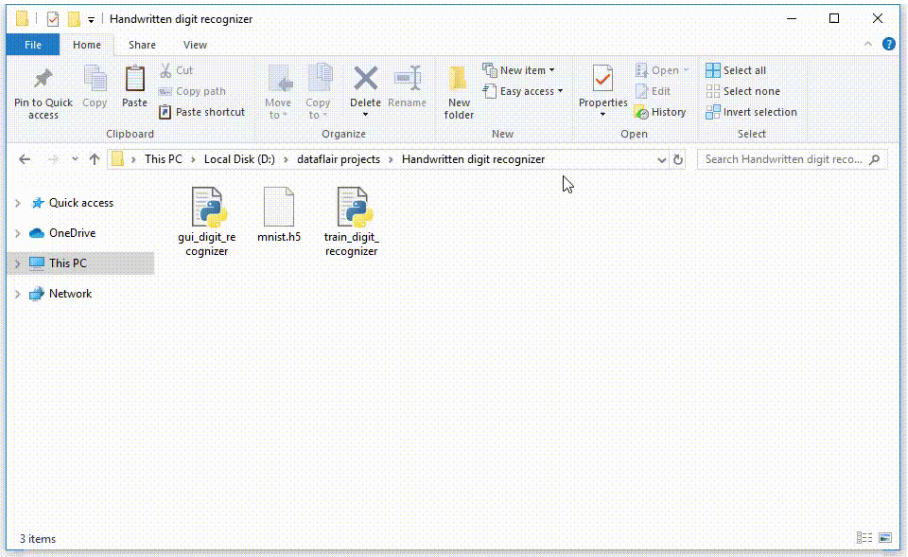
يمكن استخدام نظام الكشف عن الأشكال ثلاثية الأبعاد لتصنيف الكائنات حتىفي الوقت الفعلي. من بين التطورات الإضافيةفي هذا المشروع تحويل نموذج keras CNN إلى نموذج الحد الأدنى من edge القابلة للنشر مثل tflite.أو onnx. من أجل نشر هذا على وحدة نمطية مدمجة AIoT / MCU مثل OpenMV Cam أو Raspberry Pi. هذا النشر ممكن من خلال منصة Cainvasمن خلال الاستفادة من مترجمهم المسمى deepC. وبالتالي يخرج الذكاء الاصطناعي بشكل فعال على edge ـ في حالات استخدام العالم الواقعي والمادي.

25) التعرف على الأرقام المكتوبة بخط اليد باستخدام بايثون Handwritten Digit Recognition using Python

لجعل الآلات أكثر ذكاءً، يغوص المطورونفي تقنيات التعلم الآلي والتعلم العميق. يتعلم الإنسان أداء مهمة ما من خلال ممارستها وتكرارها مرارًا وتكرارًا حتى يحفظ كيفية أداء المهام. ثم تنشط الخلايا العصبيةفي دماغه تلقائيًا ويمكنها أداء المهمة التي تعلموها بسرعة. التعلم العميق هو أيضا مشابه جدا لهذا. يستخدم أنواعًا مختلفة من بُنى الشبكات العصبية لأنواع مختلفة من المشكلات. على سبيل المثال_ التعرف على الكائنات object recognition وتصنيف الصورة والصوت واكتشاف الكائنات object detection وتجزئة الصور image segmentation وما إلى ذلك.

ما هو التعرف على الأرقام المكتوبة بخط اليد؟

التعرف على الأرقام المكتوبة بخط اليد handwritten digit recognition هو قدرة أجهزة الكمبيوتر على التعرف على الأرقام المكتوبة بخط اليد. إنها مهمة صعبة للآلة لأن الأرقام المكتوبة بخط اليد ليست مثالية ويمكن صنعها بالعديد من الاشكال المختلفة. التعرف على الرقم المكتوب بخط اليد هو الحل لهذه المشكلة حيث يستخدم صورة الرقم ويتعرف على الرقم الموجودفي الصورة.



في هذه المقالة، سنقوم بتنفيذ تطبيق التعرف على الأرقام المكتوبة بخط اليد باستخدام مجموعة بيانات MNIST. سنستخدم نوعًا خاصًا من الشبكات العصبية العميقة وهي الشبكات العصبية التلافيفية CNN.في النهاية، سنقوم ببناء واجهة مستخدم رسومية يمكنك من خلالها رسم الرقم والتعرف عليه على الفور.

المتطلبات الأساسية

يتطلب مشروع بايثون المثير أن تكون لديك معرفة أساسية ببرمجة بايثون، وتعلم عميق باستخدام مكتبة Keras ومكتبة Tkinter لبناء واجهة المستخدم الرسومية.

قم بتثبيت المكتبات اللازمة لهذا المشروع باستخدام هذا الأمر:

```
pip install numpy, tensorflow, keras, pillow
```

مجموعة بيانات MNIST

ربما تكون هذه واحدة من أكثر مجموعات البيانات شيوعًا بين التعلم الآلي وهواة التعلم العميق. تحتوي مجموعة بيانات MNIST على 60.000 صورة تدريبية للأرقام المكتوبة بخط اليد من صفر إلى تسعة و10000 صورة للاختبار. لذلك، تحتوي مجموعة بيانات MNIST على 10 فئات مختلفة. يتم تمثيل صور الأرقام المكتوبة بخط اليد كمصفوفة 28 × 28 حيث تحتوي كل خلية على قيمة بكسل بتدرج الرمادي.

قم بتنزيل الكود المصدري الكامل للمشروع:

بناء مشروع بايثون التعلم العميق للتعرف على الأرقام المكتوبة بخط اليد

فيما يلي خطوات تنفيذ مشروع التعرف على الأرقام المكتوبة بخط اليد:

الخطوة 1: استيراد المكتبات وتحميل مجموعة البيانات

أولاً، سنقوم باستيراد جميع الوحدات التي سنحتاجها لتدريب نموذجنا. تحتوي مكتبة Keras بالفعل على بعض مجموعات البيانات وMNIST هي واحدة منها. لذلك يمكننا بسهولة استيراد مجموعة البيانات والبدءفي العمل معها. تعيد لنا طريقة mnist.load_data() بيانات التدريب والتسميات الخاصة بها وكذلك بيانات الاختبار وتسمياتها.

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(x_train.shape, y_train.shape)
```

الخطوة 2: المعالجة المسبقة للبيانات

لا يمكن تغذية بيانات الصورة مباشرةفي النموذج، لذلك نحتاج إلى إجراء بعض العمليات ومعالجة البيانات لجعلها جاهزة لشبكتنا العصبية. أبعاد بيانات التدريب (60000,28,28). سيتطلب نموذج CNN بُعدًا إضافيًا، لذا نعيد تشكيل المصفوفة لتشكيلها (60000,28,28,1).

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x train shape:', x train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

الخطوة 3: إنشاء النموذج

سنقوم الآن بإنشاء نموذج CNN الخاص بنافي مشروع علم بيانات بايثون. يتكون نموذج CNN عمومًا من طبقات تلافيفية convolutional وتجميعية pooling. إنه يعمل بشكل أفضل مع البيانات التي يتم تمثيلها على أنها هياكل شبكية grid structures، وهذا هو السبب في أن CNN تعمل بشكل جيد مع مشاكل تصنيف الصور. تُستخدم طبقة التسرب dropout layer لإلغاء تنشيط بعض الخلايا العصبية وأثناء التدريب، فإنها تقلل من فرط التعلم overfitting للنموذج. سنقوم بعد ذلك بتجميع النموذج باستخدام مُحسِّن Adadelta.

```
batch_size = 128
num_classes = 10
epochs = 10

model = Sequential()
model.add(Conv2D(32, kernel_size=(3,
3),activation='relu',input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=ker
as.optimizers.Adadelta(),metrics=['accuracy'])
```

الخطوة 4: تدريب النموذج

ستبدأ دالة model.fit() الخاصة بـ Keras بتدريب النموذج. يأخذ بيانات التدريب وبيانات التحقق والحقب epochs وحجم الدفعة batch size.

يستغرق تدريب النموذج بعض الوقت. بعد التدريب، نحفظ الأوزان وتعريف النموذجفي ملف "mnist.h5".

```
hist = model.fit(x_train,
y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=
(x_test, y_test))
print("The model has successfully trained")
```

```
model.save('mnist.h5')
print("Saving the model as mnist.h5")
```

الخطوة 5: تقييم النموذج

لدينا 10000 صورةفي مجموعة البيانات الخاصة بنا والتي سيتم استخدامها لتقييم مدى جودة عمل نموذجنا. لم يتم تضمين بيانات الاختبارفي تدريب البيانات، وبالتالي فهي بيانات جديدة لنموذجنا. مجموعة بيانات MNIST متوازنة جيداً حتى نتمكن من الحصول على دقة تصل إلى 99٪.

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

الخطوة 6: إنشاء واجهة المستخدم الرسومية للتنبؤ بالأرقام

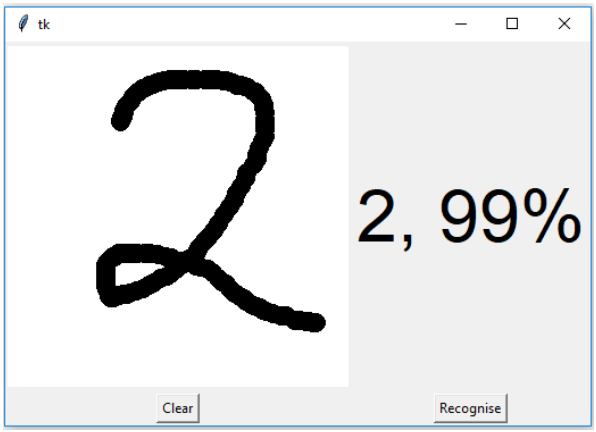
الآن بالنسبة لواجهة المستخدم الرسومية، أنشأنا ملفًا جديداً نبني فيه نافذة تفاعلية لرسم أرقام على canvas وباستخدام زر، يمكننا التعرف على الرقم. تأتي مكتبة Tkinterفي مكتبة بايثون القياسية. لقد أنشأنا دالة predict_digit() تأخذ الصورة كمدخلات ثم تستخدم النموذج المدرب للتنبؤ بالرقم.

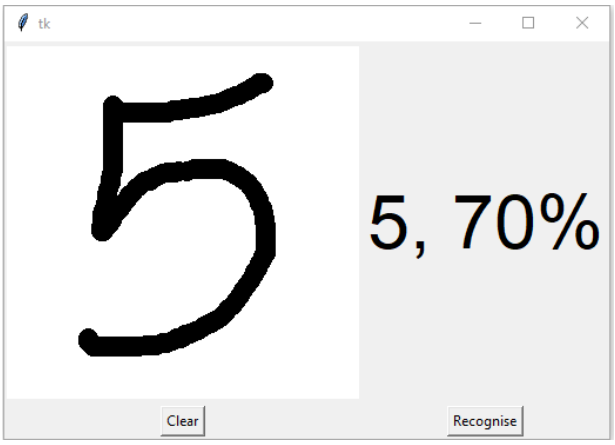
ثم نقوم بإنشاء فئة التطبيقات App class المسؤولة عن إنشاء واجهة المستخدم الرسومية لتطبيقنا. نقوم بإنشاء لوحة canvas حيث يمكننا الرسم عن طريق التقاط حدث الماوس وباستخدام زر، نقوم بتشغيل دالة predict_digit() وعرض النتائج.

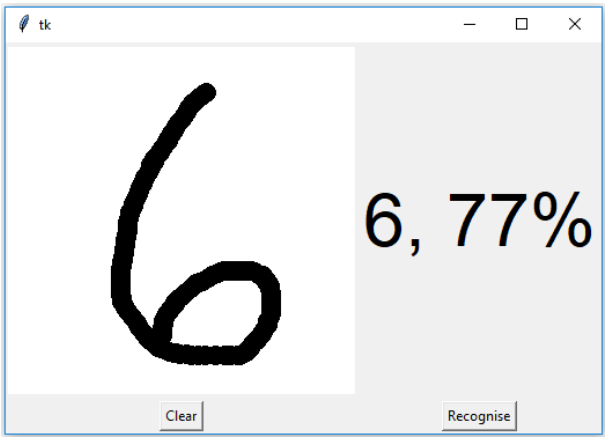
إليك الكود الكامل لملفنا gui_digit_recognizer.py:

```
from keras.models import load_model
from tkinter import *
import tkinter as tk
import win32gui
from PIL import ImageGrab, Image
import numpy as np

model = load_model('mnist.h5')

def predict_digit(img):
    #resize image to 28x28 pixels
    img = img.resize((28,28))
    #convert rgb to grayscale
    img = img.convert('L')
    img = np.array(img)
    #reshaping to support our model input and normalizing
    img = img.reshape(1,28,28,1)
    img = img/255.0
    #predicting the class
    res = model.predict([img])[0]
    return np.argmax(res), max(res)

class App(tk.Tk):
    def __init__(self):
        tk.Tk.__init__(self)
```

```
        self.x = self.y = 0

        # Creating elements
        self.canvas = tk.Canvas(self, width=300, height=300, bg =
"white", cursor="cross")
        self.label = tk.Label(self, text="Thinking..",
font=("Helvetica", 48))
        self.classify_btn = tk.Button(self, text = "Recognise",
command =          self.classify_handwriting)
        self.button_clear = tk.Button(self, text = "Clear", command =
self.clear_all)

        # Grid structure
        self.canvas.grid(row=0, column=0, pady=2, sticky=W, )
        self.label.grid(row=0, column=1,pady=2, padx=2)
        self.classify_btn.grid(row=1, column=1, pady=2, padx=2)
        self.button_clear.grid(row=1, column=0, pady=2)

        #self.canvas.bind("<Motion>", self.start_pos)
        self.canvas.bind("<B1-Motion>", self.draw_lines)

    def clear_all(self):
        self.canvas.delete("all")

    def classify handwriting(self):
        HWND = self.canvas.winfo_id() # get the handle of the canvas
        rect = win32gui.GetWindowRect(HWND) # get the coordinate of
the canvas
        im = ImageGrab.grab(rect)

        digit, acc = predict digit(im)
        self.label.configure(text= str(digit)+', '+
str(int(acc*100))+'%')

    def draw_lines(self, event):
        self.x = event.x
        self.y = event.y
        r=8
        self.canvas.create_oval(self.x-r, self.y-r, self.x + r, self.y
+ r, fill='black')

app = App()
mainloop()
```







الملخص

في هذه المقالة، قمنا ببناء مشروع بايثون للتعلم العميق بنجاح على تطبيق التعرف على الأرقام المكتوبة بخط اليد. لقد قمنا ببناء وتدريب الشبكة العصبية التلافيفية CNN وهي فعالة للغاية لأغراض تصنيف الصور. لاحقًا، قمنا ببناء واجهة المستخدم الرسومية GUI حيث نرسم رقمًا على اللوحة ثم نصنف الرقم ونعرض النتائج.

26) الكشف عن سرطان الرئة باستخدام نقل التعلم Lung Cancer Detection Using Transfer Learning

تُعد الرؤية الحاسوبية Computer Vision أحد تطبيقات الشبكات العصبية العميقة التي تمكننا من أتمتة المهام التي كانت تتطلب سابقًا سنوات من الخبرة وأحد هذه الاستخداماتفي التنبؤ بوجود الخلايا السرطانية.

في هذه المقالة، سوف نتعلم كيفية بناء مصنف باستخدام تقنية نقل التعلم Transfer Learning التي يمكنها تصنيف أنسجة الرئة الطبيعية من السرطانية. تم تطوير هذا المشروع باستخدام Collab وتم أخذ مجموعة البيانات من Kaggle التي تم توفير رابطها أيضًا.

نقل التعلم

في الشبكة العصبية التلافيفية CNN، تتمثل المهمة الرئيسية للطبقات التلافيفيةفي تعزيز السمات المهمة للصورة. إذا تم استخدام فلتر معين لتحديد الخطوط المستقيمةفي صورة ما، فسيعمل مع الصور الأخرى، وهذا ما نقوم به بشكل خاصفي نقل التعلم. هناك نماذج طورها الباحثون عن طريق التراجع عن ضبط المعامل الفائق والتدريب لأسابيع على ملايين الصور التي تنتمي إلى 1000 فئة مختلفة مثل مجموعة بيانات imagenet. النموذج الذي يعمل جيدًا لمهمة الرؤية الحاسوبية واحدة يثبت أنه جيد للآخرين أيضًا. لهذا السبب، فإننا نستفيد من معلمات الطبقات التلافيفية المدربة والمعلمة الفائقة المضبوطة لمهمتنا للحصول على دقة أعلى.

استيراد مكتبات

تجعل مكتبات بايثون من السهل جدًا علينا التعامل مع البيانات وتنفيذ المهام النموذجية والمعقدة باستخدام سطر واحد من التعليمات البرمجية.

- **Pandas**: تساعد هذه المكتبة في تحميل إطار البيانات بتنسيق مصفوفة ثنائية الأبعاد ولها وظائف متعددة لأداء مهام التحليل دفعة واحدة.
- **Numpy**: مصفوفات Numpy سريعة جدًا ويمكنها إجراء عمليات حسابية كبيرة في وقت قصير جدًا.
- **Matplotlib**: تستخدم هذه المكتبة لرسم التمثيلات المرئية.
- **Sklearn**: تحتوي هذه الوحدة على مكتبات متعددة لها وظائف منفذة مسبقًا لأداء المهام من المعالجة المسبقة للبيانات إلى تطوير النماذج وتقييمها.
- **OpenCV**: هذه مكتبة مفتوحة المصدر تركز بشكل أساسي على معالجة الصور والتعامل معها.

- **Tensorflow** : هذه مكتبة مفتوحة المصدر تُستخدم للتعلم الآلي والذكاء الاصطناعي وتوفر مجموعة من الوظائف لتحقيق وظائف معقدة بسطر واحد من التعليمات البرمجية.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
from glob import glob

from sklearn.model_selection import train_test_split
from sklearn import metrics

import cv2
import gc
import os

import tensorflow as tf
from tensorflow import keras
from keras import layers

import warnings
warnings.filterwarnings('ignore')
```

استيراد مجموعة البيانات

مجموعة البيانات التي سنستخدمها هنا مأخوذة من:

https://www.kaggle.com/datasets/andrewmvd/lung-and-colon-cancer-histopathological-images

تتضمن مجموعة البيانات هذه 5000 صورة لثلاث فئات من أمراض الرئة:

- فئة عادية Normal Class
- الأورام الغدية في الرئة Lung Adenocarcinomas
- سرطان الخلايا الحرشفية في الرئة Lung Squamous Cell Carcinomas

تم تطوير هذه الصور لكل فئة من 250 صورة عن طريق إجراء زيادة البيانات Data Augmentation عليها. لهذا السبب لن نستخدم زيادة البيانات بشكل أكبرفي هذه الصور.

```
from zipfile import ZipFile
data_path = 'lung-and-colon-cancer-histopa\
thological-images.zip'

with ZipFile(data_path,'r') as zip:
  zip.extractall()
  print('The data set has been extracted.')
```

المخرجات:

```
The data set has been extracted.
```

العرض المرئي للبيانات

في هذا القسم، سنحاول فهم تصور بعض الصور التي تم توفيرها لنا لبناء المصنف لكل فئة

```
path = '/lung_colon_image_set/lung_image_sets'
classes = os.listdir(path)
classes
```

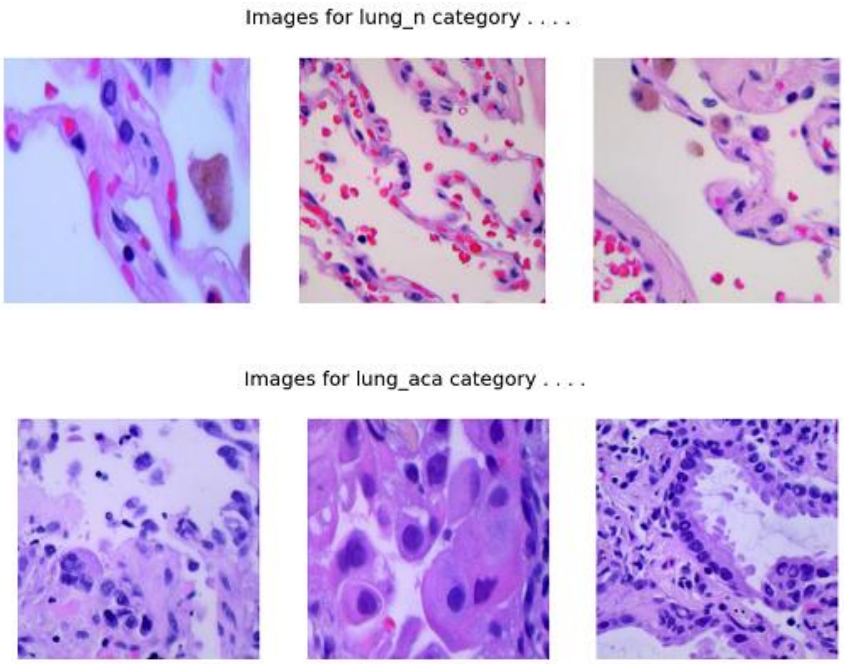
المخرجات:

```
['lung_n', 'lung_aca', 'lung_scc']
```

هذه هي الفئات الثلاث التي لدينا هنا.

```
path = '/lung_colon_image_set/lung_image_sets'

for cat in classes:
    image_dir = f'{path}/{cat}'
    images = os.listdir(image_dir)

    fig, ax = plt.subplots(1, 3, figsize = (15, 5))
    fig.suptitle(f'Images for {cat} category . . . .',
                 fontsize = 20)

    for i in range(3):
        k = np.random.randint(0, len(images))
        img = np.array(Image.open(f'{path}/{cat}/{images[k]}'))
        ax[i].imshow(img)
        ax[i].axis('off')
    plt.show()
```

المخرجات:



قد يختلف الإخراج أعلاه إذا كنت ستقوم بتشغيل هذافي النوتبوك الخاص بك لأن الكود قد تم تنفيذه بطريقة تعرض صورًا مختلفةفي كل مرة تقوم فيها بإعادة تشغيل الكود.

تحضير البيانات للتدريب

في هذا القسم، سنقوم بتحويل الصور المعطاة إلى مصفوفات NumPy لوحدات البكسل الخاصة بها بعد تغيير حجمها لأن تدريب الشبكة العصبية العميقة على الصور كبيرة الحجم غير فعال للغاية من حيث التكلفة والوقت الحسابيين.

لهذا الغرض، سوف نستخدم مكتبة OpenCV ومكتبة Numpy للبايثون لخدمة الغرض. أيضًا، بعد تحويل جميع الصور إلى التنسيق المطلوب، سنقسمها إلى بيانات تدريب وتحقق من الصحة، حتى نتمكن من تقييم أداء نموذجنا.

```
IMG_SIZE = 256
SPLIT = 0.2
EPOCHS = 10
BATCH_SIZE = 64
```

بعض المعلمات الفائقة hyperparameters التي يمكننا تعديلها من هنا للنوتبوك بأكملها.

```
X = []
Y = []

for i, cat in enumerate(classes):
  images = glob(f'{path}/{cat}/*.jpeg')

  for image in images:
    img = cv2.imread(image)

    X.append(cv2.resize(img, (IMG_SIZE, IMG_SIZE)))
    Y.append(i)

X = np.asarray(X)
one_hot_encoded_Y = pd.get_dummies(Y).values
```

سيساعدنا واحد_الترميز الساخن One hot encodingفي تدريب نموذج يمكنه التنبؤ بالاحتمالات اللينة لصورة ما من كل فئة مع أعلى احتمال للفئة التي تنتمي إليها حقًا.

المخرجات:

```
(12000, 256, 256, 3) (3000, 256, 256, 3)
```

في هذه الخطوة، سنحقق خلط shuffling البيانات تلقائيًا لأن دالة train_test_split تقسم البيانات عشوائيًافي النسبة المحددة.

تطوير النموذج

سنستخدم أوزانًا مدربًا مسبقًا لشبكة Inception التي يتم تدريبها على مجموعة بيانات imagenet. تحتوي مجموعة البيانات هذه على ملايين الصور لحوالي 1000 فئة من الصور.

معمارية النموذج

سنقوم بتنفيذ نموذج باستخدام Functional API لـ Keras والتي ستحتوي على الأجزاء التالية:

- النموذج الأساسي base model هو نموذج البداية في هذه الحالة.
- تعمل الطبقة المسطحة Flatten layer على تسطيح مخرجات إخراج النماذج الأساسية.
- ثم سيكون لدينا طبقتان متصلتان تمامًا fully connected layers متبوعًا بإخراج الطبقة المسطحة.
- لقد قمنا بتضمين بعض طبقات BatchNormalization لتمكين تدريب مستقر وسريع وطبقة Dropout قبل الطبقة النهائية لتجنب أي احتمال للضبط الزائد overfitting.
- الطبقة الأخيرة هي طبقة المخرجات output layer التي تُخرج الاحتمالات للفئات الثلاث.

```
from tensorflow.keras.applications.inception_v3 import InceptionV3

pre trained model = InceptionV3(
    input_shape = (IMG_SIZE, IMG_SIZE, 3),
    weights = 'imagenet',
    include_top = False
)
```

المخرجات:

```
87916544/87910968 [==============================] – 2s 0us/step
87924736/87910968 [==============================] – 2s 0us/step
```

```
len(pre_trained_model.layers)
```

المخرجات:

```
311
```

هذا هو مدى عمق هذا النموذج، وهذا يبرر أيضًا سبب فعالية هذا النموذجفي استخراج الميزات المفيدة من الصور التي تساعدنافي بناء المصنفات.

معلمات النموذج الذي نستورده مدربة بالفعل على ملايين الصور ولأسابيع، لا نحتاج إلى تدريبهم مرة أخرى.

```
for layer in pre_trained_model.layers:
  layer.trainable = False
```

تعد "Mixed7" إحدى الطبقاتفي شبكة البداية التي سنستخدم مخرجاتها لبناء المصنف.

```
last_layer = pre_trained_model.get_layer('mixed7')


print('last layer output shape: ', last_layer.output_shape)


last_output = last_layer.output
```

المخرجات:

```
last layer output shape:  (None, 14, 14, 768)
```

```
x = layers.Flatten()(last_output)

x = layers.Dense(256,activation='relu')(x)
x = layers.BatchNormalization()(x)

x = layers.Dense(128,activation='relu')(x)
x = layers.Dropout(0.3)(x)
x = layers.BatchNormalization()(x)

output = layers.Dense(3, activation='softmax')(x)

model = keras.Model(pre_trained_model.input, output)
```

Callback

يتم استخدام عمليات Callbacks للتحقق مما إذا كان النموذج يتحسن مع كل حقبة أم لا. إذا لم يكن الأمر كذلك، فما هي الخطوات الضرورية التي يجب اتخاذها مثل ReduceLROnPlateau لتقليل معدل التعلم بشكل أكبر؟ حتى إذا لم يتحسن أداء النموذج، فسيتم إيقاف التدريب بواسطة التوقف المبكر EarlyStopping. يمكننا أيضًا تحديد بعض عمليات Callbacks المخصصة لإيقاف التدريب فيما بينها إذا تم الحصول على النتائج المرجوة مبكرًا.

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau

class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs = {}):
    if logs.get('val_accuracy') > 0.90:
      print('\n Validation accuracy has reached upto 90%\
```

```
      so, stopping further training.')
      self.model.stop_training = True

es = EarlyStopping(patience = 3,
                   monitor = 'val_accuracy',
                   restore_best_weights = True)

lr = ReduceLROnPlateau(monitor = 'val_loss',
                       patience = 2,
                       factor = 0.5,
                       verbose = 1)
```

الآن سنقوم بتدريب نموذجنا:

```
history = model.fit(X_train, Y_train,
                    validation_data = (X_val, Y_val),
                    batch_size = BATCH_SIZE,
                    epochs = EPOCHS,
                    verbose = 1,
                    callbacks = [es, lr, myCallback()])
```

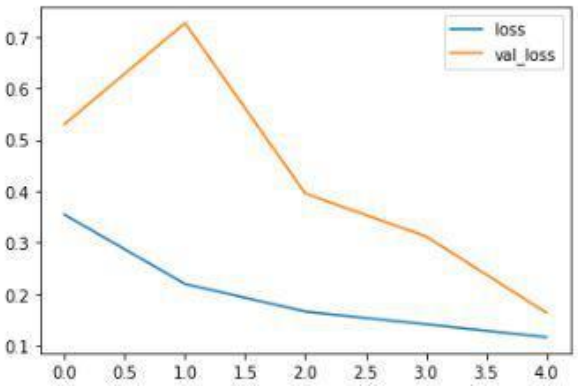
المخرجات:

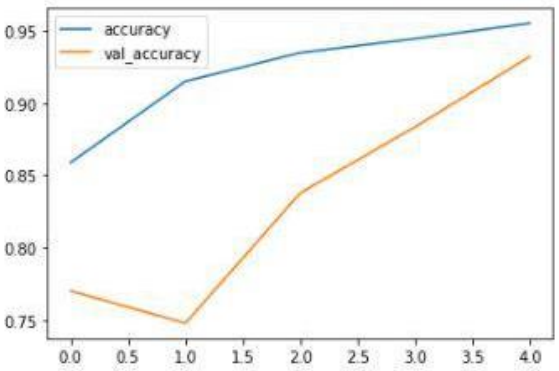
```
Epoch 1/10
188/188 [==============================] - 52s 189ms/step - loss: 0.3543 - accuracy: 0.8587 - val_loss: 0.5301 - val_accuracy: 0.7700 - lr: 0.0010
Epoch 2/10
188/188 [==============================] - 32s 169ms/step - loss: 0.2196 - accuracy: 0.9149 - val_loss: 0.7268 - val_accuracy: 0.7477 - lr: 0.0010
Epoch 3/10
188/188 [==============================] - 33s 174ms/step - loss: 0.1656 - accuracy: 0.9347 - val_loss: 0.3951 - val_accuracy: 0.8377 - lr: 0.0010
Epoch 4/10
188/188 [==============================] - 33s 176ms/step - loss: 0.1410 - accuracy: 0.9443 - val_loss: 0.3120 - val_accuracy: 0.8833 - lr: 0.0010
Epoch 5/10
188/188 [==============================] - ETA: 0s - loss: 0.1160 - accuracy: 0.9550
 Validation accuracy has reached upto 90% so, stopping further training.
188/188 [==============================] - 33s 173ms/step - loss: 0.1160 - accuracy: 0.9550 - val_loss: 0.1633 - val_accuracy: 0.9320 - lr: 0.0010
```

دعونا نرسم دقة التدريب والتحقق من الصحة مع كل حقبة.

```
history_df = pd.DataFrame(history.history)
history_df.loc[:,['loss','val_loss']].plot()
history_df.loc[:,['accuracy','val_accuracy']].plot()
plt.show()
```

المخرجات:





من الرسوم البيانية أعلاه، يمكننا بالتأكيد أن نقول إن النموذج لم يضبط بشكل زائد بيانات التدريب لأن الفرق بين دقة التدريب والتحقق من الصحة منخفض للغاية.

تقييم النموذج

الآن بعد أن أصبح نموذجنا جاهزًا، فلنقم بتقييم أدائه على بيانات التحقق من الصحة باستخدام مقاييس مختلفة. لهذا الغرض، سوف نتنبأ أولاً بفئة بيانات التحقق باستخدام هذا النموذج ثم نقارن المخرجات بالتسميات الحقيقية.

```
_pred = model.predict(X_val)

Y_val = np.argmax(Y_val, axis=1)
Y_pred = np.argmax(Y_pred, axis=1)
```

دعنا نرسم مصفوفة الارتباك confusion metrics وتقرير التصنيف classification report باستخدام التسميات المتوقعة والتسميات الحقيقية.

```
metrics.confusion_matrix(Y_val, Y_pred)
```

المخرجات:

```
array([[ 859,  127,    1],
       [  48,  923,    6],
       [   0,   22, 1014]])
```

```
print(metrics.classification_report(Y_val, Y_pred,
                                    target_names=classes))
```

المخرجات:

	precision	recall	f1-score	support
lung_scc	0.95	0.87	0.91	987
lung_aca	0.86	0.94	0.90	977
lung_n	0.99	0.98	0.99	1036
accuracy			0.93	3000
macro avg	0.93	0.93	0.93	3000
weighted avg	0.93	0.93	0.93	3000

الملخص

في الواقع، حقق أداء نموذجنا باستخدام تقنية نقل التعلم دقة أعلى دون الضبط الزائد وهو أمر جيد جدًا حيث أن f1-score لكل فئة أعلى أيضًا من 0.90 مما يعني أن توقع نموذجنا صحيح بنسبة 90٪ من الوقت.

27) الكشف عن نوبات الصرع من بيانات EEG Detecting Epileptic Seizures from EEG Data

بيان المشكلة

ما هو أداء خوارزميات التعلم العميقفي التمييز بين أشكال موجات مخطط كهربية الدماغ الصرع (EEG) وأشكال موجات EEG غير الصرع؟

مجموعة البيانات

مجموعة بيانات التعرف على نوبات الصرع، مستودع التعلم الآلي UCI.

مجموعة البيانات هذه عبارة عن نسخة تمت معالجتها مسبقًا وإعادة هيكلتها / إعادة تشكيلها من مجموعة بيانات الصرع بجامعة بون. إنه يتألف:

- 11500 عينة من 178 نقطة بيانات (178 نقطة بيانات = ثانية واحدة من تسجيل مخطط كهربية الدماغ).
- 11500 هدف مع 5 فئات: يمثل 1 أشكال موجات نوبات صرع بينما يمثل 2ـ5 أشكال موجات نوبات صرع غير مصابة بالصرع.

الأدوات المستخدمة:

- بايثون.
- Google Colab Notebook.

لقد استخدمت نوتبوك Google Colab Notebook لهذا المشروع بأكمله وقمت بتحميل مجموعة البيانات على Google Drive. نبدأ بتركيب مجموعة البيانات المخزنة على Google Driveفي Colab Notebook.

```
from google.colab import drive
drive.mount('/content/drive')
```

ستظهر واجهة منبثقة عند تشغيل هذه الخلية. افتح الرابط وانسخ رمز التفويضفي المطالبة لتركيب Google Drive على دفتر Colab Notebook الحالي.

بعد ذلك، سنقوم بتحميل مجموعة البيانات.

```
data = "/content/drive/My Drive/Colab Notebooks/EEG/data.csv"
import pandas as pd
df = pd.read_csv(data, header=0, index_col=0)
```

دعونا نستكشف مجموعة البيانات.

```
df.head()
```

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X170	X171	X172	X173	X174	X175	X176	X177	X178	y
X21.V1.791	135	190	229	223	192	125	55	-9	-33	-38	...	-17	-15	-31	-77	-103	-127	-116	-83	-51	4
X15.V1.924	386	382	356	331	320	315	307	272	244	232	...	164	150	146	152	157	156	154	143	129	1
X8.V1.1	-32	-39	-47	-37	-32	-36	-57	-73	-85	-94	...	57	64	48	19	-12	-30	-35	-35	-36	5
X16.V1.60	-105	-101	-96	-92	-89	-95	-102	-100	-87	-79	...	-82	-81	-80	-77	-85	-77	-72	-69	-65	5
X20.V1.54	-9	-65	-98	-102	-78	-48	-16	0	-21	-59	...	4	2	-12	-32	-41	-65	-83	-89	-73	5

هناك 178 نقطة بيانات (X1 إلى X178) والهدف موجودفي العمود "y" لإطار البيانات

```
df.info()
```

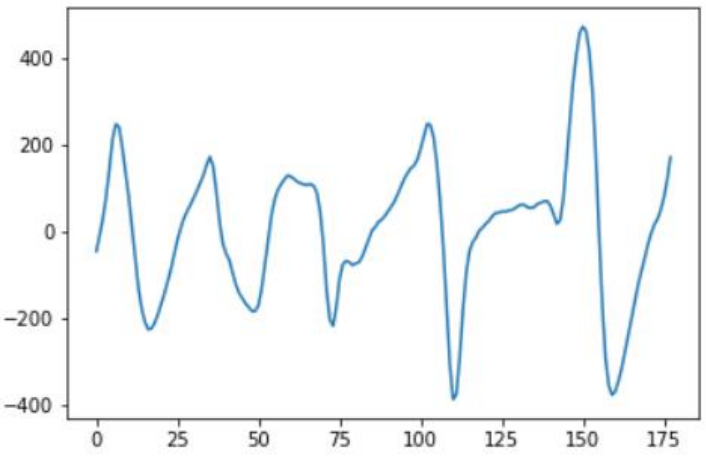
```
<class 'pandas.core.frame.DataFrame'>
Index: 11500 entries, X21.V1.791 to X16.V1.210
Columns: 179 entries, X1 to y
dtypes: int64(179)
```

بعد ذلك، سنحول المتغير المستهدف إلى نوبة صرع (مشفرة كـ 1في العمود "y") مقابل نوبة غير صرع (2–5)

```
df["seizure"] = 0
for i in range(11500):
if df["y"][i] == 1:
df["seizure"][i] = 1
else:
df["seizure"][i] = 0
```

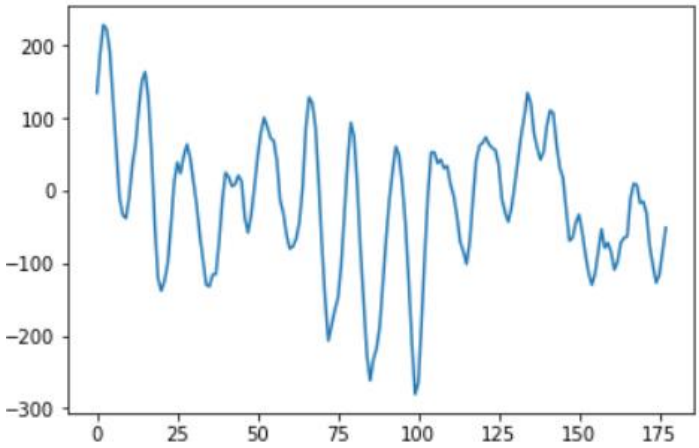
دعونا نرسم ونلقي نظرة على بعض أشكال الموجات EEG.

```
import matplotlib.pyplot as plt# plotting an epileptic wave form
plt.plot(range(178), df.iloc[11496,0:178])
plt.show()
```



شكل موجة صرع

```
plt.plot(range(178), df.iloc[0,0:178])
plt.show()
```



من الصعب تحديد الفرق بمجرد النظر إلى أشكال الموجة بشكل صحيح؟ دعونا نرى ما إذا كان بإمكان شبكتنا العصبية أن تعمل بشكل أفضل.

سنقوم الآن بإعداد البياناتفي شكل مقبول للشبكة العصبية. سنقوم أولاً بتحليل البيانات، ثم نقوم بتوحيد القيم وأخيراً إنشاء المصفوفة الهدف.

```
# create df1 which only contains the waveform data points
df1 = df.drop(["seizure", "y"], axis=1)# 1. parse the data
import numpy as np
wave = np.zeros((11500, 178))
for index, row in df1.iterrows():
    wave[index,:] = row# print the wave.shape to make sure we parsed the
data correctly
print(wave.shape) > returned (11500, 178)# 2. standardize the data such
that it has mean of 0 and standard deviation of 1mean = wave.mean(axis=0)
wave -= mean
std = wave.std(axis=0)
wave /= std# 3. create the target numpy array
target = df["seizure"].values
```

لقد استخدمت Keras لبناء شبكة كثيفة dense network مع التنظيم regularization والتسرب dropout لتقليل فرط التعلم overfitting.

```
from keras.models import Sequential
from keras import layers
from keras import regularizersmodel = Sequential()
model.add(layers.Dense(64, activation="relu",
kernel_regularizer=regularizers.l1(0.001), input_shape = (178,)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(64, activation="relu",
kernel_regularizer=regularizers.l1(0.001)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation="sigmoid"))model.summary()
```

تساعدنا دالة train_test_split من sklearnفي إنشاء مجموعات تدريب واختبار.

```
from sklearn.model_selection import train_test_splitx_train, x_test,
y train, y test = train test split(wave, target, test size=0.2,
random_state=42)
```

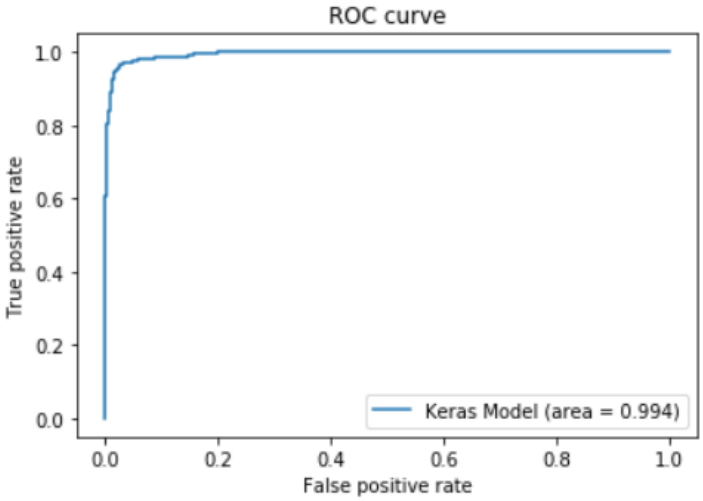
الآن، دعونا نجمع compile النموذج وندربه على 100 حقبة.

```
model.compile(optimizer="rmsprop",
loss="binary_crossentropy", metrics=["acc"])history =
model.fit(x_train, y_train, epochs=100, batch_size=128,
validation_split=0.2, verbose=2)
```

بعد 100 حقبة، حققت دقة التحقق من الصحة بحوالي 96 ـ 97٪.

أخيرًا، سنطلق العنان للنموذج في مجموعة الاختبار، ونرسم منحنى ROC ونحسب AUC.

```
from sklearn.metrics import roc_curve, aucy_pred =
model.predict(x_test).ravel()
fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_test, y_pred)
AUC = auc(fpr_keras, tpr_keras)plt.plot(fpr_keras, tpr_keras, label='Keras
Model(area = {:.3f})'.format(AUC))
plt.xlabel('False positive Rate')
plt.ylabel('True positive Rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```



نموذج التعلم العميق البسيط هذا قادر على تحقيق AUC بقيمة 0.994.

مجالات أخرى للتجربة لتحسين AUC:

- ضبط المعلمات الفائقة Hyperparameter.
- باستخدام بُنية مختلفة على سبيل المثال 1DConvnet.

28) الكشف عن الأغنام باستخدام التعلم العميق Sheep Detection using deep learning

الاستخدام الرئيسي لهذا التطبيق هو الكشف عن الأغنام من الصورة، والتي يمكن استخدامها بعد ذلك لأغراض عديدة مثل عد الأغنام وتتبع الأغنامفي الفناء.

سيساعد المزارع على الاحتفاظ بسجل مناسب للأغنام هناك أيضًا.

إذا هيا بنا نبدأ

مجموعة البيانات: https://www.kaggle.com/intelecai/sheep-detection

تحتوي مجموعة البيانات هذه على 203 صورة للأغنام. بشكل أساسي، صور الأغنام مع التعليقات التوضيحية للمربع المحيط بتنسيق باسكال VOC

استيراد المكتبات الضرورية

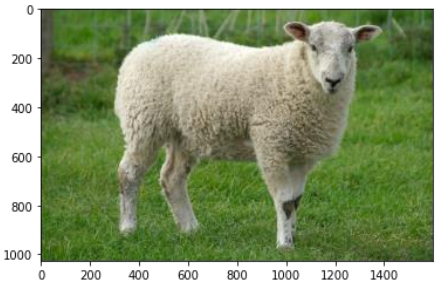
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, callbacks, optimizers
from sklearn.metrics import confusion_matrix, f1_score
from tensorflow import keras
import os, shutil
import random
from PIL import Image
import cv2
```

مجموعة البيانات

```
!wget https://cainvas-
static.s3.amazonaws.com/media/user_data/Rodio346/Sheep_Dataset.zip
!unzip -qo "Sheep_Dataset.zip"
```

النظر في مجموعة البيانات

```
img = Image.open("Sheep Dataset/Yes/17.jpg")
plt.imshow(img)
```




```
import tensorflow as tf
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import cv2
import imutils
import matplotlib.pyplot as plt
from os import listdir
import time
```

تطبيق زيادة البيانات

نظرًا لعدم وجود العديد من الصور، نحتاج إلى تطبيق زيادة البيانات data augmentation.

تساعدنا زيادة البيانات من خلال تكرار الصور أثناء تطبيق الإمالة والدوران والأساليب الأخرى التي لا تتلاعب بالكائن الرئيسي ولكنها تغيره قليلاً وبالتالي تحافظ على الهدف الرئيسي سليمًا.

```
def augment_data(file_dir, n_generated_samples, save_to_dir):
    """
    Arguments:
        file_dir: A string representing the directory where images that we
want to augment are found.
        n_generated_samples: A string representing the number of generated
samples using the given image.
        save_to_dir: A string representing the directory in which the
generated images will be saved.
    """

    #from keras.preprocessing.image import ImageDataGenerator
    #from os import listdir

    data_gen = ImageDataGenerator(rotation_range=10,
                                  width_shift_range=0.1,
                                  height_shift_range=0.1,
                                  shear_range=0.1,
                                  brightness_range=(0.3, 1.0),
                                  horizontal_flip=True,
                                  vertical_flip=True,
                                  fill_mode='nearest'
                                 )


    for filename in listdir(file_dir):
        # load the image
        if filename ==".ipynb_checkpoints":
            continue
        image = cv2.imread(file_dir +'/' + filename)
        # reshape the image
        image = cv2.resize(image,(240,240))
        image = image.reshape((1,)+image.shape)
        # prefix of the names for the generated sampels.
        save_prefix = 'aug_' + filename[:-4]
        # generate 'n_generated_samples' sample images
        i=0
        for batch in data_gen.flow(x=image, batch_size=1,
save_to_dir=save_to_dir,
                                   save_prefix=save_prefix,
save_format='jpg'):
            i += 1
            if i > n_generated_samples:
                break
```

```
data_dir = 'Sheep Dataset/'

batch_size = 64
# image_size = (32, 32)
image_size = (240, 240)

print("Training set")
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
                data_dir,
                validation_split=0.2,
                subset="training",
                color_mode="grayscale",
                image_size=image_size,
                seed=113,
                shuffle=True,
                batch_size=batch_size
            )

print("Validation set")
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
                data_dir,
                validation_split=0.2,
                subset="validation",
                color_mode="grayscale",
                image_size=image_size,
                seed=113,
                shuffle=True,
                batch_size=batch_size
```

```
Training set
Found 2022 files belonging to 2 classes.
Using 1618 files for training.
Validation set
Found 2022 files belonging to 2 classes.
Using 404 files for validation.
```

```
class_names = train_ds.class_names
print(class_names)
```

```
['No', 'Yes']
```

```
Xtrain = np.empty((0,*image_size,1))
ytrain = np.empty((0,1))

for x in train_ds.enumerate():
    for y in x[1][0]:
        Xtrain = np.append(Xtrain, np.expand_dims(np.array(y),0), axis = 0)
    #print(Xtrain.shape)
    ytrain = np.append(ytrain, np.array(x[1][1]))
    #print(ytrain.shape)

Xtrain.shape, ytrain.shape
```

```
((1618, 240, 240, 1), (1618,))
```

```
print("Number of samples - ")
for i in range(len(class_names)):
    print(class_names[i], "-", ytrain.tolist().count(float(i)))
```

```
Number of samples -
No - 1287
Yes - 331
```

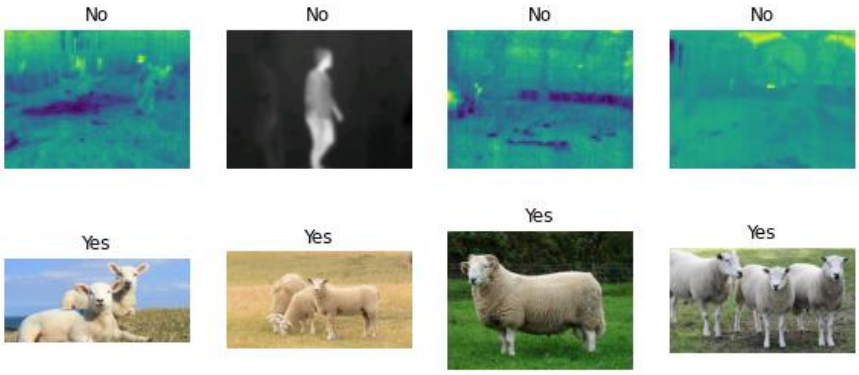
```
data_augmentation = tf.keras.Sequential(
    [
        layers.experimental.preprocessing.RandomFlip("horizontal"),    # Flip
along vertical axes
        layers.experimental.preprocessing.RandomZoom(0.1),    # Randomly zoom
images in dataset
    ])


print("Train size (number of samples) before augmentation: ", len(Xtrain))

aug_sample_count = ytrain.tolist().count(float(0.0))//2 -
ytrain.tolist().count(float(1.0))
cur_augmented = 0
# Apply only to train set
while(cur_augmented!=aug_sample_count):
    for i in range(len(Xtrain)):
        if ytrain[i] == 1: # not elephant
            aug_image = np.array(data_augmentation(np.expand_dims(Xtrain[0],
0)))
            Xtrain = np.append(Xtrain, aug_image.reshape((1, *image_size,
1)), axis = 0)
            ytrain = np.append(ytrain, [1])

            cur_augmented += 1
            if (cur_augmented == aug_sample_count):
                break


print("Size (number of samples) of final dataset: ", len(Xtrain))

print(" Dataset shapes: ", Xtrain.shape, ytrain.shape)
```

```
Train size (number of samples) before augmentation:  1618
Size (number of samples) of final dataset:  1930
 Dataset shapes:  (1930, 240, 240, 1) (1930,)
```

رسم العينات

دعونا نلقي نظرة على العينات:

```
num_samples = 4    # the number of samples to be displayed in each class

for x in class_names:
    plt.figure(figsize=(10, 10))

    filenames = os.listdir(data_dir + x)

    for i in range(num_samples):
        ax = plt.subplot(1, num_samples, i + 1)
        img = Image.open(os.path.join(data_dir, x, filenames[i]))
        plt.imshow(img)
        plt.title(x)
        plt.axis("off")
```




```
Xtrain = Xtrain/255
Xval = Xval/255
```

بناء النموذج

```
model = keras.models.Sequential([
    layers.Conv2D(8, 3, activation='relu', input_shape=Xtrain[0].shape),
    layers.MaxPool2D(pool_size=(2, 2)),

    layers.Conv2D(16, 3, activation='relu'),
    layers.MaxPool2D(pool_size=(2, 2)),

    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPool2D(pool_size=(2, 2)),

    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPool2D(pool_size=(2, 2)),


    layers.Flatten(),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

```
cb = [callbacks.EarlyStopping(monitor = 'val_loss', patience = 5,
restore_best_weights = True)]
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 238, 238, 8)       80
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 119, 119, 8)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 117, 117, 16)      1168
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 58, 58, 16)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 56, 56, 32)        4640
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 28, 28, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 26, 26, 32)        9248
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 5408)              0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                173088
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 188,257
Trainable params: 188,257
Non-trainable params: 0
```

تم تدريب هذا النموذج باستخدام binary cross entropy باستخدام Adam بمعدل تعلم 0.0001. كما يتم استخدام عمليات callbacks لإدخال الإيقاف المبكر early stopping. بمساعدة التوقف المبكر، نوقف النموذج عن مزيد من التدريب من خلال مراقبة المعلمات المحددة.

تجميع وتدريب النموذج

```
model.compile(loss=keras.losses.BinaryCrossentropy(),
optimizer=optimizers.Adam(0.0001), metrics=['accuracy'])
```

```
history = model.fit(Xtrain, ytrain, validation_data=(Xval, yval), epochs=300,
callbacks=cb)
```

```
61/61 [==============================] - 1s 17ms/step - loss: 0.0900 - accuracy: 0.9756 - val_loss: 0.0670 - val_accuracy: 0.9802
Epoch 14/300
61/61 [==============================] - 1s 17ms/step - loss: 0.0716 - accuracy: 0.9865 - val_loss: 0.0435 - val_accuracy: 0.9851
Epoch 15/300
61/61 [==============================] - 1s 17ms/step - loss: 0.0644 - accuracy: 0.9860 - val_loss: 0.0291 - val_accuracy: 0.9901
Epoch 16/300
61/61 [==============================] - 1s 17ms/step - loss: 0.0573 - accuracy: 0.9891 - val_loss: 0.0280 - val_accuracy: 0.9901
Epoch 17/300
61/61 [==============================] - 1s 17ms/step - loss: 0.0503 - accuracy: 0.9886 - val loss: 0.1105 - val accuracy: 0.9480
```

تقييم النموذج

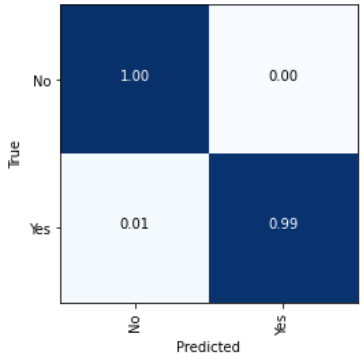
```
model.evaluate(Xval, yval)
```

```
13/13 [==============================] - 0s 7ms/step - loss: 0.0162 -
accuracy: 0.9975
```

مصفوفة الارتباك

```
ypred = (model.predict(Xval)>0.5).astype('int')
```

```
cm = confusion_matrix(yval, ypred)

cm = cm.astype('int') / cm.sum(axis=1)[:, np.newaxis]

fig = plt.figure(figsize = (4, 4))
ax = fig.add_subplot(111)

for i in range(cm.shape[1]):
    for j in range(cm.shape[0]):
        if cm[i,j] > 0.8:
            clr = "white"
        else:
            clr = "black"
        ax.text(j, i, format(cm[i, j], '.2f'), horizontalalignment="center",
color=clr)

_ = ax.imshow(cm, cmap=plt.cm.Blues)
ax.set_xticks(range(len(class_names)))
ax.set_yticks(range(len(class_names)))
ax.set_xticklabels(class_names, rotation = 90)
ax.set_yticklabels(class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



```
f1_score(yval, ypred, average = 'binary')
```

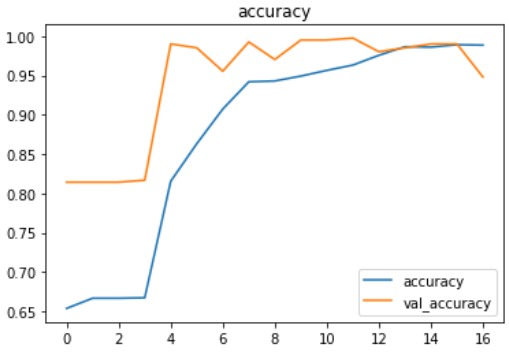
```
0.9932885906040269
```

```
def plot(history, variable1, variable2):
    plt.plot(range(len(history[variable1])), history[variable1])
    plt.plot(range(len(history[variable2])), history[variable2])
    plt.legend([variable1, variable2])
    plt.title(variable1)
```

النتائج

كما ترون، فقد وصل النموذج إلى دقة تزيد عن 90٪ مما يسمح لنا بالتنبؤ بما إذا كانت الأغنام موجودةفي الصورة أم لا.

