التعلم العميق

واستخداماته في الرعاية الصحية

٢٥ مشروع تعلم عميق في الرعاية الصحية تم حلها وشرحها باستخدام بايثون

ترجمة واعداد: د. علاء طعيمة

بـــسم تعالى

التعلم العميق ولـــتخداماته في الرعاية الصحية

25 مشروع تعلم عميق تم حلها وشرحها باستخدام بايثون

ترجمة واعداد:

د. علاء طعيمة

مقدمة المؤلف

يُستخدم التعلم العميقفي العديد من المجالات حول العالم. صناعة الرعاية الصحية ليست استثناء. يمكن أن يلعب التعلم العميق دورًا مهمًافي التنبؤ بوجود / عدم وجود اضطرابات حركية وأمراض القلب والسرطان وأمراض الرئة وغير ذلك.

يمكن لمثل هذه المعلومات، إذا تم التنبؤ بها مسبقًا، أن توفر معلومات مهمة للأطباء الذين يمكنهم بعد ذلك تكييف تشخيصهم وعلاجهم لكل مريض.

سيكون هذه الكتاب مفيداً للغاية لأولئك الذين يعملونفي مجال الذكاء الاصطناعي الذين يحاولون شق طريقهمفي مجال الطب والرعاية الصحية ولأولئك الذين يعملونفي مجال العلوم الطبية والرعاية الصحية الذين يحاولون شق طريقهمفي مجال الذكاء الاصطناعي.

في هذا الكتاب، تنقل مشاريع التعلم العميق باستخدام بايثون كل المعرفة اللازمة لتنفيذ مشاريع التعلم العميق المعقدةفي مختلف مجالات الرعاية الصحية. كل مشروع من هذه المشاريع فريد من نوعه، مما يساعدك على إتقان الموضوع تدريجيًا. ستتعلم كيفية تشخيص مجموعة متنوعة من الامراض باستخدام نماذج التعلم العميق وكذلك ستتعلم الكثير من المشاريع الجذابة الاخرى الي ستساعدكفي اكتساب المعرفة لتطوير أنظمة الرعاية الصحية باستخدام التعلم العميق.

لقد حاولت قدر المستطاع ان اترجم المشاريع الأكثر طرحاًفي مجال التعلم العميق للرعاية الصحية مع الشرح المناسب والكافي، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذا كان 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).

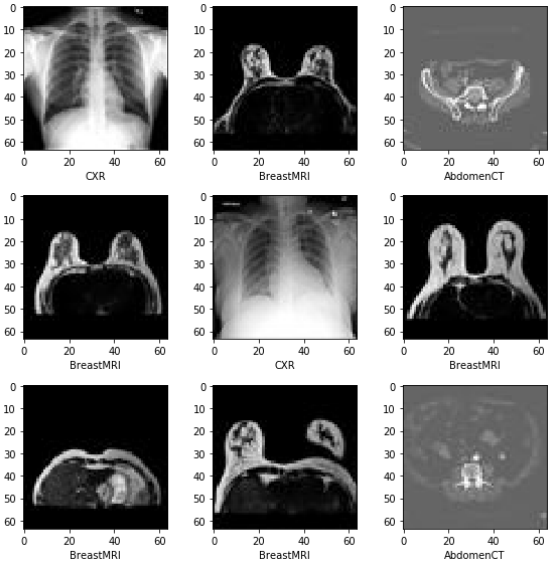
هنا، قمت بتدريب نموذج CNN باستخدام مجموعة بيانات جيدة المعالجة للصور الطبية. يمكن استخدام هذا النموذج لتصنيف الصور الطبية بناءً على الفئات المقدمة وفقًا لمجموعة بيانات التدريب.

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 البيانات في نموذج CNN، نحتاج أولاً إلى استخلاصها في مجموعة التدريب والاختبار منه.

هنا قمنا بالفعل بإنشاء دليل فارغ باسم "test_dir" مع جميع المجلدات الفرعية فيه كما هو الحال في "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

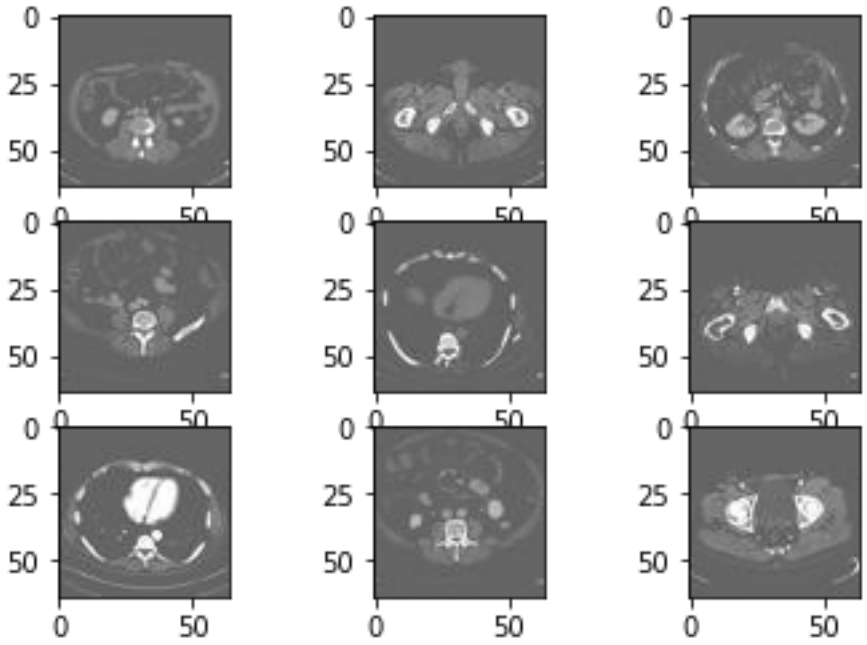
```
training_set_distribution = [len(os.listdir(os.path.join(train_dir, dir)))
for dir in os.listdir(train_dir)]
testing_set_distribution = [len(os.listdir(os.path.join(test_dir, dir)))
for dir in os.listdir(test_dir)]
```

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
```

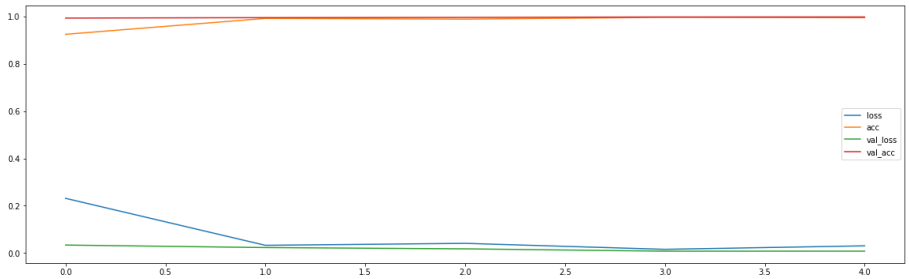
حقق النموذج دقة استثنائية بلغت 99٪في مجموعة التحقق و99٪ من الدقةفي مجموعة الاختبار.

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

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]

__________________________________________________________________________________
_________________________
```

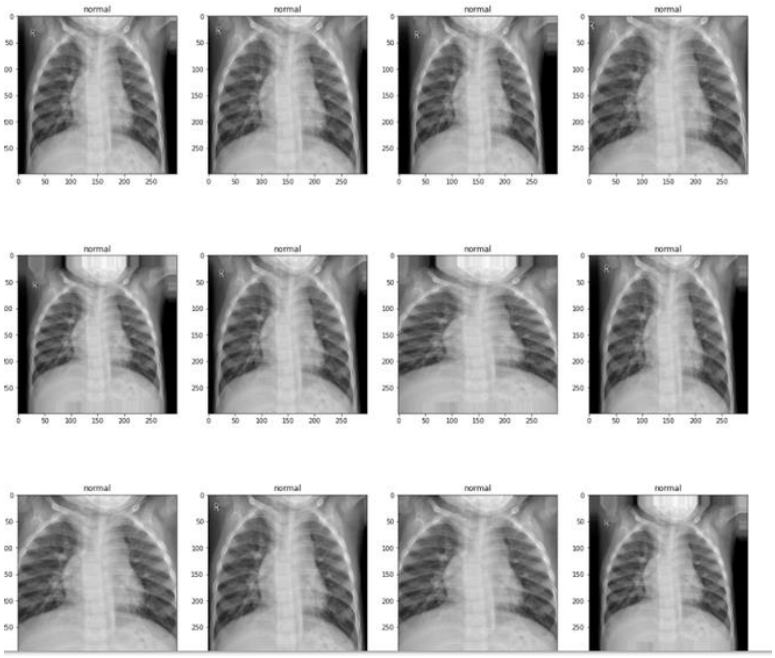
```
block2_sepconv2 (SeparableConv2 (None, 147, 147, 128 17536
block2_sepconv2_act[0][0]

__________________________________________________________________________________
__________________________
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
```

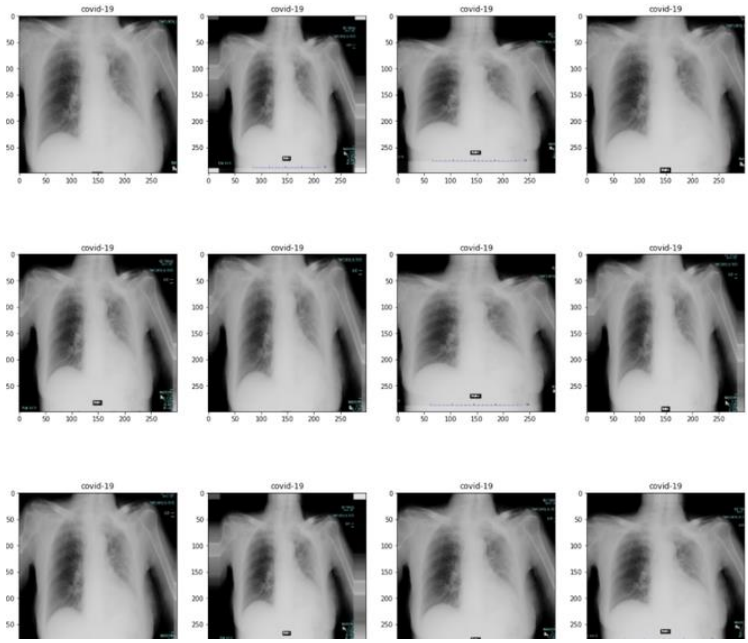
- الآن، نطبق بعض زيادة البيانات على مجموعة البيانات ونجهزها للتدريب. بعد ذلك، نرسم بعض الصور التدريبية. سنقسم مجموعة البيانات بحيث يكون لدينا 75٪ بيانات للتدريب و 25٪ للاختبار / التحقق من الصحة.

```
# 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',
```

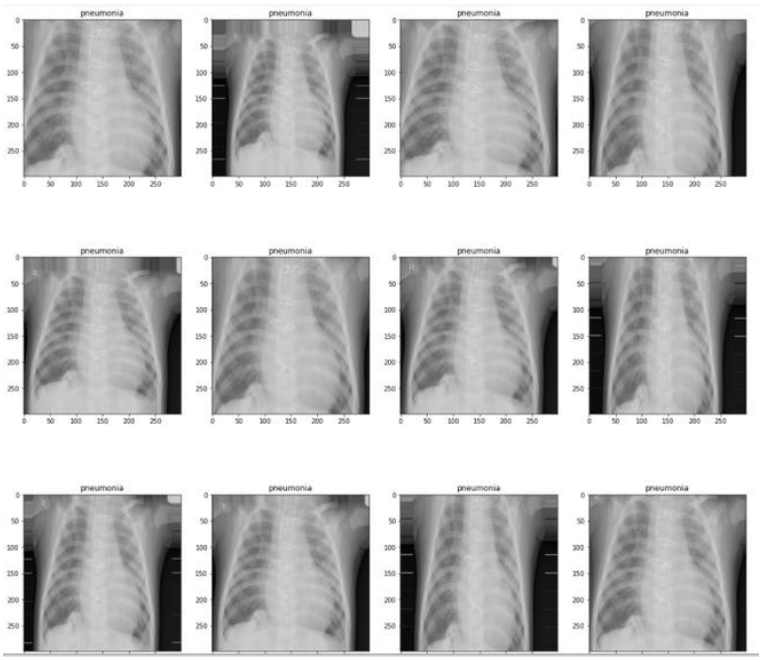
```
    subset='validation')
class_names=['covid-19','normal','pneumonia']

# 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 (الضبط الزائد 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.

3) 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
```

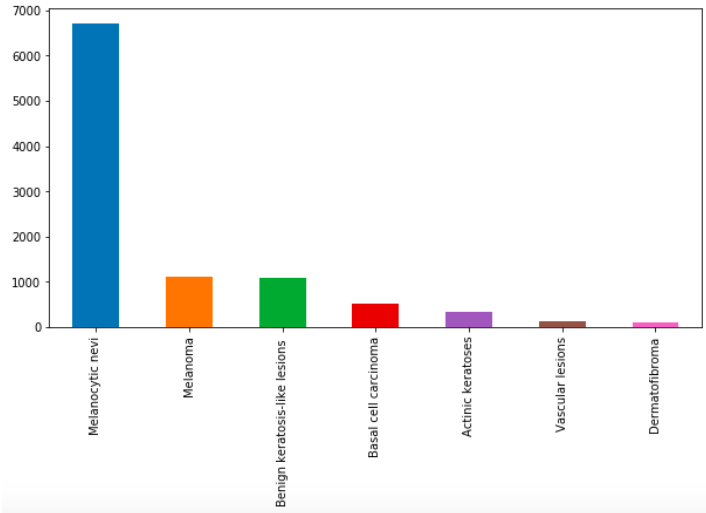
يتم التحقق من مجموعة البيانات بحثًا عن القيم أو أنواع البيانات المفقودة لكل حقل، ويتم تعبئتها بقيم خالية (بالنسبة للعمر،في هذه الحالة).

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)
```

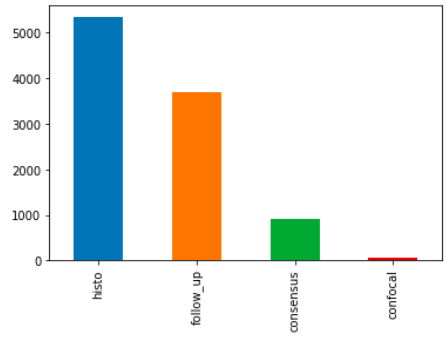
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7ae85ee48>
```



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

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

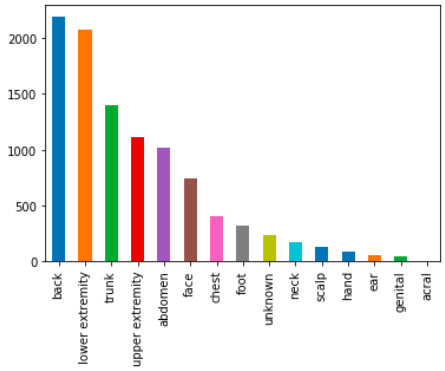
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7ae83fd30>
```



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

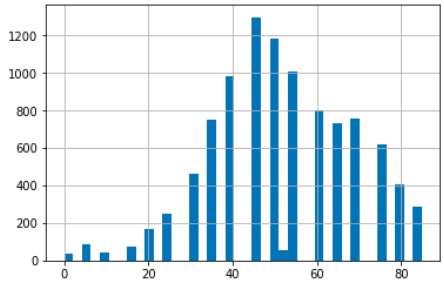
```
skin_df['localization'].value_counts().plot(kind='bar')
```

```
<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.

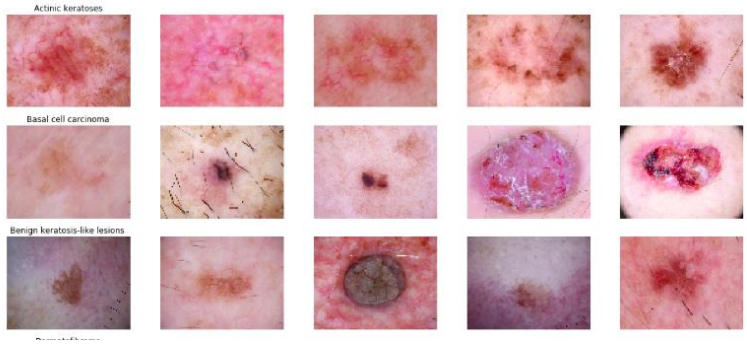
```
skin_df['image'] = skin_df['path'].map(lambda x: np.asarray(Image.open(x).resize((100,75
))))
```

```
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 المعياري.

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 – 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.

زيادة البيانات Data Augmentation ـ إنه اختياري. ولكن لتجنب مشكلة الضبط الزائد 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

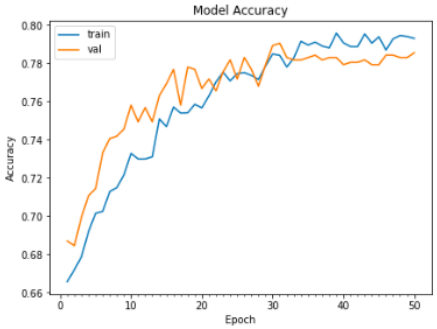
(دقة الاختبار والتحقق من الصحة، مصفوفة الارتباك، تحليل الحالات المصنفة بشكل خاطئ)

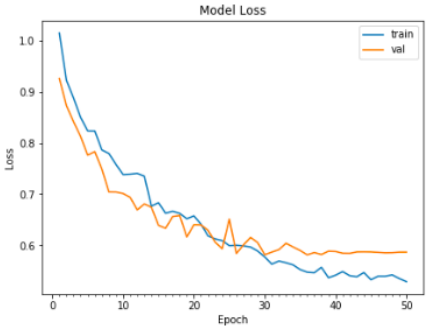
```
loss, accuracy = model.evaluate(x_test, y_test, verbose=1)
loss_v, accuracy_v = model.evaluate(x_validate, y_validate, verbose=1)
print("Validation: accuracy = %f  ;  loss_v = %f" % (accuracy_v, loss_v))
print("Test: accuracy = %f  ;  loss = %f" % (accuracy, loss))
model.save("model.h5")
```

```
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
```

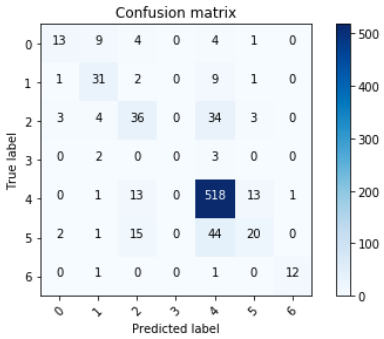
يتم فحص دقة الاختبار والتحقق من صحة نموذجنا، ويتم رسم مصفوفة الارتباك. يتم أيضًا تحديد عدد الصور المصنفة بشكل خاطئ لكل نوع.

```
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.

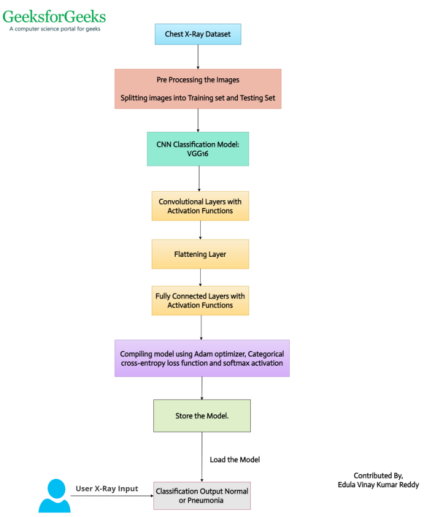
4) كشف الالتهاب الرئوي باستخدام التعلم العميق Pneumonia Detection using Deep Learning

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 X-ray 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 النموذج في 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',tar
get 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',tar
get_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 النتائجفي الفيديو أدناه:

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

5) 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.

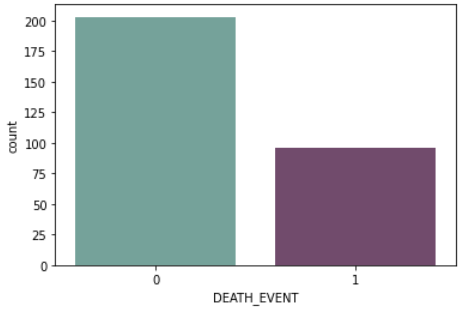
DEATH_EVENT: إذا توفي المريض خلال فترة المتابعة.

تحليل البيانات

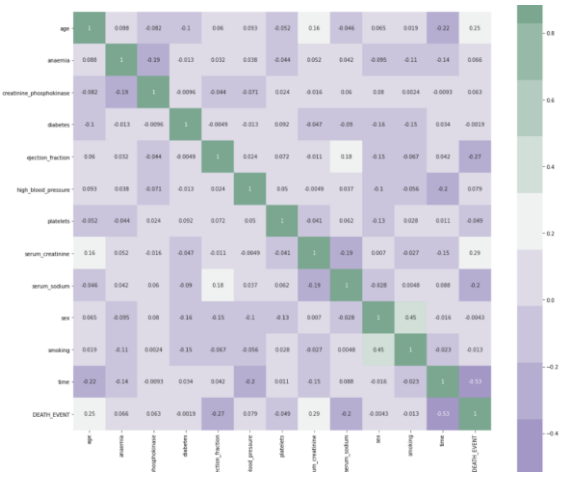
خطواتفي تحليل البيانات والتصوير:

نبدأ تحليلنا من خلال رسم مخطط عد لسمة target. 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:

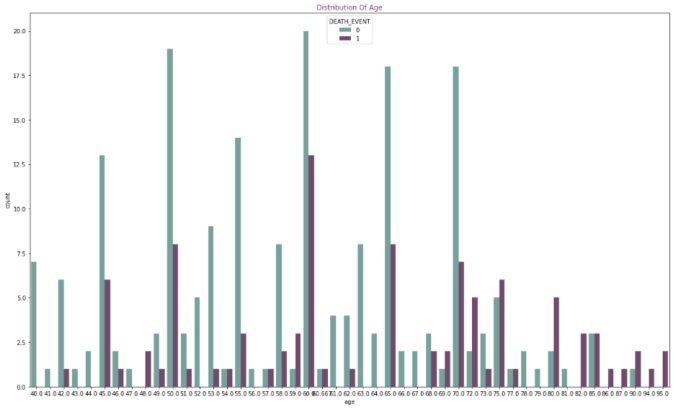
يعد وقت زيارة متابعة المريض للمرض أمرًا بالغ الأهمية لأن التشخيص الأولي لمشكلة القلب والأوعية الدموية والعلاج يقلل من فرص حدوث أي وفاة. إنها تحمل وعلاقة عكسية.

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

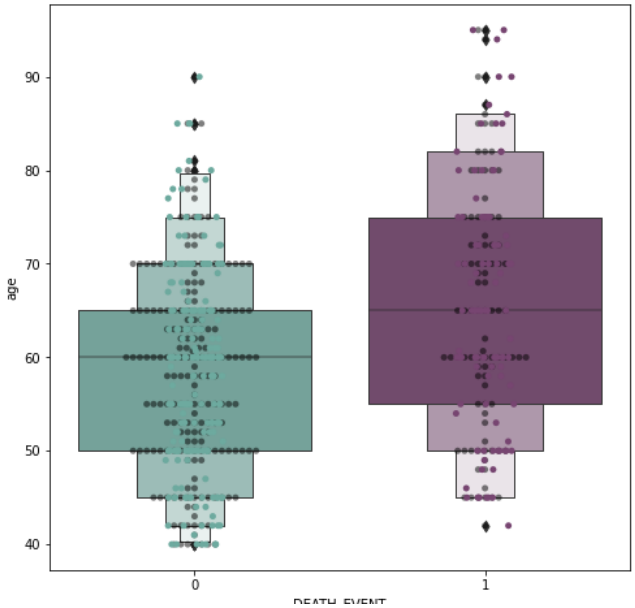
عمر المريض هو ثالث أكثر السمات ارتباطًا. من الواضح أن أداء القلب يتدهور مع تقدم العمر

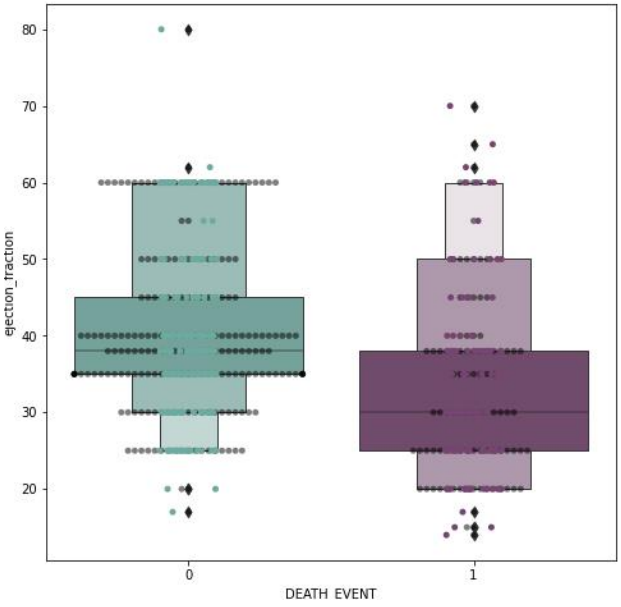
بعد ذلك، سوف نفحص رسم عد (countplot) للعمر.

