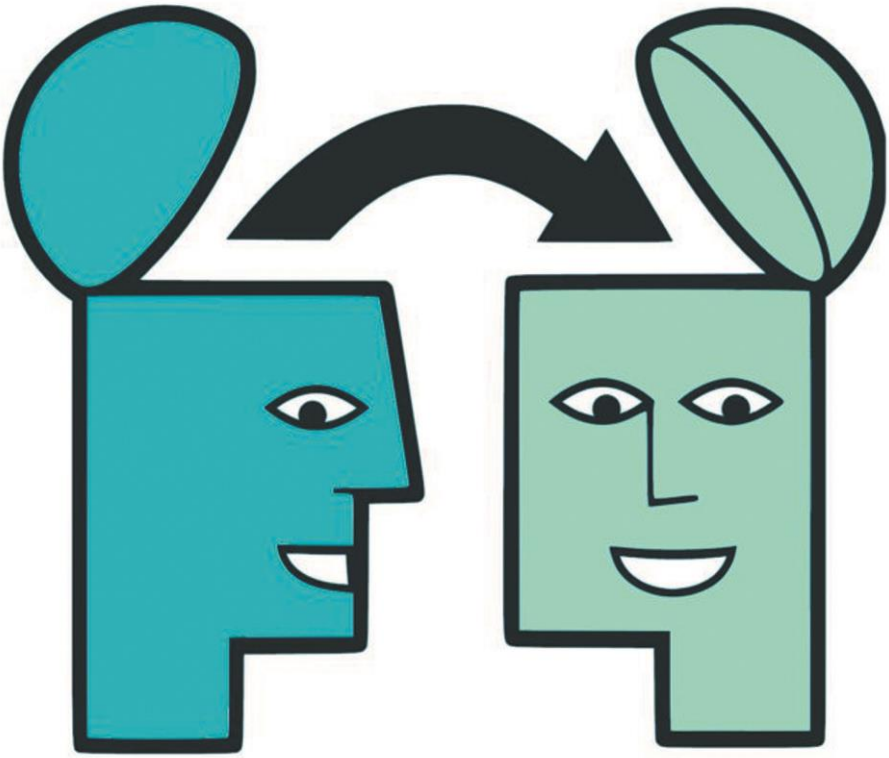
نقل التعلم
في
الرؤية الحاسوبية

20 مشروع نقل تعلم في الرؤية الحاسوبية تم حلها وشرحها باستخدام بايثون

ترجمة واعداد: د. علاء طعيمة



بـــمه تعالى

نقل التعلم في الرؤية الحاســوبية

20 مشروع نقل تعلم في الرؤية الحاسوبية تم حلها وشرحها باستخدام بايثون

ترجمة واعداد:
د. علاء طعيمة

مقدمة

قد تستغرق نماذج الشبكة العصبية التلافيفية العميقة أيامًا أو حتى أسابيع للتدريب على مجموعات بيانات كبيرة جداً.

تتمثل إحدى طرق اختصار هذه العمليةفي إعادة استخدام أوزان النموذج من النماذج المدربة مسبقًا pre-trained models التي تم تطويرها لمجموعات البيانات المعيارية لقياس الرؤية الحاسوبية، مثل مهام التعرف على الصور ImageNet. يمكن تنزيل النماذج الأفضل أداءً واستخدامها مباشرةً، أو دمجهافي نموذج جديد لمشكلات الرؤية الحاسوبية لديك.

في هذا الكتاب، سوف تكتشف عن طريق المشاريع كيفية استخدام نقل التعلم transfer learning عند تطوير الشبكات العصبية التلافيفية CNN لتطبيقات الرؤية الحاسوبية.

لقد حاولت قدر المستطاع ان اترجم المشاريع الأكثر طرحاًفي مجال نقل التعلم والرؤية الحاسوبية 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 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. ونرجو لك الاستمتاع مع الكتاب ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية

العراق

المحتويات

0) مقدمة بسيطة لنقل التعلم للتعلم العميق A Gentle Introduction to Transfer Learning for Deep Learning

نقل التعلم Transfer learning هو طريقة تعلم الآلة Machine Learning حيث يتم إعادة استخدام نموذج تم تطويره لمهمة ما كنقطة بداية لنموذج في مهمة ثانية.

إنه نهج شائع في التعلم العميق حيث يتم استخدام النماذج المدربة مسبقًا pre-trained models كنقطة انطلاق للرؤية الحاسوبية computer vision ومهام معالجة اللغة الطبيعية natural language processing نظرًا لموارد الحوسبة والوقت الهائلة المطلوبة لتطوير نماذج الشبكة العصبية بشأن هذه المشكلات ومن القفزات الهائلة في المهارات التي يقدمونها بشأن المشكلات ذات الصلة.

في هذه المقالة، سوف تكتشف كيف يمكنك استخدام نقل التعلم لتسريع التدريب وتحسين أداء نموذج التعلم العميق الخاص بك.

بعد قراءة هذه المقالة، ستعرف:

- ما هو نقل التعلم وكيفية استخدامه.
- أمثلة شائعة لنقل التعلم في التعلم العميق.
- متى تستخدم نقل التعلم في مشاكل النمذجة التنبؤية الخاصة بك.

هيا بنا نبدأ.

للحصول على مثال حول كيفية استخدام التعلم بالنقل في الرؤية الحاسوبية، راجع المنشور:

- نقل التعلم في Keras باستخدام نماذج الرؤية الحاسوبية.

ما هو نقل التعلم؟

نقل التعلم هو أسلوب تعلم آلي حيث يتم إعادة تصميم نموذج تم تدريبه على مهمة واحدة في مهمة ثانية ذات صلة.

يشير نقل التعلم وتكييف المجال إلى الموقف الذي يتم فيه استغلال ما تم تعلمه في أحد الأماكن ... لتحسين التعميم في بيئة أخرى

ـ صفحة 526، التعلم العميق ، 2016.

نقل التعلم هو تحسين يسمح بإحراز تقدم سريع أو أداء محسن عند نمذجة المهمة الثانية.

نقل التعلم هو تحسين التعلمفي مهمة جديدة من خلال نقل المعرفة من مهمة ذات صلة تم تعلمها بالفعل.

ـ الفصل 11: نقل التعلم، كتيب البحثفي تطبيقات التعلم الآلي، 2009.

يرتبط نقل التعلم بمشكلات مثل التعلم متعدد المهام multi-task learning وانحراف المفاهيم وليس مجالًا للدراسة فقط للتعلم العميق.

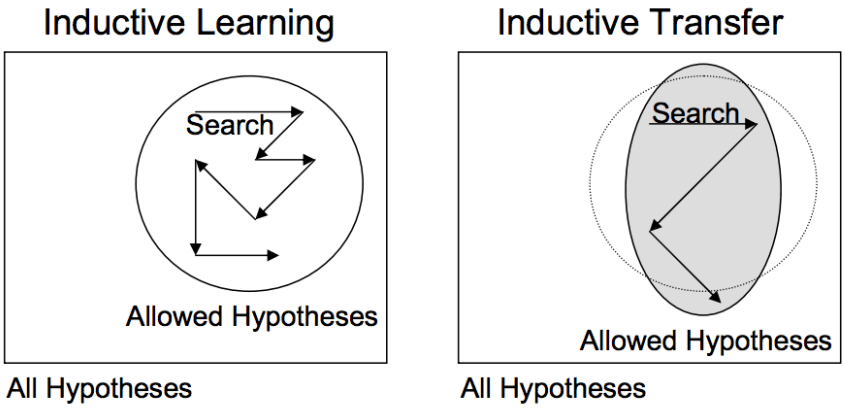
ومع ذلك، فإن نقل التعلم شائعفي التعلم العميق نظرًا للموارد الهائلة المطلوبة لتدريب نماذج التعلم العميق أو مجموعات البيانات الكبيرة والصعبة التي يتم تدريب نماذج التعلم العميق عليها.

لا يعمل نقل التعلم إلافي التعلم العميق إذاكانت ميزات النموذج التي تم تعلمها من المهمة الأولى عامة.

في نقل التعلم، نقوم أولاً بتدريب شبكة أساسية على مجموعة بيانات ومهمة أساسية، ثم نقوم بإعادة توظيف الميزات المكتسبة، أو نقلها، إلى شبكة هدف ثانية ليتم تدريبها على مجموعة بيانات ومهمة مستهدفة. تميل هذه العملية إلى العمل إذاكانت الميزات عامة، مما يعني أنها مناسبة لكل من المهام الأساسية والهدف، بدلاً من المهمة المحددة للمهمة الأساسية.

ـ ما مدى قابلية نقل الميزاتفي الشبكات العصبية العميقة؟

يسمى هذا النوع من نقل التعلم المستخدمفي التعلم العميق بالنقل الاستقرائي inductive transfer. هذا هو المكان الذي يتم فيه تضييق نطاق النماذج الممكنة (تحيز النموذج) بطريقة مفيدة باستخدام نموذج مناسب لمهمة مختلفة ولكنها ذات صلة.



كيفية استخدام نقل التعلم؟

يمكنك استخدام نقل التعلمفي مشاكل النمذجة التنبؤية الخاصة بك.

نهجان شائعان هما كما يلي:

1. تطوير نهج نموذجي Develop Model Approach.
2. نهج النموذج المدربين مسبقًا Pre-trained Model Approach.

نهج تطوير النموذج

1. **حدد مهمة المصدر Select Source Task**. يجب عليك تحديد مشكلة النمذجة التنبؤية ذات الصلة مع وفرة البيانات حيث توجد علاقة مافي بيانات الإدخال و / أو بيانات الإخراج و / أو المفاهيم التي تم تعلمها أثناء التعيين من بيانات الإدخال إلى بيانات الإخراج.
2. **تطوير نموذج المصدر Develop Source Model**. بعد ذلك، يجب عليك تطوير نموذج ماهر لهذه المهمة الأولى. يجب أن يكون النموذج أفضل من نموذج ساذج لضمان تنفيذ بعض ميزات التعلم.
3. **إعادة استخدام النموذج Reuse Model**. يمكن بعد ذلك استخدام النموذج الملائم للمهمة المصدر كنقطة بداية لنموذجفي المهمة الثانية ذات الأهمية. قد يشمل ذلك استخدام النموذج بالكامل أو أجزاء منه، اعتمادًا على تقنية النمذجة المستخدمة.
4. **ضبط النموذج Tune Model**. اختياريًا، قد يحتاج النموذج إلى تكييفه أو صقله على بيانات زوج الإدخال والإخراج المتاحة للمهمة محل الاهتمام.

نهج النموذج المدربين مسبقًا

1. **حدد النموذج المصدر Select Source Model**. يتم اختيار نموذج مصدر مدرب مسبقًا من النماذج المتاحة. تطلق العديد من المؤسسات البحثية نماذج على مجموعات بيانات كبيرة وصعبة يمكن تضمينهافي مجموعة النماذج المرشحة للاختيار من بينها.
2. **إعادة استخدام النموذج Reuse Model**. يمكن بعد ذلك استخدام النموذج الذي تم تدريبه مسبقًا كنقطة انطلاق لنموذجفي المهمة الثانية موضع الاهتمام. قد يشمل ذلك استخدام النموذج بالكامل أو أجزاء منه، اعتمادًا على تقنية النمذجة المستخدمة.
3. **ضبط النموذج Tune Model**. اختياريًا، قد يحتاج النموذج إلى تكييفه أو ضبطه على بيانات زوج الإدخال والإخراج المتاحة للمهمة محل الاهتمام.

هذا النوع الثاني من التعلم الانتقالي شائعفي مجال التعلم العميق.

أمثلة على نقل التعلم باستخدام التعلم العميق

دعنا نجعل ذلك ملموسًا من خلال مثالين شائعين لنقل التعلم باستخدام نماذج التعلم العميق.

نقل التعلم باستخدام بيانات الصورة

من الشائع إجراء نقل التعلم مع مشاكل النمذجة التنبؤية التي تستخدم بيانات الصورة كمدخلات.

قد تكون هذه مهمة تنبؤ تأخذ صورًا أو بيانات فيديو كمدخلات.

بالنسبة لهذه الأنواع من المشكلات، من الشائع استخدام نموذج التعلم العميق الذي تم تدريبه مسبقًا لمهمة تصنيف صور كبيرة وصعبة مثل مسابقة تصنيف الصور الفوتوغرافية لـ 1000 فئة من ImageNet.

تقوم المنظمات البحثية التي تطور نماذج لهذه المسابقة وتعمل بشكل جيدفي كثير من الأحيان بإصدار نموذجها النهائي بموجب ترخيص مسموح بإعادة الاستخدام. قد تستغرق هذه النماذج أيامًا أو أسابيع للتدريب على الأجهزة الحديثة.

يمكن تنزيل هذه النماذج ودمجها مباشرةفي النماذج الجديدة التي تتوقع بيانات الصورة كمدخلات.

ثلاثة أمثلة على نماذج من هذا النوع تشمل:

- Oxford VGG Model
- Google Inception Model
- Microsoft ResNet Model

لمزيد من الأمثلة، راجع Caffe Model Zoo حيث يتم مشاركة المزيد من النماذج المدربة مسبقًا.

هذا النهج فعال لأن الصور تم تدريبها على مجموعة كبيرة من الصور وتتطلب النموذج عمل تنبؤات على عدد كبير نسبيًا من الفئات، مما يتطلب بدوره أن يتعلم النموذج بكفاءة لاستخراج الميزات من الصور من أجل الأداء الجيد على المشكلة.

في مقررهم الدراسيفي جامعة ستانفورد حول الشبكات العصبية التلافيفية من أجل التعرف البصري، يحذر المؤلفون من الاختيار بعناية لمقدار النموذج المدرَّب مسبقًا لاستخدامهفي نموذجك الجديد.

> تعد ميزات [الشبكات العصبية التلافيفية] أكثر عموميةفي الطبقات المبكرة وأكثر تحديدًا لمجموعة البيانات الأصليةفي الطبقات اللاحقة
>
> ـ نقل التعلم ، CS231n الشبكات العصبية التلافيفية للتمييز البصري

نقل التعلم باستخدام بيانات اللغة

من الشائع إجراء نقل التعلم مع مشاكل معالجة اللغة الطبيعية التي تستخدم النص كمدخل أو إخراج.

بالنسبة لهذه الأنواع من المشكلات، يتم استخدام تضمين الكلمة word embedding وهو تعيين الكلمات لمساحة متجهية عالية الأبعاد حيث يكون لكلمات مختلفة ذات معنى مماثل تمثيل متجه مماثل.

توجد خوارزميات فعالة لتعلم تمثيلات الكلمات الموزعة هذه، ومن الشائع أن تقوم المنظمات البحثية بإصدار نماذج مدربة مسبقًا مدربة على مجموعة كبيرة جدًا من المستندات النصية بموجب ترخيص مسموح به.

من أمثلة النماذج من هذا النوع ما يلي:

- Google's word2vec Model
- Stanford's GloVe Model

يمكن تنزيل نماذج تمثيل الكلمات الموزعة هذه ودمجهافي نماذج لغة التعلم العميق إمافي تفسير الكلمات كمدخلات أو توليد الكلمات كمخرجات من النموذج.

في كتابه عن التعلم العميق لمعالجة اللغة الطبيعية، يحذر يوآف غولدبرغ:

> ... يمكن للمرء تنزيل متجهات الكلمات المدربة مسبقًا والتي تم تدريبها على كميات كبيرة جدًا من النص [...] الاختلافاتفي أنظمة التدريب والمجموعات الأساسية لها تأثير قوي على التمثيلات الناتجة، وأن العروض المتاحة المدربة مسبقًا قد لا تكون الأفضل اختيار حالة الاستخدام الخاصة بك.
>
> ـ صفحة 135 ، طرق الشبكة العصبيةفي معالجة اللغة الطبيعية ، 2017.