```
plot(history.history, "accuracy", 'val_accuracy')
```



التنبؤ

دعونا نلقي نظرة على التوقعات

```
x = random.randint(0, 32 - 1) # default batch size is 32

for i in val_ds.as_numpy_iterator():
    img, label = i
    plt.axis('off')   # remove axes
    plt.imshow(img[x])    # shape from (64, 64, 64, 1) --> (64, 64, 1)
    output = model.predict(np.expand_dims(img[x],0))[0][0]    # getting
output; input shape (64, 64, 3) --> (1, 64, 64, 1)
    pred = (output > 0.5).astype('int')
    print("Predicted: ", class_names[pred], '(', output, '-->', pred, ')')
# Picking the label from class_names base don the model output
    print("True: ", class_names[label[x]])
    break
```

```
Predicted:  Yes ( 1.0 --> 1 )
True:  Yes
```

حفظ النموذج

```
model.save("Sheep_detection.h5")
```

رابط الكود الكامل: انقر هنا.

29) بناء بوت الدردشة باستخدام NLTK و Keras Learn to build your first chatbot using NLTK & Keras

مرحبًا Siri، ما معنى الحياة؟

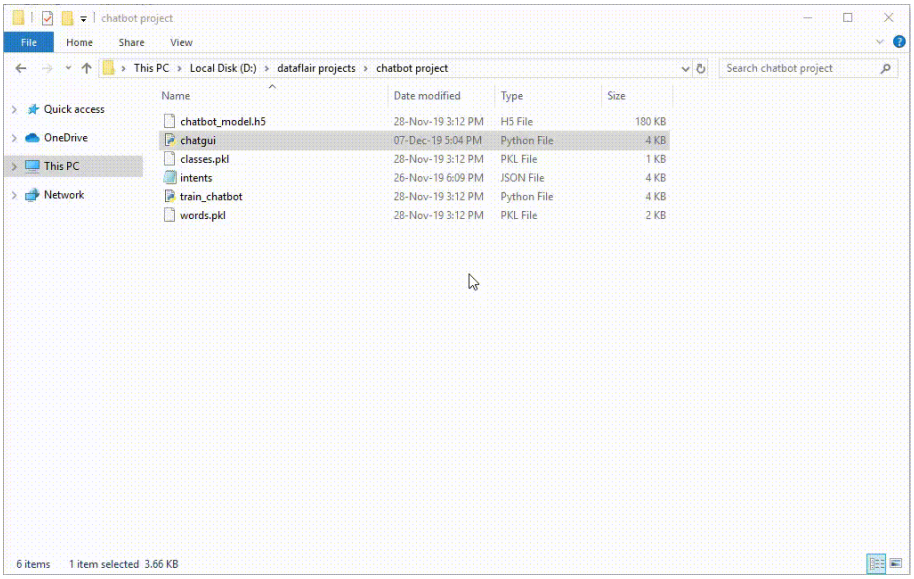
وفقًا لجميع الأدلة، إنها شوكولاتة لك.

بمجرد أن سمعت هذا الرد من Siri، علمت أنني وجدت شريكًا مثاليًا للاستمتاع بساعات العزلة. من الأسئلة الغبية إلى بعض النصائح الجادة، كان Siri دائمًا موجودًا من أجلي.

كم هو مذهل أن تخبر شخصًا بكل شيء وأي شيء دون أن يتم الحكم عليك على الإطلاق. إنه شعور من الدرجة الأولى وهذا هو جمال برنامج الدردشة الآلي chatbot.

ما هو Chatbot؟

بوت الدردشة Chatbot هو برنامج ذكي قادر على التواصل وتنفيذ أفعال مشابهة للإنسان. تُستخدم بوتات الدردشة كثيرًافي تفاعل العملاء والتسويق على مواقع الشبكات الاجتماعية ومراسلة العميل على الفور. هناك نوعان أساسيان من نماذج بوتات الدردشة بناءً على كيفية بنائها؛ النماذج القائمة على الاسترجاع Retrieval والمولدة Generative.



1. بوتات الدردشة القائمة على الاسترجاع

يستخدم بوت الدردشة القائم على الاسترجاع أنماط إدخال واستجابات محددة مسبقًا. ثم يستخدم نوعًا من النهج الإرشادي لتحديد الاستجابة المناسبة. يتم استخدامه على نطاق واسعفي

الصناعة لإنشاء بوتات دردشة موجهة نحو الهدف حيث يمكننا تخصيص نغمة وتدفق chatbot لدفع عملائنا بأفضل تجربة.

2. بوتات الدردشة القائمة على التوليد

لا تعتمد النماذج التوليدية على بعض الاستجابات المحددة مسبقًا.

وهي تستند إلى الشبكات العصبية seq2seq. إنها نفس فكرة الترجمة الآلية.في الترجمة الآلية، نترجم الكود المصدري من لغة إلى لغة أخرى، ولكن هنا سنحول المدخلات إلى مخرجات. يحتاج إلى كمية كبيرة من البيانات ويعتمد على الشبكات العصبية العميقة.

حول مشروع Chatbot

في مشروع بايثون هذا مع الكود المصدري، سنقوم ببناء بوت محادثة باستخدام تقنيات التعلم العميق. سيتم تدريب بوت الدردشة على مجموعة البيانات التي تحتوي على فئات (نوايا intents) ونمط pattern وردود responses. نستخدم شبكة عصبية خاصة متكررة (LSTM) لتصنيف الفئة التي تنتمي إليها رسالة المستخدم، ثم سنقدم استجابة عشوائية من قائمة الردود.

لنقم بإنشاء بوت دردشة قائم على الاسترجاع باستخدام NLTK و Keras وبايثون وما إلى ذلك.

تنزيل مجموعة بيانات وكود Chatbot

مجموعة البيانات التي سنستخدمها هي "intents.json". هذا ملف JSON يحتوي على الأنماط التي نحتاج إلى العثور عليها والردود التي نريد إرجاعها إلى المستخدم.

يرجى تنزيل كود ومجموعة بيانات بايثون chatbot من الرابط التالي:

Python Chatbot Code & Dataset

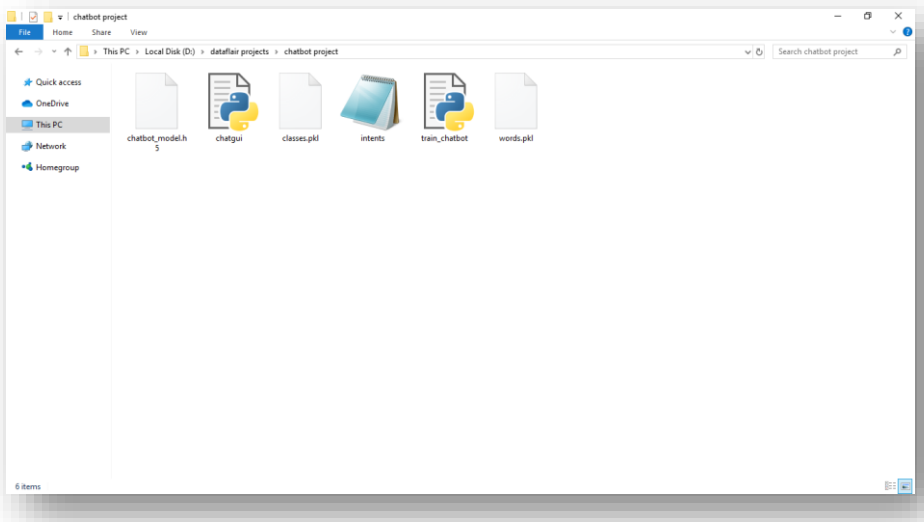
المتطلبات الأساسية

يتطلب المشروع أن تكون لديك معرفة جيدة ببايثون وKeras ومعالجة اللغة الطبيعية (NLTK). إلى جانبهم، سنستخدم بعض وحدات المساعدة التي يمكنك تنزيلها باستخدام الأمر python- pip.

```
pip install tensorflow, keras, pickle, nltk
```

كيفية بناء Chatbot في بايثون؟

سنقوم الآن ببناء chatbot باستخدام بايثون ولكن أولاً، دعونا نرى بُنية الملف ونوع الملفات التي سننشئها:



- **Intents.json:** ملف البيانات الذي يحتوي على أنماط واستجابات محددة مسبقًا.
- **train_chatbot.py**: في ملف بايثون هذا، كتبنا سكريبتاً لبناء النموذج وتدريب بوت الدردشة الخاص بنا.
- **Words.pkl** : هذا ملف pickle نخزن فيه الكلمات كائن بايثون الذي يحتوي على قائمة بمفرداتنا.
- **Classes.pkl** : يحتوي فئات ملف pickle على قائمة الفئات.
- **Chatbot_model.h5** : هذا هو النموذج المدرب الذي يحتوي على معلومات حول النموذج وله أوزان الخلايا العصبية.
- **Chatgui.py**: هذا هو سكريبت بايثون الذي قمنا فيه بتنفيذ واجهة المستخدم الرسومية لروبوت الدردشة الخاص بنا. يمكن للمستخدمين التفاعل بسهولة مع بوت المحادثة.

فيما يلي الخطوات الخمس لإنشاء روبوت محادثةفي بايثون من البداية:

1. استيراد وتحميل ملف البيانات.
2. المعالجة المسبقة للبيانات .
3. إنشاء بيانات التدريب والاختبار.
4. بناء النموذج.
5. توقع الرد.

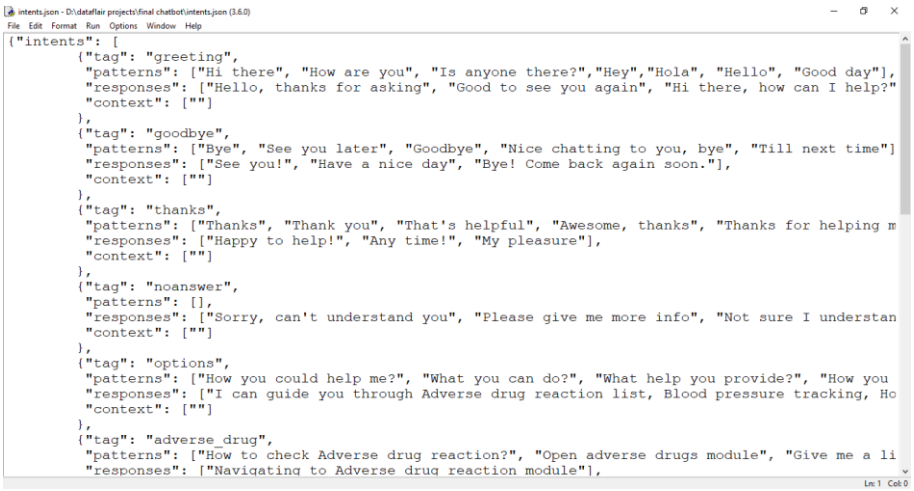
الخطوة 1: استيراد وتحميل ملف البيانات

أولاً، قم بإنشاء اسم ملف كـ train_chatbot.py. نقوم باستيراد الحزم الضرورية لبوت الدردشة الخاص بنا ونقوم بتهيئة المتغيرات التي سنستخدمهافي مشروع بايثون الخاص بنا.

ملف البيانات بتنسيق JSON لذلك استخدمنا حزمة json لتحليل ملف JSON إلى بايثون.

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random

words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('intents.json').read()
intents = json.loads(data_file)
```

هكذا يبدو ملف intents.json الخاص بنا.



الخطوة 2: المعالجة المسبقة للبيانات

عند العمل باستخدام البيانات النصية، نحتاج إلى إجراء معالجة مسبقة مختلفة على البيانات قبل أن نجعل التعلم الآلي أو نموذج التعلم العميق. بناءً على المتطلبات، نحتاج إلى تطبيق عمليات مختلفة لمعالجة البيانات مسبقًا.

الترميز Tokenizing هو أبسط وأول شيء يمكنك القيام به على البيانات النصية. الترميز هو عملية تقسيم النص بأكمله إلى أجزاء صغيرة مثل الكلمات.

نحن هنا نكرر الأنماط ونرمز الجملة باستخدام دالة nltk.word_tokenize() ونلحق كل كلمةفي قائمة الكلمات. نقوم أيضًا بإنشاء قائمة بالفئات الخاصة بعلاماتنا.

```
for intent in intents['intents']:
    for pattern in intent['patterns']:

        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

الآن سنقوم بـ lemmatize كل كلمة وإزالة الكلمات المكررة من القائمة. Lemmatizing هي عملية تحويل كلمة إلى شكل lemma الخاص بها ثم إنشاء ملف مخلل لتخزين كائنات بايثون التي سنستخدمها أثناء التنبؤ.

```
# lemmatize, lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in
ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)

pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))
```

الخطوة 3: إنشاء بيانات التدريب والاختبار

الآن، سننشئ بيانات التدريب التي سنوفر فيها المدخلات والمخرجات. سيكون المدخلات الخاصة بنا هي النمط والمخرجات ستكون الفئة التي ينتمي إليها نمط الإدخال الخاص بنا. لكن الكمبيوتر لا يفهم النص لذا سنحول النص إلى أرقام.

```
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent
related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in
pattern_words]
```

```
    # create our bag of words array with 1, if word match found in
current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each
pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])
# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")
```

الخطوة 4: بناء النموذج

لدينا بيانات التدريب الخاصة بنا جاهزة، والآن سنبني شبكة عصبية عميقة بها 3 طبقات. نحن نستخدم واجهة برمجة تطبيقات Keras التسلسلية (Keras sequential API) لهذا الغرض. بعد تدريب النموذج لمدة 200 حقبة، حققنا دقة بنسبة 100٪في نموذجنا. دعونا نحفظ النموذج باسم "chatbot_model.h5".

```
# Create model - 3 layers. First layer 128 neurons, second layer 64
neurons and 3rd output layer contains number of neurons
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),),
activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov accelerated
gradient gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd,
metrics=['accuracy'])

#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200,
batch_size=5, verbose=1)
model.save('chatbot_model.h5', hist)

print("model created")
```

الخطوة 5: توقع الرد (واجهة المستخدم الرسومية)

للتنبؤ بالجمل والحصول على رد من المستخدم للسماح لنا بإنشاء ملف جديد "chatapp.py".

سنقوم بتحميل النموذج المدرب ثم نستخدم واجهة مستخدم رسومية تتنبأ بالاستجابة من بوت الدردشة. سيخبرنا النموذج فقط بالفئة التي ينتمي إليها، لذلك سنقوم بتنفيذ بعض الدوال التي ستحدد الفئة ثم نسترجع لنا استجابة عشوائية من قائمة الردود.

مرة أخرى، نقوم باستيراد الحزم الضرورية وتحميل ملفات pickle "Words.pkl" و "classes.pkl" التي أنشأناها عندما قمنا بتدريب نموذجنا:

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from keras.models import load_model
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
```

للتنبؤ بالفئة، سنحتاج إلى تقديم المدخلات بنفس الطريقة التي فعلنا بها أثناء التدريب. لذلك سنقوم بإنشاء بعض الدوال التي ستؤدي معالجة نصية مسبقة ومن ثم التنبؤ بالفئة.

```
def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in
sentence_words]
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that
exists in the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary
position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability":
str(r[1])})
    return return_list
```

بعد توقع الفئة، سنحصل على استجابة عشوائية من قائمة النوايا.

```
def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result

def chatbot_response(text):
    ints = predict_class(text, model)
    res = getResponse(ints, intents)
    return res
```

الآن سنقوم بتطوير واجهة مستخدم رسومية GUI. دعنا نستخدم مكتبة Tkinter التي يتم شحنها مع العديد من المكتبات المفيدة لواجهة المستخدم الرسومية. سنأخذ رسالة الإدخال من المستخدم ثم نستخدم الدوال المساعدة التي أنشأناها للحصول على استجابة من البوت وعرضها على واجهة المستخدم الرسومية. هنا هو كود المصدر الكامل لواجهة المستخدم الرسومية.

```
#Creating GUI with tkinter
import tkinter
from tkinter import *


def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)

base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)

#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50",
font="Arial",)

ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set

#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send",
width="12", height=5,
                    bd=0, bg="#32de97",
activebackground="#3c9d9b",fg='#ffffff',
                    command= send )
```

```
#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5",
font="Arial")
#EntryBox.bind("<Return>", send)


#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)

base.mainloop()
```

الخطوة 6: تشغيل chatbot

لتشغيل بوت المحادثة، لدينا ملفان رئيسيان؛ train_chatbot.py و chatapp.py.

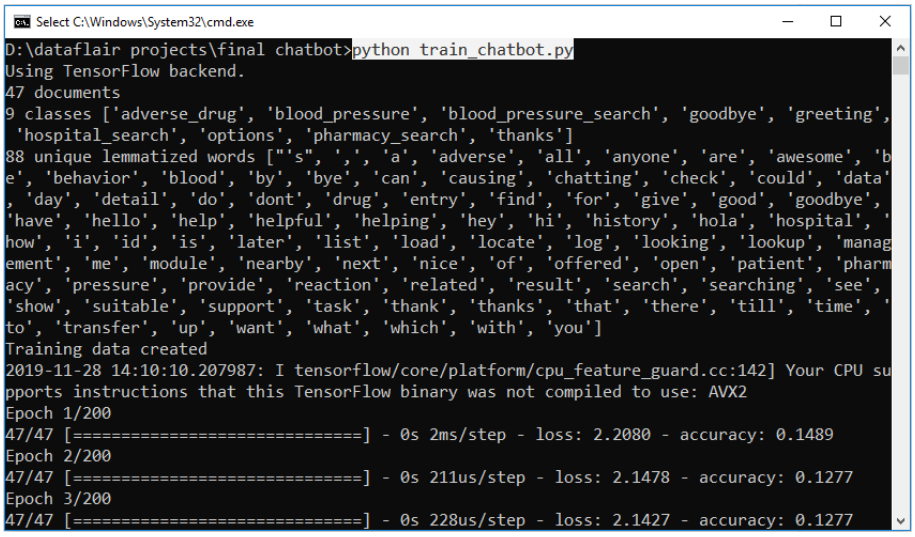
أولاً، نقوم بتدريب النموذج باستخدام الأمرفي التيرمينال:

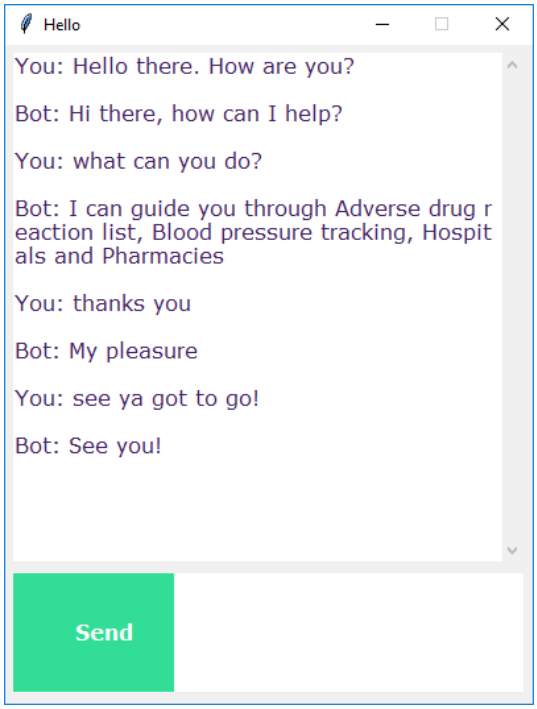
```
python train_chatbot.py
```

إذا لم نشاهد أي خطأ أثناء التدريب، فقد نجحنافي إنشاء النموذج. ثم لتشغيل التطبيق، نقوم بتشغيل الملف الثاني.

```
python chatgui.py
```

سيفتح البرنامج نافذة واجهة المستخدم الرسوميةفي غضون ثوان قليلة. باستخدام واجهة المستخدم الرسومية، يمكنك بسهولة الدردشة مع البوت.





الملخص

في مشروع علم بيانات بايثون هذا، فهمنا عن بوتات الدردشة وقمنا بتطبيق نسخة تعليمية عميقة من chatbotفي بايثون وهي دقيقة. يمكنك تخصيص البيانات وفقًا لمتطلبات العمل وتدريب بوت الدردشة بدقة كبيرة. تُستخدم بوتات الدردشةفي كل مكان وتتطلع جميع الشركات إلى تطبيق البوتاتفي سير عملها.

30) تصنيف خلايا الدم باستخدام التعلم العميق Blood Cell classification using Deep Learning

مقدمة

الدم Blood هو سائل الجسم المنتشر باستمرار والذي يقوم بتوصيل المغذيات والأكسجين إلى الخلايا ويساعدفي نقل المنتجات الثانوية الأيضية بعيداً عن الخلايا. وهو من أهم مكونات جسم الإنسان، وتعتمد الوظائف المتعددة لأعضاء الجسم على الدم السليم. يمكن تقييم صحة الدم من خلال تحليل صحة مكونات الدم المختلفة.

يتكون دم الإنسان من خلايا دم معلقةفي جزء سائل يعرف بالبلازما. تشكل خلايا الدم حوالي 45٪ من حجم الدم، بينما تشكل البلازما 55٪ المتبقية. تتكون خلايا الدم من ثلاثة أنواع تشمل خلايا الدم الحمراء وخلايا الدم البيضاء والصفائح الدموية.

غالبًا ما يتضمن تشخيص أمراض الدم تحديد وتوصيف عينات دم المريض. ستسهل الطرق الآلية للكشف عن الأنواع الفرعية لخلايا الدم وتصنيفهافي التشخيص الأسرع وتحقيق نتائج أفضل للمرضى.

في هذه المقالة سيتمحور تركيزنا الرئيسي حول تصنيف الأنواع المختلفة من خلايا الدم البيضاء باستخدام تقنيات التعلم العميق على منصة Cainvas.

مجموعة البيانات

يمكن جلب مجموعة البيانات المستخدمةفي هذه المقالة من هنا.

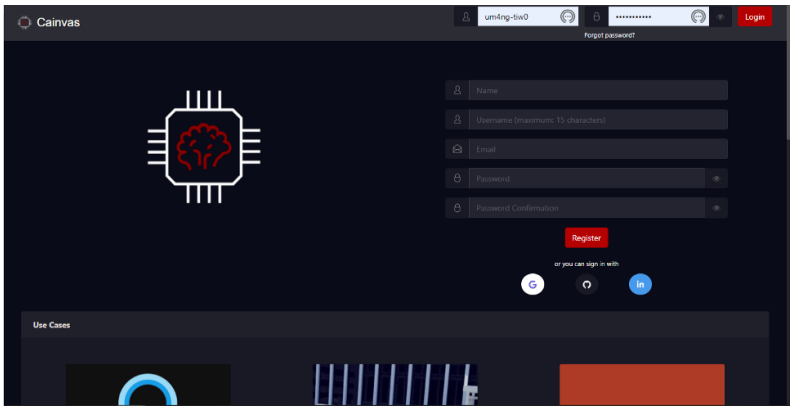
تحتوي مجموعة البيانات على 12500 صورة ملونة مكثفة لخلايا الدم بتنسيق JPEG. هناك 4 فئات مختلفة من خلايا الدم البيضاء ـ الحامضية Eosinophil، واللمفية Lymphocyte، والوحيدة Monocyte، والمتعادلة Neutrophilفي مجموعة البيانات.

في هذه المقالة، سنستخدم Tensorflow ـ مكتبة برامج مفتوحة المصدر توفر الأدوات والموارد لإنشاء خوارزميات التعلم الآلي، و Keras ـ واجهة لمكتبة Tensorflow لتطوير نماذج التعلم العميق، لإنشاء شبكة عصبية تلافيفية CNN ومحاولة التنبؤ بدقة فئات خلايا الدم البيضاء من صور عينات الدم.

ستتم كتابة الكود بالكامل على خادم Notebook لمنصة Cainvas للحصول على أداء أفضل بالإضافة إلى توسيع نطاق النموذج لاحقًا لاستخدامهفي أجهزة EDGE.

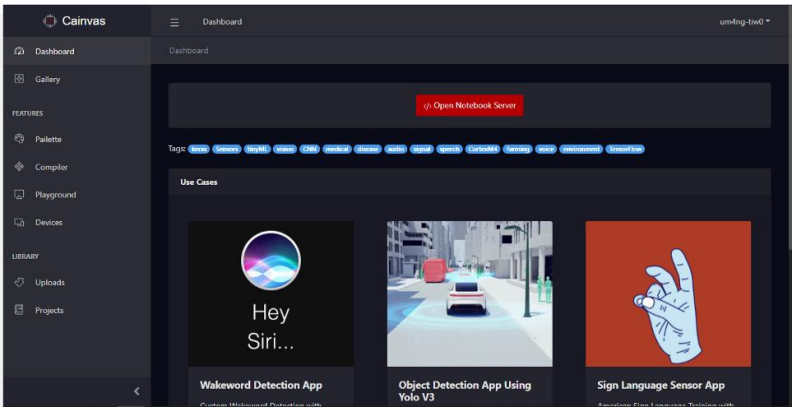
إنشاء المنصة

يمكنك إنشاء حساب على موقع Cainvas هنا.

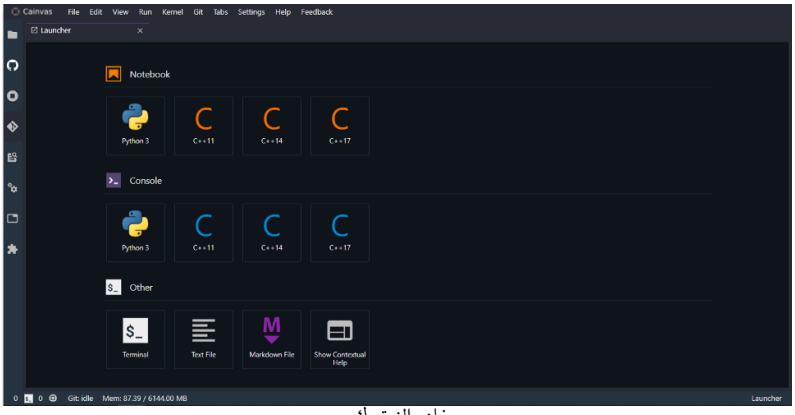


Cainvas landing page

بعد إنشاء حساب بنجاح، قم بتسجيل الدخول إلى النظام الأساسي وانتقل إلى قسم Dashboard لفتح خادم Notebook.



dashboard of the platform



خادم النوتبوك

استيراد المكتبات اللازمة

سنستخدم بعض المكتبات الشائعة الاستخدام مثل Numpy وMatplotlib. سنستخدم OpenCV2 وMatplotlib للوصول إلى الصور وعرضهافي notebook.

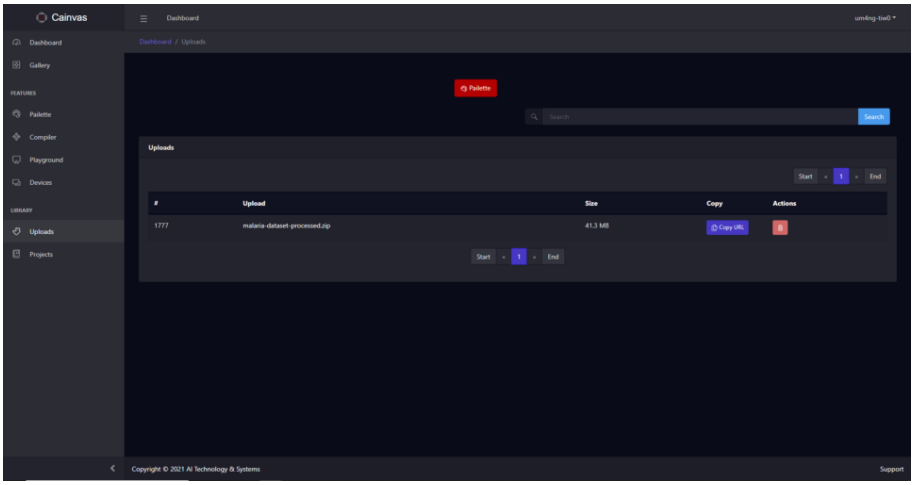
تشمل المكتبات الأخرى Tensorflow و Keras لإنشاء الشبكة العصبية التلافيفية CNN وإجراء المعالجة المسبقة للبيانات لأداء التدريب عليها.

```
#Importing necessary libraries
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import ZeroPadding2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

تحميل مجموعة البيانات

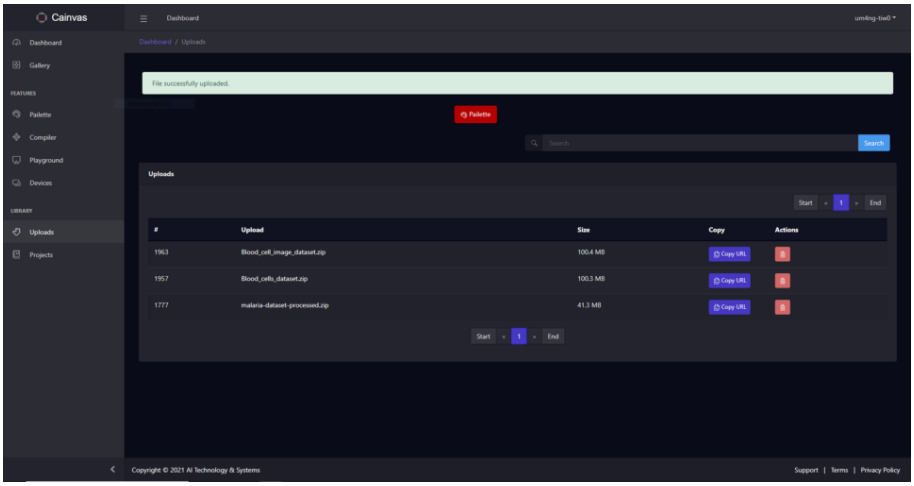
تتيح لنا منصة Cainvas تحميل مجموعات البيانات على المنصة مما يسهل الاستخدام. يمكن بعد ذلك تحميل مجموعات البيانات هذه بسهولة على notebook واستخدامها بمرونة كافية لإنشاء النموذج دون أي متاعب.

لتحميل مجموعة البيانات الخاصة بك، يمكنك التوجه إلى قسم Pailette الذي يسمح بتحميل الملفات والصور ومقاطع الفيديو وحتى بيانات المستشعر.



ميزة التحميل في منصة Cainvas

سنقوم بتحميل مجموعة البيانات كملف مضغوطفي هذه المقالة. يمكن الحصول على عنوان URL للملف الذي تم تحميله بعد التحميل واستخدامهفي notebook لجلبه. لعرض الملفات المرفوعة، ما عليك سوى النقر فوق أقسام التحميلات. انقر فوق الزر نسخ URL لنسخ عنوان URL الخاص بالملف.



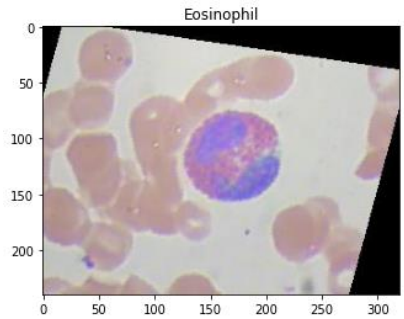
قسم التحميلات مع البيانات التي تم تحميلها

يمكننا استخدام عنوان URL مع الأمرwget! لتحميلهفي دفتر ملاحظاتنا. يمكننا بعد ذلك فك ضغط الملف المضغوطفي الوضع الصامت باستخدام unzip! -qo filename.zip

```
!wget -N "https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-admin/Blood_cell_image_dataset.zip"
!unzip -qo Blood_cell_image_dataset.zip
!rm Blood_cell_image_dataset.zip
```

يمكننا الوصول إلى صورة للتحقق مما إذا تم تحميل مجموعة البيانات بنجاح.

```
img = cv2.imread("Blood_cell_image_dataset/images/TRAIN/EOSINOPHIL/_0_1169.jpeg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB )
plt.title("Eosinophil")
plt.imshow(img)
```



تجهيز البيانات

توجد بيانات التدريب داخل أربع مجلدات ـ Eosinophil، وLymphocyte ، وMonocyte، وNeutrophil، والتي تمثل الفئات الأربع لخلايا الدم البيضاء. سنستخدم ImageDataGenerator الذي توفره Keras لإعداد البيانات والحصول على التسميات المناسبة المتعلقة بهيكل المجلد. يوفر لنا المولد أيضًا المرونةفي إنشاء تقسيمات مجموعات التدريب والتحقق من الصحة من مجموعة بيانات التدريب بأكملها.

```
datagen = ImageDataGenerator(rescale = 1/255.0, validation_split =
0.2)
train_data_generator =
datagen.flow_from_directory(directory="Blood_cell_image_dataset/images
/TRAIN/",
                                                    target_size =
(img width, img height), color mode="rgb",

class_mode="categorical", batch_size = 16, shuffle=True ,subset =
"training")
validation_data_generator =
datagen.flow_from_directory(directory="Blood_cell_image_dataset/images
/TRAIN/",
                                                        target_size =
(img_width, img_height),  color_mode="rgb",

class_mode="categorical", batch_size = 16, shuffle=True, subset =
"validation")
```

يمكننا الآن التحقق من التسميات التي تم جلبها من خلال بُنية مجلد بيانات التدريب الخاصة بنا.

```
train_data_generator.next()[1]
```

```
array([[0., 0., 1., 0.],
       [0., 0., 1., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [0., 0., 1., 0.],
       [0., 1., 0., 0.],
       [1., 0., 0., 0.],
       [0., 0., 1., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]], dtype=float32)
```

نتلقى متجهات ترميز واحد ساخن one hot encoded بسبب الطبيعة الفئوية categorical للبيانات. يشير موضع الفهرس لـ 1s إلى الفئة المقابلة لكرات الدم البيضاءفي الصورة.

إنشاء النموذج

كما ذكرنا، سننشئ شبكة عصبية تلافيفية CNN للتنبؤ بالفئات الصحيحة للخلايا من الصور. لقد استخدمنا 3 طبقات Conv2D مع طبقات MaxPool2D بعد كل منها لاستخراج الميزة من الصور. دالة التنشيط المستخدمة هي ReLU. تحتوي طبقة الإخراج على أربعة خلايا عصبية فقط تتوافق مع الفئات الأربع من خلايا الدم البيضاء، مع دالة تنشيط Softmax.

```
model = Sequential()

model.add(Conv2D(32, (3,3), input_shape=(64,64,3), activation="relu"))
model.add(MaxPool2D(2,2))

model.add(Conv2D(32, (3,3), activation="relu"))
model.add(MaxPool2D(2,2))

model.add(Conv2D(16, (3,3), activation="relu"))
model.add(MaxPool2D(2,2))

model.add(Flatten())

model.add(Dense(128, activation="relu"))

model.add(Dense(4, activation="softmax"))
```

ملخص النموذج model summary للنموذج الذي تم إنشاؤه أعلاه هو كما يلي:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 16)        4624
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 128)               73856
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 516
=================================================================
Total params: 89,140
Trainable params: 89,140
Non-trainable params: 0
```

سنستخدم الإيقاف المبكر Early Stopping حتى يتوقف نموذجنا عن التدريب إذا لم تتغير المعلمة المراقبة بمرور الوقت. هذا سيجعل عملية التدريب أكثر كفاءة.

```
my_callback = [tf.keras.callbacks.EarlyStopping(monitor = 'val_loss', patience
= 5, restore_best_weights = True)]
```

تجميع النموذج والتدريب عليه

سنقوم بتجميع compile النموذج مع آدم كمحسِّن وCategorical Crossentropy كدالة خطأ. سنقوم بتدريب النموذج لمدة 100 حقبة مع callback. سنقوم بتخزين accuracy، وloss، و val_accuracy و val_loss في كل حقبة من التاريخ لرسم البيانات ذات المعنى لاحقًا.

```
history=model.fit(train_data_generator,
steps_per_epoch=len(train_data_generator), epochs=100,
validation_data=validation_data_generator, validation_steps =
len(validation_data_generator), callbacks=my_callback)
```

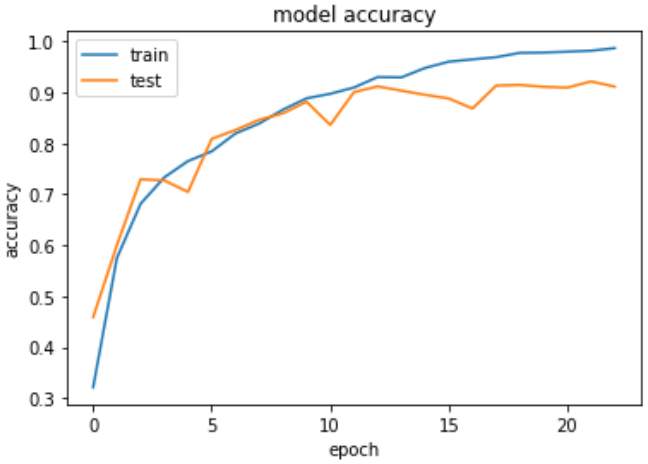
الحقب الثلاثة الأخيرة من مرحلة التدريب:

```
Epoch 21/100
622/622 [==============================] - 13s 21ms/step - loss: 0.1190 -
accuracy: 0.9555 - val_loss: 0.3611 - val_accuracy: 0.8697
Epoch 22/100
622/622 [==============================] - 13s 21ms/step - loss: 0.1205 -
accuracy: 0.9530 - val_loss: 0.3319 - val_accuracy: 0.8777
Epoch 23/100
622/622 [==============================] - 13s 21ms/step - loss: 0.1353 -
accuracy: 0.9481 - val_loss: 0.4199 - val_accuracy: 0.8572
```

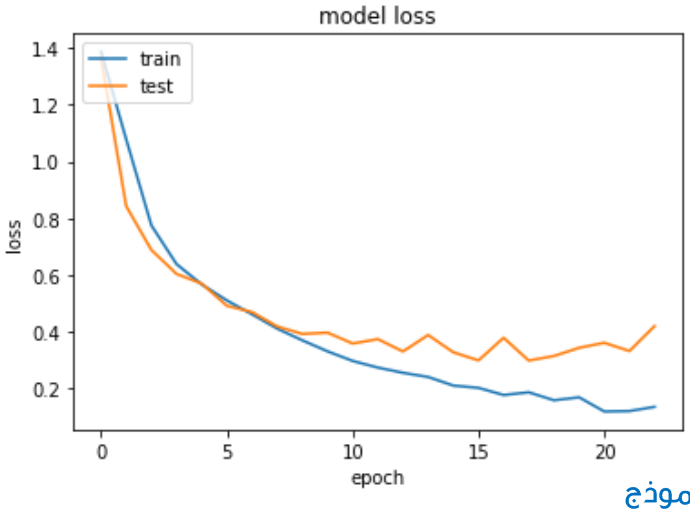
الدقة والخطأ

سنقوم برسم أداء النموذج في كل فترة خلال مرحلة التدريب.

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



اختبار النموذج

سنقوم الآن باختبار النموذج من خلال التقييم على بيانات الاختبار غير المرئية التي تحتوي على 54 صورة موزعةفي 4 فئات.

```
datagen_test = ImageDataGenerator(rescale = 1/255.0)
test_data_generator =
datagen.flow_from_directory(directory="Blood_cell_image_dataset/images
/TEST/",
                                                    target_size =
(img_width, img_height), color_mode="rgb",

class_mode="categorical", batch_size = 16, subset = "training")
model.evaluate(test_data_generator)
```

```
4/4 [==============================] - 0s 18ms/step - loss: 0.1172 -
accuracy: 0.9661
[0.11723806709051132, 0.9661017060279846]
```

سنتوقع الآن الصور العشر الأولىفي بيانات الاختبار. سنقوم أولاً بإنشاء مخرجات أكثر وضوحًا من متجهات ترميز واحد ساخن.

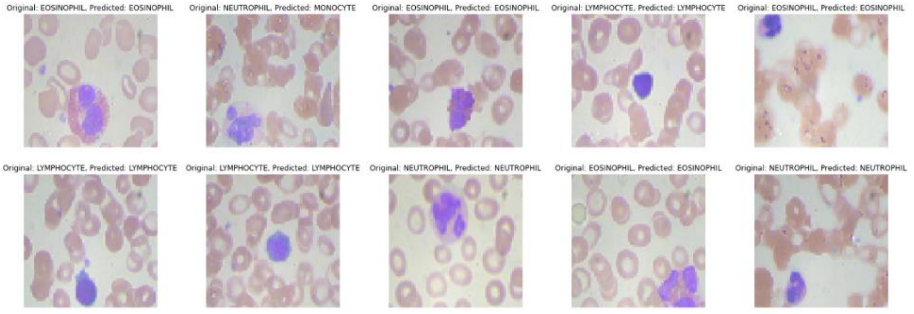
```
# Getting the predicted classes from one hot encoded predicted outputs

x,y = test_data_generator.next()
pred_array=[]
max_index_arr = []
for i in range(10):
    img = x[i]
    img = img.reshape(-1,64,64,3)
    pred_val = model.predict(img)
    max_idx = np.argmax(pred_val)
    pred_array.append(max_idx)
#Making the Output meaningful using named classes
```

```
cell_dict = {0:"EOSINOPHIL", 1:"LYMPHOCYTE", 2:"MONOCYTE", 3:"NEUTROPHIL"}
predictions = {}
actual_val = {}

k=0
for arr in y[:10]:
    actual_val[k] = cell_dict[np.argmax(arr)]
    k+=1

k=0
for pred in pred_array:
    predictions[k] = cell_dict[pred]
    k+=1

print("ACTUAL:", actual_val)
print("PREDICTIONS:", predictions)
```

```
ACTUAL: {0: 'EOSINOPHIL', 1: 'NEUTROPHIL', 2: 'EOSINOPHIL', 3:
'LYMPHOCYTE', 4: 'EOSINOPHIL', 5: 'LYMPHOCYTE', 6: 'LYMPHOCYTE', 7:
'NEUTROPHIL', 8: 'EOSINOPHIL', 9: 'NEUTROPHIL'}PREDICTIONS: {0:
'EOSINOPHIL', 1: 'MONOCYTE', 2: 'EOSINOPHIL', 3: 'LYMPHOCYTE', 4:
'EOSINOPHIL', 5: 'LYMPHOCYTE', 6: 'LYMPHOCYTE', 7: 'NEUTROPHIL', 8:
'EOSINOPHIL', 9: 'NEUTROPHIL'}
```

رسم التنبؤات للحصول على رؤى أفضل:

```
plt.figure(figsize = (20,20))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.imshow(x[i])
    plt.title('Original: {}, Predicted: {}'.format(actual_val[i],
predictions[i]))
    plt.axis('Off')

plt.subplots_adjust(left=1.5, right=2.5, top=1)
plt.show()
```



الملخص:

في هذه المقالة، رأينا كيفية التنبؤ بالفئة الصحيحة لصورة خلايا الدم البيضاء باستخدام شبكة عصبية تلافيفية CNN تم إنشاؤها على منصة Cainvas. لاحظنا قدرات الذكاء الاصطناعي وحالة استخدام بسيطة لكيفية استخدامه لأتمتة أنظمة الرعاية الصحية.

31) كشف السكتة الدماغية باستخدام التعلم العميق Stroke Detection using Deep Learning

تحدث السكتات الدماغية Strokes غالبًا بسبب فقدان أو نقص إمداد الدماغ بالأكسجين. ينتج هذا الفقدفي الإمداد عن فقدان الدم أو تلف الأوعية الدموية.

في هذه المقالة، نحاول استخدام مهارات التعلم الآلي والتعلم العميق لدينا للتنبؤ ببدء السكتة الدماغية بناءً على نمط حياة الشخص. نحن نأخذفي الاعتبار العديد من العوامل ذات الصلة مثل العمر ومؤشر كتلة الجسم (BMI) والحالة الاجتماعية وحالة التدخين وغيرها الكثير. من أجل الوصول إلى مجموعة البيانات، يمكنك اتباع هذا الرابط.

الآن، بعد أن أصبح لدينا البيانات، نحتاج إلى نظام أساسي يمكننا من خلاله أداء تصورنا للبيانات والمعالجة المسبقة وتدريب مصنف الشبكة العصبية الخاص بنا. للقيام بذلك، يمكننا استخدام AITS Cainvas Platform. يتيح لنا هذا الوصول إلى وحدات معالجة الرسومات عالية الكفاءة ويمكننا إعداد jupyter notebooks الخاصة بنا بسهولة.

استيراد المكتبات الضرورية

```
# Import all the necessary libraries

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Model
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import Sequential
from tensorflow.keras import regularizers
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

فك الضغط عن البيانات

```
!wget 'https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-
admin/archive1_PCwQmaT.zip'

!unzip -qo archive1_PCwQmaT.zip
!rm archive1_PCwQmaT.zip
```

سنبدأ بتحميل البيانات والوصول إليها باستخدام مكتبة الباندا. Pandas هي مكتبة توفر هياكل بيانات سهلة الاستخدام لتخزين المعلومات وأداء مهام التصور والمعالجة المسبقة للبيانات بمساعدة المكتبات الأخرى.

```
#Loading the data file using pandas library

data = pd.read_csv('healthcare-dataset-stroke-data.csv', sep = ",")
data.head(3)
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1

بعد ذلك، سنقوم بإزالة عمود المعرف لأننا لن نستخدمه لتدريب البيانات (لأنه ليس له أي صلة على الإطلاقفي تحديد حدوث السكتة الدماغية). بعد التخلص من البيانات غير ذات الصلة، نحتاج إلى التحقق من القيم المفقودة "NULL".

```
data=data.drop(["id"], axis=1)
data.isna().sum()
```

```
gender                0
age                   0
hypertension          0
heart_disease         0
ever_married          0
work_type             0
Residence_type        0
avg_glucose_level     0
bmi                 201
smoking_status        0
stroke                0
dtype: int64
```

تعبئة قيم NAفي مؤشر كتلة الجسم BMI بالقيمة المتوسطة.

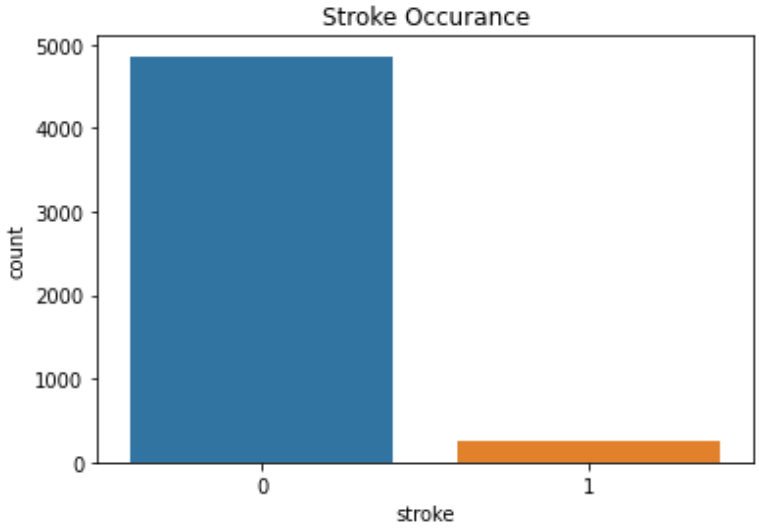
```
data['bmi'] = data['bmi'].fillna(np.mean(data['bmi']))
```

رسم البيانات

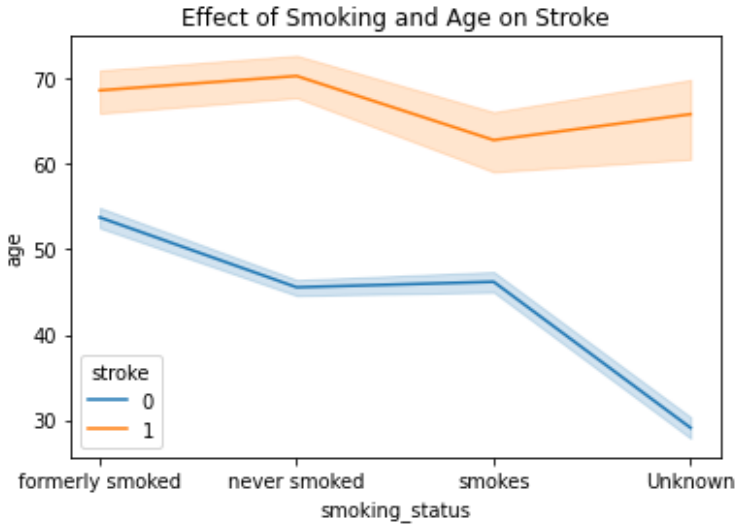
نتعامل مع العديد من المخططات لفهم العلاقة بين البيانات باستخدام مكتبة seaborn وmatplotlib. البدء بـ countplot للتحقق من توزيع بياناتنا. نلاحظ أن بياناتنا غير متوازنة إلى حد كبير مع زيادة تمثيل أحد الفئات عن الآخر. علاوة على ذلك، ندرس تأثير التدخين والعمر على حدوث السكتة الدماغية. من خلال تحليل الرسم البياني، نفهم أن الأفراد المسنين والذين يميلون إلى التدخين لديهم فرصة أكبر للإصابة بسكتة دماغية مقارنة بالشباب غير المدخنين.

بعد ذلك، نحاول فهم تأثير الحالة الاجتماعية للشخص على حدوث السكتة الدماغية.