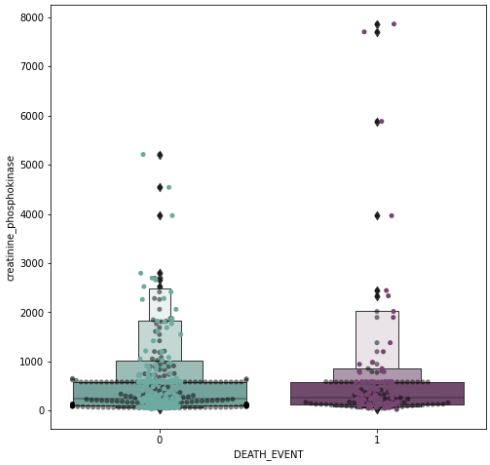
```
#Evauating age distribution
plt.figure(figsize=(20,12))
#colours =["#774571","#b398af","#f1f1f1" ,"#afcdc7", "#6daa9f"]
Days_of_week=sns.countplot(x=data['age'],data=data, hue
="DEATH_EVENT",palette = cols)
Days_of_week.set_title("Distribution Of Age", color="#774571")
```

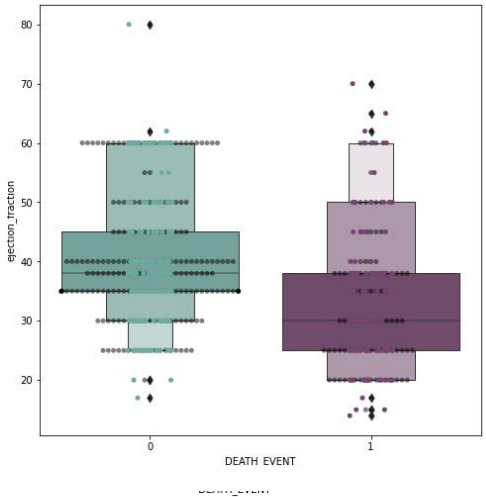


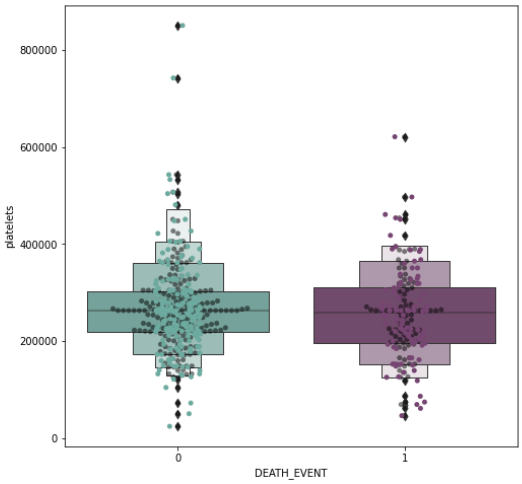
```
# Boxen and swarm plot of some non binary features.
feature =
["age","creatinine_phosphokinase","ejection_fraction","platelets","serum_c
reatinine","serum_sodium", "time"]
for i in feature:
    plt.figure(figsize=(8,8))
    sns.swarmplot(x=data["DEATH_EVENT"], y=data[i], color="black",
alpha=0.5)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

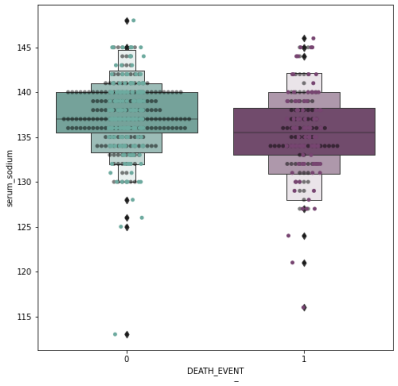


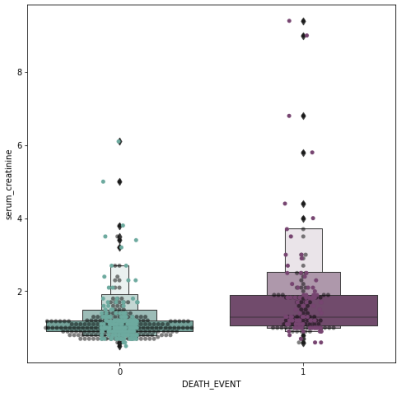



creatinine_phosphokinase
DEATH_EVENT


ejection_fraction
DEATH_EVENT


platelets
DEATH_EVENT

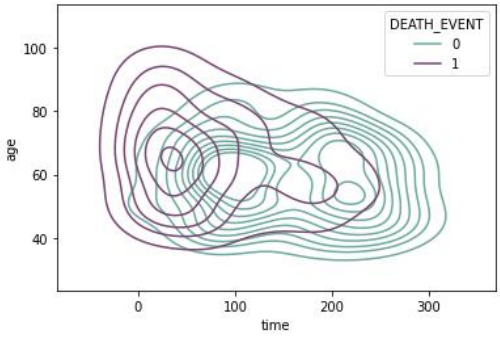




لقد رصدت القيم المتطرفة outliersفي مجموعة البيانات الخاصة بنا. لم أقم بإزالتها بعد لأنها قد تؤدي إلى الضبط الزائد overfitting. على الرغم من أننا قد ننتهي بإحصائيات أفضل.في هذه الحالة، مع البيانات الطبية، قد تكون القيم المتطرفة عاملاً حاسمًا مهمًا.

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

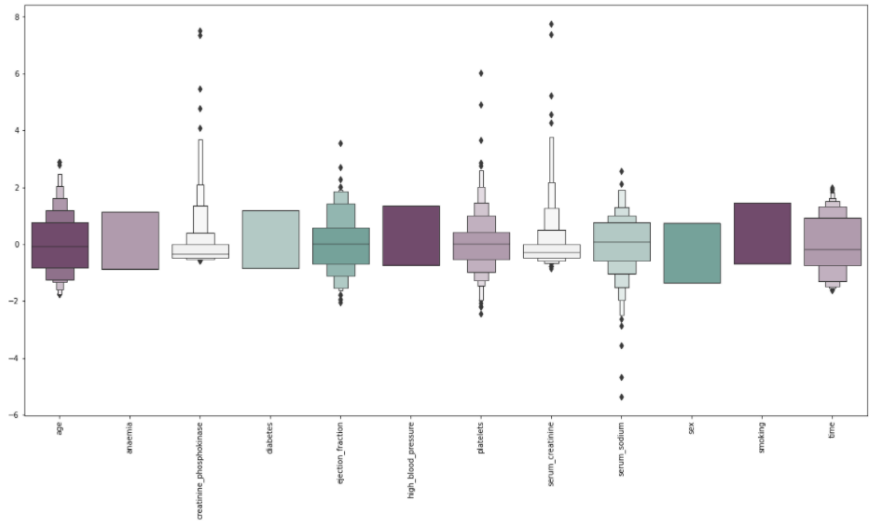
الخطوات المتبعةفي المعالجة المسبقة للبيانات:

- 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()
```

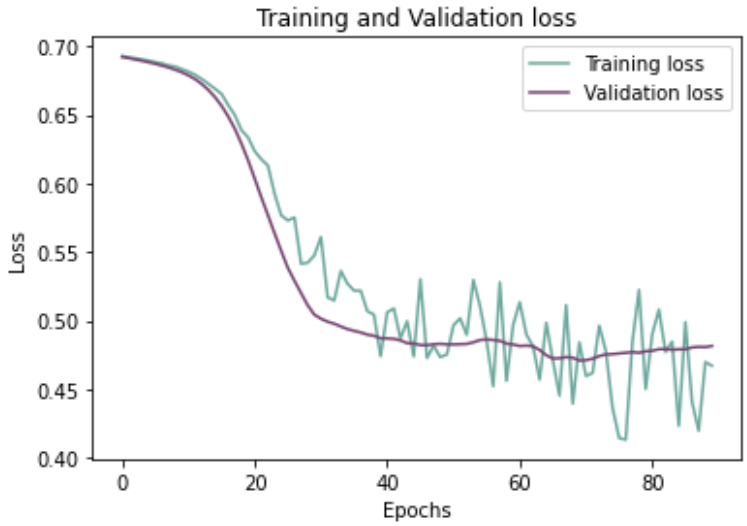
```
# layers
model.add(Dense(units = 16, kernel_initializer = 'uniform', activation =
'relu', input_dim = 12))
model.add(Dense(units = 8, kernel_initializer = 'uniform', activation =
'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 4, kernel_initializer = 'uniform', activation =
'relu'))
model.add(Dropout(0.5))
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))
```

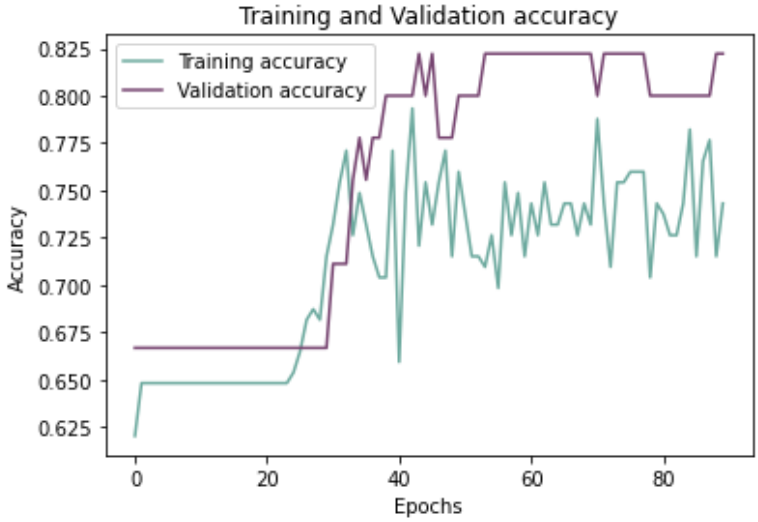
```
val_accuracy: 79.81%
```

```
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()
```



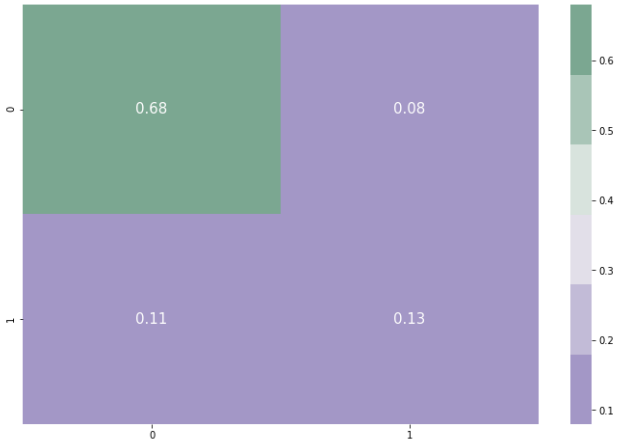
الاستنتاجات

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')
```

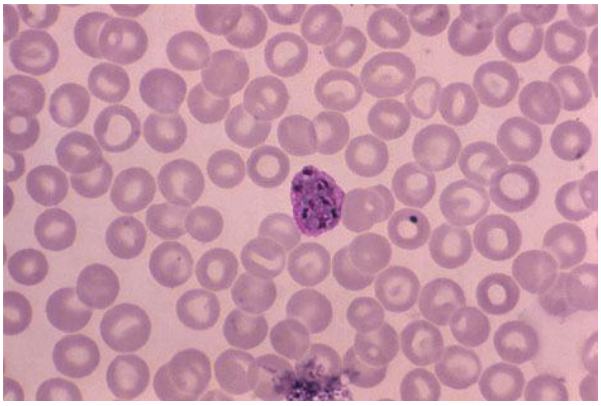
6) كشف الملاريا باستخدام التعلم العميق Malaria Detection using Deep Learning

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 المجلدات 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 وتسوية 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")
```

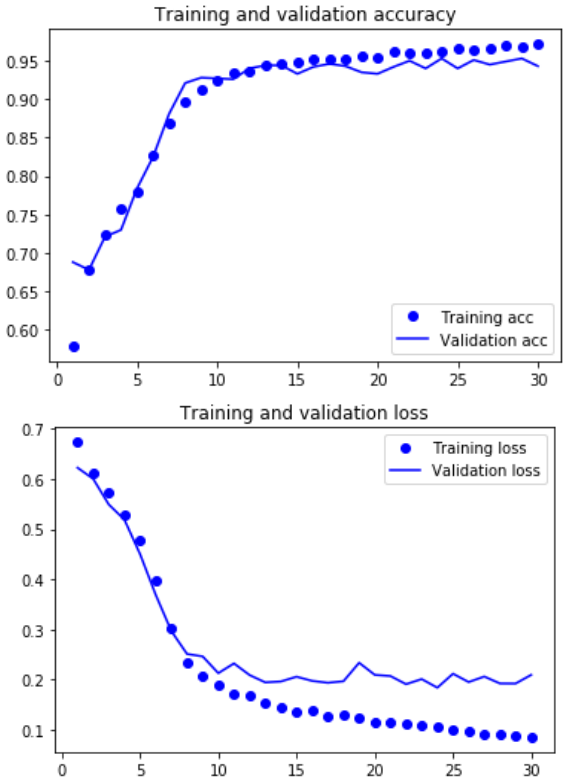
بعد الانتهاء من إعداد جميع البيانات، يمكننا الآن بناء شبكتنا. تتكون بُنية CNN الخاصة بنا من عدة طبقات تلافيفية convolutional layers لتمثيل الميزات وطبقات متصلة بالكامل fully-connected layers متبوعة بمصنف sigmoid للتصنيف الثنائي.

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")
```

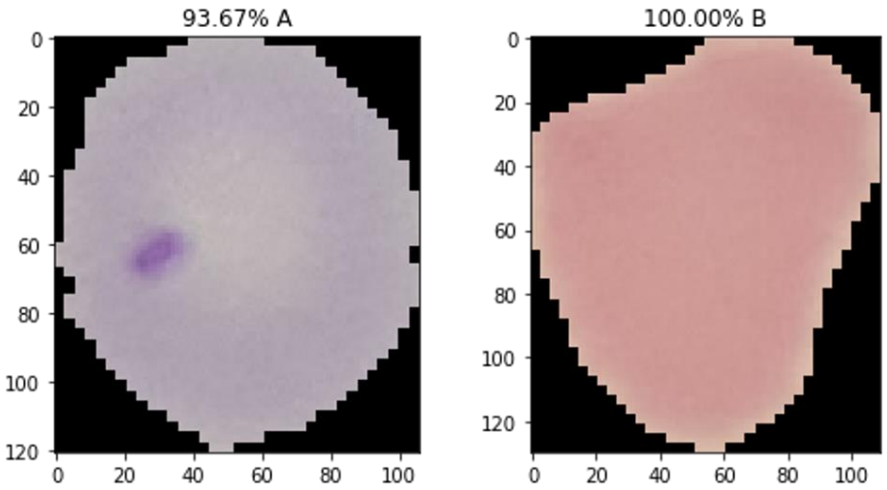
فلنبدأ الآنفي تدريب نموذجنا. سوف نتدرب لمدة 30 حقبة لتحقيق التوازن بين الدقة والكفاءة الحسابية. ثم يتم حفظ نموذجنا المُدربفي متغير السجل، حيث يمكننا من خلاله رسم مقاييس الخطأ والدقة.

```
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 بدأنا في 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. جرب واكتشف ما إذا كان بإمكانك تحسينه، وصنع أداة قابلة للتطبيق لإنقاذ الأرواح!

7) 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 (RNN) والشبكات العصبية التلافيفية(CNN) وشبكات الاعتقاد العميق (DBN) 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 (CPU) ووحدة معالجة الرسومات (GPU). 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 277،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 وy 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.

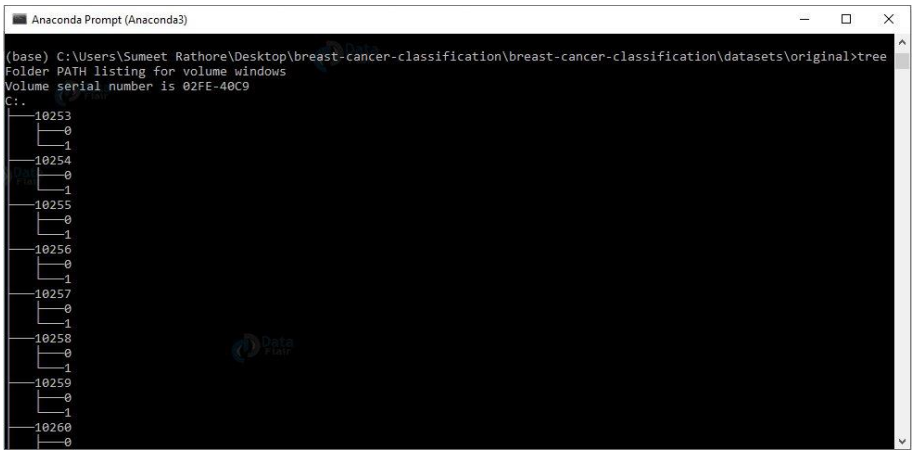
الخطوة 2: الآن، داخل دليل تصنيف سرطان الثدي الداخلي، أنشئ مجموعات بيانات الدليل- بداخله، أنشئ الدليل الأصلي:

```
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 tree:

```
cd breast-cancer-classification\breast-cancer-
classification\datasets\original
tree
```

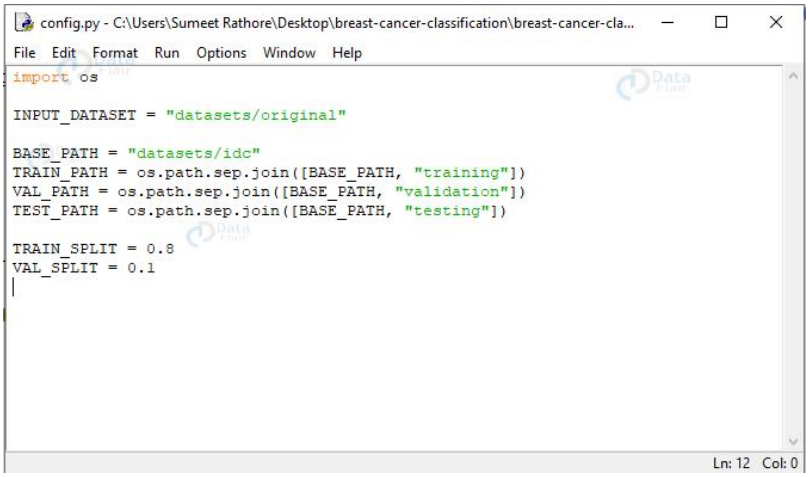


لدينا دليل لكل مريض معرف patient ID. وفي كل دليل من هذا القبيل، لدينا دليلين 0 و1 للصور ذات المحتوى الحميد والخبيث.

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 ـ 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

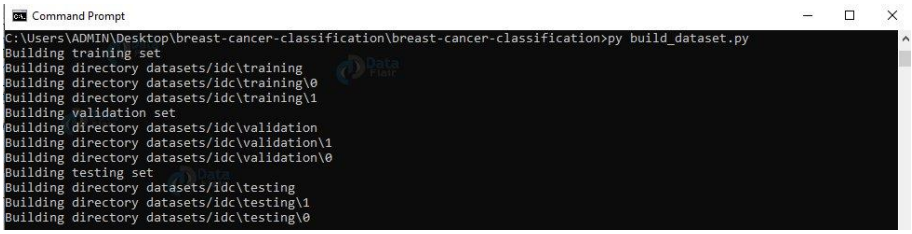
في هذا، سنستورد من config وimutils وrandom وshutil وos. سننشئ قائمة بالمسارات الأصلية للصور، ثم نغير القائمة. بعد ذلك، نحسب فهرسًا بضرب طول هذه القائمةفي 0.8 حتى نتمكن من تقسيم هذه القائمة للحصول على قوائم فرعية لمجموعات بيانات التدريب والاختبار. بعد ذلك، نحسب أيضًا فهرسًا يوفر 10٪ من القائمة لمجموعة 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) , 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=["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)

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')
```

```
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

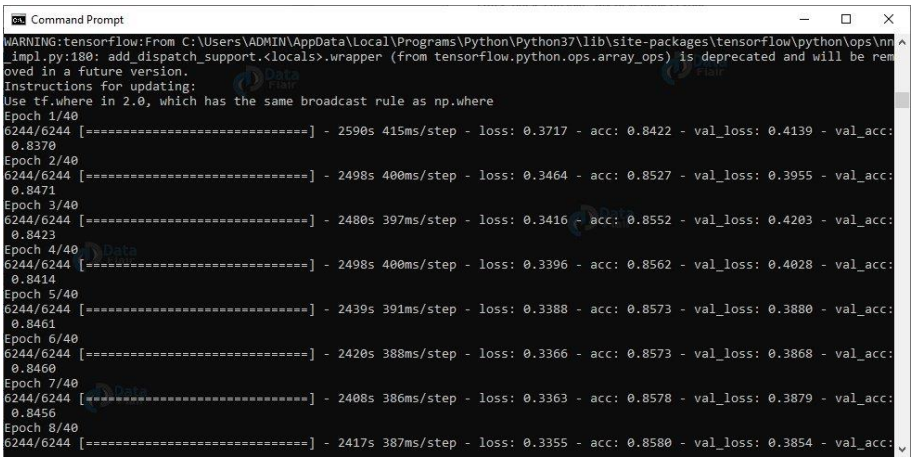
للتدريب والتحقق من الصحة والاختبار. بعد ذلك، سنحصل على وزن الفئة لبيانات التدريب حتى نتمكن من التعامل مع عدم التوازن.

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.

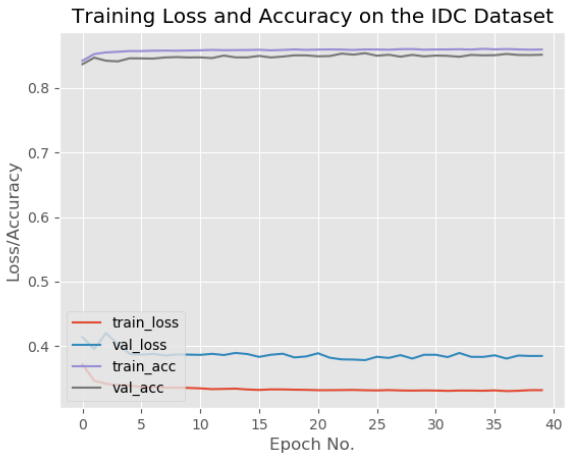
سنقوم بتهيئة مولدات التدريب والتحقق والاختبار حتى يتمكنوا من إنشاء مجموعات من الصور بحجم batch_size. بعد ذلك، سنقوم بتهيئة النموذج باستخدام مُحسِّن Adagrad وتجميعه باستخدام دالة خطأ binary_crossentropy. الآن، للتدريب fit النموذج، نقوم بإجراء استدعاء لـ fit_generator().

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.

8) 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()
```

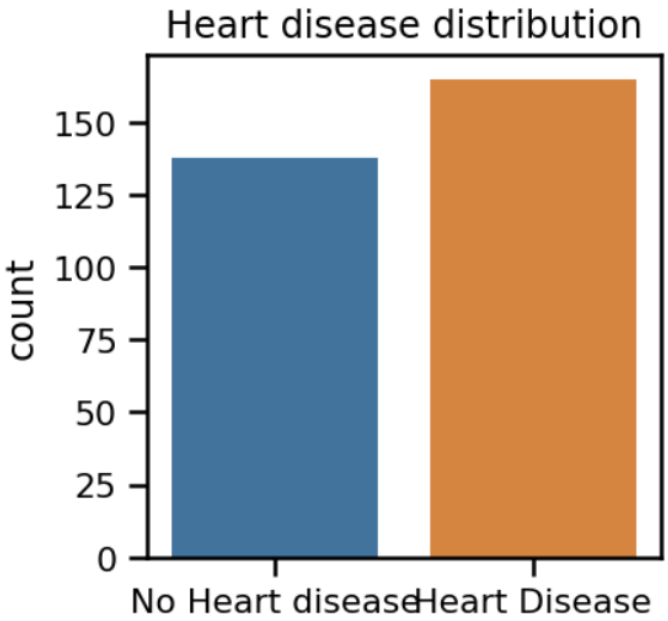
قم باستيراد مجموعة البيانات إلى pandas DataFrame وطباعة أول 20 سجلًا.

```
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]
```