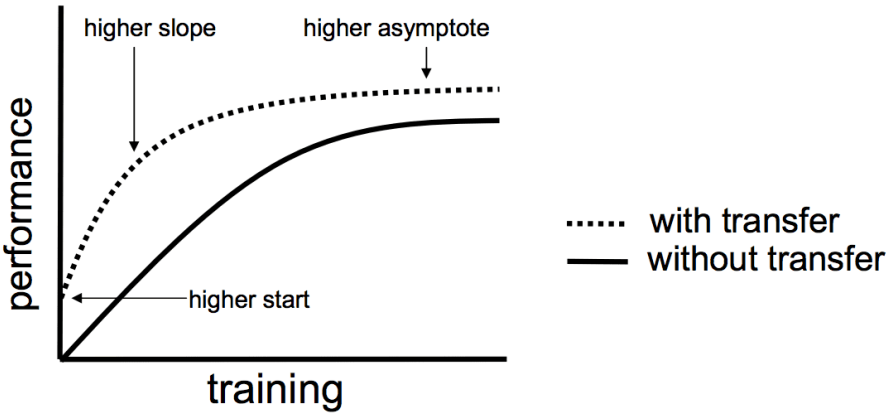
متى تستخدم نقل التعلم؟

نقل التعلم هو تحسين optimization، اختصار لتوفير الوقت أو الحصول على أداء أفضل.

بشكل عام، ليس من الواضح أنه ستكون هناك فائدة من استخدام نقل التعلمفي المجال إلا بعد تطوير النموذج وتقييمه.

تصف ليزا توري وجود شافليكفي الفصل الخاص بنقل التعلم ثلاث فوائد محتملة يجب البحث عنها عند استخدام نقل التعلم:

1. **بداية أعلى Higher start**. المهارة الأولية (قبل ضبط النموذج)في النموذج المصدر أعلى مما ستكون عليه.
2. **منحدر أعلى Higher slope**. معدل تحسين المهارة أثناء تدريب نموذج المصدر أكثر حدة مما كان يمكن أن يكون.
3. **خط مقارب أعلى Higher asymptote**. المهارة المتقاربة للنموذج المدرّب أفضل مما ستكون عليهفي غير ذلك.



من الناحية المثالية، سترى جميع الفوائد الثلاثة من التطبيق الناجح لنقل التعلم.

إنها طريقة لتجربتها إذاكان بإمكانك تحديد مهمة ذات صلة ببيانات وفيرة ولديك الموارد اللازمة لتطوير نموذج لتلك المهمة وإعادة استخدامهفي مشكلتك الخاصة، أو أن هناك نموذجًا مدربًا مسبقًا متاحًا يمكنك استخدامه نقطة انطلاق لنموذجك الخاص.

في بعض المشكلات التي قد لا يكون لديك فيها الكثير من البيانات، يمكن أن يمكّنك نقل التعلم من تطوير نماذج ماهرة لا يمكنك تطويرها ببساطةفي غياب نقل التعلم.

يعد اختيار مصدر البيانات أو نموذج المصدر مشكلة مفتوحة وقد يتطلب خبرةفي المجال و / أو حدسًا تم تطويره عبر التجربة.

الملخص

في هذه المقالة، اكتشفت كيف يمكنك استخدام نقل التعلم لتسريع التدريب وتحسين أداء نموذج التعلم العميق الخاص بك.

على وجه التحديد، لقد تعلمت:

- ما هو نقل التعلم وكيف يتم استخدامهفي التعلم العميق.

- متى تستخدم نقل التعلم.
- أمثلة على نقل التعلم المستخدمفي الرؤية الحاسوبية ومهام معالجة اللغة الطبيعية.

المصدر:

https://machinelearningmastery.com/transfer-learning-for-deep-learning/

1)نقل التعلم وفن استخدام النماذج المدربة مسبقًا في التعلم العميق

Transfer learning and the art of using Pre-trained Models in Deep Learning

مقدمة

الشبكات العصبية Neural networks هي سلالة مختلفة من النماذج مقارنة بخوارزميات التعلم الآلي الخاضعة للإشراف supervised machine learning algorithms. لماذا اقول ذلك؟ هناك أسباب متعددة لذلك، ولكن أبرزها تكلفة تشغيل الخوارزميات على الجهاز.

في عالم اليوم، تعتبر ذاكرة الوصول العشوائي على الجهاز رخيصة ومتوفرة بكثرة. أنت بحاجة إلى مئات الجيجابايت من ذاكرة الوصول العشوائي لتشغيل مشكلة التعلم الآلي المعقدة للغاية تحت الإشراف ـ يمكن أن تكون ملكك مقابل القليل من الاستثمار / الإيجار. من ناحية أخرى، فإن الوصول إلى وحدات معالجة الرسومات GPU ليس رخيصًا. تحتاج إلى الوصول إلى مائة جيجابايت من VRAM على وحدات معالجة الرسوميات ـ لن يكون الأمر مباشرًا وسيتضمن تكاليف كبيرة.

الآن، قد يتغير ذلكفي المستقبل. ولكنفي الوقت الحالي، هذا يعني أنه يتعين علينا أن نكون أكثر ذكاءً بشأن الطريقة التي نستخدم بها مواردنافي حل مشكلات التعلم العميق. على وجه الخصوص، عندما نحاول حل مشاكل الحياة الواقعية المعقدةفي مجالات مثل التعرف على الصور والصوت. بمجرد أن يكون لديك بعض الطبقات المخفيةفي النموذج الخاص بك، فإن إضافة طبقة أخرى من الطبقة المخفية ستحتاج إلى موارد هائلة.

لحسن الحظ، هناك شيء يسمى "نقل التعلم Transfer Learning" والذي يمكننا من استخدام نماذج مدربة مسبقًا pre-trained models من أشخاص آخرين من خلال إجراء تغييرات صغيرة.في هذه المقالة، سأخبرك كيف يمكننا استخدام النماذج المدربة مسبقًا لتسريع حلولنا.

لمعرفة المزيد حول النماذج المدربة مسبقًا ونقل التعلم وحالات الاستخدام الخاصة بها، يمكنك الاطلاع على المقالات التالية:

- Deep Learning for Everyone: Master the Powerful Art of Transfer Learning using PyTorch
- Top 10 Pretrained Models to get you Started with Deep Learning (Part 1 — Computer Vision)
- 8 Excellent Pretrained Models to get you Started with Natural Language Processing (NLP)

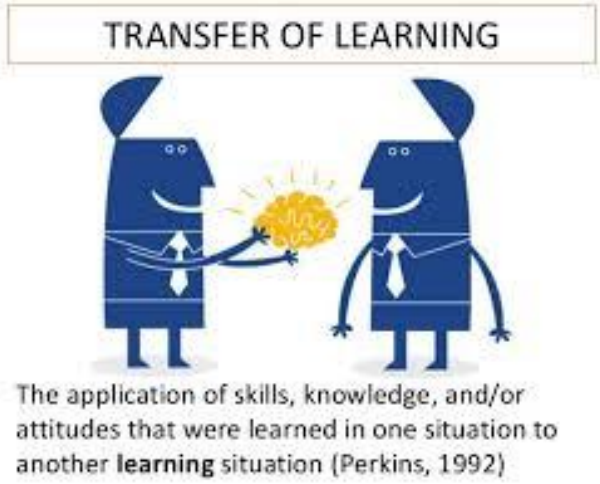
ملاحظة: تفترض هذه المقالة الإلمام الأساسي بالشبكات العصبية Neural networks والتعلم العميق deep learning. إذا كنت جديداًفي التعلم العميق، فإنني أوصي بشدة بقراءة المقالات التالية أولاً:

- What is deep learning and why is it getting so much attention?
- Deep Learning vs. Machine Learning – the essential differences you need to know!
- 25 Must Know Terms & concepts for Beginners in Deep Learning
- Why are GPUs necessary for training Deep Learning models?

ما هو نقل التعلم؟

دعونا نبدأ بتطوير حدس لنقل التعلم. دعونا نفهم من مدرس بسيط ـ تشبيه الطالب.

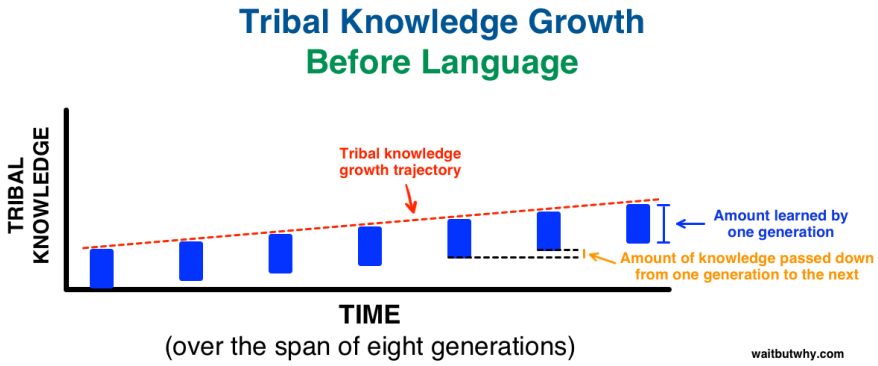
يتمتع المعلم بسنوات من الخبرةفي موضوع معين يقوم بتدريسه. مع كل هذه المعلومات المتراكمة، فإن المحاضرات التي يتلقاها الطلاب هي نظرة عامة موجزة وموجزة عن الموضوع. لذلك يمكن اعتباره بمثابة "نقل Transfer" للمعلومات من المتعلم إلى المبتدئ.



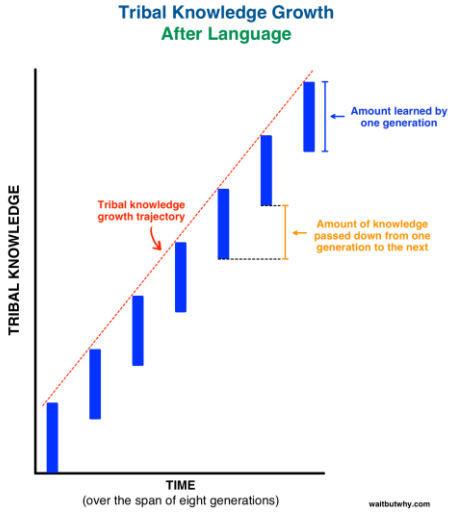
مع الأخذفي الاعتبار هذا التشبيه، نقارن هذا بالشبكة العصبية. يتم تدريب الشبكة العصبية على البيانات. تكتسب هذه الشبكة المعرفة من هذه البيانات، والتي يتم تجميعها على أنها "أوزان weights" للشبكة. يمكن استخراج هذه الأوزان ثم نقلها إلى أي شبكة عصبية أخرى. بدلاً من تدريب الشبكة العصبية الأخرى من البداية، نقوم "بنقل transfer" الميزات المكتسبة.

الآن، دعونا نفكرفي أهمية نقل التعلم من خلال الارتباط بتطورنا. وما هي أفضل طريقة من استخدام نقل التعلم لهذا الغرض! لذلك أنا أختار مفهومًا تطرق إليه Tim Urban من إحدى مقالاته الأخيرة على waitbutwhy.com

يوضح Tim أنه قبل اختراع اللغة، كان على كل جيل من البشر إعادة ابتكار المعرفة لأنفسهم، وهكذا كان نمو المعرفة يحدث من جيل إلى آخر:



ثم اخترعنا اللغة! طريقة لنقل التعلم من جيل إلى آخر وهذا ما حدث خلال نفس الإطار الزمني:



أليست هذه اللعبة استثنائية وذات قوة عظمى؟ لذا، فإن نقل التعلم عن طريق تمرير الأوزان يعادل اللغة المستخدمة لنشر المعرفة عبر الأجيالفي التطور البشري.

ما هو النموذج الذي تم تدريبه مسبقًا؟

ببساطة، النموذج المدرب مسبقًا هو نموذج تم إنشاؤه بواسطة شخص آخر لحل مشكلة مماثلة. بدلاً من بناء نموذج من البداية لحل مشكلة مماثلة، يمكنك استخدام النموذج المدرب على مشكلة أخرى كنقطة بداية.

على سبيل المثال، إذا كنت ترغبفي بناء سيارة التعلم الذاتي. يمكنك قضاء سنواتفي إنشاء خوارزمية جيدة للتعرف على الصور من البداية أو يمكنك أن تأخذ نموذجًا أوليًا (نموذجًا مدربًا مسبقًا) من Google والذي تم إنشاؤه على بيانات ImageNet لتحديد الصورفي تلك الصور.

قد لا يكون النموذج المدرَّب مسبقًا دقيقًا بنسبة 100٪في تطبيقك، ولكنه يوفر الجهود الضخمة المطلوبة لإعادة اختراع العجلة. دعني أريكم هذا بمثال حديث.

لماذا نستخدم النماذج المدربة مسبقًا؟

قضيت الأسبوع الماضي أعمل على حل مشكلةفي منصة CrowdAnalytix ـ تحديد السمات من صور حالة الهاتف المحمول. كانت هذه مشكلةفي تصنيف الصور حيث حصلنا على 4591 صورةفي مجموعة بيانات التدريب و1200 صورةفي مجموعة بيانات الاختبار. كان الهدف هو تصنيف الصور إلى واحدة من الفئات الـ 16. بعد خطوات المعالجة المسبقة الأساسية، بدأت بنموذج MLP بسيط مع البُنية التالية:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 150528)            0
_________________________________________________________________
dense_9 (Dense)              (None, 500)               75264500
_________________________________________________________________
activation_9 (Activation)    (None, 500)               0
_________________________________________________________________
dropout_7 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_10 (Dense)             (None, 500)               250500
_________________________________________________________________
activation_10 (Activation)   (None, 500)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_11 (Dense)             (None, 500)               250500
_________________________________________________________________
activation_11 (Activation)   (None, 500)               0
_________________________________________________________________
dropout_9 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_12 (Dense)             (None, 16)                8016
_________________________________________________________________
activation_12 (Activation)   (None, 16)                0
=================================================================
Total params: 75,773,516
Trainable params: 75,773,516
Non-trainable params: 0
```

لتبسيط البنية أعلاه بعد تسوية صورة الإدخال [224 X 224 X 3]في [150528]، استخدمت ثلاث طبقات مخفية تحتوي على 500 و 500 و 500 خلية عصبية على التوالي. تحتوي طبقة المخرجات على 16 خلية عصبية تتوافق مع عدد الفئات التي نحتاج فيها لتصنيف الصورة المدخلة.

بالكاد تمكنت من تحقيق دقة تدريب بلغت 6.8٪ واتضح أنها سيئة للغاية. حتى تجربة الطبقات المخفية hidden layers، وعدد الخلايا العصبيةفي الطبقات المخفية ومعدلات التسرب dropout rates. لم أتمكن من زيادة دقة التدريب بشكل كبير. أدت زيادة الطبقات المخفية وعدد الخلايا العصبية إلى تشغيل حقبة (epoch) واحدة لمدة 20 ثانية على وحدة معالجة الرسوميات Titan X GPU الخاصة بي مع ذاكرة VRAM بسعة 12 جيجابايت.

يوجد أدناه ناتج التدريب باستخدام نموذج MLP مع البنية المذكورة أعلاه