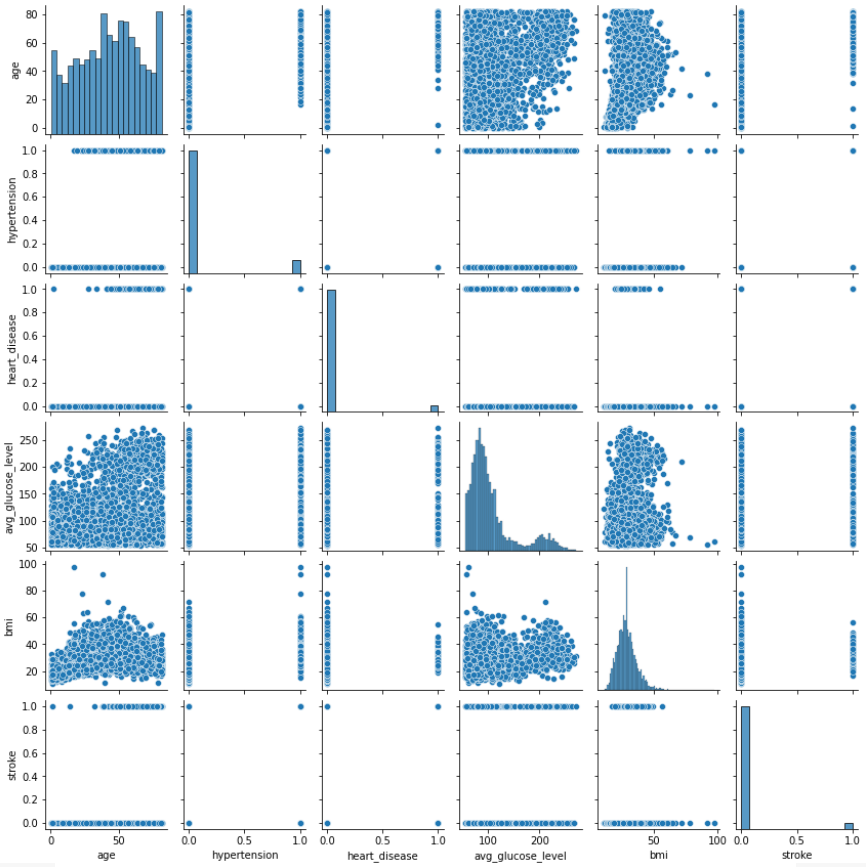
```
sns.countplot(data = data, x = 'stroke')
plt.title("Stroke Occurance")
```



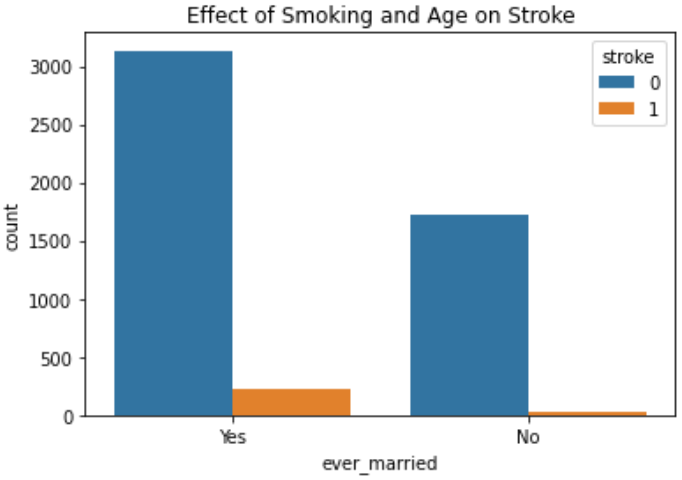

```
sns.lineplot(data = data, x = 'smoking_status', y = 'age', hue = 'stroke')
plt.title("Effect of Smoking and Age on Stroke")
```




```
# Visualising the relationship between different columns of the data
sns.pairplot(data, height = 2)
plt.show()
```



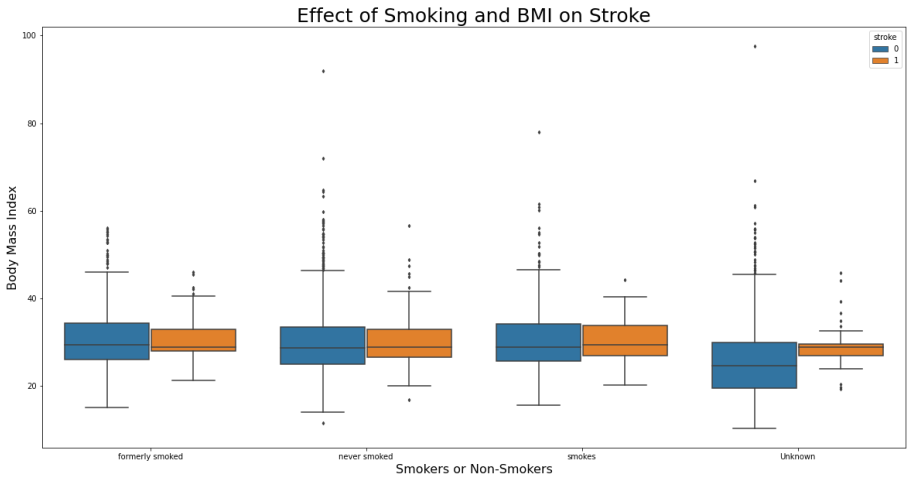

```
# Counting the occurance of stroke in the data
sns.countplot(data = data, x = 'ever_married',  hue = 'stroke')
plt.title("Effect of Smoking and Age on Stroke")
```



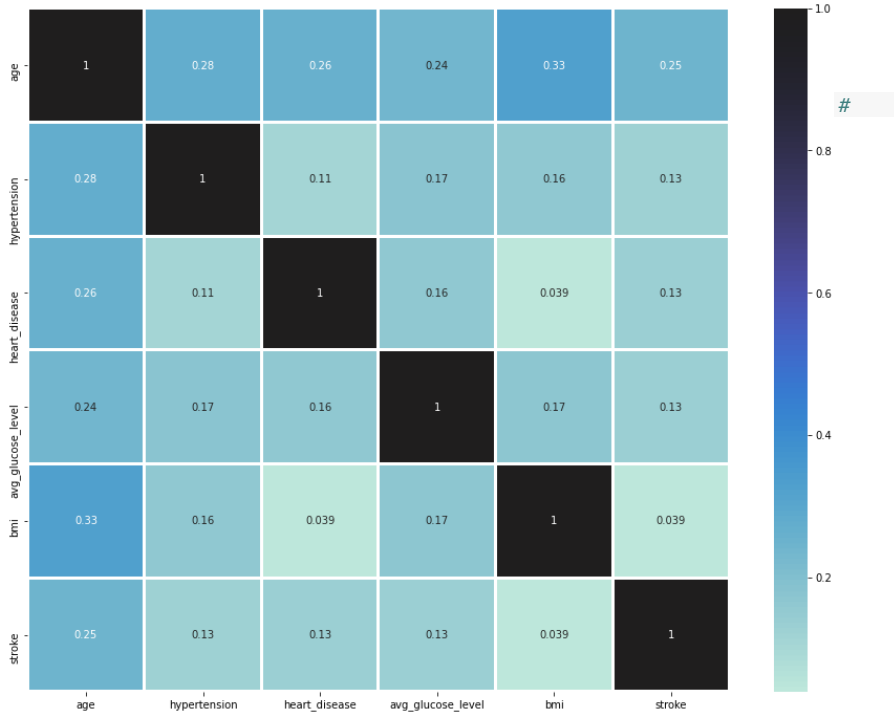
يتبع ذلك مخطط boxplot الذي يساعدنا على فهم تأثير مؤشر كتلة الجسم (BMI) وحالة التدخين على السكتات الدماغية. يعد boxplot طريقة مفيدة للغاية لتوزيع البيانات وفقًا للإجراءات التالية:

1. القيم المتطرفة للبيانات Outliers of the data.
2. الحد الأدنى لقيمة البيانات Minimum value of the data.
3. الربع الأول (الربع الأول ـ 25 النسبة المئوية) First Quartile (Q1—25th Percentile)
4. القيمة المتوسطة Median value.
5. الربع الثالث (الربع الثالث ـ 75 النسبة المئوية) Third Quartile (Q3—75th Percentile)
6. القيمة القصوى للبيانات Maximum value of the data.

```
fig, ax = plt.subplots(figsize = (20,10))
sns.boxplot(data = data, x = 'smoking_status', y = 'bmi', hue = 'stroke',
fliersize = 3)
plt.title("Effect of Smoking and BMI on Stroke", fontdict = {'size' : 25})
plt.xlabel("Smokers or Non-Smokers", fontdict = {'size' : 16})
plt.ylabel("Body Mass Index", fontdict = {'size' : 16})
```




```
# Plotting a heatmap/correlation plot to see how different values are related
to each other
plt.figure(figsize=(15,12))
sns.heatmap(data.corr(),annot=True,linewidths=2, center = True)
plt.show()
```




```
Making sure that no NA values are left in the data
data.isna().sum()
```

```
gender              0
age                 0
hypertension        0
heart_disease       0
ever_married        0
work_type           0
Residence_type      0
avg_glucose_level   0
bmi                 0
smoking_status      0
stroke              0
dtype: int64
```

```
data.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.600000	formerly smoked	1
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.893237	never smoked	1
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	never smoked	1
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	smokes	1
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.000000	never smoked	1

المعالجة المسبقة

بعد رسم البيانات، نحتاج إلى البدءفي المعالجة المسبقة لبياناتنا لتدريب نموذج الشبكة العصبية. نلاحظ أن بياناتنا تحتوي على معلوماتفي شكل فئات نصية. من أجل جعل نموذجنا يفهم البيانات، نحتاج إلى تحويل هذه البيانات إلى تنسيق رقمي. لإنجاز هذه المهمة، نستخدم LabelEncoderفي جميع الأعمدة ذات الصلة.

استخدام Label Encoder لتحويل الفئات المختلفةفي البيانات إلى تنسيق رقمي من أجل تغذية البيانات إلى نموذج DNN.

```
le=LabelEncoder()
data.gender=le.fit_transform(data.gender)
data.ever_married=le.fit_transform(data.ever_married)
data.work_type=le.fit_transform(data.work_type)
data.Residence_type=le.fit_transform(data.Residence_type)
data.smoking_status=le.fit_transform(data.smoking_status)
```

```
data.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	2	1	228.69	36.600000	1	1
1	0	61.0	0	0	1	3	0	202.21	28.893237	2	1
2	1	80.0	0	1	1	2	0	105.92	32.500000	2	1
3	0	49.0	0	0	1	2	1	171.23	34.400000	3	1
4	0	79.0	1	0	1	3	0	174.12	24.000000	2	1

```
print(data.shape)
X = data.iloc[:,:10].values
y = data.iloc[:,-1].values
```

```
(5110, 11)
```

تقسيم مجموعة البيانات إلى تدريب واختبار

الخطوة التالية هي تقسيم البيانات إلى أجزاء تدريب واختبار بتقسيم 40٪ للاختبار و60٪ للتدريب. بعد تقسيم البيانات، قمنا بتوحيدها عن طريق إزالة المتوسط والتحجيم حسب تباين الوحدة باستخدام دالة StandardScalar()في بيانات التدريب.

```
# Splitting our dataset into train-test split
X_train, X_test, Y_train, Y_test = train_test_split(X, y,test_size =
0.4,random_state = 0, stratify = y, shuffle = True)
```

```
#Feature Scaling

sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# # convert the data to categorical labels

from tensorflow.keras.utils import to_categorical
Y_train = to_categorical(Y_train, num_classes=None)
Y_test = to_categorical(Y_test, num_classes=None)
print ("Y = ",Y_train.shape)
print ("X = ",X_train.shape)
```

```
Y =  (3066, 2)
X =  (3066, 10)
```

بناء النموذج

```
es = EarlyStopping(monitor='val_loss', patience=5)
```

```
# Defining the architecture of our deep learning model

model = Sequential()

model.add(Dense(100, activation = "relu", input_dim = 10))
model.add(Dropout(0.3))
model.add(Dense(100, activation = "relu"))
model.add(Dense(50, activation = "relu"))
model.add(Dropout(0.3))
model.add(Dense(40, activation = "relu"))

model.add(Dropout(0.3))
model.add(Dense(2, activation = "softmax"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               1100
_________________________________________________________________
dropout (Dropout)            (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 40)                2040
_________________________________________________________________
dropout_2 (Dropout)          (None, 40)                0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 82
=================================================================
Total params: 18,372
Trainable params: 18,372
Non-trainable params: 0
_________________________________________________________________
```

بعد أن تصبح البيانات جاهزة، نحتاج إلى إعداد النموذج. بدءًا من معمارية النموذج، يتكون نموذج الشبكة العصبية الخاص بنا من 5 طبقات كثيفة و3 طبقات Dropout ذات قيمة إسقاط (drop value) 30٪ لكل منها. قمنا بعرض ملخص النموذج واستنتجنا أن هناك أكثر من 18 ألف معلمة لنموذجنا.

تجميع وتدريب النموذج

```
# Compiling the model
model.compile(optimizer = Adam(lr = 0.1), loss = 'categorical_crossentropy',
metrics = ['accuracy'])
```

بعد تجميع compiling النموذج باستخدام مُحسِّن آدم ومعدل تعلم 0.1، قمنا بتعيين دالة الخطأ على categorical cross entropy، نبدأ تدريبنا.

```
# Run the model for a batch size of 35 for 100 epochs
history = model.fit(X_train,
                    Y_train,
                    validation_data = (X_test, Y_test),
                    batch_size = 35,
                    epochs = 100,
                    validation_steps = 10,
                    callbacks = [es]
                   )
```

```
Epoch 1/100
88/88 [==============================] - 0s 3ms/step - loss: 0.5402 - accuracy: 0.9260 - val_loss: 0.1947 - val_accuracy: 0.9514
Epoch 2/100
88/88 [==============================] - 0s 2ms/step - loss: 0.1959 - accuracy: 0.9514 - val_loss: 0.1949 - val_accuracy: 0.9514
Epoch 3/100
88/88 [==============================] - 0s 2ms/step - loss: 0.2307 - accuracy: 0.9514 - val_loss: 0.1946 - val_accuracy: 0.9514
Epoch 4/100
88/88 [==============================] - 0s 2ms/step - loss: 0.2962 - accuracy: 0.9488 - val_loss: 0.1944 - val_accuracy: 0.9514
Epoch 5/100
88/88 [==============================] - 0s 2ms/step - loss: 0.1951 - accuracy: 0.9514 - val_loss: 0.1945 - val_accuracy: 0.9514
Epoch 6/100
88/88 [==============================] - 0s 2ms/step - loss: 0.2556 - accuracy: 0.9514 - val_loss: 0.1943 - val_accuracy: 0.9514
Epoch 7/100
88/88 [==============================] - 0s 2ms/step - loss: 0.1960 - accuracy: 0.9514 - val_loss: 0.1969 - val_accuracy: 0.9514
Epoch 8/100
88/88 [==============================] - 0s 2ms/step - loss: 0.1967 - accuracy: 0.9514 - val_loss: 0.1965 - val_accuracy: 0.9514
Epoch 9/100
88/88 [==============================] - 0s 2ms/step - loss: 0.1959 - accuracy: 0.9514 - val_loss: 0.1944 - val_accuracy: 0.9514
Epoch 10/100
88/88 [==============================] - 0s 2ms/step - loss: 0.1958 - accuracy: 0.9514 - val_loss: 0.1944 - val_accuracy: 0.9514
Epoch 11/100
88/88 [==============================] - 0s 2ms/step - loss: 0.1978 - accuracy: 0.9514 - val_loss: 0.1990 - val_accuracy: 0.9514
```

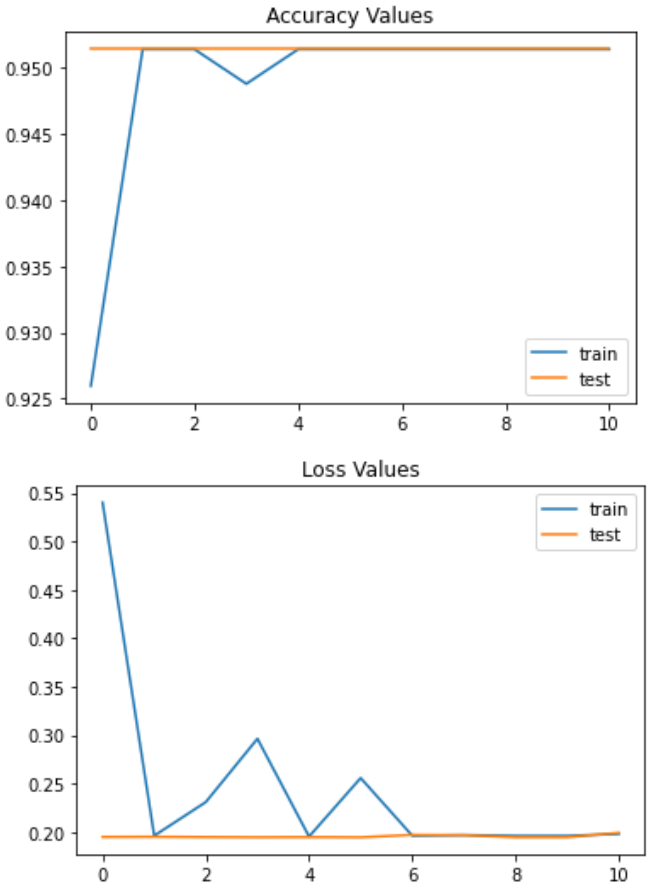
رسم النتائج

من أجل تدريب نموذجنا ومنع أي فرط التعلم overfitting، قمنا بإعداد فحص EarlyStopping الذي يراقب خطأ التحقق من الصحة لدينا. بدأنا تدريب نموذجنا ولمدة 100 حقبة تتوقف بعد حوالي 7ـ8 فترات بسبب دالة الفحص الخاصة بنا على خطأ التحقق من الصحة. نحقق دقة تحقق كبيرة تتجاوز 95٪. لرسم منحنيات التدريب على نموذجنا لمراقبة قيم الدقة والخطأ مع كل فترة، نستخدم الدالة التالية.

```
# Function to plot "accuracy vs epoch" graphs and "Loss vs epoch" graphs for
training and validation data
def plot_metrics(model_name, metric = 'accuracy'):
    if metric == 'loss':
```

```
        plt.title("Loss Values")
        plt.plot(model_name.history['loss'], label = 'train')
        plt.plot(model_name.history['val_loss'], label = 'test')
        plt.legend()
        plt.show()
    else:
        plt.title("Accuracy Values")
        plt.plot(model_name.history['accuracy'], label='train')
        plt.plot(model_name.history['val_accuracy'], label='test')
        plt.legend()
        plt.show()
```

```
plot_metrics(history, 'accuracy')
plot_metrics(history, 'loss')
```



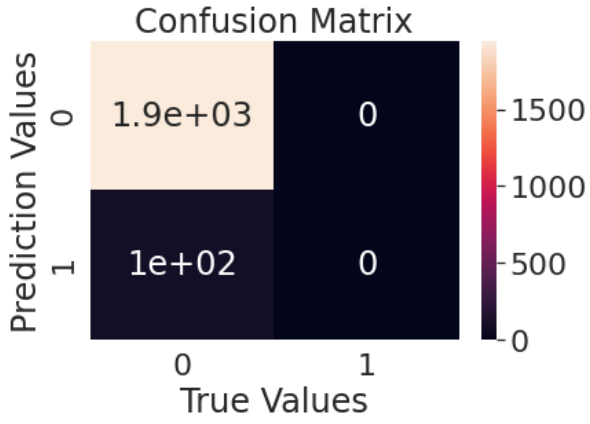
حفظ النموذج

```
# Saving our trained model
from tensorflow.keras.models import save_model
if os.path.isfile('best_model.h5') is False:
    model.save('best_model.h5')
```

مصفوفة الارتباك

تحققنا النهائي هو إجراء بعض التنبؤات حول بيانات الاختبار التي نقوم بتقييمها باستخدام مصفوفة الارتباك confusion matrix. بعد إجراء التنبؤات، يمكننا رسم مصفوفة الارتباك باستخدام الكود التالي:

```
#Plotting a confusion matrix for checking the performance of our model
Y_pred = np.argmax(model.predict(X_test), axis = 1)
cnf = confusion_matrix(Y_test.argmax(axis = 1), Y_pred)


df_cnf = pd.DataFrame(cnf, range(2), range(2))
sns.set(font_scale = 2)
sns.heatmap(df_cnf, annot = True)
plt.title("Confusion Matrix")
plt.xlabel("True Values")
plt.ylabel("Prediction Values")
plt.show()
```



الملخص

نموذجنا هو نجاح يعتمد على البيانات. لقد أكملنا هذا المشروع أخيرًا ويمكننا أن نستنتج أن مزايا التعلم الآلي وصناعة التعلم العميق لا حصر لها. نحن، بصفتنا متعلمين آليًا، يمكننا استخدام مهاراتنا ومعرفتنا بشكل فعال ويمكننا رد الجميل للمجتمع من خلال إحراز تقدم كبيرفي مجالات مثل الرعاية الصحية.

32) كشف ضغط الإطارات باستخدام CNN Tyre Pressure Detection using CNN

يستخدم هذا النموذج صورًا لإطار السيارة للتنبؤ بما إذا كان الإطار ممتلئًا أو مسطحًا أو إذا لم يتم اكتشاف أي إطار.

تحتوي مجموعة البيانات على 3 فئات:

إطار كامل

يحتوي على صور لإطارات السيارات ذات ضغط هواء مناسب وبالتالي تعتبر ممتلئة full. فيما يلي مثال لكيفية ظهور إطار كامل full tyre:

الإطار المثقوب

لا تحتوي هذه الإطارات على ضغط هواء كافٍ لذا فهي مثقوبة flat، أي يجب تعبئتها مرة أخرى وفحصها وإلا فقد تصبح سبب وقوع حادث. فيما يلي مثال على كيف يبدو الإطار المثقوب:

لايوجد إطار

تمثل هذه الصور السيناريو عندما تحتوي الصورة على شيء آخر غير الإطار أو قد لا تحتوي على صورة مناسبة للإطار. فيما يلي مثال على فئة no-tyre:

إليك عرض توضيحي لكيفية بناء هذا النموذج باستخدام Tensorflow و Keras باستخدام بايثون.

استيراد مكتبات

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import tensorflow as tf
import cv2 as cv
from sklearn import preprocessing
from tensorflow import keras
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dropout,
Dense
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras.callbacks import ModelCheckpoint
```

تحميل مجموعة البيانات

```
!wget "https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-
admin/tyre.zip"

!unzip -qo tyre.zip
```

المعالجة المسبقة

```
train_dir= "./tire-dataset/"
```

```
train_datagen= tf.keras.preprocessing.image.ImageDataGenerator(rescale=
1./255, validation_split=0.2)
rain_generator= train_datagen.flow_from_directory(train_dir, target_size=
(100,100), color_mode= 'grayscale', batch_size= 20, class_mode=
'categorical', subset= 'training')
val_generator= train_datagen.flow_from_directory(train_dir, target_size=
(100,100), color_mode= 'grayscale', batch_size= 20, class_mode=
'categorical', subset= 'validation')
```

بناء النموذج

```
model= Sequential([
    layers.Conv2D(32, (3,3), activation= 'relu', input_shape= (100,100,1)),
    layers.MaxPooling2D(pool_size= (2,2), padding= 'same'),
    layers.Dropout(0.3),
    layers.Conv2D(16, (3,3), activation= 'relu'),
    layers.MaxPooling2D(pool_size= (2,2), padding= 'same'),
    layers.Flatten(),
    layers.Dropout(0.3),
    layers.Dense(25, activation= 'relu'),
#     layers.Dense(64, activation= 'relu'),
    layers.Dense(3, activation= 'softmax')
])
model.summary()
```

المخرجات:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 98, 98, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 47, 47, 16)        4624
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 16)        0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 25)                230425
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 78
=================================================================
Total params: 235,447
Trainable params: 235,447
Non-trainable params: 0
_________________________________________________________________
```

تجميع النموذج وتدريبه

```
model.compile(optimizer= 'adam', loss= 'categorical_crossentropy', metrics=
['accuracy'])
```

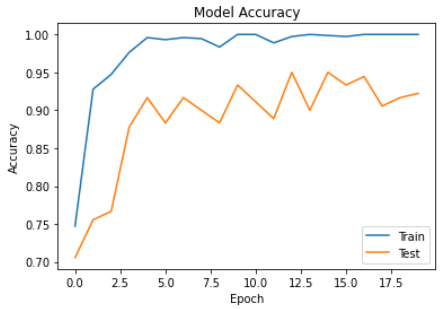
```
history= model.fit_generator(train_generator, epochs= 20, validation_data=
val_generator)
```

المخرجات:

```
Epoch 17/20
36/36 [==============================] - 7s 205ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.2115 - val_accuracy: 0.9444
Epoch 18/20
36/36 [==============================] - 7s 206ms/step - loss: 5.2553e-04 - accuracy: 1.0000 - val_loss: 0.2605 - val_accuracy: 0.9056
Epoch 19/20
36/36 [==============================] - 7s 208ms/step - loss: 3.4928e-04 - accuracy: 1.0000 - val_loss: 0.2436 - val_accuracy: 0.9167
Epoch 20/20
36/36 [==============================] - 7s 205ms/step - loss: 0.0022 - accuracy: 1.0000 - val_loss: 0.2278 - val_accuracy: 0.9222
```

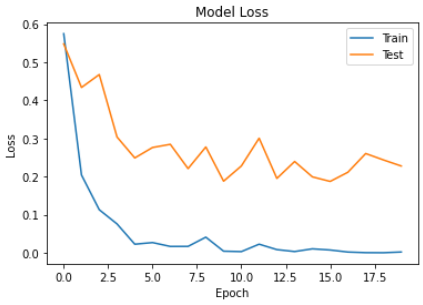
رسم دقة للنموذج

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title("Model Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(["Train", "Test"], loc= "lower right");
```



رسم خطأ للنموذج

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train','Test'], loc= 'upper right');
```

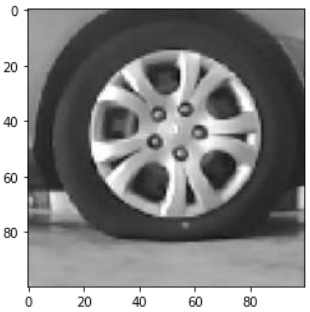


التنبؤ

```
tyre = ["Flat Tyre","Full Tyre","No Tyre"]
def Single_Image_Prediction(file):
    #image = load_img(file, color_mode='rgb', target_size=(128, 128))
    image= file
    plt.imshow(image,cmap= 'gray')
    plt.show()
```

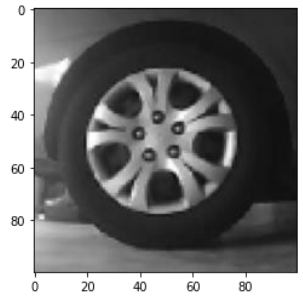
```
    print(image.shape)
#     cv.imshow('image',file)
#     cv.waitKey(0)
#     cv.destroyAllWimdows()
#     image = cv.cvtColor(image, cv.COLOR_RGB2GRAY)
    img_arr = img_to_array(image)
#     img_arr = img_arr/255.
    np_image = np.expand_dims(img_arr, axis=0)
    return np_image
```

```
image = Single_Image_Prediction(val_generator[0][0][11])
pred_value = model.predict(image)
print(pred_value)
index_value = np.argmax(pred_value,axis=1) #For categorical model
print(tyre[index_value[0]])
```




```
(100, 100, 1)
[[1.00000000e+00 5.63426283e-08 1.03149524e-08]]
Flat Tyre
```

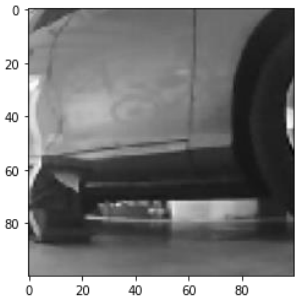
```
image = Single_Image_Prediction(val_generator[0][0][12])
pred_value = model.predict(image)
print(pred_value)
index_value = np.argmax(pred_value,axis=1) #For categorical model
print(tyre[index_value[0]])
```




```
(100, 100, 1)
[[3.6402596e-07 9.9999964e-01 5.2352363e-12]]
Full Tyre
```

```
image = Single_Image_Prediction(val_generator[0][0][0])
pred_value = model.predict(image)
```

```
print(pred_value)
index_value = np.argmax(pred_value,axis=1) #For categorical model
print(tyre[index_value[0]])
```




```
(100, 100, 1)
[[2.9872587e-12 7.5309132e-15 1.0000000e+00]]
No Tyre
```

يمكننا أن نرى أن نموذجنا يعمل بشكل جيد والتي تتحقق التنبؤات أيضًا. الآن سنقوم بحفظ نموذجنا.

حفظ النموذج

```
model.save('tyre_v1.h5')
```

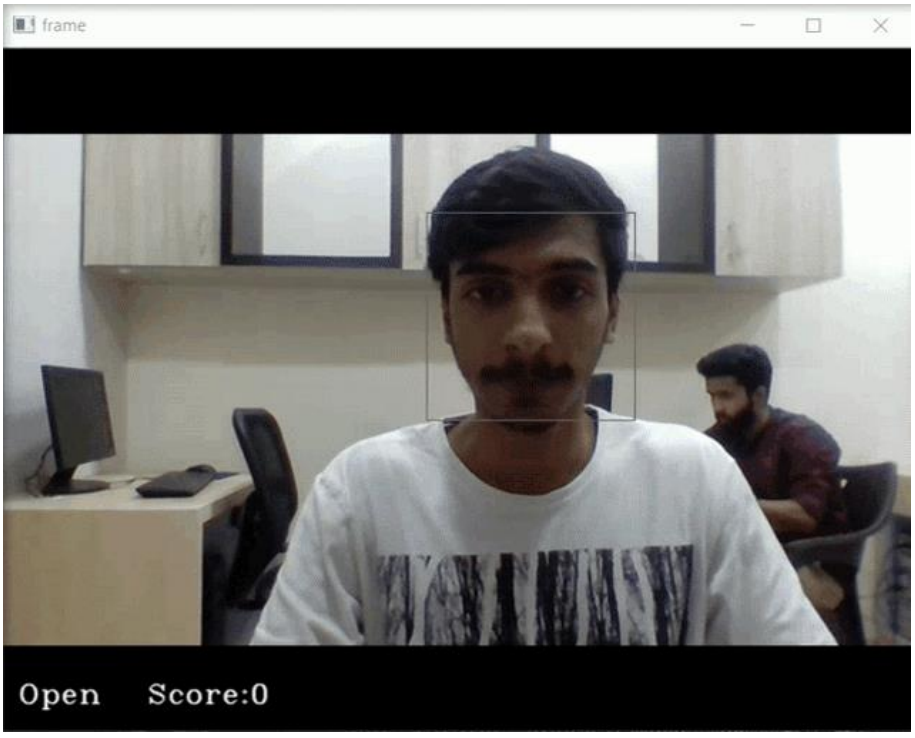
رابط الكود ـ انقر هنا.

33) نظام الكشف عن نعاس السائق مع Driver OpenCV & Keras
Drowsiness Detection System with OpenCV & Keras

مع مشروع بايثون هذا، سنصنع نظامًا لاكتشاف النعاس drowsiness detection system. عدد لا يحصى من الناس يقودون سياراتهم على الطريق السريع ليلًا ونهارًا. سائقي سيارات الأجرة وسائقي الحافلات وسائقي الشاحنات والأشخاص الذين يسافرون لمسافات طويلة يعانون من قلة النوم. بسبب ذلك يصبح من الخطورة جدًا القيادة عند الشعور بالنعاس.

تحدث غالبية الحوادث بسبب نعاس السائق. لذلك، لمنع هذه الحوادث، سنقوم ببناء نظام باستخدام بايثون وOpenCV وKeras الذي سينبه السائق عندما يشعر بالنعاس.

نظام تنبيه السائق النعسان



اكتشاف النعاس Drowsiness detection هو تقنية أمان يمكنها منع الحوادث التي يتسبب فيها السائقون الذين ناموا أثناء القيادة.

الهدف من مشروع بايثون الوسيط هذا هو بناء نظام للكشف عن النعاس يكتشف أن عيون الشخص مغلقة لبضع ثوان. سيقوم هذا النظام بتنبيه السائق عند اكتشاف النعاس.

نظام الكشف عن نعاس السائق

في مشروع بايثون هذا، سنستخدم OpenCV لتجميع الصور من كاميرا الويب وإدخالهافي نموذج التعلم العميق الذي سيصنف ما إذا كانت عيون الشخص "مفتوحة Open" أو "مغلقة Closed". النهج الذي سنستخدمه لمشروع بايثون هذا هو كما يلي:

الخطوة 1: اخذ الصورة كمدخلات من الكاميرا.

الخطوة 2: كشف الوجهفي الصورة وأنشئ منطقة اهتمام Region of Interest (ROI).

الخطوة 3: كشف العيون من ROI وإدخالها إلى المصنف.

الخطوة 4: سيصنف المصنف ما إذا كانت العيون مفتوحة أم مغلقة.

الخطوة 5: حساب النتيجة للتحقق مما إذا كان الشخص يعاني من النعاس.

مجموعة بيانات الكشف عن نعاس السائق

تم إنشاء مجموعة البيانات المستخدمة لهذا النموذج من قبلنا. لإنشاء مجموعة البيانات، كتبنا نصًا يلتقط العيون من الكاميرا ويخزنفي قرصنا المحلي. فصلناها إلى تسمياتها "مفتوحة" أو "مغلقة". تم تنظيف البيانات يدويًا عن طريق إزالة الصور غير المرغوب فيها والتي لم تكن ضرورية لبناء النموذج. تتكون البيانات من حوالي 7000 صورة لعيون الأشخاص تحت ظروف الإضاءة المختلفة. بعد تدريب النموذج على مجموعة البيانات الخاصة بنا، قمنا بإرفاق الأوزان النهائية وملف بنية النموذج."Models / cnnCat2.h5"

الآن، يمكنك استخدام هذا النموذج لتصنيف ما إذا كانت عين الشخص مفتوحة أم مغلقة.

بدلاً من ذلك، إذا كنت ترغبفي إنشاء نموذج خاص بك وتدريبه، يمكنك تنزيل مجموعة البيانات: Driver Drowsiness Dataset

بُنية النموذج

تم تصميم النموذج الذي استخدمناه باستخدام Keras باستخدام الشبكات العصبية التلافيفية (CNN). الشبكة العصبية التلافيفية هي نوع خاص من الشبكات العصبية العميقة التي تؤدي أداءً جيدًا للغاية لأغراض تصنيف الصور. تتكون شبكة CNN بشكل أساسي من طبقة إدخال وطبقة إخراج وطبقة مخفية يمكن أن تحتوي على طبقات متعددة. يتم تنفيذ عملية الالتفاف على هذه الطبقات باستخدام فلتر يقوم بضرب المصفوفة ثنائية الأبعاد على الطبقة والفلتر.

تتكون بُنية نموذج CNN من الطبقات التالية:

- طبقة تلافيفية 32 عقدة ، حجم النواة 3.
- طبقة تلافيفية 32 عقدة ، حجم النواة 3.
- طبقة تلافيفية 64 عقدة ، حجم النواة 3.

- طبقة متصلة بالكامل ؛ 128 عقدة.

الطبقة النهائية هي أيضًا طبقة متصلة بالكامل بها عقدتان. يتم استخدام دالة تنشيط Reluفي جميع الطبقات باستثناء طبقة الإخراج التي استخدمنا فيها Softmax.

متطلبات المشروع

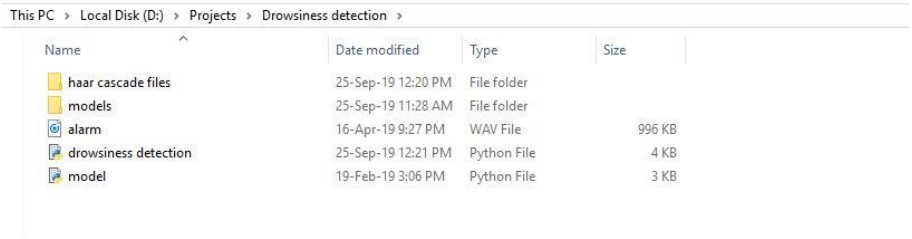
متطلبات مشروع بايثون هذا هو كاميرا ويب سنلتقط الصور من خلالها. تحتاج إلى تثبيت بايثون (الإصدار 3.6 موصى به) على نظامك، ثم باستخدام pip، يمكنك تثبيت الحزم الضرورية.

- **OpenCV**: pip install opencv-python (اكتشاف الوجه والعين).
- **Tensorflow**: pip install tensorflow (يستخدم TensorFlow Keras كخلفية).
- **Keras** : pip install keras (لبناء نموذج التصنيف الخاص بنا).
- **Pygame**: pip install pygame (لتشغيل صوت التنبيه).

خطوات الكشف عن نعاس السائق

قم بتنزيل الكود المصدري لمشروع نظام الكشف عن نعاس السائق من zip واستخرج الملفات في نظامك: Driver Drowsiness Project Code

محتويات zip هي:



- يتكون مجلد "**haar cascade files**" من ملفات xml اللازمة لاكتشاف الكائنات من الصورة. في حالتنا ، نحن نكتشف وجه وعين الشخص.
- يحتوي مجلد النماذج "**models**" على ملف النموذج "cnnCat2.h5" الذي تم تدريبه على الشبكات العصبية التلافيفية.
- لدينا مقطع صوتي "**alarm.wav**" يتم تشغيله عندما يشعر الشخص بالنعاس.
- يحتوي ملف "**Model.py**" على البرنامج الذي قمنا من خلاله ببناء نموذج التصنيف الخاص بنا من خلال التدريب على مجموعة البيانات الخاصة بنا. يمكنك أن ترى تنفيذ الشبكة العصبية التلافيفية في هذا الملف.

- الملف الرئيسي لمشروعنا هو "Drowsiness detection.py". لبدء إجراء الكشف، علينا تشغيل هذا الملف.

دعونا نفهم الآن كيف تعمل الخوارزمية لدينا خطوة بخطوة.

الخطوة 1 - أخذ الصورة كمدخلات من الكاميرا

باستخدام كاميرا الويب، سنلتقط الصور كمدخلات. للوصول إلى كاميرا الويب، قمنا بعمل حلقة لا نهائية من شأنها أن تلتقط كل إطار. نستخدم الطريقة التي يوفرها OpenCV، cv2.VideoCapture(0) للوصول إلى الكاميرا وتعيين كائن الالتقاط (الغطاء) object (cap). سوف يقرأ cap.read() كل إطار ونخزن الصورةفي متغير إطار.

الخطوة 2 - اكتشاف الوجه في الصورة وإنشاء منطقة اهتمام (ROI)

لاكتشاف الوجهفي الصورة، نحتاج أولاً إلى تحويل الصورة إلى درجات رمادية لأن خوارزمية OpenCV لاكتشاف الكائن تأخذ صورًا رماديةفي الإدخال. لا نحتاج إلى معلومات الألوان لاكتشاف الكائنات. سنستخدم مصنف haar cascade لاكتشاف الوجوه. يستخدم هذا السطر face = cv2.CascadeClassifier(' path to our haar cascade xml file') لتعيين وجه المصنف لدينا ثم نقوم بالكشف باستخدام faces = face.detectMultiScale(gray). تقوم بإرجاع مصفوفة من الاكتشافات بإحداثيات x و y والارتفاع وعرض مربع حدود الكائن. الآن يمكننا التكرار فوق الوجوه ورسم مربعات حدود لكل وجه.

```
for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y), (x+w, y+h), (100,100,100), 1 )
```

الخطوة 3 - كشف العيون من ROI وإدخالها إلى المصنف

يتم استخدام نفس الإجراء للكشف عن الوجوه للكشف عن العين. أولاً، قمنا بتعيين المصنف التعاقبي للعيونفي leye و reye على التوالي ثم اكتشفنا العينين باستخدام left_eye = leye.detectMultiScale(gray). نحتاج الآن إلى استخراج بيانات العيون فقط من الصورة الكاملة. يمكن تحقيق ذلك عن طريق استخراج مربع حدود العين ومن ثم يمكننا سحب صورة العين من الإطار باستخدام هذا الكود.

```
l_eye = frame[ y : y+h, x : x+w ]
```

يحتوي l_eye فقط على بيانات صورة العين. سيتم إدخال هذافي مصنف CNN الخاص بنا والذي سيتنبأ بما إذا كانت العيون مفتوحة أم مغلقة. وبالمثل، سنقوم باستخراج العين اليمنىفي r_eye.

الخطوة 4 - المصنف سيصنف ما إذا كانت العيون مفتوحة أم مغلقة

نحن نستخدم مصنف CNN للتنبؤ بحالة العين. لإدخال صورتنافي النموذج، نحتاج إلى إجراء عمليات معينة لأن النموذج يحتاج إلى الأبعاد الصحيحة للبدء بها. أولاً، نقوم بتحويل الصورة الملونة إلى درجات رمادية باستخدام r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY). بعد ذلك، نقوم بتغيير حجم الصورة إلى 24 * 24 بكسل حيث تم تدريب نموذجنا على صور 24 * 24 بكسل ((r_eye = cv2.resize(r_eye, (24,24)). نقوم بتسوية بياناتنا من أجل تقارب أفضل r_eye = r_eye/255 (ستكون جميع القيم بين 0 ـ 1). قم بتوسيع الأبعاد لتغذية المصنف الخاص بنا. قمنا بتحميل نموذجنا باستخدام النموذج model = load_model('models/cnnCat2.h5'). الآن نتوقع كل عين بنموذجنا

lpred = model.predict_classes(l_eye). إذا كانت قيمة lpred [0] = 1 ، فإنها تنص على أن العيون مفتوحة ، وإذا كانت قيمة lpred [0] = 0 ، فإنها تنص على أن العيون مغلقة.

الخطوة 5 - حساب النتيجة للتحقق مما إذا كان الشخص يعاني من النعاس

النتيجة هي في الأساس قيمة سنستخدمها لتحديد المدة التي أغلق فيها الشخص عينيه. لذلك إذا تم إغلاق كلتا العينين، فسوف نستمر في زيادة النتيجة وعندما تكون العيون مفتوحة، فإننا نخفض النتيجة. نقوم برسم النتيجة على الشاشة باستخدام دالة cv2.putText() التي ستعرض حالة الشخص في الوقت الفعلي.

```
cv2.putText(frame, "Open", (10, height-20), font, 1, (255,255,255), 1,
cv2.LINE_AA )
```

يتم تحديد العتبة على سبيل المثال إذا أصبحت النتيجة أكبر من 15، فهذا يعني أن عيني الشخص مغمضتان لفترة طويلة من الزمن. هذا عندما نطلق صفير التنبيه باستخدام sound.play()

يبدو كود المصدر لملفنا الرئيسي كما يلي:

```
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time

mixer.init()
sound = mixer.Sound('alarm.wav')

face = cv2.CascadeClassifier('haar cascade
files\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('haar cascade
files\haarcascade lefteye 2splits.xml')
reye = cv2.CascadeClassifier('haar cascade
files\haarcascade_righteye_2splits.xml')

lbl=['Close','Open']

model = load model('models/cnncat2.h5')
path = os.getcwd()
cap = cv2.VideoCapture(0)
```

```
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
score=0
thicc=2
rpred=[99]
lpred=[99]

while(True):
    ret, frame = cap.read()
    height,width = frame.shape[:2]

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces =
face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,
25))
    left_eye = leye.detectMultiScale(gray)
    right_eye = reye.detectMultiScale(gray)

    cv2.rectangle(frame, (0,height-50) , (200,height) , (0,0,0) ,
thickness=cv2.FILLED )

    for (x,y,w,h) in faces:
cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )

    for (x,y,w,h) in right_eye:
        r_eye=frame[y:y+h,x:x+w]
        count=count+1
        r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
        r_eye = cv2.resize(r_eye,(24,24))
        r_eye= r_eye/255
        r_eye= r_eye.reshape(24,24,-1)
        r_eye = np.expand_dims(r_eye,axis=0)
        rpred = model.predict_classes(r_eye)
        if(rpred[0]==1):
            lbl='Open'
        if(rpred[0]==0):
            lbl='Closed'
        break

    for (x,y,w,h) in left_eye:
        l_eye=frame[y:y+h,x:x+w]
        count=count+1
        l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
        l_eye = cv2.resize(l_eye,(24,24))
        l_eye= l_eye/255
        l_eye=l_eye.reshape(24,24,-1)
        l_eye = np.expand_dims(l_eye,axis=0)
        lpred = model.predict_classes(l_eye)
        if(lpred[0]==1):
            lbl='Open'
        if(lpred[0]==0):
            lbl='Closed'
        break

    if(rpred[0]==0 and lpred[0]==0):
        score=score+1
        cv2.putText(frame,"Closed",(10,height-20), font,
1,(255,255,255),1,cv2.LINE_AA)
    # if(rpred[0]==1 or lpred[0]==1):
    else:
        score=score-1
```

```
        cv2.putText(frame,"Open",(10,height-20), font,
1,(255,255,255),1,cv2.LINE_AA)

    if(score<0):
        score=0
    cv2.putText(frame,'Score:'+str(score),(100,height-20), font,
1,(255,255,255),1,cv2.LINE_AA)
    if(score>15):
        #person is feeling sleepy so we beep the alarm
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
            sound.play()

        except: # isplaying = False
            pass
        if(thicc<16):
            thicc= thicc+2
        else:
            thicc=thicc-2
            if(thicc<2):
                thicc=2
        cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

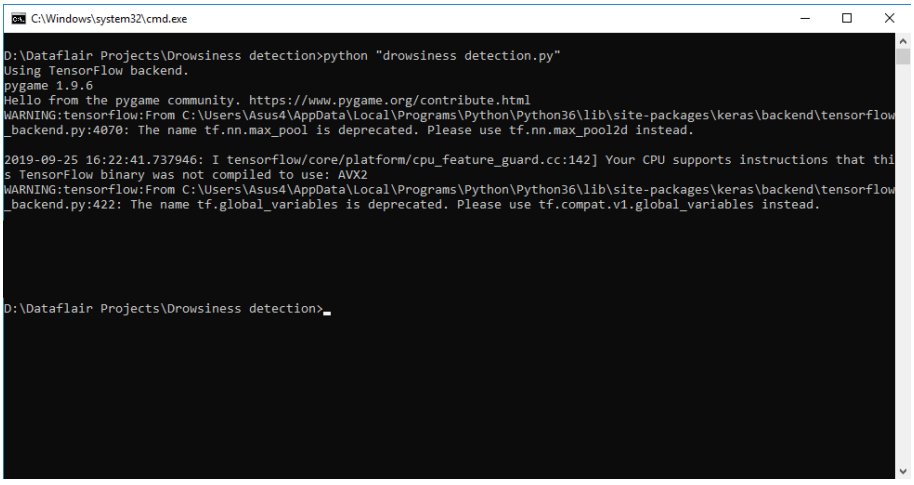
تنفيذ الكشف عن نعاس السائق

دعونا ننفذ نظام الكشف عن النعاسفي محرك الأقراص ونرى عمل مشروع التعلم الآلي الخاص بنا. لبدء المشروع، تحتاج إلى فتح موجه الأوامر، انتقل إلى الدليل حيث يوجد ملفنا الرئيسي "drowsiness detection.py". قم بتشغيل السكريبت باستخدام هذا الأمر.

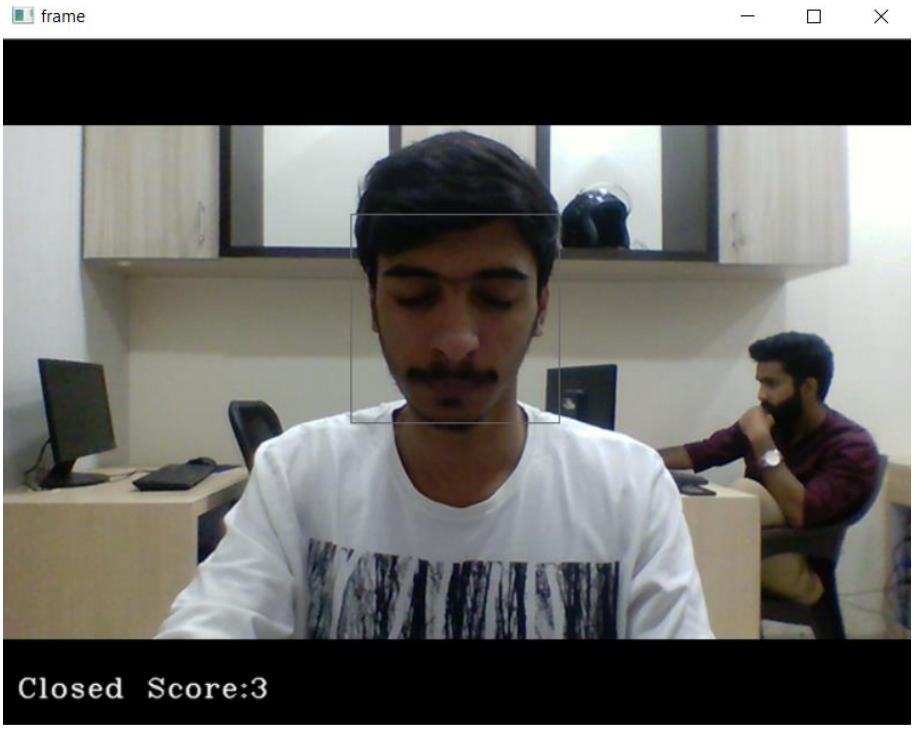
```
python "drowsiness detection.py"
```

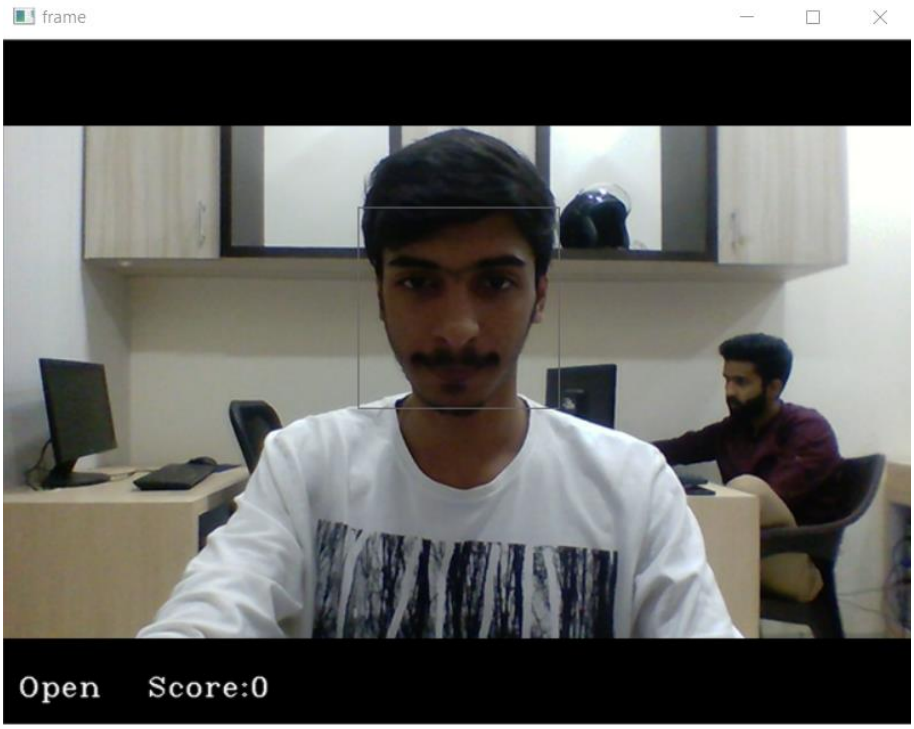
قد يستغرق فتح كاميرا الويب وبدء الاكتشاف بضع ثوانٍ.

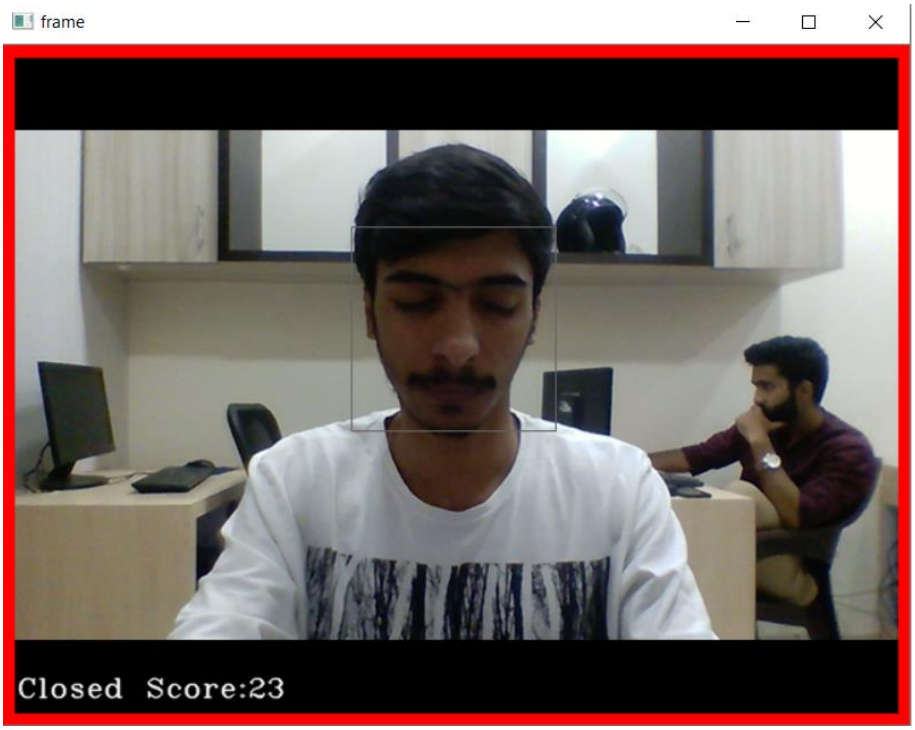
مثال لقطة الشاشة:



لقطة شاشة الإخراج:







الملخص

في مشروع بايثون هذا، قمنا ببناء نظام تنبيه السائق النعاس الذي يمكنك تنفيذه بعدة طرق. استخدمنا OpenCV لاكتشاف الوجوه والعينين باستخدام مصنف haar cascade ثم استخدمنا نموذج CNN للتنبؤ بالحالة.

34) توقع عدم انتظام ضربات القلب على بيانات ECG باستخدام CNN
Arrhythmia prediction on ECG data using CNN CNN

يشير عدم انتظام ضربات القلب Arrhythmia إلى عدم انتظام ضربات القلب أو إيقاعها. وهذا يشمل الضرب بسرعة كبيرة أو بطيئة للغاية أو بإيقاع غير منتظم.

أثبتت نماذج التعلم العميق أنها مفيدة وفعالة للغايةفي المجال الطبي لمعالجة عمليات المسح والأشعة السينية والبيانات الطبية الأخرى لإخراج معلومات مفيدة.

في هذه المقالة، نستخدم التعلم العميق لتصنيف دقات القلب إلى خمس فئات.

رابط التنفيذ على cAInvas – هنا.

استيراد المكتبات الضرورية

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, confusion_matrix
from sklearn.utils import resample
import tensorflow.keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dense, Dropout,
Flatten, MaxPool1D, Convolution1D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import random
```

مجموعة البيانات

مصدر البيانات: Physionet's MIT-BIH Arrhythmia Dataset

تتوافق الإشارات الموجودةفي مجموعة البيانات مع أشكال مخطط كهربية القلب (ECG) لنبضات القلب للحالة الطبيعية والحالات المتأثرة باختلاف ضربات القلب واحتشاء عضلة القلب. تتم معالجة هذه الإشارات وتجزئتها مسبقًا، حيث يتوافق كل جزء مع نبضات القلب.

تحتوي مجموعة البيانات على ملفين CSV، أحدهما يحتوي على عينات للتدريب والآخر للاختبار. يحتوي ملف train.csv على 87554 عينة.