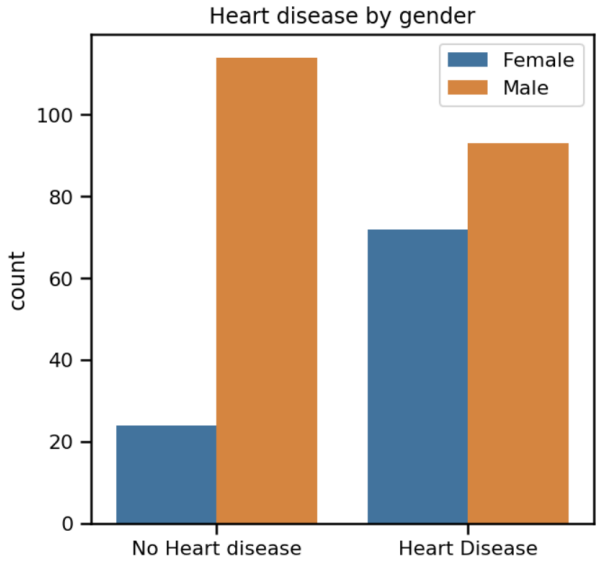
مع استيراد البيانات بنجاح، دعنا نرسم التوزيع بين أمراض القلب وغيابها، المشار إليها بواسطة الميزة الهدف.

```
f = sns.countplot(x='target', data=Data)
f.set_title("Heart disease distribution")
f.set_xticklabels(['No Heart disease', 'Heart Disease'])
plt.xlabel("");
```



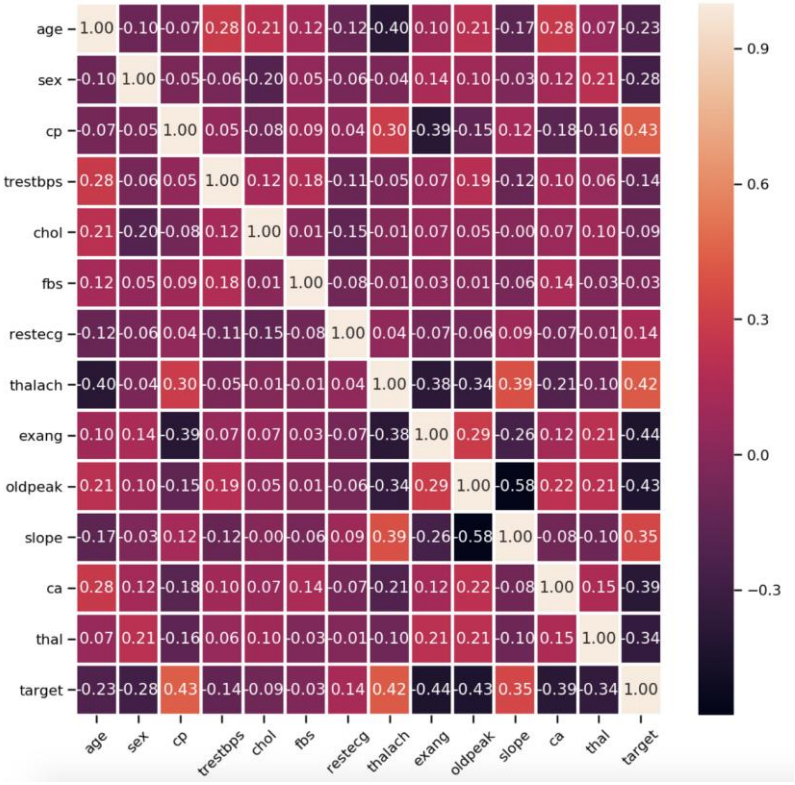
يمكننا أن نرى من الرسم البياني أعلاه أن التوزيع بين أمراض القلب الإيجابية والسلبية هو نفسه تقريبًا. إعداد جيد لمشكلة DL للتصنيف الثنائي.

```
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 sigmoidالمناسبة لمشاكل التصنيف الثنائي.

دعونا نجمع (compile)وندرب (fit)النموذج:

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accura
cy'])
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 ثنائيفي 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 هنافي github repo، فلا تترددفي تنزيله وتجربته.

9) 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

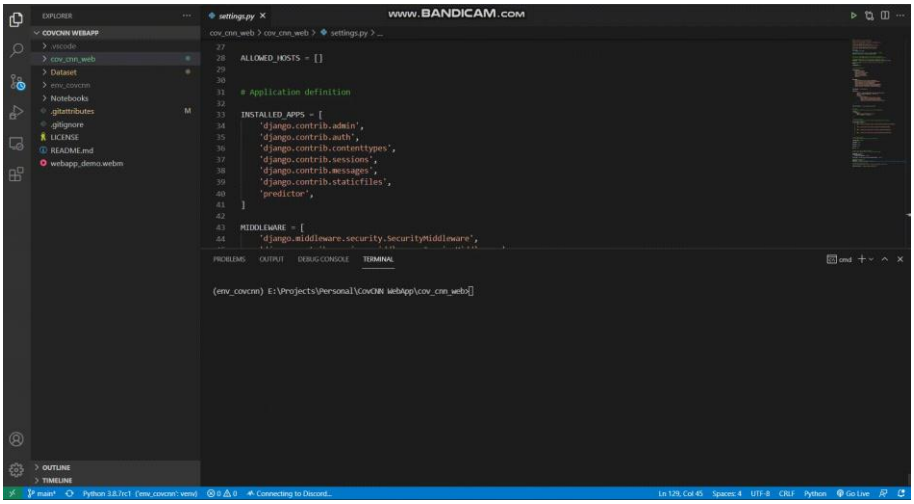
```
<form method="post" id="imageForm" enctype="multipart/form-data">
   {% csrf token %}
   <label for="ImgFile">Upload Image</label>
   <input type="file" name="ImgFile" class="form-control"/>
   <input type="submit" id="submitButton" class="btn" name="submit"
value="Solve"/>
</form>
```

- 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 CSS (Bootstrap) 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



10) تشخيص مرض الزهايمر باستخدام التعلم العميق Alzheimer Diagnosis using Deep Learning

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: انحدار 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/'
```

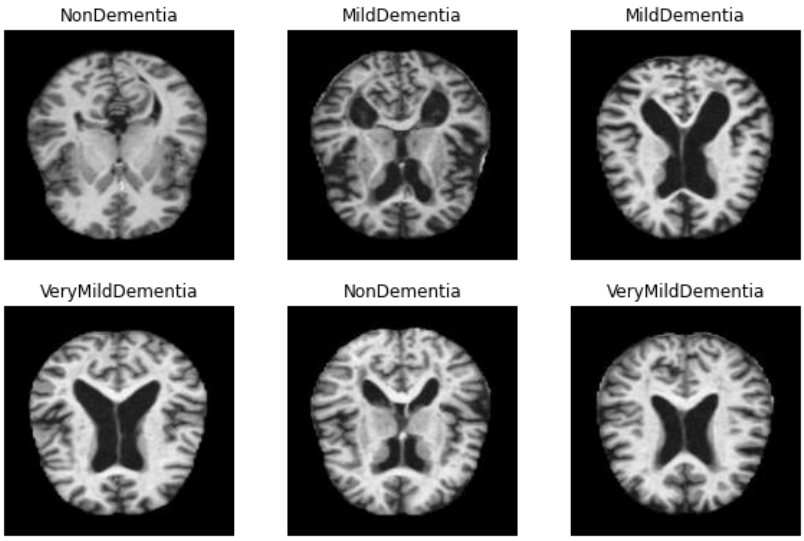
```
val_data = tf.keras.preprocessing.image_dataset_from_directory(
    val,
    validation_split=0.2,
    image_size=(224,224),
    batch_size=32,
    subset='validation',
    seed=1000
    )
```

```
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
```

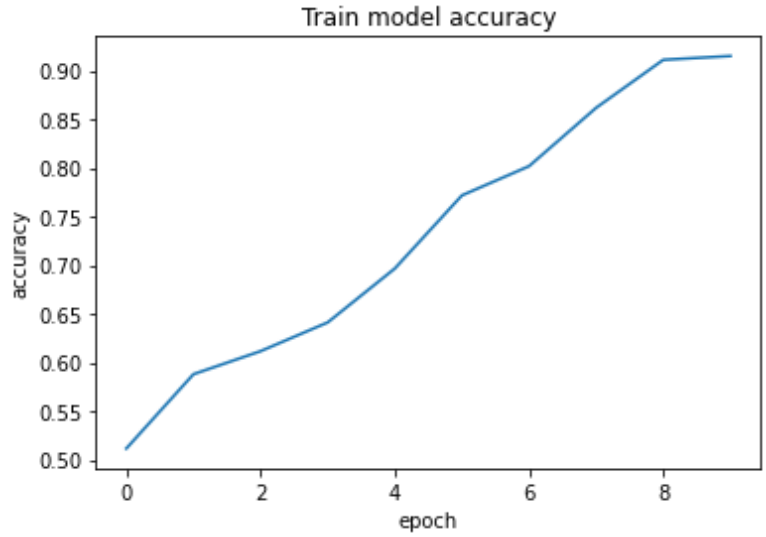
حفظ النموذج:

```
model.save("alz_model1.h5")
```

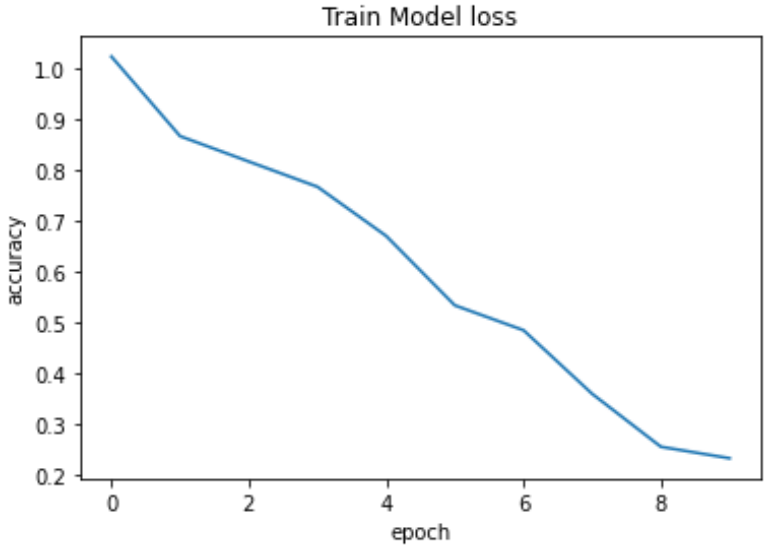
```
model.evaluate(val_data)
```

رسم النتائج:

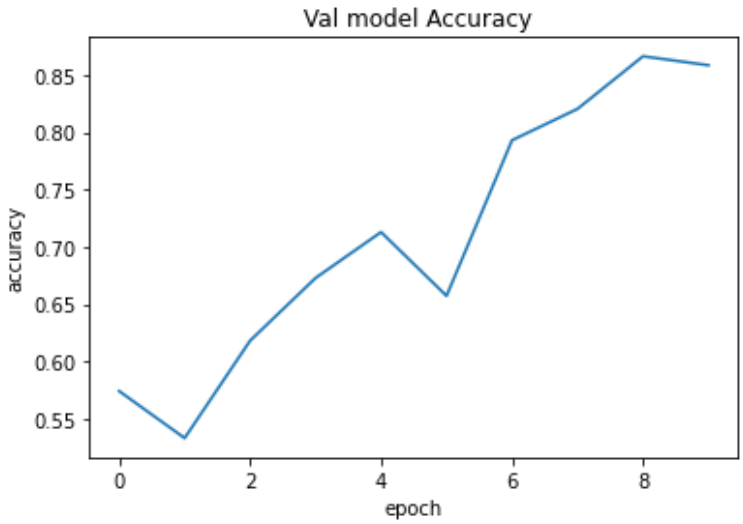
```
plt.plot(history.history['accuracy'])
plt.title('Train model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



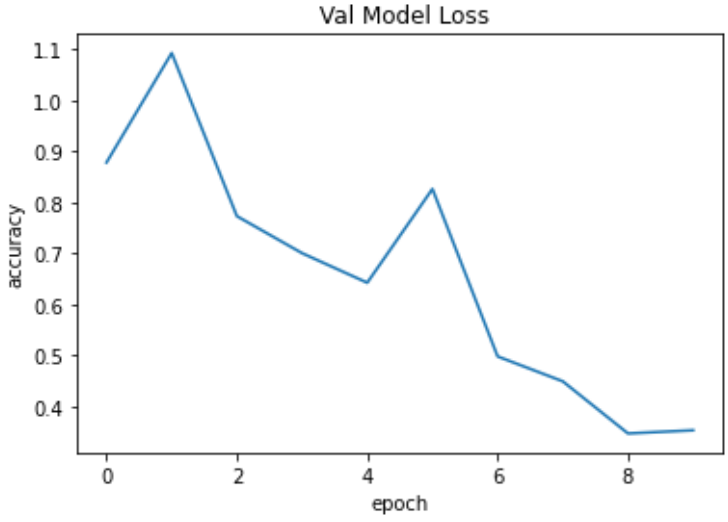

```
plt.plot(history.history['loss'])
plt.title('Train Model loss')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```




```
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()
```

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!!!!!
```

11) 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
```

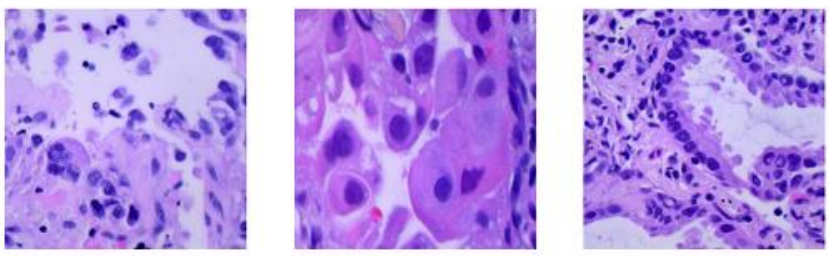
المخرجات:

```
['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 الترميز واحد_ساخن 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()])
```

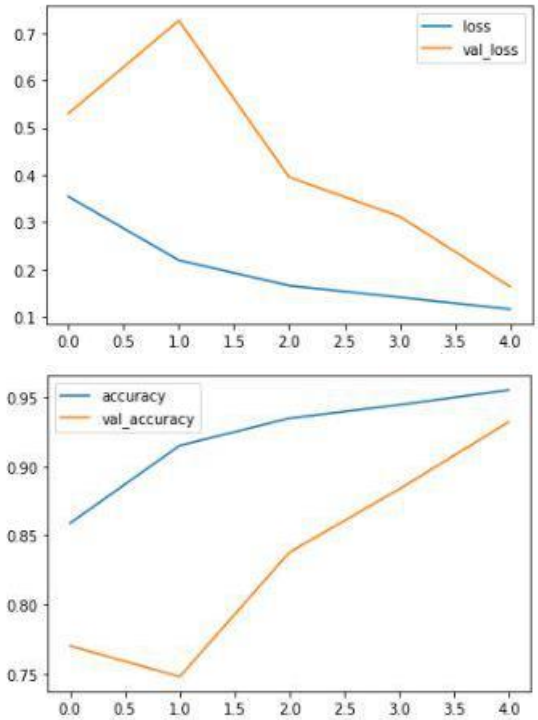
المخرجات:

```
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.

12) 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 Bonn. 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()
```

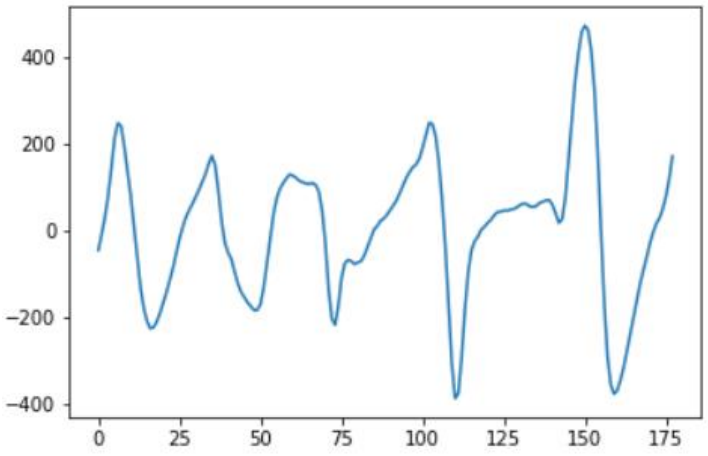
```
<class 'pandas.core.frame.DataFrame'>
Index: 11500 entries, X21.V1.791 to X16.V1.210
Columns: 179 entries, X1 to y
dtypes: int64(179)
```

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
```

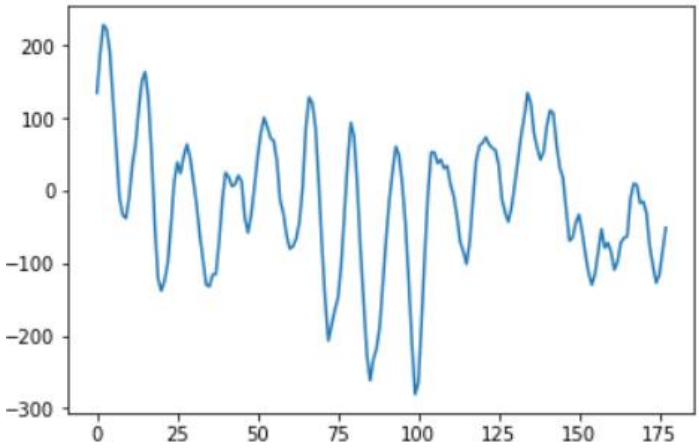
دعونا نرسم ونلقي نظرة على بعض أشكال الموجات EEG.

```
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 الضبط الزائد 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)
```

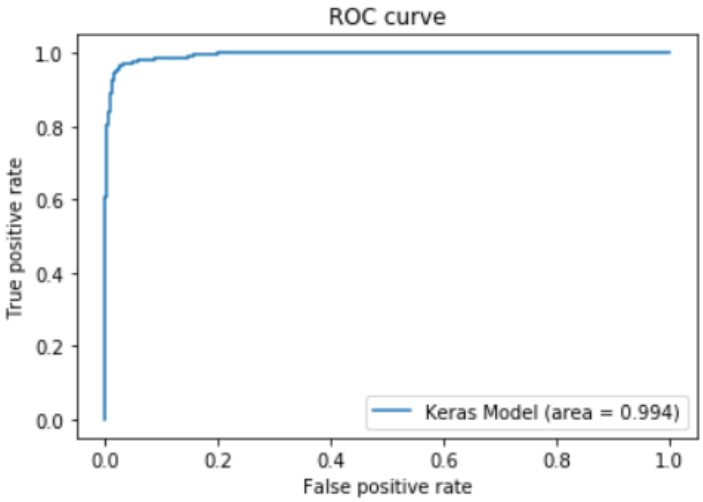
الآن، دعونا نجمع compile النموذج وندربه على 100 حقبة.

```
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 96ـ97٪.

أخيرًا، سنطلق العنان للنموذجفي 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.

13) 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.

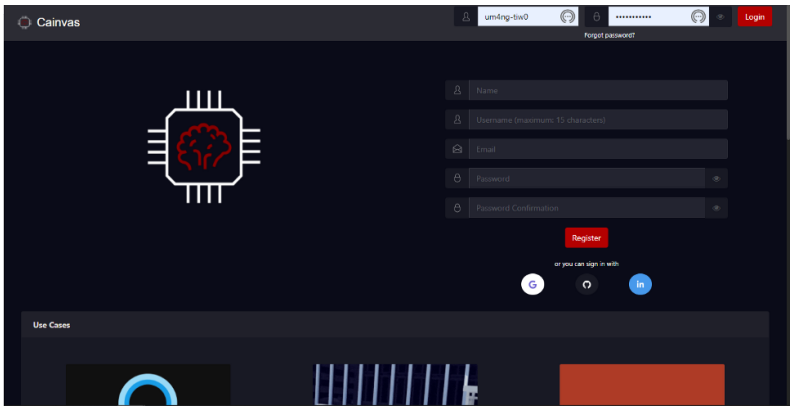
تحتوي مجموعة البيانات على 12500 صورة ملونة مكثفة لخلايا الدم بتنسيق JPEG. هناك 4 فئات مختلفة من خلايا الدم البيضاء ـ الحامضية Eosinophil، واللمفية Lymphocyte، والوحيدة Monocyte، والمتعادلة Neutrophilفي مجموعة البيانات.

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.

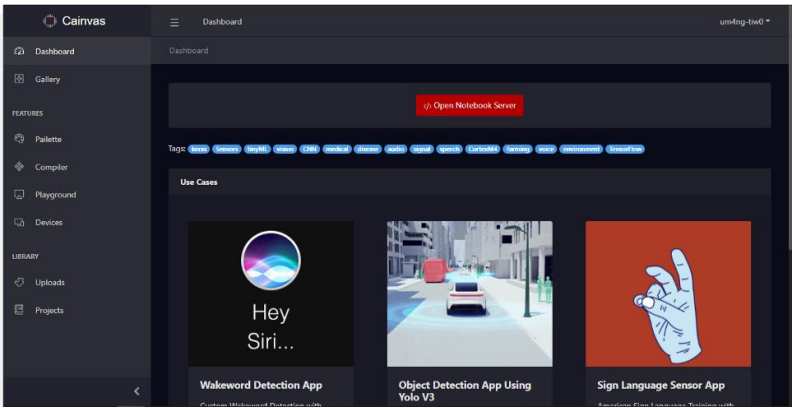
إنشاء المنصة

يمكنك إنشاء حساب على موقع Cainvas هنا.

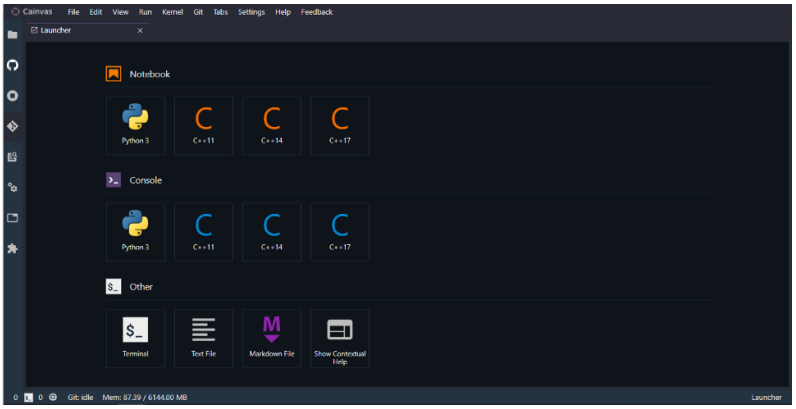


Cainvas landing page

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.