```
Epoch 10/10
50/50 [==============================] - 21s - loss: 15.0100 - acc:
0.0688
```

كما يمكنك أن ترى أن MLP لن تعطيني أي نتائج أفضل دون زيادة وقت التدريب بشكل كبير. لذلك انتقلت إلى الشبكة العصبية التلافيفية CNN لمعرفة كيفية أدائها على مجموعة البيانات هذه وما إذا كنت سأتمكن من زيادة دقة التدريب.

كان لدى CNN البُنية أدناه :

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 220, 220, 32)	2432
activation_27 (Activation)	(None, 220, 220, 32)	0
max_pooling2d_7 (MaxPooling2	(None, 55, 55, 32)	0
conv2d_8 (Conv2D)	(None, 51, 51, 32)	25632
activation_28 (Activation)	(None, 51, 51, 32)	0
max_pooling2d_8 (MaxPooling2	(None, 12, 12, 32)	0
conv2d_9 (Conv2D)	(None, 8, 8, 64)	51264
activation_29 (Activation)	(None, 8, 8, 64)	0
max_pooling2d_9 (MaxPooling2	(None, 2, 2, 64)	0
flatten_8 (Flatten)	(None, 256)	0
dense_21 (Dense)	(None, 64)	16448
activation_30 (Activation)	(None, 64)	0
dropout_12 (Dropout)	(None, 64)	0
dense_22 (Dense)	(None, 16)	1040
activation_31 (Activation)	(None, 16)	0

```
Total params: 96,816
Trainable params: 96,816
Non-trainable params: 0
```

تستخدم 3 كتل تلافيفية convolutional blocks مع كل كتلة تتبع العمارة أدناه:

- 32 فلتر مقاس 5×5.
- دالة التنشيط (Activation function) relu
- طبقة التجميع الأقصى (Max pooling layer) مقاس 4 × 4

النتيجة التي تم الحصول عليها بعد أن تم تسطيح الكتلة التلافيفية النهائية إلى حجم [256] وتمريرها إلى طبقة مخفية واحدة بها 64 خلية عصبية. تم تمرير ناتج الطبقة المخفية إلى طبقة الإخراج بعد معدل تسرب 0.5.

يتم تلخيص النتيجة التي تم الحصول عليها من خلال البُنية أعلاه :

على الرغم من زيادة دقتي مقارنةً بإخراج MLP، إلا أنها زادت أيضًا من الوقت المستغرق لتشغيل حقبة واحدة ـ 21 ثانية.

لكن النقطة الرئيسية التي يجب ملاحظتها هي أن فئة الأغلبيةفي مجموعة البيانات كانت حوالي 17.6٪. لذلك، حتى لو توقعنا أن تكون فئةكل صورةفي مجموعة بيانات التدريب هي فئة الأغلبية، فسنكون قد حققنا أداءً أفضل من MLP وCNN على التوالي. أدت إضافة المزيد من الكتل التلافيفية إلى زيادة وقت التدريب بشكل كبير. قادني هذا إلى التبديل إلى استخدام النماذج المدربة مسبقًا حيث لن أضطر إلى تدريب المعمارية بأكملها ولكن فقط بضع طبقات. لذلك، استخدمت نموذج VGG16 الذي تم تدريبه مسبقًا على مجموعة بيانات ImageNet وتم توفيرهفي مكتبة keras للاستخدام. يوجد أدناه بنية نموذج VGG16 الذي استخدمته.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0

```
block3_conv1 (Conv2D)           (None, 56, 56, 256)         295168
_________________________________________________________________
block3_conv2 (Conv2D)           (None, 56, 56, 256)         590080
_________________________________________________________________
block3_conv3 (Conv2D)           (None, 56, 56, 256)         590080
_________________________________________________________________
block3_pool (MaxPooling2D)      (None, 28, 28, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)           (None, 28, 28, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)           (None, 28, 28, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)           (None, 28, 28, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)      (None, 14, 14, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)           (None, 14, 14, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)           (None, 14, 14, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)           (None, 14, 14, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)      (None, 7, 7, 512)           0
_________________________________________________________________
flatten (Flatten)               (None, 25088)               0
_________________________________________________________________
fc1 (Dense)                     (None, 4096)                102764544
_________________________________________________________________
fc2 (Dense)                     (None, 4096)                16781312
_________________________________________________________________
predictions (Dense)             (None, 1000)                4097000
_________________________________________________________________
dense_1 (Dense)                 (None, 16)                  16016
=================================================================
Total params: 138,373,560
Trainable params: 138,373,560
Non-trainable params: 0
```

التغيير الوحيد الذي أجريته على البنية الحالية لـ VGG16 هو تغيير طبقة softmax مع 1000 إخراج إلى 16 فئة مناسبة لمشكلتنا وإعادة تدريب الطبقة الكثيفة dense layer.

أعطتني هذه البنية دقة 70٪ أفضل بكثير من MLP و CNN. أيضًا ، كانت أكبر فائدة لاستخدام نموذج VGG16 المدرَّب مسبقًا هي الوقت الضئيل تقريبًا لتدريب الطبقة الكثيفة بدقة أكبر.

لذلك، تقدمتفي هذا النهج المتمثلفي استخدام نموذج مدرب مسبقًا وكانت الخطوة التالية هي ضبط طراز VGG16 الخاص بي ليناسب هذه المشكلة.

كيف يمكنني استخدام النماذج المدربة مسبقًا؟

ما هو هدفنا عندما نقوم بتدريب شبكة عصبية؟ نرغبفي تحديد الأوزان الصحيحة للشبكة من خلال التكرارات المتعددة للأمام والخلف. من خلال استخدام النماذج المدربة مسبقًا والتي تم تدريبها مسبقًا على مجموعات البيانات الكبيرة، يمكننا استخدام الأوزان والبنية التي تم الحصول عليها بشكل مباشر وتطبيق التعلم على بيان المشكلة الخاص بنا. هذا هو المعروف باسم نقل التعلم. نحن "ننقل التعلم" للنموذج المدرب مسبقًا إلى بيان المشكلة المحدد لدينا.

يجب أن تكون حذرًا للغاية أثناء اختيار الطراز المدرب مسبقًا الذي يجب أن تستخدمهفي حالتك. إذا كانت عبارة المشكلة التي لدينا مختلفة تمامًا عن تلك التي تم تدريب النموذج المدرب مسبقًا عليها فإن التنبؤ الذي سنحصل عليه سيكون غير دقيق للغاية. على سبيل المثال، قد يعمل النموذج الذي تم تدريبه مسبقًا على التعرف على الكلام بشكل مروع إذا حاولنا استخدامه لتحديد الكائنات التي تستخدمها.

نحن محظوظون لأن العديد من الهياكل المدربة مسبقًا متاحة لنا مباشرةفي مكتبة Keras. تم استخدام مجموعة بيانات Imagenet على نطاق واسع لبناء هياكل مختلفة نظرًا لأنها كبيرة بما يكفي (1.2 مليون صورة) لإنشاء نموذج عام. بيان المشكلة هو تدريب نموذج يمكنه تصنيف الصور بشكل صحيح إلى 1000 فئة كائن منفصلة. تمثل فئات الصور التي يبلغ عددها 1000 فئات الكائنات التي نواجههافي حياتنا اليومية، مثل أنواع الكلاب والقطط والأشياء المنزلية المختلفة وأنواع المركبات وما إلى ذلك.

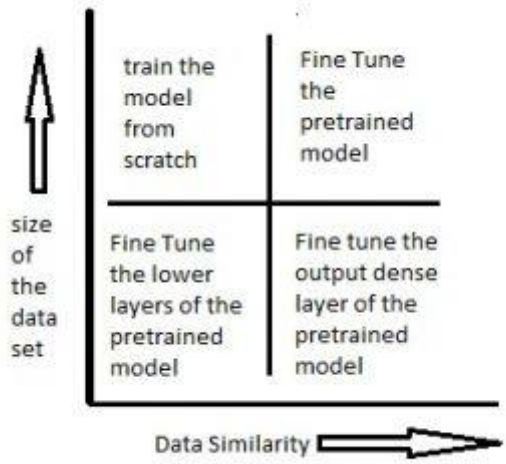
تُظهر هذه الشبكات المُدرَّبة مسبقًا قدرة قوية على التعميم على الصور خارج مجموعة بيانات ImageNet عبر نقل التعلم. نقوم بإجراء تعديلات على النموذج الموجود مسبقًا عن طريق ضبط النموذج. نظرًا لأننا نفترض أن الشبكة المدربة مسبقًا قد تم تدريبها جيدًا، فلن نرغبفي تعديل الأوزانفي وقت مبكر جدًا وبالكثير. أثناء التعديل، نستخدم بشكل عام معدل تعلم أصغر من المعدل المستخدم لتدريب النموذجفي البداية.

طرق ضبط النموذج

1. **استخراج الميزات Feature extraction**: يمكننا استخدام نموذج مدرب مسبقًا كآلية لاستخراج الميزات. ما يمكننا القيام به هو أنه يمكننا إزالة طبقة الإخراج (الطبقة التي تعطي احتمالات التواجدفي كل فئة من الفئات 1000) ثم استخدام الشبكة بالكامل كمستخرج ميزة ثابتة لمجموعة البيانات الجديدة.
2. **استخدام بُنية النموذج المدرب مسبقًا**: ما يمكننا فعله هو استخدام بُنية النموذج بينما نقوم بتهيئة جميع الأوزان بشكل عشوائي وتدريب النموذج وفقًا لمجموعة البيانات الخاصة بنا مرة أخرى.

3. **تدريب بعض الطبقات بينما نقوم بتجميد البعض الآخر**: هناك طريقة أخرى لاستخدام نموذج تم تدريبه مسبقًا وهي التدريب جزئيًا. ما يمكننا القيام به هو الاحتفاظ بأوزان الطبقات الأولية للنموذج مجمدة بينما نقوم بإعادة تدريب الطبقات العليا فقط. يمكننا تجربة واختبار عدد الطبقات التي سيتم تجميدها وعدد الطبقات التي سيتم تدريبها.

يجب أن يساعدك الرسم البياني أدناهفي اتخاذ قرار بشأن كيفية المضي قدمًافي استخدام النموذج المدرب مسبقًافي حالتك:



السيناريو 1: حجم مجموعة البيانات صغير بينما تشابه البيانات مرتفع جدًا ـ في هذه الحالة، نظرًا لأن تشابه البيانات مرتفع جدًا، لا نحتاج إلى إعادة تدريب النموذج. كل ما نحتاج إلى القيام به هو تخصيص وتعديل طبقات الإخراج وفقًا لبيان المشكلة. نحن نستخدم النموذج الجاهز كمستخرج للميزات. لنفترض أننا قررنا استخدام نماذج مدربة على Imagenet لتحديد ما إذا كانت مجموعة الصور الجديدة بها قطط أم كلاب. هنا الصور التي نحتاج إلى تحديدها ستكون مشابهة لـ imagenet، لكننا نحتاج فقط إلى فئتين كمخرجات ـ قطط أو كلاب.في هذه الحالة، كل ما نقوم به هو تعديل الطبقات الكثيفة وطبقة softmax النهائية لإخراج فئتين بدلاً من 1000.

السيناريو 2: حجم البيانات صغير وكذلك تشابه البيانات منخفض جدًا ـ في هذه الحالة يمكننا تجميد الطبقات الأولية (دعنا نقول k) للنموذج الذي تم اختباره مسبقًا وتدريب الطبقات المتبقية (n-k) مرة أخرى. سيتم بعد ذلك تخصيص الطبقات العليا لمجموعة البيانات الجديدة. نظرًا لأن مجموعة البيانات الجديدة ذات تشابه منخفض، فمن المهم إعادة تدريب الطبقات العليا وتخصيصها وفقًا لمجموعة البيانات الجديدة. يتم تعويض الحجم الصغير لمجموعة

البيانات من خلال حقيقة أن الطبقات الأولية يتم الاحتفاظ بها مسبقًا (والتي تم تدريبها على مجموعة بيانات كبيرة مسبقًا) ويتم تجميد أوزان تلك الطبقات.

السيناريو 3: حجم مجموعة البيانات كبير ولكن تشابه البيانات منخفض جدًا ـ في هذه الحالة، نظرًا لأن لدينا مجموعة بيانات كبيرة، سيكون تدريب الشبكة العصبية لدينا فعالاً. ومع ذلك، نظرًا لأن البيانات التي لدينا مختلفة جدًا مقارنة بالبيانات المستخدمة لتدريب نماذجنا التي تم اختبارها مسبقًا. لن تكون التنبؤات التي تم إجراؤها باستخدام النماذج التي تم اختبارها مسبقًا فعالة. وبالتالي، من الأفضل تدريب الشبكة العصبية من البداية وفقًا لبياناتك.

السيناريو 4: حجم البيانات كبير وكذلك يوجد تشابه كبيرفي البيانات ـ هذا هو الوضع المثالي. في هذه الحالة، يجب أن يكون النموذج الذي تم اختباره مسبقًا هو الأكثر فعالية. أفضل طريقة لاستخدام النموذج هي الاحتفاظ ببُنية النموذج والأوزان الأولية للنموذج. ثم يمكننا إعادة تدريب هذا النموذج باستخدام الأوزان كما تم تهيئتهافي النموذج المدرب مسبقًا.

استخدام النماذج المدربة مسبقًا لتحديد الأرقام المكتوبة بخط اليد

دعونا نحاول الآن استخدام نموذج تم اختباره مسبقًا لحل مشكلة بسيطة. هناك العديد من البُنى التي تم تدريبها على مجموعة بيانات imageNet. يمكنك الاطلاع على العديد من البنى هنا. لقد استخدمت vgg16 كنموذج معماري تم اختباره مسبقًا وحاولت تحديد الأرقام المكتوبة بخط اليد handwritten digits باستخدامه. دعونا نريفي أي من السيناريوهات المذكورة أعلاه قد تقع هذه المشكلة. لدينا حوالي 60.000 صورة تدريبية للأرقام المكتوبة بخط اليد. مجموعة البيانات هذه صغيرة بالتأكيد. لذا فإن الموقف إما أن يقعفي السيناريو 1 أو السيناريو 2. سنحاول حل المشكلة باستخدام هذين السيناريوهين. يمكن تنزيل مجموعة البيانات من هنا.

1. **أعادة تدريب طبقات الإخراج الكثيفة فقط** : هنا نستخدم vgg16 كمستخرج للميزات. ثم نستخدم هذه الميزات ونرسلها إلى طبقات كثيفة يتم تدريبها وفقًا لمجموعة البيانات الخاصة بنا. يتم أيضًا استبدال طبقة الإخراج بطبقة softmax الجديدة الخاصة بنا ذات الصلة بمشكلتنا. طبقة الإخراجفي vgg16 عبارة عن تنشيط softmax مع 1000 فئة. نقوم بإزالة هذه الطبقة واستبدالها بطبقة softmax من 10 فئات. نقوم فقط بتدريب أوزان هذه الطبقات ونحاول تحديد الأرقام.