فيما يلي نظرة خاطفة على ملف train.csv:

```
train = pd.read_csv('https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-
admin/mitbih_train.csv',header=None)
test = pd.read_csv('https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-
admin/mitbih_test.csv',header=None)

train
```

	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	18
0	0.977941	0.926471	0.681373	0.245098	0.154412	0.191176	0.151961	0.085784	0.058824	0.049020	...	0.0	0.0	0.0	0.0	0.0	0.
1	0.960114	0.863248	0.461538	0.196581	0.094017	0.125356	0.099715	0.088319	0.074074	0.082621	...	0.0	0.0	0.0	0.0	0.0	0.
2	1.000000	0.659459	0.186486	0.070270	0.070270	0.059459	0.056757	0.043243	0.054054	0.045946	...	0.0	0.0	0.0	0.0	0.0	0.
3	0.925414	0.665746	0.541436	0.276243	0.196133	0.077348	0.071823	0.060773	0.066298	0.058011	...	0.0	0.0	0.0	0.0	0.0	0.
4	0.967136	1.000000	0.830986	0.586854	0.356808	0.248826	0.145540	0.089202	0.117371	0.150235	...	0.0	0.0	0.0	0.0	0.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
87549	0.807018	0.494737	0.536842	0.529825	0.491228	0.484211	0.456140	0.396491	0.284211	0.136842	...	0.0	0.0	0.0	0.0	0.0	0.
87550	0.718333	0.605000	0.486667	0.361667	0.231667	0.120000	0.051667	0.001667	0.000000	0.013333	...	0.0	0.0	0.0	0.0	0.0	0.
87551	0.906122	0.624490	0.595918	0.575510	0.530612	0.481633	0.444898	0.387755	0.322449	0.191837	...	0.0	0.0	0.0	0.0	0.0	0.
87552	0.858228	0.645570	0.845570	0.248101	0.167089	0.131646	0.121519	0.121519	0.118987	0.103797	...	0.0	0.0	0.0	0.0	0.0	0.
87553	0.901506	0.845886	0.800695	0.748552	0.687138	0.599073	0.512167	0.427578	0.395133	0.402086	...	0.0	0.0	0.0	0.0	0.0	0.

87554 rows × 188 columns

يحتوي كل نموذج،في ملف التدريب والاختبار، على 187 ميزة إدخال وعمود واحد يشير إلى تسميات التصنيف.

تنقسم ضربات القلبفي مجموعة البيانات إلى خمس فئات على النحو التالي:

0 ـ ضربات غير منتبذة (إيقاع عادي) Non-ectopic beats (normal beat)

1 ـ ضربات خارج الرحم فوق البطينية Supraventricular ectopic beats

2 ـ ضربات بطينية خارج الرحم Ventricular ectopic beats

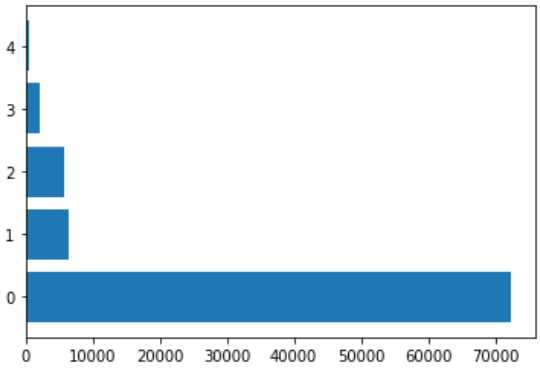
3 ـ دقات الاندماج Fusion beats

4 ـ دقات غير معروفة Unknown beats

دعونا نرى انتشار العينات عبر التسميات:

```
# The classes

label_names = ['Non-ecotic beats (normal beat)', 'Supraventricular ectopic
beats', 'Ventricular ectopic beats', 'Fusion beats', 'Unknown beats']

labels = train[187].astype('int64')   # last column has the labels

print("Count in each label: ")
print(labels.value_counts())

plt.barh(list(set(labels)), list(labels.value_counts()))
```

```
Count in each label:
0    72471
4     6431
2     5788
1     2223
3      641
Name: 187, dtype: int64
```



مجموعة البيانات غير متوازنة (imbalanced) للغاية.

هناك طريقتان لموازنة مجموعة البيانات هذه:

- تحديد العينات من الفئة ذي العدد الأعلى لمطابقة ذلك مع العدد الأقل.
- إعادة أخذ عينات من الفئة بعدد أقل لمطابقة عدد الفصل مع عدد أعلى.

للقيام بذلك، سنقوم أولاً بفصل مجموعة البيانات إلى خمسة، كل منها يحتوي على عينات تنتمي إلى فئة معينة.

ثم يتم إعادة تشكيل كل مجموعة بيانات للحصول على 50000 عينةفي كل مجموعة.

يتم بعد ذلك تجميع مجموعات البيانات الخمس للحصول على مجموعة بيانات متوازنة تضم 250000 عينة إجمالاً.

```
# Separating the train dataframe into 5 individual ones based on class
labels, and sampling 50000 from each.

train_lbl0 = resample(train[train[187]==0], replace=True, n_samples=50000,
random_state=113)
train_lbl1 = resample(train[train[187]==1], replace=True, n_samples=50000,
random_state=113)
train_lbl2 = resample(train[train[187]==2], replace=True, n_samples=50000,
random_state=113)
train_lbl3 = resample(train[train[187]==3], replace=True, n_samples=50000,
random_state=113)
train_lbl4 = resample(train[train[187]==4], replace=True, n_samples=50000,
random_state=113)
```

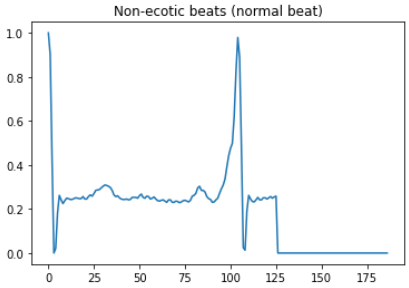
```
# Concatenate the 5 dataframes into 1
train = pd.concat([train_lbl0, train_lbl1, train_lbl2, train_lbl3,
train_lbl4])
labels = train[187].astype('int64')   # last column has the labels
print("Count in each label: ")
print(labels.value_counts())
```

```
Count in each label:
4    50000
3    50000
2    50000
1    50000
0    50000
Name: 187, dtype: int64
```

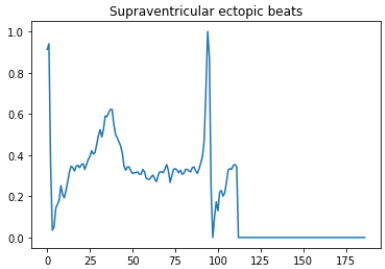
التمثيل المرئي للبيانات

فيما يلي نظرة خاطفة بصرية على فئة مختلفة من دقات القلب:

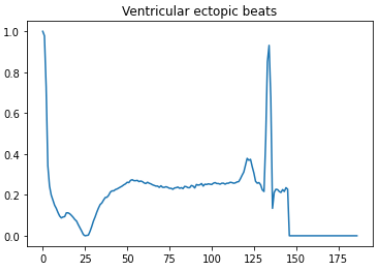
```
plt.plot(np.array(train_lbl0.sample(1))[0, :187])
plt.title(label_names[0])
```



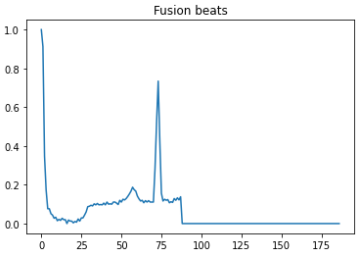

```
plt.plot(np.array(train_lbl1.sample(1))[0, :187])
plt.title(label_names[1])
```



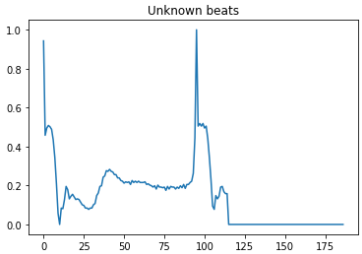

```
plt.plot(np.array(train_lbl2.sample(1))[0, :187])
plt.title(label_names[2])
```




```
plt.plot(np.array(train_lbl3.sample(1))[0, :187])
plt.title(label_names[3])
```




```
plt.plot(np.array(train_lbl4.sample(1))[0, :187])
plt.title(label_names[4])
```



المعالجة المسبقة

اضافة الضوضاء

يتم إضافة الضوضاء (noise) إلى البيانات لتقليد العمليات العشوائية الخارجية التي يمكن أن تتداخل في عملية تسجيل البيانات. تعد الضوضاء الغاوسية البيضاء المضافة (AWGN) نموذجًا مستخدمًا على نطاق واسع لهذا الغرض.

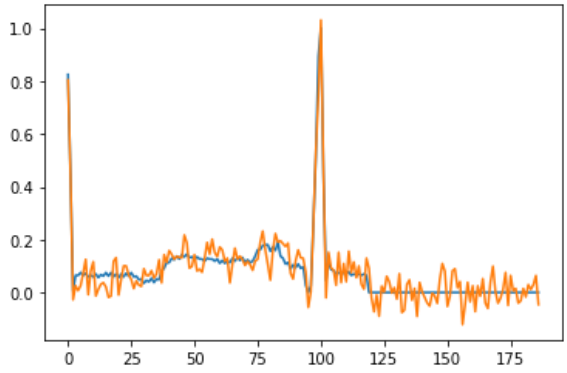
```
# Adding some noise to increase efficiency of the trained model

def gaussian_noise(signal):
    noise = np.random.normal(0,0.05,187)
    return signal + noise
```

تضيف هذه الدالة ضوضاء للإشارة التي تم تمريرها كمعامل. لا تترددفي التلاعب بالمعامل الفائق للانحراف المعياري (هنا، 0.05) ورسم الإشارة بالضوضاء.

يتم أخذ أول 187 عمودًا كإشارة إدخال في كل من مجموعات بيانات التدريب والاختبار.

إليك عينة إشارة مع ضوضاء إضافية:

```
# Visualization with added noise

sample = train_lbl0.sample(1).values[0]

sample_with_noise = gaussian_noise(sample[:187])

plt.subplot(1, 1, 1)

plt.plot(sample[:187])
plt.plot(sample_with_noise)
```



ترميز واحد ساخن

تكون تسميات فئة مجموعة البيانات عبارة عن أعداد صحيحة (0ـ4). نظرًا لأن هذه مشكلة تصنيف، فإن تسميات الفئات هي مشفرة واحد ساخن one hot encoded باستخدام دالة keras.utils.to_categorical.

نموذج لترميز واحد ساخن: قيمة عدد صحيح 1 → [0، 1، 0، 0، 0]

```
# One hot encoding the output of the model

ytrain = tensorflow.keras.utils.to_categorical(train[187])
ytest = tensorflow.keras.utils.to_categorical(test[187])

# Input to the model
xtrain = train.values[:, :187]
xtest = test.values[:, :187]

# Adding noise
for i in range(xtrain.shape[0]):
    xtrain[i, :187] = gaussian_noise(xtrain[i, :187])
```

```
# Viewing the shapes

xtrain = np.expand_dims(xtrain, 2)
xtest = np.expand_dims(xtest, 2)

print("Shape of training data: ")
print("Input: ", xtrain.shape)
print("Output: ", ytrain.shape)

print("\nShape of test data: ")
print("Input: ", xtest.shape)
print("Output: ", ytest.shape)
```

```
Shape of training data:
Input:  (250000, 187, 1)
Output:  (250000, 5)
```

```
Shape of test data:
Input:  (21892, 187, 1)
Output:  (21892, 5)
```

بناء النموذج

يحتوي النموذج على ثلاثة أزواج من طبقات Convolution1D-MaxPool1D متبوعة بطبقة Flatten تقلل القيم إلى 1D. ثم يتبع ذلك 3 طبقات كثيفة، اثنتان منها لها دالة تنشيط ReLU بينما تحتوي الطبقة الأخيرة على 5 عقد، تتوافق مع تسميات فئة الإخراج الخمس، مع دالة تنشيط Softmax.

يتم استخدام دالة تنشيط softmax عندما تكون المخرجات المعطاة للتدريب مشفرة واحد ساخن حيث تحول هذه الدالة متجهًا من قيم n إلى متجه مع قيم n التي تصل إلى 1، وبالتالي تمثل احتمالية كل فئة ممثلة بقيم n.

```
model = Sequential()
model.add(Conv1D(64, 6, activation = 'relu', input_shape = xtrain[0].shape))
model.add(MaxPool1D(3, 2))

model.add(Conv1D(64, 6, activation = 'relu'))
model.add(MaxPool1D(3, 2))

model.add(Conv1D(64, 6, activation = 'relu'))
model.add(MaxPool1D(3, 2))

model.add(Flatten())

model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(5, activation = 'softmax'))

model.compile(optimizer = tensorflow.keras.optimizers.Adam(0.001), loss =
'categorical_crossentropy', metrics = ['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 182, 64)           448
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 90, 64)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 85, 64)            24640
_________________________________________________________________
```

```
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 18, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 64)                73792
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 165
=================================================================
Total params: 125,765
Trainable params: 125,765
Non-trainable params: 0
_________________________________________________________________
```

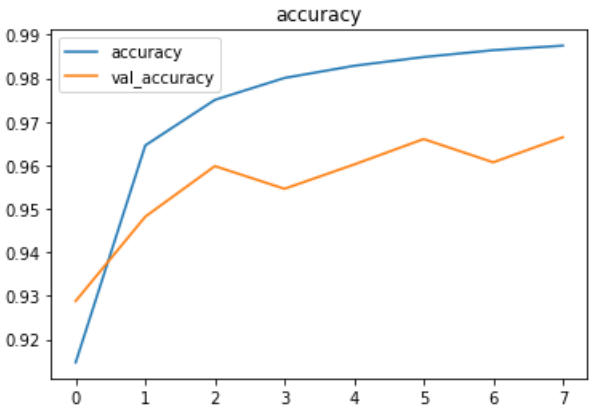
كان النموذج قادرًا على تحقيق دقة تصل إلى 96٪في مجموعة بيانات الاختبار.

ولكن، كما نتذكر، كانت بيانات التدريب فقط متوازنة. هذا يعني أن مجموعة بيانات الاختبار لا تزال غير متوازنة وربما يكون السبب وراء قيمة الدقة العالية.

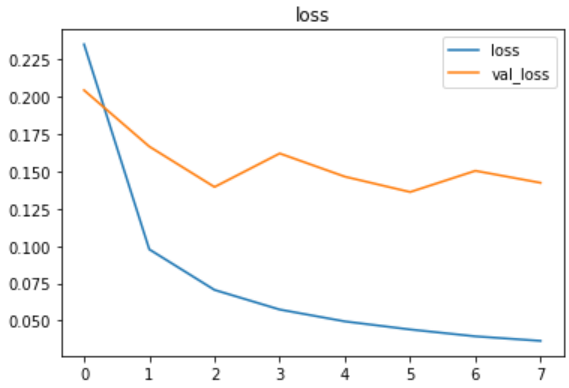
في حالة مجموعات البيانات غير المتوازنة، تعد الدرجة f1 مقياسًا جيدًا لسهولة استخدام النموذج.

رسم المقاييس

```
def plot(history, variable, variable2):
    plt.plot(range(len(history[variable])), history[variable])
    plt.plot(range(len(history[variable2])), history[variable2])
    plt.legend([variable, variable2])
    plt.title(variable)

plot(history.history, "accuracy", "val_accuracy")
```



```
plot(history.history, "loss", "val_loss")
```



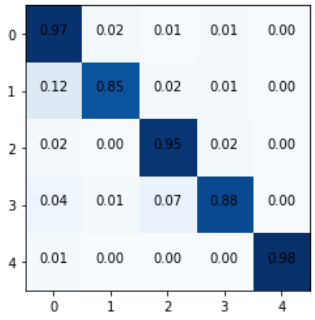
حفظ النموذج

```
model.save('ecg_arryhtmia.h5')
```

تقييم النموذج

دعونا نرسم مصفوفة الارتباك لنرى أداء النموذج على فئات مختلفة.

```
ypred = model.predict(xtest)

cm = confusion_matrix(ytest.argmax(axis=1), ypred.argmax(axis=1))
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

for i in range(cm.shape[1]):
    for j in range(cm.shape[0]):
        plt.text(j, i, format(cm[i, j], '.2f'), horizontalalignment="center",
color="black")


plt.imshow(cm, cmap=plt.cm.Blues)
```



يبدو أن نموذجنا يعمل بشكل جيد حقًا.

```
# Test data class labels spread

print("The distribution of test set labels")
print(test[187].value_counts())

print('F1_score = ', f1_score(ytest.argmax(axis=1), ypred.argmax(axis=1),
average = 'macro'))
```

```
The distribution of test set labels
0.0    18118
4.0     1608
2.0     1448
1.0      556
3.0      162
Name: 187, dtype: int64
F1_score =  0.8411960276398339
```

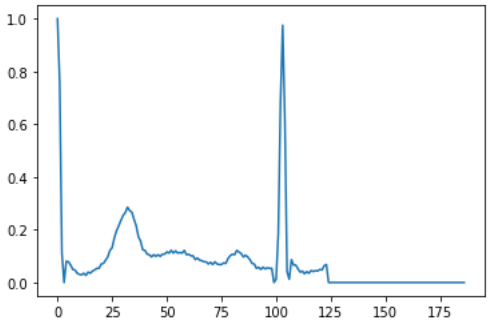
بالنظر إلى أن مجموعة الاختبار الخاصة بنا غير متوازنة، تشير f1-score العالية إلى أن نموذجنا يتمتع بأداء جيد.

التنبؤ

دعونا نرسم بشكل عشوائي بعض إشارات دقات القلب أثناء إجراء التنبؤات باستخدام نموذجنا.

```
i = random.randint(0, len(xtest)-1)

output = model(np.expand_dims(xtest[i], 0))

pred = output.numpy()[0]

plt.plot(xtest[0])

print("Actual label: ", label_names[np.argmax(ytest[i])])
print("Model prediction : ", label_names[np.argmax(pred)], " with probability
", pred[np.argmax(pred)])
```

```
Actual label:  Non-ecotic beats (normal beat)
Model prediction :  Non-ecotic beats (normal beat)  with probability
0.9999658
```



35) اكتشاف أمراض الطماطم باستخدام CNN Tomato Disease Detection with CNN

مقدمة

الهدف من هذا المشروع هو التعرف على الأمراض المختلفة التي تصيب الطماطم بناءً على أوراقها. من المهم جدًافي الزراعة التعرف على الأمراض على الفور. لاكتشاف المشكلةفي الوقت الفعلي، نقوم بتطوير نموذج التعلم العميق الذي يمكن تثبيته على الأجهزة المضمنة ويمكن استخدامهفي البيوت الزجاجية أو عن طريق إضافة النموذج إلى بعض التطبيقات التي يمكن للأشخاص تحميل صور أوراق الطماطم يدويًا والتحقق من مدى صحتها. نموذجنا قادر على إيجاد طماطم صحية و9 أمراض مختلفة. تم استخدام مجموعة البيانات العامة، المتوفرة على Kaggle لتدريب النموذج واختباره.

المعالجة المسبقة

استيراد المكتبات اللازمة

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import cv2
import os
```

تحميل الصور من السيرفر

```
!wget -N "https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-admin/tomato.zip"
!unzip -qo tomato.zip
!rm tomato.zip
```

القراءة والمعالجة المسبقة للصور

يحتوي المجلد الرئيسي لدينا على مجلدين، 'train' لصور التدريب، و 'val' لصور الاختبار. يحتوي كل مجلد من هذه المجلدات على 10 مجلدات بدورها. أحدهما يسمى 'healthy' يحتوي على صور لأوراق الطماطم الصحية و9 مجلدات أخرى تحتوي على صور أوراق طماطم مصابة بأمراض. لدينا 1000 صورةفي كل فئة لغرض التدريب و1000 صورة (100 صورة لكل منها) لاختبار نموذجنا. ننتقل إلى المجلد الرئيسي وسرد مجلدات التدريب ومجلد الاختبار، ثم ندرج أسماء 10 مجلداتفي مجلدات 'train' و 'val' ، ونزيل اسم 'tomato' من هناك ونحصل على تسمية 'healthy' أو اسم المرض المقابل. بعد ذلك، نقرأ الصور من كل مجلد من هذه المجلدات الصحية / الخاصة بالأمراض ونتأكد من أن كل ملف بتنسيق "jpg".في النهاية، نقوم بفصل الصور إلى قنوات r وg وb، ثم نقوم بتطبيعها وإعادة تكديسها معًا.

```
train_images = []
train_labels = []
test_images = []
test_labels = []
```

```
dataset_path = 'tomato'
for train_test_folder in os.listdir(dataset_path):
    # if we are in train folder, we go through disease/healthy folders there
    if train_test_folder == 'train':
        train_path = os.path.join(dataset_path, train_test_folder)
        # for each disease/healthy folder we take folder name as label and go
through it to read images
        for disease_folder in os.listdir(train_path):
            disease_path = os.path.join(train_path, disease_folder)
            label = disease_folder.split('___')[1]
            # in each disease/healthy folder we read files with jpg format,
i.e images and normalize them
            for file in os.listdir(disease_path):
                if file.endswith('jpg'):
                    img_path = os.path.join(disease_path, file)
                    img = cv2.imread(img_path)
                    r, g, b = img[:, :, 0]/255, img[:, :, 1]/255, img[:, :,
2]/255
                    img = np.dstack((r, g, b))
                    train_images.append(img)
                    train_labels.append(label)

    # if we are in val folder, we go through disease/healthy folders there
    if train_test_folder == 'val':
        test_path = os.path.join(dataset_path, train_test_folder)
        # for each disease/healthy folder we take folder name as label and go
through it to read images
        for disease_folder in os.listdir(test_path):
            disease_path = os.path.join(test_path, disease_folder)
            label = disease_folder.split('___')[1]
            # in each disease/healthy folder we read files with jpg format,
i.e images and normalize them
            for file in os.listdir(disease_path):
                if file.endswith('jpg'):
                    img_path = os.path.join(disease_path, file)
                    img = cv2.imread(img_path)
                    r, g, b = img[:, :, 0]/255, img[:, :, 1]/255, img[:, :,
2]/255
                    img = np.dstack((r, g, b))
                    test_images.append(img)
                    test_labels.append(label)

train_images = np.array(train_images)
train_labels = np.array(train_labels)
test_images = np.array(test_images)
test_labels = np.array(test_labels)
print('Shape of the stacked train images:', train_images.shape)
print('Shape of the train labels:', train_labels.shape)
print('Shape of the stacked test images:', test_images.shape)
print('Shape of the test_labels:', test_labels.shape)
```

المخرجات:

```
Shape of the stacked train images: (10000, 64, 64, 3)
Shape of the train labels: (10000,)
Shape of the stacked test images: (1000, 64, 64, 3)
Shape of the test_labels: (1000,)
```

التحقق من جميع فئات الأوراق

دعونا نلقي نظرة على فئاتنا لمعرفة أسماء الأمراض

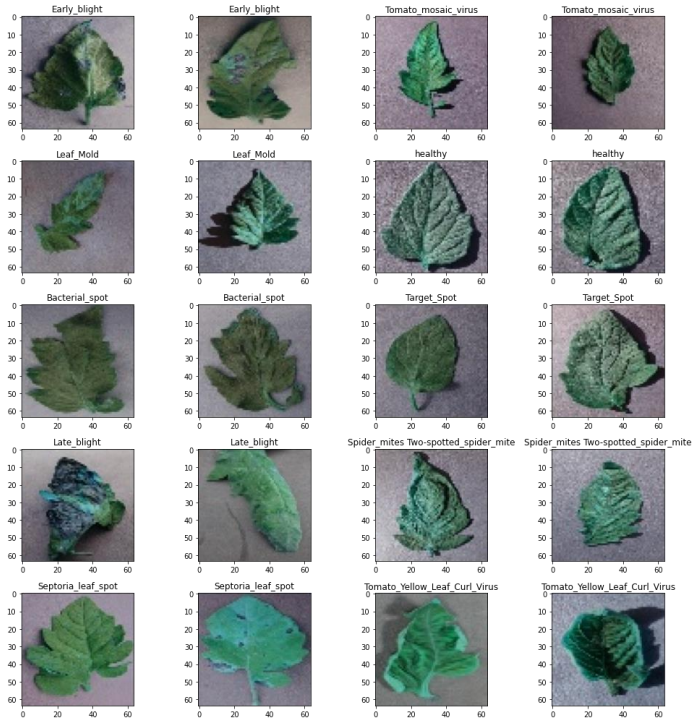
```
unique_labels = np.unique(train_labels)
unique_labels
```

المخرجات:

```
array(['Bacterial_spot', 'Early_blight', 'Late_blight', 'Leaf_Mold',
       'Septoria_leaf_spot', 'Spider_mites Two-spotted_spider_mite',
       'Target_Spot', 'Tomato_Yellow_Leaf_Curl_Virus',
       'Tomato_mosaic_virus', 'healthy'], dtype='<U36')
```

دعونا نرسم صورتين من كل فئة بالتسميات المقابلة لها:

```
row = 5
col = 4
fig, axes = plt.subplots(row, col, figsize=(14, 14))
c = 0
count = 0
for i in range(row):
    for j in range(col):
        axes[i][j].imshow(train_images[c])
        axes[i][j].set_title(train_labels[c])
        c += 500

plt.tight_layout()
plt.show()
```



دوال الترميز وفك الترميز

نظرًا لأن خوارزميات التعلم الآلي لا يمكنها تحديد النصوص، فنحن بحاجة إلى تغذية أهدافنا كبيانات رقمية. نظرًا لأنه ليس لدينا أي أولوية من خلال فئاتنا، فإننا نستخدم طريقة ترميز واحد ساخن One Hot Encoding لترميز تسمياتنا. لهذا الغرض، نقوم بوضع القاموس حيث يكون كل تسمية مفتاحًا وله قيمة عدد صحيح تمثل العمود، ثم نقوم بإنشاء مصفوفة من 10 أعمدة، كل منها يتوافق مع تسمية واحدة ثم ننتقل إلى العينات، ونأخذ قيمة العمود لهذه التسمية من القاموس ونخصص 1 هذا العمود. الدالة الأخرى هي وحدة فك الترميز التي تقوم بالعكس: تأخذ المصفوفة المتوقعة، وتجد العمود ذي الاحتمال الأكبر، وتعيد تسمية النص المقابل لذلك العمود.

```
def encoder(labels):
    train_labels = np.zeros((labels.shape[0], 10))
    dic = {'Bacterial_spot':0, 'Early_blight':1, 'Late_blight':2,
'Leaf_Mold':3, 'Septoria_leaf_spot':4, 'Spider_mites Two-
spotted_spider_mite':5,
           'Target_Spot':6, 'Tomato_Yellow_Leaf_Curl_Virus':7,
'Tomato_mosaic_virus':8, 'healthy':9}
    for i in range(len(labels)):
        train_labels[i, dic[labels[i]]] = 1
    return train_labels
def decoder(labels):
    preds = np.argmax(labels, axis=1)
    test_labels = []
    dic = {0:'Bacterial_spot', 1:'Early_blight', 2:'Late_blight',
3:'Leaf_Mold', 4:'Septoria_leaf_spot', 5:'Spider_mites Two-
spotted_spider_mite',
           6:'Target_Spot', 7:'Tomato_Yellow_Leaf_Curl_Virus',
8:'Tomato_mosaic_virus', 9:'healthy'}

    for i in preds:
        test_labels.append(dic[i])
    return np.array(test_labels)
```

ترميز تسميات التدريب والاختبار الخاصة بنا:

```
train_labels = encoder(train_labels)
test_labels = encoder(test_labels)
```

تقسيم البيانات

```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(train_images, train_labels,
random_state=123)
```

زيادة البيانات

في الصور، توضع الأوراق بشكل أساسي إما أفقيًا أو رأسيًا، لذلك نحتاج إلى مزيد من البيانات مع التقلبات لنموذجنا حتى يتمكن من تحديد العرض الأفقي والرأسي لنفس الورقة. لهذا الغرض،

نستخدم ImageDataGenerator من tensorflow ونغير الوسيطات orizontal_flip وvertical_flip إلى True.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen_train = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)
train_iter = datagen_train.flow(X_train, y_train, batch_size=64)
```

بناء معمارية CNN

نستخدم 3 كتل تحتوي على طبقة الالتفاف وتنشيط ReLU وطبقة Max Pooling لاستخراج الميزة والعنوان الذي يعمل على تسطيح النتائج واستخدام طبقة كثيفة تحتوي على 40 وحدة، واستخدام تنشيط softmax للحصول على النتيجة النهائية.

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten

model = tf.keras.Sequential([
    Conv2D(8, (3, 3), input_shape=(X_train.shape[1], X_train.shape[2],
X_train.shape[3]), activation='relu', padding='same'),
    MaxPooling2D((2, 2), padding='same'),

    Conv2D(16, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2), padding='same'),

    Conv2D(32, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2), padding='same'),

    Flatten(),
    Dense(40, activation='relu'),
    Dense(10, activation='softmax')

])

model.summary()
```

المخرجات:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 64, 64, 8)         224
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 32, 32, 8)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 32, 32, 16)        1168
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 32)        4640
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 40)                81960
_________________________________________________________________
dense_5 (Dense)              (None, 10)                410
=================================================================
Total params: 88,402
Trainable params: 88,402
Non-trainable params: 0
_________________________________________________________________
```

لتدريب نموذجنا، نستخدم مُحسِّن آدم Adam optimizer بمعدل التعلم = 0.0005، categorical_crossentropy كدالة خطأ والدقة كمقياس. بعد تدريب نموذجنا على 120 حقبة، دعنا نلقي نظرة على كيفية تغير الخطأ والدقة عبر الحقب وحفظ نموذجنا. (تم استخدام عمليات callbacks أثناء عملية التدريب لمراقبة خطأ التحقق من الصحة، إذا لم يكن لدينا تحسن لمدة 10 حقب، فإننا نستخدم التوقف المبكر early stopping وإنشاء نقطة فحص check point).

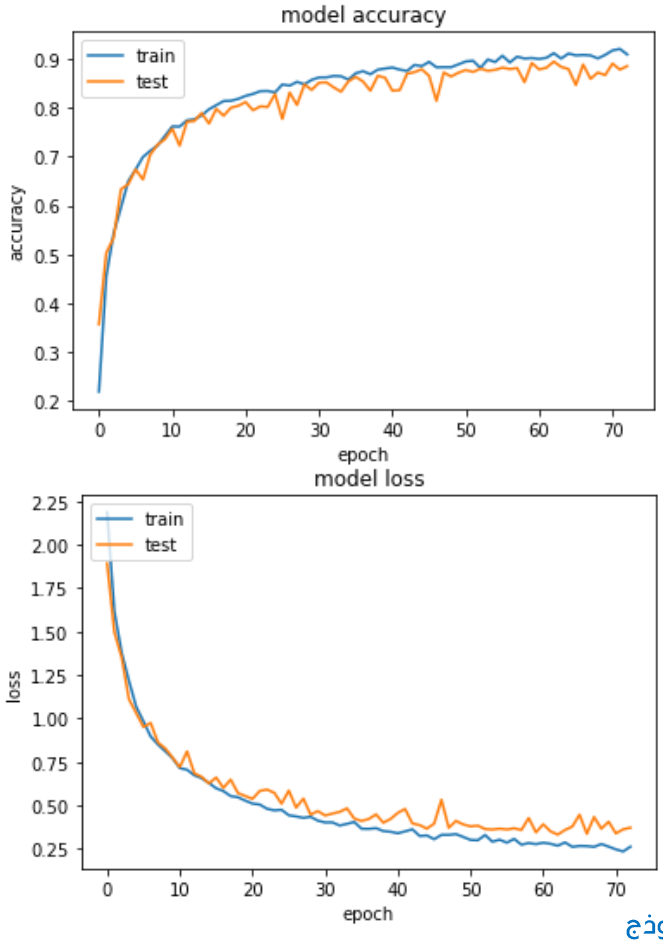
```
# training our model with callbacks: if we have no improvement on validation
loss for 10 epochs, we stop and create a checkpoint
cb = [
        tf.keras.callbacks.EarlyStopping(monitor = 'val_loss', patience = 10,
restore_best_weights = True),
        tf.keras.callbacks.ModelCheckpoint('model_tomato.h5', monitor =
"val_loss", save_best_only = True)
    ]

model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.0005),
loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_iter, steps_per_epoch=len(train_iter), epochs=120,
validation_data=(X_val, y_val), callbacks = cb)
```

```
118/118 [==============================] - 1s 5ms/step - loss: 0.2770 - accuracy: 0.9009 - val_loss: 0.3650 - val_accuracy: 0.8720
Epoch 70/120
118/118 [==============================] - 1s 5ms/step - loss: 0.2619 - accuracy: 0.9081 - val_loss: 0.4054 - val_accuracy: 0.8668
Epoch 71/120
118/118 [==============================] - 1s 5ms/step - loss: 0.2462 - accuracy: 0.9173 - val_loss: 0.3382 - val_accuracy: 0.8904
Epoch 72/120
118/118 [==============================] - 1s 5ms/step - loss: 0.2339 - accuracy: 0.9204 - val_loss: 0.3617 - val_accuracy: 0.8780
Epoch 73/120
118/118 [==============================] - 1s 5ms/step - loss: 0.2617 - accuracy: 0.9091 - val_loss: 0.3710 - val_accuracy: 0.8852
```

رسم الدقة والخطأ وحفظ النموذج

```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

model.save('model_tomato.h5')
print('Weights saved.')
```

المخرجات:



اختبار النموذج

لاختبار نموذجنا، نستخدم 100 صورة لكل فئة. دعونا نحسب الدقة والخطأ على صور الاختبار.

```
acc = model.evaluate(test_images, test_labels)
```

المخرجات:

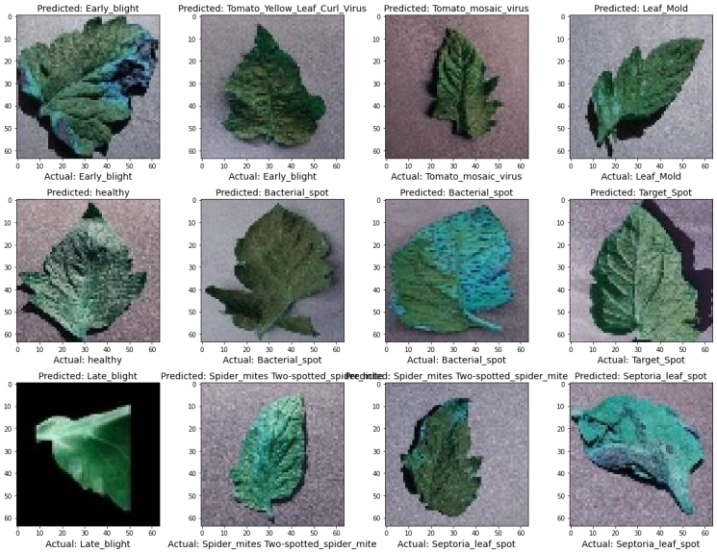
```
32/32 [==============================] - 0s 2ms/step - loss: 0.3077 - accuracy: 0.9130
```

رسم بعض نتائج الاختبار مع تسمياتها الفعلية والمتوقعة:

```
# decoding the labels
predicted_labels = decoder(model.predict(test_images))
test_labels = decoder(test_labels)

# visualizing some of our results
row = 3
col = 4
fig, axes = plt.subplots(row, col, figsize=(16, 12))
c = 0
```

```
count = 0
for i in range(row):
    for j in range(col):
        axes[i][j].imshow(test_images[c])
        axes[i][j].set_title(f'Predicted: {predicted_labels[c]}',
fontsize=14)
        axes[i][j].set_xlabel(f'Actual: {test_labels[c]}', fontsize=14)
        if (predicted_labels[c] != test_labels[c]):
            count+=1
        c += 80

plt.tight_layout()
plt.show()
```

المخرجات:



الملخص

في الختام، قمنا ببناء بنية CNN مع 3 كتل استخلاص مميزة متبوعة برأس للكشف عن 9 أمراض والطماطم الصحية بناءً على أوراقها. نظرًا لزيادة البيانات data augmentation، فإن نموذجنا قادر أيضًا على تحديد الصور المقلوبة ولديه دقة عالية تزيد عن 90٪في بيانات الاختبار.

36) التعرف على إشارات المرور باستخدام التعلم العميق Traffic Signs Recognition

يجب أن تكون قد سمعت عن السيارات ذاتية القيادة التي يمكن للراكب فيها الاعتماد بالكامل على السيارةفي السفر. ولكن لتحقيق المستوى 5 بشكل مستقل، من الضروري أن تفهم المركبات جميع قواعد المرور وتتبعها.

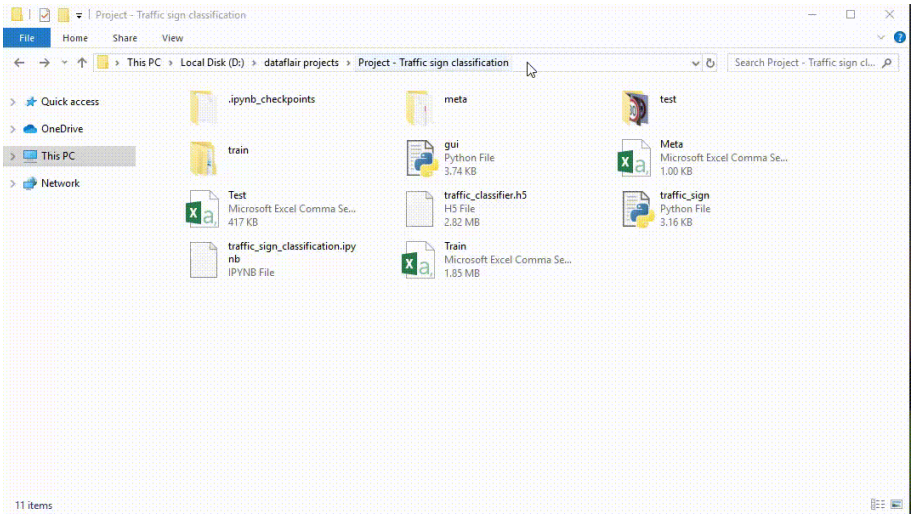
في عالم الذكاء الاصطناعي والتقدمفي التقنيات، يعمل العديد من الباحثين والشركات الكبرى مثل Tesla وUber وGoogle وMercedes-Benz وToyota وFord وAudi وغيرها على المركبات ذاتية القيادة والسيارات ذاتية القيادة. لذلك، لتحقيق الدقةفي هذه التقنية، يجب أن تكون المركبات قادرة على تفسير إشارات المرور واتخاذ القرارات وفقًا لذلك.

ما هو التعرف على إشارات المرور؟

هناك عدة أنواع مختلفة من إشارات المرور مثل حدود السرعة، ممنوع الدخول، انعطف يسارًا أو يمينًا، عبور الأطفال، ممنوع مرور المركبات الثقيلة، إلخ. تصنيف إشارات المرور هو عملية تحديد الفئة التي تنتمي إليها إشارة المرور.

التعرف على إشارات المرور - حول مشروع بايثون

في مثال مشروع بايثون هذا، سنقوم ببناء نموذج شبكة عصبية عميقة يمكنه تصنيف إشارات المرور الموجودةفي الصورة إلى فئات مختلفة. مع هذا النموذج، نحن قادرون على قراءة وفهم إشارات المرور التي تعتبر مهمة للغاية لجميع المركبات ذاتية القيادة.



مشروع مجموعة بيانات بايثون

بالنسبة لهذا المشروع، نستخدم مجموعة البيانات العامة المتاحةفي Kaggle:

Traffic Signs Dataset

تحتوي مجموعة البيانات على أكثر من 50000 صورة لإشارات المرور المختلفة. يتم تصنيفها كذلك إلى 43 فئة مختلفة. تختلف مجموعة البيانات اختلافًا كبيرًا، فبعض الفئات بها العديد من الصور بينما تحتوي بعض الفئات على عدد قليل من الصور. حجم مجموعة البيانات حوالي 300 ميغا بايت. تحتوي مجموعة البيانات على مجلد تدريب train يحتوي على صور داخل كل فئة ومجلد اختبار test سنستخدمه لاختبار نموذجنا.

المتطلبات الأساسية

يتطلب هذا المشروع معرفة مسبقة بـ Keras وMatplotlib وScikit-Learn وPandas وPIL وتصنيف الصور.

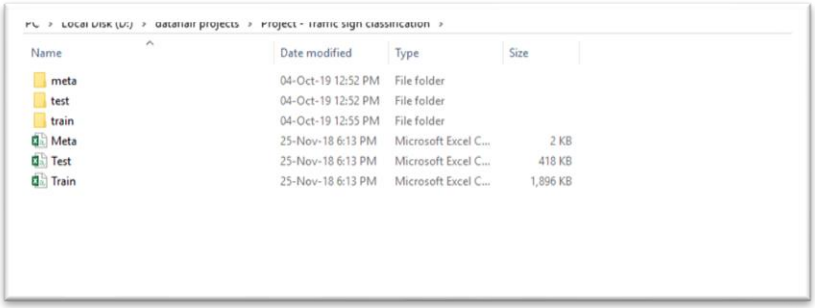
لتثبيت الحزم الضرورية المستخدمةفي مشروع علم بيانات بايثون هذا، أدخل الأمر التاليفي جهازك الطرفي:

```
pip install tensorflow keras sklearn matplotlib pandas pil
```

خطوات بناء مشروع بايثون

لبدء المشروع، قم بتنزيل وفك ضغط الملف من هذا الرابط ـ ملف مضغوط للتعرف على إشارات المرور

واستخرج الملفاتفي مجلد بحيث يكون لديك مجلد تدريب train واختبار test وميتا meta.



قم بإنشاء سكريبت بلغة بايثون وقم بتسميته traffic_signs.py في مجلد المشروع.

تتم مناقشة نهجنا لبناء نموذج تصنيف إشارات المرور هذافي أربع خطوات:

- استكشاف مجموعة البيانات.
- بناء نموذج CNN.
- التدريب والتحقق من صحة النموذج.
- اختبار النموذج باستخدام مجموعة بيانات الاختبار.

الخطوة 1: استكشف مجموعة البيانات

يحتوي مجلد "train" الخاص بنا على 43 مجلداً يمثل كل منها فئة مختلفة. يتراوح نطاق المجلد من 0 إلى 42. بمساعدة وحدة OS، نقوم بالتكرار عبر جميع الفئات وإلحاق الصور والتسميات الخاصة بكل منهافي قائمة البيانات والتسميات.

تُستخدم مكتبة PIL لفتح محتوى الصورةفي مصفوفة.

```
[9]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import tensorflow as tf
     from PIL import Image
     import os
     from sklearn.model_selection import train_test_split
     from keras.utils import to_categorical
     from keras.models import Sequential
     from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

     data = []
     labels = []
     classes = 43
     cur_path = os.getcwd()

     for i in range(classes):
         path = os.path.join(cur_path,'train',str(i))
         images = os.listdir(path)

         for a in images:
             try:
                 image = Image.open(path + '\\'+ a)
                 image = image.resize((30,30))
                 image = np.array(image)
                 #sim = Image.fromarray(image)
                 data.append(image)
                 labels.append(i)
             except:
                 print("Error loading image")
     data = np.array(data)
     labels = np.array(labels)
```

أخيرًا، قمنا بتخزين جميع الصور وتسمياتهافي قوائم (بيانات data وتسميات labels).

نحتاج إلى تحويل القائمة إلى مصفوفات فارغة لتغذية النموذج.

شكل البيانات هو (39209, 30, 30, 3) مما يعني أن هناك 39209 صورة بحجم 30 × 30 بكسل وآخر 3 يعني أن البيانات تحتوي على صور ملونة (قيمة RGB).

باستخدام حزمة sklearn، نستخدم طريقة train_test_split () لتقسيم بيانات التدريب والاختبار.

من حزمة keras.utils ، نستخدم طريقة to_categorical لتحويل التسميات الموجودةفي y_train و t_test إلى ترميز واحد ساخن one-hot encoding.

```
[10]: print(data.shape, labels.shape)
      X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)


      print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

      y_train = to_categorical(y_train, 43)
      y_test = to_categorical(y_test, 43)
```

```
(39209, 30, 30, 3) (39209,)
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

الخطوة 2: بناء نموذج CNN

لتصنيف الصور إلى فئاتها الخاصة، سنقوم ببناء نموذج CNN (الشبكة العصبية التلافيفية). CNN هي الأفضل لأغراض تصنيف الصور.

تصميم نموذجنا هو:

- 2 Conv2D layer (filter=32, kernel_size=(5,5), activation="relu")
- MaxPool2D layer (pool_size=(2,2))
- Dropout layer (rate=0.25)
- 2 Conv2D layer (filter=64, kernel_size=(3,3), activation="relu")
- MaxPool2D layer (pool_size=(2,2))
- Dropout layer (rate=0.25)
- Flatten layer to squeeze the layers into 1 dimension
- Dense Fully connected layer (256 nodes, activation="relu")
- Dropout layer (rate=0.5)
- Dense layer (43 nodes, activation="softmax")

نقوم بتجميع النموذج باستخدام مُحسِّن آدم Adam optimizer الذي يعمل بشكل جيد والخطأ "categorical_crossentropy" لأن لدينا فئات متعددة لتصنيفها.

```
[11]: model = Sequential()
      model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
      model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
      model.add(MaxPool2D(pool_size=(2, 2)))
      model.add(Dropout(rate=0.25))
      model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
      model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
      model.add(MaxPool2D(pool_size=(2, 2)))
      model.add(Dropout(rate=0.25))
      model.add(Flatten())
      model.add(Dense(256, activation='relu'))
      model.add(Dropout(rate=0.5))
      model.add(Dense(43, activation='softmax'))

      #Compilation of the model
      model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

الخطوة 3: تدريب النموذج والتحقق منه

بعد بناء بنية النموذج، نقوم بعد ذلك بتدريب النموذج باستخدام model.fit(). حاولت باستخدام حجم الدفعة (batch siz) 32 و64. كان أداء نموذجنا أفضل مع حجم 64 دفعة. وبعد 15 حقبة epochs كانت الدقة ثابتة.

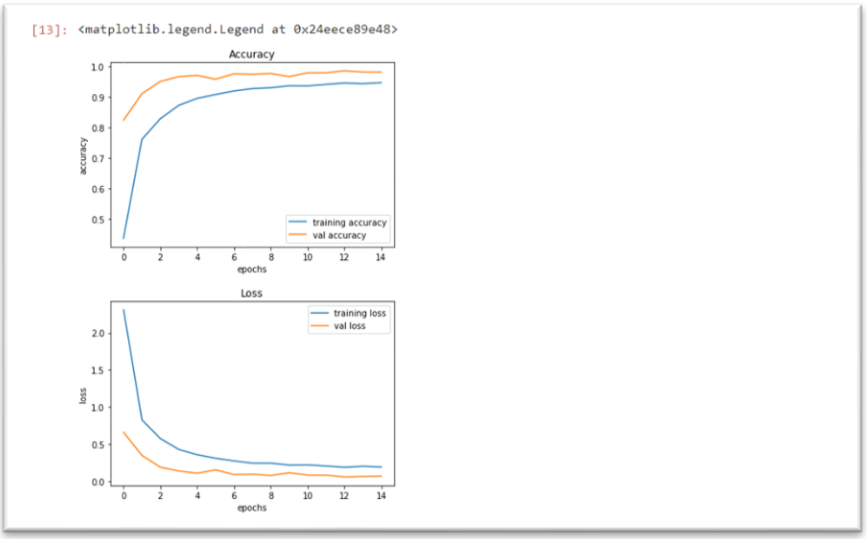
```
[12]: epochs = 15
history = model.fit(X_train, y_train, batch_size=64, epochs=epochs,validation_data=(X_test, y_test))
```

```
Train on 31367 samples, validate on 7842 samples
Epoch 1/15
31367/31367 [==============================] - 82s 3ms/step - loss: 2.3108 - accuracy: 0.4369 - val_lo
ss: 0.6590 - val_accuracy: 0.8234
Epoch 2/15
31367/31367 [==============================] - 82s 3ms/step - loss: 0.8266 - accuracy: 0.7606 - val_lo
ss: 0.3468 - val_accuracy: 0.9100
Epoch 3/15
31367/31367 [==============================] - 83s 3ms/step - loss: 0.5738 - accuracy: 0.8283 - val_lo
ss: 0.1882 - val_accuracy: 0.9504
Epoch 4/15
31367/31367 [==============================] - 85s 3ms/step - loss: 0.4282 - accuracy: 0.8720 - val_lo
ss: 0.1373 - val_accuracy: 0.9661
Epoch 5/15
31367/31367 [==============================] - 84s 3ms/step - loss: 0.3565 - accuracy: 0.8950 - val_lo
ss: 0.1068 - val_accuracy: 0.9702
Epoch 6/15
31367/31367 [==============================] - 81s 3ms/step - loss: 0.3081 - accuracy: 0.9074 - val_lo
ss: 0.1527 - val_accuracy: 0.9575
Epoch 7/15
31367/31367 [==============================] - 81s 3ms/step - loss: 0.2730 - accuracy: 0.9192 - val_lo
ss: 0.0888 - val_accuracy: 0.9753
Epoch 8/15
31367/31367 [==============================] - 81s 3ms/step - loss: 0.2429 - accuracy: 0.9271 - val_lo
ss: 0.0934 - val_accuracy: 0.9737
Epoch 9/15
31367/31367 [==============================] - 84s 3ms/step - loss: 0.2429 - accuracy: 0.9299 - val_lo
ss: 0.0772 - val_accuracy: 0.9763
Epoch 10/15
31367/31367 [==============================] - 81s 3ms/step - loss: 0.2176 - accuracy: 0.9364 - val_lo
ss: 0.1133 - val_accuracy: 0.9663
Epoch 11/15
31367/31367 [==============================] - 82s 3ms/step - loss: 0.2200 - accuracy: 0.9360 - val_lo
ss: 0.0823 - val_accuracy: 0.9786
Epoch 12/15
31367/31367 [==============================] - 80s 3ms/step - loss: 0.2046 - accuracy: 0.9406 - val_lo
ss: 0.0806 - val_accuracy: 0.9787
Epoch 13/15
31367/31367 [==============================] - 80s 3ms/step - loss: 0.1876 - accuracy: 0.9452 - val_lo
ss: 0.0569 - val_accuracy: 0.9852
Epoch 14/15
31367/31367 [==============================] - 81s 3ms/step - loss: 0.2007 - accuracy: 0.9430 - val_lo
ss: 0.0629 - val_accuracy: 0.9811
Epoch 15/15
31367/31367 [==============================] - 81s 3ms/step - loss: 0.1914 - accuracy: 0.9463 - val_lo
ss: 0.0676 - val_accuracy: 0.9813
```

حصل نموذجنا على دقة 95٪في مجموعة بيانات التدريب. باستخدام matplotlib، نرسم الرسم البياني للدقة والخطأ.

```
[13]: plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()

plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
```

```
[13]: <matplotlib.legend.Legend at 0x24eece89e48>
```



الخطوة 4: اختبار النموذج باستخدام مجموعة بيانات الاختبار

تحتوي مجموعة البيانات الخاصة بنا على مجلد اختبار وفي ملف test.csv، لدينا التفاصيل المتعلقة بمسار الصورة وتسميات الفئات الخاصة بكل منها. نقوم باستخراج مسار الصورة والتسميات باستخدام pandas. ثم للتنبؤ بالنموذج، يتعين علينا تغيير حجم صورنا إلى 30 × 30 بكسل وإنشاء مصفوفة عددية تحتوي على جميع بيانات الصورة. من sklearn.metrics ، قمنا باستيراد accuracy_score ولاحظنا كيف توقع نموذجنا التسميات الفعلية. حققنا دقة 95٪ في هذا النموذج.

```
[14]: from sklearn.metrics import accuracy_score
      import pandas as pd
      y_test = pd.read_csv('Test.csv')

      labels = y_test["ClassId"].values
      imgs = y_test["Path"].values

      data=[]

      for img in imgs:
          image = Image.open(img)
          image = image.resize((30,30))
          data.append(np.array(image))

      X_test=np.array(data)

      pred = model.predict_classes(X_test)

      #Accuracy with the test data
      from sklearn.metrics import accuracy_score
      accuracy_score(labels, pred)

[14]: 0.9532066508313539
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

data = []
labels = []
classes = 43
cur_path = os.getcwd()

#Retrieving the images and their labels
for i in range(classes):
    path = os.path.join(cur_path,'train',str(i))
    images = os.listdir(path)

    for a in images:
        try:
            image = Image.open(path + '\\'+ a)
            image = image.resize((30,30))
            image = np.array(image)
            #sim = Image.fromarray(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")

#Converting lists into numpy arrays
data = np.array(data)
labels = np.array(labels)

print(data.shape, labels.shape)
#Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data, labels,
test_size=0.2, random_state=42)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

#Converting the labels into one hot encoding
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)

#Building the model
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',
input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
```

```
#Compilation of the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

epochs = 15
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,
validation_data=(X_test, y_test))
model.save("my_model.h5")

#plotting graphs for accuracy
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()

plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()

#testing accuracy on test dataset
from sklearn.metrics import accuracy score

y_test = pd.read_csv('Test.csv')

labels = y_test["ClassId"].values
imgs = y_test["Path"].values

data=[]

for img in imgs:
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))

X_test=np.array(data)

pred = model.predict_classes(X_test)

#Accuracy with the test data
from sklearn.metrics import accuracy_score
print(accuracy_score(labels, pred))

model.save('traffic_classifier.h5')
```

واجهة مستخدم رسومية لمصنف إشارات المرور

سنقوم الآن ببناء واجهة مستخدم رسومية لمصنف إشارات المرور الخاص بنا باستخدام Tkinter. Tkinter هي مجموعة أدوات واجهة المستخدم الرسومية في مكتبة بايثون القياسية. أنشئ ملفًا جديدًا في مجلد المشروع وانسخ الكود أدناه. احفظه كـ gui.py ويمكنك تشغيل الكود بكتابة python gui.py في سطر الأوامر.

في هذا الملف، قمنا أولاً بتحميل النموذج المدرب "traffic_classifier.h5" باستخدام Keras. وبعد ذلك نقوم ببناء واجهة المستخدم الرسومية لتحميل الصورة ويستخدم زر لتصنيف الذي يستدعي دالة classify(). تقوم دالة classify() بتحويل الصورة إلى أبعاد الشكل (1, 30, 30, 3). هذا لأنه للتنبؤ بإشارة المرور، يتعين علينا توفير نفس البعد الذي استخدمناه عند بناء النموذج. ثم نتوقع الفئة، تعيد لنا model.predict_classes(image) رقمًا بين (0–42) يمثل الفئة التي تنتمي إليها. نستخدم القاموس للحصول على معلومات عن الفصل. إليك كود الملف gui.py.