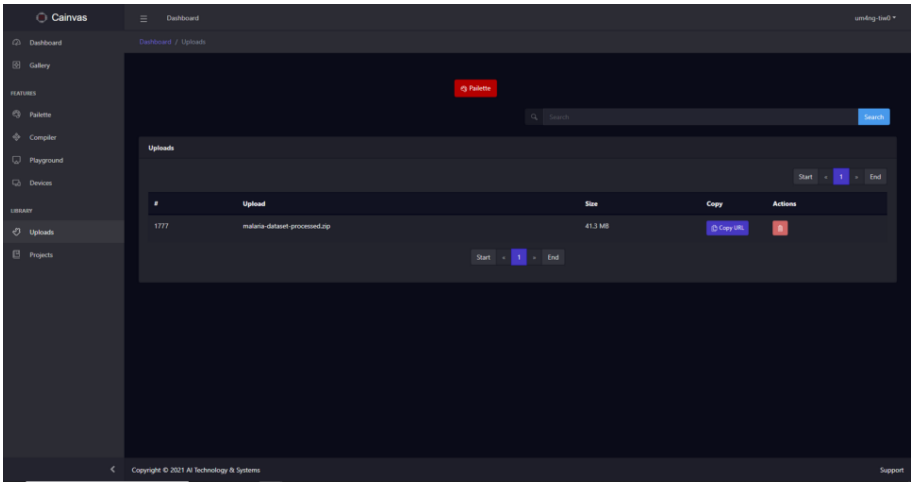
تشمل المكتبات الأخرى Tensorflow و Keras لإنشاء الشبكة العصبية التلافيفية CNN وإجراء المعالجة المسبقة للبيانات لأداء التدريب عليها.

```
#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

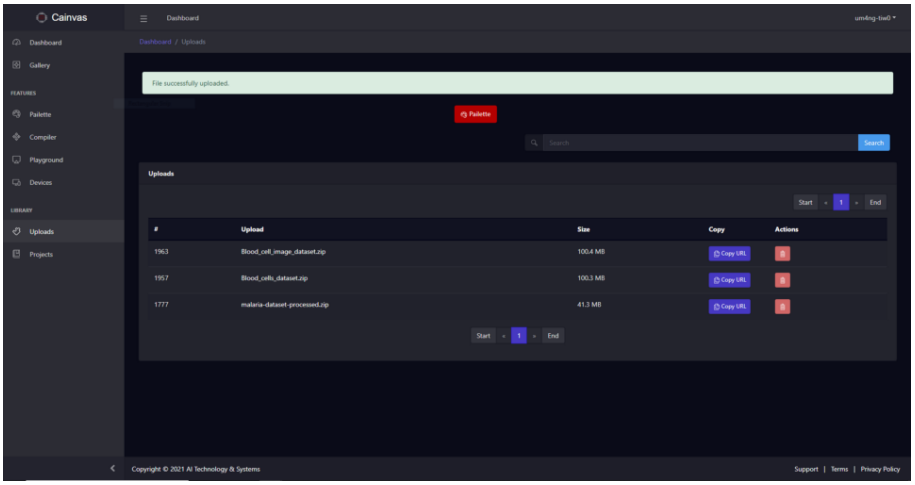
تتيح لنا منصة Cainvas تحميل مجموعات البيانات على المنصة مما يسهل الاستخدام. يمكن بعد ذلك تحميل مجموعات البيانات هذه بسهولة على notebook واستخدامها بمرونة كافية لإنشاء النموذج دون أي متاعب.

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.



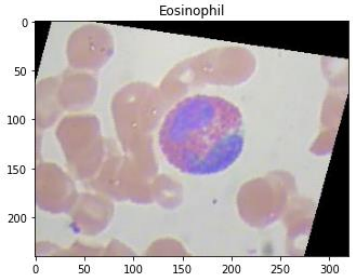
قسم التحميلات مع البيانات التي تم تحميلها

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 – 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/TRA
IN/",
                                                    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/TRA
IN/",
                                                        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)
```

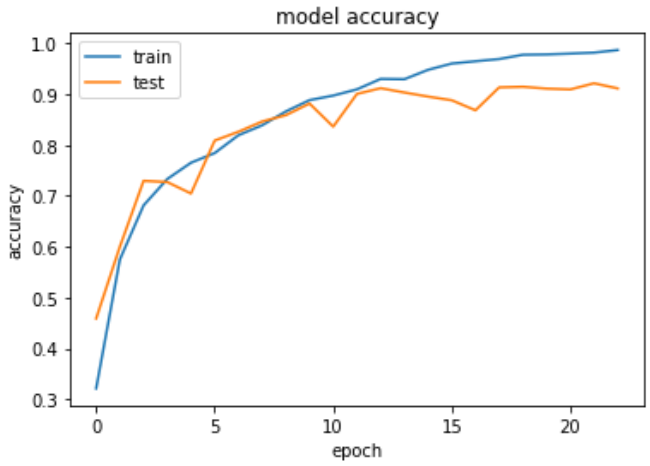
الحقب الثلاثة الأخيرة من مرحلة التدريب:

```
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

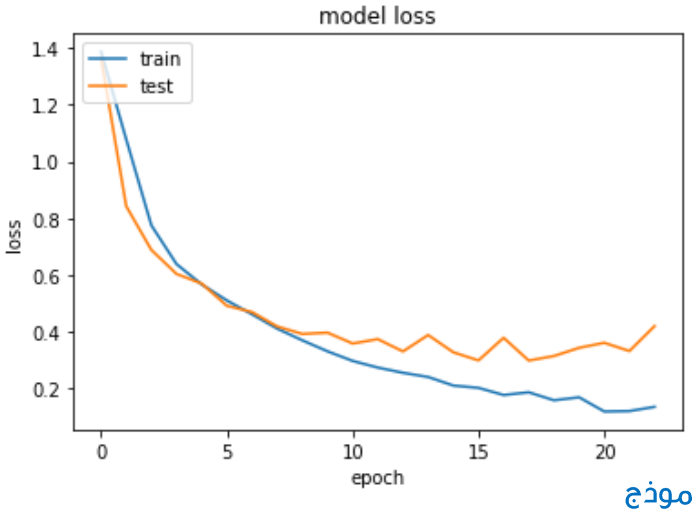
سنقوم برسم أداء النموذج في كل فترة خلال مرحلة التدريب.

```
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/TES
T/",
                                                    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.

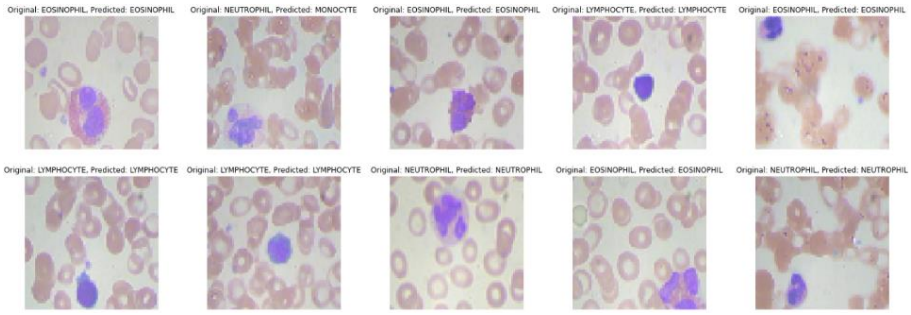
```
# Getting the predicted classes from one hot encoded predicted outputs

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.

14) 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.

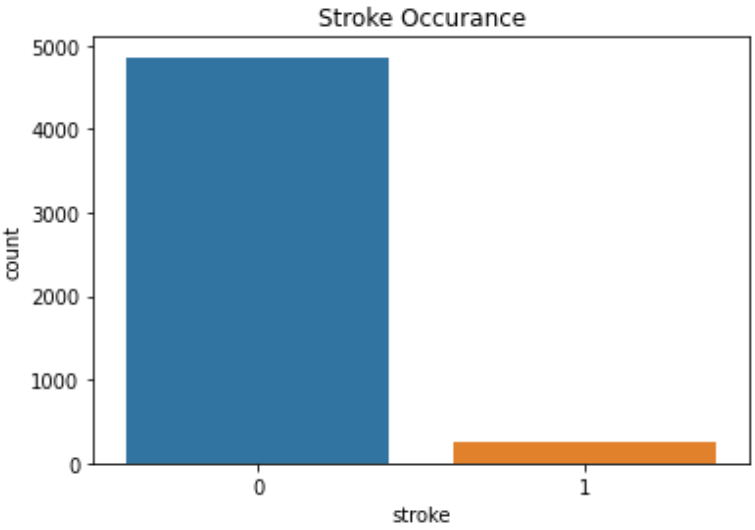
```
data['bmi'] = data['bmi'].fillna(np.mean(data['bmi']))
```

رسم البيانات

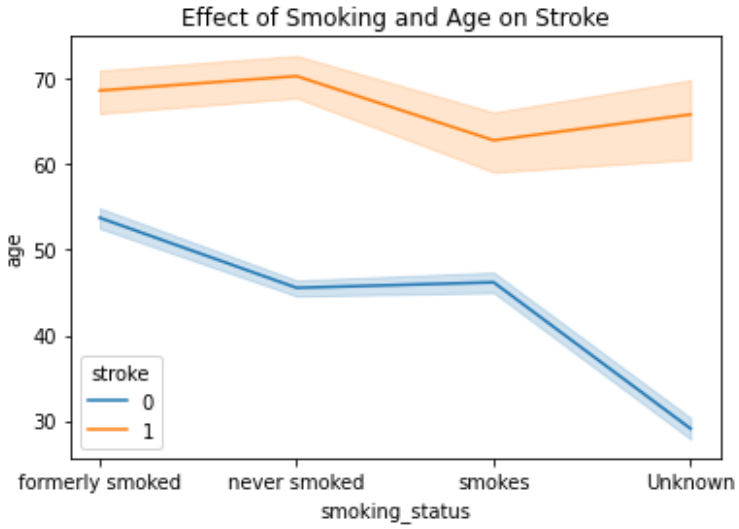
نتعامل مع العديد من المخططات لفهم العلاقة بين البيانات باستخدام مكتبة seaborn وmatplotlib. البدء بـ countplot للتحقق من توزيع بياناتنا. نلاحظ أن بياناتنا غير متوازنة إلى حد كبير مع زيادة تمثيل أحد الفئات عن الآخر. علاوة على ذلك، ندرس تأثير التدخين والعمر على حدوث السكتة الدماغية. من خلال تحليل الرسم 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.

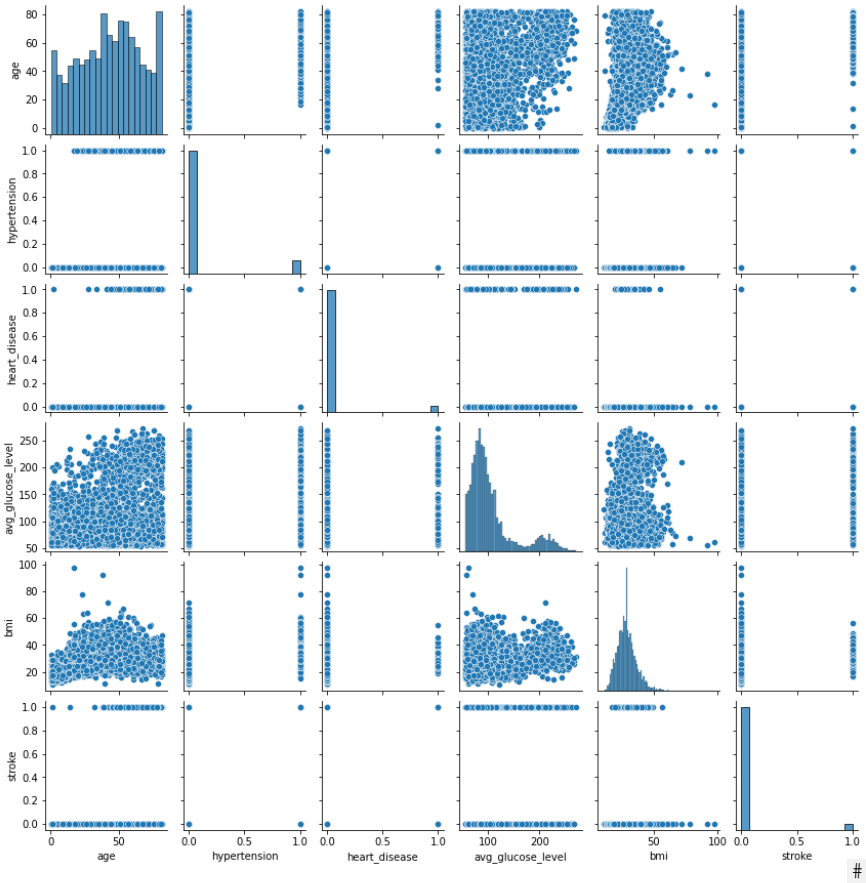
```
sns.countplot(data = data, x = 'stroke')
plt.title("Stroke Occurance")
```




```
sns.lineplot(data = data, x = 'smoking_status', y = 'age', hue = 'stroke')
plt.title("Effect of Smoking and Age on Stroke")
```

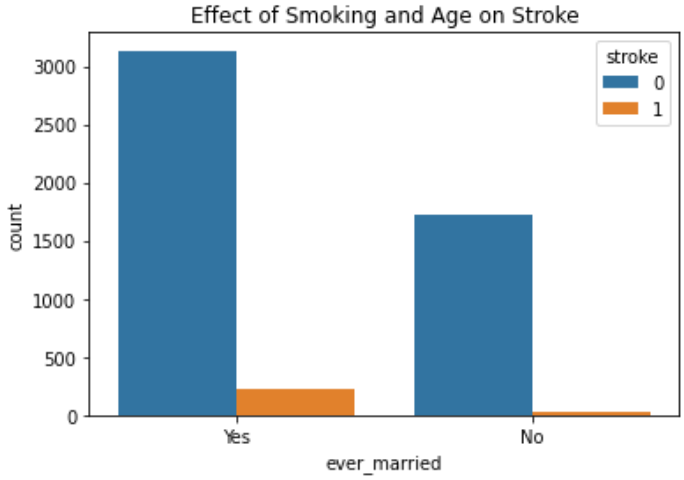



```
# Visualising the relationship between different columns of the data
sns.pairplot(data, height = 2)
plt.show()
```



#

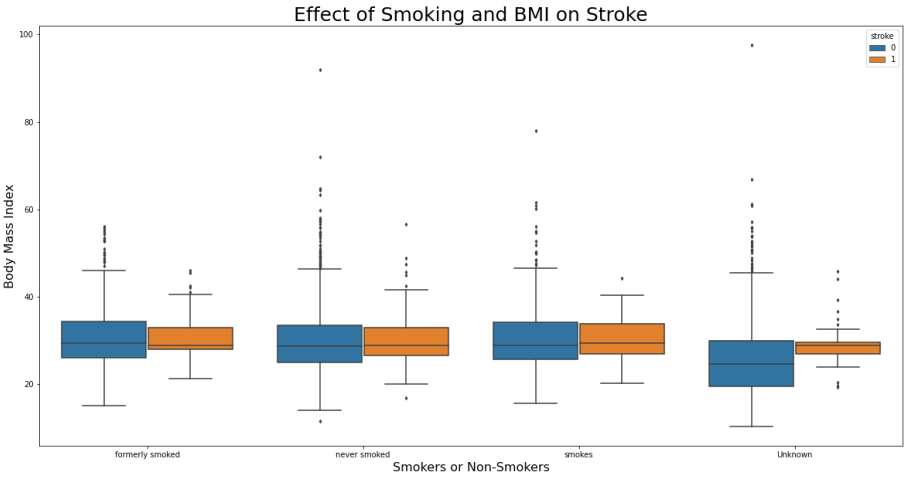
```
Counting the occurance of stroke in the data
sns.countplot(data = data, x = 'ever_married',  hue = 'stroke')
plt.title("Effect of Smoking and Age on Stroke")
```



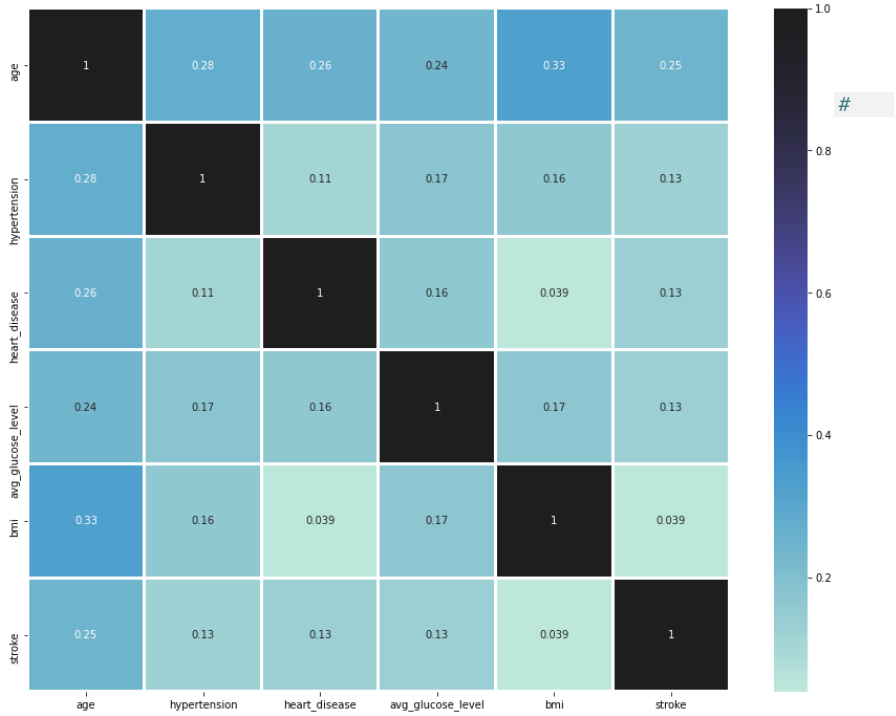
يتبع ذلك المخطط الصندوقي boxplot الذي يساعدنا على فهم تأثير مؤشر كتلة الجسم (BMI) وحالة التدخين على السكتات الدماغية. يعد boxplot طريقة مفيدة للغاية لتوزيع البيانات وفقًا للإجراءات التالية:

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
```

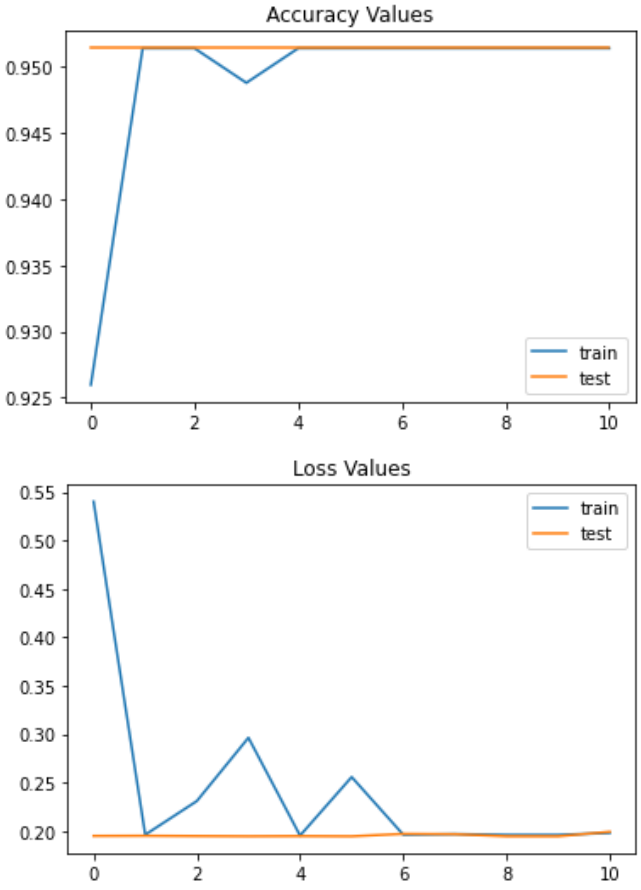
رسم النتائج

من أجل تدريب نموذجنا ومنع أي الضبط الزائد 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')
```



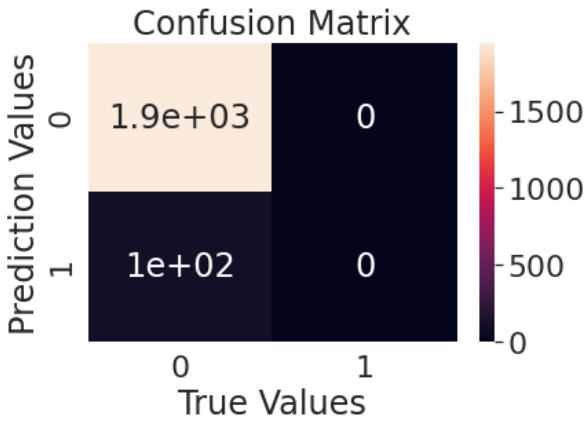
حفظ النموذج

```
# 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.

15) توقع عدم انتظام ضربات القلب على بيانات ECG باستخدام التعلم العميق Arrhythmia prediction on ECG data using Deep Learning

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

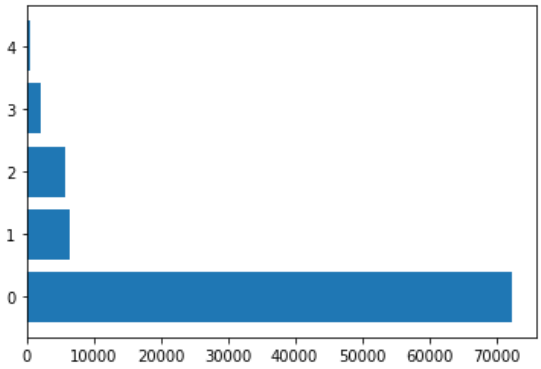
4 ـ دقات غير معروفة Unknown beats

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)
```

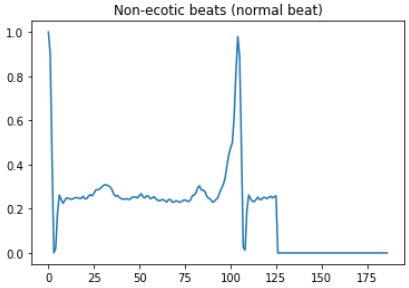
```
# Concatenate the 5 dataframes into 1
train = pd.concat([train_lbl0, train_lbl1, train_lbl2, train_lbl3,
train_lbl4])
labels = train[187].astype('int64')   # last column has the labels
print("Count in each label: ")
print(labels.value_counts())
```

```
Count in each label:
4    50000
3    50000
2    50000
1    50000
0    50000
Name: 187, dtype: int64
```

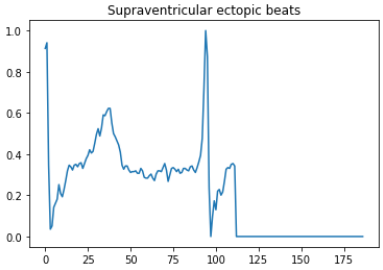
التمثيل المرئي للبيانات

فيما يلي نظرة خاطفة بصرية على فئة مختلفة من دقات القلب:

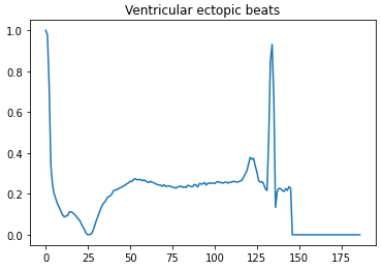
```
plt.plot(np.array(train_lbl0.sample(1))[0, :187])
plt.title(label_names[0])
```



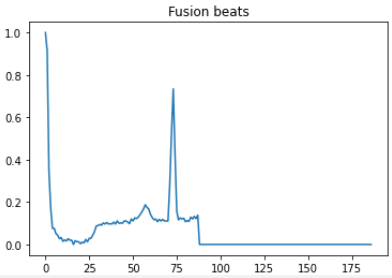

```
plt.plot(np.array(train_lbl1.sample(1))[0, :187])
plt.title(label_names[1])
```



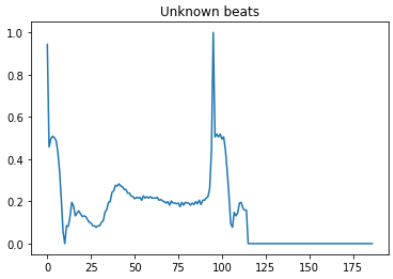

```
plt.plot(np.array(train_lbl2.sample(1))[0, :187])
plt.title(label_names[2])
```




```
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
```

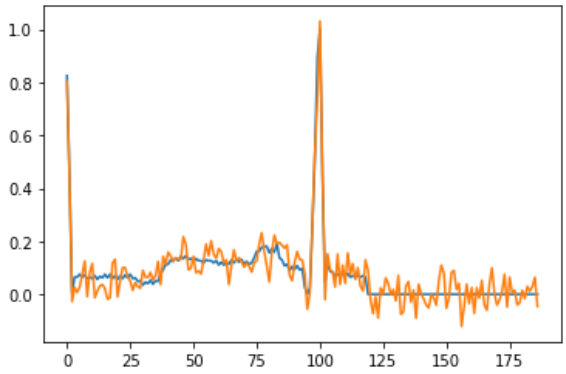
تضيف هذه الدالة ضوضاء للإشارة التي تم تمريرها كمعامل. لا تترددفي التلاعب بالمعامل الفائق للانحراف المعياري (هنا، 0.05) ورسم الإشارة بالضوضاء.

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
_________________________________________________________________
```

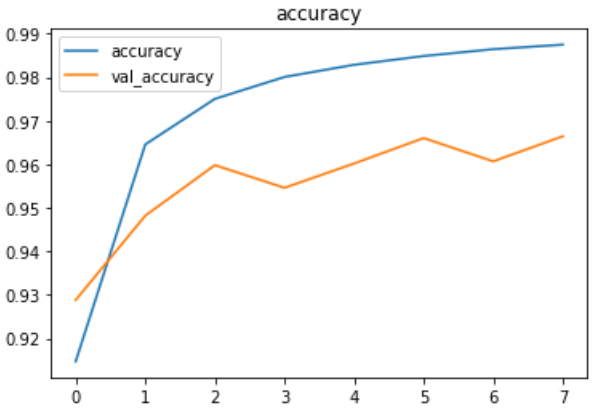
كان النموذج قادرًا على تحقيق دقة تصل إلى 96٪في مجموعة بيانات الاختبار.

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.

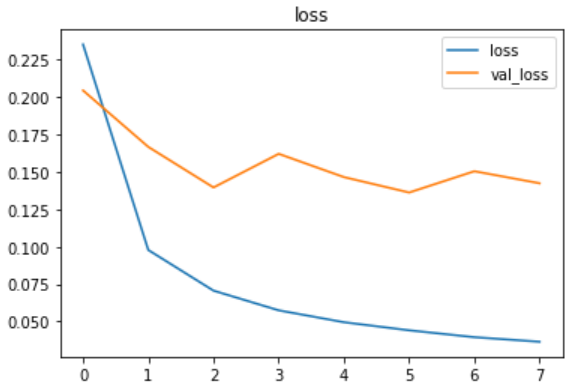
في حالة مجموعات البيانات غير المتوازنة، تعد الدرجة f1 مقياسًا جيدًا لسهولة استخدام النموذج.