```
# importing required libraries
from keras.models import Sequential
from scipy.misc import imread
get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras.layers import Dense
import pandas as pd
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
```

```
from keras.applications.vgg16 import preprocess input
import numpy as np
from keras.applications.vgg16 import decode_predictions
train=pd.read_csv("R/Data/Train/train.csv")
test=pd.read_csv("R/Data/test.csv")
train_path="R/Data/Train/Images/train/"
test path="R/Data/Train/Images/test/"
from scipy.misc import imresize
# preparing the train dataset
train_img=[]
for i in range(len(train)):
    temp_img=image.load_img(train_path+train['filename'][i],target_siz
e=(224,224))
    temp_img=image.img_to_array(temp_img)
    train_img.append(temp_img)
#converting train images to array and applying mean subtraction
processing
train_img=np.array(train_img)
train_img=preprocess_input(train_img)
# applying the same procedure with the test dataset
test_img=[]
for i in range(len(test)):
    temp_img=image.load_img(test_path+test['filename'][i],target_size=
(224,224))
    temp_img=image.img_to_array(temp_img)
    test_img.append(temp_img)
test_img=np.array(test_img)
test_img=preprocess_input(test_img)
# loading VGG16 model weights
model = VGG16(weights='imagenet', include_top=False)
# Extracting features from the train dataset using the VGG16 pre-
trained model
features_train=model.predict(train_img)
# Extracting features from the train dataset using the VGG16 pre-
trained model
features_test=model.predict(test_img)
# flattening the layers to conform to MLP input
train_x=features_train.reshape(49000,25088)
# converting target variable to array
train_y=np.asarray(train['label'])
# performing one-hot encoding for the target variable
train_y=pd.get_dummies(train_y)
train_y=np.array(train_y)
# creating training and validation set
from sklearn.model_selection import train_test_split
X_train, X_valid, Y_train,
Y_valid=train_test_split(train_x,train_y,test_size=0.3,
random_state=42)

# creating a mlp model
from keras.layers import Dense, Activation
model=Sequential()
model.add(Dense(1000, input_dim=25088,
activation='relu',kernel_initializer='uniform'))
keras.layers.core.Dropout(0.3, noise_shape=None, seed=None)
model.add(Dense(500,input_dim=1000,activation='sigmoid'))
keras.layers.core.Dropout(0.4, noise_shape=None, seed=None)
model.add(Dense(150,input_dim=500,activation='sigmoid'))
keras.layers.core.Dropout(0.2, noise_shape=None, seed=None)
model.add(Dense(units=10))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer="adam",
metrics=['accuracy'])
```

```
# fitting the model
model.fit(X_train, Y_train, epochs=20,
batch_size=128,validation_data=(X_valid,Y_valid))
```

2. **تجميد أوزان الطبقات القليلة الأولى**: ما نقوم به هنا هو تجميد أوزان الطبقات الثمانية الأولى من شبكة vgg16، بينما نقوم بإعادة تدريب الطبقات اللاحقة. هذا لأن الطبقات القليلة الأولى تلتقط ميزات عامة مثل المنحنيات والحواف التي ترتبط أيضًا بمشكلتنا الجديدة. نريد الحفاظ على هذه الأوزان كما هي وسنعمل على جعل الشبكة تركز على تعلم الميزات الخاصة بمجموعة البياناتفي الطبقات اللاحقة.

كود لتجميد أوزان الطبقات الأولى.

```
from keras.models import Sequential
from scipy.misc import imread
get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras.layers import Dense
import pandas as pd
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import numpy as np
from keras.applications.vgg16 import decode_predictions
from keras.utils.np_utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.optimizers import SGD
from keras.layers import Input, Dense, Convolution2D, MaxPooling2D,
AveragePooling2D, ZeroPadding2D, Dropout, Flatten, merge, Reshape,
Activation
from sklearn.metrics import log_loss

train=pd.read_csv("R/Data/Train/train.csv")
test=pd.read_csv("R/Data/test.csv")
train_path="R/Data/Train/Images/train/"
test_path="R/Data/Train/Images/test/"
from scipy.misc import imresize
train_img=[]
for i in range(len(train)):
    temp_img=image.load_img(train_path+train['filename'][i],target_siz
e=(224,224))
    temp_img=image.img_to_array(temp_img)
    train_img.append(temp_img)
train_img=np.array(train_img)
train_img=preprocess_input(train_img)
test_img=[]
for i in range(len(test)):
temp_img=image.load_img(test_path+test['filename'][i],target_size=(224
,224))
    temp_img=image.img_to_array(temp_img)
    test_img.append(temp_img)
test_img=np.array(test_img)
test_img=preprocess_input(test_img)
from keras.models import Model
def vgg16_model(img_rows, img_cols, channel=1, num_classes=None):
    model = VGG16(weights='imagenet', include_top=True)
    model.layers.pop()
    model.outputs = [model.layers[-1].output]
```

```
    model.layers[-1].outbound_nodes = []
          x=Dense(num_classes, activation='softmax')(model.output)
    model=Model(model.input,x)
#To set the first 8 layers to non-trainable (weights will not be
updated)
          for layer in model.layers[:8]:
       layer.trainable = False
# Learning rate is changed to 0.001
    sgd = SGD(lr=1e-3, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(optimizer=sgd, loss='categorical_crossentropy',
metrics=['accuracy'])
    return model
train_y=np.asarray(train['label'])
le = LabelEncoder()
train_y = le.fit_transform(train_y)
train_y=to_categorical(train_y)
train_y=np.array(train_y)
from sklearn.model_selection import train_test_split
X_train, X_valid, Y_train,
Y_valid=train_test_split(train_img,train_y,test_size=0.2,
random_state=42)
# Example to fine-tune on 3000 samples from Cifar10
img_rows, img_cols = 224, 224 # Resolution of inputs
channel = 3
num_classes = 10
batch_size = 16
nb_epoch = 10
# Load our model
model = vgg16_model(img_rows, img_cols, channel, num_classes)
model.summary()
# Start Fine-tuning
model.fit(X_train,
Y_train,batch_size=batch_size,epochs=nb_epoch,shuffle=True,verbose=1,v
alidation_data=(X_valid, Y_valid))
# Make predictions
predictions_valid = model.predict(X_valid, batch_size=batch_size,
verbose=1)
# Cross-entropy loss score
score = log_loss(Y_valid, predictions_valid)
```

الملخص

آمل أن تتمكن الآن من تطبيق نماذج مدربة مسبقًا على بيانات مشكلتك. تأكد من أن النموذج المدرب مسبقًا الذي حددته قد تم تدريبه على مجموعة بيانات مماثلة لتلك التي ترغبفي استخدامه عليها. هناك العديد من البنى التي جربها الأشخاص على أنواع مختلفة من مجموعات البيانات وأنا أشجعك بشدة على متابعة هذه البُنى وتطبيقها على بيانات المشكلة الخاصة بك. لا تترددفي مناقشة شكوكك واهتماماتكفي قسم التعليقات.

المصدر:

https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/

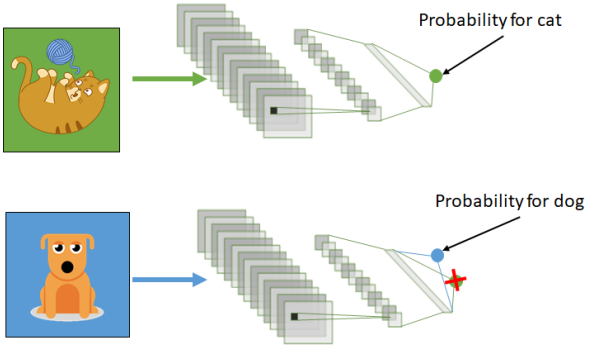
2) فهم نقل التعلم للتعلم العميق Understanding Transfer Learning for Deep Learning

تُعرف إعادة استخدام نموذج تم تعلمه مسبقًافي مشكلة جديدة باسم نقل التعلم transfer learning. إنه شائع بشكل خاصفي التعلم العميقفي الوقت الحالي لأنه يمكنه تدريب الشبكات العصبية العميقة بكمية صغيرة من البيانات. هذا مهم بشكل خاصفي مجال علم البيانات، حيث أن معظم مواقف العالم الحقيقي لا تتطلب الملايين من نقاط البيانات المصنفة لتدريب النماذج المعقدة.

ما هو نقل التعلم وكيف يعمل

تُعرف إعادة استخدام نموذج مدرب مسبقًافي مشكلة جديدة باسم نقل التعلمفي التعلم الآلي. تستخدم الآلة المعرفة المكتسبة من مهمة سابقة لزيادة التنبؤ بمهمة جديدةفي نقل التعلم. يمكنك، على سبيل المثال، استخدام المعلومات المكتسبة أثناء التدريب لتمييز المشروبات عند تدريب المصنف للتنبؤ بما إذا كانت الصورة تحتوي على مطبخ.

يتم نقل المعرفة بنموذج التعلم الآلي الذي تم تدريبه بالفعل إلى مشكلة مختلفة ولكنها مرتبطة ارتباطًا وثيقًا خلال عملية نقل التعلم. على سبيل المثال، إذا قمت بتدريب المصنف البسيط للتنبؤ بما إذا كانت الصورة تحتوي على حقيبة ظهر، فيمكنك استخدام المعرفة التدريبية للنموذج لتحديد كائنات أخرى مثل النظارات الشمسية.

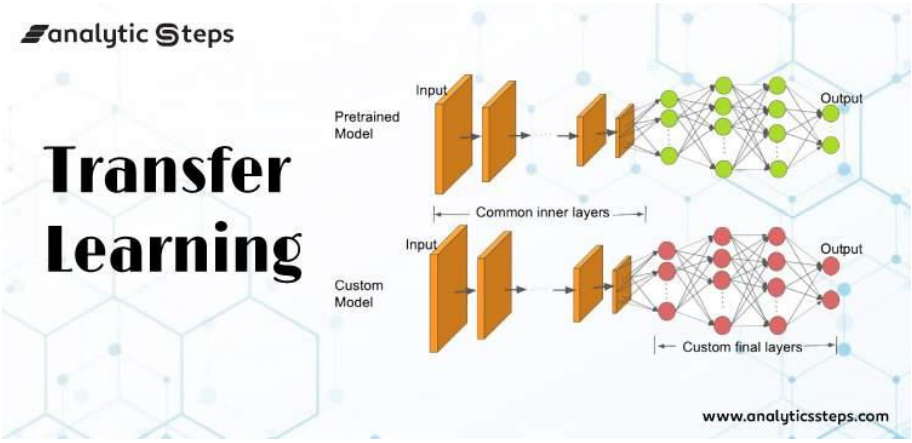


من خلال نقل التعلم، نحاول بشكل أساسي استخدام ما تعلمناهفي مهمة واحدة لفهم المفاهيم بشكل أفضلفي مهمة أخرى. يتم تحويل الأوزان تلقائيًا إلى شبكة تؤدي "المهمة A" من شبكة تؤدي "المهمة B" الجديدة.

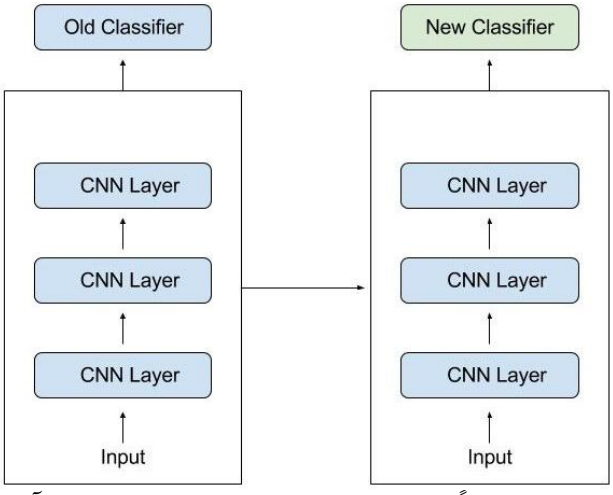
نظرًا للكم الهائل من طاقة وحدة المعالجة المركزية المطلوبة، يتم تطبيق تعلم النقل عادةًفي الرؤية الحاسوبية ومهام معالجة اللغة الطبيعية مثل تحليل المشاعر sentiment analysis.

كيف يعمل نقل التعلم

في الرؤية الحاسوبية، تهدف الشبكات العصبية عادةً إلى اكتشاف الحواففي الطبقة الأولى والنماذجفي الطبقة الوسطى والميزات الخاصة بالمهمةفي الطبقات الأخيرة. يتم استخدام الطبقات الأولى والمركزيةفي نقل التعلم، ويتم إعادة تدريب الطبقات الأخيرة فقط. يستفيد من البيانات المصنفة من المهمة التي تم التدريب عليها.



لنعد إلى مثال النموذج الذي تم تصميمه لتحديد حقيبة ظهرفي صورة ما وسيتم استخدامه الآن لاكتشاف النظارات الشمسية. نظرًا لأن النموذج قد تدرب على التعرف على الكائناتفي المستويات السابقة، فسنقوم ببساطة بإعادة تدريب الطبقات اللاحقة لفهم ما يميز النظارات الشمسية عن الكائنات الأخرى.



نظرًا لأن النموذج قد تم تدريبه مسبقًا بالفعل، يمكن إنشاء نموذج جيد للتعلم الآلي مع القليل من بيانات التدريب إلى حد ما باستخدام نقل التعلم. هذا مفيد بشكل خاصفي معالجة اللغة الطبيعية، حيث تتطلب مجموعات البيانات ذات العلامات الضخمة الكثير من المعرفة

المتخصصة. بالإضافة إلى ذلك، يتم تقليل وقت التدريب لأن بناء شبكة عصبية عميقة من بداية مهمة معقدة قد يستغرق أيامًا أو حتى أسابيع.

متى تستخدم نقل التعلم

عندما لا تتوفر لدينا بيانات مشروحة كافية لتدريب نموذجنا عليها. عندما يكون هناك نموذج مدرب مسبقًا تم تدريبه على بيانات ومهام مماثلة. إذا استخدمت TensorFlow لتدريب النموذج الأصلي، فيمكنك ببساطة استعادته وإعادة تدريب بعض الطبقات لعملك. من ناحية أخرى، لا يعمل نقل التعلم إلا إذا كانت الميزات التي تم تعلمهافي المهمة الأولى عامة، مما يعني أنه يمكن تطبيقها على نشاط آخر. علاوة على ذلك، يجب أن يكون إدخال النموذج بنفس الحجم الذي كان عليه عندما تم تدريبه لأول مرة. إذا لم يكن لديك، أضف خطوة لتغيير حجم إدخالك إلى الحجم المطلوب.

1. تدريب نموذج لإعادة استخدامه

ضعفي اعتبارك الموقف الذي ترغبفي معالجة المهمة Aولكن تفتقر إلى البيانات اللازمة لتدريب شبكة عصبية عميقة. يعد العثور على مهمة B ذات صلة بالكثير من البيانات إحدى الطرق للتغلب على هذا الأمر.

استخدم الشبكة العصبية العميقة للتدريب على المهمة "B" ثم استخدم النموذج لحل المهمة "A". ستقرر المشكلة التي تسعى إلى حلها ما إذا كنت بحاجة إلى استخدام النموذج بأكمله أو بضع طبقات فقط.

إذا كانت المدخلاتفي كلتا الوظيفتين هي نفسها، فيمكنك إعادة تطبيق النموذج وإجراء تنبؤات لمدخلاتك الجديدة. من ناحية أخرى، يعد تغيير وإعادة تدريب الطبقات المميزة الخاصة بالمهمة وطبقة المخرجات طريقة للتحقيق.

2. استخدام نموذج تم تدريبه مسبقًا

الخيار الثاني هو استخدام نموذج تم تدريبه بالفعل. هناك عدد من هذه النماذج، لذا قم بإجراء بعض الأبحاث مسبقًا. يتم تحديد عدد الطبقات التي سيتم إعادة استخدامها وإعادة التدريب من خلال المهمة.

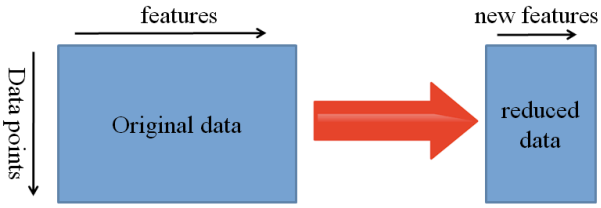
يتكون Keras من تسعة نماذج مدربة مسبقًا تستخدمفي نقل التعلم transfer learning والتنبؤ prediction والضبط الدقيق fine-tuning. يمكن العثور هنا على هذه النماذج، بالإضافة إلى بعض الدروس السريعة حول كيفية الاستفادة منها. كما تتيح العديد من المؤسسات البحثية إمكانية الوصول إلى النماذج المدربة.

التطبيق الأكثر شيوعًا لهذا النوع من التعلم هو التعلم العميق deep learning.

3. استخراج الميزات

خيار آخر هو استخدام التعلم العميق لتحديد التمثيل الأمثل لمشكلتك، والذي يتضمن تحديد الميزات الرئيسية. تُعرف هذه الطريقة باسم التعلم التمثيلي representation learning، ويمكن أن تؤدي غالبًا إلى نتائج أفضل بكثير من التمثيلات المصممة يدويًا.

يتم إنشاء الميزاتفي التعلم الآلي بشكل أساسي يدويًا بواسطة الباحثين والمتخصصينفي المجال. لحسن الحظ، يمكن للتعلم العميق استخراج الميزات تلقائيًا. بالطبع، هذا لا يقلل من أهمية هندسة الميزات ومعرفة المجال؛ لا يزال يتعين عليك اختيار الميزات التي تريد تضمينهافي شبكتك.



من ناحية أخرى، تتمتع الشبكات العصبية بالقدرة على معرفة الميزات المهمة وغير المهمة. حتى بالنسبة للمهام المعقدة التي قد تتطلب الكثير من الجهد البشري، يمكن لخوارزمية التعلم التمثيلي أن تجد مزيجًا لائقًا من الخصائصفي فترة زمنية قصيرة.

يمكن بعد ذلك تطبيق التمثيل المكتسب على مجموعة متنوعة من التحديات الأخرى. ما عليك سوى استخدام الطبقات الأولية للعثور على تمثيل الميزة المناسب، ولكن تجنب استخدام مخرجات الشبكة لأنها شديدة التحديد بالمهمة. بدلاً من ذلك، أرسل البيانات إلى شبكتك وأخرجها من خلال إحدى الطبقات الوسيطة.

يمكن بعد ذلك فهم البيانات الأولية على أنها تمثيل لهذه الطبقة.

تُستخدم هذه الطريقة بشكل شائعفي الرؤية الحاسوبية حيث يمكنها تقليص مجموعة البيانات الخاصة بك وتقليل وقت الحساب وجعلها أكثر ملاءمة للخوارزميات الكلاسيكية.

النماذج التي تم تدريبها مسبقًا

يتوفر عدد من نماذج التعلم الآلي الشائعة المدربة مسبقًا. نموذج Inception-v3، الذي تم تطويره لـ ImageNet "تحدي التعرف البصري الكبير Large Visual Recognition Challenge" ، هو أحد هذه النماذج. "كان على المشاركينفي هذا التحدي تصنيف الصور إلى 1000 فئة فرعية مثل "الحمار الوحشي" و "الدلماسي" و "غسالة الصحون".

تنفيذ التعليمات البرمجية لنقل التعلم باستخدام Python

(مجموعة البيانات هي Chest-CT Scan من Kaggle)

استيراد مكتبات