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image

import numpy
#load the trained model to classify sign
from keras.models import load_model
model = load_model('traffic_classifier.h5')

#dictionary to label all traffic signs class.
classes = { 1:'Speed limit (20km/h)',
            2:'Speed limit (30km/h)',
            3:'Speed limit (50km/h)',
            4:'Speed limit (60km/h)',
            5:'Speed limit (70km/h)',
            6:'Speed limit (80km/h)',
            7:'End of speed limit (80km/h)',
            8:'Speed limit (100km/h)',
            9:'Speed limit (120km/h)',
            10:'No passing',
            11:'No passing veh over 3.5 tons',
            12:'Right-of-way at intersection',
            13:'Priority road',
            14:'Yield',
            15:'Stop',
            16:'No vehicles',
            17:'Veh > 3.5 tons prohibited',
            18:'No entry',
            19:'General caution',
            20:'Dangerous curve left',
            21:'Dangerous curve right',
            22:'Double curve',
            23:'Bumpy road',
            24:'Slippery road',
            25:'Road narrows on the right',
            26:'Road work',
            27:'Traffic signals',
            28:'Pedestrians',
            29:'Children crossing',
            30:'Bicycles crossing',
            31:'Beware of ice/snow',
            32:'Wild animals crossing',
            33:'End speed + passing limits',
            34:'Turn right ahead',
            35:'Turn left ahead',
            36:'Ahead only',
            37:'Go straight or right',
            38:'Go straight or left',
```

```
            39:'Keep right',
            40:'Keep left',
            41:'Roundabout mandatory',
            42:'End of no passing',
            43:'End no passing veh > 3.5 tons' }

#initialise GUI
top=tk.Tk()
top.geometry('800x600')
top.title('Traffic sign classification')
top.configure(background='#CDCDCD')

label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)

def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    pred = model.predict_classes([image])[0]
    sign = classes[pred+1]
    print(sign)
    label.configure(foreground='#011638', text=sign)

def show_classify_button(file_path):
    classify_b=Button(top,text="Classify Image",command=lambda:
classify(file_path),padx=10,pady=5)
    classify_b.configure(background='#364156',
foreground='white',font=('arial',10,'bold'))
    classify_b.place(relx=0.79,rely=0.46)

def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)

uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)
))
        im=ImageTk.PhotoImage(uploaded)

        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass

upload=Button(top,text="Upload an
image",command=upload_image,padx=10,pady=5)
upload.configure(background='#364156',
foreground='white',font=('arial',10,'bold'))

upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Know Your Traffic Sign",pady=20,
font=('arial',20,'bold'))
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

المخرجات:

الملخص

في مشروع بايثون هذا مع الكود المصدري، نجحنافي تصنيف مصنف إشارات المرور بدقة 95٪ وتصورنا أيضًا كيف تتغير الدقة والخطأ بمرور الوقت، وهو أمر جيد جدًا من نموذج بسيط لشبكة CNN.

37) كشف حرائق الغابات باستخدام التعلم العميق Forest Fire Detection using Deep Learning

تعتبر حرائق الغابات ظاهرة مهمة على نطاق عالمي، لأنها مسؤولة عن كميات كبيرة من الأضرار الاقتصادية والبيئية. تتفاقم هذه الآثار بسبب تأثير تغير المناخ.

من المهم اكتشاف الحريق وتحذير المسؤولين. لذلك سننشئ تطبيق كشف حرائق الغابات بالتعلم العميق.

تحميل المكتبات الضرورية

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, callbacks, optimizers
import os
import random
from PIL import Image
```

مجموعة البيانات

مجموعة البيانات المنسقة من Forest_fires_classification.

استخدم هذا السكريبت لتغيير حجم الصور إلى (45 × 45) وضغط مجموعة البيانات.

```
import os, cv2, glob

import multiprocessing as mp



def resize(file_path):

  image = cv2.imread(file_path)

  image = cv2.resize(image, (45, 45))

  target_path = os.path.splitext(file_path)[0] + ".jpg"

  cv2.imwrite(target_path, image)

  print("Resized '{}' to '{}'".format(file_path, target_path))

  return



def main():

  image_dir = "forest_fire_dataset"
```

```
  file_paths = []

  file_paths += glob.glob(image_dir+"/**/*.png", recursive = True)

  file_paths += glob.glob(image_dir+"/**/*.jpg", recursive = True)

  pool = mp.Pool(processes = (mp.cpu_count()*50))

  pool.map(resize, file_paths)



if __name__=="__main__":

  main()
```

تنزيل مجموعة البيانات المنسقة

```
data_dir = "forest_fire"
if not os.path.exists(data_dir):
    !wget https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-
admin/forest_fire.zip -O forest_fire.zip
    !unzip -qo forest_fire.zip
    !rm forest_fire.zip
```

```
tf.random.set_seed(50)
```

يحتوي مجلد مجموعة البيانات على مجلدين فرعيين – fire و no_fire يحتويان على صور من الأنواع المعنية.

```
print("Number of samples")
for f in os.listdir(data_dir + '/'):
    if os.path.isdir(data_dir + '/' + f):
        print(f, " : ", len(os.listdir(data_dir + '/' + f +'/')))
```

```
Number of samples
no_fire  :  23845
fire  :  30155
```

إنها مجموعة بيانات متوازنة.

```
batch_size = 64
image_size = (45, 45)


print("Training set")
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  image_size=image_size,
  validation_split=0.2,
  subset="training",
  seed=113,
  batch_size=batch_size)

print("Validation set")
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
```

```
  image_size=image_size,
  validation_split=0.2,
  subset="validation",
  seed=113,
  batch_size=batch_size)
```

```
Training set
Found 54000 files belonging to 2 classes.
Using 43200 files for training.
Validation set
Found 54000 files belonging to 2 classes.
Using 10800 files for validation.
```

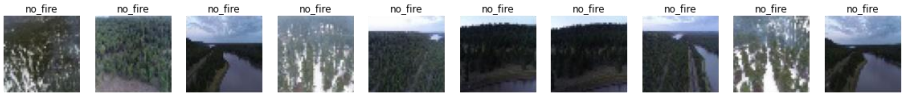
حدد class_names لاستخدامها لاحقًا.

```
class_names = train_ds.class_names
print(class_names)
```

```
['fire', 'no_fire']
```

رسم البيانات

```
num_samples = 10    # the number of samples to be displayed in each class

for x in class_names:
    plt.figure(figsize=(20, 20))
    filenames = os.listdir(os.path.join(data_dir, x))

    for i in range(num_samples):
        ax = plt.subplot(1, num_samples, i + 1)
        img = Image.open(os.path.join(data_dir, x, filenames[i]))
        plt.imshow(img)
        plt.title(x)
        plt.axis("off")
```



المعالجة المسبقة

تحديد شكل الإدخال

```
print("Shape of one training batch")

for image_batch, labels_batch in train_ds:
    input_shape = image_batch[0].shape
    print("Input: ", image_batch.shape)
    print("Labels: ", labels_batch.shape)
    print("Input Shape: ", input_shape)
    break
```

```
Shape of one training batch
Input:  (64, 45, 45, 3)
```

```
Labels:  (64,)
Input Shape:  (45, 45, 3)
```

تطبيع قيم البكسل

قيم البكسل هي الآن أعداد صحيحة بين 0 و 255. تغييرها إلى النطاق [0 ، 1] لتقارب أسرع.

```
# Normalizing the pixel values

normalization_layer = layers.experimental.preprocessing.Rescaling(1./255)

train_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
val_ds = val_ds.map(lambda x, y: (normalization_layer(x), y))
```

بناء النموذج

```
filepath = 'forest_fire.h5'

model = tf.keras.models.Sequential([
    layers.Conv2D(6, 3, activation='relu', input_shape=input_shape),
    layers.MaxPool2D(pool_size=(2, 2)),

    layers.Conv2D(12, 3, activation='relu'),
    layers.MaxPool2D(pool_size=(2, 2)),

#     layers.Conv2D(32, 3, activation='relu'),
#     layers.MaxPool2D(pool_size=(2, 2)),

    layers.Conv2D(6, 3, activation='relu'),
    layers.MaxPool2D(pool_size=(2, 2)),

    layers.Flatten(),
    layers.Dense(32, activation='relu'),
    layers.Dense(16, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 43, 43, 6)         168
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 21, 21, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 19, 19, 12)        660
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 9, 9, 12)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 6)           654
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 6)           0
_________________________________________________________________
flatten (Flatten)            (None, 54)                0
_________________________________________________________________
dense (Dense)                (None, 32)                1760
_________________________________________________________________
dense_1 (Dense)              (None, 16)                528
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17
=================================================================
Total params: 3,787
Trainable params: 3,787
Non-trainable params: 0
_________________________________________________________________
```

تجميع النموذج وتدريبه

```
cb = [
    callbacks.EarlyStopping(monitor = 'val_loss', patience = 10,
restore_best_weights = True),
    callbacks.ModelCheckpoint(filepath, monitor = "val_loss", save_best_only
= True)
    ]

model.compile(loss=tf.keras.losses.BinaryCrossentropy(),
              optimizer=optimizers.Adam(0.0001),
              metrics=['accuracy'])

history = model.fit(train_ds, validation_data =  val_ds, epochs=200,
callbacks = cb)
```

```
675/675 [==============================] - 2s 3ms/step - loss: 0.0209 - accuracy: 0.9930 - val_loss: 0.0440 - val_accuracy: 0.9871
Epoch 164/200
675/675 [==============================] - 2s 3ms/step - loss: 0.0224 - accuracy: 0.9924 - val_loss: 0.0421 - val_accuracy: 0.9879
Epoch 165/200
675/675 [==============================] - 2s 3ms/step - loss: 0.0203 - accuracy: 0.9932 - val_loss: 0.0440 - val_accuracy: 0.9874
Epoch 166/200
675/675 [==============================] - 2s 3ms/step - loss: 0.0204 - accuracy: 0.9931 - val_loss: 0.0421 - val_accuracy: 0.9880
Epoch 167/200
675/675 [==============================] - 2s 3ms/step - loss: 0.0200 - accuracy: 0.9934 - val_loss: 0.0422 - val_accuracy: 0.9878
Epoch 168/200
675/675 [==============================] - 2s 3ms/step - loss: 0.0212 - accuracy: 0.9929 - val_loss: 0.0437 - val_accuracy: 0.9873
```

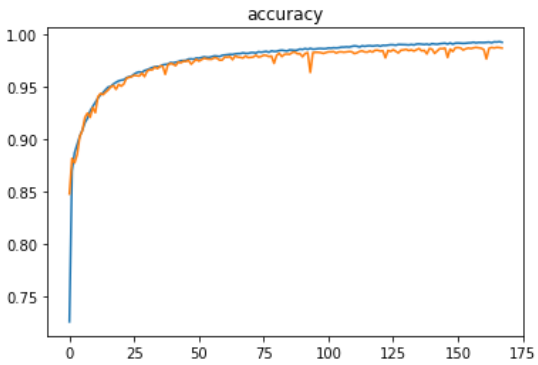
تقييم النموذج

```
model.evaluate(val_ds)
```

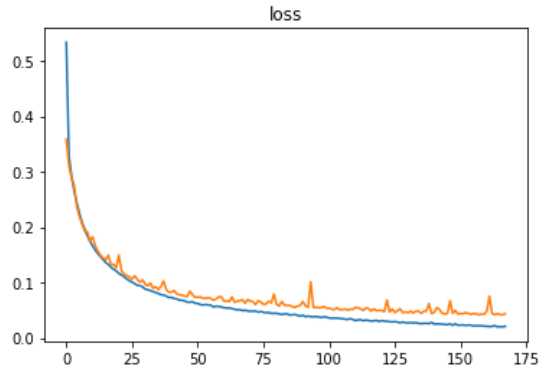
```
169/169 [==============================] - 0s 2ms/step - loss: 0.0419 -
accuracy: 0.9880
```

رسم المعايير

```
def plot(history, variable, variable2):
    plt.plot(range(len(history[variable])), history[variable])
    plt.plot(range(len(history[variable2])), history[variable2])
    plt.title(variable)

plot(history.history, "accuracy", 'val_accuracy')
```



```
plot(history.history, "loss", "val_loss")
```



التنبؤ

```
def predict(x):
    for i in val_ds.as_numpy_iterator():
        img, label = i
#         plt.figure()
#         plt.axis('off')   # remove axes
#         plt.imshow(img[x])    # shape from (32, 64, 64, 3) --> (64, 64, 3)

        print("True: ", class_names[label[x]])
        output = model.predict(np.expand_dims(img[x],0))[0][0]    # getting
output; input shape (64, 64, 3) --> (1, 64, 64, 3)
        pred = (output > 0.5).astype('int')
        print("Predicted: ", class_names[pred])    # Picking the label from
class_names base don the model output

        probability = (output*100) if pred==1 else (100 - (output*100))
        if (class_names[pred] == class_names[label[x]]):
            print("PREDICTION MATCHED | Probability:
{:.2f}%".format(probability))
        else:
            print("PREDICTION DID NOT MATCH | Probability:
{:.2f}%".format(probability))
        break
```

```
for i in range(10):
    # pick random test data sample from one batch
    x = random.randint(0, batch_size - 1)
    predict(x)
    print()
```

المخرجات:

```
True:  fire
Predicted:  fire
PREDICTION MATCHED | Probability: 100.00%

True:  no_fire
Predicted:  no_fire
PREDICTION MATCHED | Probability: 100.00%

True:  no_fire
```

```
Predicted:  no_fire
PREDICTION MATCHED | Probability: 99.88%

True:  fire
Predicted:  fire
PREDICTION MATCHED | Probability: 100.00%

True:  fire
Predicted:  fire
PREDICTION MATCHED | Probability: 100.00%

True:  fire
Predicted:  fire
PREDICTION MATCHED | Probability: 100.00%

True:  no_fire
Predicted:  no_fire
PREDICTION MATCHED | Probability: 100.00%

True:  no_fire
Predicted:  no_fire
PREDICTION MATCHED | Probability: 100.00%

True:  fire
Predicted:  fire
PREDICTION MATCHED | Probability: 99.50%

True:  fire
Predicted:  fire
PREDICTION MATCHED | Probability: 100.00%
```

38) كشف ورم الدماغ باستخدام التعلم العميق Brain Tumor Detection using Deep Learning

تعتبر أورام الدماغ Brain tumors من أكثر الأمراض شيوعًا وخطورة، مما يؤدي إلى قصر متوسط العمر المتوقعفي أعلى درجاته. وبالتالي، فإن تخطيط العلاج هو مرحلة أساسية لتحسين نوعية حياة المرضى. بشكل عام، تقنيات التصوير المختلفة مثل التصوير المقطعي (CT) والتصوير بالرنين المغناطيسي (MRI) ... إلخ

على وجه الخصوص،في هذا العمل، يتم استخدام صور التصوير بالرنين المغناطيسي لتشخيص الأورامفي الدماغ. ومع ذلك، فإن الكم الهائل من البيانات الناتجة عن فحوصات التصوير بالرنين المغناطيسي يحبط التصنيف اليدوي للورم مقابل غير الورمفي وقت معين. ولكن لديها بعض القيود (أي) يتم توفير قياسات كمية دقيقة لعدد محدود من الصور.

ومن ثم فإن مخططات التصنيف الموثوقة والتلقائية ضرورية لمنع معدل وفيات البشر. يعتبر التصنيف التلقائي لورم الدماغ مهمة صعبة للغايةفي التباين المكاني والبنيوي الكبير للمنطقة المحيطة بورم الدماغ.في هذا العمل، تم اقتراح الكشف التلقائي عن ورم الدماغ باستخدام تصنيف الشبكات العصبية التلافيفية (CNN).

مجموعة البيانات

تتكون مجموعة البيانات Dataset من مجلدين مختلفين هما نعم أو لا. يحتوي كلا المجلدين على صور مختلفة للرنين المغناطيسي للمرضى. نعم يحتوي المجلد على مرضى يعانون من أورام في المخ بينما لا يحتوي أي مجلد على صور التصوير بالرنين المغناطيسي للمرضى الذين لا يعانون من ورمفي المخ.

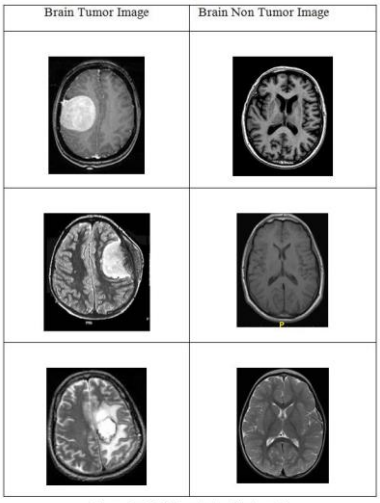


Figure 2: CNN based classified results

الخطوة 1: استيراد المكتبات ومجموعة البيانات المطلوبة.

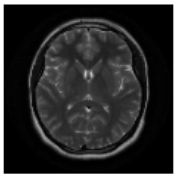
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import math
import cv2
import matplotlib.pyplot as plt
import os
import seaborn as sns
import umap
from PIL import Image
from scipy import misc
from os import listdir
from os.path import isfile, join
import numpy as np
from scipy import misc
from random import shuffle
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils.np_utils import to_categorical
```

الخطوة 2: تحميل مجموعة البيانات

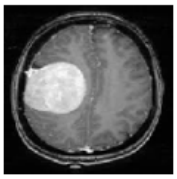
```
os.listdir('../input/brain_tumor_dataset')
from google.colab import drive
drive.mount('/content/drive')
```

الخطوة 3: عرض إحدى الصور من مجموعة البيانات "لا" و "نعم"

```
im =Image.open('../input/brain_tumor_dataset/no/1 no.jpeg').resize((128,128))
im
```



```
im =Image.open('../input/brain_tumor_dataset/yes/Y1.jpg').resize((128,128))
im
```



الخطوة 4: إنشاء مجموعات البيانات المستهدفة والتحقق منها

```
yes=os.listdir('../input/brain_tumor_dataset/yes')
no=os.listdir('../input/brain_tumor_dataset/no')
```

```
data=np.concatenate([yes,no])
len(data)==len(yes)+len(no)
```

```
True
```

```
target_x=np.full(len(yes),1)
target_y=np.full(len(no),0)
data_target=np.concatenate([target_x,target_y])
len(data_target)==len(target_x)+len(target_y)
```

```
True
```

```
len(data_target)==len(data)
```

```
True
```

Data_Target

```
data_target
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

الخطوة 5: المعالجة المسبقة للبيانات لفئتي "لا" و "نعم".

```
yes_values=os.listdir('../input/brain_tumor_dataset/yes')
no_values=os.listdir('../input/brain_tumor_dataset/no')
```

```
X_data =[]
for file in yes_values:
    #face = misc.imread('../input/brain_tumor_dataset/yes/'+file)
    img = cv2.imread('../input/brain_tumor_dataset/yes/'+file)
    face = cv2.resize(img, (32, 32) )
    (b, g, r)=cv2.split(face)
    img=cv2.merge([r,g,b])
    X_data.append(img)
```

```
#X_data =[]
for file in no_values:
    #face = misc.imread('../input/brain_tumor_dataset/yes/'+file)
    img = cv2.imread('../input/brain_tumor_dataset/no/'+file)
    face = cv2.resize(img, (32, 32) )
    (b, g, r)=cv2.split(face)
    img=cv2.merge([r,g,b])
    X_data.append(img)
```

```
len(X_data)==len(data)==len(data_target)
```

```
True
```

```
X = np.squeeze(X_data)
```

```
X.shape
```

```
(253, 32, 32, 3)
```

```
# normalize data
X = X.astype('float32')
X /= 255
```

الخطوة 6: التحقق من Data_target

```
data_target
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
(x_train, y_train), (x_test, y_test) = (X[:190],data_target[:190]) , (X[190:] , data_target[190:])
(x_valid , y_valid) = (x_test[:63], y_test[:63])
#(x_test, y_test) = (x_test[63:], y_test[63:])
```

الخطوة 7: إنشاء شبكة عصبية باستخدام Keras

```
model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=16,kernel_size=9, padding='same', activation='relu', input_shape=(32,32,3)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.45))

model.add(tf.keras.layers.Conv2D(filters=16,kernel_size=9,padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.25))

model.add(tf.keras.layers.Conv2D(filters=36, kernel_size=9, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.25))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dropout(0.15))


model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# Take a look at the model summary
model.summary()
```

إخراج ملخص النموذج:

```
Colocations handled automatically by placer.
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/tensorflow/python/keras/layers/core.py:143: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is depr
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 16)	3904
max_pooling2d (MaxPooling2D)	(None, 16, 16, 16)	0
dropout (Dropout)	(None, 16, 16, 16)	0
conv2d_1 (Conv2D)	(None, 16, 16, 16)	20752
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 16)	0
dropout_1 (Dropout)	(None, 8, 8, 16)	0
conv2d_2 (Conv2D)	(None, 8, 8, 36)	46692
max_pooling2d_2 (MaxPooling2	(None, 4, 4, 36)	0
dropout_2 (Dropout)	(None, 4, 4, 36)	0
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 512)	295424
dropout_3 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1)	513

```
Total params: 367,285
Trainable params: 367,285
Non-trainable params: 0
```

الخطوة 8: تجميع وتدريب النموذج

```
model.compile(loss='binary_crossentropy',
              optimizer=tf.keras.optimizers.Adam(),
              metrics=['acc'])
```

```
model.fit(x_train,
          y_train,
          batch_size=128,
          epochs=150,
          validation_data=(x_valid, y_valid),)
```

```
Train on 190 samples, validate on 63 samples
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be rem
Instructions for updating:
Use tf.cast instead.
Epoch 1/150
190/190 [==============================] - 1s 7ms/sample - loss: 0.6482 - acc: 0.4316 - val_loss: 1.8161 - val_acc: 0.0000e+00
Epoch 2/150
190/190 [==============================] - 1s 3ms/sample - loss: 0.5339 - acc: 0.8158 - val_loss: 0.9960 - val_acc: 0.0000e+00
Epoch 3/150
190/190 [==============================] - 1s 3ms/sample - loss: 0.4784 - acc: 0.8158 - val_loss: 0.8274 - val_acc: 0.0000e+00
Epoch 4/150
190/190 [==============================] - 1s 4ms/sample - loss: 0.5064 - acc: 0.8158 - val_loss: 0.9084 - val_acc: 0.0000e+00
Epoch 5/150
190/190 [==============================] - 1s 4ms/sample - loss: 0.4589 - acc: 0.8158 - val_loss: 1.1866 - val_acc: 0.0000e+00
Epoch 6/150
```

الخطوة 9: تقييم النموذج واختبار الدقة

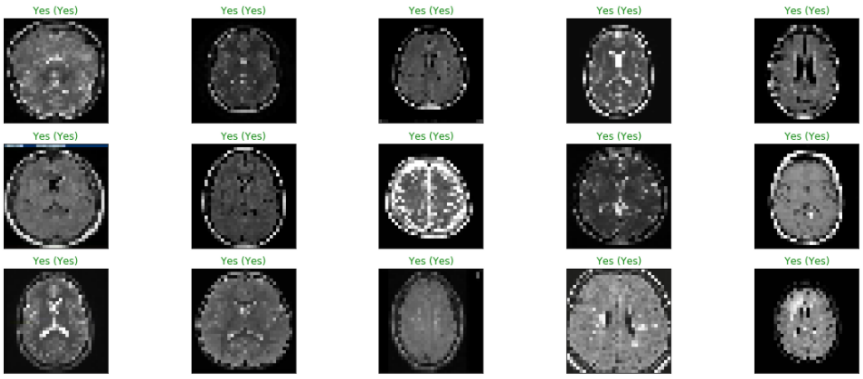
```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score[1])
```

```
Test accuracy: 0.7619048
```

الخطوة 10: التوقع باستخدام صور اختبار مختلفة.

```
y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                                  color=("green" if predict_index == true_index else "red"))
plt.show()
```

إخراج صور الاختبار المتوقعة.



39) تصنيف الفواكه باستخدام التعلم العميق Fruits Classification using Deep Learning

في هذا المشروع، سنقوم بتصنيف الفاكهة وعرض اسمها كمخرج من الصورة المعطاة للفاكهة كمدخلات.

مجموعة البيانات: https://www.kaggle.com/sshikamaru/fruit-recognition

تتكون مجموعة البيانات من 33 نوعًا مختلفًا من الفاكهة. تمت تسمية كل مجلد باسم نوع من الفاكهة ويحتوي على أكثر من 400 صورة لتلك الفاكهة بزوايا وإضاءة مختلفة. بناءً على الصورة المقدمة، نحتاج إلى تصنيف الفاكهة كواحدة من 33 فئة.

المعالجة المسبقة

أولاً، قم باستيراد جميع المكتبات المطلوبة:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.model_selection import train_test_split
```

قم بتنزيل البيانات وفك ضغطها للوصول إلى الصور والتسميات من النوتبوك الخاص بك.

```
!wget -N "https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-admin/fruits.zip"
!unzip -qo fruits.zip
```

يوجد إجمالي 33 نوعًا من الفاكهةفي مجموعة البيانات الخاصة بنا. ارسم خريطة لهم وطباعتهم وفقًا لذلك.

```
Name=[]
for file in os.listdir(directory):
    Name+=[file]
print(Name)
print(len(Name))

fruit_map = dict(zip(Name, [t for t in range(len(Name))]))
print(fruit_map)
r_fruit_map=dict(zip([t for t in range(len(Name))],Name))
```

المخرجات:

```
['Pepper Green', 'Lemon', 'Cantaloupe', 'Passion Fruit', 'Pineapple',
'Apricot', 'Banana', 'Pomegranate', 'Pear', 'Avocado', 'Potato Red', 'Plum',
```

```
'Cucumber Ripe', 'Strawberry', 'Cactus fruit', 'Raspberry', 'Tomato', 'Pepper
Red', 'Peach', 'Blueberry', 'Onion White', 'Orange', 'Watermelon', 'Kiwi',
'Limes', 'Apple Granny Smith', 'Apple Braeburn', 'Cherry', 'Grape Blue',
'Corn', 'Mango', 'Clementine', 'Papaya']
33
```

تقسيم الصور إلى مجموعات تدريب، والتحقق من الصحة، واختبار

قم بإجراء زيادة البيانات باستخدام ImageDataGenerator حتى نتمكن من الحصول على المزيد من البيانات ذات الصلة من الصور الموجودة عن طريق إجراء تعديلات طفيفة على مجموعة البيانات.

```
img_datagen = ImageDataGenerator(rescale=1./255,
                                 vertical_flip=True,
                                 horizontal_flip=True,
                                 rotation_range=40,
                                 width_shift_range=0.2,
                                 height_shift_range=0.2,
                                 zoom_range=0.1,
                                 validation_split=0.2)
test_datagen = ImageDataGenerator(rescale=1./255)
```

قسّم مجموعة بيانات التدريب إلى مجموعة تدريب ومجموعة تحقق.

```
train_generator = img_datagen.flow_from_directory(directory,
                                                  shuffle=True,
                                                  batch_size=32,
                                                  subset='training',
                                                  target_size=(100, 100))

valid_generator = img_datagen.flow_from_directory(directory,
                                                  shuffle=True,
                                                  batch_size=16,
                                                  subset='validation',
                                                  target_size=(100, 100))
```

المخرجات:

```
Found 13309 images belonging to 33 classes.
Found 3314 images belonging to 33 classes.
```

تدريب النموذج

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=(100,100,3),
activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(len(fruit_map)))
model.add(Activation('softmax'))

model.summary()
```

المخرجات:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 100, 100, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 50, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 50, 50, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 6, 6, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 256)               16640
_________________________________________________________________
activation (Activation)      (None, 256)               0
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 33)                8481
_________________________________________________________________
activation_1 (Activation)    (None, 33)                0
=================================================================
Total params: 192,225
Trainable params: 192,225
Non-trainable params: 0
_________________________________________________________________
```

تجميع النموذج وتدريبه

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(train_generator, validation_data=valid_generator,


steps_per_epoch=train_generator.n//train_generator.batch_size,


validation_steps=valid_generator.n//valid_generator.batch_size,
                    epochs=10)
```

المخرجات:

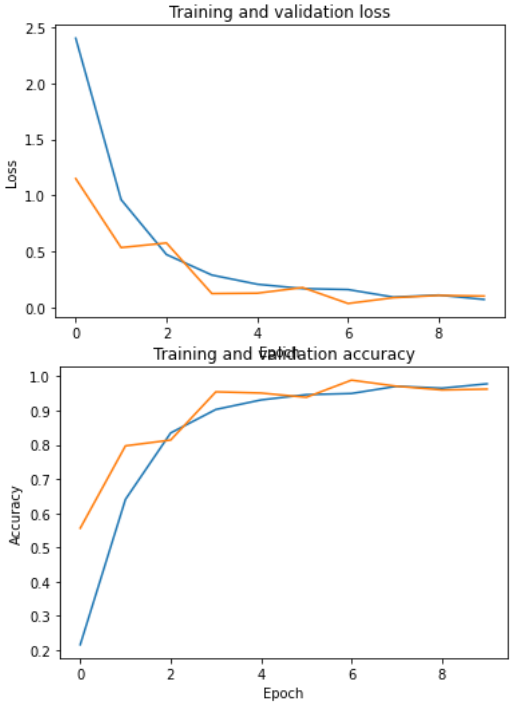
```
Epoch 1/10
415/415 [==============================] - 44s 107ms/step - loss: 2.4032 -
accuracy: 0.2161 - val_loss: 1.1493 - val_accuracy: 0.5565
Epoch 2/10
415/415 [==============================] - 44s 105ms/step - loss: 0.9614 -
accuracy: 0.6407 - val_loss: 0.5330 - val_accuracy: 0.7971
Epoch 3/10
415/415 [==============================] - 44s 105ms/step - loss: 0.4707 -
accuracy: 0.8348 - val_loss: 0.5742 - val_accuracy: 0.8140
Epoch 4/10
415/415 [==============================] - 44s 106ms/step - loss: 0.2877 -
accuracy: 0.9034 - val_loss: 0.1222 - val_accuracy: 0.9550
Epoch 5/10
415/415 [==============================] - 44s 105ms/step - loss: 0.2055 -
accuracy: 0.9313 - val_loss: 0.1255 - val_accuracy: 0.9514
Epoch 6/10
415/415 [==============================] - 44s 106ms/step - loss: 0.1668 -
accuracy: 0.9467 - val_loss: 0.1759 - val_accuracy: 0.9390
Epoch 7/10
415/415 [==============================] - 44s 106ms/step - loss: 0.1580 -
accuracy: 0.9501 - val_loss: 0.0335 - val_accuracy: 0.9891
Epoch 8/10
415/415 [==============================] - 44s 105ms/step - loss: 0.0906 -
accuracy: 0.9713 - val_loss: 0.0852 - val_accuracy: 0.9713
Epoch 9/10
415/415 [==============================] - 39s 94ms/step - loss: 0.1087 -
accuracy: 0.9657 - val_loss: 0.1057 - val_accuracy: 0.9604
Epoch 10/10
415/415 [==============================] - 37s 89ms/step - loss: 0.0695 -
accuracy: 0.9785 - val_loss: 0.0999 - val_accuracy: 0.9629
```

الرسوم البيانية للخطأ والدقة

ارسم منحنيات الخطأ والدقة لفهم أداء نموذجنا.

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and validation loss')
plt.show()
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and validation accuracy')
plt.show()
```



توقع فاكهة من مجموعة الاختبار

- لقد اخترت صورة عشوائية من مجموعة الاختبار.

- قم بتحميل الصورة وتحويلها إلى الحجم المناسب.

```
image=load_img("fruits/test/test/0030.jpg",target_size=(100,100))

image=img_to_array(image)
image=image/255.0
```

```
prediction_image=np.array(image)
prediction_image= np.expand_dims(image, axis=0)
```

- قم بعمل تنبؤات.

```
prediction=model.predict(prediction_image)
value=np.argmax(prediction)
move_name=mapper(value)
print("Prediction is {}.".format(move_name))
```

المخرجات:

```
Prediction is Pineapple.
```

الملخص

قمنا بتدريب نموذج متسلسل في keras للتنبؤ باسم الفاكهة مع صورة للفاكهة كمدخل.

رابط الكود: https://cainvas.ai-tech.systems/use-cases/fruits-classification-app/

40) مولد تسميات توضيحية للصور مع CNN وLSTM Image Caption Generator with CNN & LSTM

لقد رأيت صورة ويمكن لعقلك أن يخبرك بسهولة عن ماهية الصورة، ولكن هل يستطيع الكمبيوتر معرفة ما تمثله الصورة؟ لقد عمل باحثو الرؤية الحاسوبية على هذا الأمر كثيرًا واعتبروه مستحيلًا حتى الآن! مع التقدمفي تقنيات التعلم العميق، وتوافر مجموعات البيانات الضخمة وقوة الكمبيوتر، يمكننا بناء نماذج يمكنها إنشاء تسميات توضيحية للصورة.

هذا ما سنقوم بتنفيذهفي هذا المشروع المستند إلى بايثون حيث سنستخدم تقنيات التعلم العميق للشبكات العصبية التلافيفية (CNN) ونوع من الشبكات العصبية المتكررة (LSTM) معًا.

ما هو مولد التسمية التوضيحية للصور؟

مولد التسمية التوضيحية Image caption generator الصور هي مهمة تتضمن الرؤية الحاسوبية ومفاهيم معالجة اللغة الطبيعية للتعرف على سياق الصورة ووصفها بلغة طبيعية مثل اللغة الإنجليزية.

مولد التسمية التوضيحية للصور مع CNN - حول المشروع المستند إلى بايثون

الهدف من مشروعنا هو تعلم مفاهيم نموذج CNN وLSTM وبناء نموذج عمل لمولد تسمية توضيحية للصور من خلال تطبيق CNN مع LSTM.

في مشروع بايثون هذا، سنقوم بتنفيذ منشئ التسميات التوضيحية باستخدام CNN (الشبكات العصبية التلافيفية) وLSTM (الذاكرة طويلة قصيرة المدى). سيتم استخراج ميزات الصورة من Xception وهو نموذج CNN تم تدريبه على مجموعة بيانات imagenet ثم نقوم بإدخال الميزاتفي نموذج LSTM الذي سيكون مسؤولاً عن إنشاء تعليقات على الصورة.

مجموعة بيانات مشروع بايثون

بالنسبة لمولد التسمية التوضيحية للصور، سنستخدم مجموعة بيانات Flickr_8K. هناك أيضًا مجموعات بيانات كبيرة أخرى مثل مجموعة بيانات Flickr_30K وMSCOCO، ولكن قد يستغرق الأمر أسابيع فقط لتدريب الشبكة، لذلك سنستخدم مجموعة بيانات Flickr8k صغيرة. ميزة مجموعة البيانات الضخمة هي أنه يمكننا بناء نماذج أفضل.

بفضل جيسون براونلي لتوفيره رابطًا مباشرًا لتنزيل مجموعة البيانات (الحجم: 1 جيجابايت).

- Flicker8k_Dataset
- Flickr_8k_text

يحتوي المجلد Flickr_8k_text على ملف Flickr8k.token وهو الملف الرئيسي لمجموعة البيانات التي تحتوي على اسم الصورة والتعليقات التوضيحية الخاصة بها مفصولة بسطر جديد ("\n").

المتطلبات الاساسية

يتطلب هذا المشروع معرفة جيدة بالتعلم العميق وبايثون والعمل على دفاتر Jupyter ومكتبة Keras وNumpy ومعالجة اللغة الطبيعية.

تأكد من تثبيت جميع المكتبات الضرورية التالية:

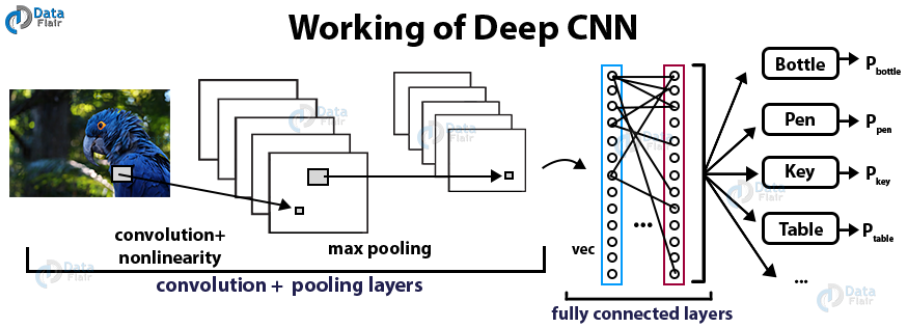
- pip install tensorflow
- keras
- pillow
- numpy
- tqdm
- jupyterlab

مولد التسمية التوضيحية للصورة – مشروع قائم على بايثون

ما هي CNN؟

الشبكات العصبية التلافيفية CNN هي شبكات عصبية عميقة متخصصة يمكنها معالجة البيانات التي لها شكل إدخال مثل مصفوفة ثنائية الأبعاد. يتم تمثيل الصور بسهولة على شكل مصفوفة ثنائية الأبعاد، وتعد شبكة CNN مفيدة جدًافي العمل مع الصور.

تُستخدم CNN أساسًا لتصنيفات الصور وتحديد ما إذا كانت الصورة عبارة عن طائر أو طائرة أو سوبرمان، إلخ.

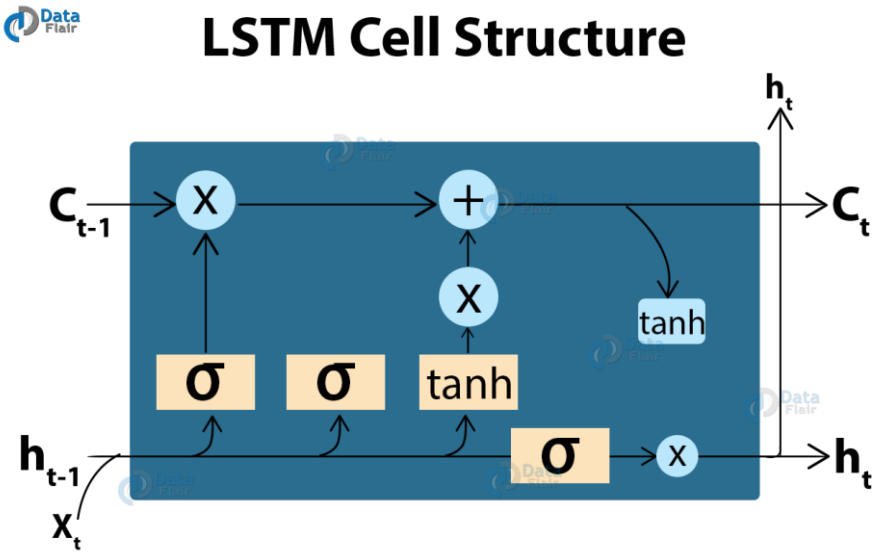


يقوم بمسح الصور من اليسار إلى اليمين ومن أعلى إلى أسفل لسحب الميزات المهمة من الصورة ويجمع الميزة لتصنيف الصور. يمكنه التعامل مع الصور التي تمت ترجمتها وتدويرها وتغيير حجمها والتغييراتفي المنظور.

ما هو LSTM؟

LSTM تعني الذاكرة طويلة قصيرة المدى، وهي نوع من RNN (الشبكة العصبية المتكررة) وهي مناسبة تمامًا لمشاكل التنبؤ بالتسلسل. بناءً على النص السابق، يمكننا توقع الكلمة التالية. لقد أثبتت فعاليتها من RNN التقليدية من خلال التغلب على قيود RNN التي كانت لها ذاكرة قصيرة المدى. يمكن لـ LSTM تنفيذ المعلومات ذات الصلة خلال معالجة المدخلات ومع بوابة النسيان forget gate، فإنها تتجاهل المعلومات غير ذات الصلة.

هذا ما تبدو عليه خلية LSTM:

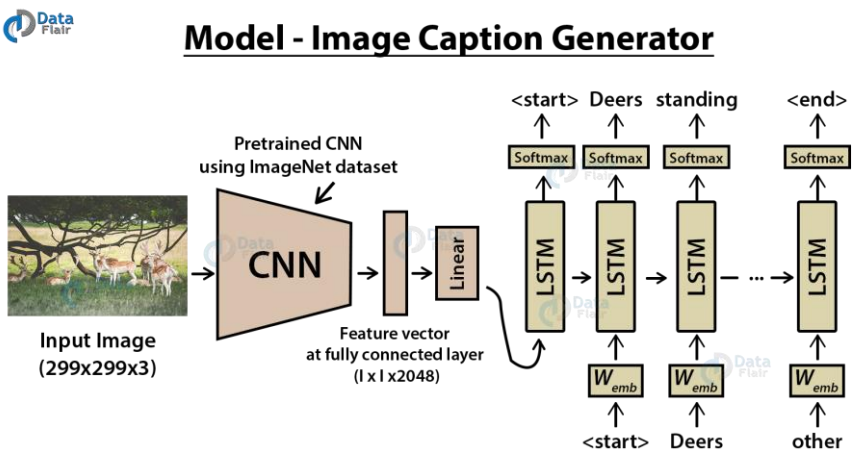


نموذج مولد التسمية التوضيحية للصورة

لذلك، لإنشاء نموذج مولد تسمية توضيحية للصور، سنقوم بدمج هذه البُنى. ويسمى أيضًا نموذج CNN-RNN.

يستخدم CNN لاستخراج الميزات من الصورة. سوف نستخدم نموذج Xception المدربين مسبقًا.

ستستخدم LSTM المعلومات من CNN للمساعدةفي إنشاء وصف للصورة.



هيكل ملف المشروع

تم التنزيل من مجموعة البيانات:

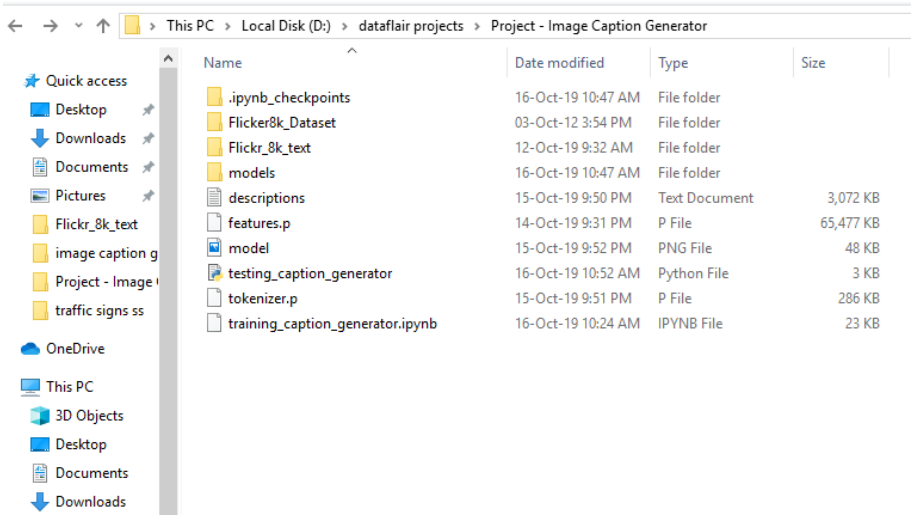
- Flicker8k_Dataset: مجلد مجموعة البيانات الذي يحتوي على 8091 صورة.
- Flickr_8k_text: مجلد مجموعة البيانات الذي يحتوي على ملفات نصية وتعليقات توضيحية للصور.

سيتم إنشاء الملفات أدناه من قبلنا أثناء إنشاء المشروع.

- **Models** – ستحتوي على نماذجنا المدربة.
- **Descriptions.txt** – يحتوي هذا الملف النصي على جميع أسماء الصور والتعليقات التوضيحية الخاصة بها بعد المعالجة المسبقة.
- **Features.p** – كائن Pickle الذي يحتوي على صورة ومتجه سماتها المستخرج من نموذج Xception المدربين مسبقًا على CNN.
- **Tokenizer.p** – يحتوي على الرموز المميزة التي تم تعيينها بقيمة فهرس.
- **Model.png** – تمثيل مرئي لأبعاد مشروعنا.
- **Testing_caption_generator.py** – ملف بايثون لتوليد تسمية توضيحية لأي صورة.
- **Training_caption_generator.ipynb** – دفتر Jupyter الذي ندرب فيه ونبني منشئ التسمية التوضيحية للصور.

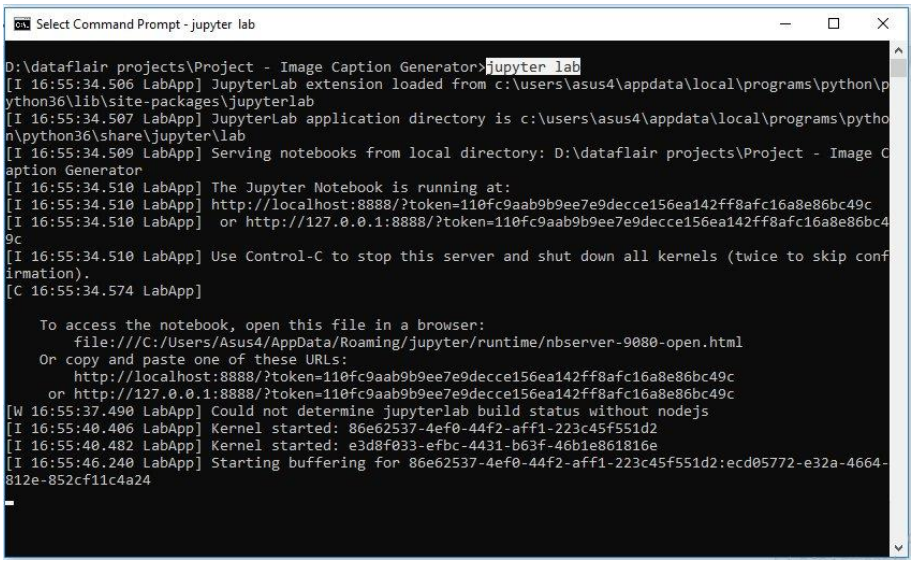
يمكنك تحميل جميع الملفات من الرابط:

Image Caption Generator – Python Project Files



بناء مشروع بايثون

لنبدأ بتهيئة خادم jupyter notebook عن طريق كتابة مختبر jupyterفي وحدة التحكم الخاصة بمجلد مشروعك. سيفتح النوتبوك بايثون التفاعلي حيث يمكنك تشغيل التعليمات البرمجية الخاصة بك. قم بإنشاء دفتر ملاحظات Python3 وقم بتسميته **training_caption_generator.ipynb**



الخطوة 1: أولاً ، نقوم باستيراد جميع الحزم اللازمة:

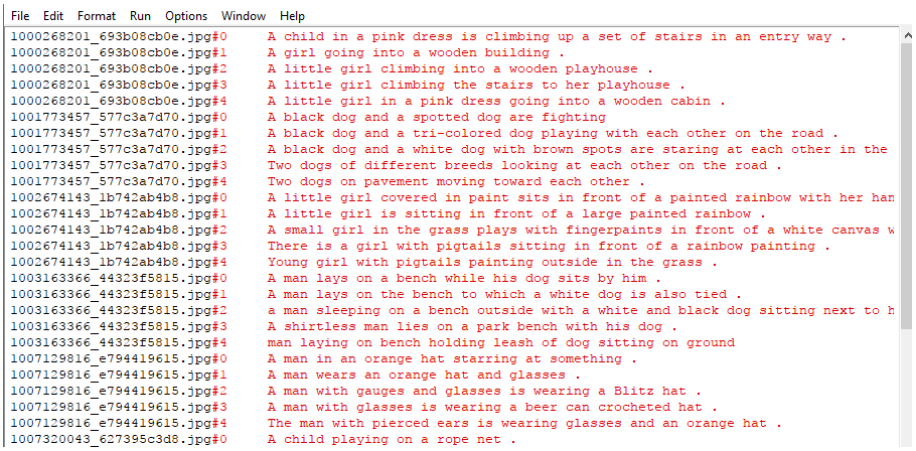
```
import string
```

```
import numpy as np
from PIL import Image
import os
from pickle import dump, load
import numpy as np

from keras.applications.xception import Xception, preprocess_input
from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.layers.merge import add
from keras.models import Model, load model
from keras.layers import Input, Dense, LSTM, Embedding, Dropout

# small library for seeing the progress of loops.
from tqdm import tqdm_notebook as tqdm
tqdm().pandas()
```

الخطوة 2: الحصول على البيانات وتنظيفها

الملف النصي الرئيسي الذي يحتوي على جميع تعليقات الصور هو Flickr8k.token في مجلد Flickr_8k_text الخاص بنا.

الق نظرة على الملف:



File Edit Format Run Options Window Help

```
1000268201_693b08cb0e.jpg#0     A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg#1     A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2     A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg#3     A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg#4     A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg#0     A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg#1     A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg#2     A black dog and a white dog with brown spots are staring at each other in the
1001773457_577c3a7d70.jpg#3     Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg#4     Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg#0     A little girl covered in paint sits in front of a painted rainbow with her han
1002674143_1b742ab4b8.jpg#1     A little girl is sitting in front of a large painted rainbow .
1002674143_1b742ab4b8.jpg#2     A small girl in the grass plays with fingerpaints in front of a white canvas w
1002674143_1b742ab4b8.jpg#3     There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_1b742ab4b8.jpg#4     Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg#0     A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg#1     A man lays on the bench to which a white dog is also tied .
1003163366_44323f5815.jpg#2     a man sleeping on a bench outside with a white and black dog sitting next to h
1003163366_44323f5815.jpg#3     A shirtless man lies on a park bench with his dog .
1003163366_44323f5815.jpg#4     man laying on bench holding leash of dog sitting on ground
1007129816_e794419615.jpg#0     A man in an orange hat starring at something .
1007129816_e794419615.jpg#1     A man wears an orange hat and glasses .
1007129816_e794419615.jpg#2     A man with gauges and glasses is wearing a Blitz hat .
1007129816_e794419615.jpg#3     A man with glasses is wearing a beer can crocheted hat .
1007129816_e794419615.jpg#4     The man with pierced ears is wearing glasses and an orange hat .
1007320043_627395c3d8.jpg#0     A child playing on a rope net .
```

تنسيق الملف لدينا هو صورة وتسمية توضيحية مفصولة بسطر جديد ("\n").

تحتوي كل صورة على 5 تسميات توضيحية ويمكننا أن نرى أنه تم تخصيص رقم من (0 إلى 5) لكل تسمية توضيحية.

سنحدد 5 دوال:

- **load_doc (filename)** : لتحميل ملف المستند وقراءة المحتويات داخل الملف في سلسلة.

- **all_img_captions (filename)** : ستنشئ هذه الدالة قاموسًا للأوصاف يقوم بتعيين الصور بقائمة من 5 تسميات توضيحية. سيبدو قاموس الأوصاف بالشكل التالي:

```
dataflair.py - C:/Users/Asus4/AppData/Local/Programs/Python/Python37-32/dataflair.py (3.7.4)
File  Edit  Format  Run  Options  Window  Help
{
'3461437556_cc5e97f3ac.jpg': ['dogs on grass',
                              'three dogs are running on the grass',
                              'three dogs one white and two brown are running together
                              'three dogs run along grassy yard',
                              'three dogs run together in the grass'
                              ],

'3461583471_2b8b6b4d73.jpg': ['buy is grinding rail on snowboard',
                              'person is jumping ramp on snowboard',
                              'snowboarder goes down ramp',
                              'snowboarder going over ramp',
                              'snowboarder performs jump on the clean white snow'
                              ],
'997722733_0cb5439472.jpg' : ['man in pink shirt climbs rock face',
                              'man is rock climbing high in the air',
                              'person in red shirt climbing up rock face covered in as
                              'rock climber in red shirt',
                              'rock climber practices on rock climbing wall'
                              ]

}
```

- **cleaning_text(descriptions)** ـ تأخذ هذه الدالة جميع الأوصاف وتقوم بتنظيف البيانات. هذه خطوة مهمة عندما نعمل مع البيانات النصية ، وفقًا لهدفنا، نقرر نوع التنظيف الذي نريد القيام به على النص. في حالتنا ، سنزيل علامات الترقيم، ونحول كل النص إلى أحرف صغيرة ونزيل الكلمات التي تحتوي على أرقام. لذلك ، سيتم تحويل تعليق مثل "A man riding on a three-wheeled wheelchair " إلى "man riding on three wheeled wheelchair "
- **text_vocabulary(descriptions)** : هذه دالة بسيطة تفصل كل الكلمات الفريدة وتخلق المفردات من كل الأوصاف.
- **save_ save_descriptions(descriptions, filename)** : ستنشئ هذه الدالة قائمة بجميع الأوصاف التي تمت معالجتها مسبقًا وتخزينها في ملف. سننشئ ملف descriptions.txt لتخزين جميع التسميات التوضيحية. سيبدو شيئا من هذا القبيل:

```
File  Edit  Format  Run  Options  Window  Help
1000268201_693b08cb0e.jpg       child in pink dress is climbing up set of stairs ir
1000268201_693b08cb0e.jpg       girl going into wooden building
1000268201_693b08cb0e.jpg       little girl climbing into wooden playhouse
1000268201_693b08cb0e.jpg       little girl climbing the stairs to her playhouse
1000268201_693b08cb0e.jpg       little girl in pink dress going into wooden cabin
1001773457_577c3a7d70.jpg       black dog and spotted dog are fighting
1001773457_577c3a7d70.jpg       black dog and tricolored dog playing with each othe
1001773457_577c3a7d70.jpg       black dog and white dog with brown spots are starir
1001773457_577c3a7d70.jpg       two dogs of different breeds looking at each other
1001773457_577c3a7d70.jpg       two dogs on pavement moving toward each other
1002674143_1b742ab4b8.jpg       little girl covered in paint sits in front of paint
1002674143_1b742ab4b8.jpg       little girl is sitting in front of large painted ra
1002674143_1b742ab4b8.jpg       small girl in the grass plays with fingerpaints in
1002674143_1b742ab4b8.jpg       there is girl with pigtails sitting in front of rai
1002674143_1b742ab4b8.jpg       young girl with pigtails painting outside in the gr
1003163366_44323f5815.jpg       man lays on bench while his dog sits by him
```

الكود:

```
# Loading a text file into memory
def load_doc(filename):
    # Opening the file as read only
    file = open(filename, 'r')
    text = file.read()
    file.close()
    return text

# get all imgs with their captions
def all_img_captions(filename):
    file = load_doc(filename)
    captions = file.split('\n')
    descriptions ={}
    for caption in captions[:-1]:
        img, caption = caption.split('\t')
        if img[:-2] not in descriptions:
            descriptions[img[:-2]] = [ caption ]
        else:
            descriptions[img[:-2]].append(caption)
    return descriptions

#Data cleaning- lower casing, removing puntuations and words
containing numbers
def cleaning_text(captions):
    table = str.maketrans('','',string.punctuation)
    for img,caps in captions.items():
        for i,img_caption in enumerate(caps):

            img_caption.replace("-"," ")
            desc = img_caption.split()

            #converts to lowercase
            desc = [word.lower() for word in desc]
            #remove punctuation from each token
            desc = [word.translate(table) for word in desc]
            #remove hanging 's and a
            desc = [word for word in desc if(len(word)>1)]
            #remove tokens with numbers in them
            desc = [word for word in desc if(word.isalpha())]
            #convert back to string

            img_caption = ' '.join(desc)
            captions[img][i]= img_caption
    return captions

def text_vocabulary(descriptions):
    # build vocabulary of all unique words
    vocab = set()

    for key in descriptions.keys():
        [vocab.update(d.split()) for d in descriptions[key]]

    return vocab

#All descriptions in one file
def save_descriptions(descriptions, filename):
    lines = list()
    for key, desc_list in descriptions.items():
        for desc in desc_list:
            lines.append(key + '\t' + desc )
```

```
    data = "\n".join(lines)
    file = open(filename,"w")
    file.write(data)
    file.close()


# Set these path according to project folder in you system
dataset_text = "D:\dataflair projects\Project - Image Caption
Generator\Flickr_8k_text"
dataset_images = "D:\dataflair projects\Project - Image Caption
Generator\Flicker8k_Dataset"

#we prepare our text data
filename = dataset_text + "/" + "Flickr8k.token.txt"
#loading the file that contains all data
#mapping them into descriptions dictionary img to 5 captions
descriptions = all_img_captions(filename)
print("Length of descriptions =" ,len(descriptions))

#cleaning the descriptions
clean_descriptions = cleaning_text(descriptions)

#building vocabulary
vocabulary = text_vocabulary(clean_descriptions)
print("Length of vocabulary = ", len(vocabulary))

#saving each description to file
save_descriptions(clean_descriptions, "descriptions.txt")
```

الخطوة 3: استخراج متجه الميزة من جميع الصور.

تسمى هذه التقنية أيضًا بنقل التعلم transfer learning، ولا يتعين علينا القيام بكل شيء بمفردنا، فنحن نستخدم النموذج المدرب مسبقًا الذي تم تدريبه بالفعل على مجموعات البيانات الكبيرة واستخراج الميزات من هذه النماذج واستخدامهافي مهامنا. نحن نستخدم نموذج Xception الذي تم تدريبه على مجموعة بيانات imagenet التي تحتوي على 1000 فئة مختلفة لتصنيفها. يمكننا استيراد هذا النموذج مباشرة من تطبيقات keras. تأكد من اتصالك بالإنترنت حيث يتم تنزيل الأوزان تلقائيًا. نظرًا لأن نموذج Xception قد تم إنشاؤهفي الأصل من أجل imagenet، فسنقوم بإجراء تغييرات صغيرة للتكامل مع نموذجنا. شيء واحد يجب ملاحظته هو أن نموذج Xception يأخذ حجم الصورة 299 * 299 * 3 كمدخل. سنزيل طبقة التصنيف الأخيرة ونحصل على 2048 متجه الميزة.