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")
```



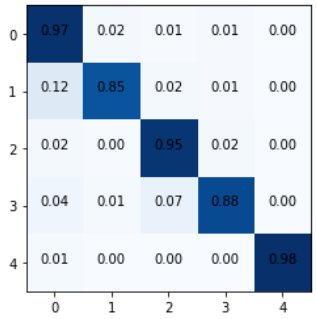
حفظ النموذج

```
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
```

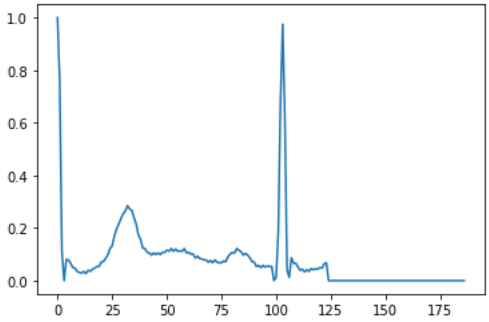
بالنظر إلى أن مجموعة الاختبار الخاصة بنا غير متوازنة، تشير f1-score العالية إلى أن نموذجنا يتمتع بأداء جيد.

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
```



16) 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 MRI للمرضى. نعم يحتوي المجلد على مرضى يعانون من أورامفي المخ بينما لا يحتوي أي مجلد على صور التصوير بالرنين المغناطيسي للمرضى الذين لا يعانون من ورمفي المخ.

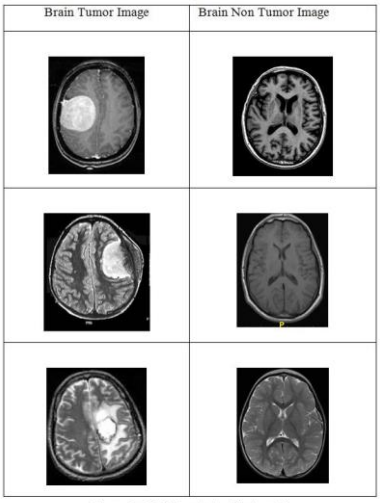


Figure 2: CNN based classified results

الخطوة 1: استيراد المكتبات ومجموعة البيانات المطلوبة.

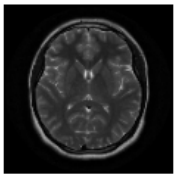
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import math
import cv2
import matplotlib.pyplot as plt
import os
import seaborn as sns
import umap
from PIL import Image
from scipy import misc
from os import listdir
from os.path import isfile, 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

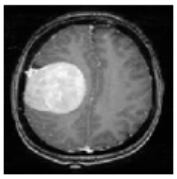
```
os.listdir('../input/brain_tumor_dataset')
from google.colab import drive
drive.mount('/content/drive')
```

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

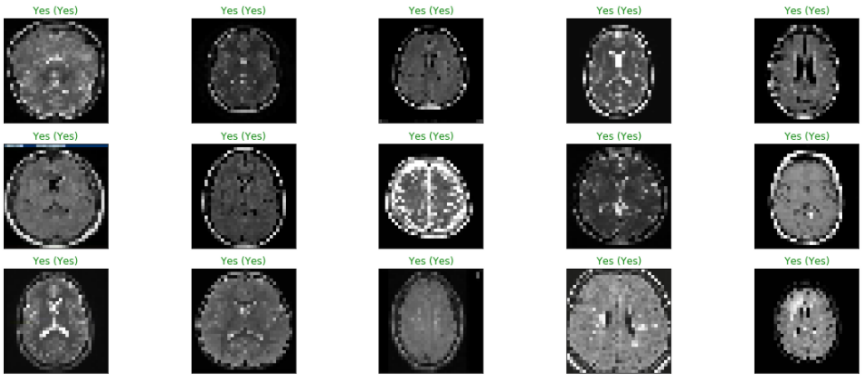
```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test, verbose=0)

# 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.



17) 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.

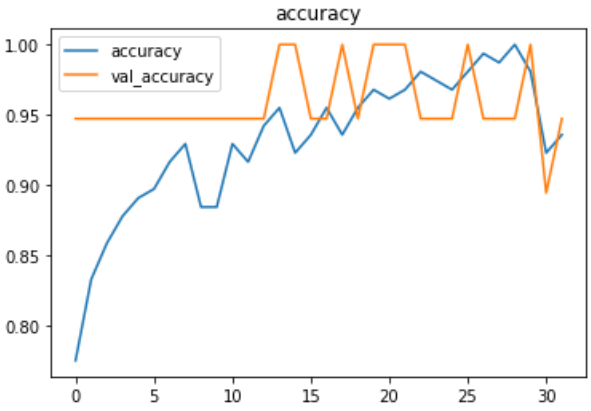
```
ypred = model.predict_classes(xtest)

confusion_matrix(np.argmax(ytest, axis = 1), ypred)
```

```
array([[ 6,  1],
       [ 0, 13]])
```

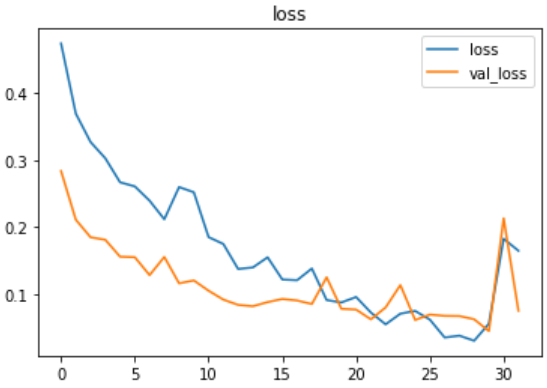
تشير الصفوف إلى القيم الفعلية وتشير الأعمدة إلى القيم المتوقعة. تشير المصفوفة إلى وجود إيجابي خاطئ واحد وفي هذه الحالة، تكون النتيجة الإيجابية الخاطئة أفضل من السلبية الخاطئة.

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
```

18) 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

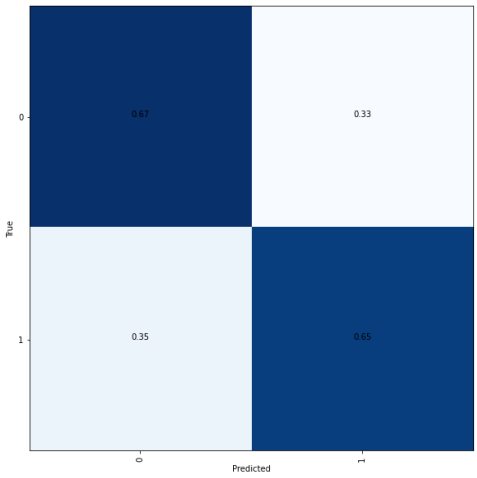
```
model.evaluate(Xtest, ytest)
```

```
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

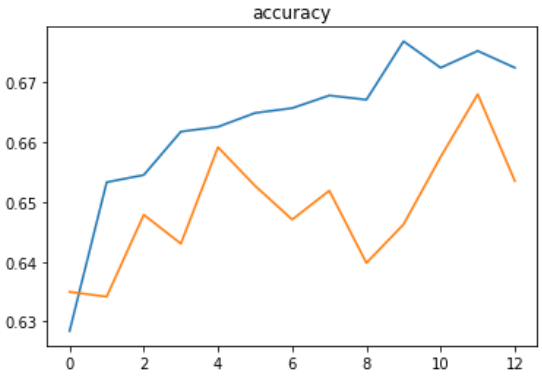
رسم مصفوفة الارتباك لفهم النتائج بشكل أفضل.



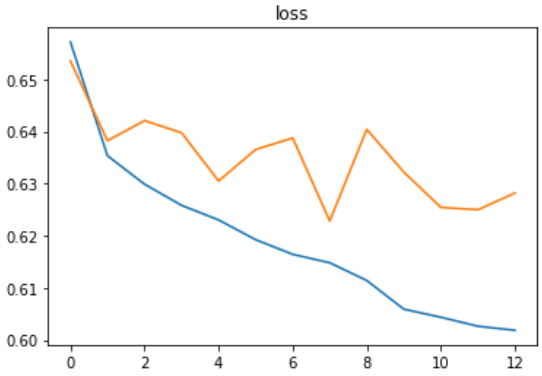
يمكن زيادة معدل الدقة المنخفض ببيانات ذات تصنيف أفضل. من السهل وصف البيانات الذاتية التي تشير إلى الحالة العقلية بأنها خاطئة mislabelled.

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
```

19) 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 ـ 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 (غالبًا في 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 True في دالة get_dummies() كما هو موضح في خلية الكود أدناه.

نظرًا لأن هذا العمود يحتوي على قيمتين فريدتين فقط في إطار البيانات، فلن يكون هناك أي اختلاف بين ترميز واحد ساخن one-hot encoding وترميز التسمية label encoding للعمود.

تمثل القيم الموجودة في 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 الموجودة في 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 ترميز التسمية 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
```

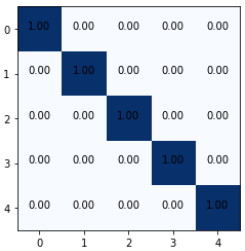
تقييم النموذج

```
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)
```

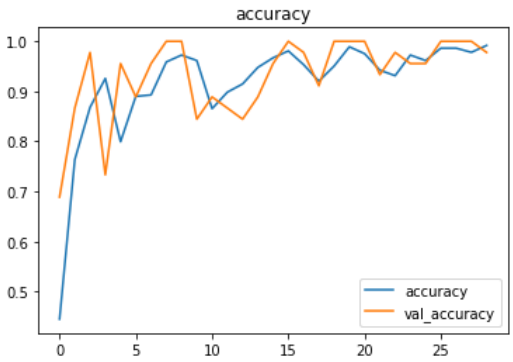


من المهم الحفاظ على الدقة عالية للغاية (100٪) حيث لا يمكن أخذ الفرص مع دواء المريض.

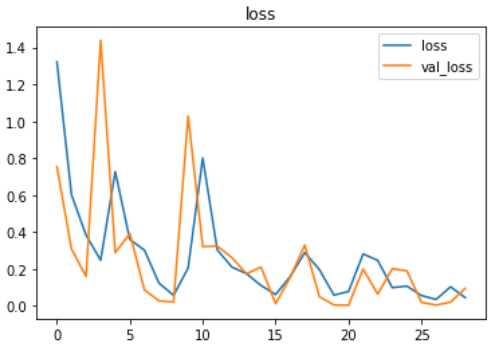
رسم المقاييس

```
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
```

20) 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)
```

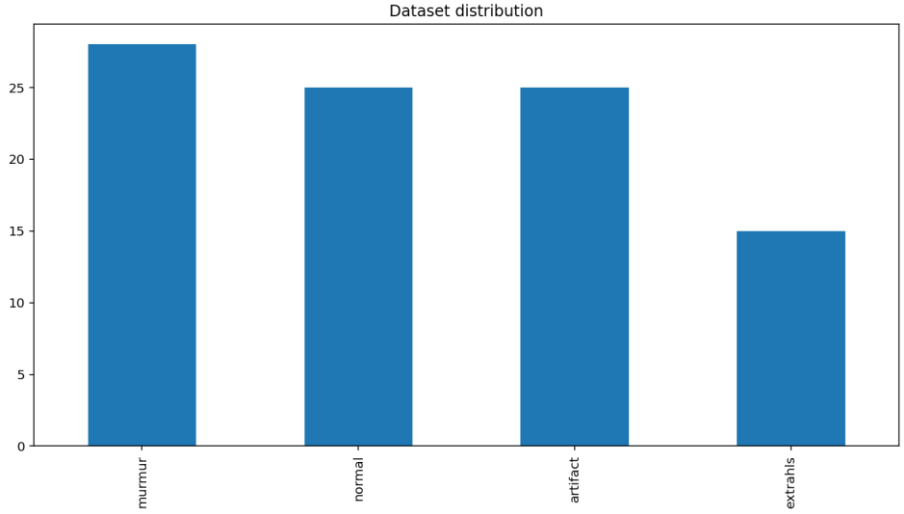
تحليل البيانات

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"
```

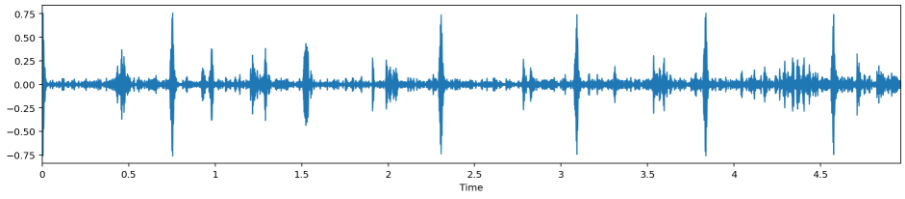
```
# Load use wave
import wave
wav = wave.open(normal_file)
print("Sampling (frame) rate = ", wav.getframerate())
print("Total samples (frames) = ", wav.getnframes())
print("Duration = ", wav.getnframes()/wav.getframerate())
```

```
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)
```



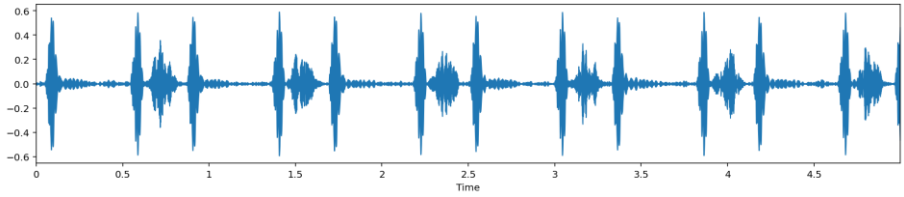
حالة Murmur

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

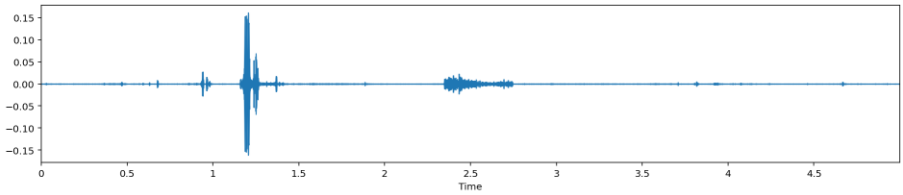
في فئة Artifact هناك مجموعة واسعة من الأصوات المختلفة، بمافي ذلك صرير ردود الفعل والصدى والكلام والموسيقى والضوضاء. عادة لا توجد أصوات قلب يمكن تمييزها، وبالتالي فإن دورية زمنية قليلة أو معدومة على ترددات أقل من 195 هرتز.

```
# 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)
```



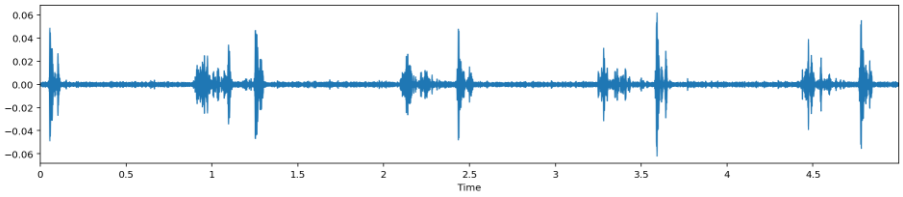
Extrahls

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)
```



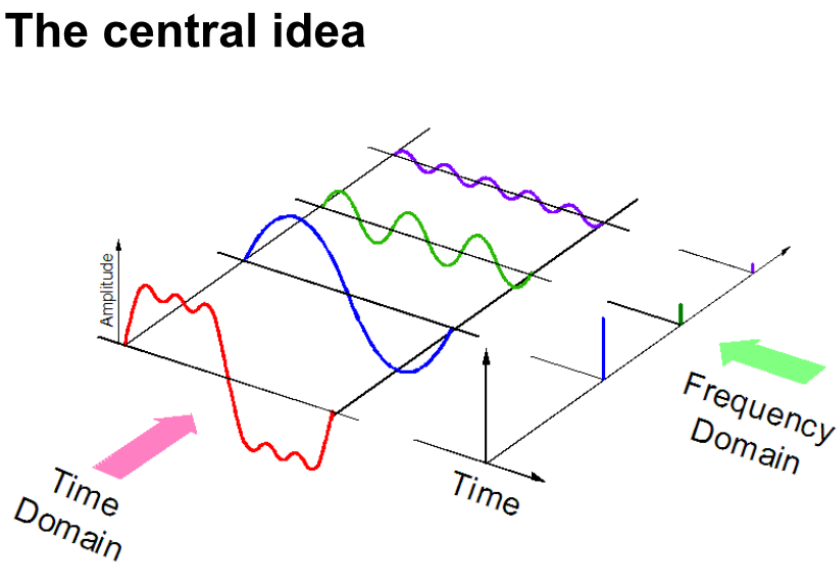
تقسيم مجموعة البيانات الى تدريب واختبار

```
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.

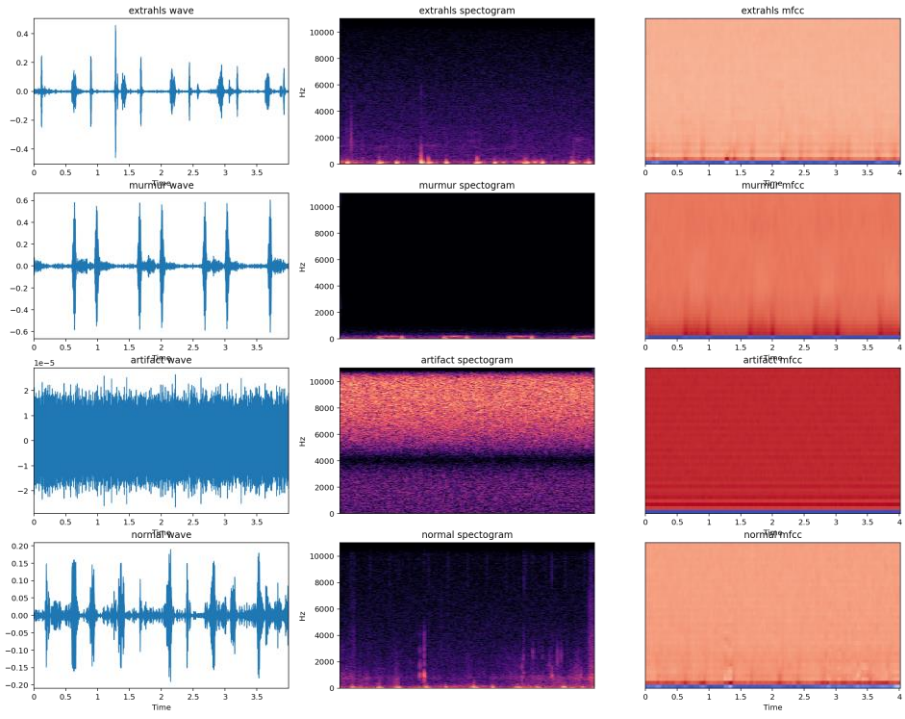
تشمل ميزات الصوت العامة:

- 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
```

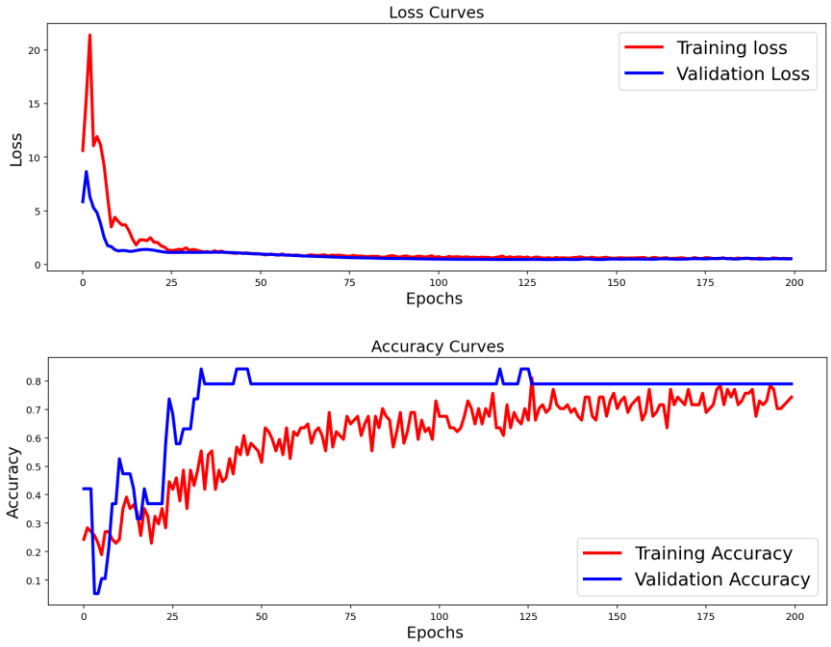
يمكن زيادة أداء النموذج من خلال زيادة عدد الحقب، وبيانات التدريب وإنشاء callbacks مخصصة.

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

21) 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.

"10_left" 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 بترميز 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 :

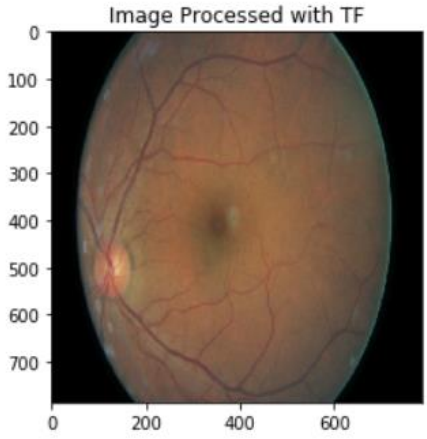
```
"""
#this is a OpenCV implementation
i = 0
for f, breed in tqdm(df_train.values):
    if
type(cv2.imread('/storage/train/{}.jpeg'.format(f)))==type(None):
        continue
    else:
        img = cv2.imread('/storage/train/{}.jpeg'.format(f))
        label = one_hot_labels[i]
        x train.append(cv2.resize(img, (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 / Tensorflow في شكلين ـ متسلسل (model = Sequential()) أو باستخدام Funtional API.

يستخدم الكود أدناه Funtional API والذي يستخدم عادة للنماذج المعقدة. 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 وتسوية الدُفعات والتسرب ودوال الخسارة المخصصة للحصول على أداء أفضل.

22) الكشف عن ورم الدماغ وتحديد موقعه باستخدام التعلم العميق: الجزء 1 Brain Tumor Detection and Localization using Deep Learning: Part 1

بيان المشكلة:

للتنبؤ بأورام الدماغ وتحديد موقعها من خلال تجزئة الصور من مجموعة بيانات التصوير بالرنين المغناطيسي (MRI) المتوفرةفي Kaggle.

لقد قسمت هذه المقالة إلى سلسلة من جزأين حيث سنقوم بتدريب نموذجين للتعلم العميق لنفس مجموعة البيانات ولكن للمهام المختلفة.

النموذجفي هذا الجزء هو نموذج تصنيف يكتشف الأورام من صورة التصوير بالرنين المغناطيسي MRI، وبعد ذلك إذا كان هناك ورم، فسنقوم بتحديد موضع الورمفي الدماغفي الجزء التالي من هذه السلسلة.

المتطلبات المسبقة:

التعلم العميق

سأحاول شرح كل جزء بدقة. دعنا نتوجه إلى جزء التنفيذ باستخدام بايثون.

مجموعة البيانات

- اهم الاشياء اولا! لنبدأ باستيراد المكتبات المطلوبة.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import cv2
from skimage import io
import tensorflow as tf
from tensorflow.python.keras import Sequential
from tensorflow.keras import layers, optimizers
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras import backend as K
from sklearn.preprocessing import StandardScaler
%matplotlib inline
```

- قم بتحويل ملف CSV الخاص بمجموعة البيانات إلى إطار بيانات لإجراء عمليات محددة عليه.

```
# data containing path to Brain MRI and their corresponding mask
brain_df = pd.read_csv('/Healthcare AI Datasets/Brain_MRI/data_mask.csv')
```

- اعرض تفاصيل DataFrame.

```
brain_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3929 entries, 0 to 3928
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   patient_id  3929 non-null   object
 1   image_path  3929 non-null   object
 2   mask_path   3929 non-null   object
 3   mask        3929 non-null   int64 
dtypes: int64(1), object(3)
memory usage: 122.9+ KB
```

```
brain_df.head(5)
```

	patient_id	image_path	mask_path	mask
0	TCGA_CS_5395_19981004	TCGA_CS_5395_19981004/TCGA_CS_5395_19981004_1.tif	TCGA_CS_5395_19981004/TCGA_CS_5395_19981004_1_...	0
1	TCGA_CS_5395_19981004	TCGA_CS_4944_20010208/TCGA_CS_4944_20010208_1.tif	TCGA_CS_4944_20010208/TCGA_CS_4944_20010208_1_...	0
2	TCGA_CS_5395_19981004	TCGA_CS_4941_19960909/TCGA_CS_4941_19960909_1.tif	TCGA_CS_4941_19960909/TCGA_CS_4941_19960909_1_...	0
3	TCGA_CS_5395_19981004	TCGA_CS_4943_20000902/TCGA_CS_4943_20000902_1.tif	TCGA_CS_4943_20000902/TCGA_CS_4943_20000902_1_...	0
4	TCGA_CS_5395_19981004	TCGA_CS_5396_20010302/TCGA_CS_5396_20010302_1.tif	TCGA_CS_5396_20010302/TCGA_CS_5396_20010302_1_...	0

1. معرّف المريض (Patient id): معرّف المريض لكل سجل (dtype: Object)
2. مسار الصورة (Image path): المسار إلى صورة التصوير بالرنين المغناطيسي (dtype: Object)
3. مسار القناع (Mask path): المسار إلى قناع الصورة المقابلة (dtype: Object)
4. القناع (Mask): له قيمتان: 0 و 1 حسب صورة القناع. (dtype: int64).