```
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras import Model
from tensorflow.keras.layers import Conv2D, Dense, MaxPooling2D,
Dropout, Flatten,GlobalAveragePooling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 import
preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import
ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
```

تحميل البيانات عبر Kaggle API

```
from google.colab import files
files.upload()

Saving kaggle.json to kaggle.json

!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d mohamedhanyyy/chest-ctscan-images
#downloading data from kaggle API of Dataset
from zipfile import ZipFile
file_name = "chest-ctscan-images.zip"

with ZipFile(file_name,'r') as zip:
  zip.extractall()
  print('Done')
```

تصميم نموذج CNN الخاص بنا بمساعدة نموذج التدريب المسبق

```
InceptionV3_model =
tf.keras.applications.InceptionV3(weights='imagenet',
include_top=False, input_shape=(224, 224, 3))

from tensorflow.keras import Model
from tensorflow.keras.layers import Conv2D, Dense, MaxPooling2D,
Dropout, Flatten,GlobalAveragePooling2D
from tensorflow.keras.models import Sequential

# The last 15 layers fine tune
for layer in InceptionV3_model.layers[:-15]:
    layer.trainable = False

x = InceptionV3_model.output
x = GlobalAveragePooling2D()(x)
```

```
x = Flatten()(x)
x = Dense(units=512, activation='relu')(x)
x = Dropout(0.3)(x)
x = Dense(units=512, activation='relu')(x)
x = Dropout(0.3)(x)
output  = Dense(units=4, activation='softmax')(x)
model = Model(InceptionV3_model.input, output)


model.summary()
```

تكبير الصورة (لمنع مشكلة الضبط الزائد overfitting)

```
# Use the Image Data Generator to import the images from the dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
#no flip and zoom for test datase


# Make sure you provide the same target size as initialied for the
image size
training_set =
train_datagen.flow_from_directory('/content/Data/train',
                                                  target_size = (224,
224),
                                                  batch size = 32,
                                                  class_mode =
'categorical')
```

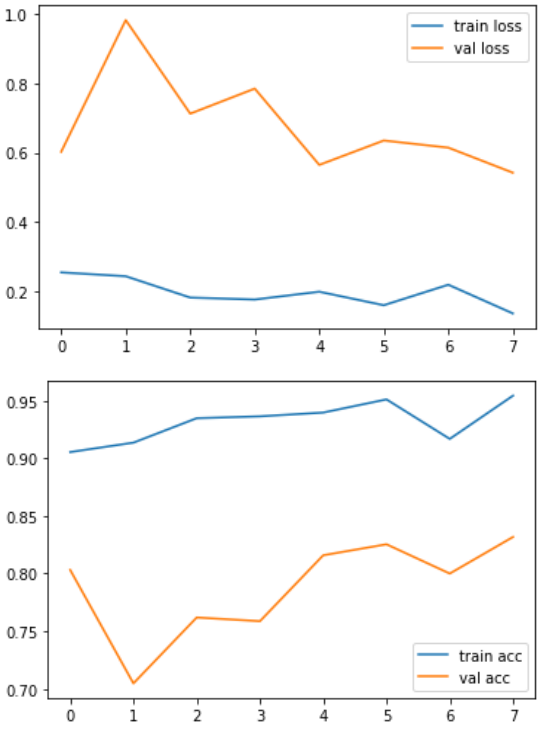
تدريب النموذج

```
# fit the model
# Run the cell. It will take some time to execute
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=8,
  steps per epoch=len(training set),
  validation_steps=len(test_set)
)

# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

التنبؤات

```
import numpy as np
y pred = np.argmax(y pred, axis=1)
y_pred
```

يتم تنفيذ الكود أعلاه ويتم عرض الإخراج الخاص بالتصنيف باستخدام نقل التعلم أسفل النوتبوك المضمن:



يمكنك الوصول إلى رابط Github الخاص بنوتبوك Google Colab من هنا.

المصدر:

https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/

3) نقل التعلم في Keras باستخدام نماذج الرؤية الحاسوبية Transfer Learning in Keras with Computer Vision Models

قد تستغرق نماذج الشبكة العصبية التلافيفية العميقة أيامًا أو حتى أسابيع للتدريب على مجموعات بيانات كبيرة جدًا.

تتمثل إحدى طرق اختصار هذه العمليةفي إعادة استخدام أوزان النموذج من النماذج المدربة مسبقًا pre-trained models التي تم تطويرها لمجموعات البيانات المعيارية لقياس الرؤية الحاسوبية، مثل مهام التعرف على الصور ImageNet. يمكن تنزيل النماذج الأفضل أداءً واستخدامها مباشرةً، أو دمجهافي نموذج جديد لمشكلات الرؤية الحاسوبية لديك.

في هذه المقالة، سوف تكتشف كيفية استخدام نقل التعلم transfer learning عند تطوير الشبكات العصبية التلافيفية CNN لتطبيقات الرؤية الحاسوبية.

بعد قراءة هذه المقالة، ستعرف:

- يتضمن نقل التعلم استخدام نماذج مدربة على مشكلة واحدة كنقطة بداية لمشكلة ذات صلة.
- يتسم نقل التعلم بالمرونة، مما يسمح باستخدام النماذج المدربة مسبقًا بشكل مباشر، كعملية معالجة مسبقة لاستخراج الميزات، ومتكاملةفي نماذج جديدة تمامًا.
- يوفر Keras وصولاً مناسبًا إلى العديد من النماذج عالية الأداءفي مهام التعرف على الصور على ImageNet مثل VGG و Inception و ResNet.

ينقسم هذا البرنامج التعليمي إلى خمسة أجزاء:

3. ما هو نقل التعلم؟
4. نقل التعلم للتعرف على الصور.
5. كيفية استخدام النماذج التي تم تدريبها مسبقًا.
6. نماذج لنقل التعلم.
7. أمثلة على استخدام النماذج التي تم تدريبها مسبقًا.

ما هو نقل التعلم؟

يشير نقل التعلم عمومًا إلى عملية يتم فيها استخدام نموذج مدرب على مشكلة واحدة بطريقة ما في مشكلة ثانية ذات صلة.

في التعلم العميق، يعد نقل التعلم أسلوبًا يتم من خلاله تدريب نموذج الشبكة العصبية أولاً على مشكلة مشابهة للمشكلة التي يتم حلها. ثم يتم استخدام طبقة واحدة أو أكثر من النموذج المدرب في نموذج جديد يتم تدريبه على مشكلة الاهتمام.

يُفهم هذا عادةً في سياق التعلم الخاضع للإشراف، حيث تكون المدخلات هي نفسها ولكن الهدف قد يكون ذا طبيعة مختلفة. على سبيل المثال، قد نتعرف على مجموعة واحدة من الفئات المرئية، مثل القطط والكلاب، في الإعداد الأول، ثم نتعرف على مجموعة مختلفة من الفئات المرئية، مثل النمل والدبابير، في الإعداد الثاني.

ـ صفحة 536، التعلم العميق، 2016.

يستفيد نقل التعلم من تقليل وقت التدريب لنموذج الشبكة العصبية ويمكن أن يؤدي إلى انخفاض خطأ التعميم generalization error.

يمكن استخدام الأوزان في الطبقات المعاد استخدامها كنقطة انطلاق لعملية التدريب وتكييفها استجابةً للمشكلة الجديدة. هذا الاستخدام يعامل نقل التعلم كنوع من مخطط تهيئة الوزن. قد يكون هذا مفيداً عندما تحتوي المشكلة الأولى ذات الصلة على بيانات مصنفة أكثر بكثير من مشكلة الاهتمام وقد يكون التشابه في بُنية المشكلة مفيداً في كلا السياقين.

... الهدف هو الاستفادة من البيانات من الإعداد الأول لاستخراج المعلومات التي قد تكون مفيدة عند التعلم أو حتى عند إجراء تنبؤات مباشرة في الإعداد الثاني.

ـ صفحة 538، التعلم العميق، 2016.

نقل التعلم للتعرف على الصور

تم تطوير مجموعة من النماذج عالية الأداء لتصنيف الصور وعرضها في تحدي التعرف المرئي على نطاق واسع ImageNet السنوي، أو ILSVRC.

هذا التحدي، الذي يشار إليه غالبًا باسم ImageNet، نظرًا لمصدر الصورة المستخدمة في المسابقة، أدى إلى عدد من الابتكارات في البنُى وتدريب الشبكات العصبية التلافيفية. بالإضافة إلى ذلك، تم إصدار العديد من النماذج المستخدمة في المسابقات بموجب ترخيص مسموح به.

يمكن استخدام هذه النماذج كأساس لنقل التعلم في تطبيقات الرؤية الحاسوبية.

هذا مرغوب فيه لعدد من الأسباب، ليس أقلها:

- **الميزات المستفادة المفيدة Useful Learned Features:** لقد تعلمت النماذج كيفية اكتشاف الميزات العامة من الصور، نظرًا لتدريبها على أكثر من 1،000،000 صورة لـ 1،000 فئة.
- **أداء متطور State-of-the-Art Performance:** حققت النماذج أداءً متطورًا وظلت فعالةفي مهمة التعرف على الصور المحددة التي تم تطويرها من أجلها.
- **يمكن الوصول إليها بسهولة Easily Accessible:** يتم توفير أوزان النموذج كملفات قابلة للتنزيل مجانًا وتوفر العديد من المكتبات واجهات برمجة تطبيقات API ملائمة لتنزيل النماذج واستخدامها مباشرةً.

يمكن تنزيل أوزان النموذج واستخدامهافي نفس بُنية النموذج باستخدام مجموعة من مكتبات التعلم العميق المختلفة، بمافي ذلك Keras.

كيفية استخدام النماذج التي تم تدريبها مسبقًا

يقتصر استخدام نموذج مدرب مسبقًا على إبداعك فقط.

على سبيل المثال، يمكن تنزيل نموذج واستخدامه كما هو، مثل تضمينهفي تطبيق ما واستخدامه لتصنيف الصور الفوتوغرافية الجديدة.

بالتناوب، يمكن تنزيل النماذج واستخدامها كنماذج لاستخراج الميزات. هنا، يتم استخدام إخراج النموذج من طبقة قبل طبقة الإخراج للنموذج كمدخل إلى نموذج مصنف جديد.

تذكر أن الطبقات التلافيفية الأقرب إلى طبقة الإدخال للنموذج تتعلم ميزات منخفضة المستوى مثل الخطوط، وأن الطبقات الموجودةفي منتصف الطبقة تتعلم ميزات مجردة معقدة تجمع بين ميزات المستوى الأدنى المستخرجة من الإدخال، والطبقات الأقرب إلى الإخراج تفسير الميزات المستخرجةفي سياق مهمة التصنيف.

مسلحًا بهذا الفهم، يمكن اختيار مستوى من التفاصيل لاستخراج الميزة من نموذج حالي تم تدريبه مسبقًا. على سبيل المثال، إذا كانت المهمة الجديدة مختلفة تمامًا عن تصنيف الكائنات في الصور (على سبيل المثال مختلفة عن ImageNet)، فربما يكون إخراج النموذج المدرب مسبقًا بعد الطبقات القليلة مناسبًا. إذا كانت المهمة الجديدة مشابهة تمامًا لمهمة تصنيف الكائناتفي الصور، فربما يمكن استخدام الإخراج من طبقات أعمق بكثيرفي النموذج، أو حتى يمكن استخدام إخراج الطبقة المتصلة بالكامل قبل طبقة الإخراج.

يمكن استخدام النموذج المدرب مسبقًا كبرنامج منفصل لاستخراج الميزات، وفي هذه الحالة يمكن معالجة المدخلات مسبقًا بواسطة النموذج أو جزء من النموذج لمخرج معين (مثل متجه الأرقام) لكل صورة إدخال، والتي يمكن ثم استخدم كمدخل عند تدريب نموذج جديد.