```
model = Xception( include_top=False, pooling='avg' )
```

ستستخرج الدالة **extract_features()** ميزات لجميع الصور وسنقوم بتعيين أسماء الصور باستخدام مصفوفة الميزات الخاصة بها. ثم سنقوم بتفريغ قاموس الميزاتفي ملف pickle "features.p".

الكود:

```
def extract_features(directory):
        model = Xception( include_top=False, pooling='avg' )
        features = {}
        for img in tqdm(os.listdir(directory)):
            filename = directory + "/" + img
            image = Image.open(filename)
            image = image.resize((299,299))
            image = np.expand_dims(image, axis=0)
            #image = preprocess input(image)
            image = image/127.5
            image = image - 1.0

            feature = model.predict(image)
            features[img] = feature
        return features

#2048 feature vector
features = extract_features(dataset_images)
dump(features, open("features.p","wb"))
```

In [44]:
```
# Now let's extract the features from our xception model
def extract_features(directory):
        model = Xception( include_top=False, pooling='avg' )
        features = {}
        for img in tqdm(os.listdir(directory)):
            filename = directory + "/" + img
            image = Image.open(filename)
            image = image.resize((299,299))
            image = np.expand_dims(image, axis=0)
            #image = preprocess_input(image)
            image = image/127.5
            image = image - 1.0

            feature = model.predict(image)
            features[img] = feature
        return features
```

In [45]:
```
#2048 feature vector
features = extract_features(dataset_images)
dump(features, open("features.p","wb"))
```

```
100% ████████████████████████████████ 8091/8091 [06:29<00:00, 20.78it/s]
```

In [21]:
```
features = load(open("features.p","rb"))
```

قد تستغرق هذه العملية الكثير من الوقت حسب نظامك. أنا أستخدم وحدة معالجة الرسومات Nvidia 1050 لأغراض التدريب، لذلك استغرق الأمر مني حوالي 7 دقائق لأداء هذه المهمة. ومع ذلك، إذا كنت تستخدم وحدة المعالجة المركزية CPU، فقد تستغرق هذه العملية من ساعة إلى ساعتين. يمكنك التعليق على الكود وتحميل الميزات مباشرة من ملف pickle الخاص بنا.

```
features = load(open("features.p","rb"))
```

الخطوة 4: تحميل مجموعة البيانات لتدريب النموذج.

في مجلد Flickr_8k_test لدينا، لدينا ملف Flickr_8k.trainImages.txt يحتوي على قائمة تضم 6000 صورة سنستخدمها للتدريب.

لتحميل مجموعة بيانات التدريب، نحتاج إلى المزيد من الدوال:

- **load_photos (filename):** سيؤدي هذا إلى تحميل الملف النصيفي سلسلة وإرجاع قائمة أسماء الصور.
- **load_clean_descriptions(filename, photos)** : ستنشئ هذه الدالة قاموسًا يحتوي على تسميات توضيحية لكل صورة من قائمة الصور. نلحق أيضًا المعرّف <start> و<end> لكل تسمية توضيحية. نحتاج إلى هذا حتى يتمكن نموذج LSTM الخاص بنا من تحديد بداية التسمية التوضيحية ونهايتها.
- **load_features (photos):** ستعطينا هذه الدالة قاموسًا لأسماء الصور ومتجه الميزات الذي استخرجناه سابقًا من نموذج Xception.

الكود:

```
#load the data
def load_photos(filename):
    file = load_doc(filename)
    photos = file.split("\n")[:-1]
    return photos


def load_clean_descriptions(filename, photos):
    #loading clean_descriptions
    file = load_doc(filename)
    descriptions = {}
    for line in file.split("\n"):

        words = line.split()
        if len(words)<1 :
            continue

        image, image_caption = words[0], words[1:]

        if image in photos:
            if image not in descriptions:
                descriptions[image] = []
            desc = '<start> ' + " ".join(image_caption) + ' <end>'
            descriptions[image].append(desc)

    return descriptions


def load_features(photos):
    #loading all features
    all_features = load(open("features.p","rb"))
    #selecting only needed features
    features = {k:all_features[k] for k in photos}
    return features
```

```
filename = dataset_text + "/" + "Flickr_8k.trainImages.txt"

#train = loading_data(filename)
train_imgs = load_photos(filename)
train_descriptions = load_clean_descriptions("descriptions.txt",
train_imgs)
train_features = load_features(train_imgs)
```

الخطوة 5: ترميز المفردات.

أجهزة الكمبيوتر لا تفهم الكلمات الإنجليزية، بالنسبة لأجهزة الكمبيوتر، سيتعين علينا تمثيلها بالأرقام. لذلك، سنقوم بتعيين كل كلمة من المفردات بقيمة فهرس فريدة. تزودنا مكتبة Keras بدالة tokenizer التي سنستخدمها لإنشاء الرموز tokens من مفرداتنا وحفظهافي ملف pickle "tokenizer.p".

الكود:

```
#converting dictionary to clean list of descriptions
def dict_to_list(descriptions):
    all_desc = []
    for key in descriptions.keys():
        [all_desc.append(d) for d in descriptions[key]]
    return all_desc

#creating tokenizer class
#this will vectorise text corpus
#each integer will represent token in dictionary

from keras.preprocessing.text import Tokenizer

def create_tokenizer(descriptions):
    desc_list = dict_to_list(descriptions)
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(desc_list)
    return tokenizer

# give each word an index, and store that into tokenizer.p pickle file
tokenizer = create_tokenizer(train_descriptions)
dump(tokenizer, open('tokenizer.p', 'wb'))
vocab_size = len(tokenizer.word_index) + 1
vocab_size
```

مفرداتنا تحتوي على 7577 كلمة.

نحسب الحد الأقصى لطول الأوصاف. هذا مهم لتقرير معلمات هيكل النموذج. أقصى طول للوصف 32.

```
#calculate maximum length of descriptions
def max_length(descriptions):
    desc_list = dict_to_list(descriptions)
    return max(len(d.split()) for d in desc_list)

max_length = max_length(descriptions)
max_length
```

الخطوة 6: إنشاء مولد البيانات.

دعونا نرى أولاً كيف ستبدو مدخلات ومخرجات نموذجنا. لتحويل هذه المهمة إلى مهمة تعليمية خاضعة للإشراف، يتعين علينا توفير المدخلات والمخرجات لنموذج التدريب. يتعين علينا تدريب نموذجنا على 6000 صورة وستحتوي كل صورة على 2048 متجهًا لميزة الطول ويتم تمثيل التسمية التوضيحية أيضًا كأرقام. لا يمكن الاحتفاظ بهذه الكمية من البيانات لـ 6000 صورةفي الذاكرة، لذلك سنستخدم طريقة مولد تنتج دفعات.

سوف ينتج المولد تسلسل الإدخال والإخراج.

فمثلا:

المدخلاتفي نموذجنا هي [x1, x2] والمخرجات ستكون y، حيث x1 هو 2048 متجه الميزة لتلك الصورة، x2 هو تسلسل نص الإدخال وy هو تسلسل نص الإخراج الذي يجب أن يتنبأ به النموذج.

x1(feature vector)	x2(Text sequence)	y(word to predict)
feature	start,	two
feature	start, two	dogs
feature	start, two, dogs	drink
feature	start, two, dogs, drink	water
feature	start, two, dogs, drink, water	end

```
#create input-output sequence pairs from the image description.

#data generator, used by model.fit_generator()
def data_generator(descriptions, features, tokenizer, max_length):
    while 1:
        for key, description_list in descriptions.items():
            #retrieve photo features
            feature = features[key][0]
            input_image, input_sequence, output_word =
create_sequences(tokenizer, max_length, description_list, feature)
            yield [[input_image, input_sequence], output_word]

def create_sequences(tokenizer, max_length, desc_list, feature):
    X1, X2, y = list(), list(), list()
    # walk through each description for the image
    for desc in desc_list:
        # encode the sequence
        seq = tokenizer.texts_to_sequences([desc])[0]
        # split one sequence into multiple X,y pairs
        for i in range(1, len(seq)):
            # split into input and output pair
            in_seq, out_seq = seq[:i], seq[i]
            # pad input sequence
            in_seq = pad_sequences([in_seq], maxlen=max_length)[0]
            # encode output sequence
            out_seq = to_categorical([out_seq],
num_classes=vocab_size)[0]
            # store
            X1.append(feature)
```

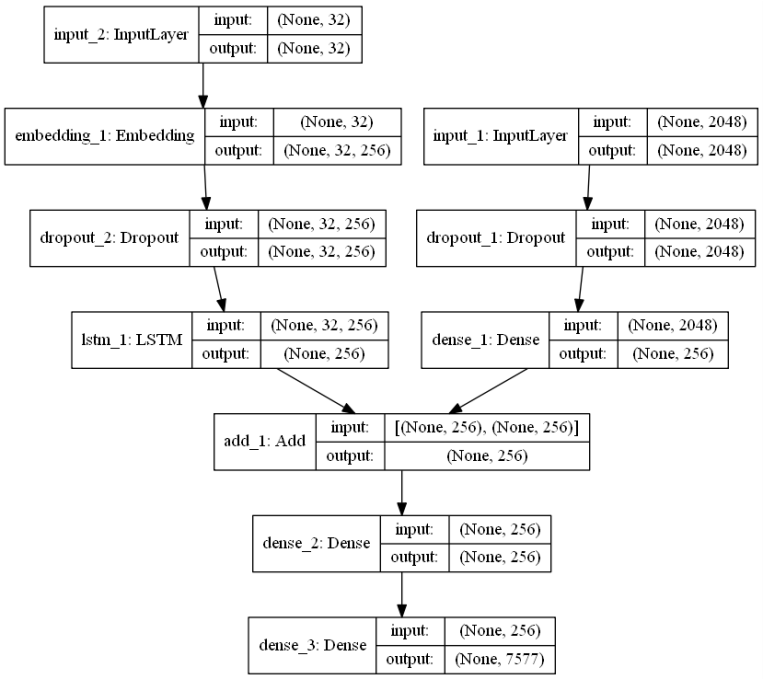
```
            X2.append(in_seq)
            y.append(out_seq)
    return np.array(X1), np.array(X2), np.array(y)

#You can check the shape of the input and output for your model
[a,b],c = next(data_generator(train_descriptions, features, tokenizer,
max_length))
a.shape, b.shape, c.shape
#((47, 2048), (47, 32), (47, 7577))
```

الخطوة 7: تحديد نموذج CNN-RNN.

لتحديد هيكل النموذج، سنستخدم نموذج Keras من Functional API. سيتكون من ثلاثة أجزاء رئيسية:

- **Feature Extractor**: الميزة المستخرجة من الصورة بحجم 2048 ، مع طبقة كثيفة، سنقوم بتقليل الأبعاد إلى 256 عقدة.
- **Sequence Processor**: ستتعامل طبقة التضمين مع الإدخال النصي، متبوعًا بطبقة LSTM.
- **Decoder**: من خلال دمج المخرجات من الطبقتين السابقتين، سنقوم بمعالجة الطبقة الكثيفة لعمل التنبؤ النهائي. ستحتوي الطبقة الأخيرة على عدد من العقد يساوي حجم مفرداتنا.

فيما يلي التمثيل المرئي للنموذج النهائي:




```
from keras.utils import plot_model
```

```
# define the captioning model
def define_model(vocab_size, max_length):

    # features from the CNN model squeezed from 2048 to 256 nodes
    inputs1 = Input(shape=(2048,))
    fe1 = Dropout(0.5)(inputs1)
    fe2 = Dense(256, activation='relu')(fe1)

    # LSTM sequence model
    inputs2 = Input(shape=(max_length,))
    se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
    se2 = Dropout(0.5)(se1)
    se3 = LSTM(256)(se2)

    # Merging both models
    decoder1 = add([fe2, se3])
    decoder2 = Dense(256, activation='relu')(decoder1)
    outputs = Dense(vocab_size, activation='softmax')(decoder2)

    # tie it together [image, seq] [word]
    model = Model(inputs=[inputs1, inputs2], outputs=outputs)
    model.compile(loss='categorical_crossentropy', optimizer='adam')

    # summarize model
    print(model.summary())
    plot_model(model, to_file='model.png', show_shapes=True)

    return model
```

الخطوة 8: تدريب النموذج.

لتدريب النموذج، سنستخدم 6000 صورة تدريبية عن طريق إنشاء تسلسل الإدخال والإخراج على دفعات وتناسبها مع النموذج باستخدام طريقة model.fit_generator(). نقوم أيضًا بحفظ النموذجفي مجلد النماذج الخاص بنا. سيستغرق هذا بعض الوقت حسب قدرة النظام لديك.

```
# train our model
print('Dataset: ', len(train_imgs))
print('Descriptions: train=', len(train_descriptions))
print('Photos: train=', len(train_features))
print('Vocabulary Size:', vocab_size)
print('Description Length: ', max_length)

model = define_model(vocab_size, max_length)
epochs = 10
steps = len(train_descriptions)
# making a directory models to save our models
os.mkdir("models")
for i in range(epochs):
    generator = data_generator(train_descriptions, train_features,
tokenizer, max_length)
    model.fit_generator(generator, epochs=1, steps_per_epoch= steps,
verbose=1)
    model.save("models/model_" + str(i) + ".h5")
```

الخطوة 9: اختبار النموذج.

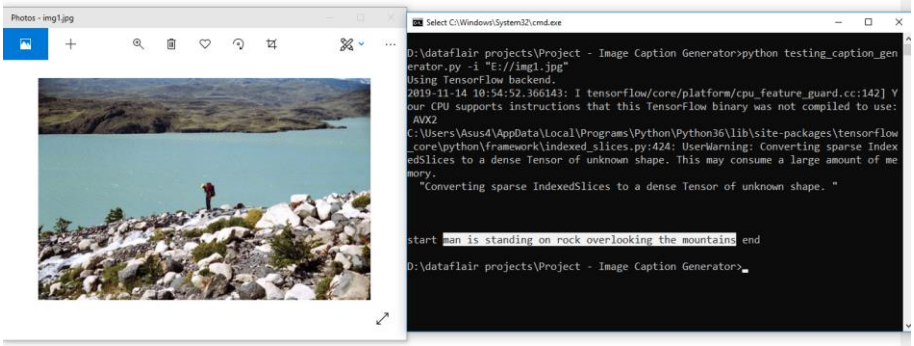
تم تدريب النموذج، الآن، سنقوم بإنشاء ملف منفصل test_caption_generator.py والذي سيحمل النموذج ويولد تنبؤات. تحتوي التنبؤات على الحد الأقصى لطول قيم الفهرس، لذا سنستخدم نفس ملف tokenizer.p pickle للحصول على الكلمات من قيم الفهرس الخاصة بها.

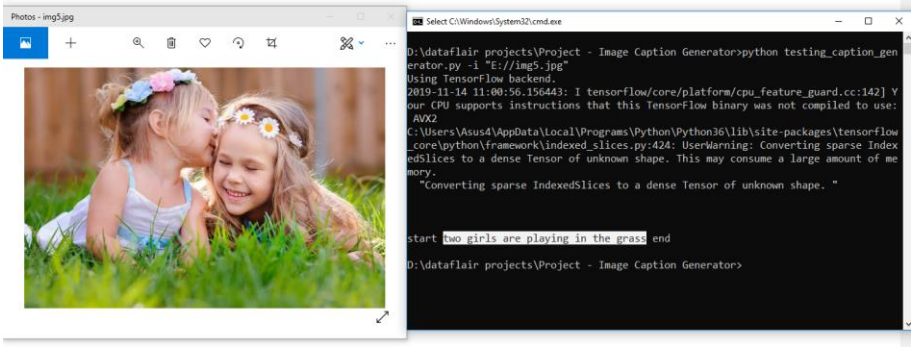
الكود:

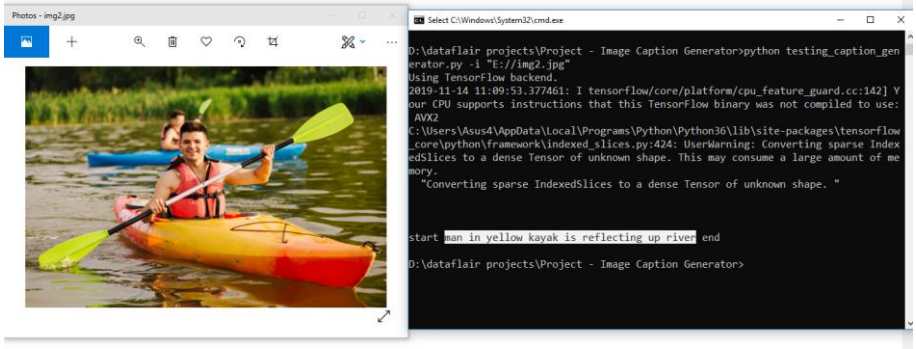
```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import argparse


ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required=True, help="Image Path")
args = vars(ap.parse_args())
img_path = args['image']

def extract_features(filename, model):
        try:
            image = Image.open(filename)

        except:
            print("ERROR: Couldn't open image! Make sure the image
path and extension is correct")
        image = image.resize((299,299))
        image = np.array(image)
        # for images that has 4 channels, we convert them into 3
channels
        if image.shape[2] == 4:
            image = image[..., :3]
        image = np.expand_dims(image, axis=0)
        image = image/127.5
        image = image - 1.0
        feature = model.predict(image)
        return feature

def word_for_id(integer, tokenizer):
for word, index in tokenizer.word_index.items():
     if index == integer:
         return word
return None


def generate_desc(model, tokenizer, photo, max_length):
    in_text = 'start'
    for i in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence], maxlen=max_length)
        pred = model.predict([photo,sequence], verbose=0)
        pred = np.argmax(pred)
        word = word_for_id(pred, tokenizer)
        if word is None:
            break
        in_text += ' ' + word
        if word == 'end':
```

```
            break
    return in_text


#path = 'Flicker8k_Dataset/111537222_07e56d5a30.jpg'
max_length = 32
tokenizer = load(open("tokenizer.p","rb"))
model = load_model('models/model_9.h5')
xception_model = Xception(include_top=False, pooling="avg")

photo = extract_features(img_path, xception_model)
img = Image.open(img_path)

description = generate_desc(model, tokenizer, photo, max_length)
print("\n\n")
print(description)
plt.imshow(img)
```

النتائج:







الملخص

في مشروع بايثون المتقدم هذا، قمنا بتنفيذ نموذج CNN-RNN من خلال إنشاء مولد تسمية توضيحية للصور. بعض النقاط الرئيسية التي يجب ملاحظتها هي أن نموذجنا يعتمد على البيانات، لذلك، لا يمكنه التنبؤ بالكلمات التي خرجت من مفرداته. استخدمنا مجموعة بيانات صغيرة تتكون من 8000 صورة. بالنسبة للنماذج على مستوى الإنتاج، نحتاج إلى التدريب على مجموعات بيانات أكبر من 100000 صورة والتي يمكن أن تنتج نماذج دقة أفضل.

41) الكشف عن مرض باركنسون في المرضى باستخدام إشارات الكلام Detecting Parkinson's disease in patients using speech signals.

مرض باركنسون Parkinson's disease (PD) هو اضطراب تنكسي طويل الأمدفي الجهاز العصبي المركزي. يؤدي إلى الاهتزاز والتصلب وصعوبة المشي والتوازن والتنسيق. تظهر الأعراض ببطء وتتفاقم بمرور الوقت.

إنترنت الأشياء هو حل فعالفي الحالات التي تتطلب المراقبة المستمرة للمرضى.

وكالعادة، أثبت التعلم العميق فعاليتهفي تحليل واستخلاص الاستنتاجات من البيانات الطبية.

في هذه المقالة، سنقوم بتدريب شبكة عصبية لتحليل البيانات الصوتية من المرضى الذين يعانون من مرض باركنسون وبدونه ونصنفهم وفقًا لذلك.

رابط التنفيذ على cAInvas ـ هنا.

استيراد المكتبات الضرورية

```
import pandas as pd
from sklearn.preprocessing import normalize, MinMaxScaler
from sklearn.metrics import confusion_matrix
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
import tensorflow.keras
import random
import matplotlib.pyplot as plt
```

مجموعة البيانات

تتكون مجموعة البيانات هذه من مجموعة من قياسات الصوت الطبية الحيوية من 31 شخصًا، 23 مصابًا بمرض باركنسون (PD). كل عمودفي الجدول هو مقياس صوت معين، وكل صف يتوافق مع واحد من 195 تسجيل صوتي لهؤلاء الأفراد. الهدف الرئيسي من البيانات هو التمييز بين الأشخاص الأصحاء وأولئك الذين يعانون من شلل الرعاش، وفقًا لعمود "الحالة status" الذي تم ضبطه على 0 بالنسبة للصحة و1 من أجل PD.

```
parkinson = pd.read_csv('https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-admin/parkinsons.data')
```

```
parkinson
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	M
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	
...	...	...	...	...	...	...	...	...	...	...
190	phon_R01_S50_2	174.188	230.978	94.261	0.00459	0.00003	0.00263	0.00259	0.00790	
191	phon_R01_S50_3	209.516	253.017	89.488	0.00564	0.00003	0.00331	0.00292	0.00994	
192	phon_R01_S50_4	174.688	240.005	74.287	0.01360	0.00008	0.00624	0.00564	0.01873	
193	phon_R01_S50_5	198.764	396.961	74.904	0.00740	0.00004	0.00370	0.00390	0.01109	
194	phon_R01_S50_6	214.289	260.277	77.973	0.00567	0.00003	0.00295	0.00317	0.00885	

195 rows × 24 columns

يجب تبديل shuffled مجموعة البيانات عشوائيًا منذ أن تم ترتيب عمود الحالة.

```
# shuffling the dataset because the status column is ordered above

parkinson = parkinson.sample(frac=1, random_state=13)
parkinson
```

دعونا نلقي نظرة على انتشار القيم بين الطبقات المختلفة.

```
# looking into the classes

parkinson['status'].value_counts()
```

```
# looking into the classes

parkinson['status'].value_counts()
```

```
1    147
0     48
Name: status, dtype: int64
```

إنها مجموعة بيانات غير متوازنة.

المعالجة المسبقة

تحديد أعمدة الإدخال والإخراج

يستخدم النموذجفي هذه المقالة دالة خطأ الانتروبيا الفئوية وبالتالي يوجد عمودين للإخراج – نعم ولا.

```
input_columns = list(parkinson.columns)
input_columns.remove('status')
input_columns.remove('name')

#output_columns = ['status']    # use for sigmoid activated last layer of
model
```

```
output_columns = ['no', 'yes']    # use for one hot encoded data in the last
Layer

print("Input columns: ", input_columns)
print("Output columns: ", output_columns)
```

MinMaxScaler

دعونا نلقي نظرة على نطاق قيم السمات المختلفة.

```
# The range of values in different attributes vary a lot. Thus, we represent
them on the same scale using MinMaxScaler

scaler = MinMaxScaler()
parkinson[input_columns] = scaler.fit_transform(parkinson[input_columns])
parkinson.describe()
```

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer
count	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000
mean	0.383623	0.193841	0.292748	0.144233	0.146083	0.126513	0.135389	0.126504	0.184126
std	0.240959	0.186761	0.250564	0.154007	0.137636	0.142956	0.147855	0.142934	0.172147
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.170220	0.066786	0.108323	0.056544	0.051383	0.047206	0.050375	0.047279	0.063584
50%	0.351961	0.150411	0.223606	0.103558	0.090909	0.087669	0.094855	0.087494	0.122604
75%	0.549775	0.249162	0.429160	0.180591	0.209486	0.151975	0.162647	0.151951	0.258764
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 23 columns

تقع القيم الآنفي النطاق [0، 1].

One hot encoding

يستخدم النموذج الانتروبيا الفئوية categorical cross-entropy، وبالتالي فإن التسميات مشفرة واحدة ساخنة.

نموذج لترميز واحد ساخن: قيمة عدد صحيح 1 ⟶ [0 ، 1] ، 0 ⟶ [1 ، 0].

يمكننا الاحتفاظ بالتسميات كعدد صحيح واحد (1/0) إذا كنا نستخدم خطأ ثنائية عبر الانتروبيا cross entropy loss.في هذه الحالة، يجب أن تحتوي الطبقة الأخيرة من نموذجنا على عقدة واحدة.

```
# one hot encoding the output columns

parkinson[['no', 'yes']] = pd.get_dummies(parkinson.status)
```

```
parkinson
```

MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	HNR	status	RPDE	DFA	spread1	spread2	D2	PPE	no	yes
0.126686	0.128617	0.126826	0.205222	...	0.545964	1	0.906209	0.818028	0.515241	0.547523	0.264458	0.446279	0	1
0.093931	0.089496	0.094076	0.286014	...	0.450134	0	0.447684	0.333127	0.257894	0.260408	0.549049	0.183318	1	0
0.048651	0.048232	0.048643	0.072850	...	0.580956	0	0.113145	0.318279	0.118322	0.207947	0.441997	0.104578	1	0
0.105491	0.117899	0.105635	0.507303	...	0.222751	1	0.808025	0.663550	0.442946	0.660204	0.762172	0.365408	0	1
0.403179	0.437835	0.403275	0.476173	...	0.268999	1	0.894202	0.275073	0.680218	0.684097	0.540509	0.646901	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.365125	0.301715	0.365067	0.340971	...	0.284158	1	0.733677	0.464571	0.507293	0.440936	0.453018	0.447025	0	1
0.068882	0.107181	0.068711	0.099781	...	0.564578	1	0.725207	0.280511	0.456512	0.525619	0.374227	0.359600	0	1
0.092486	0.102358	0.092471	0.135841	...	0.690766	1	0.421066	0.962630	0.479415	0.440294	0.357017	0.414170	0	1
0.048651	0.064845	0.048643	0.070385	...	0.719418	0	0.244206	0.752811	0.352217	0.231827	0.258659	0.269019	1	0
0.136320	0.137192	0.136298	0.134471	...	0.491547	1	0.695411	0.851031	0.532041	0.316677	0.447099	0.455912	0	1

تقسيم التدريب - اختبار

استخدام تقسيم 80–10–10 لتقسيم مجموعة البيانات إلى مجموعة تدريب وتحقق من الصحة ومجموعة اختبار.

```
# Using 80-10-10 split of train-val-test data

train_df, val_df = train_test_split(parkinson, train_size=0.8)    # 80-20
split
val_df, test_df = train_test_split(val_df, train_size = 0.5)    #splitting
the 20% into 2 halves

print("Train dataset")
print(len(train_df))
print(train_df['status'].value_counts())

print("Val dataset")
print(len(val_df))
print(val_df['status'].value_counts())

print("Test dataset")
print(len(test_df))
print(test_df['status'].value_counts())
```

```
Train dataset
156
1    118
0     38
Name: status, dtype: int64
Val dataset
19
1    16
0     3
Name: status, dtype: int64
Test dataset
20
1    13
0     7
Name: status, dtype: int64
```

بناء النموذج

يحتوي النموذج على 4 طبقات كثيفة dense layers (بمافي ذلك طبقة الإدخال). تحتوي الطبقة النهائية على عقدتين وتنشيط softmax.

يتم استخدام تنشيط softmax جنبًا إلى جنب مع خطأ الانتروبيا الفئوية عندما يكون ناتج النموذج مشفرًا واحدًا ساخنًا.في حالة مخرجات الأعداد الصحيحة، يتم استخدام دالة التنشيط sigmoid مع خطأ الانتروبيا الثنائي وعقدة واحدةفي الطبقة الأخيرة.

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(256, activation="relu", input_shape =
xtrain[0].shape))
model.add(tf.keras.layers.Dense(128, activation="relu"))
model.add(tf.keras.layers.Dense(64, activation="relu"))
model.add(tf.keras.layers.Dense(2, activation="softmax"))
```

```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics
= ['accuracy'])

history = model.fit(xtrain, ytrain, epochs = 32, batch_size = 8,
validation_data=(xval, yval))
```

```
20/20 [==============================] - 0s 12ms/step - loss: 0.0313 - accuracy: 1.0000 - val_loss: 0.0633 - val_accuracy: 0.9474
Epoch 30/32
20/20 [==============================] - 0s 12ms/step - loss: 0.0567 - accuracy: 0.9808 - val_loss: 0.0456 - val_accuracy: 1.0000
Epoch 31/32
20/20 [==============================] - 0s 2ms/step - loss: 0.1830 - accuracy: 0.9231 - val_loss: 0.2134 - val_accuracy: 0.8947
Epoch 32/32
20/20 [==============================] - 0s 4ms/step - loss: 0.1652 - accuracy: 0.9359 - val_loss: 0.0760 - val_accuracy: 0.9474
```

كان النموذج المدرب قادرًا على تحقيق دقة تصل إلى 95٪.

```
model.evaluate(xtest, ytest)
model.summary()
```

```
1/1 [==============================] - 0s 988us/step - loss: 0.4036 - accuracy: 0.9000
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               5888
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130
=================================================================
Total params: 47,170
Trainable params: 47,170
Non-trainable params: 0
```

على الرغم من عدم التوازنفي مجموعة البيانات، تظهر مصفوفة الارتباك نتائج واعدة.

```
ypred = model.predict_classes(xtest)

confusion_matrix(np.argmax(ytest, axis = 1), ypred)
```

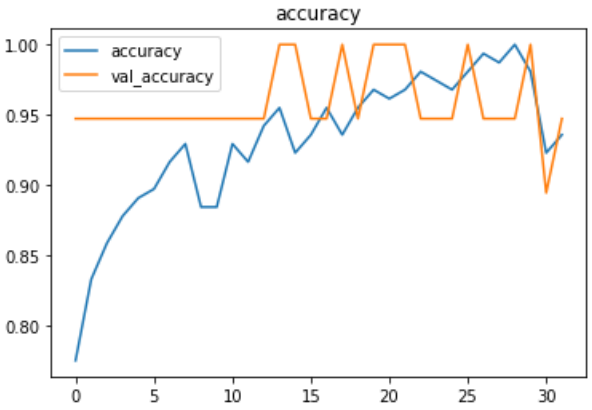
```
array([[ 6,  1],
       [ 0, 13]])
```

تشير الصفوف إلى القيم الفعلية وتشير الأعمدة إلى القيم المتوقعة. تشير المصفوفة إلى وجود إيجابي خاطئ واحد وفي هذه الحالة، تكون النتيجة الإيجابية الخاطئة أفضل من السلبية الخاطئة.

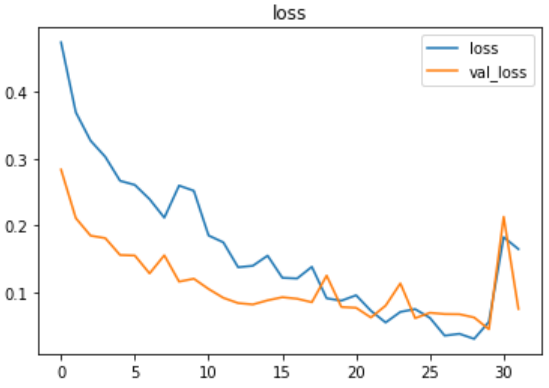
رسم المقاييس

```
def plot(history, variable, variable1):
    plt.plot(range(len(history[variable])), history[variable])
    plt.plot(range(len(history[variable1])), history[variable1])
    plt.legend([variable, variable1])
    plt.title(variable)
```

```
plot(history.history, "accuracy", "val_accuracy")
```




```
plot(history.history, "loss", "val_loss")
```



التنبؤ

إجراء تنبؤات على عينات اختبار عشوائية.

```
# pick random test data sample from one batch
x = random.randint(0, len(xtest)- 1)
```

```
pred = model.predict(xtest[x].reshape(1, -1))
diagnosis = np.argmax(pred[0])

print("Actual diagnosis: ", output_columns[np.argmax(ytest[x])])
print("Model diagnosis: ", output_columns[diagnosis], " with probability ",
pred[0][diagnosis])
```

```
Actual diagnosis:  no
Model diagnosis:  no with probability  0.9623155
```

42) كشف الارتباك مع إشارات EEG باستخدام التعلم العميق
Confusion detection with EEG signals Using Deep Learning

اكتشاف ما إذا كان الشخص مرتبكًا أم لا بناءً على تسجيلات EEG.

هل يسأل الطلاب دائمًا عن شكوك عندما يكونونفي حيرة من أمرهم؟ كيف تعرف إذا كان شخص ما مرتبكًا؟ ربما تعابير الوجه. تحدث الارتباكات عندما لا نكون قادرين على فهم ما نراه / نسمعه.

EEG ، التي تعني تخطيط كهربية الدماغ ، هي طريقة لتسجيل النشاط الكهربائي للدماغ باستخدام المراقبة الكهربية.

يتم ذلك من خلال أقطاب كهربائية غير جراحية (في معظم الحالات) موضوعة على طول فروة الرأس تسجل النشاط الكهربائي التلقائي للدماغ على مدار فترة زمنية.

هنا، نتتبع إشارات EEG هذه لاكتشاف ما إذا كان الشخص مرتبكًا أم لا.

تنفيذ الكود على cAInvas ـ هنا!

استيراد المكتبات الضرورية

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, models, optimizers, losses, callbacks
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
import matplotlib.pyplot as plt
import random
```

مجموعة البيانات

تم جمع بيانات إشارة EEG من 10 طلاب جامعيين أثناء مشاهدة مقاطع فيديو MOOC لموضوعات تتراوح من المواد البسيطة مثل الجبر الأساسي أو الهندسة إلى أبحاث الخلايا الجذعية وميكانيكا الكم التي يمكن أن تكون مربكة إذا لم نكن على دراية بالموضوع. كان هناك 20 مقطع فيديو، 10 مقاطع بسيطة و10 مقاطع فيديو معقدة، مدة كل منها دقيقتان. تم قص المقاطعفي منتصف الموضوع لجعله أكثر إرباكًا.

ارتدى الطلاب مجموعة MindSet لاسلكية أحادية القناة تقيس النشاط فوق الفص الأمامي. يقيس جهاز MindSet الجهد الكهربي بين قطب كهربائي يستقر على الجبهة وقطبين (أرضي ومرجع واحد) كل منهما على اتصال بأذن.

يوجد عمودين للتسمية: تسمية معرّفة من قبل المستخدم user-defined label (تم تسميتها ذاتيًا من قبل الطلاب بناءً على خبرتهم) والتسمية المحددة مسبقًا predefined label (حيث من المتوقع أن يتم الخلط بينهم).

```
df = pd.read_csv('https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-admin/EEG_data.csv')
df
```

	SubjectID	VideoID	Attention	Mediation	Raw	Delta	Theta	Alpha1	Alpha2	Beta1	Beta2	Gamma1	Gamma2
0	0.0	0.0	56.0	43.0	278.0	301963.0	90612.0	33735.0	23991.0	27946.0	45097.0	33228.0	8293.0
1	0.0	0.0	40.0	35.0	-50.0	73787.0	28083.0	1439.0	2240.0	2746.0	3687.0	5293.0	2740.0
2	0.0	0.0	47.0	48.0	101.0	758353.0	383745.0	201999.0	62107.0	36293.0	130536.0	57243.0	25354.0
3	0.0	0.0	47.0	57.0	-5.0	2012240.0	129350.0	61236.0	17084.0	11488.0	62462.0	49960.0	33932.0
4	0.0	0.0	44.0	53.0	-8.0	1005145.0	354328.0	37102.0	88881.0	45307.0	99603.0	44790.0	29749.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
12806	9.0	9.0	64.0	38.0	-39.0	127574.0	9951.0	709.0	21732.0	3872.0	39728.0	2598.0	960.0
12807	9.0	9.0	61.0	35.0	-275.0	323061.0	797464.0	153171.0	145805.0	39829.0	571280.0	36574.0	10010.0
12808	9.0	9.0	60.0	29.0	-426.0	680989.0	154296.0	40068.0	39122.0	10966.0	26975.0	20427.0	2024.0
12809	9.0	9.0	60.0	29.0	-84.0	366269.0	27346.0	11444.0	9932.0	1939.0	3283.0	12323.0	1764.0
12810	9.0	9.0	64.0	29.0	-49.0	1164555.0	1184366.0	50014.0	124208.0	10634.0	445383.0	22133.0	4482.0

12811 rows × 15 columns

المعالجة المسبقة

إطار البيانات المستندة إلى الوقت

نظرًا لأن هذه مجموعة بيانات تستند إلى الوقت، يتم إلحاق الميزات لتضمين قيم الخطوات الزمنية السابقة لنفس الموضوع الذي يشاهد نفس الفيديو.

يتم تحديد نافذة زمنية من 5، أي يتم دمج قيم السمات من 5 خطوات زمنية لإنشاء صف واحد من مجموعة البيانات النهائية. إذا كانت المجموعة الفرعية لإطار بيانات الموضوع x الفيديو تحتوي على أختام زمنية أقل من النافذة الزمنية المحددة، فسيتم تجاهلها.

```
# Defining the time window, that is, how many timesteps to include
time_window = 5

# Dataframes that hold rows grouped by subject
df_subject_grouped = df.groupby('SubjectID')

# Column values affected by time
time_affected_columns = list(df.columns)
time_affected_columns.remove('SubjectID')
time_affected_columns.remove('VideoID')
time_affected_columns.remove('predefinedlabel')
time_affected_columns.remove('user-definedlabeln')


# Final dataframe
df_final = pd.DataFrame()

# For each subject
for subject in df_subject_grouped:
    # For each video:
    for video in subject[1].groupby('VideoID'):
        # If the df has timesteps greater than or equal to the time window,
else discard
        if time_window <= len(video[1]):
            # Skipping time_window-1 rows from the beginning, and looping to
till the end
```

```
            for row_num in range(time_window, len(video[1])+1):
                # picking the time_window th row
                df_temp = video[1].iloc[row_num-1, :]
                # Appending values from time_window-1 rows before that
                for i in range(time_window-1):
                    df_temp_i = video[1].iloc[row_num-1-
i][time_affected_columns]    # Pick necessary columns
                    df_temp = pd.concat([df_temp, df_temp_i], axis = 0)    #
Append values

                df_temp = df_temp.to_frame().transpose()    # Series to
DataFrame

                df_final = pd.concat([df_final, df_temp])    # Add as row to
final dataframe

# Reset index
df_final = df_final.reset_index(drop = True)
```

حذف الأعمدة غير المرغوب فيها

يجب ألا يؤثر معرف الموضوع SubjectID ومعرف الفيديو VideoID على النتائج النهائية وبالتالي تتم إزالتهما. تعتبر التسميات التي يحددها المستخدم أكثر موثوقيةفي تقييم مستوى الارتباك بدلاً من التسميات المحددة مسبقًا.

```
df = df_final.drop(columns = ['SubjectID', 'VideoID', 'predefinedlabel'])
```

```
df['user-definedlabeln'].value_counts()
```

```
1.0    6363
0.0    6048
Name: user-definedlabeln, dtype: int64
```

هذه مجموعة بيانات متوازنة تقريبًا.

تقسيم التدريب والتحقق من الصحة والاختبار

تقسيمها إلى مجموعة تدريب، والتحقق من الصحة، والاختبار باستخدام نسبة انقسام 80ـ10ـ 10. ثم يتم تقسيمها إلى X (input) وy (target) من الفئات المعنية لمزيد من المعالجة.

```
# Splitting into train, val and test set -- 80-10-10 split

# First, an 80-20 split
train_df, val_test_df = train_test_split(df, test_size = 0.2, random_state =
113)

# Then split the 20% into half
val_df, test_df = train_test_split(val_test_df, test_size = 0.5, random_state
= 113)

len(train_df), len(val_df), len(test_df)
```

```
(9928, 1241, 1242)
```

```
ic = df.columns.tolist()
ic.remove('user-definedlabeln')

oc = ['user-definedlabeln']

ytrain = train_df[oc]
Xtrain = train_df.drop(columns = oc)

yval = val_df[oc]
Xval = val_df.drop(columns = oc)

ytest = test_df[oc]
Xtest = test_df.drop(columns = oc)
```

التوحيد Standardization

تحجيم القيم ليكون متوسط = 0 والانحراف المعياري = 1.

الانحراف المعياري لقيم السماتفي مجموعة البيانات ليس هو نفسهفي كل منهم. قد يؤدي هذا إلى ترجيح سمات معينة أعلى من غيرها. يتم قياس القيم الموجودةفي جميع السمات بحيث يكون متوسطها = 0 والانحراف المعياري = 1 فيما يتعلق بأعمدة معينة.

يتم استخدام دالة StandardScalerفي وحدة sklearn.preprocessing لتنفيذ هذا المفهوم. يتلاءم المثيل أولاً مع بيانات التدريب ويستخدم لتحويل بيانات التدريب والتحقق من الصحة والاختبار.

```
ss = StandardScaler()

Xtrain = ss.fit_transform(Xtrain)
Xval = ss.transform(Xval)
Xtest = ss.transform(Xtest)
```

بناء النموذج

النموذج بسيط يحتوي على 3 طبقات كثيفة، اثنتان منها لها دوال تنشيط ReLU والأخيرة لها دالة تنشيط Sigmoid التي تنتج قيمةفي النطاق [0، 1]. يتداخل هذا مع طبقة تسرب Dropout layer واحدة مع احتمال الاحتفاظ 0.2.

نظرًا لأنها مشكلة تصنيف ثنائي، يتم تجميع النموذج باستخدام دالة خطأ الانتروبيا الثنائية. يتم استخدام مُحسِّن آدم ويتم تتبع دقة النموذج على مر الحقب.

دالة رد الاتصال EarlyStopping للوحدة keras.callbacks تراقب خطأ التحقق وتوقف التدريب إذا لم ينقص لمدة 5 فترات بشكل مستمر. تضمن معلمة rest_best_weights استعادة النموذج الذي لديه أقل خطأفي التحقق من الصحة إلى متغير النموذج.

```
model = models.Sequential([
    layers.Dense(32, activation = 'relu', input_shape = Xtrain[0].shape),
    layers.Dropout(0.2),
    layers.Dense(16, activation = 'relu'),
    layers.Dense(1, activation = 'sigmoid')
])

cb = callbacks.EarlyStopping(patience = 5, restore_best_weights = True)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 32)                1792
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 16)                528
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17
=================================================================
Total params: 2,337
Trainable params: 2,337
Non-trainable params: 0
_________________________________________________________________
```

تجميع وتدريب النموذج

```
model.compile(optimizer = optimizers.Adam(0.01), loss =
losses.BinaryCrossentropy(), metrics = ['accuracy'])

history = model.fit(Xtrain, ytrain, validation_data = (Xval, yval), epochs =
256, callbacks = cb)
```

```
Epoch 1/256
311/311 [==============================] - 1s 2ms/step - loss: 0.6571 - accuracy: 0.6284 - val_loss: 0.6535 - val_accuracy: 0.6350
Epoch 2/256
311/311 [==============================] - 0s 1ms/step - loss: 0.6353 - accuracy: 0.6533 - val_loss: 0.6382 - val_accuracy: 0.6342
Epoch 3/256
311/311 [==============================] - 0s 1ms/step - loss: 0.6299 - accuracy: 0.6545 - val_loss: 0.6421 - val_accuracy: 0.6479
Epoch 4/256
311/311 [==============================] - 0s 1ms/step - loss: 0.6258 - accuracy: 0.6618 - val_loss: 0.6397 - val_accuracy: 0.6430
Epoch 5/256
311/311 [==============================] - 0s 2ms/step - loss: 0.6230 - accuracy: 0.6626 - val_loss: 0.6305 - val_accuracy: 0.6591
Epoch 6/256
311/311 [==============================] - 0s 1ms/step - loss: 0.6192 - accuracy: 0.6649 - val_loss: 0.6365 - val_accuracy: 0.6527
Epoch 7/256
311/311 [==============================] - 0s 1ms/step - loss: 0.6164 - accuracy: 0.6657 - val_loss: 0.6387 - val_accuracy: 0.6471
Epoch 8/256
311/311 [==============================] - 0s 1ms/step - loss: 0.6148 - accuracy: 0.6678 - val_loss: 0.6228 - val_accuracy: 0.6519
Epoch 9/256
311/311 [==============================] - 0s 2ms/step - loss: 0.6114 - accuracy: 0.6671 - val_loss: 0.6404 - val_accuracy: 0.6398
Epoch 10/256
311/311 [==============================] - 0s 1ms/step - loss: 0.6059 - accuracy: 0.6769 - val_loss: 0.6322 - val_accuracy: 0.6463
Epoch 11/256
311/311 [==============================] - 0s 1ms/step - loss: 0.6044 - accuracy: 0.6724 - val_loss: 0.6254 - val_accuracy: 0.6575
Epoch 12/256
311/311 [==============================] - 0s 1ms/step - loss: 0.6027 - accuracy: 0.6753 - val_loss: 0.6250 - val_accuracy: 0.6680
Epoch 13/256
311/311 [==============================] - 0s 2ms/step - loss: 0.6019 - accuracy: 0.6724 - val loss: 0.6282 - val accuracy: 0.6535
```

تقييم النموذج

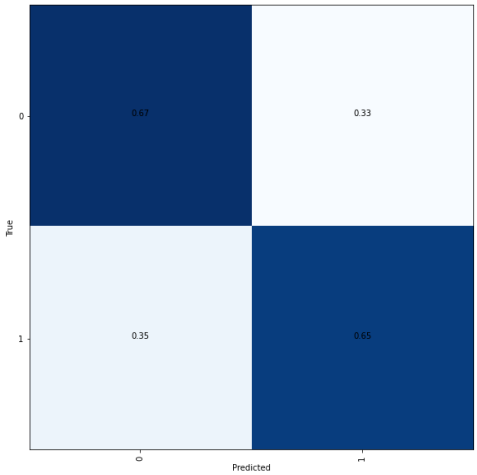
```
model.evaluate(Xtest, ytest)
```

```
39/39 [==============================] - 0s 1ms/step - loss: 0.6206 -
accuracy: 0.6586
```

تم تدريب النموذج بمعدل تعلم 0.01 ودقة ~ 65 ٪ على مجموعة الاختبار.

مصفوفة الارتباك

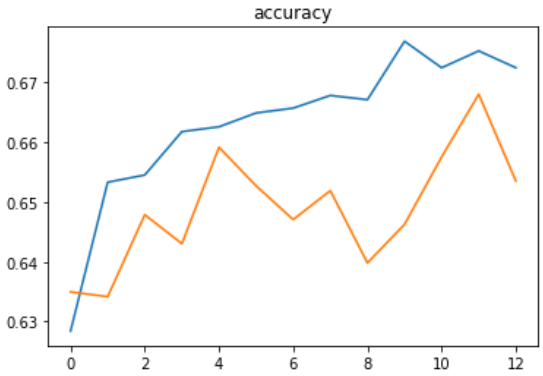
رسم مصفوفة الارتباك لفهم النتائج بشكل أفضل.



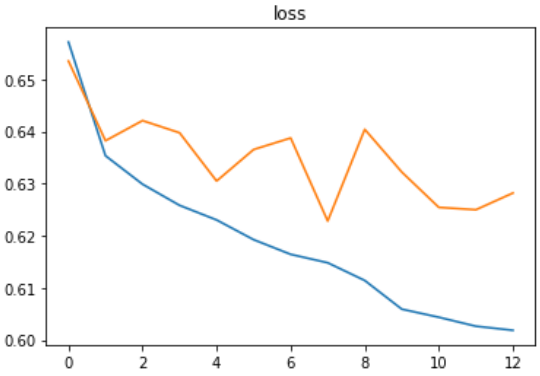
يمكن زيادة معدل الدقة المنخفض ببيانات ذات تصنيف أفضل. من السهل وصف البيانات الذاتية التي تشير إلى الحالة العقلية بأنها خاطئة mislabelled.

رسم المقاييس

```
def plot(history, variable, variable2):
    plt.plot(range(len(history[variable])), history[variable])
    plt.plot(range(len(history[variable2])), history[variable2])
    plt.title(variable)
plot(history.history, "accuracy", 'val_accuracy')
```




```
plot(history.history, "loss", "val_loss")
```



التنبؤ

لننفذ تنبؤات على عينات بيانات الاختبار العشوائية:

```
# pick random test data sample from one batch
x = random.randint(0, len(Xtest) - 1)

output = model.predict(Xtest[x].reshape(1, -1))[0][0]
pred = (output>0.5).astype('int')
print("Predicted: ", pred, "(", output, "-->", pred, ")")

print("True: ", np.array(ytest)[x][0])
```

```
Predicted:  1 ( 0.5476615 --> 1 )
True:  1.0
```

43) التعرف على الرقم المنطوق باستخدام التعلم العميق Recognizing the spoken digit using Deep Learning

نموذج يستخدم صور مخطط ميل الطيفي Mel spectrogram لعينات الصوت للتعرف على الرقم المنطوق، والذي يمكن توسيعه بسهولة إلى أوامر مكونة من كلمة واحدة.

التواصل هو جانب مهم من حياة الإنسان. نظرًا لأن التكنولوجيا أصبحت الآن جزءًا لا يتجزأ من حياتنا، فإننا نميل إلى التركيز على الطرق المختلفة التي يمكننا من خلالها توصيل أفكارنا ونوايانا للأجهزة التي تعمل بالتكنولوجيا من حولنا.

الكلام هو إحدى الطرق التي يتحدث بها البشر مع بعضهم البعض. إذا قمنا بتقسيم عملية فهم ما يتم التحدث به إلينا، فسيكون الجزء الأول هو التعرف على الكلمات. من منظور الآلة، سيكون هذا المقطع نسخًا.

في هذه المقالة، سننظرفي التعرف على تسجيلات الكلام لمدة ثانية واحدة للأرقام من 0 إلى 9.