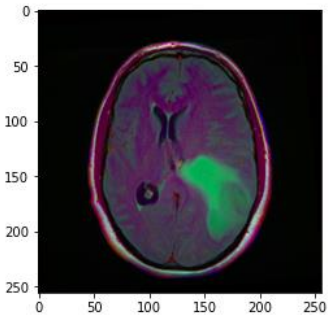
- احسب قيم كل فئة.

```
brain_df['mask'].value_counts()
```

```
0    2556
1    1373
Name: mask, dtype: int64
```

- عرض صورة MRI بشكل عشوائي من مجموعة البيانات.

```
image = cv2.imread(brain df.image path[1301])
plt.imshow(image)
```

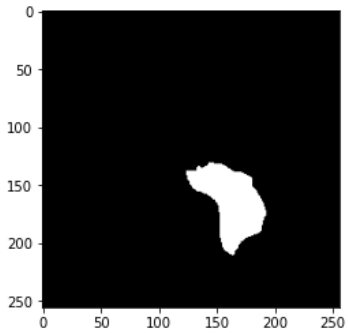


يخزن مسار image_path مسار التصوير بالرنين المغناطيسي (MRI) للدماغ حتى نتمكن من عرض الصورة باستخدام matplotlib.

تلميح: الجزء المخضرفي الصورة أعلاه يمكن اعتباره الورم.

- اعرض أيضًا صورة القناع المقابلة.

```
image1 = cv2.imread(brain_df.mask_path[1301])
plt.imshow(image1)
```



الآن، ربما تكون قد حصلت على تلميح لما هو القناع بالفعل. القناع (MASK) هو صورة لجزء من الدماغ متأثر بورمفي صورة MRI المقابلة. هنا، القناع من تصوير MRI المعروض أعلاه.

- تحليل قيم البكسل لصورة القناع.

```
cv2.imread(brain_df.mask_path[1301]).max()
```

```
Output: 255
```

قيمة البكسل القصوىفي صورة القناع هي 255 مما يشير إلى اللون الأبيض.

```
cv2.imread(brain_df.mask_path[1301]).min()
```

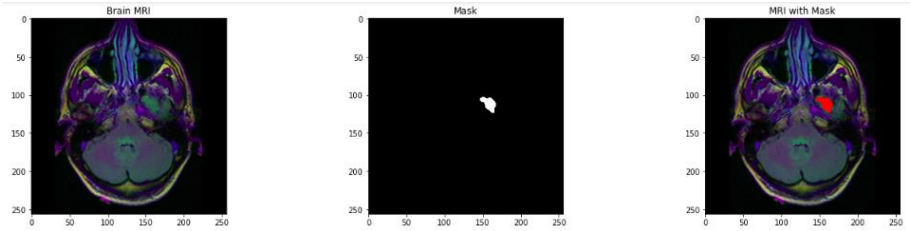
```
Output: 0
```

الحد الأدنى لقيمة البكسلفي صورة القناع هو 0 مما يشير إلى اللون الأسود.

- رسم صور MRI للدماغ، والقناع المقابل، وMRI مع القناع.

```
count = 0
fig, axs = plt.subplots(12, 3, figsize = (20, 50))
for i in range(len(brain_df)):
  if brain_df['mask'][i] ==1 and count <5:
    img = io.imread(brain_df.image_path[i])
    axs[count][0].title.set_text('Brain MRI')
    axs[count][0].imshow(img)

    mask = io.imread(brain_df.mask_path[i])
    axs[count][1].title.set_text('Mask')
    axs[count][1].imshow(mask, cmap = 'gray')
```

```
    img[mask == 255] = (255, 0, 0) #Red color
    axs[count][2].title.set_text('MRI with Mask')
    axs[count][2].imshow(img)
    count+=1

fig.tight_layout()
```



- قم بإسقاط (حذف) المعرف (id) لأنه ليس مطلوبًا للمعالجة.

```
# Drop the patient id column
brain_df_train = brain_df.drop(columns = ['patient_id'])
brain_df_train.shape
```

ستحصل على حجم إطار البياناتفي الإخراج: (3929، 3)

قم بتحويل البيانات الموجودةفي عمود القناع (mask column) من عدد صحيح إلى تنسيق سلسلة نصية لأننا سنطلب البيانات بتنسيق سلسلة نصية.

```
brain_df_train['mask'] = brain_df_train['mask'].apply(lambda x: str(x))
brain_df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3929 entries, 0 to 3928
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   image_path  3929 non-null   object
 1   mask_path   3929 non-null   object
 2   mask        3929 non-null   object
dtypes: object(3)
memory usage: 92.2+ KB
```

كما ترى، تحتوي كل ميزة الآن على نوع البيانات ككائن (object).

- قسّم البيانات إلى مجموعات التدريب (train) والاختبار (test).

```
# split the data into train and test data
from sklearn.model_selection import train_test_split
train, test = train_test_split(brain_df_train, test_size = 0.15)
```

- قم بزيادة البيانات باستخدام ImageDataGenerator. يولد ImageDataGenerator دفعات من بيانات صورة الموتر مع زيادة البياناتفي الوقت الفعلي.

سننشئ train_generator و validation_generator من بيانات التدريب ومولد اختبار من بيانات الاختبار.

```
# create an image generator
from keras_preprocessing.image import ImageDataGenerator

#Create a data generator which scales the data from 0 to 1 and makes
validation split of 0.15
datagen = ImageDataGenerator(rescale=1./255., validation_split = 0.15)

train_generator=datagen.flow_from_dataframe(
dataframe=train,
directory= './',
x_col='image_path',
y_col='mask',
subset="training",
batch_size=16,
shuffle=True,
class_mode="categorical",
target_size=(256,256))

valid_generator=datagen.flow_from_dataframe(
dataframe=train,
directory= './',
x_col='image_path',
y_col='mask',
subset="validation",
batch_size=16,
shuffle=True,
class_mode="categorical",
target_size=(256,256))

# Create a data generator for test images
test_datagen=ImageDataGenerator(rescale=1./255.)

test_generator=test_datagen.flow_from_dataframe(
dataframe=test,
directory= './',
x_col='image_path',
y_col='mask',
batch_size=16,
shuffle=False,
class_mode='categorical',
target_size=(256,256))
```

الآن، سوف نتعلم مفهوم نقل التعلم (Transfer Learning) ونموذج ResNet50 الذي سيتم استخدامه لنموذج تدريب إضافي.

نقل التعلم كما يوحي الاسم، هو أسلوب لاستخدام النماذج المدربة (pre-trained models) مسبقًافي تدريبك. يمكنك بناء النموذج الخاص بك فوق هذا النموذج المدرب مسبقًا. هذه عملية تساعدك على تقليل وقت التطوير وزيادة الأداء.

ResNet (الشبكة المتبقية) هي شبكة ANN المدربة على مجموعة بيانات ImageNet والتي يمكن استخدامها لتدريب النموذج فوقها. ResNet50 هو البديل من نموذج ResNet الذي يحتوي على 48 طبقة التفاف مع طبقة واحدة MaxPool وطبقة واحدة Average pool.

- هنا، نحن نستخدم نموذج ResNet50 وهو نموذج نقل التعلم. باستخدام هذا، سنضيف المزيد من الطبقات لبناء نموذجنا.

```
# Get the ResNet50 base model (Transfer Learning)
basemodel = ResNet50(weights = 'imagenet', include_top = False,
input_tensor = Input(shape=(256, 256, 3)))
basemodel.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
input_1 (InputLayer)            [(None, 256, 256, 3) 0
__________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 262, 262, 3)  0           input_1[0][0]
__________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 128, 128, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 128, 128, 64) 256         conv1_conv[0][0]
__________________________________________________________________________________________
conv1_relu (Activation)         (None, 128, 128, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 130, 130, 64) 0           conv1_relu[0][0]
__________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 64, 64, 64)   0           pool1_pad[0][0]
__________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 64, 64, 64)   4160        pool1_pool[0][0]
```

يمكنك عرض الطبقاتفي نموذج resnet50 باستخدام summary() كما هو موضح أعلاه.

- قم بتجميد أوزان النموذج. هذا يعني أننا سنحافظ على ثبات الأوزان حتى لا يتم تحديثها أكثر. سيؤدي ذلك إلى تجنب إتلاف أي معلومات أثناء التدريب الإضافي.

```
# freeze the model weights
for layer in basemodel.layers:
  layers.trainable = False
```

- الآن، كما هو مذكور أعلاه، سنضيف المزيد من الطبقاتفي الجزء العلوي من طبقات ResNet50. ستتعلم هذه الطبقات تحويل الميزات القديمة إلى تنبؤاتفي مجموعة البيانات الخاصة بنا.

```
headmodel = basemodel.output
headmodel = AveragePooling2D(pool_size = (4,4))(headmodel)
headmodel = Flatten(name= 'flatten')(headmodel)
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dropout(0.3)(headmodel)
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dropout(0.3)(headmodel)
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dropout(0.3)(headmodel)
headmodel = Dense(2, activation = 'softmax')(headmodel)

model = Model(inputs = basemodel.input, outputs = headmodel)
model.summary()
```

```
average_pooling2d (AveragePooli (None, 2, 2, 2048)  0           conv5_block3_out[0][0]
flatten (Flatten)               (None, 8192)        0           average_pooling2d[0][0]
dense (Dense)                   (None, 256)         2097408     flatten[0][0]
dropout (Dropout)               (None, 256)         0           dense[0][0]
dense_1 (Dense)                 (None, 256)         65792       dropout[0][0]
dropout_1 (Dropout)             (None, 256)         0           dense_1[0][0]
dense_2 (Dense)                 (None, 256)         65792       dropout_1[0][0]
dropout_2 (Dropout)             (None, 256)         0           dense_2[0][0]
dense_3 (Dense)                 (None, 2)           514         dropout_2[0][0]
==================================================================================================
```

تتم إضافة هذه الطبقات ويمكنك رؤيتهافي الملخص.

1. تُستخدم طبقات التجميع (Pooling layers) لتقليل أبعاد خرائط المعالم. تقوم طبقة Average Pooling بإرجاع متوسط القيم.
2. تقوم الطبقات المسطحة بتحويل بياناتنا إلى متجه.
3. الطبقة الكثيفة (dense layer) هي طبقة الشبكة العصبية العادية المتصلة بعمق. بشكل أساسي، يأخذ المدخلات ويحسب:

output = activation(dot(input, kernel) + bias)

4. تمنع طبقة التسرب (dropout layer) النموذج من الضبط الزائد (overfitting). يقوم بشكل عشوائي بتعيين وحدات الإدخال للطبقات المخفية على 0 أثناء التدريب.

- قم بتجميع (Compile) النموذج المبني أعلاه. يحدد Compile دالة الخطأ (loss function) والمحسن (optimizer) والمقاييس (metrics).

```
# compile the model
model.compile(loss = 'categorical_crossentropy', optimizer='adam',
metrics= ["accuracy"])
```

- إجراء إيقاف مبكر (early stopping) لحفظ أفضل نموذج بأقل خطأ للتحقق من الصحة. يؤدي التوقف المبكر عددًا كبيرًا من فترات التدريب ويتوقف عن التدريب بمجرد أن لا يتحسن أداء النموذجفي مجموعة بيانات التحقق من الصحة.
- يتم استخدام ModelCheckpoint من callback مع التدريب باستخدام model.fit() لحفظ الأوزانفي فاصل زمني معين، لذلك يمكن تحميل الأوزان لاحقًا لمواصلة التدريب من الحالة المحفوظة.

```
# use early stopping to exit training if validation loss is not decreasing
even after certain epochs
earlystopping = EarlyStopping(monitor='val_loss', mode='min', verbose=1,
patience=20)

# save the model with least validation loss
checkpointer = ModelCheckpoint(filepath="classifier-resnet-weights.hdf5",
verbose=1, save_best_only=True)
```

الآن، يمكنك تدريب النموذج وإعطاء عمليات (callbacks) المحددة أعلاهفي المعلمة.

```
model.fit(train_generator, steps_per_epoch= train_generator.n // 16,
epochs = 1, validation_data= valid_generator, validation_steps=
valid_generator.n // 16, callbacks=[checkpointer, earlystopping])
```

التنبؤ وتحويل البيانات المتوقعة إلى قائمة (list).

```
# make prediction
test_predict = model.predict(test_generator, steps = test_generator.n //
16, verbose =1)

# Obtain the predicted class from the model prediction
predict = []
for i in test predict:
  predict.append(str(np.argmax(i)))
predict = np.asarray(predict)
```

قياس دقة النموذج.

```
# Obtain the accuracy of the model
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(original, predict)
accuracy
```

```
0.9826388888888888
```

اطبع تقرير التصنيف (classification report).

```
from sklearn.metrics import classification_report
report = classification_report(original, predict, labels = [0,1])
print(report)
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       365
           1       0.99      0.97      0.98       211

   micro avg       0.98      0.98      0.98       576
   macro avg       0.98      0.98      0.98       576
weighted avg       0.98      0.98      0.98       576
```

الآن، يمكنك أن ترتاح لأنك انتهيت للتو من الجزء الأول، أي اكتشاف ورم الدماغ (brain tumor detection). الآن، سوف نتجه نحو تجزئة ورم الدماغ (brain tumor segmentation)في الجزء الثاني من هذه السلسلة.

الكشف عن ورم الدماغ وتحديده باستخدام التعلم العميق: الجزء 2.

23) الكشف عن ورم الدماغ وتحديد موقعه باستخدام التعلم العميق: الجزء 2 Brain Tumor Detection and Localization using Deep Learning: Part 2

بيان المشكلة:

للتنبؤ بأورام الدماغ وتحديد موقعها من خلال تجزئة الصور (image segmentation) من مجموعة بيانات MRI المتوفرةفي Kaggle.

هذا هو الجزء الثاني من السلسلة. إذا لم تكن قد قرأت الجزء الأول بعد، فإنني أوصي بزيارة اكتشاف ورم الدماغ وتحديد موقعه باستخدام التعلم العميق: الجزء 1 لفهم الكود بشكل أفضل حيث أن كلا الجزأين مترابطان.

لقد قمنا بتدريب نموذج تصنيف أعلى ResNet50 والذي يصنف ما إذا كان MRI للدماغ به ورم أو لا باستخدام عمليات (callbacks) لزيادة أدائنا.في هذا الجزء، سنقوم بتدريب نموذج لتحديد مكان الورم باستخدام تجزئة الصورة.

المتطلبات المسبقة:

تعلم عميق

رابط مجموعة البيانات

الآن، دعونا نتجه نحو تنفيذ الجزء الثاني، أي بناء نموذج تجزئة لتحديد مكان الأورام (Building a segmentation model to localize tumors).

الهدف من تجزئة الصورة هو فهم الصورة على مستوى البكسل. يربط كل بكسل بفئة معينة. يُطلق على الإخراج الناتج عن نموذج تجزئة الصورة اسم قناع الصورة (the mask of the image).

- بادئ ذي بدء، حدد السجلات التي لها قيمة القناع 1 من إطار البيانات الذي أنشأناهفي الجزء الأخير لأنه لا يمكننا تحديد موقع الورم إلا إذا كان موجودًا.

```
# Get the dataframe containing MRIs which have masks associated with them.
brain_df_mask = brain_df[brain_df['mask'] == 1]
brain_df_mask.shape
```

```
Output: (1373, 4)
```

- قسّم البيانات إلى مجموعات بيانات تدريب (train) واختبار (test). هنا أولاً، قمنا بتقسيم البيانات بأكملها إلى تدريب (train) والتحقق من الصحة (validation)، ثم قمنا بتقسيم نصف بيانات التحقق إلى بيانات اختبار (test).

```
from sklearn.model_selection import train_test_split
X_train, X_val = train_test_split(brain_df_mask, test_size=0.15)
```

```
X_test, X_val = train_test_split(X_val, test_size=0.5)
```

- سنقوم مرة أخرى بإنشاء بيانات وهمية (dummy data)، على سبيل المثال، مولد التدريب (training_generator) والتحقق من الصحة (validation_generator) باستخدام DataGenerator. لهذا الغرض، سننشئ أولاً قائمة بالصورة ومسار القناع لتمريرها إلى المولد.

```
train_ids = list(X_train.image_path)
train_mask = list(X_train.mask_path)

val_ids = list(X_val.image_path)
val_mask= list(X_val.mask_path)

# Utilities file contains the code for custom data generator
from utilities import DataGenerator

# create image generators
training_generator = DataGenerator(train_ids,train_mask)
validation_generator = DataGenerator(val_ids,val_mask)
```

- حدد طريقة Resblock كما هو موضح أدناه لاستخدامهافي نموذج التعلم العميق الخاص بنا.

يتم استخدام Resblocksفي النموذج للحصول على نتائج أفضل. هذه الكتل هي مجرد مجموعة من الطبقات. الغرابة الرئيسيةفي resblocks هو أن الدالة المتبقية (residual function) يتم تعلمهافي الأعلى ويتم تمرير المعلومات على طول الجزء السفلي دون تغيير.

```
def resblock(X, f):

  # make a copy of input
  X_copy = X

  X = Conv2D(f, kernel_size = (1,1) ,strides = (1,1),kernel_initializer
='he_normal')(X)
  X = BatchNormalization()(X)
  X = Activation('relu')(X)

  X = Conv2D(f, kernel_size = (3,3), strides =(1,1), padding = 'same',
kernel_initializer ='he_normal')(X)
  X = BatchNormalization()(X)

  X_copy = Conv2D(f, kernel_size = (1,1), strides =(1,1),
kernel_initializer ='he_normal')(X_copy)
  X_copy = BatchNormalization()(X_copy)

  # Adding the output from main path and short path together
  X = Add()([X,X copy])
  X = Activation('relu')(X)

  return X
```

- وبالمثل، حدد طريقة upsample_concat التي تقوم بترقية وتسلسل القيم التي تم تمريرها. طبقة Upsampling عبارة عن طبقة بسيطة بدون أوزان تضاعف أبعاد الإدخال.

```
def upsample_concat(x, skip):
  x = UpSampling2D((2,2))(x)
  merge = Concatenate()([x, skip])

  return merge
```

- قم ببناء نموذج تجزئة (segmentation model) مضيفًا الطبقات الموضحة أدناه بمافي ذلك resblock الموضح أعلاه و upsample_concat.

```
input_shape = (256,256,3)

# Input tensor shape
X_input = Input(input_shape)

# Stage 1
conv1_in = Conv2D(16,3,activation= 'relu', padding = 'same',
kernel_initializer ='he_normal')(X_input)
conv1_in = BatchNormalization()(conv1_in)
conv1_in = Conv2D(16,3,activation= 'relu', padding = 'same',
kernel_initializer ='he_normal')(conv1_in)
conv1_in = BatchNormalization()(conv1_in)
pool_1 = MaxPool2D(pool_size = (2,2))(conv1_in)

# Stage 2
conv2_in = resblock(pool_1, 32)
pool_2 = MaxPool2D(pool_size = (2,2))(conv2_in)

# Stage 3
conv3_in = resblock(pool_2, 64)
pool_3 = MaxPool2D(pool_size = (2,2))(conv3_in)

# Stage 4
conv4_in = resblock(pool_3, 128)
pool_4 = MaxPool2D(pool_size = (2,2))(conv4_in)

# Stage 5 (Bottle Neck)
conv5_in = resblock(pool_4, 256)

# Upscale stage 1
up_1 = upsample_concat(conv5_in, conv4_in)
up_1 = resblock(up_1, 128)

# Upscale stage 2
up_2 = upsample_concat(up_1, conv3_in)
up_2 = resblock(up_2, 64)

# Upscale stage 3
up_3 = upsample_concat(up_2, conv2_in)
up_3 = resblock(up_3, 32)

# Upscale stage 4
up_4 = upsample_concat(up_3, conv1_in)
up_4 = resblock(up_4, 16)

# Final Output
output = Conv2D(1, (1,1), padding = "same", activation = "sigmoid")(up_4)

model_seg = Model(inputs = X_input, outputs = output )
```

- قم بتجميع (compile) النموذج الذي تم تدريبه أعلاه. هذه المرة سنقوم بتخصيص معلمات المحسن. TVersky البؤري هو دالة الخطأ و tversky هو المقياس.

```
# Utilities file also contains the code for custom loss function
from utilities import focal_tversky, tversky

# Compile the model
adam = tf.keras.optimizers.Adam(lr = 0.05, epsilon = 0.1)
model_seg.compile(optimizer = adam, loss = focal_tversky, metrics =
[tversky])
```

- الآن، أنت تعرف عمليات (callbacks) التي استخدمناهافي نموذج المصنف الخاص بنا. سوف نستخدم نفس الشيء للحصول على أداء أفضل. أخيرًا، نقوم بتدريب نموذج التجزئة الخاص بنا.

```
# use early stopping to exit training if validation loss is not decreasing
even after certain epochs.
earlystopping = EarlyStopping(monitor='val_loss', mode='min', verbose=1,
patience=20)

# save the best model with lower validation loss
checkpointer = ModelCheckpoint(filepath="ResUNet-weights.hdf5", verbose=1,
save_best_only=True)

model_seg.fit(training_generator, epochs = 1, validation_data =
validation_generator, callbacks = [checkpointer, earlystopping])
```

- توقع القناع لمجموعة بيانات الاختبار الخاصة بنا. هنا، النموذج هو نموذج المصنف الذي تم تدريبهفي الجزء السابق و model_seg هو نموذج التجزئة الذي تم تدريبه أعلاه.

```
from utilities import prediction

# making prediction
image_id, mask, has_mask = prediction(test, model, model_seg)
```

سيعطينا الإخراج مسار الصورة (image path) والقناع المتوقع (predicted mask) وتسمية الفئة (class label).

- إنشاء إطار بيانات من النتيجة المتوقعة ودمجها مع إطار بيانات الاختبار على image_path.

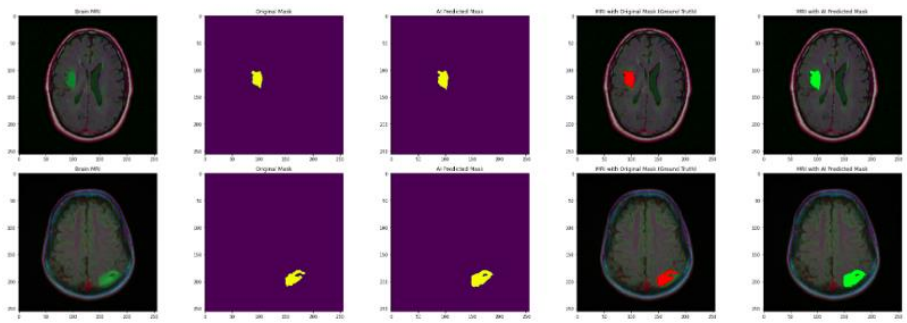
```
# creating a dataframe for the result
df_pred = pd.DataFrame({'image_path': image_id,'predicted_mask':
mask,'has_mask': has_mask})

# Merge the dataframe containing predicted results with the original test
data.
df_pred = test.merge(df_pred, on = 'image_path')
df_pred.head()
```

	image_path	mask_path	mask	predicted_mask	has_mask
0	TCGA_HT_8106_19970727/TCGA_HT_8106_19970727_7.tif	TCGA_HT_8106_19970727/TCGA_HT_8106_19970727_7_...	0	No mask	0
1	TCGA_CS_5396_20010302/TCGA_CS_5396_20010302_7.tif	TCGA_CS_5396_20010302/TCGA_CS_5396_20010302_7_...	0	No mask	0
2	TCGA_DU_6399_19830416/TCGA_DU_6399_19830416_30...	TCGA_DU_6399_19830416/TCGA_DU_6399_19830416_30...	1	[[[[8.8394046e-07], [3.956604e-06], [1.1406702...	1
3	TCGA_DU_7294_19890104/TCGA_DU_7294_19890104_8.tif	TCGA_DU_7294_19890104/TCGA_DU_7294_19890104_8_...	0	No mask	0
4	TCGA_HT_7602_19951103/TCGA_HT_7602_19951103_18...	TCGA_HT_7602_19951103/TCGA_HT_7602_19951103_18...	0	No mask	0

كما ترىفي الإخراج، لدينا الآن قناعنا النهائي المتوقع مدمجًافي إطار البيانات الخاص بنا.