بالتناوب، يمكن دمج النموذج المدرب مسبقًا أو الجزء المطلوب من النموذج مباشرةًفي نموذج شبكة عصبية جديد.في هذا الاستخدام، يمكن تجميد أوزان النماذج المدربة مسبقًا بحيث لا يتم تحديثها مع تدريب النموذج الجديد. بالتناوب، قد يتم تحديث الأوزان أثناء تدريب النموذج الجديد، ربما بمعدل تعلم أقل، مما يسمح للنموذج المدرب مسبقًا بالعمل كمخطط تهيئة الوزن عند تدريب النموذج الجديد.

يمكننا تلخيص بعض أنماط الاستخدام هذه على النحو التالي:

- **المصنف Classifier**: يستخدم النموذج المدربين مسبقًا مباشرةً لتصنيف الصور الجديدة.
- **مستخرج الميزات المستقلة Standalone Feature Extractor**: يتم استخدام النموذج المدرب مسبقًا، أو جزء من النموذج، لمعالجة الصور مسبقًا واستخراج الميزات ذات الصلة.
- **مستخرج الميزات المتكاملة Integrated Feature Extractor**: تم دمج النموذج المدرب مسبقًا، أو جزء من النموذج،في نموذج جديد، ولكن يتم تجميد طبقات النموذج المدرب مسبقًا أثناء التدريب.
- **تهيئة الوزن Weight Initialization**: تم دمج النموذج المدرَّب مسبقًا، أو جزء من النموذج،في نموذج جديد، ويتم تدريب طبقات النموذج المدرَّب مسبقًا بالتنسيق مع النموذج الجديد.

يمكن أن يكون كل نهج فعال ويوفر وقتًا كبيرًافي تطوير وتدريب نموذج شبكة عصبية تلافيفية عميقة.

قد لا يكون من الواضح فيما يتعلق باستخدام النموذج المدرب مسبقًا الذي قد يؤدي إلى أفضل النتائجفي مهمة الرؤية الحاسوبية الجديدة، لذلك قد تكون هناك حاجة إلى بعض التجارب.

نماذج لنقل التعلم

ربما يكون هناك عشرات أو أكثر من النماذج عالية الأداء للتعرف على الصور التي يمكن تنزيلها واستخدامها كأساس للتعرف على الصور ومهام الرؤية الحاسوبية ذات الصلة.

ربما تكون ثلاثة من النماذج الأكثر شيوعًا كما يلي:

- VGG (مثل VGG16 أو VGG19).
- GoogLeNet (مثل InceptionV3).
- الشبكة المتبقية Residual Network (مثل ResNet50).

تُستخدم هذه النماذج على نطاق واسع في نقل التعلم بسبب أدائها، ولكن أيضًا لأنها كانت أمثلة قدمت ابتكارات معمارية محددة، وهي الهياكل المتسقة والمتكررة (VGG)، ووحدات الاستهلال inception modules (GoogLeNet)، والوحدات المتبقية residual modules (ResNet).

يوفر Keras الوصول إلى عدد من النماذج عالية الأداء المدربة مسبقًا والتي تم تطويرها لمهام التعرف على الصور.

وهي متوفرة عبر واجهة برمجة التطبيقات Applications API ، وتتضمن دوال لتحميل نموذج بأوزان مُدربة مسبقًا أو بدونها، وإعداد البيانات بطريقة قد يتوقعها نموذج معين (مثل قياس الحجم وقيم البكسل).

في المرة الأولى التي يتم فيها تحميل نموذج مدرب مسبقًا، ستقوم Keras بتنزيل أوزان النموذج المطلوبة، والتي قد تستغرق بعض الوقت نظرًا لسرعة اتصالك بالإنترنت. يتم تخزين الأوزانفي دليل keras / Models ضمن دليلك وسيتم تحميلها من هذا الموقعفي المرة التالية التي يتم استخدامها فيها.

عند تحميل نموذج معين، يمكن ضبط وسيطة "include_top" على False، وفي هذه الحالة لا يتم تحميل طبقات الإخراج المتصلة بالكامل للنموذج المستخدم لعمل التنبؤات، مما يسمح بإضافة طبقة إخراج جديدة وتدريبها. على سبيل المثال:

```
...
# load model without output layer
model = VGG16(include_top=False)
```

بالإضافة إلى ذلك، عندما تكون الوسيطة "include_top" لى False، يجب تحديد وسيطة "input_tensor"، مما يسمح بتغيير المدخلات المتوقعة ذات الحجم الثابت للنموذج. على سبيل المثال:

```
...
# load model and specify a new input shape for images
new_input = Input(shape=(640, 480, 3))
model = VGG16(include_top=False, input_tensor=new_input)
```

نموذج بدون قمة سينتج عمليات التنشيط من آخر طبقة تلافيفية convolutional أو تجميع pooling مباشرة. تتمثل إحدى طرق تلخيص عمليات التنشيط هذه للاستخدامفي المصنف أو كتمثيل متجه للمدخلاتفي إضافة طبقة تجميع عالمية global pooling layer، مثل الحد الأقصى للتجميع العالمي max global pooling أو متوسط التجميع العالمي average global pooling. والنتيجة هي متجه يمكن استخدامه كواصف ميزة لمدخل. يوفر Keras هذه الإمكانية مباشرة عبر وسيطة "pooling" التي يمكن ضبطها على "avg " أو "max ". على سبيل المثال:

```
...
# load model and specify a new input shape for images and avg pooling
output
new_input = Input(shape=(640, 480, 3))
model = VGG16(include_top=False, input_tensor=new_input,
pooling='avg')
```

يمكن تحضير الصور لنموذج معين باستخدام الدالة preprocess_input () ؛ على سبيل المثال، يتم إجراء قياس البكسل بطريقة تم إجراؤها على الصورفي مجموعة بيانات التدريب عندما تم تطوير النموذج. على سبيل المثال:

```
...
# prepare an image
from keras.applications.vgg16 import preprocess_input
images = ...
prepared_images = preprocess_input(images)
```

أخيرًا، قد ترغبفي استخدام بُنية نموذجيةفي مجموعة البيانات الخاصة بك، ولكن لا تستخدم الأوزان المدربة مسبقًا، وبدلاً من ذلك، قم بتهيئة النموذج باستخدام أوزان عشوائية وتدريب النموذج من البداية.

يمكن تحقيق ذلك عن طريق تعيين وسيطة "weights" على بلا بدلاً من "imagenet" الافتراضي. بالإضافة إلى ذلك، يمكن تعيين وسيطة "classes" لتحديد عدد الفئاتفي مجموعة البيانات الخاصة بك، والتي سيتم تهيئتها بعد ذلك لكفي طبقة الإخراج من النموذج. على سبيل المثال:

```
...
#define a new model with random weights and 10 classes
new_input = Input(shape=(640, 480, 3))
model = VGG16(weights=None, input_tensor=new_input, classes=10)
```

الآن بعد أن أصبحنا على دراية بواجهة برمجة التطبيقات API، دعنا نلقي نظرة على تحميل ثلاثة نماذج باستخدام واجهة برمجة تطبيقات Keras .

تحميل النموذج VGG16 المدرب مسبقًا

تم تطوير نموذج VGG16 بواسطة مجموعة الرسومات المرئية (VGG)في أكسفورد وتم وصفه في ورقة عام 2014 بعنوان "شبكات تلافيفية عميقة جدًا للتعرف على الصور على نطاق واسع."

بشكل افتراضي، يتوقع النموذج إعادة قياس صور إدخال الألوان إلى حجم 224 × 224 مربعًا.

يمكن تحميل النموذج على النحو التالي:

```
# example of loading the vgg16 model
from keras.applications.vgg16 import VGG16
# load model
model = VGG16()
# summarize the model
model.summary()
```

سيؤدي تشغيل المثال إلى تحميل نموذج VGG16 وتنزيل أوزان النموذج إذا لزم الأمر.

يمكن بعد ذلك استخدام النموذج مباشرة لتصنيف صورة فوتوغرافية إلى فئة من 1000 فئة.في هذه الحالة، يتم تلخيص بُنية النموذج للتأكد من أنه تم تحميله بشكل صحيح.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
```

تحميل نموذج التدريب المسبق InceptionV3

إن InceptionV3 هو التكرار الثالث لبنية الاستهلال inception architecture، الذي تم تطويره لأول مرة لنموذج GoogLeNet.

تم تطوير هذا النموذج من قبل باحثينفي Google وتم وصفهفي ورقة عام 2015 بعنوان "إعادة التفكيرفي البنية التأسيسية للرؤية الحاسوبية".

يتوقع النموذج أن يكون للصور الملونة شكل مربع 299 × 299.

يمكن تحميل النموذج على النحو التالي:

```
# example of loading the inception v3 model
from keras.applications.inception_v3 import InceptionV3
# load model
model = InceptionV3()
# summarize the model
model.summary()
```

سيؤدي تشغيل المثال إلى تحميل النموذج وتنزيل الأوزان إذا لزم الأمر، ثم تلخيص بُنية النموذج لتأكيد تحميله بشكل صحيح.

يتم حذف الإخراجفي هذه الحالة للإيجاز، لأنه نموذج عميق متعدد الطبقات.

تحميل نموذج ResNet50 المدرب مسبقًا

الشبكة المتبقية Residual Network، أو ResNet باختصار، هي نموذج يستخدم الوحدة النمطية المتبقية residual module التي تتضمن اتصالات مختصرة shortcut connections.

تم تطويره من قبل باحثينفي Microsoft وتم وصفهفي ورقة عام 2015 بعنوان "التعلم العميق المتبقي للتعرف على الصور".

يتوقع النموذج أن يكون للصور الملونة شكل مربع 224 × 224.

```
# example of loading the resnet50 model
from keras.applications.resnet50 import ResNet50
# load model
model = ResNet50()
# summarize the model
model.summary()
```

سيؤدي تشغيل المثال إلى تحميل النموذج وتنزيل الأوزان إذا لزم الأمر، ثم تلخيص بُنية النموذج لتأكيد تحميله بشكل صحيح.

يتم حذف الإخراجفي هذه الحالة للإيجاز، لأنه نموذج عميق.

أمثلة على استخدام النماذج التي تم تدريبها مسبقًا

الآن بعد أن أصبحنا على دراية بكيفية تحميل النماذج المدربة مسبقًافي Keras، دعنا نلقي نظرة على بعض الأمثلة حول كيفية استخدامهافي الممارسة العملية.

في هذه الأمثلة، سنعمل مع نموذج VGG16 لأنه نموذج مباشر نسبيًا للاستخدام وبُنية نموذجية بسيطة لفهمها.

نحتاج أيضًا إلى صورة للعمل بهافي هذه الأمثلة. يوجد أدناه صورة لكلب، التقطها جوستين مورغان وتم إتاحتها بموجب ترخيص مسموح به.

قم بتنزيل الصورة وضعهافي دليل العمل الحالي الخاص بك باسم الملف "dog.jpg".

- صورة كلب (dog.jpg)

نموذج تم تدريبه مسبقًا كمصنف

يمكن استخدام نموذج مدرب مسبقًا مباشرةً لتصنيف الصور الجديدة كواحدة من 1000 فئة معروفةفي مهمة تصنيف الصورفي ILSVRC.

سنستخدم نموذج VGG16 لتصنيف الصور الجديدة.

أولاً، يجب تحميل الصورة وإعادة تشكيلها إلى مربع 224 × 224، الذي يتوقعه النموذج، وقيم البكسل المقاسة بالطريقة التي يتوقعها النموذج. يعمل النموذج على مجموعة من العينات، لذلك يجب توسيع أبعاد الصورة المحملة بمقدار 1، لصورة واحدة بحجم 224 × 224 بكسل وثلاث قنوات.

```
# load an image from file
image = load_img('dog.jpg', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1],
image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
```

بعد ذلك، يمكن تحميل النموذج وإجراء تنبؤ.

هذا يعني أنه تم عمل احتمالية متوقعة للصورة التي تنتمي إلى كل فئة من 1000 فئة.في هذا المثال، نحن معنيون فقط بالفئة الأكثر احتمالًا، لذلك يمكننا فك شفرة التنبؤات واسترداد تسمية أو اسم الفئة ذات الاحتمالية الأعلى.

```
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
```

بربط كل هذا معًا، يقوم المثال الكامل أدناه بتحميل صورة جديدة ويتنبأ بالفئة الأكثر احتمالية.

```
# example of using a pre-trained model as a classifier
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode predictions
from keras.applications.vgg16 import VGG16
# load an image from file
image = load_img('dog.jpg', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1],
image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# load the model
model = VGG16()
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

تشغيل المثال يتنبأ بأكثر من مجرد كلب؛ كما أنه يتنبأ بالسلالة المحددة من "Doberman" باحتمالية 33.59٪، والتي قد تكون صحيحةفي الواقع.

```
Doberman(%33.59)
```

نموذج تم تدريبه مسبقًا باعتباره معالجًا أوليًا لمستخرج الميزات

يمكن استخدام النموذج المدرب مسبقًا كبرنامج مستقل لاستخراج الميزات features extraction من الصور الجديدة.

على وجه التحديد، قد تكون الميزات المستخرجة من الصورة عبارة عن متجه للأرقام التي سيستخدمها النموذج لوصف الميزات المحددةفي الصورة. يمكن بعد ذلك استخدام هذه الميزات كمدخلاتفي تطوير نموذج جديد.

الطبقات القليلة الأخيرة من نموذج VGG16 هي طبقات متصلة بالكامل fully connected layers قبل طبقة الإخراج. ستوفر هذه الطبقات مجموعة معقدة من الميزات لوصف صورة إدخال معينة وقد توفر مدخلات مفيدة عند تدريب نموذج جديد لتصنيف الصور أو مهمة الرؤية الحاسوبية ذات الصلة.

يمكن تحميل الصورة وتجهيزها للنموذج كما فعلنا من قبلفي المثال السابق.

سنقوم بتحميل النموذج بجزء إخراج المصنف من النموذج، لكننا نزيل طبقة الإخراج النهائية يدويًا. هذا يعني أن آخر طبقة متصلة بالكامل تحتوي على 4096 عقدة ستكون طبقة الإخراج الجديدة.

```
# load model
model = VGG16()
# remove the output layer
model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
```

سيتم استخدام هذا المتجه المكون من 4096 رقمًا لتمثيل الميزات المعقدة لصورة إدخال معينة يمكن حفظها بعد ذلكفي ملف ليتم تحميلها لاحقًا واستخدامها كمدخلات لتدريب نموذج جديد. يمكننا حفظه كملف pickle.

```
# get extracted features
features = model.predict(image)
print(features.shape)
# save to file
dump(features, open('dog.pkl', 'wb'))
```

بربط كل هذا معًا، يتم سرد المثال الكامل لاستخدام النموذج كنموذج استخراج ميزة مستقل أدناه.

```
# example of using the vgg16 model as a feature extraction model
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16
from keras.models import Model
from pickle import dump
# load an image from file
image = load_img('dog.jpg', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1],
image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# load model
model = VGG16()
# remove the output layer
model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
# get extracted features
features = model.predict(image)
print(features.shape)
```

```
# save to file
dump(features, open('dog.pkl', 'wb'))
```

يؤدي تشغيل المثال إلى تحميل الصورة، ثم تحضير النموذج كنموذج استخراج ميزة.

يتم استخراج الميزات من الصورة التي تم تحميلها ويتم طباعة شكل متجه الميزة feature vector، مما يوضح أنه يحتوي على 4096 رقمًا. ثم يتم حفظ هذه الميزةفي ملف جديد dog.pklفي دليل العمل الحالي.

```
(4096,1)
```

يمكن تكرار هذه العملية لكل صورةفي مجموعة بيانات تدريب جديدة.