تطبيق مفكرة cAInvas لهذا المفهوم موجود هنا.

استيراد المكتبات الضرورية

```
import torch
from torch.utils.data import Dataset, random_split, DataLoader, TensorDataset
import torchvision
from torchvision.datasets.utils import download_url
import torch.nn as nn
import torch.nn.functional as F
import os
import librosa
import pandas as pd
import numpy as np
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import librosa.display
import sklearn
import matplotlib
import csv
from PIL import Image
from sklearn.metrics import f1_score
import IPython.display as ipd
import random
```

مجموعة البيانات

مجموعة البيانات الصوتية المستخدمة هنا هي مجموعة فرعية من مجموعة بيانات أوامر الكلام Tensorflow.

كل عينة عبارة عن صوت أحادي مدته ثانية واحدة مسجلة بتردد 8000 هرتز.

مجموعة البيانات متوازنة مع حوالي 2360 عينةفي كل فئة.

تحتوي الخلية أدناه على ملف مضغوط لمجموعة البيانات والملفات الأخرى المطلوبةفي النوتبوك هذا.

```
!wget -N https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-admin/spoken_digit_melimages.zip
!unzip -qo spoken_digit_melimages.zip
digit = ['zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

```
# Random sample selection
digit_audio_sample = ['dataset_sample_0.wav', 'dataset_sample_1.wav', 'dataset_sample_2.wav', 'dataset_sample_3.wav', 'dataset_sample_4.wav', 'dataset_sample_5.wav', 'dataset_sample_6.wav', 'dataset_sample_7.wav', 'dataset_sample_8.wav', 'dataset_sample_9.wav']
i = random.randint(0, 9)

# Selecting the sample by label
#i = 4    # Label = 4

data, sr = librosa.load('melimages/'+digit_audio_sample[i])

fig = plt.figure()
# Two subplots
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)

print("Label: ", i, '\n')
print("Sampling rate: ", sr, "\n")

# mel spectogram
S = librosa.feature.melspectrogram(y=data, sr=sr)
librosa.display.specshow(librosa.power_to_db(S, ref=np.max), x_axis='time', y_axis='mel', ax = ax1)
# waveplot
librosa.display.waveplot(np.array(data), sr=sr, ax = ax2)

ipd.Audio(data = data, rate = sr)
```

```
Label:  8

Sampling rate:  22050
```

استخراج الميزات

هناك العديد من الطرق لتمثيل البيانات الصوتية، مثل الموجي waveform وMFCCs و Mel Spectrograms وSpectrograms وغيرها الكثير.

من بينها جميعًا، يعد مقياس Mel تمثيلًا أقرب لإدراك الصوت البشري من المقياس القياسي. وبالتالي، يتم استخراج صور مخطط ميل الطيفي للتسجيلات الصوتية لاستخدامها كمدخلاتفي النموذج.

لا حاجة لتنفيذ التعليمات البرمجية المعلقة التالية، الملفات المستخرجة المضمنةفي مجموعة البيانات.

يتطلب تنفيذ خلايا استخراج البيانات أدناه تنزيل مجموعة بيانات أوامر الكلام tensorflow (حوالي 1 غيغابايت).

رابط لتنزيل مجموعة البيانات الصوتية:

http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz

إنشاء ملف المرجع CSV.

```
'''

# Creating a reference csv file

with open("Spoken_digit.csv", 'w') as csvfile:
    csvwriter = csv.writer(csvfile)
    csvwriter.writerow(["File", "Label"])
    for x in digit:
        if os.path.isdir('/content/data/'+x):
            for name in os.listdir('/content/data/'+x):
                if os.path.isfile('/content/data/'+x+"/"+name):
                    csvwriter.writerow([x+'/'+name, x])

# shuffle
df = pd.read_csv('Spoken_digit.csv')
df = df.sample(frac=1)
df.to_csv('Spoken_digit.csv', index = False)

'''
```

استخلاص صور المخطط الطيفي من العينات الصوتية

```
'''

# -- Extraction can take about 8 hours

# Making folders to store the extracted images

path = ''
os.makedirs(path+'/melimages')
for i in digit:
    os.makedirs(path+'/melimages/'+i)
    os.makedirs(path+'/melimages100/'+i)
    os.makedirs(path+'/melimages200/'+i)

for i in range(0, len(sp)):
    print(i)    # Print to keep track of which file is being processed

    f = sp.loc[i]

    name = f.File
    data, sr = librosa.load('/content/data/'+name)

    fig = plt.figure(figsize=[1,1])
```

```
    ax = fig.add_subplot(111)
    ax.axes.get_xaxis().set_visible(False)
    ax.axes.get_yaxis().set_visible(False)
    ax.set_frame_on(False)

    S = librosa.feature.melspectrogram(y=data, sr=sr)
    librosa.display.specshow(librosa.power_to_db(S, ref=np.max),
x_axis='time', y_axis='mel')

    file  = path+'/melimages/'+str(name[:-4]) + '.jpg'
    plt.savefig(file, bbox_inches='tight',pad_inches=0)

    plt.close()

'''
```

ملحوظة

تتوفر صور mel المستخرجة وملف csv مباشرة كمجموعة بيانات على رابط التنزيل في بداية النوتبوك.

المجلد ـ Melimages

يحتوي مجلد مجموعة البيانات على:

- 10 مجلدات، واحد لكل رقم. تحتوي هذه المجلدات على صور طيفية لعينات صوتية مقابلة.
- 10 عينات، واحدة لكل رقم، من مجموعة البيانات كما هو موضح أعلاه. يساعد هذا مستخدم النوتبوك على فهم مجموعة البيانات بشكل أفضل حيث لا يتم تضمين العينات الأصلية هنا.
- سجل 4 مستخدمين عينات خارجية غير موجودة في مجموعة البيانات (train أو val).

Spoken_digit.csv: ملف csv يتكون من عمودين:

- ملف File: اسم الملف / مسار الملف في مجموعة البيانات.
- التسمية Label: تسمية العينة.

```
# Spoken_digit.csv

spokendigit = pd.read_csv('https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-admin/Spoken_digit.csv')
spokendigit
```

يجب تنفيذ التعليمات البرمجية لاستخراجها مرة واحدة فقط وينشئ الناتج التالي:

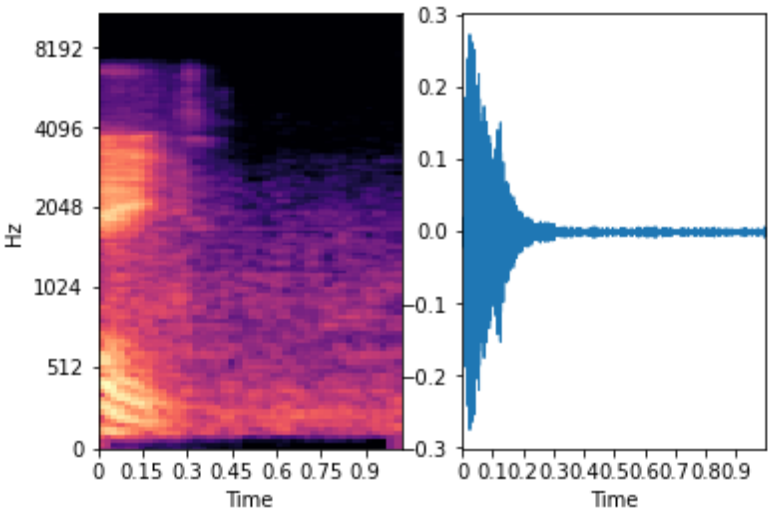
- ملف مرجعي CSV

	File	Label
0	eight/e32ff49d_nohash_0.wav	eight
1	five/840eab5a_nohash_0.wav	five
2	six/541e4079_nohash_0.wav	six
3	one/b9cccd01_nohash_0.wav	one
4	seven/28e47b1a_nohash_2.wav	seven
...	...	...
23661	four/d33df435_nohash_0.wav	four
23662	two/1365dd89_nohash_0.wav	two
23663	zero/471a0925_nohash_0.wav	zero
23664	two/11b1df78_nohash_1.wav	two
23665	three/541e4079_nohash_2.wav	three

23666 rows × 2 columns

عينة من صفوف ملف csv

- صور مخطط ميل الطيفي لجميع العينات الصوتية.



صورة المخطط الطيفي للميل | شكل موجة العينة

المدخلات في النموذج

يتم تقسيم مجموعة بيانات الصور إلى مجموعات تدريب والتحقق من الصحة بنسبة 90ـ10. تحتوي مجموعة التدريب على 21300 صورة ومجموعة التحقق من الصحة بها 2366.

```
# train - val split of 90-10

size = len(meldset)
val_size = int(0.1 * size)
train_size = size - val_size

train_dset, val_dset = random_split(meldset, [train_size, val_size])

print("Number of samples in train set: ", train_size)
print("Number of samples in validation set: ", val_size)
```

```
Number of samples in train set:  21300
Number of samples in validation set:  2366
```

كل صورة بحجم x5554.

يتم وصف كلاس مجموعة البيانات على النحو التالي:

```
class SpokenDigit(Dataset):
    def __init__(self, file = None, rootdir = None, transform = None):
        self.df = pd.read_csv(file)
        self.rootdir = rootdir    # root directory for the images
        self.transform = transform

    def __len__(self):
        return len(self.df)

    def __getitem__(self, i):
        row = self.df.loc[i]
        fname, label = row['File'], row['Label']
        ik = self.rootdir+fname[:-4]+'.jpg'
        img = Image.open(ik)
        if self.transform:
            img = self.transform(img)
        return img, torch.tensor(digit.index(label))    # return image tensor
and numeric label

    def getsr(self, i):
        row = self.df.loc[i]
        fname, label = row['File'], row['Label']
        _, sr = librosa.load(self.rootdir+fname)
        return sr
```

```
meldset = SpokenDigit('https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-admin/Spoken_digit.csv',
'melimages/', transforms.Compose([transforms.ToTensor()]))
```

بناء النموذج

يحتوي النموذج على ست طبقات متتالية Convolution2D-Maxpooling2D، تليها طبقة تسطيح flattening layer. يتبع ذلك ثلاث طبقات كثيفة dense layers حيث تحتوي الطبقة الأخيرة على عشر عقد مع دالة تنشيط softmax.

```
class SpokenDigitModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.network = nn.Sequential(
            nn.Conv2d(3, 16, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),

            nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),

            nn.Conv2d(128, 128, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(128, 256, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.AdaptiveAvgPool2d(1),

            nn.Flatten(),
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 10),
            nn.Softmax(dim = 1)
        )

    def forward(self, x):
        return self.network(x)

    def training_step(self, batch):
        inputs, labels = batch
        outputs = self(inputs)
        loss = F.cross_entropy(outputs, labels)    # cros entropy loss
        return loss

    def validation_step(self, batch):
        inputs, labels = batch
        outputs = self(inputs)
        loss = F.cross_entropy(outputs, labels)
        _, pred = torch.max(outputs, 1)
        accuracy = torch.tensor(torch.sum(pred==labels).item()/len(pred))
# calculate accuracy
        return [loss.detach(), accuracy.detach()]
```

```
def evaluate(model, loader):
    model.eval()
    outputs = [model.validation_step(batch) for batch in loader]
    outputs = torch.tensor(outputs).T
    loss, accuracy = torch.mean(outputs, dim=1)
    return {"loss" : loss.item(), "accuracy" : accuracy.item()}
```

```
def fit(model, train_loader, val_loader, epochs, lr, optimizer_function =
torch.optim.Adam):
    history = []
    optimizer = optimizer_function(model.parameters(), lr)

    for epoch in range(epochs):
        print("Epoch ", epoch)
        #Train
        model.train()

        for batch in train_loader:
            loss = model.training_step(batch)
            loss.backward()

            optimizer.step()
            optimizer.zero_grad()

        #Validate
        result = evaluate(model, val_loader)

        print(" Val_loss: ", result['loss'], " Accuracy: ",
result['accuracy'])

        history.append(result)
    return history
```

التدريب والاختبار

تم تدريب النموذج على 16 حقبة بمعدل تعلم 0.001 و16 أكثر بمعدل تعلم 0.0001.

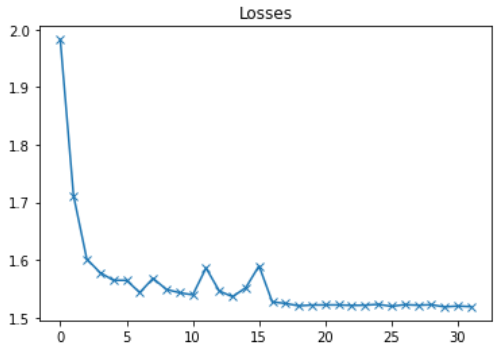
```
model = SpokenDigitModel()
history = []

def count_parameters(model):
    return sum(p.numel() for p in model.parameters() if p.requires_grad)
# evaluating on an untrained model
evaluate(model, val_dl)
from torchsummary import summary
import torch
import torchvision
from torch import nn
from torchvision import models
print(model)
count_parameters(model)
```

```
history.append(fit(model, train_dl, val_dl, 16, 0.001))
history.append(fit(model, train_dl, val_dl, 16, 0.0001))
evaluate(model, val_dl)
torch.save(model, 'spokendigit_cnn_mel.pth')
model = torch.load('spokendigit_cnn_mel.pth')
```

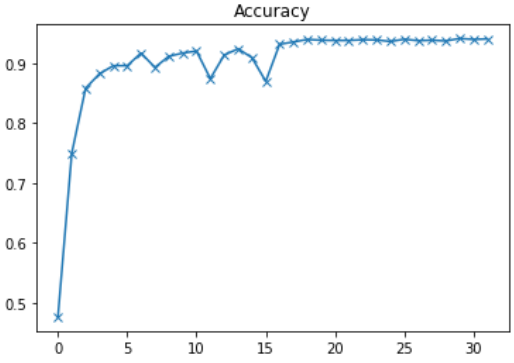
رسم المقاييس

```
losses = []
accuracies = []


# gathering metrics across all epochs
```

```
for i in range(len(history)):
    for j in history[i]:
        losses.append(j['loss'])
        accuracies.append(j['accuracy'])

# function to plot metrics
def plot(var, title):
    plt.plot(var, '-x')
    plt.title(title)

plot(losses, 'Losses')
```




```
plot(accuracies, 'Accuracy')
```



التنبؤ بالبيانات الخارجية

توجه إلى النوتبوك للاستماع إلى عينة الصوت الخارجي وشاهد كيف يعمل النموذج!

```
# function to get mel spectogram image
def get_mel(data, sr):
    fig = plt.figure(figsize=[1,1])
    ax = fig.add_subplot(111)
```

```
    ax.axes.get_xaxis().set_visible(False)
    ax.axes.get_yaxis().set_visible(False)
    ax.set_frame_on(False)

    # mel spectograms
    S = librosa.feature.melspectrogram(y=data, sr=sr)
    librosa.display.specshow(librosa.power_to_db(S, ref=np.max),
x_axis='time', y_axis='mel', fmin=50, fmax=280)
    file  = 'sample.jpg'
    plt.savefig(file, bbox_inches='tight',pad_inches=0)
    plt.close()

    img = Image.open(file)

    transform = transforms.Compose([transforms.ToTensor()])

    img = transform(np.asarray(img))

    # delete unnecessary file, not needed anymore.
    os.remove('sample.jpg')

    return img

# function to get output from audio sample
def get_prediction(model, img):
    # adding dimension corresponding to batch (3, 54, 55) --> (1, 3, 54, 55)
    output = model(img.unsqueeze(0)).detach().numpy()
    num = np.argmax(output)

    return output, num
```

```
# List of user recorded samples
user_samples = ['user_sample_1.wav', 'user_sample_2.wav',
'user_sample_3.wav', 'user_sample_5.wav']
sample_id = random.randint(0,3)    # select random sample

data, sr = librosa.load('melimages/'+user_samples[sample_id])
output, prediction = get_prediction(model, get_mel(data, sr))
print("Predicted {} with probability {}.".format(prediction,
output[0][prediction]))

ipd.Audio(data = data, rate = sr)
```

```
Predicted 1 with probability 1.0.
```

44) إنشاء الرموز التعبيرية الخاصة بك باستخدام التعلم العميق

الرموز التعبيرية Emojis أو الصور الرمزية avatars هي طرق للإشارة إلى الإشارات غير اللفظية. أصبحت هذه الإشارات جزءًا أساسيًا من الدردشة عبر الإنترنت، ومراجعة المنتجات، وعاطفة العلامة التجارية، وغير ذلك الكثير. كما أنه يؤدي إلى زيادة أبحاث علوم البيانات المخصصة لرواية القصص التي تعتمد على الرموز التعبيرية.

مع التقدمفي الرؤية الحاسوبية والتعلم العميق، أصبح من الممكن الآن اكتشاف المشاعر البشرية من الصور.في مشروع التعلم العميق هذا، سنقوم بتصنيف تعبيرات الوجه البشرية لتصفية ورسم الرموز التعبيرية أو الصور الرمزية المقابلة.

مجموعة البيانات

تتكون مجموعة بيانات FER2013 (التعرف على تعبيرات الوجه facial expression recognition) من صور وجه بتدرج الرمادي 48 * 48 بكسل. يتم توسيط الصور وتشغل مساحة متساوية. تتكون مجموعة البيانات هذه من مشاعر الوجه من الفئات التالية:

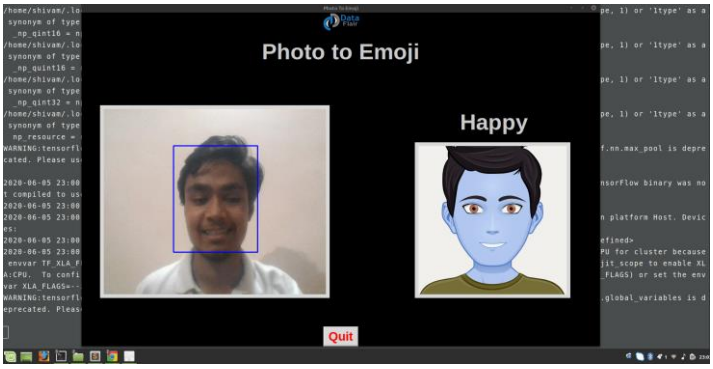
- 0:angry
- 1:disgust
- 2:feat
- 3:happy
- 4:sad
- 5:surprise
- 6:natural

تنزيل مجموعة البيانات: **Facial Expression Recognition Dataset**

تنزيل كود المشروع

قبل المتابعة، يرجى تنزيل الكود المصدري: **Emoji Creator Project Source Code**

إنشاء الرموز التعبيرية الخاصة بك مع التعلم العميق



سنقوم ببناء نموذج تعليمي عميق لتصنيف تعابير الوجه من الصور. ثم سنقوم بتعيين المشاعر المصنفة إلى رمز تعبيري أو أفاتار.

التعرف على مشاعر الوجه باستخدام CNN

في الخطوات التالية، سيتم إنشاء بنية شبكة عصبية التفافية CNN وتدريب النموذج على مجموعة بيانات FER2013 للتعرف على المشاعر من الصور.

قم بتنزيل مجموعة البيانات من الرابط أعلاه. استخرجهفي مجلد البيانات باستخدام مجلدات تدريب واختبار منفصلة.

قم بإنشاء ملف train.py واتبع الخطوات:

الخطوة 1: الاستيراد

```
import numpy as np
import cv2

from keras.emotion_models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from keras.optimizers import Adam
from keras.layers import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
```

الخطوة 2: بدء مولدات التدريب والتحقق من الصحة

```
train_dir = 'data/train'
val_dir = 'data/test'
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(48,48),
        batch_size=64,
        color_mode="gray_framescale",
        class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(48,48),
        batch_size=64,
        color_mode="gray_framescale",
        class_mode='categorical')
```

الخطوة 3: بناء بُنية شبكة الالتفاف

```
emotion_model = Sequential()

emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(48,48,1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion model.add(MaxPooling2D(pool size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))
```

الخطوة 4: تجميع النموذج وتدريبه

```
emotion_model.compile(loss='categorical_crossentropy',optimizer=Adam(l
r=0.0001, decay=1e-6),metrics=['accuracy'])

emotion_model_info = emotion_model.fit_generator(
        train_generator,
        steps_per_epoch=28709 // 64,
        epochs=50,
        validation_data=validation_generator,
        validation_steps=7178 // 64)
```

الخطوة 5: حفظ أوزان النموذج

```
emotion_model.save_weights('model.h5')
```

الخطوة 6: استخدام openCV haarcascade xml

باستخدام openCV haarcascade xml يكتشف المربعات المحيطة بالوجهفي كاميرا الويب ويتنبأ بالعواطف:

```
cv2.ocl.setUseOpenCL(False)

emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy",
4: "Neutral", 5: "Sad", 6: "Surprised"}

cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    bounding_box =
cv2.CascadeClassifier('/home/shivam/.local/lib/python3.6/site-
packages/cv2/data/haarcascade_frontalface_default.xml')
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2gray_frame)
    num_faces =
bounding_box.detectMultiScale(gray_frame,scaleFactor=1.3,
minNeighbors=5)

    for (x, y, w, h) in num_faces:
        cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)
        roi_gray_frame = gray_frame[y:y + h, x:x + w]
        cropped_img =
np.expand dims(np.expand dims(cv2.resize(roi gray frame, (48, 48)), -
1), 0)
        emotion_prediction = emotion_model.predict(cropped_img)
        maxindex = int(np.argmax(emotion_prediction))
        cv2.putText(frame, emotion_dict[maxindex], (x+20, y-60),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

    cv2.imshow('Video', cv2.resize(frame,(1200,860),interpolation =
cv2.INTER_CUBIC))
    if cv2.waitKey(1) & 0xFF == ord('q'):
cap.release()
cv2.destroyAllWindows()
```

```
    break
```

كود واجهة المستخدم الرسومية GUI والمطابقة باستخدام الرموز التعبيرية

أنشئ مجلداً باسم emojis واحفظ الرموز التعبيرية المقابلة لكل من المشاعر السبعةفي مجموعة البيانات.

الصق الكود أدناهفي gui.py وقم بتشغيل الملف.

```
import tkinter as tk
from tkinter import *
import cv2
from PIL import Image, ImageTk
import os
import numpy as np
import cv2
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from keras.optimizers import Adam
from keras.layers import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator

emotion_model = Sequential()

emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(48,48,1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))
emotion_model.load_weights('model.h5')

cv2.ocl.setUseOpenCL(False)

emotion_dict = {0: "   Angry   ", 1: "Disgusted", 2: "  Fearful  ", 3:
"   Happy   ", 4: "  Neutral  ", 5: "    Sad    ", 6: "Surprised"}


emoji_dist={0:"./emojis/angry.png",2:"./emojis/disgusted.png",2:"./emo
jis/fearful.png",3:"./emojis/happy.png",4:"./emojis/neutral.png",5:"./
emojis/sad.png",6:"./emojis/surpriced.png"}

global last_frame1
last_frame1 = np.zeros((480, 640, 3), dtype=np.uint8)
global cap1
show_text=[0]
def show_vid():
    cap1 = cv2.VideoCapture(0)
```

```
    if not cap1.isOpened():
        print("cant open the camera1")
    flag1, frame1 = cap1.read()
    frame1 = cv2.resize(frame1,(600,500))

    bounding_box =
cv2.CascadeClassifier('/home/shivam/.local/lib/python3.6/site-
packages/cv2/data/haarcascade_frontalface_default.xml')
    gray_frame = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
    num_faces =
bounding_box.detectMultiScale(gray_frame,scaleFactor=1.3,
minNeighbors=5)

    for (x, y, w, h) in num_faces:
        cv2.rectangle(frame1, (x, y-50), (x+w, y+h+10), (255, 0, 0),
2)
        roi_gray_frame = gray_frame[y:y + h, x:x + w]
        cropped_img =
np.expand_dims(np.expand_dims(cv2.resize(roi_gray_frame, (48, 48)), -
1), 0)
        prediction = emotion_model.predict(cropped_img)

        maxindex = int(np.argmax(prediction))
        cv2.putText(frame1, emotion_dict[maxindex], (x+20, y-60),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)
        show_text[0]=maxindex
    if flag1 is None:
        print ("Major error!")
    elif flag1:
        global last_frame1
        last_frame1 = frame1.copy()
        pic = cv2.cvtColor(last_frame1, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(pic)
        imgtk = ImageTk.PhotoImage(image=img)
        lmain.imgtk = imgtk
        lmain.configure(image=imgtk)
        lmain.after(10, show_vid)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        exit()


def show_vid2():
    frame2=cv2.imread(emoji_dist[show_text[0]])
    pic2=cv2.cvtColor(frame2,cv2.COLOR_BGR2RGB)
    img2=Image.fromarray(frame2)
    imgtk2=ImageTk.PhotoImage(image=img2)
    lmain2.imgtk2=imgtk2

lmain3.configure(text=emotion_dict[show_text[0]],font=('arial',45,'bol
d'))

    lmain2.configure(image=imgtk2)
    lmain2.after(10, show_vid2)

if __name__ == '__main__':
    root=tk.Tk()
    img = ImageTk.PhotoImage(Image.open("logo.png"))
    heading = Label(root,image=img,bg='black')

    heading.pack()
    heading2=Label(root,text="Photo to Emoji",pady=20,
font=('arial',45,'bold'),bg='black',fg='#CDCDCD')
```

```
    heading2.pack()
    lmain = tk.Label(master=root,padx=50,bd=10)
    lmain2 = tk.Label(master=root,bd=10)

    lmain3=tk.Label(master=root,bd=10,fg="#CDCDCD",bg='black')
    lmain.pack(side=LEFT)
    lmain.place(x=50,y=250)
    lmain3.pack()
    lmain3.place(x=960,y=250)
    lmain2.pack(side=RIGHT)
    lmain2.place(x=900,y=350)



    root.title("Photo To Emoji")
    root.geometry("1400x900+100+10")
    root['bg']='black'
    exitbutton = Button(root,
text='Quit',fg="red",command=root.destroy,font=('arial',25,'bold')).pa
ck(side = BOTTOM)
    show_vid()
    show_vid2()
    root.mainloop()
```

الملخص

في مشروع التعلم العميق هذا للمبتدئين، قمنا ببناء شبكة عصبية التفافية للتعرف على مشاعر الوجه. لقد قمنا بتدريب نموذجنا على مجموعة بيانات FER2013. ثم نقوم بتعيين تلك المشاعر باستخدام الرموز التعبيرية أو الصور الرمزية (الافاتار) المقابلة.

باستخدام haar cascade xml من OpenCV، نحصل على المربع المحيط للوجوهفي كاميرا الويب. ثم نقوم بتغذية هذه الصناديق بالنموذج المدرب للتصنيف.

45) وصف الدواء بناءً على بيانات المريض باستخدام التعلم العميق Prescribing drug based on patient data using deep learning

الدواء الموصوف prescription drug هو الدواء الذي يتطلب وصفة طبية يتم صرفها بموجب القانون. من ناحية أخرى، فإن الدواء الذي لا يستلزم وصفة طبية هو الدواء الذي يمكن الاستغناء عنه بدون وصفة طبية.

عندما يتعلق الأمر بوصفة الأدوية، يبحث الأطباءفي السمات المختلفة للبيانات المتعلقة بالمريض قبل التوصل إلى نتيجة. يمكن أن يكون لذلك عواقب تتراوح من فعالية الدواءفي جسم المريض إلى الآثار الجانبية التي تسببها والوصفات الطبية غير الصحيحة قد تؤديفي بعض الحالات إلى آثار لا رجعة فيهافي المرضى (بمافي ذلك الوفاة).

بادئ ذي بدء، هل يمكننا تدريب نموذج التعلم العميق لوصف الأدوية للمرضى بناءً على بياناتهم الطبية؟ تابع القراءة لمعرفة ذلك!

تنفيذ الكود على cAInvas – هنا!

استيراد المكتبات الضرورية

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import CategoricalCrossentropy
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping
import random
import matplotlib.pyplot as plt
```

مجموعة البيانات

مجموعة البيانات عبارة عن ملف CSV يحتوي على ميزات تتعلق بالمريض الذي يؤثر على وصفات الأدوية مثل العمر والجنس ومستوى ضغط الدم والكوليسترول ونسبة الصوديوم والبوتاسيوم والوصفات الطبية المقابلةفي كل حالة.

```
df = pd.read_csv('https://cainvas-
static.s3.amazonaws.com/media/user_data/cainvas-admin/drug200.csv')
df
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

```
df['Drug'].value_counts()
```

```
DrugY    91
drugX    54
drugA    23
drugC    16
drugB    16
Name: Drug, dtype: int64
```

هذه مجموعة بيانات غير متوازنة unbalanced dataset.

المعالجة المسبقة

موازنة مجموعة البيانات

من أجل تحقيق التوازنفي مجموعة البيانات، هناك خياران:

- upsampling: إعادة تشكيل القيم لجعل عددها مساويًا لتسمية الفئة ذات العدد الأعلى (هنا ، 1655).
- downsampling: انتقاء عينات من كل تصنيف فئة حيث n = عدد العيناتفي الفئة مع أقل عدد (هنا ، 16)

هنا، سنطبق upsampling.

```
categories = np.unique(df.Drug.to_list())
df_balanced = pd.DataFrame()
```

```
for i in range(len(categories)):
    # separating into individual dataframes, one for each class
    dfi = df[df['Drug'] == categories[i]]
    # resampling
    dfi = dfi.sample(91, replace = True)
    # appending all to one to form a final balanced dataframe
    df_balanced = df_balanced.append(dfi)

df_balanced['Drug'].value_counts()
```

```
drugX    91
drugB    91
drugC    91
drugA    91
DrugY    91
Name: Drug, dtype: int64
```

تم تعيين معلمة الاستبدال الخاصة بـ sample() على True للإشارة إلى أنه يمكن تكرار العينات في كل فئة لتحقيق العدد المحدد. يحتوي إطار البيانات df_balanced على 455 عينة، 91 من كل فئة.

المتغيرات الفئوية

لا يحدد عمود "Sex" نطاقًا، وبالتالي يتم ترميزه مرة واحدة أثناء التغيير من سمة فئوية إلى سمة رقمية. هذا يعني أنه إذا كانت هناك قيم فريدة nفي العمود، فسيتم إنشاء مصفوفة بطول n لكل منها حيث يتم تعيين قيمة i فقط على 1 مع الإشارة إلى مصفوفة تحدد فهارس قيم العمودفي المصفوفة.

```
dfx = pd.get_dummies(df_balanced[df_balanced.columns[:-1]], drop_first =
True, columns = ['Sex'])
dfx
```

	Age	BP	Cholesterol	Na_to_K	Sex_M
49	28	LOW	HIGH	19.796	0
172	39	NORMAL	NORMAL	17.225	0
178	39	NORMAL	HIGH	15.969	1
62	67	LOW	NORMAL	20.693	1
71	28	NORMAL	HIGH	19.675	0
...	...	...	...	...	...
16	69	LOW	NORMAL	11.455	1
95	36	LOW	NORMAL	11.424	1
58	60	NORMAL	NORMAL	10.091	1
35	46	NORMAL	NORMAL	7.285	1
181	59	NORMAL	HIGH	13.884	0

في كثير من الحالات (غالبًافي أعمدة الإدخال)، إذا كانت هناك قيم فريدة n، يتم إنشاء مصفوفة بطول n-1 حيث يمكن أن يكون العمود الإضافي زائداً عن الحاجة لتحديد قيمة العمود من المصفوفة المشفرة. يتم تحقيق ذلك عن طريق تعيين معلمة drop_first على أنها Trueفي دالة get_dummies() كما هو موضحفي خلية الكود أدناه.

نظرًا لأن هذا العمود يحتوي على قيمتين فريدتين فقطفي إطار البيانات، فلن يكون هناك أي اختلاف بين ترميز واحد ساخن one-hot encoding وترميز التسمية label encoding للعمود.

تمثل القيم الموجودةفي العمودين Cholesterol و BP قيم نوع النطاق كما تراها القيم أدناه:

```
print("Values in BP column:", np.unique(dfx['BP']))
print("Values in Cholesterol column:", np.unique(dfx['Cholesterol']))
```

```
Values in BP column: ['HIGH' 'LOW' 'NORMAL']
Values in Cholesterol column: ['HIGH' 'NORMAL']
```

تمثل القيم الموجودةفي العمودين Cholesterol و BP النطاق كما تراه القيم أعلاه.

```
le_bp = LabelEncoder()
le_bp.fit(['LOW', 'NORMAL', 'HIGH'])
dfx['BP'] = le_bp.transform(dfx['BP'], )
print("BP classes:", le_bp.classes_)

le_ch = LabelEncoder()
le_ch.fit(['NORMAL', 'HIGH'])
dfx['Cholesterol'] = le_ch.transform(dfx['Cholesterol'])
print("Cholesterol classes:", le_ch.classes_)

print(dfx)
```

```
BP classes: ['HIGH' 'LOW' 'NORMAL']
Cholesterol classes: ['HIGH' 'NORMAL']
     Age  BP  Cholesterol  Na_to_K  Sex_M
49    28   1            0   19.796      0
172   39   2            1   17.225      0
178   39   2            0   15.969      1
62    67   1            1   20.693      1
71    28   2            0   19.675      0
..   ...  ..          ...      ...    ...
16    69   1            1   11.455      1
95    36   1            1   11.424      1
58    60   2            1   10.091      1
35    46   2            1    7.285      1
181   59   2            0   13.884      0

[455 rows x 5 columns]
```

هذه الأعمدة عبارة عن تسمية مشفرة بدلاً من ترميز واحد ساخن، أي يتم استبدال كل قيمة بقيمة رقمية.

نظرًا لأن هذه مشكلة تصنيف، فإن إخراج النموذج الذي أصبح الآن كعدد صحيح يجب أن يكون مشفرًا واحدًا ساخنًا.

```
df_cat = pd.get_dummies(df_balanced['Drug'])
df_cat
```

	DrugY	drugA	drugB	drugC	drugX
49	1	0	0	0	0
172	1	0	0	0	0
178	1	0	0	0	0
62	1	0	0	0	0
71	1	0	0	0	0
...	...	...	...	...	...
16	0	0	0	0	1
95	0	0	0	0	1
58	0	0	0	0	1
35	0	0	0	0	1
181	0	0	0	0	1

```
# defining the input and output columns to separate the dataset in the later cells.

input_columns = dfx.columns.to_list()
output_columns = df_cat.columns.to_list()

print("Number of input columns: ", len(input_columns))
#print("Input columns: ", ', '.join(input_columns))

print("Number of output columns: ", len(output_columns))
#print("Output columns: ", ', '.join(output_columns))
```

```
Number of input columns:  5
Number of output columns:  5
```

```
for i in output_columns:
    dfx[i] = df_cat[i]

del df_cat

dfx
```

	Age	BP	Cholesterol	Na_to_K	Sex_M	DrugY	drugA	drugB	drugC	drugX
49	28	1	0	19.796	0	1	0	0	0	0
172	39	2	1	17.225	0	1	0	0	0	0
178	39	2	0	15.969	1	1	0	0	0	0
62	67	1	1	20.693	1	1	0	0	0	0
71	28	2	0	19.675	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
16	69	1	1	11.455	1	0	0	0	0	1
95	36	1	1	11.424	1	0	0	0	0	1
58	60	2	1	10.091	1	0	0	0	0	1
35	46	2	1	7.285	1	0	0	0	0	1
181	59	2	0	13.884	0	0	0	0	0	1

455 rows × 10 columns

تقسيم الاختبار والتدريب

استخدام نسبة 80ـ10ـ10 لتقسيم إطار البيانات إلى مجموعات تدريبـ التحقق من الصحةـ اختبار. ثم يتم تقسيمها إلى X و y (المدخلات والمخرجات) لمزيد من المعالجة.

```
# Splitting into train, val and test set -- 80-10-10 split

# First, an 80-20 split
train_df, val_test_df = train_test_split(dfx, test_size = 0.2, random_state =
13)

# Then split the 20% into half
val_df, test_df = train_test_split(val_test_df, test_size = 0.5, random_state
= 13)

print("Number of samples in...")
print("Training set: ", len(train_df))
print("Validation set: ", len(val_df))
print("Testing set: ", len(test_df))
```

```
Number of samples in...
Training set:  364
Validation set:  45
Testing set:  46
```

تحتوي مجموعة التدريب على 364 عينة بينما تحتوي مجموعة التحقق من الصحة على 45 عينة ومجموعة الاختبار بها 46 عينة.

```
# Splitting into X (input) and y (output)

Xtrain, ytrain = np.array(train_df[input_columns]),
np.array(train_df[output_columns])

Xval, yval = np.array(val_df[input_columns]),
np.array(val_df[output_columns])
```

```
Xtest, ytest = np.array(test_df[input_columns]),
np.array(test_df[output_columns])
```

تحجيم القيم

نظرة خاطفة على لقطة إطار بيانات dfx تجعل من الواضح أن الأعمدة لها قيمفي نطاقات مختلفة. يمكن استخدام أداة تغيير الحجم min-max لقياس القيم بين الحد الأدنى والحد الأقصى للقيم المحددة (min-0، max-1 افتراضيًا).

```
# Each feature has a different range.
# Using min_max_scaler to scale them to values in the range [0,1].

min_max_scaler = MinMaxScaler()

# Fit on training set alone
Xtrain = min_max_scaler.fit_transform(Xtrain)

# Use it to transform val and test input
Xval = min_max_scaler.transform(Xval)
Xtest = min_max_scaler.transform(Xtest)
```

يتم استخدام دالة MinMaxScaler لوحدة sklearn.preprocessing. منطقيًا، مجموعة التدريب هي البيانات الوحيدة التي يُسمح لنا برؤيتها أو العمل بها أثناء تدريب النموذج بينما يتم استخدام الاثنان الآخران لتقييم أدائه، ويكون كائن MinMaxScaler مناسبًا لبيانات التدريب ويتم استخدام النموذج المجهز لتحويل البياناتفي جميع مجموعات البيانات الثلاث.

بناء النموذج

النموذج بسيط يتكون فقط من طبقات كثيفة.

```
model = Sequential([
    Dense(1024, activation = 'relu', input_shape = Xtrain[0].shape),
    Dense(512, activation = 'relu'),
    Dense(256, activation = 'relu'),
    Dense(64, activation = 'relu'),
    Dense(len(output_columns), activation = 'softmax')
])

cb = [EarlyStopping(monitor = 'val_loss', patience=8,
restore_best_weights=True)]
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1024)              6144
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 64)                16448
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 325
=================================================================
Total params: 679,045
Trainable params: 679,045
Non-trainable params: 0
```

يتم تجميع النموذج باستخدام دالة الخطأ عبر الانتروبيا لأن الطبقة الأخيرة من النموذج لها دالة تنشيط softmax والتسميات مشفرة واحدة ساخنة. يتم استخدام مُحسِّن آدم ويتم تتبع دقة النموذج على مر الحقب.

تراقب دالة رد الاتصال EarlyStopping خطأ التحقق من الصحة وتوقف التدريب إذا لم ينخفض لمدة 8 فترات بشكل مستمر. تضمن معلمة rest_best_weights استعادة النموذج الذي لديه أقل خطأفي التحقق من الصحة إلى متغير النموذج.

يتم تدريب النموذج بمعدل تعلم 0.01 لحقبة 64 ولكن النموذج يتوقف قبل ذلك بسبب callbacks.

تجميع وتدريب النموذج

```
model.compile(optimizer=Adam(0.01), loss=CategoricalCrossentropy(),
metrics=['accuracy'])

history = model.fit(Xtrain, ytrain, validation_data = (Xval, yval),
epochs=64, callbacks=cb)
```

```
12/12 [==============================] - 0s 3ms/step - loss: 0.1079 - accuracy: 0.9615 - val_loss: 0.1907 - val_accuracy: 0.9556
Epoch 26/64
12/12 [==============================] - 0s 3ms/step - loss: 0.0567 - accuracy: 0.9863 - val_loss: 0.0198 - val_accuracy: 1.0000
Epoch 27/64
12/12 [==============================] - 0s 3ms/step - loss: 0.0354 - accuracy: 0.9863 - val_loss: 0.0048 - val_accuracy: 1.0000
Epoch 28/64
12/12 [==============================] - 0s 3ms/step - loss: 0.1038 - accuracy: 0.9780 - val_loss: 0.0210 - val_accuracy: 1.0000
Epoch 29/64
12/12 [==============================] - 0s 3ms/step - loss: 0.0453 - accuracy: 0.9918 - val_loss: 0.0955 - val_accuracy: 0.9778
```

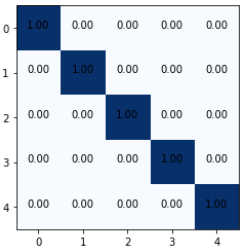
تقييم النموذج

```
model.evaluate(Xtest, ytest)
```

```
2/2 [==============================] - 0s 1ms/step - loss: 0.0076 - accuracy:
1.0000
```

مصفوفة الارتباك

```
cm = confusion_matrix(np.argmax(ytest, axis = 1),
np.argmax(model.predict(Xtest), axis = 1))
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

for i in range(cm.shape[1]):
    for j in range(cm.shape[0]):
        plt.text(j, i, format(cm[i, j], '.2f'), horizontalalignment="center",
color="black")

plt.imshow(cm, cmap=plt.cm.Blues)
```

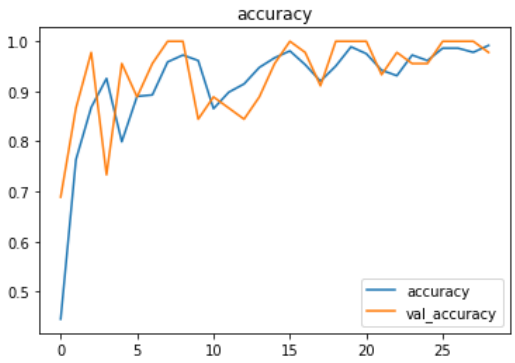


من المهم الحفاظ على الدقة عالية للغاية (100٪) حيث لا يمكن أخذ الفرص مع دواء المريض.

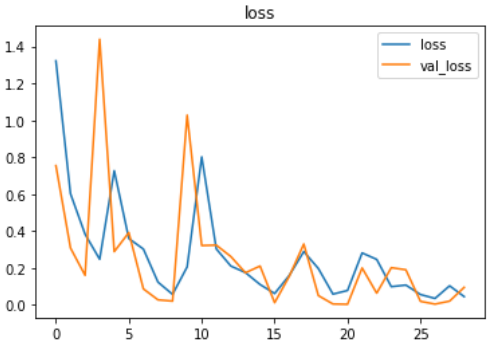
رسم المقاييس

```
def plot(history, variable, variable1):
    plt.plot(range(len(history[variable])), history[variable])
    plt.plot(range(len(history[variable1])), history[variable1])
    plt.title(variable)
    plt.legend([variable, variable1])
```

```
plot(history.history, "accuracy", "val_accuracy")
```




```
plot(history.history, "loss", "val_loss")
```



التنبؤ

لننفذ تنبؤات على عينات بيانات الاختبار العشوائية:

```
gender = ['M', 'F']

def print_sample(x):
    print("\nSample:")
    sample = np.array(test_df)[x]
    print("Age :", sample[0])
    print("Sex :", gender[int(sample[4])])
    print("Na to K ratio :", sample[3])
    print("BP :", le_bp.classes_[int(sample[1])])
```

```
    print("Cholesterol :", le_ch.classes_[int(sample[2])])
    print()
```

```
# pick random test data sample from one batch
x = random.randint(0, len(Xtest) - 1)

print_sample(x)

output = model.predict(Xtest[x].reshape(1, -1))    # getting output; input
shape (256, 256, 3) --> (1, 256, 256, 3)
pred = np.argmax(output[0])    # finding max
print("Predicted: ", output_columns[pred])    # Picking the label from
class_names base don the model output

output_true = np.array(ytest)[x]

print("True: ", output_columns[np.argmax(output_true)])
print("Probability: ", output[0][pred])
```

```
Sample:
Age : 59.0
Sex : F
Na to K ratio : 13.935
BP : HIGH
Cholesterol : HIGH

Predicted:  drugB
True:  drugB
Probability:  0.99355537
```

46) تصنيف المركبات العضوية باستخدام التعلم العميق Classification on Organic Compounds using Deep Learning

أنشأ شبكة عصبية اصطناعية بسيطة باستخدام TensorFlow و Keras التي تصنف المركبات العضوية على أنها إما مركبات مسك Musk أو غير مسك Non-Musk.

الهدف

لتطوير نموذج التعلم العميق الذي يصنف المركبات العضوية كمركبات مسك أو غير مسك باستخدام لغة برمجة بايثون ومكتبات التعلم العميق.

المتطلبات الأساسية

قبل البدء، يجب أن يكون لديك فهم جيد لما يلي:

1. لغة برمجة بايثون.
2. مكتبات التعلم العميق (Tensorflow ، Keras).

مجموعة البيانات

رابط لتنزيل مجموعة البيانات: https://datahub.io/machine-learning/musk

الحصول على البيانات:

```
# get data file
!wget -N "https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-admin/musk.csv"
```

المخرجات:

```
--2021-07-13 11:06:24--  https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-admin/musk.csv
Resolving cainvas-static.s3.amazonaws.com (cainvas-static.s3.amazonaws.com)... 52.219.66.124
Connecting to cainvas-static.s3.amazonaws.com (cainvas-static.s3.amazonaws.com)|52.219.66.124|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4622724 (4.4M) [text/csv]
Saving to: ‘musk.csv’
```

استيراد المكتبات المطلوبة:

```
# Import the required libraries
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
```

قم بتحميل البيانات:

```
# read the csv file
dataset = pd.read_csv('musk.csv')
dataset.head()
```

المخرجات:

	ID	molecule_name	conformation_name	f1	f2	f3	f4	f5	f6	f7	...	f158	f159	f160	f161	f162	f163	f164	f165	f166	class
0	1	MUSK-211	211_1+1	46	-108	-60	-69	-117	49	38	...	-308	52	-7	39	126	156	-50	-112	96	1
1	2	MUSK-211	211_1+10	41	-188	-145	22	-117	-6	57	...	-59	-2	52	103	136	169	-61	-136	79	1
2	3	MUSK-211	211_1+11	46	-194	-145	28	-117	73	57	...	-134	-154	57	143	142	165	-67	-145	39	1
3	4	MUSK-211	211_1+12	41	-188	-145	22	-117	-7	57	...	-60	-4	52	104	136	168	-60	-135	80	1
4	5	MUSK-211	211_1+13	41	-188	-145	22	-117	-7	57	...	-60	-4	52	104	137	168	-60	-135	80	1

المعالجة المسبقة للبيانات

```
X = dataset.iloc[:, 3:-1].values
y = dataset.iloc[:, -1].values

# Scaling
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X = ss.fit_transform(X)
```

التقسيم الى بيانات تدريب واختبار

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
random_state = 42)

X_train.shape,y_train.shape
```

المخرجات:

```
((4618, 166), (4618,))
```

بناء النموذج وتدريبه وحفظه

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(33, input_shape=(166,),
                          activation=tf.nn.tanh),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model.summary()

# compile
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Save the model
model.save("Simple ANN.h5")
```

المخرجات:

```
Epoch 1/15
145/145 [==============================] - 0s 2ms/step - loss: 0.4147 -
accuracy: 0.8367 - val_loss: 0.2794 - val_accuracy: 0.9146
Epoch 2/15
145/145 [==============================] - 0s 2ms/step - loss: 0.2333 -
accuracy: 0.9290 - val_loss: 0.2037 - val_accuracy: 0.9293
Epoch 3/15
145/145 [==============================] - 0s 2ms/step - loss: 0.1808 -
accuracy: 0.9415 - val_loss: 0.1635 - val_accuracy: 0.9500
Epoch 4/15
145/145 [==============================] - 0s 2ms/step - loss: 0.1522 -
accuracy: 0.9511 - val_loss: 0.1428 - val_accuracy: 0.9545
```

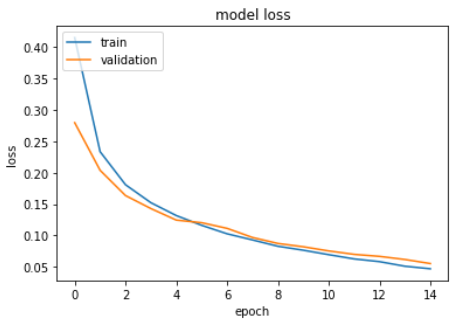
```
Epoch 5/15
145/145 [==============================] - 0s 2ms/step - loss: 0.1317 -
accuracy: 0.9580 - val_loss: 0.1245 - val_accuracy: 0.9626
Epoch 6/15
145/145 [==============================] - 0s 2ms/step - loss: 0.1162 -
accuracy: 0.9643 - val_loss: 0.1204 - val_accuracy: 0.9581
Epoch 7/15
145/145 [==============================] - 0s 2ms/step - loss: 0.1027 -
accuracy: 0.9664 - val_loss: 0.1113 - val_accuracy: 0.9616
Epoch 8/15
145/145 [==============================] - 0s 2ms/step - loss: 0.0929 -
accuracy: 0.9701 - val_loss: 0.0969 - val_accuracy: 0.9677
Epoch 9/15
145/145 [==============================] - 0s 2ms/step - loss: 0.0829 -
accuracy: 0.9729 - val_loss: 0.0874 - val_accuracy: 0.9737
Epoch 10/15
145/145 [==============================] - 0s 2ms/step - loss: 0.0766 -
accuracy: 0.9729 - val_loss: 0.0821 - val_accuracy: 0.9747
Epoch 11/15
145/145 [==============================] - 0s 2ms/step - loss: 0.0695 -
accuracy: 0.9786 - val_loss: 0.0755 - val_accuracy: 0.9753
Epoch 12/15
145/145 [==============================] - 0s 2ms/step - loss: 0.0627 -
accuracy: 0.9816 - val_loss: 0.0700 - val_accuracy: 0.9788
Epoch 13/15
145/145 [==============================] - 0s 2ms/step - loss: 0.0584 -
accuracy: 0.9827 - val_loss: 0.0668 - val_accuracy: 0.9783
Epoch 14/15
145/145 [==============================] - 0s 2ms/step - loss: 0.0511 -
accuracy: 0.9864 - val_loss: 0.0618 - val_accuracy: 0.9803
Epoch 15/15
145/145 [==============================] - 0s 2ms/step - loss: 0.0471 -
accuracy: 0.9857 - val_loss: 0.0554 - val_accuracy: 0.9833
```

الرسوم البيانية

الخطأ loss مقابل خطأ التحقق من الصحة validation loss

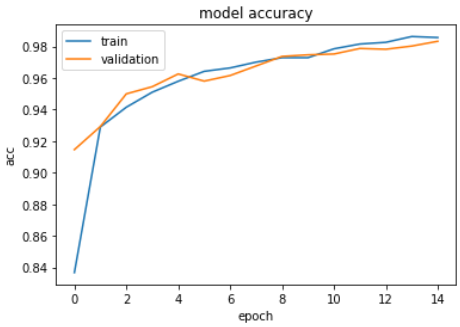
```
# train_loss vs Val_loss
from matplotlib import pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

المخرجات:



الدقة accuracy مقابل دقة التحقق من الصحة validation accuracy

```
# train_accuracy vs val_accuracy
from matplotlib import pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('acc')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



دقة النموذج

```
# accuracy and loss
model.evaluate(X_test, y_test)
```

المخرجات:

```
62/62 [==============================] - 0s 951us/step - loss: 0.0554 -
accuracy: 0.9833
```

التنبؤ

```
y_pred = model.predict(X_test)
y_pred[5:10]
```

المخرجات:

```
array([[8.6293503e-04],
       [4.8145223e-01],
       [3.1779730e-03],
       [2.4487602e-03],
       [2.7712667e-04]], dtype=float32)
```

```
y_pred1 = []
for element in y_pred:
    if element > 0.5:
        y_pred1.append(1)
    else:
        y_pred1.append(0)
```

```
y_pred1[25:40]
```

المخرجات:

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
```

```
y_test[25:40]
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0])
```

هنا يمكننا أن نرى أن القيم المتوقعة هي نفس القيم الفعلية.

تقرير التصنيف

```
# print the classification report
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test,y_pred1))
```

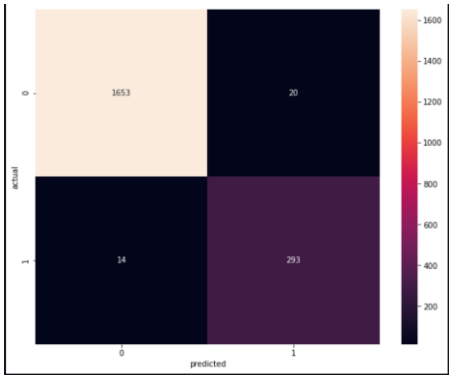
المخرجات:

```
   precision    recall  f1-score   support

           0       0.98      1.00      0.99      1673
           1       0.98      0.91      0.94       307

    accuracy                           0.98      1980
   macro avg       0.98      0.95      0.97      1980
weighted avg       0.98      0.98      0.98      1980
```

خريطة الحرارة

```
import seaborn as sn

cm = tf.math.confusion_matrix(labels = y_test, predictions = y_pred1)

plt.figure(figsize = (10,8))
sn.heatmap(cm, annot = True, fmt = 'd')
plt.xlabel("predicted")
plt.ylabel("actual")
```

المخرجات:



الملخص

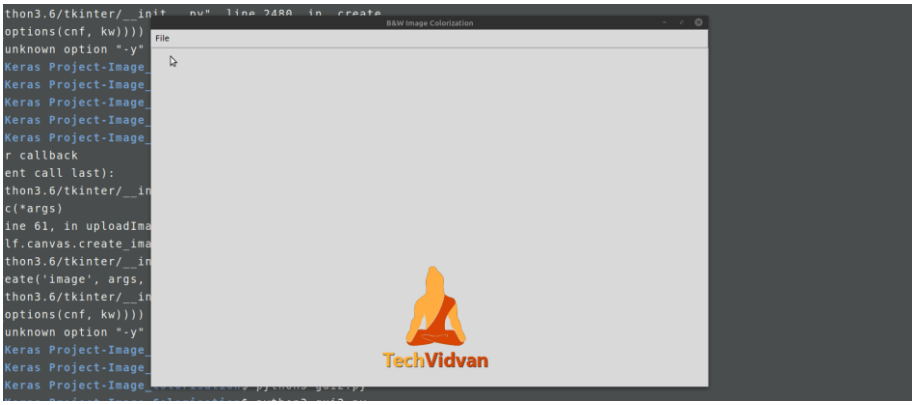
لقد قمنا بتدريب ANN البسيط الخاص بنا باستخدام TensorFlow و Keras لتصنيف مركبات المسك / غير المسك وحصلنا على دقة تصل إلى 98٪.