- أخيرًا، رسم الصورة الأصلية والقناع الأصلي والقناع المتوقع معًا لتحليل دقة نموذج التجزئة الخاص بنا.

```
count = 0
fig, axs = plt.subplots(10, 5, figsize=(30, 50))
for i in range(len(df_pred)):
  if df_pred['has_mask'][i] == 1 and count < 5:
    # read the images and convert them to RGB format
    img = io.imread(df_pred.image_path[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    axs[count][0].title.set_text("Brain MRI")
    axs[count][0].imshow(img)

    # Obtain the mask for the image
    mask = io.imread(df_pred.mask_path[i])
    axs[count][1].title.set_text("Original Mask")
    axs[count][1].imshow(mask)

    # Obtain the predicted mask for the image
    predicted_mask =
np.asarray(df_pred.predicted_mask[i])[0].squeeze().round()
    axs[count][2].title.set_text("AI Predicted Mask")
    axs[count][2].imshow(predicted_mask)

    # Apply the mask to the image 'mask==255'
    img[mask == 255] = (255, 0, 0)
    axs[count][3].title.set_text("MRI with Original Mask (Ground Truth)")
    axs[count][3].imshow(img)

    img_ = io.imread(df_pred.image_path[i])
    img_ = cv2.cvtColor(img_, cv2.COLOR_BGR2RGB)
    img_[predicted_mask == 1] = (0, 255, 0)
    axs[count][4].title.set_text("MRI with AI Predicted Mask")
    axs[count][4].imshow(img_)
    count += 1

fig.tight_layout()
```



يوضح الناتج أن نموذج التجزئة الخاص بنا يحدد مكان الورم جيداً. أحسنت!

علاوة على ذلك، يمكنك محاولة إضافة المزيد من الطبقات إلى النماذج التي تم تدريبها حتى الآن وتحليل الأداء. أيضًا، يمكنك تطبيق حلول مماثلة على عبارات المشكلة الأخرى حيث أن تجزئة الصورة مجال ذو أهمية كبيرةفي الوقت الحاضر.

24) تجزئة صوت القلب باستخدام التعلم العميق Heart Sound Segmentation using Deep Learning

مقدمة

جاءت فكرة القيام بمشروع حول تجزئة صوت القلب من حيلة حديثه سمعتها عبر الإنترنت. أحد المؤثرين الذين أتابعهم – نشر Andrew Ng ورقة بحثية منذ فترة – والتي تعدفي الأساس طريقة حديثة للكشف عن أمراض القلب.

لقد كانت فكرة مثيرة للاهتمام بالنسبة لي، لذلك قمت بمراجعة جميع المواد المنشورة حول هذا الموضوع لفهم الفكرة الأصلية وراء ذلك.

لإبقاء القصة قصيرة، قال المؤلفون إنهم استخدموا التعلم العميق – وهي تقنية كانت موجودةفي الأخبار لفترة من الوقت الآن، لاستخراج الأنماط التي استخدمها الخبراء لتحديد المريض المصاب. أصبحت هذه الخوارزمية بعد التدريب جيدة جدًافي المهمة التي يدعي المؤلفون أنها تفوقت حتى على الأطباء المخضرمين. أثرت هذه الفكرةفي أنني حتى – وإن كنت صغيرًا – يمكن أن يكون لي تأثير على التقدم الكبير الذي يحرزه هؤلاء الباحثون!

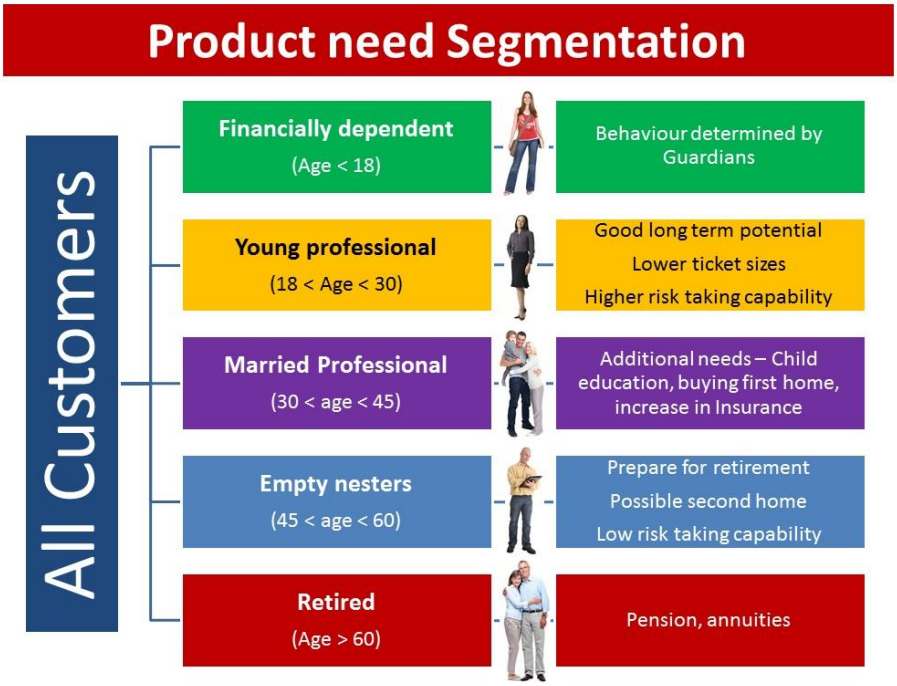
تركز هذه المقالة على مشكلة تجزئة الصوت (audio segmentation)في إشارات تخطيط القلب (ECG signals) وكيف نستفيد من التعلم العميق لحل المهمة. سأناقش أولاً قليلاً حول مشكلة التجزئة بشكل عام ثم أعرض لك الطرق التي يمكن استخدامها لحل المشكلة. سأناقش أيضًا ماهية "صوت القلب heart sound" ثم أريكم تنفيذًا لتجزئة صوت القلب (heart sound segmentation).

فلنبدأفي ذلك!

ما هي مشكلة التجزئة (بشكل عام)؟

قبل أن نغوصفي تجزئة صوت القلب، دعونا نعود ونفهم ما تنطوي عليه مشكلة التجزئة (segmentation). تعني التجزئة حرفياً تقسيم كائن معين إلى أجزاء (أو مقاطع) بناءً على مجموعة محددة من الخصائص. يمكن أن يكون هذا الكائن أي شيء – بدءًا من الأشياء الملموسة مثل إطار صورة أو إشارة صوتية، إلى أشياء مجردة مثل السوق أو المستهلكين.

قد تسأل، لماذا تجزء الأشياء؟ الإجابة بسيطة – إذا قمت بتفكيك كائن ما، فستصبح مهمة استخراج المعلومات منه أسهل. على سبيل المثال،في إدارة العملاء، لا يكشف العمل بالمعدلات أبدًا عن رؤى قابلة للتنفيذ حتى يتم تجزئتها إلى شرائح. كما هو مذكورفي المقالة، هذا مثال على تجزئة العملاء لاستخدام بطاقة الائتمان على أساس أعمارهم.



نهج التجزئة الخاضع للإشراف

الآن بعد أن عرفت التجزئة باعتباره مشكلة، دعنا نفهم طرق حل مشكلة التجزئة.

تعتبر التجزئة، خاصة لتحليل البيانات الصوتية، خطوة مهمة قبل المعالجة. هذا لأنه يمكنك تقسيم إشارة صوتية طويلة وصاخبة إلى مقاطع قصيرة متجانسة، وهي عبارة عن تسلسلات قصيرة سهلة الاستخدام من الصوت المستخدم لمزيد من المعالجة. الآن لحل مشكلة التجزئة، يمكنك إما القيام بذلك مباشرة باستخدام طرق غير خاضعة للإشراف أو تحويلها إلى مشكلة خاضعة للإشراف ثم تجميعها وفقًا لفئتها.

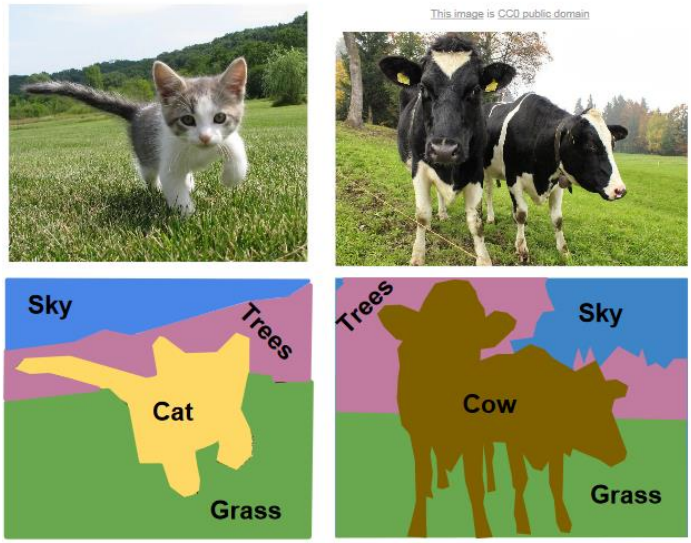
لشرح ذلك بشكل أكثر حدسية، دعنا نأخذ مثالاً على مهمة تجزئة الصورة (Image Segmentation).

لنفترض أن لديك صورة قطةفي أحد الحقول كما نرى أدناه. ما تريده هو تجزئة الصورة إلى أجزاء ـ بحيث يمكن التعرف على كائن فردي بشكل منفصل عن الآخر. يمكنك القيام بذلك بطريقتين

5. النهج الاول: من كل بكسلفي الصورة، اكتشف البيكسلات القريبة من بعضها البعض ولها لون مشابه تقريبًا. يمكنك تجميع وحدات البكسل معًا لتشكيل صورة أكبر لكائن ما.في المثال أدناه، قطتنا بيضاء رماديةفي الغالب. لذلك سيكون من الأسهل العثور على

وحدات البكسل وتجزئة القطة خارج الصورة. هذا نهج غير خاضع للاشراف (unsupervised approach) للتجزئة.

6. النهج الثاني: تدريب نموذج من خلال إعطائه أمثلة واضحة للفئات التي تنتمي إلى الصورة ـ على وجه التحديد القط والأشجار والسماء. ثم احصل على تنبؤات النموذج على أي فئة موجودةفي مكان الصورة. هذا هو نهج خاضع للإشراف (supervised approach) للتجزئة.



على الرغم من أن كلا النهجين لهما إيجابيات وسلبيات، فإن قرار البدء بأي من النهجين سيعتمد على مدى صعوبة الحصول على أمثلة تدريبية لمتابعة النهج الخاضع للإشراف.

بيان المشكلة

دون إضاعة أي وقت، دعنا ننتقل إلى ماهية مشكلتنا الفعلية ونحاول حلها.

وفقًا لمنظمة الصحة العالمية، فإن أمراض القلب والأوعية الدموية cardiovascular diseases (CVDs) هي السبب الأول للوفاة على مستوى العالم: يموت عدد أكبر من الأشخاص سنويًا بسبب الأمراض القلبية الوعائية أكثر من أي سبب آخر. ما يقدر بنحو 17.1 مليون شخص ماتوا من الأمراض القلبية الوعائية (coronary heart disease)في عام 2004، وهو ما يمثل 29 ٪ من جميع الوفيات العالمية. من بين هذه الوفيات، كان ما يقدر بنحو 7.2 مليون بسبب أمراض القلب التاجية. وبالتالي فإن أي طريقة يمكن أن تساعدفي الكشف عن علامات أمراض القلب يمكن أن يكون لها تأثير كبير على الصحة العالمية. هذا التحدي هو إنتاج طرق للقيام بذلك بالضبط.

تتمثل المهمةفي التحديفي العثور على طريقة يمكنها تحديد موقع الأصوات الخاصة بالقلب (المعروف أيضًا باسم lub & dub ، والتي تسمى تقنيًا S1 و S2) ضمن البيانات الصوتية ثم تقسيم الملفات الصوتية على أساس هذه الأصوات. بعد تقسيم الأصوات، يطلب منا التحدي إنتاج طريقة يمكنها تصنيف ضربات القلب إلى فئات عادية ومريضة. لغرض هذه المقالة، سنتولى المهمة الأولى فقط من التحدي، أي تقسيم صوت القلب.

لإعطائك لمحة عملية، هكذا يبدو صوت القلب:

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 (عندما يكون معدل ضربات القلب أقل من 140 نبضةفي الدقيقة). وصف مؤقت للمواقع "lub" و "dub" بمرور الوقتفي الرسم التوضيحي التالي:

```
lub……..dub…….. lub……..dub…….. lub……..dub…….. lub……..dub
```

تنفيذ تجزئة صوت القلب

الخطوة الأساسية التي يتعين عليك القيام بها عندما تبدأفي حل مشكلة ما هي فهم البيانات وتصفحها سجلًا تلو الآخر. لنبدأ بهذا:

ملاحظة: يمكنك تنزيل مجموعة البيانات المطلوبة من صفحة الويب هذه. قم بتنزيل Dataset A من التحدي 1 فقط (Atraining_normal.zip و Atraining_normal_seg.csv)

```
# import modules%pylab inline

import librosa
import numpy as np
import pandas as pd
from librosa import display
```

```
# read csv filetemp = pd.read_csv('../misc/Atraining_normal_seg.csv')
temp.head()
```

```
In [3]: temp.head()
Out[3]:
```

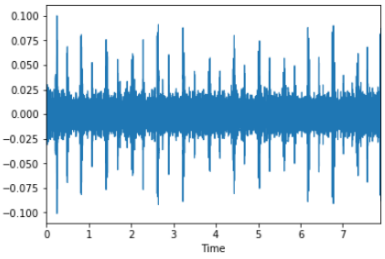
	Unnamed: 0	S1	S2	S1.1	S2.1	S1.2	S2.2	S1.3	S2.3	S1.4	...	S2.14	S1.15	S2.15	S1.16	S2.1
0	201102081321.aif	10021.0	20759	35075	47244	62992	73729	88761	101646	115246	...	NaN	NaN	NaN	NaN	Na
1	201102260502.aif	NaN	11526	27941	42197	58163	71278	88955	102641	122028	...	NaN	NaN	NaN	NaN	Na
2	201103090635.aif	5366.0	17632	31432	44464	59030	71296	86629	99661	116527	...	NaN	NaN	NaN	NaN	Na
3	201103140132.aif	16358.0	29272	89539	105036	128282	142057	170469	183383	207490	...	NaN	NaN	NaN	NaN	Na
4	201103140822.aif	3444.0	18080	44770	58545	84374	98149	123977	134309	157555	...	NaN	NaN	NaN	NaN	Na

5 rows × 40 columns

- دعونا نرسم عينة من نبضات القلب من مجموعة البيانات الخارجية:

```
# load sampledata, sampling_rate =
librosa.load('../misc/Atraining_normal/201102081321.wav', sr=44100 )
display.waveplot(data, sr=sampling_rate)
```

```
In [5]: display.waveplot(data, sr=sampling_rate)
Out[5]: <matplotlib.collections.PolyCollection at 0x1f78a74c278>
```



نرى أن هناك دورات من ضربات القلب، بصوت أعلى شدة يتبعه صوت أقل شدة.

- بالنسبة لمشكلتنا، سيتعين علينا إنشاء بيانات تدريب من الملفات الأولية. يقوم الكود أدناه بذلك ببساطة عن طريق الاطلاع على جميع الملفات الأولية، واستخراج جزء من الصوت مع التسمية الخاص به.

```
# create training data
data_x = []
data_y = []
for j in range(temp.shape[0]):
   for i in range(1, temp.shape[1] - 1):
     try:
       data, sampling_rate = librosa.load('../misc/Atraining_normal/'+
temp.iloc[j, 0].split('.')[0] +'.wav', sr=44100 )
       temp_data = data[int(temp.iloc[j, i]):int(temp.iloc[j, i+1])]
       temp_label = temp.iloc[:, i].name.split('.')[0]

       data_x.append(temp_data)
       data_y.append(temp_label)
     except:
       pass
```

- عندما نقوم بإنشاء هذه البيانات، يلزم القليل من المعالجة المسبقة (pre-processing). الأول هو جعل جميع العينات المستخرجة من نفس الشكل، والثاني هو تسوية البيانات (data normalization) والثالث هو إنشاء X و Y المناسبين لنموذج التعلم العميق الخاص بنا.

```
# preprocessing from keras.preprocessing.sequence import pad_sequences

# step 1data_x = pad_sequences(data_x, maxlen=20000, dtype='float',
padding='post', truncating='post', value=0.)

# step 2data_x = data_x / np.max(data_x)

# step 3
data_x = data_x[:,:,np.newaxis]
data_y = pd.Series(data_y)
data_y.value_counts()

data_y = data_y.map({'S1':0, 'S2':1}).values
```

- الآن دعونا نبني نموذج التعلم العميق الخاص بنا. سنقوم ببناء نموذج CNN، حيث أثبتت CNN أنها بُنية حديثة لفهم التسلسل والتصنيف.

```
from keras.layers import InputLayer, Conv1D, Dense, Flatten, MaxPool1D
from keras.models import Sequential
model = Sequential()
model.add(InputLayer(input_shape=data_x.shape[1:]))
model.add(Conv1D(filters=50, kernel_size=10, activation='relu'))
model.add(MaxPool1D(strides=8))
model.add(Conv1D(filters=50, kernel_size=10, activation='relu'))
model.add(MaxPool1D(strides=8))
model.add(Flatten())
model.add(Dense(units=1, activation='softmax'))
```

```
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

- سيحتوي نموذجنا على هذا النوع من المعمارية:

```
In [19]: model.summary()
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 20000, 1)	0
conv1d_1 (Conv1D)	(None, 19991, 50)	550
max_pooling1d_1 (MaxPooling1	(None, 2499, 50)	0
conv1d_2 (Conv1D)	(None, 2490, 50)	25050
max_pooling1d_2 (MaxPooling1	(None, 312, 50)	0
flatten_1 (Flatten)	(None, 15600)	0
dense_1 (Dense)	(None, 1)	15601

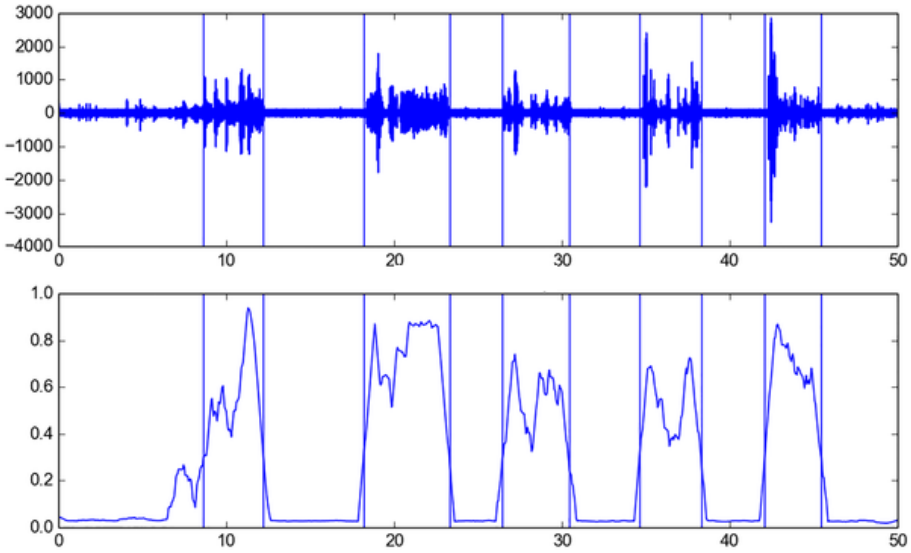
```
Total params: 41,201
Trainable params: 41,201
Non-trainable params: 0
```

- الخطوة التالية هي تدريب النموذج الخاص بك على مجموعة البيانات المحولة (transformed dataset).

```
# train modelmodel.fit(data_x, data_y, batch_size=32, epochs=1)
```

نحن نقصر التدريب على حقبة (epoch) واحدة فقط هنا. ولكن يمكنك زيادة هذا لتحسين أداء النموذج الخاص بك.

لديك نموذج مدرب يمكن استخدامه لأداء مهمة التجزئة. الآن للحصول على الفترات الزمنية التي يجب فيها تجزئة نبضات القلب، ما عليك سوى تقسيم ملف الاختبار الأولي إلى أجزاء متعددة والحصول على أفضل التنبؤات منه. سيعطي النموذج تنبؤًا مثل هذا:



الملخص

آمل أن يكون هذا المقال قد أعطاك لمحة عن كيف يمكن أن تساعدنا التطوراتفي التحليل الصوتيفي إنشاء تقنيات مذهلة يمكن أن تغير حياتنا. يمكن أن تكون الاحتمالات التي يفتحها للبشر هائلة.

25) تشخيص COVID-19 باستخدام التعلم العميق COVID-19 Diagnosis using Deep Learning

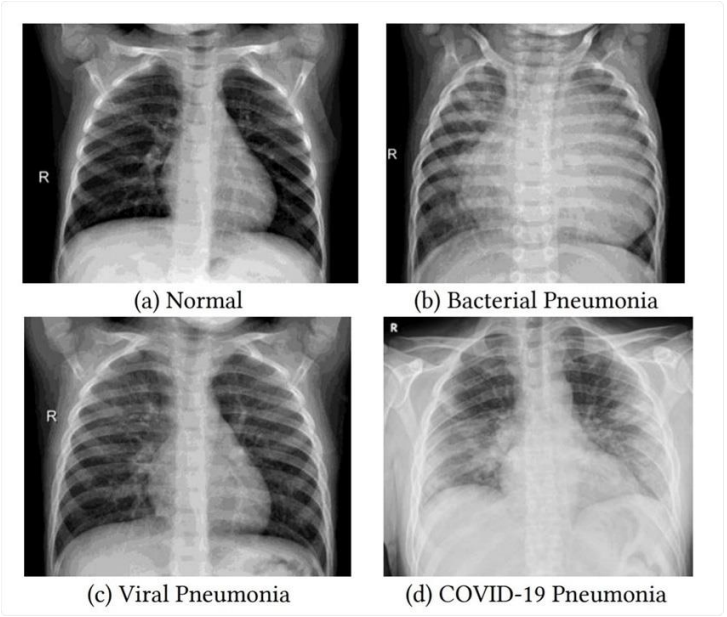
مقدمة

الوباء العالمي المستمر المسمى COVID-19 ناجم عن SARS-COV-2، الذي تم اكتشافه وتحديده لأول مرةفي ووهان، الصين،في ديسمبر 2019. فشل الإغلاقفي ووهان والمناطق المجاورة لهافي احتواء هذا الفيروس الذي أدى إلى أحد أسوأ الأزمات الإنسانيةفي العالم الحديث، والتي تؤثر على ملايين الأشخاصفي جميع أنحاء العالم.

وسرعان ما انتشر الفيروس وتحور مما أدى إلى ظهور موجات عديدة منه أثرتفي الغالب على دول العالم الثالث والدول النامية. يتزايد عدد الأشخاص المصابين بشكل كبير حيث تحاول حكومات العالم السيطرة على انتشار المرض.

في هذه المقالة، سوف نستخدم مجموعة بيانات CoronaHack- Chest X-Ray. يحتوي على صور للصدر بالأشعة السينية وعلينا أن نجد الصور المصابة بفيروس كورونا.

إن فيروس SARS-COV-2 الذي تحدثنا عنه سابقًا هو نوع الفيروس الذي يؤثر بشكل كبير على الجهاز التنفسي (respiratory system)، لذا فإن تصوير الصدر بالأشعة السينية (X-ray) هو أحد طرق التصوير المهمة التي يمكننا استخدامها لتحديد الرئة المصابة. إليك مقارنة جنبًا إلى جنب:



كما ترون، كيف يمكن أن يبتلع الالتهاب الرئوي (pneumonia) لـ COVID-19 الرئتين بالكامل وهو أكثر خطورة من الالتهاب الرئوي البكتيري والفيروسي.

في هذه المقالة، سوف نستخدم التعلم العميق ونقل التعلم لتصنيف وتحديد صور الأشعة السينية للرئتين المصابة بـ Covid-19.