نموذج مدرب مسبقًا كمستخرج ميزة في النموذج

يمكننا استخدام بعض أو كل الطبقاتفي نموذج مدرب مسبقًا كمكون لاستخراج ميزة لنموذج جديد مباشرةً.

يمكن تحقيق ذلك عن طريق تحميل النموذج، ثم ببساطة إضافة طبقات جديدة. قد يتضمن ذلك إضافة طبقات تلافيفية وتجميعية جديدة للتوسعفي إمكانات استخراج المعالم للنموذج أو إضافة طبقات نوع مصنف جديد متصل بالكامل لمعرفة كيفية تفسير الميزات المستخرجةفي مجموعة بيانات جديدة، أو بعض التركيبات.

على سبيل المثال، يمكننا تحميل نماذج VGG16 بدون جزء المصنف من النموذج عن طريق تحديد وسيطة "include_top" إلى "False" ، وتحديد الشكل المفضل للصورفي مجموعة البيانات الجديدة لدينا على أنه 300 × 300.

```
# load model without classifier layers
model = VGG16(include_top=False, input_shape=(300, 300, 3))
```

يمكننا بعد ذلك استخدام Keras function API لإضافة طبقة مسطحةFlatten layer جديدة بعد آخر طبقة تجميع pooling layerفي نموذجVGG16 ، ثم تحديد نموذج مصنف جديد بطبقة كثيفة متصلة بالكامل Dense fully connected layer وطبقة إخراج تتنبأ باحتمالية 10 فئات.

```
# add new classifier layers
flat1 = Flatten()(model.layers[-1].output)
class1 = Dense(1024, activation='relu')(flat1)
output = Dense(10, activation='softmax')(class1)
# define new model
model = Model(inputs=model.inputs, outputs=output)
```

تتمثل الطريقة البديلة لإضافة طبقة مسطحة Flatten layerفي تعريف نموذج VGG16 بطبقة تجميع متوسطة average pooling layer، ثم إضافة طبقات متصلة بالكامل Flatten layer. ربما جرب كلا الأسلوبينفي تطبيقك واعرف أيهما يحقق أفضل أداء.

سيتم تدريب أوزان طراز VGG16 وأوزان الطراز الجديد معًا على مجموعة البيانات الجديدة.

المثال الكامل مدرج أدناه.

```
# example of tending the vgg16 model
from keras.applications.vgg16 import VGG16
from keras.models import Model
from keras.layers import Dense
from keras.layers import Flatten
# load model without classifier layers
model = VGG16(include_top=False, input_shape=(300, 300, 3))
# add new classifier layers
flat1 = Flatten()(model.layers[-1].output)
class1 = Dense(1024, activation='relu')(flat1)
output = Dense(10, activation='softmax')(class1)
# define new model
model = Model(inputs=model.inputs, outputs=output)
# summarize
model.summary()
# ...
```

يؤدي تشغيل المثال إلى تحديد النموذج الجديد الجاهز للتدريب ويلخص بُنية النموذج.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 300, 300, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 300, 300, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 300, 300, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 150, 150, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 150, 150, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 150, 150, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 75, 75, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 75, 75, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 75, 75, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 75, 75, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 37, 37, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 37, 37, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 18, 18, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 41472)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              42468352
_________________________________________________________________
dense_2 (Dense)              (None, 10)                10250
=================================================================
Total params: 57,193,290
Trainable params: 57,193,290
Non-trainable params: 0
_________________________________________________________________
```

يمكننا أن نرى أننا قمنا بتسوية إخراج آخر طبقة تجميع وإضافة طبقات جديدة متصلة بالكامل.

بالتناوب، قد نرغبفي استخدام طبقات نموذج VGG16، ولكن تدريب الطبقات الجديدة للنموذج دون تحديث أوزان طبقات VGG16. سيسمح ذلك لطبقات الإخراج الجديدة بتعلم تفسير الميزات التي تم تعلمها لنموذج VGG16.

يمكن تحقيق ذلك عن طريق تعيين خاصية "trainable" على كل طبقةفي نموذج VGG المحمل إلى False قبل التدريب. على سبيل المثال:

```
# load model without classifier layers
model = VGG16(include_top=False, input_shape=(300, 300, 3))
# mark loaded layers as not trainable
for layer in model.layers:
        layer.trainable = False
...
```

يمكنك انتقاء واختيار الطبقات التي يمكن تدريبها.

على سبيل المثال، ربما ترغبفي إعادة تدريب بعض الطبقات التلافيفيةفي عمق النموذج، لكن لا تريد إعادة تدريب أي من الطبقات السابقةفي النموذج. على سبيل المثال:

```
# load model without classifier layers
model = VGG16(include_top=False, input_shape=(300, 300, 3))
# mark some layers as not trainable
model.get_layer('block1_conv1').trainable = False
model.get_layer('block1_conv2').trainable = False
model.get_layer('block2_conv1').trainable = False
model.get_layer('block2_conv2').trainable = False
...
```

الملخص

في هذا المنشور، اكتشفت كيفية استخدام نقل التعلم عند تطوير الشبكات العصبية التلافيفية لتطبيقات الرؤية الحاسوبية. على وجه التحديد، لقد تعلمت:

- يتضمن نقل التعلم استخدام نماذج مدربة على مشكلة واحدة كنقطة بداية لمشكلة ذات صلة.
- يتسم نقل التعلم بالمرونة، مما يسمح باستخدام النماذج المدربة مسبقًا بشكل مباشر كمعالجة مسبقة لاستخراج الميزات ودمجهافي نماذج جديدة تمامًا.
- يوفر Keras وصولاً مناسبًا إلى العديد من النماذج عالية الأداءفي مهام التعرف على الصور على ImageNet مثل VGG و Inceptionو.ResNet .

المصدر:

https://machinelearningmastery.com/how-to-use-transfer-learning-/when-developing-convolutional-neural-network-models

4) التعرف على الصور: الكلاب مقابل القطط! باستخدام نقل التعلم Image Recognition: Dogs vs Cats! using Transfer Learning

المقدمة

في هذه المقالة، ستقوم ببناء شبكة عصبية تلافيفية CNN من البداية لتصنيف الصور إلى فئتين، كلب أو قطة، بدقة 92٪.

لن نستخدم نقل التعلم transfer learning هذه المرة (لذا لا غش!)، وسأشرح بالتفصيل العملية التي اتبعتها لحل هذه المشكلة الكلاسيكية.

سوف تتعلم كيفية:

- بناء وضبط شبكة تلافيفية مع keras لتصنيف الصور.
- اختر المحسن المناسب للتأكد من أن شبكتك قادرة على التعلم.
- استخدم برنامج keras ImageDataGenerator لزيادة مجموعة البيانات الخاصة بك والحد من الضبط الزائد overfitting.

أخيرًا، ستجرب نموذج ResNet50 المدرب مسبقًا، فقط لترى.

تنفيذ هذا البرنامج التعليمي

على عكس معظم المنشوراتفي هذه المدونة، لا أقدم وصفة لتشغيل هذا النوتبوك على Google Colab. لقد جربته لكن يبدو أن:

- عمليات نقل القرص على أجهزة Google Colab الافتراضية بطيئة جدًا. يؤدي هذا إلى إبطاء تدريب الشبكات العصبية على مجموعات بيانات كبيرة نسبيًا مثل مجموعة بيانات الكلاب والقطط.
- عدد نوى وحدة المعالجة المركزية على هذه الأجهزة منخفض جدًا للتعامل مع المعالجة المسبقة للصور الثقيلة التي سنحتاج إلى تطبيقها.

نتيجة لذلك، ستحتاج إلى التشغيل على جهازك.

أولاً ، قم بتثبيت TensorFlow لجهاز الكمبيوتر الذي يعمل بنظام التشغيل Linux أو Windows. باستخدام هذه الطرق، ستقوم أيضًا بتثبيت anaconda، بمافي ذلك حزمة keras.

بعد ذلك، قم بتثبيت هذه الحزم مع conda:

```
conda install numpy matplotlib
```

استنسخ github repo محليًا، وابدأ النوتبوك:

```
git clone https://github.com/cbernet/maldives.git
cd maldives/dogs_vs_cats
jupyter notebook
```

وافتح النوتبوك dogs_vs_cats_local.ipynb.

لم أختبر هذه الطريقة، لذا إذا لم تنجح، أخبرني في التعليقات وسأساعدك على الفور.

مجموعة بيانات الكلاب والقطط

تم تقديم مجموعة بيانات الكلاب والقطط لأول مرة في مسابقة Kaggle في عام 2013. للوصول إلى مجموعة البيانات، ستحتاج إلى إنشاء حساب Kaggle وتسجيل الدخول. لا ضغوط، لسنا هنا من أجل المنافسة، ولكن للتعلم!

مجموعة البيانات متوفرة هنا. يمكنك استخدام الأداة المساعدة kaggle للحصول على مجموعة البيانات، أو ببساطة قم بتنزيل ملف train.zip (حوالي 540 ميجابايت). لا تنس تسجيل الدخول إلى Kaggle أولاً.

تعليمات تحضير مجموعة البيانات مخصصة لنظام التشغيل Linux أو macOS. إذا كنت تعمل على Windows، فأنا متأكد من أنه يمكنك بسهولة العثور على طريقة للقيام بذلك (على سبيل المثال، استخدم 7ـzip قم بفك ضغط الأرشيف، وWindows Explorer لإنشاء أدلة ونقل الملفات).

بمجرد التنزيل، قم بفك ضغط الأرشيف:

```
unzip train.zip
```

قائمة محتويات دليل التدريب:

```
ls train
```

سترى الكثير من صور الكلاب والقطط.

في الأقسام التالية، سنستخدم Keras لاسترداد الصور من القرص باستخدام طريقة flow_from_directory لكلاس ImageDataGenerator.

ومع ذلك، تتطلب هذه الطريقة فرز الصور من الفئات المختلفة في أدلة مختلفة. لذلك سنضع كل صور الكلاب في dogs وكل صور القطط في cats:

```
mkdir cats
mkdir dogs
find train -name 'dog.*' -exec mv {} dogs/ \;
find train -name 'cat.*' -exec mv {} cats/ \;
```

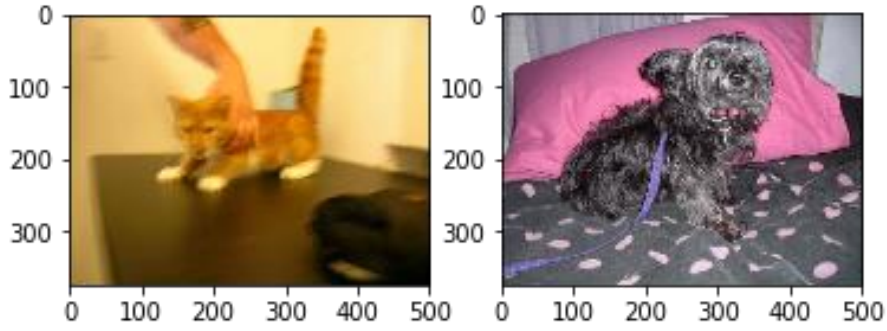
قد تتساءل عن سبب استخدامي للبحث بدلاً من ملف mv البسيط لنقل الملفات. ذلك لأنه مع mv، يحتاج shell إلى تمرير عدد كبير جدًا من الوسائط إلى الأمر (جميع أسماء الملفات)،

وهناك قيود على هذا الرقم على macOS (في Linux، يعمل بشكل جيد). من خلال البحث، يمكننا التغلب على هذا القيد.

التهيئة

الآن، أدخل الخلية الموجودة أسفل موقع دليل مجموعة البيانات، تلك التي تحتوي على الدلائل الفرعية `dogs` و `cats` ، وقم بتنفيذها:

```
# define and move to dataset directory
datasetdir = '/data2/cbernet/maldives/dogs_vs_cats'
import os
os.chdir(datasetdir)

# import the needed packages
import matplotlib.pyplot as plt
import matplotlib.image as img
from tensorflow import keras
# shortcut to the ImageDataGenerator class
ImageDataGenerator = keras.preprocessing.image.ImageDataGenerator
```



رائع. لكن لنكن أكثر تحديدًا ونطبع بعض المعلومات حول صورنا:

```
images = []
for i in range(10):
  im = img.imread('cats/cat.{}.jpg'.format(i))
  images.append(im)
  print('image shape', im.shape, 'maximum color level', im.max())
```

في شكل الصورة، يتوافق العمودان الأولان مع ارتفاع الصورة وعرضها بالبكسل، على التوالي، والعمود الثالث يتوافق مع قنوات الألوان الثلاثة. لذلك يحتوي كل بكسل على ثلاث قيم (للأحمر والأخضر والأزرق على التوالي). نقوم أيضًا بطباعة الحد الأقصى لمستوى اللونفي كل قناة، ويمكننا أن نستنتج أن مستويات RGB تقعفي النطاق من 0ـ255.

تنظيف الحيوانات الأليفة: تحسين جودة مجموعة البيانات

إذا كان هناك شيء واحد فقط يجب أن تستخلصه من هذا البرنامج التعليمي فهو هذا:

يجب ألا تثق أبدًافي بياناتك

البيانات دائما قذرة.

لإلقاء نظرة فاحصة على مجموعة البيانات هذه، استخدمت متصفح صور سريعًا للتحقق من جميع الصورفي مجلدات الكلاب والقطط.في الواقع، لقد استخدمت تطبيق Preview على جهاز Mac الخاص بي لتصفح أيقونات المعاينة الصغيرة. الدماغ سريع جدًافي اكتشاف المشكلات الواضحة حتى لو تركت عينيك تتجولفي عدد كبير من الصور. لذلك لم يستغرقني هذا العمل (الممل) أكثر من 20 دقيقة. لكن بالطبع، لقد فاتني بالتأكيد الكثير من القضايا الأقل وضوحًا.

على أي حال، هذا ما وجدته.

أولاً، ها هي مؤشرات الصور السيئة التي يمكن أن أجدهافي كل فئة:

```
bad_dog_ids = [5604, 6413, 8736, 8898, 9188, 9517, 10161,
               10190, 10237, 10401, 10797, 11186]

bad_cat_ids = [2939, 3216, 4688, 4833, 5418, 6215, 7377,
               8456, 8470, 11565, 12272]
```

يمكننا بعد ذلك استرداد الصور بهذه المعرفات من مجلدات `cats`و`dogs`:

```
def load_images(ids, categ):
  '''return the images corresponding to a list of ids'''
  images = []
  dirname = categ+'s' # dog -> dogs
  for theid in ids:
    fname = '{dirname}/{categ}.{theid}.jpg'.format(
        dirname=dirname,
        categ=categ,
        theid=theid
    )
    im = img.imread(fname)
    images.append(im)
  return images
```

```
bad_dogs = load_images(bad_dog_ids, 'dog')
bad_cats = load_images(bad_cat_ids, 'cat')
```

```
def plot_images(images, ids):
    ncols, nrows = 4, 3
    fig = plt.figure( figsize=(ncols*3, nrows*3), dpi=90)
    for i, (img, theid) in enumerate(zip(images,ids)):
      plt.subplot(nrows, ncols, i+1)
      plt.imshow(img)
      plt.title(str(theid))
      plt.axis('off')
```