رابط كود المشروع اعلاه: هنا

47) تلوين الصور بالأبيض والأسود باستخدام التعلم العميق Colorize Black & White Images using Deep Learning

يهدف مشروع التعلم العميق هذا إلى توفير تلوين الصور بالأبيض والأسود باستخدام لغة بايثون.

في تلوين الصورة، نأخذ صورة بالأبيض والأسود كمدخلات وننتج صورة ملونة. سنحل هذا المشروع باستخدام شبكة OpenCV العصبية العميقة.



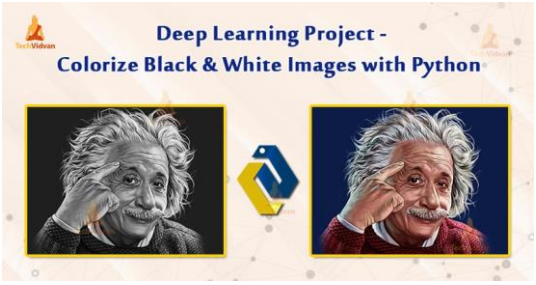
الفضاء اللوني L*a*b:

مثل RGB، يعد الفضاء اللوني L*a*b أخرى. إنها أيضًا مساحة لونية ثلاثية القنوات مثل RGB حيث تكون القنوات:

- قناة L: تمثل هذه القناة الخفة.
- قناة a: هذه القناة تمثل الأخضر والأحمر.
- قناة b: تمثل هذه القناة الأزرق والأصفر.

في الفضاء اللوني هذه، يتم ترميز جزء التدرج الرمادي من الصورةفي قناة L فقط. لذلك فإن الفضاء اللوني L*a*b أكثر ملاءمة لمشروعنا.

عرض المشكلة:



يمكننا صياغة بيان المشكلة الخاص بنا للتنبؤ بقناتي a و b، بالنظر إلى صورة الإدخال بتدرج الرمادي.

في مشروع التعلم العميق هذا، سنستخدم بنية OpenCV DNN التي يتم تدريبها على مجموعة بيانات ImageNet. يتم تدريب الشبكة العصبية باستخدام قناة L للصور كبيانات إدخال وقنوات a و b كبيانات مستهدفة.

خطوات تنفيذ مشروع تلوين الصورة:

لتلوين الصور بالأبيض والأسود، سنستخدم نموذج caffe تم تدريبه مسبقًا وملف prototxt وملف NumPy.

يحدد ملف prototxt الشبكة ويقوم الملف numpy بتخزين نقاط مركز الكلاستر بتنسيق numpy.

الخطوة 1: قم بعمل دليل مع نماذج الأسماء.

```
mkdir models
```

الخطوة 2: افتح الجهاز وقم بتشغيل الأوامر التالية لتنزيل ملف caffemodel و prototxt وملف NumPy.

```
wget
https://github.com/richzhang/colorization/blob/master/colorization/res
ources/pts_in_hull.npy?raw=true -O ./pts_in_hull.npy

wget
https://raw.githubusercontent.com/richzhang/colorization/master/colori
zation/models/colorization_deploy_v2.prototxt -O
./models/colorization_deploy_v2.prototxt

wget
http://eecs.berkeley.edu/~rich.zhang/projects/2016_colorization/files/
demo_v2/colorization_release_v2.caffemodel -O
./models/colorization_release_v2.caffemodel
```

الخطوة 3: قم بإنشاء ملف python image_colorization.py والصق الكود المحدد في الخطوات أدناه

الخطوة 4: استيراد المكتبات

```
import numpy as np
import cv2 as cv
import os.path
```

الخطوة 5: اقرأ الصورة بالأبيض والأسود وحمِّل caffemodel:

```
frame = cv.imread("__test_image_path__")

numpy_file = np.load('./pts_in_hull.npy')

Caffe_net =
cv.dnn.readNetFromCaffe("./models/colorization_deploy_v2.prototxt",
"./models/colorization_release_v2.caffemodel")
```

الخطوة 6: أضف طبقات إلى نموذج caffe:

```
numpy_file = numpy_file.transpose().reshape(2, 313, 1, 1)
Caffe_net.getLayer(Caffe_net.getLayerId('class8_ab')).blobs =
[numpy_file.astype(np.float32)]
Caffe_net.getLayer(Caffe_net.getLayerId('conv8_313_rh')).blobs =
[np.full([1, 313], 2.606, np.float32)]
```

الخطوة 7: استخراج قناة L وتغيير حجمها:

```
input_width = 224
input_height = 224

rgb_img = (frame[:,:,[2, 1, 0]] * 1.0 / 255).astype(np.float32)
lab_img = cv.cvtColor(rgb_img, cv.COLOR_RGB2Lab)
l channel = lab img[:,:,0]

l_channel_resize = cv.resize(l_channel, (input_width, input_height))
l_channel_resize -= 50
```

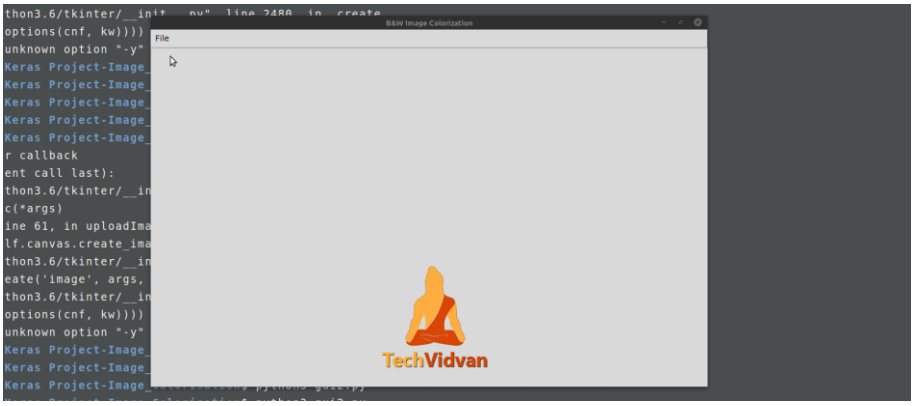
الخطوة 8: توقع قناة ab وحفظ النتيجة:

```
Caffe_net.setInput(cv.dnn.blobFromImage(l_channel_resize))
ab_channel = Caffe_net.forward()[0,:,:,:].transpose((1,2,0))

(original_height,original_width) = rgb_img.shape[:2]
ab_channel_us = cv.resize(ab_channel, (original_width,
original_height))
lab_output =
np.concatenate((l_channel[:,:,np.newaxis],ab_channel_us),axis=2)
bgr_output = np.clip(cv.cvtColor(lab_output, cv.COLOR_Lab2BGR), 0, 1)

cv.imwrite("./result.png", (bgr_output*255).astype(np.uint8))
```

النتائج:

قم الآن بتنفيذ مشروع التعلم العميق: قم بتشغيل ملف بايثون بمسار صورة ذات تدرج رمادي لاختبار نتائجنا.

```
python3 image_colorization.py
```



كود واجهة المستخدم الرسومية:

قم بإنشاء ملف بايثون جديد gui.py في الدليل الحالي. الصق الكود أدناه لواجهة تلوين الصورة.

```
import tkinter as tk
```

```
from tkinter import *
from tkinter import filedialog
from PIL import Image, ImageTk
import os
import numpy as np
import cv2 as cv
import os.path


numpy_file = np.load('./pts_in_hull.npy')
Caffe_net =
cv.dnn.readNetFromCaffe("./models/colorization_deploy_v2.prototxt",
"./models/colorization release v2.caffemodel")
numpy_file = numpy_file.transpose().reshape(2, 313, 1, 1)

class Window(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)

        self.master = master
        self.pos = []
        self.master.title("B&W Image Colorization")
        self.pack(fill=BOTH, expand=1)

        menu = Menu(self.master)
        self.master.config(menu=menu)

        file = Menu(menu)
        file.add_command(label="Upload Image",
command=self.uploadImage)
        file.add command(label="Color Image", command=self.color)
        menu.add_cascade(label="File", menu=file)

        self.canvas = tk.Canvas(self)
        self.canvas.pack(fill=tk.BOTH, expand=True)
        self.image = None
        self.image2 = None

        label1=Label(self,image=img)
        label1.image=img
        label1.place(x=400,y=370)



    def uploadImage(self):
        filename = filedialog.askopenfilename(initialdir=os.getcwd())
        if not filename:
            return
        load = Image.open(filename)

        load = load.resize((480, 360), Image.ANTIALIAS)

        if self.image is None:
            w, h = load.size
            width, height = root.winfo_width(), root.winfo_height()
            self.render = ImageTk.PhotoImage(load)
            self.image = self.canvas.create_image((w / 2, h / 2),
image=self.render)

        else:
            self.canvas.delete(self.image3)
            w, h = load.size
```

```
            width, height = root.winfo_screenmmwidth(),
root.winfo_screenheight()

            self.render2 = ImageTk.PhotoImage(load)
            self.image2 = self.canvas.create_image((w / 2, h / 2),
image=self.render2)


        frame = cv.imread(filename)

        Caffe_net.getLayer(Caffe_net.getLayerId('class8_ab')).blobs =
[numpy_file.astype(np.float32)]
        Caffe_net.getLayer(Caffe_net.getLayerId('conv8_313_rh')).blobs
= [np.full([1, 313], 2.606, np.float32)]

        input_width = 224
        input_height = 224

        rgb_img = (frame[:,:,[2, 1, 0]] * 1.0 /
255).astype(np.float32)
        lab_img = cv.cvtColor(rgb_img, cv.COLOR_RGB2Lab)
        l_channel = lab_img[:,:,0]

        l_channel_resize = cv.resize(l_channel, (input_width,
input_height))
        l_channel_resize -= 50

        Caffe_net.setInput(cv.dnn.blobFromImage(l_channel_resize))
        ab_channel = Caffe_net.forward()[0,:,:,:].transpose((1,2,0))

        (original_height,original_width) = rgb_img.shape[:2]
        ab_channel_us = cv.resize(ab_channel, (original_width,
original_height))
        lab_output =
np.concatenate((l_channel[:,:,np.newaxis],ab_channel_us),axis=2)
        bgr_output = np.clip(cv.cvtColor(lab_output,
cv.COLOR_Lab2BGR), 0, 1)


        cv.imwrite("./result.png", (bgr_output*255).astype(np.uint8))

    def color(self):

        load = Image.open("./result.png")
        load = load.resize((480, 360), Image.ANTIALIAS)

        if self.image is None:
            w, h = load.size
            self.render = ImageTk.PhotoImage(load)
            self.image = self.canvas.create_image((w / 2, h/2),
image=self.render)
            root.geometry("%dx%d" % (w, h))
        else:
            w, h = load.size
            width, height = root.winfo_screenmmwidth(),
root.winfo_screenheight()

            self.render3 = ImageTk.PhotoImage(load)
            self.image3 = self.canvas.create_image((w / 2, h / 2),
image=self.render3)
            self.canvas.move(self.image3, 500, 0)
```

```
root = tk.Tk()
root.geometry("%dx%d" % (980, 600))
root.title("B&W Image Colorization GUI")
img = ImageTk.PhotoImage(Image.open("logo2.png"))

app = Window(root)
app.pack(fill=tk.BOTH, expand=1)
root.mainloop()
```

الملخص

يرشدك هذا البرنامج التعليمي إلى كيفية إنشاء مشروع تعليمي عميق لتلوين الصور بالأبيض والأسود. يقدم لك أولاً الفضاء اللوني L*a*b ولماذا يكون مناسبًا لبيان مشكلتنا. ثم يصف خطوة بخطوة كيفية تنفيذ تلوين الصور بالأبيض والأسود.

48) تصنيف العلامات التجارية باستخدام التعلم العميق logos classification using deep learning

هناك 6 أنواع مختلفة من الشعارات (Burger King, McDonalds, Other, Starbucks, Subway, KFC). لقد استخدمنا CNN للتنبؤ بصور الكرة الجديدة.

يتنبأ هذا النموذج بما إذا كانت الصورة تخص أي علامة تجارية من خلال رؤية الشعارات باستخدام شبكة CNN.

استيراد المكتبات:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Dense, Activation,Dropout,Conv2D,
MaxPooling2D,BatchNormalization, Flatten
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, Sequential
import numpy as np
import pandas as pd
import shutil
import time
import cv2 as cv2
from tqdm import tqdm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import os
import seaborn as sns
sns.set_style('darkgrid')
from PIL import Image
from sklearn.metrics import confusion_matrix, classification_report
from IPython.core.display import display, HTML
```

فك ضغط مجموعة البيانات:

```
!wget https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-
admin/logos3.zip
!unzip -qo logos3.zip
# zip folder is not needed anymore
!rm logos3.zip
```

المعالجة المسبقة للبيانات:

قم بتعيين التصنيفات، أي الفئات إلى عدد صحيح، واعرض قائمة بجميع العلامات التجارية الستة الفريدة.

```
directory = "logos3/train"
Name=[]
for file in os.listdir(directory):
    Name+=[file]
print(Name)
print(len(Name))
brand_map = dict(zip(Name, [t for t in range(len(Name))]))
print(brand_map)
r_brand_map=dict(zip([t for t in range(len(Name))],Name))
```

المخرجات:

```
{'Burger King': 0, 'McDonalds': 1, '.DS_Store': 2, 'Other': 3, 'Starbucks':
4, 'Subway': 5, 'KFC': 6}
```

عرض بعض الصور من مجموعة البيانات:

```
Brand = 'logos3/train/Starbucks'
import os
sub_class = os.listdir(Brand)

fig = plt.figure(figsize=(10,5))
for e in range(len(sub_class[:10])):
    plt.subplot(2,5,e+1)
    img = plt.imread(os.path.join(Brand,sub_class[e]))
    plt.imshow(img, cmap=plt.get_cmap('gray'))
    plt.axis('off')
```

المخرجات:

تقسيم مجموعة بيانات التدريب إلى مجموعة تدريب ومجموعة التحقق من الصحة:

```
img_datagen = ImageDataGenerator(rescale=1./255,
                                 vertical_flip=True,
                                 horizontal_flip=True,
                                 rotation_range=40,
                                 width_shift_range=0.2,
                                 height_shift_range=0.2,
```

```
                                zoom_range=0.1,
                                validation_split=0.2)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = img_datagen.flow_from_directory(directory,
                                                  shuffle=True,
                                                  batch_size=32,
                                                  subset='training',
                                                  target_size=(100, 100))
```

المخرجات:

```
Found 1393 images belonging to 6 classes.
Found 345 images belonging to 6 classes.
```

تدريب النموذج التسلسلي:

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=(100,100,3),
activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding =
'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(6))
# model.add(Dense(len(brand_map)))
model.add(Activation('softmax'))

model.summary()
```

المخرجات:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 100, 100, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 50, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 50, 50, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 6, 6, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
dropout (Dropout)            (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 256)               16640
_________________________________________________________________
activation (Activation)      (None, 256)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 1542
_________________________________________________________________
activation_1 (Activation)    (None, 6)                 0
=================================================================
Total params: 185,286
Trainable params: 185,286
Non-trainable params: 0
_________________________________________________________________
```

تجميع وتدريب النموذج:

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
```

```
            metrics=['accuracy'])

history = model.fit(train_generator,
validation_data=valid_generator,batch_size= 32,epochs=50)
```

المخرجات:

```
44/44 [==============================] - 43s 976ms/step - loss: 0.2197 - accuracy: 0.9289 - val_loss: 0.4022 - val_accuracy: 0.8841
Epoch 46/50
44/44 [==============================] - 42s 945ms/step - loss: 0.2045 - accuracy: 0.9318 - val_loss: 0.3505 - val_accuracy: 0.8899
Epoch 47/50
44/44 [==============================] - 43s 972ms/step - loss: 0.2034 - accuracy: 0.9354 - val_loss: 0.5845 - val_accuracy: 0.8522
Epoch 48/50
44/44 [==============================] - 43s 975ms/step - loss: 0.1802 - accuracy: 0.9361 - val_loss: 0.4730 - val_accuracy: 0.8319
Epoch 49/50
44/44 [==============================] - 43s 966ms/step - loss: 0.2496 - accuracy: 0.9196 - val_loss: 0.5039 - val_accuracy: 0.8348
Epoch 50/50
44/44 [==============================] - 41s 941ms/step - loss: 0.2247 - accuracy: 0.9275 - val_loss: 0.5120 - val_accuracy: 0.8435
```

رسم المنحنيات:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and validation accuracy')
plt.show()
```


```
training_accuracy = history.history['loss']
validation_accuracy = history.history['val_loss']
plt.plot(training_accuracy, 'r', label = 'training loss')
plt.plot(validation_accuracy, 'b', label = 'validation loss')
plt.title('training and test loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

التنبؤ

```
# from tensorflow.keras.models import load_img
from tensorflow.keras.preprocessing.image import load_img
load_img("logos3/test/McDonalds/armada_image_755.jpg",target_size=(180,180))
```


```
# from tensorflow.keras import image
from tensorflow.keras.preprocessing import image
test_image = image.load_img('logos3/test/KFC/armada_image_169.jpg',
target_size = (100, 100))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = model.predict(test_image)
print(result)
```

المخرجات:

```
[[1. 0. 0. 0. 0. 0.]]
```

رابط الكود: انقر هنا.

49) الكشف عن شذوذ ضربات القلب باستخدام التعلم العميق Heartbeat Anomaly Detection using Deep Learning

وفقًا لتقرير صادر عن منظمة الصحة العالمية، يموت حوالي 17.9 مليون شخص سنويًا بسبب أمراض القلب والأوعية الدموية Cardiovascular Diseases، وعلى مر السنين تم اكتشاف أنه يمكن منع هذه الوفيات إذا تم تشخيص الأمراض في مرحلة مبكرة وحتى يمكن علاج المرض.

استيراد مجموعة البيانات

```
!wget -N "https://cainvas-static.s3.amazonaws.com/media/user_data/cainvas-admin/heart.zip"
!unzip -qo heart.zip
!rm heart.zip
```

استيراد المكتبات الضرورية

```
# Pandas
import pandas as pd

# Scikit learn
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import shuffle
from sklearn.utils import class_weight

# Keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.utils import to_categorical
from keras.optimizers import Adam

# Audio
import librosa
import librosa.display

# Plot
%matplotlib inline
%pylab inline
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'

# Utility
import os
import glob
import numpy as np
from tqdm import tqdm
import itertools

# To ignore any warnings
import warnings
warnings.filterwarnings("ignore")


# gather software versions
```

```
import tensorflow as tf; print('tensorflow version: ', tf.__version__)
import keras; print('keras version: ',keras.__version__)

# If any warning pops up run the cell again.There is nothing to worry about.
```

بناء مجموعة البيانات

```
dataset = []
for folder in ["heart/set_a/**"]:
    for filename in glob.iglob(folder):
        if os.path.exists(filename):
            label = os.path.basename(filename).split("_")[0]
            # skip audio smaller than 4 secs
            if librosa.get_duration(filename=filename)>=4:
                if label not in ["Aunlabelledtest"]:
                    dataset.append({
                        "filename": filename,
                        "label": label
                    })
dataset = pd.DataFrame(dataset)
```

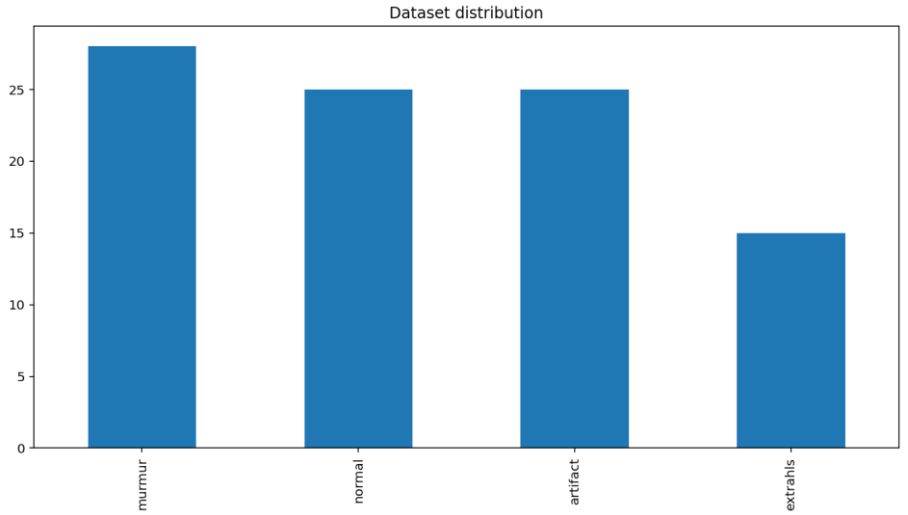
تحليل البيانات

تتكون مجموعة البيانات التي عملنا عليها من أربعة أنواع من أصوات ضربات القلب وهي:

1. Normal
2. Murmur
3. Artifact
4. Extrahls

دعونا نرى توزيع مجموعة البيانات:

```
dataset.info()
plt.figure(figsize=(12,6))
dataset.label.value_counts().plot(kind='bar', title="Dataset distribution")
plt.show()
```



```
# parent folder of sound files
INPUT_DIR="heart"
# 16 KHz
SAMPLE_RATE = 16000
# seconds
MAX_SOUND_CLIP_DURATION=12
```

```
set_a=pd.read_csv(INPUT_DIR+"/set_a.csv")
set_a.head()
set_a.info()
```

```
train_ab=set_a
train_ab.describe()
```

	sublabel
count	0.0
mean	NaN
std	NaN
min	NaN
25%	NaN
50%	NaN
75%	NaN
max	NaN

```
#get all unique labels
nb_classes=train_ab.label.unique()

print("Number of training examples=", train_ab.shape[0], "  Number of
classes=", len(train_ab.label.unique()))
print (nb_classes)
```

```
Number of training examples= 176   Number of classes= 5
['artifact' 'extrahls' 'murmur' 'normal' nan]
```

```
print('Minimum samples per category = ', min(train_ab.label.value_counts()))
print('Maximum samples per category = ', max(train_ab.label.value_counts()))
```

```
Minimum samples per category =  19
Maximum samples per category =  40
```

حالة Normal

في الفئة "Normal "، توجد أصوات قلب طبيعية وصحية. يحتوي صوت القلب الطبيعي على نمط "lub dub، lub dub" واضح، مع الوقت من "lub" إلى "dub" أقصر من الوقت من "dub" إلى "lub" التالي.

```
normal_file=INPUT_DIR+"/set_a/normal__201106111136.wav"
```

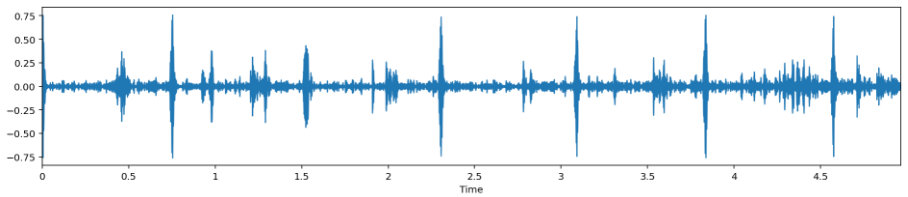
```
# Load use wave
import wave
wav = wave.open(normal_file)
print("Sampling (frame) rate = ", wav.getframerate())
print("Total samples (frames) = ", wav.getnframes())
print("Duration = ", wav.getnframes()/wav.getframerate())
```

```
Sampling (frame) rate =  44100
Total samples (frames) =  218903
Duration =  4.963786848072562
```

```
# Load using Librosa
y, sr = librosa.load(normal_file, duration=5)   #default sampling rate is 22
HZ
dur=librosa.get_duration(y)
print ("duration:", dur)
print(y.shape, sr)
```

```
duration: 4.963809523809524
(109452,) 22050
```

```
# librosa plot
plt.figure(figsize=(16, 3))
librosa.display.waveplot(y, sr=sr)
```



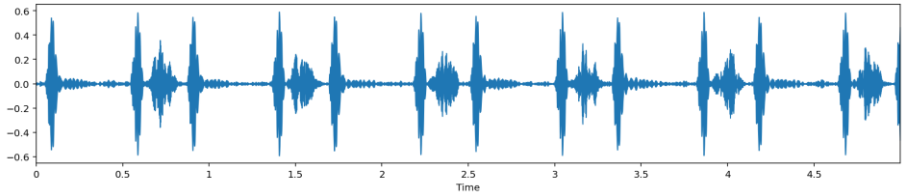
حالة Murmur

صوت لغط القلب كما لو كان هناك ضوضاء "أزيز أو هدير أو هدير أو سائل مضطرب"في أحد الموقعين الزمنيين: (1) بين "lub" و "dub"، أو (2) بين "dub" و "lub ". يمكن أن تكون من أعراض العديد من اضطرابات القلب، وبعضها خطير. سيظل هناك "lub" و "dub".

```
# murmur case
murmur_file=INPUT_DIR+"/set_a/murmur__201108222231.wav"
y2, sr2 = librosa.load(murmur_file,duration=5)
dur=librosa.get_duration(y)
print ("duration:", dur)
print(y2.shape,sr2)
```

```
duration: 4.963809523809524
(110250,) 22050
```

```
# show it
plt.figure(figsize=(16, 3))
librosa.display.waveplot(y2, sr=sr2)
```



Artifact

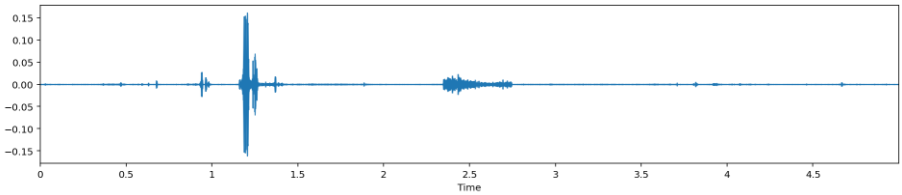
في فئة Artifact هناك مجموعة واسعة من الأصوات المختلفة، بمافي ذلك صرير ردود الفعل والصدى والكلام والموسيقى والضوضاء. عادة لا توجد أصوات قلب يمكن تمييزها، وبالتالي فإن دورية زمنية قليلة أو معدومة على ترددات أقل من 195 هرتز.

```
# sample file
artifact_file=INPUT_DIR+"/set_a/artifact__201012172012.wav"
y4, sr4 = librosa.load(artifact_file, duration=5)
dur=librosa.get_duration(y)
print ("duration:", dur)
print(y4.shape,sr4)
```

```
duration: 4.963809523809524
(110250,) 22050
```

```
# show it
plt.figure(figsize=(16, 3))
```

```
librosa.display.waveplot(y4, sr=sr4)
```



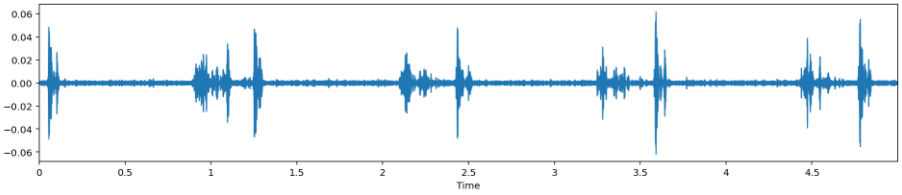
Extrahls

قد تظهر أصوات Extrahls من حين لآخر ويمكن التعرف عليها نظرًا لوجود صوت قلب خارج الإيقاع يتضمن ضربات قلب زائدة أو متقطعة، على سبيل المثال "lub-lub dub" أو "lub dub-dub". يمكن أن يكون علامة على مرض.

```
# sample file
extrahls_file=INPUT_DIR+"/set_a/extrahls__201101070953.wav"
y5, sr5 = librosa.load(extrahls_file, duration=5)
dur=librosa.get_duration(y)
print ("duration:", dur)
print(y5.shape,sr5)
```

```
duration: 4.963809523809524
(110250,) 22050
```

```
# show it
plt.figure(figsize=(16, 3))
librosa.display.waveplot(y5, sr=sr5)
```



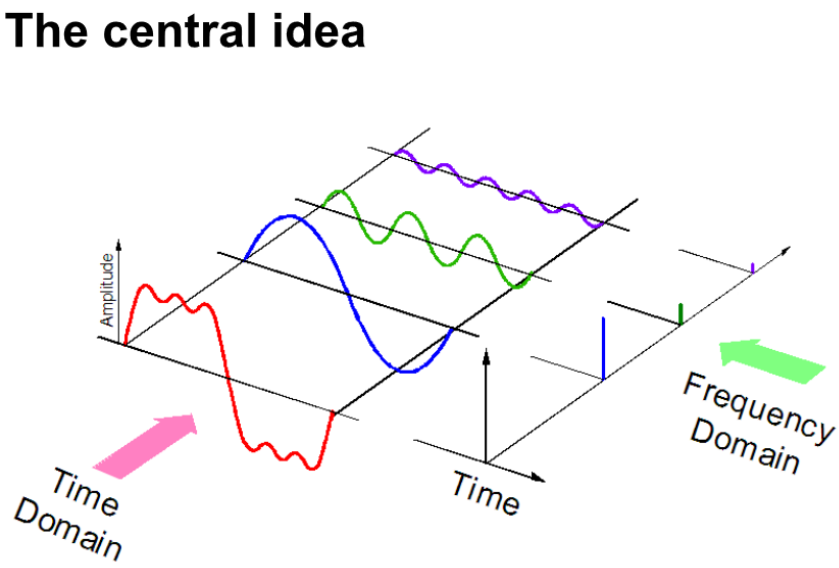
تقسيم مجموعة البيانات الى تدريب واختبار

```
train, test = train_test_split(dataset, test_size=0.2, random_state=42)

print("Train: %i" % len(train))
print("Test: %i" % len(test))
```

```
Train: 74
Test: 19
```

إظهار معلومات الصوت

كما هو الحال مع جميع تنسيقات البيانات غير المهيكلة، تحتوي البيانات الصوتية على خطوتين للمعالجة المسبقة التي يجب اتباعها قبل تقديمها للتحليل. هناك طريقة أخرى لتمثيل البيانات الصوتية وهي تحويلها إلى مجال مختلف لتمثيل البيانات، أي مجال التردد frequency domain.



هناك عدد من الطرق التي يمكن من خلالها تمثيل البيانات الصوتية مثل استخدام MFCs (Mel-Frequency cepstrums) و Mel-Spectograms وغيرها.

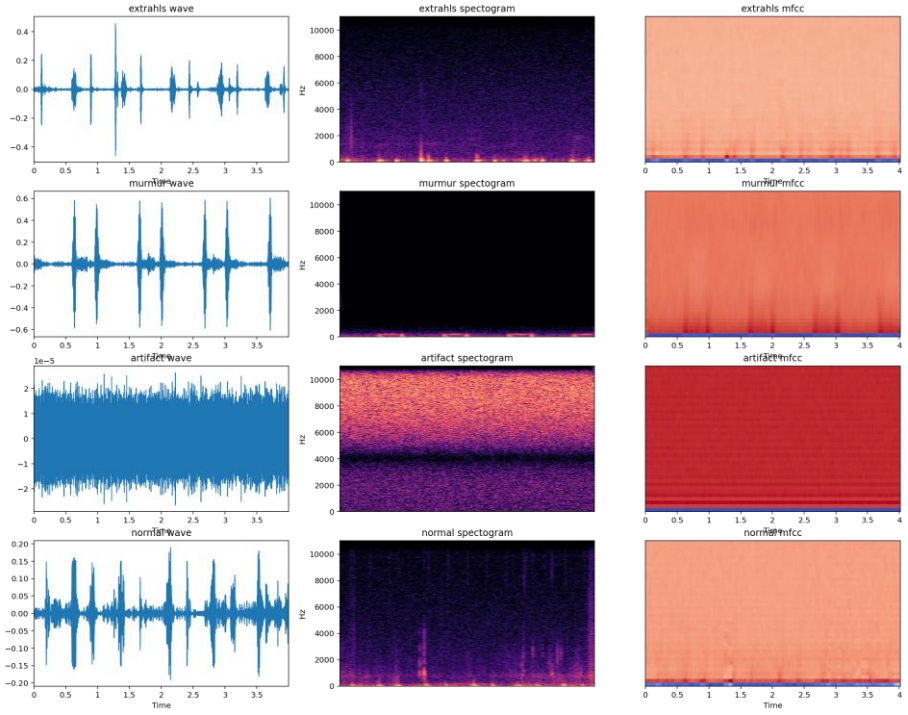
تشمل ميزات الصوت العامة:

- ميزات Time Domain (مثل RMSE للشكل الموجي)
- ميزات Frequency domain (مثل سعة الترددات الفردية)
- ميزات Perceptual (مثل MFCC)
- ميزات Windowing (على سبيل المثال، Hamming distances of windows)

بعد استخراج هذه الميزات، يتم إرسالها بعد ذلك إلى نموذج التعلم الآلي لمزيد من التحليل. بالنسبة لنموذجنا، استخدمنا MFCC كميزة صوتية لدينا ويمكن استخلاصها بسهولة من

الصوتيات باستخدام مكتبة بايثون تسمى Librosa. لاستخراج الميزات نحن فقط يجب عليك إنشاء دالة واستخدام librosa لاستخراج ميزات MFCC لنا.

```
plt.figure(figsize=(20,20))
idx = 0
for label in dataset.label.unique():
    y, sr = librosa.load(dataset[dataset.label==label].filename.iloc[0],
duration=4)
    idx+=1
    plt.subplot(5, 3, idx)
    plt.title("%s wave" % label)
    librosa.display.waveplot(y, sr=sr)
    idx+=1
    plt.subplot(5, 3, idx)
    D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
    librosa.display.specshow(D, y_axis='linear')
    plt.title("%s spectogram" % label)
    idx+=1
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
    plt.subplot(5, 3, idx)
    librosa.display.specshow(mfccs, x_axis='time')
    plt.title("%s mfcc" % label)
plt.show()
```



استخراج الميزات من الصوت

```
def extract_features(audio_path):
    y, sr = librosa.load(audio_path, duration=4)
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
    return mfccs
```

```
%%time
x_train, x_test = [], []
print("Extract features from TRAIN  and TEST dataset")
for idx in tqdm(range(len(train))):
    x_train.append(extract_features(train.filename.iloc[idx]))

for idx in tqdm(range(len(test))):
    x_test.append(extract_features(test.filename.iloc[idx]))


x_test = np.asarray(x_test)
x_train = np.asarray(x_train)

print("X train:", x_train.shape)
print("X test:", x_test.shape)
```

```
X train: (74, 40, 173)
X test: (19, 40, 173)
CPU times: user 23.3 s, sys: 28 s, total: 51.2 s
Wall time: 17.3 s
```

ترميز التسميات

```
encoder = LabelEncoder()
encoder.fit(train.label)

y_train = encoder.transform(train.label)
y_test = encoder.transform(test.label)
```

شكل الادخال

```
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1],
x_train.shape[2], 1)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)



print("X train:", x_train.shape)
print("Y train:", y_train.shape)
print()
print("X test:", x_test.shape)
print("Y test:", y_test.shape)
```

```
X train: (74, 40, 173, 1)
Y train: (74, 4)
```

```
X test: (19, 40, 173, 1)
Y test: (19, 4)
```

بناء النموذج

بعد استخراج الميزات، كانت الخطوة التالية هي تمرير الميزة المستخرجة لدينا كبيانات تدريب لنموذج التعلم العميق الخاص بنا جنبًا إلى جنب مع التسميات المستهدفة حتى يتمكن نموذجنا من تعلم تصنيف أصوات نبضات القلب والعثور على الحالات الشاذةفي أصوات ضربات القلب. كانت بُنية النموذج المستخدمة هي:

```
# Model architecture

model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2,
input_shape=(x_train.shape[1],x_train.shape[2],x_train.shape[3]),
activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.3))

model.add(Conv2D(filters=32, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.3))

model.add(Conv2D(filters=64, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.3))

model.add(Conv2D(filters=128, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.3))
model.add(GlobalAveragePooling2D())

model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))

model.add(Dense(len(encoder.classes_), activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 39, 172, 16)       80
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 19, 86, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 19, 86, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 18, 85, 32)        2080
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 9, 42, 32)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 9, 42, 32)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 41, 64)         8256
```

```
max_pooling2d_2 (MaxPooling2 (None, 4, 20, 64)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 20, 64)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 19, 128)        32896
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 9, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 1, 9, 128)         0
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 256)               33024
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 516
=================================================================
Total params: 109,748
Trainable params: 109,748
Non-trainable params: 0
_________________________________________________________________
```

تجميع النموذج

```
model.compile(loss='categorical_crossentropy', metrics=['accuracy'],
optimizer=Adam(lr = 0.001))
```

تدريب النموذج

```
history = model.fit(x_train, y_train,
              batch_size=256,
              epochs=200,
              validation_data=(x_test, y_test),
              shuffle=True)
```

```
1/1 [==============================] - 0s 29ms/step - loss: 0.6135 - accuracy: 0.7703 - val_loss: 0.5327 - val_accuracy: 0.7895
Epoch 196/200
1/1 [==============================] - 0s 28ms/step - loss: 0.5791 - accuracy: 0.7027 - val_loss: 0.5404 - val_accuracy: 0.7895
Epoch 197/200
1/1 [==============================] - 0s 27ms/step - loss: 0.5258 - accuracy: 0.7027 - val_loss: 0.5314 - val_accuracy: 0.7895
Epoch 198/200
1/1 [==============================] - 0s 30ms/step - loss: 0.5717 - accuracy: 0.7162 - val_loss: 0.5192 - val_accuracy: 0.7895
Epoch 199/200
1/1 [==============================] - 0s 28ms/step - loss: 0.5407 - accuracy: 0.7297 - val_loss: 0.5185 - val_accuracy: 0.7895
Epoch 200/200
1/1 [==============================] - 0s 27ms/step - loss: 0.5488 - accuracy: 0.7432 - val_loss: 0.5206 - val_accuracy: 0.7895
```

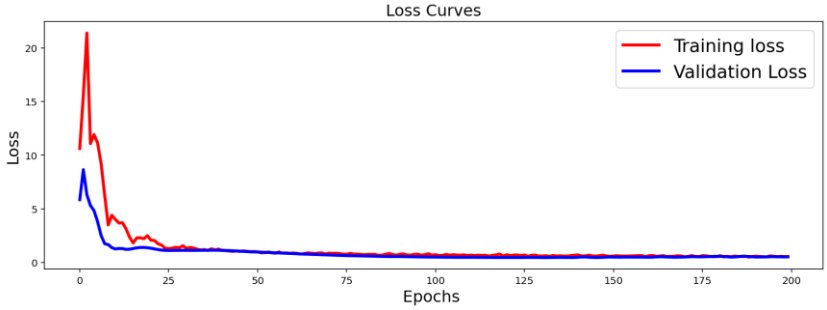
يمكن زيادة أداء النموذج من خلال زيادة عدد الحقب، وبيانات التدريب وإنشاء callbacks مخصصة.

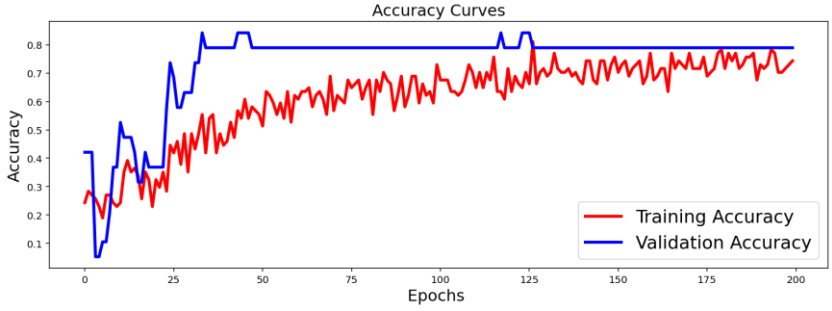
رسم التدريب

دالة الخطأ المستخدمة كانت "categorical_crossentropy" والمحسن المستخدم كان "Adam" لتدريب النموذج استخدمنا Keras API مع Tensorflow في الخلفية، أظهر النموذج أداءً جيدًا محققًا دقة مناسبة، وهنا منحنى الخطأ والدقة للنموذج:

```
# Loss Curves
plt.figure(figsize=[14,10])
```

```
plt.subplot(211)
plt.plot(history.history['loss'],'r',linewidth=3.0)
plt.plot(history.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Loss',fontsize=16)
plt.title('Loss Curves',fontsize=16)

# Accuracy Curves
plt.figure(figsize=[14,10])
plt.subplot(212)
plt.plot(history.history['accuracy'],'r',linewidth=3.0)
plt.plot(history.history['val_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.title('Accuracy Curves',fontsize=16)
```





حفظ النموذج

```
# Save model and weights
model_name = "HAD.h5"
model.save(model_name)
print('Saved trained model at %s ' % model_name)
```

تقييم النموذج

```
scores = model.evaluate(x_test, y_test, verbose=1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

```
1/1 [==============================] - 0s 1ms/step - loss: 0.5206 - accuracy:
0.7895
Test loss: 0.520578145980835
Test accuracy: 0.7894737124443054
```

الوصول إلى أداء النموذج

```
predictions = model.predict(x_test, verbose=1)
```

```
1/1 [==============================] - 0s 1ms/step
```

```
y_preds = predictions.argmax(axis=-1)
y_test = y_test.argmax(axis=-1)
y_pred = encoder.inverse_transform(y_preds)
y_test = encoder.inverse_transform(y_test)
df = pd.DataFrame(columns=['Predicted Labels', 'Actual Labels'])
df['Predicted Labels'] = y_pred.flatten()
df['Actual Labels'] = y_test.flatten()
df.head(19)
```

	Predicted Labels	Actual Labels
0	murmur	normal
1	artifact	artifact
2	murmur	murmur
3	extrahls	normal
4	murmur	extrahls
5	artifact	normal
6	artifact	artifact
7	artifact	artifact
8	murmur	murmur
9	artifact	artifact
10	artifact	artifact
11	murmur	murmur
12	murmur	murmur
13	artifact	artifact
14	murmur	murmur
15	murmur	murmur
16	murmur	murmur
17	artifact	artifact
18	murmur	murmur

50) الكشف عن اعتلال الشبكية السكري باستخدام التعلم العميق Diabetic Retinopathy Detection using Deep Learning

الهدف من هذا البرنامج التعليمي هو تطوير نظام الكشف الآلي عن اعتلال الشبكية السكري diabetic retinopathy باستخدام الشبكة العصبية التلافيفية CNN. كانت هذه إحدى المسابقات التي أقيمتفي Kaggle. تحتاج إلى إنشاء حساب على Kaggle لتتمكن من تنزيل قاعدة البيانات. رابط دفتر ipython الذي يحتوي على هذا الكود موجودفي نهاية هذا البرنامج التعليمي.

لنبدأ مباشرة مع التدريب العملي:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import kerasfrom tqdm import tqdm
import os
from sklearn.model_selection import train_test_split
from cv2 import cv2
from PIL import Image
import tensorflow as tf
from matplotlib import pyplot as pltfrom keras.layers import Dense,
Dropout, Flatten, Input
from keras.preprocessing.image import ImageDataGenerator, array_to_img,
img_to_array, load_img
from keras.preprocessing import image
from keras.utils import plot_model
from keras.models import Model
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from numpy import array
```

لقد قمت بالفعل بتنزيل مجموعة البيانات من Kaggle. (بمجرد تنزيل مجموعة البيانات، يجب عليك فك ضغط جميع ملفات التدريب والاختبار، يجب عليك ربط جميع ملفات التدريب لإنشاء مجلد بسيط لصور التدريب)

أدناه أقوم بتحميل ملف CSV يحتوي على تسميات التدريب

```
df_train = pd.read_csv('/storage/trainLabels.csv')
```

دعونا نلقي نظرة على جميع التسميات.

"left_10" هو اسم الملف بينما '0/1/2/3/4' هي التسميات.

"Left_10" صورة للعين اليسرى بالمثل صورة "right _10" للعين اليمنى لنفس الشخص.

```
df_train.values
```

المخرجات:

```
array([['10_left', 0],
['10_right', 0],
['13_left', 0],
...,
```

```
['44348_right', 0],
['44349_left', 0],
['44349_right', 1]], dtype=object)
```

هناك 35125 صورةفي مجموعة التدريب، 'level'هو العمود الذي يشير إلى تسميات الصور الخاصة بها.

```
df_train.tail()
```

	image	level
35121	44347_right	0
35122	44348_left	0
35123	44348_right	0
35124	44349_left	0
35125	44349_right	1

سنستخدم Pandas لتحويل df_train إلى سلسلة و get_dummies للقيام بتشفير واحد ساخن (لمعلوماتك، أنا لا أستخدم تشفيرًا واحدًا ساخنًا أثناء التدريب حتى الآن).

```
targets_series = pd.Series(df_train['level'])
one_hot = pd.get_dummies(targets_series, sparse = True)
```

كما قلت من قبل، هناك 5 أنواع من التسميات 4/3/2/1/0، وهي مميزة على النحو التالي: اعتلال الشبكية السكري غير التكاثري (Non Proliferative Diabetic — NDPR Retinopathy)

فئة – الاسم (Class — Name)

1. عادي Normal
2. خفيف Mild NPDR
3. معتدل Moderate NPDR
4. شديد Severe NPDR
5. PDR

```
targets_series[:10]
```

```
0    0
1    0
2    0
3    0
4    1
5    2
6    4
7    4
8    0
9    1
Name: level, dtype: int64
```

```
one_hot[:10]
```

	0	1	2	3	4
0	1	0	0	0	0
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	0	1	0	0	0
5	0	0	1	0	0
6	0	0	0	0	1
7	0	0	0	0	1
8	1	0	0	0	0
9	0	1	0	0	0

دعنا نلقي نظرة على المصفوفة التي تحتوي على التسميات فقط:

```
one_hot_labels = np.asarray(one_hot)
one_hot_labelsY = np.asarray(targets_series)
one_hot_labelsY[:10]
```

المخرجات:

```
array([0, 0, 0, 0, 1, 2, 4, 4, 0, 1])
```

سنقوم الآن بتهيئة شكل الصورة والمصفوفات لتحميل الصور والتسميات:

```
im_size1 = 786
im_size2 = 786
x_train = []
y_train = []
```

إذا كنت مهتمًا بالتحقق من جميع أسماء الصور:

```
i = 0
for f, breed in tqdm(df_train.values):
print(f)
```

المخرجات:

```
10_left
10_right
13_left
13_right
15_left
15 right
16_left
16_right
17_left
```

```
17_right
19_left
```

```
df_test = df_train[:1000]
```

إذا كنت تخطط لتشغيل هذا الكود على جميع الصور البالغ عددها 35125 ، فاستبدل df_test بـ .df_train سيؤدي مقتطف الكود أدناه إلى تحميل جميع الصور والتسمياتفي مصفوفة عددية.

يمكنك أيضًا تحميل الصور باستخدامOpenCV ، لقد ذكرت كود OpenCV في التعليقات أدناه.

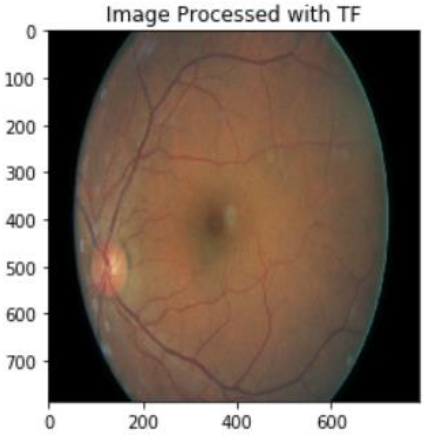
```
"""
#this is a OpenCV implementation
i = 0
for f, breed in tqdm(df_train.values):
    if
type(cv2.imread('/storage/train/{}.jpeg'.format(f)))==type(None):
        continue
    else:
        img = cv2.imread('/storage/train/{}.jpeg'.format(f))
        label = one_hot_labels[i]
        x train.append(cv2.resize(img, (im size1, im size2)))
        y_train.append(label)
        i += 1
np.save('x_train2',x_train)
np.save('y_train2',y_train)
print('Done')
"""i=0
for f, breed in tqdm(df_test.values):
    try:
        img = image.load_img(('/storage/train/{}.jpeg'.format(f)),
target_size=(786, 786))
        arr = image.img_to_array(img)
        label = one hot labelsY[i]
        x_train.append(arr)
        y_train.append(label)
        i += 1
    except:
        pass
```

المخرجات:

```
100%|██████████| 1000/1000 [01:43<00:00, 7.06it/s]
```

دعنا فقط نتحقق من إحدى الصور من المصفوفة الرقمية:

```
plt.imshow(x_train[681]/255) #681 > Try some other number too
plt.show()
```



من المهم تقسيم مجموعة البيانات بأكملها إلى مجموعة بيانات للتدريب والتحقق من الصحة بصرف النظر عن مجموعة بيانات الاختبار التي لدينا بشكل منفصل.

```
x_valid = []
y_valid = []
X_train, X_valid, Y_train, Y_valid = train_test_split(x_train, y_train,
test_size=0.1, random_state=1)
```

الآن سوف نحدد النموذج >>

يحتوي النموذج على 2 طبقات تلافيفية، وطبقتي تجميع بحد أقصى، وطبقة تسطيح للصورة وطبقة dence. تأتي النماذجفي Keras / TF في شكلين ـ متسلسل (= model Sequential()) أو باستخدام Funtional API.

يستخدم الكود أدناه Funtional API والذي يستخدم عادة للنماذج المعقدة، سأترك نموذج Sequential()خفيف الوزن ()في التعليقات. يمكنك تعديل الطبقات لإنشاء النموذج المخصص الخاص بك.

```
visible = Input(shape=(786,786,3))
conv1 = Conv2D(32, kernel_size=4, activation='relu')(visible)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
conv2 = Conv2D(16, kernel_size=4, activation='relu')(pool1)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
flat = Flatten()(pool2)
hidden1 = Dense(10, activation='relu')(flat)
output = Dense(1, activation='sigmoid')(hidden1)
model = Model(inputs=visible, outputs=output)
```

إذا كنت تخطط لتشغيل نموذج أقل تعقيداً يتم تشغيله أسفل الأسطر، فسأقترح استخدام تقنية نقل التعلم transfer learning إذا كنت تخطط لاستخدام النموذج أدناه للحصول على نتائج أفضل.

```
model = keras.Sequential([
keras.layers.Flatten(input_shape=(786, 786, 3)),
```

```
keras.layers.Dense(128, activation=tf.nn.relu),
keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

```
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

لنحول المصفوفة إلى مصفوفة numpy ، قد يستغرق هذا بعض الوقت.

```
y_train_raw = np.array(Y_train)
x_train_raw = np.array(X_train)
```

هذه هي الطريقة التي تتكدس بها الطبقات فوق بعضها البعض.

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 786, 786, 3)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 783, 783, 32)      1568
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 391, 391, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 388, 388, 16)      8208
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 194, 194, 16)      0
_________________________________________________________________
flatten_1 (Flatten)          (None, 602176)            0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                6021770
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 6,031,557
Trainable params: 6,031,557
Non-trainable params: 0
_________________________________________________________________
```

سيقوم هذا الأمر بالفعل بتدريب النموذج. حتى مع وجود عدد أقل من الصور، قد تصادف "خطأ غير كافٍ في الذاكرة Insufficient memory error" أو "خطأ في إعادة تشغيل الكيرنل Kernel restart error"

```
model.fit(x_train_raw, y_train_raw, epochs=5)
```

يتيح تحويل المصفوفة إلى مصفوفة numpy ، لمجموعة بيانات التحقق.

```
x_valid_raw = np.array(X_valid)
y_valid_raw = np.array(Y_valid)
```

بمجرد تدريب النموذج، نحتاج إلى تقييم أداء النموذج بجميع مجموعة بيانات التحقق من الصحة.

```
test_loss, test_acc = model.evaluate(x_valid_raw, y_valid_raw)
test_loss
test_acc
```

يمكنك العثور على كتاب عقدة iPython على:

https://github.com/swanandM/Diabetic-Retinopathy-Detection-with-TF.git

يمكنك تعديل هذا الكود لتنفيذ بعض الأشياء مثل زيادة البيانات وتطبيع الدُفعات والتسرب ودوال الخسارة المخصصة للحصول على أداء أفضل.

المصادر

1. Deep Learning Projects with Python, Aman Kharwal, https://thecleverprogrammer.com/2020/11/22/deep-learning-projects-with-python/.
2. 23 Amazing Deep Learning Project Ideas, https://data-flair.training/blogs/deep-learning-project-ideas/.
3. 23 Amazing Deep Learning Project Ideas [Source Code Included], https://data-flair.training/blogs/deep-learning-project-ideas/
4. Use Cases, https://cainvas.ai-tech.systems/
5. Deep Learning Project, https://techvidvan.com/tutorials/
6. Deep Learning Project, https://www.analyticsvidhya.com/

Deep Learning

By Example

20 Deep Learning Projects Solved and Explained with Python

By: Dr. Alaa Taima