استيراد المكتبات وتحميل البيانات

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import numpy as np
import pandas as pd
sns.set()
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import  *
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from tensorflow.keras.applications import DenseNet121, VGG19, ResNet50

import PIL.Image
import matplotlib.pyplot as mpimg
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
img_to_array
from tensorflow.keras.preprocessing import image

from tqdm import tqdm
import warnings
warnings.filterwarnings("ignore")

from sklearn.utils import shuffle
```

```
train_df = pd.read_csv('../input/coronahack-chest-
xraydataset/Chest_xray_Corona_Metadata.csv')
train_df.shape
```

```
> (5910, 6)
```

```
train_df.head(5)
```

Out[3]:

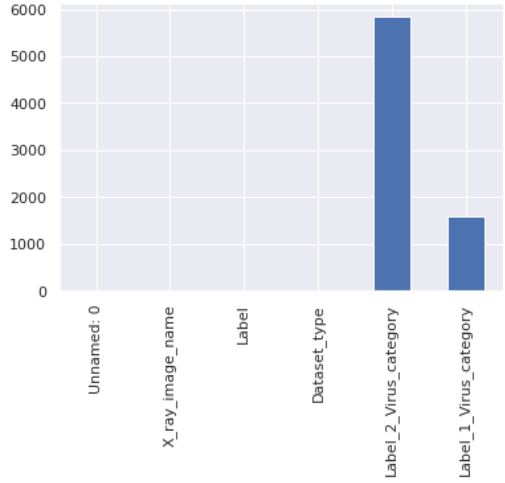
	Unnamed: 0	X_ray_image_name	Label	Dataset_type	Label_2_Virus_category	Label_1_Virus_category
0	0	IM-0128-0001.jpeg	Normal	TRAIN	NaN	NaN
1	1	IM-0127-0001.jpeg	Normal	TRAIN	NaN	NaN
2	2	IM-0125-0001.jpeg	Normal	TRAIN	NaN	NaN
3	3	IM-0122-0001.jpeg	Normal	TRAIN	NaN	NaN
4	4	IM-0119-0001.jpeg	Normal	TRAIN	NaN	NaN

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5910 entries, 0 to 5909
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              5910 non-null   int64
 1   X_ray_image_name        5910 non-null   object
 2   Label                   5910 non-null   object
 3   Dataset_type            5910 non-null   object
 4   Label_2_Virus_category  69 non-null     object
 5   Label_1_Virus_category  4334 non-null   object
dtypes: int64(1), object(5)
memory usage: 277.2+ KB
```

التعامل مع القيم المفقودة

```
missing_vals = train_df.isnull().sum()
missing_vals.plot(kind = 'bar')
```




```
train_df.dropna(how = 'all')
train_df.isnull().sum()
```

Out[6]:

```
Unnamed: 0                  0
X_ray_image_name            0
Label                       0
Dataset_type                0
Label_2_Virus_category    5841
Label_1_Virus_category    1576
dtype: int64
```

```
train_df.fillna('unknown', inplace=True)
train_df.isnull().sum()
```

Out[7]:

```
Unnamed: 0                0
X_ray_image_name          0
Label                     0
Dataset_type              0
Label_2_Virus_category    0
Label_1_Virus_category    0
dtype: int64
```

```
train_data = train_df[train_df['Dataset_type'] == 'TRAIN']
test_data = train_df[train_df['Dataset_type'] == 'TEST']
assert train_data.shape[0] + test_data.shape[0] == train_df.shape[0]
print(f"Shape of train data : {train_data.shape}")
print(f"Shape of test data : {test_data.shape}")
test_data.sample(10)
```

```
Shape of train data : (5286, 6)
Shape of test data : (624, 6)
```

	Unnamed: 0	X_ray_image_name	Label	Dataset_type	Label_2_Virus_category	Label_1_Virus_category
5752	5775	person83_bacteria_412.jpeg	Pnemonia	TEST	unknown	bacteria
5850	5873	person1_virus_11.jpeg	Pnemonia	TEST	unknown	Virus
5348	5371	IM-0070-0001.jpeg	Normal	TEST	unknown	unknown
5394	5417	NORMAL2-IM-0313-0001.jpeg	Normal	TEST	unknown	unknown
5469	5492	NORMAL2-IM-0123-0001.jpeg	Normal	TEST	unknown	unknown
5502	5525	NORMAL2-IM-0359-0001.jpeg	Normal	TEST	unknown	unknown
5550	5573	person157_bacteria_740.jpeg	Pnemonia	TEST	unknown	bacteria
5675	5698	person112_bacteria_539.jpeg	Pnemonia	TEST	unknown	bacteria
5858	5881	person173_bacteria_831.jpeg	Pnemonia	TEST	unknown	bacteria
5567	5590	person150_bacteria_715.jpeg	Pnemonia	TEST	unknown	bacteria

سنملأ القيم المفقودة بـ "unknown".

```
print((train_df['Label_1_Virus_category']).value_counts())
print('--------------------------')
print((train_df['Label_2_Virus_category']).value_counts())
```

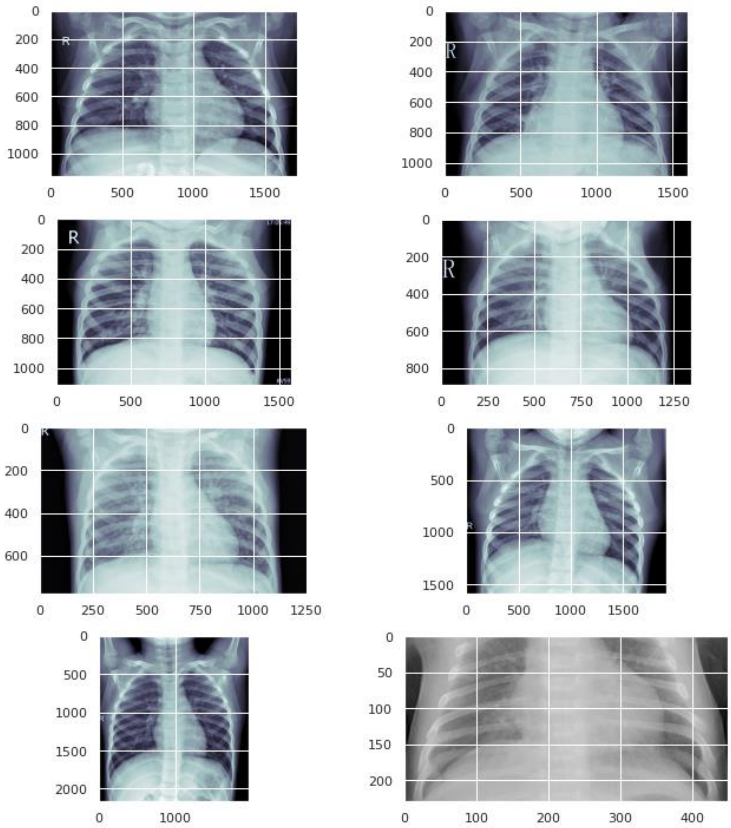
```
bacteria          2777
unknown           1576
Virus             1555
Stress-Smoking       2
Name: Label_1_Virus_category, dtype: int64
-------------------------
unknown          5841
COVID-19           58
Streptococcus       5
SARS                4
ARDS                2
Name: Label_2_Virus_category, dtype: int64
```

وبالتالي فإن فئة Label 2 تحتوي على حالات COVID-19!

عرض الصور

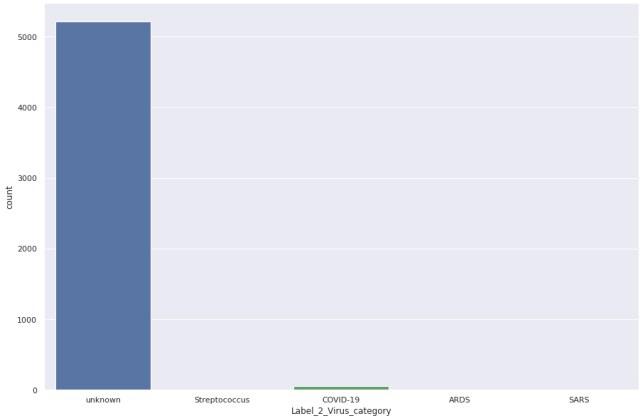
```
test_img_dir = '/kaggle/input/coronahack-chest-xraydataset/Coronahack-
Chest-XRay-Dataset/Coronahack-Chest-XRay-Dataset/test'
train_img_dir = '/kaggle/input/coronahack-chest-xraydataset/Coronahack-
Chest-XRay-Dataset/Coronahack-Chest-XRay-Dataset/train'

sample_train_images = list(os.walk(train_img_dir))[0][2][:8]
sample_train_images = list(map(lambda x: os.path.join(train_img_dir, x),
sample_train_images))

sample test images = list(os.walk(test img dir))[0][2][:8]
sample_test_images = list(map(lambda x: os.path.join(test_img_dir, x),
sample_test_images))
```

```
plt.figure(figsize = (10,10))
for iterator, filename in enumerate(sample_train_images):
    image = PIL.Image.open(filename)
    plt.subplot(4,2,iterator+1)
    plt.imshow(image, cmap=plt.cm.bone)

plt.tight_layout()
```



رسم النتائج

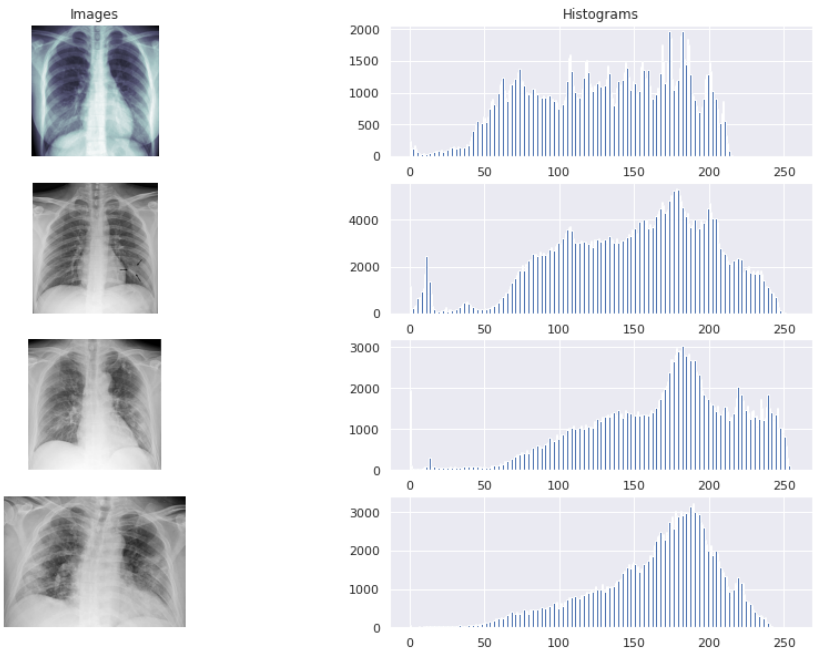
```
plt.figure(figsize=(15,10))
sns.countplot(train_data['Label_2_Virus_category']);
```



لحالات COVID-19:

```
fig, ax = plt.subplots(4, 2, figsize=(15, 10))
```

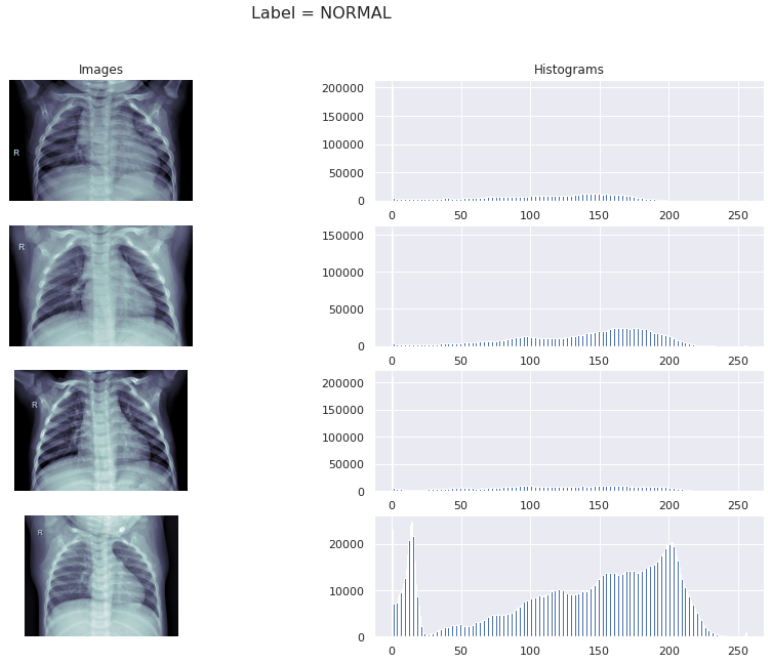
```
covid_path = train_data[train_data['Label_2_Virus_category']=='COVID-
19']['X_ray_image_name'].values

sample covid path = covid path[:4]
sample_covid_path = list(map(lambda x: os.path.join(train_img_dir, x),
sample_covid_path))

for row, file in enumerate(sample_covid_path):
    image = plt.imread(file)
    ax[row, 0].imshow(image, cmap=plt.cm.bone)
    ax[row, 1].hist(image.ravel(), 256, [0,256])
    ax[row, 0].axis('off')
    if row == 0:
        ax[row, 0].set_title('Images')
        ax[row, 1].set_title('Histograms')
fig.suptitle('Label 2 Virus Category = COVID-19', size=16)
plt.show()
```



للحالات العادية:

```
fig, ax = plt.subplots(4, 2, figsize=(15, 10))


normal_path =
train_data[train_data['Label']=='Normal']['X_ray_image_name'].values

sample_normal_path = normal_path[:4]
sample_normal_path = list(map(lambda x: os.path.join(train_img_dir, x),
sample_normal_path))

for row, file in enumerate(sample_normal_path):
```

```
    image = plt.imread(file)
    ax[row, 0].imshow(image, cmap=plt.cm.bone)
    ax[row, 1].hist(image.ravel(), 256, [0,256])
    ax[row, 0].axis('off')
    if row == 0:
        ax[row, 0].set_title('Images')
        ax[row, 1].set_title('Histograms')
fig.suptitle('Label = NORMAL', size=16)
plt.show()
```




```
final_train_data = train_data[(train_data['Label'] == 'Normal') |
                              ((train_data['Label'] == 'Pnemonia') &
                               (train data['Label 2 Virus category'] ==
'COVID-19'))]
```

```
final_train_data['class'] = final_train_data.Label.apply(lambda x:
'negative' if x=='Normal' else 'positive')
test_data['class'] = test_data.Label.apply(lambda x: 'negative' if
x=='Normal' else 'positive')

final_train_data['target'] = final_train_data.Label.apply(lambda x: 0 if
x=='Normal' else 1)
test_data['target'] = test_data.Label.apply(lambda x: 0 if x=='Normal'
else 1)
```

```
final_train_data = final_train_data[['X_ray_image_name', 'class',
'target', 'Label_2_Virus_category']]
final_test_data = test_data[['X_ray_image_name', 'class', 'target']]
```

```
test_data['Label'].value_counts()
```

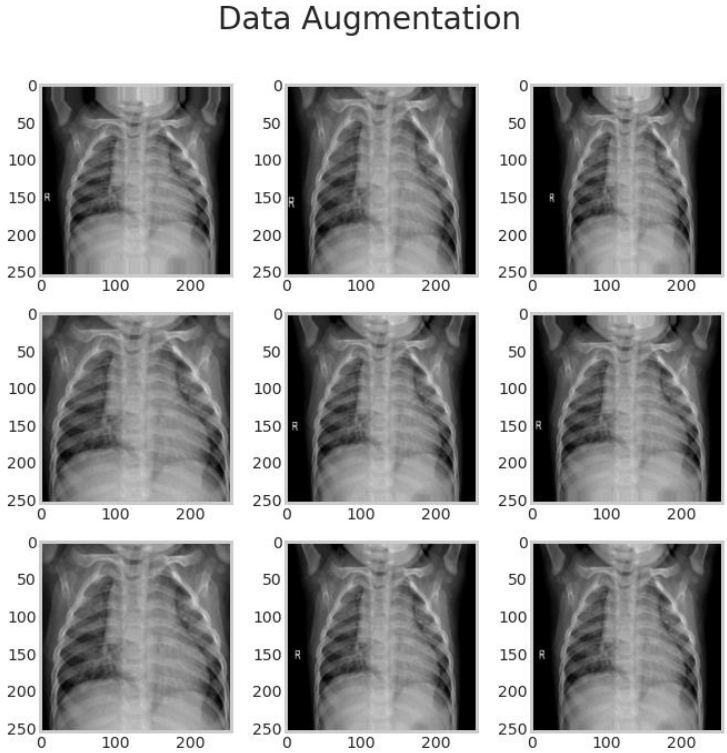
Out[15]:

```
Pnemonia    390
Normal      234
Name: Label, dtype: int64
```

زيادة البيانات

```
datagen =  ImageDataGenerator(
  shear_range=0.2,
  zoom_range=0.2,
)

def read_img(filename, size, path):
    img = image.load_img(os.path.join(path, filename), target_size=size)
    #convert image to array
    img = image.img_to_array(img) / 255
    return img
```

```
samp_img = read_img(final_train_data['X_ray_image_name'][0],
                                  (255,255),
                                  train_img_path)

plt.figure(figsize=(10,10))
plt.suptitle('Data Augmentation', fontsize=28)

i = 0


for batch in datagen.flow(tf.expand_dims(samp_img,0), batch_size=6):
    plt.subplot(3, 3, i+1)
    plt.grid(False)
    plt.imshow(batch.reshape(255, 255, 3));

    if i == 8:
        break
    i += 1

plt.show();
```




```
corona_df = final_train_data[final_train_data['Label_2_Virus_category'] ==
'COVID-19']
with_corona_augmented = []


def augment(name):
    img = read_img(name, (255,255), train_img_path)
    i = 0
    for batch in tqdm(datagen.flow(tf.expand_dims(img, 0),
batch_size=32)):
        with_corona_augmented.append(tf.squeeze(batch).numpy())
        if i == 20:
            break
        i =i+1


corona_df['X_ray_image_name'].apply(augment)
```

ملاحظة: كان الناتج أطول من أن يتم تضمينهفي المقالة. هذا جزء صغير منه.

```
20it [00:00, 59.16it/s]
20it [00:00, 61.47it/s]
20it [00:00, 62.62it/s]
20it [00:00, 45.51it/s]
20it [00:00, 38.29it/s]
20it [00:00, 58.95it/s]
20it [00:00, 59.08it/s]
20it [00:00, 60.17it/s]
20it [00:00, 61.57it/s]
20it [00:00, 62.97it/s]
20it [00:00, 62.54it/s]
20it [00:00, 63.79it/s]
20it [00:00, 64.53it/s]
20it [00:00, 63.31it/s]
```

```
train_arrays = []
final_train_data['X_ray_image_name'].apply(lambda x:
train_arrays.append(read_img(x, (255,255), train_img_dir)))
test_arrays = []
final_test_data['X_ray_image_name'].apply(lambda x:
test_arrays.append(read_img(x, (255,255), test_img_dir)))
```

```
print(len(train_arrays))
print(len(test_arrays))
```

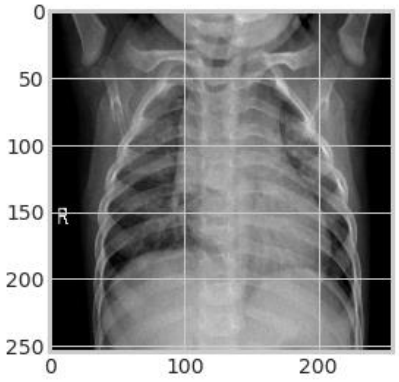
```
1400
624
```

```
y_train = np.concatenate((np.int64(final_train_data['target'].values),
np.ones(len(with_corona_augmented), dtype=np.int64)))
```

تحويل جميع البيانات إلى موتر

```
train_tensors =
tf.convert_to_tensor(np.concatenate((np.array(train_arrays),
np.array(with_corona_augmented))))
test tensors  = tf.convert to tensor(np.array(test arrays))
y_train_tensor = tf.convert_to_tensor(y_train)
y_test_tensor = tf.convert_to_tensor(final_test_data['target'].values)

train_dataset = tf.data.Dataset.from_tensor_slices((train_tensors,
y_train_tensor))
test_dataset = tf.data.Dataset.from_tensor_slices((test_tensors,
y_test_tensor))
```

```
for i,l in train_dataset.take(1):
    plt.imshow(i);
```



توليد الدفعات

```
BATCH_SIZE = 16
BUFFER = 1000

train_batches = train_dataset.shuffle(BUFFER).batch(BATCH_SIZE)
test_batches = test_dataset.batch(BATCH_SIZE)

for i,l in train_batches.take(1):
    print('Train Shape per Batch: ',i.shape);
for i,l in test_batches.take(1):
    print('Test Shape per Batch: ',i.shape);
```

```
Train Shape per Batch:  (16, 255, 255, 3)
Test Shape per Batch:  (16, 255, 255, 3)
```

نقل التعلم مع ResNet50

```
INPUT_SHAPE = (255,255,3)

base_model = tf.keras.applications.ResNet50(input_shape= INPUT_SHAPE,
                                                    include_top=False,
                                                    weights='imagenet')

# We set it to False because we don't want to mess with the pretrained
weights of the model.
base_model.trainable = False
```

الآن أصبح تعلم التحويل ناجحًا!!

```
Total params: 23,587,712
Trainable params: 0
Non-trainable params: 23,587,712
```

```
for i,l in train_batches.take(1):
    pass
base_model(i).shape
> TensorShape([16, 8, 8, 2048])
```

إضافة طبقة كثيفة لتصنيف الصور

```
model = Sequential()
model.add(base_model)
model.add(Layers.GlobalAveragePooling2D())
```

```
model.add(Layers.Dense(128))
model.add(Layers.Dropout(0.2))
model.add(Layers.Dense(1, activation = 'sigmoid'))
model.summary()
callbacks = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 8, 8, 2048)        23587712
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 23,850,113
Trainable params: 262,401
Non-trainable params: 23,587,712
_________________________________________________________________
```

```
patience=2)


model.compile(optimizer='adam',
              loss = 'binary_crossentropy',
              metrics=['accuracy'])
```

التنبؤ

```
model.fit(train_batches, epochs=10, validation_data=test_batches,
callbacks=[callbacks])
```

```
Epoch 1/10
164/164 [==============================] - 12s 73ms/step - loss: 0.6019 - accuracy: 0.7846 -
val_loss: 0.5413 - val_accuracy: 0.7115
Epoch 2/10
164/164 [==============================] - 10s 62ms/step - loss: 0.3667 - accuracy: 0.8724 -
val_loss: 0.5326 - val_accuracy: 0.7532
Epoch 3/10
164/164 [==============================] - 10s 62ms/step - loss: 0.2427 - accuracy: 0.9240 -
val_loss: 0.5632 - val_accuracy: 0.7596
Epoch 4/10
164/164 [==============================] - 10s 60ms/step - loss: 0.1744 - accuracy: 0.9488 -
val_loss: 0.7198 - val_accuracy: 0.7147

<tensorflow.python.keras.callbacks.History at 0x7fc4a05c94d0>
```

```
pred = model.predict_classes(np.array(test_arrays))
```

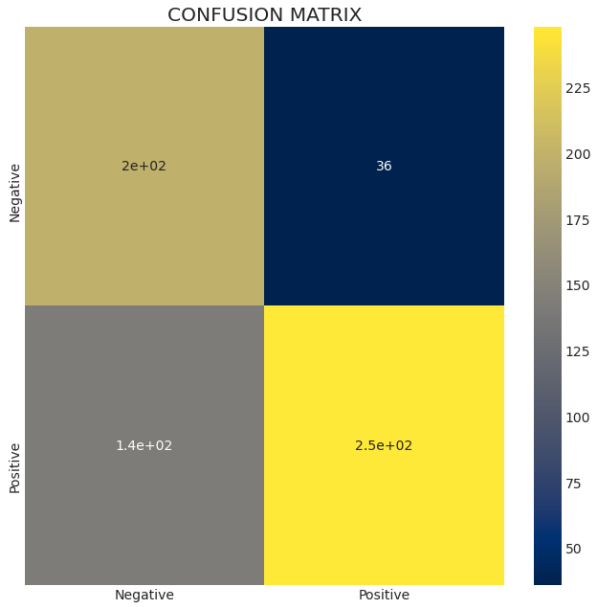
```
# classification report
from sklearn.metrics import classification_report, confusion_matrix
```

```
print(classification_report(test_data['target'], pred.flatten()))
```

```
              precision    recall  f1-score   support

           0       0.58      0.85      0.69       234
           1       0.87      0.64      0.74       390

    accuracy                           0.71       624
   macro avg       0.73      0.74      0.71       624
weighted avg       0.76      0.71      0.72       624
```

لذا كما ترى فإن التنبؤ ليس سيئاً. سنرسم مصفوفة ارتباك (confusion matrix) لتصوير أداء نموذجنا:

```
con_mat = confusion_matrix(test_data['target'], pred.flatten())
plt.figure(figsize = (10,10))
plt.title('CONFUSION MATRIX')
sns.heatmap(con_mat, cmap='cividis',
            yticklabels=['Negative', 'Positive'],
            xticklabels=['Negative', 'Positive'],
            annot=True);
```



الملخص

كانت مجموعة البيانات هذه مثيرة للاهتمام، وكلما تعلمت علوم البيانات والتعلم الآلي، وجدت هذا الموضوع ممتعًا أكثر. هناك العديد من الطرق التي يمكننا من خلالها استخدام البياناتفي الوقت الحاضر ويمكن أن ينقذ استخدامها عددًا لا يحصى من الأرواح.

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 In Healthcare

25 Deep Learning Projects in Healthcare Solved and Explained with Python

By: Dr. Alaa Taima