```
plot_images(bad_dogs, bad_dog_ids)
```

بعض هذه الصور لا معنى لها تمامًا، مثل 5604 و8736. بالنسبة إلى 10401 و10797، نرى بالفعل قطةفي الصورة! تنهد ... الاحتفاظ بالكلاب الكرتونية أم لا أمر قابل للنقاش. أشعر أنه سيكون من الأفضل إزالتها. نفس الشيء بالنسبة لـ 6413، يمكننا الاحتفاظ به، لكنني أخشى أن تركز الشبكة على الرسومات حول صورة الكلب.

الآن دعونا نلقي نظرة على القطط السيئة:

```
plot_images(bad_cats, bad_cat_ids)
```

مرة أخرى، لست من محبي الاحتفاظ بقطط الرسوم المتحركة للتدريب، ولكن من يدري، فقد يكون الأمر جيداً.في الصورة 4688، لدينا قطة كرتونية صغيرة وكلب كرتوني كبير ... يجب رفض هذا بوضوح.في الصورة 6215، نرى فروًا فقط، لذا يمكن أن يكون إما قطة أو كلبًا، على الرغم من أن هذا يشبه فراء القطط. ولماذا يوصف الرجلفي الصورة رقم 7377 بأنه قطة؟ لا يوجد فكرة...

ومع ذلك، تجدر الإشارة إلى أنه حتى إذا قررنا رفض صور الكارتون للتدريب، فقد تتمكن الشبكة المدربة من التعرف عليها بشكل صحيح.

الآن دعنا ننفذ دالة صغيرة لتنظيف مجموعة البيانات:

```
import glob
import re
import shutil

# matches any string with the substring ".<digits>."
# such as dog.666.jpg
pattern = re.compile(r'.*\.(\d+)\..*')

def trash_path(dirname):
    '''return the path of the Trash directory,
    where the bad dog and bad cat images will be moved.
    Note that the Trash directory should not be within the dogs/
    or the cats/ directory, or Keras will still find these pictures.
    '''
    return os.path.join('../Trash', dirname)

def cleanup(ids, dirname):
  '''move away images with these ids in dirname
  '''
  os.chdir(datasetdir)
  # keep track of current directory
  oldpwd = os.getcwd()
  # go to either cats/ or dogs/
  os.chdir(dirname)
  # create the trash directory.
  # if it exists, it is first removed
  trash = trash_path(dirname)
  if os.path.isdir(trash):
    shutil.rmtree(trash)
  os.makedirs(trash, exist_ok=True)
  # loop on all cat or dog files
  fnames = os.listdir()
  for fname in fnames:
    m = pattern.match(fname)
    if m:
      # extract the id
      the_id = int(m.group(1))
      if the_id in ids:
        # this id is in the list of ids to be trashed
        print('moving to {}: {}'.format(trash, fname))
        shutil.move(fname, trash)
  # going back to root directory
  os.chdir(oldpwd)

def restore(dirname):
  '''restores files in the trash
  I will need this to restore this tutorial to initial state for you
  and you might need it if you want to try training the network
  without the cleaning of bad images
  '''
  os.chdir(datasetdir)
  oldpwd = os.getcwd()
  os.chdir(dirname)
  trash = trash_path(dirname)
  print(trash)
```

```
  for fname in os.listdir(trash):
    fname = os.path.join(trash,fname)
    print('restoring', fname)
    print(os.getcwd())
    shutil.move(fname, os.getcwd())
  os.chdir(oldpwd)
```

```
cleanup(bad_cat_ids,'cats')
```

```
moving to ../Trash/cats: cat.4688.jpg
moving to ../Trash/cats: cat.6215.jpg
moving to ../Trash/cats: cat.11565.jpg
moving to ../Trash/cats: cat.8470.jpg
moving to ../Trash/cats: cat.3216.jpg
moving to ../Trash/cats: cat.2939.jpg
moving to ../Trash/cats: cat.4833.jpg
moving to ../Trash/cats: cat.8456.jpg
moving to ../Trash/cats: cat.7377.jpg
moving to ../Trash/cats: cat.12272.jpg
moving to ../Trash/cats: cat.5418.jpg
```

```
cleanup(bad_dog_ids, 'dogs')
```

```
moving to ../Trash/dogs: dog.10190.jpg
moving to ../Trash/dogs: dog.10797.jpg
moving to ../Trash/dogs: dog.5604.jpg
moving to ../Trash/dogs: dog.10237.jpg
moving to ../Trash/dogs: dog.8736.jpg
moving to ../Trash/dogs: dog.6413.jpg
moving to ../Trash/dogs: dog.10161.jpg
moving to ../Trash/dogs: dog.8898.jpg
moving to ../Trash/dogs: dog.9517.jpg
moving to ../Trash/dogs: dog.10401.jpg
moving to ../Trash/dogs: dog.9188.jpg
moving to ../Trash/dogs: dog.11186.jpg
```

إذا كنت ترغبفي استعادة مجموعة البيانات إلى نسختها الأصلية قبل التنظيف، فما عليك سوى إلغاء التعليق وتنفيذ ما يلي:

```
# restore('dogs')
# restore('cats')
```

تحميل مجموعة بيانات صور الكلاب والقطط باستخدام Keras

يتم تدريب الشبكات العصبية من خلال تقديمها مع مجموعات من الصور، كل منها مع تسمية label يحدد الطبيعة الحقيقية للصورة (إما قطة أو كلبفي حالتنا). قد تحتوي الدُفعة batch على ترتيب يتراوح بين بضعة أعشار إلى بضع مئات من الصور. إذا كنت تريد مقدمة عن الشبكات العصبية والتعلم الخاضع للإشراف من أجل التصنيف، فيمكنك التحقق من منشوري على التعرف على الأرقام المكتوبة بخط اليد باستخدام scikit-Learn.

لكل صورة، تتم مقارنة توقع الشبكة مع التسمية المقابلة، ويتم تقييم المسافة بين تنبؤات الشبكة والحقيقة للمجموعة بأكملها. بعد ذلك، بعد معالجة الدفعة، يتم تغيير معلمات الشبكة بطريقة

تقلل هذه المسافة، وبالتالي تحسين قدرة التنبؤ للشبكة. ثم يبدأ التدريب بشكل متكرر، دفعة بعد دفعة.

لذلك نحن بحاجة إلى طريقة لتحويل صورنا، الآن الملفات الموجودة على القرص، إلى مجموعات من مصفوفات البياناتفي الذاكرة والتي يمكن تغذيتها للشبكة أثناء التدريب.

يمكن استخدام ImageDataGenerator بسهولة لهذا الغرض. دعنا نستورد هذه الفئة وننشئ مثيلًا للمولد:

```
gen = ImageDataGenerator()
```

الآن، سوف نستخدم طريقة flow_from_directory للكائن gen لبدء إنشاء الدُفعات.

ستعيد هذه الطريقة مكررًا يقوم بإرجاع دفعةفي كل مرة يتم تكرارها. لمعرفة كيفية تنظيم البيانات، يمكننا ببساطة إنشاء هذا المكرر، والحصول على أول دفعة لإلقاء نظرة عليها:

```
iterator = gen.flow_from_directory(
    os.getcwd(),
    target_size=(256,256),
    classes=('dogs','cats')
)


Found 24977 images belonging to 2 classes.
```

```
# we can guess that the iterator has a next function,
# because all python iterators have one.
batch = iterator.next()
len(batch)

2
```

تحتوي الدفعة على عنصرين. ما هو نوعها؟

```
print(type(batch[0]))
print(type(batch[1]))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

مصفوفتان numpy! حسنًا، هذا يعني أنه يمكننا طباعة شكلها وكتابتها:

```
print(batch[0].shape)
print(batch[0].dtype)
print(batch[0].max())
print(batch[1].shape)
print(batch[1].dtype)
```

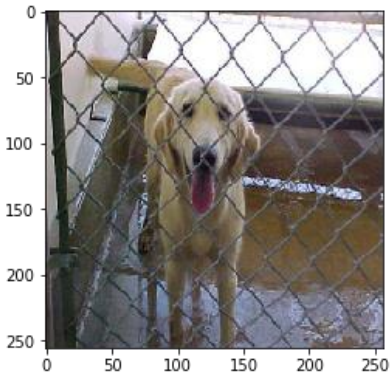
```
(32, 256, 256, 3)
float32
255.0
(32, 2)
float32
```

من الواضح أن العنصر الأول عبارة عن مصفوفة من 32 صورة مع 256 × 256 بكسل، و 3 قنوات ملونة، تم ترميزها على أنها عائمةفي النطاق من 0 إلى 255. لذا فقد فرض ImageDataGenerator الصورة على 256 × 256 بكسل حسب الطلب، لكنه لم يقم بتسوية اللون بين 0 و 1. سيتعين علينا القيام بذلك لاحقًا.

يحتوي العنصر الثاني على 32 تسمية مقابلة.

قبل إلقاء نظرة مفصلة على التسميات، يمكننا رسم الصورة الأولى:

```
import numpy as np
# we need to cast the image array to integers
# before plotting as imshow either takes arrays of integers,
# or arrays of floats normalized to 1.
plt.imshow(batch[0][0].astype(np.int))
```



وهنا التسمية المقابلة:

```
batch[1][0]

array([1., 0.], dtype=float32)
```

نرى أن ImageDataGenerator أنتج تلقائيًا تسمية لكل صورة اعتمادًا على الدليل الذي تم العثور عليه فيه. يتم استخدام ترميز واحد ساخن One-hot encoding للتسميات، وهذا بالضبط ما نحتاجه لمهمة التصنيف التي نحن بصدد القيام بها. إذا كنت تريد معرفة المزيد عن الترميز الواحد الساخن، فتحقق من منشوري حول أول شبكة عصبية مع Keras.

يمكننا أيضًا تخمين أن التصنيف `[0., 1.]` يتوافق مع قطة حقيقية، و `[1.,0.]` لكلب حقيقي. سيقع توقع الشبكة لصورة معينةفي مكان ما بينهما، مثل `[0.6,0.4]` لكلب. ومع ذلك، هذا مجرد تخمين، والتخمين لا يكفي! نحتاج إلى التأكد، أو نخاطر بتغذية شبكتنا بصور خاطئة (قمامة تدخل، قمامة تخرج garbage in, garbage out).

رسم الحيوانات الأليفة للتحقق من صحة التسمية

للتحقق من صحة تسميات مجموعة البيانات، نريد التحقق من تعيين التسميات بشكل صحيح على دفعات قليلة. لذا فنحن بحاجة إلى دالة يمكنها رسم عدد كبير نسبيًا من الصور وتسميتها. ها هو:

نظرًا لأنها قد تكون المرة الأولى التي تستخدم فيها ImageDataGenerator ، فمن المحتمل أنك تريد أن تكون أكثر ثقة بشأن هذه الأداة. لذلك، سنقوم بتطوير دالة صغيرةفي القسم التالي للتحقق من صحة مجموعة بيانات الإدخال.

```
def plot_images(batch):
    imgs = batch[0]
    labels = batch[1]
    ncols, nrows = 4,8
    fig = plt.figure( figsize=(ncols*3, nrows*3), dpi=90)
    for i, (img,label) in enumerate(zip(imgs,labels)):
      plt.subplot(nrows, ncols, i+1)
      plt.imshow(img.astype(np.int))
      assert(label[0]+label[1]==1.)
      categ = 'dog' if label[0]>0.5 else 'cat'
      plt.title( '{} {}'.format(str(label), categ))
      plt.axis('off')
```

```
plot_images(iterator.next())
```

يرجى تكرار الخلية السابقة حتى تتأكد من صحة التسميات.

تقسيم عينات التدريب والتحقق من الصحة باستخدام ImageDataGenerator

سنقوم بتدريب شبكتنا العصبية على مجموعة فرعية من صور الكلاب والقطط تسمى مجموعة بيانات التدريب.

إذا كانت الشبكة معقدة بدرجة كافية (إذا كانت تحتوي على معلمات كافية)، فيمكنها البدءفي التخصيص، مما يعني أنها قادرة على تعلم الميزات المحددة للصورفي مجموعة بيانات التدريب. بمعنى آخر، تفقد الشبكة عمومتها وقدرتها على تصنيف صورة عشوائية على أنها صورة كلب أو قطة.

للتأكد من عدم حدوث الضبط الزائد، سنقوم بتقييم أداء الشبكة على مجموعة بيانات التحقق من الصحة، المنفصلة عن مجموعة بيانات التدريب.

لنقم أولاً بإنشاء `ImageDataGenerator` جديد. فيما يتعلق بالسابقة، نطلب:

- إعادة قياس جميع مستويات اللون لتكونفي النطاق 0ـ1. نحن نقوم بهذا لأن الشبكات العصبية تتصرف بشكل أفضلفي متغيرات الإدخال لترتيب الوحدة.
- استخدم 20٪ من الصور للتحقق، وبالتالي 80٪ للتدريب.

```
imgdatagen = ImageDataGenerator(
    rescale = 1/255.,
    validation_split = 0.2,
)
```

بعد ذلك، نحدد مكرراتنا لمجموعات بيانات التدريب والتحقق من الصحة. نستخدم حجم دفعة batch size من 30 لأنه، عادةً، لا تستطيع الشبكات معرفة ما إذا كان حجم الدُفعة كبيرًا جدًا أو صغيرًا جدًا. يمكنك المحاولة مرة أخرى بحجم دفعة مختلف بعد إكمال هذا البرنامج التعليمي.

يتم فرض الصور على 256 × 256 بكسل. هنا، نحتاج فقط إلى التأكد من أن تنسيق جميع الصور هو نفسه، لأن الشبكة العصبية التلافيفية التي سنستخدمها بها عدد ثابت من المدخلات. اخترت شكل مربع لتجنب حدوث الكثير من التشويهفي كل من الصور الأفقية والصورة. ولكن إذا كانت الغالبية العظمى من صورنا صورة شخصية، فربما يكون الشكل الأقرب للصورة هو الخيار الأفضل. لم أتحقق من ذلك.

```
batch_size = 30
height, width = (256,256)

train_dataset = imgdatagen.flow_from_directory(
    os.getcwd(),
    target_size = (height, width),
    classes = ('dogs','cats'),
    batch_size = batch_size,
    subset = 'training'
)

val_dataset = imgdatagen.flow_from_directory(
    os.getcwd(),
    target_size = (height, width),
    classes = ('dogs','cats'),
    batch_size = batch_size,
    subset = 'validation'
)
```

```
Found 19983 images belonging to 2 classes.
Found 4994 images belonging to 2 classes.
```

شبكة عصبية تلافيفية بسيطة

الشبكات العصبية العميقة التلافيفية هي الخيار المفضل عندما يتعلق الأمر بتصنيف الصور. للحصول على مقدمة مفصلة لهذا النوع من الشبكات، راجع منشوري على ضبط مثل هذه الشبكة للتعرف على الأرقام المكتوبة بخط اليد. النموذج أدناه مشابه جدًا للنموذج الذي استخدمناهفي هذه المقالة.

```
model = keras.models.Sequential()
```

```
initializers = {

}
model.add(
    keras.layers.Conv2D(
        24, 5, input_shape=(256,256,3),
        activation='relu',
    )
)
model.add( keras.layers.MaxPooling2D(2) )
model.add(
    keras.layers.Conv2D(
        48, 5, activation='relu',
    )
)
model.add( keras.layers.MaxPooling2D(2) )
model.add(
    keras.layers.Conv2D(
        96, 5, activation='relu',
    )
)
model.add( keras.layers.Flatten() )
model.add( keras.layers.Dropout(0.9) )

model.add( keras.layers.Dense(
    2, activation='softmax',
    )
)

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 252, 252, 24)      1824
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 126, 126, 24)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 122, 122, 48)      28848
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 61, 61, 48)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 57, 57, 96)        115296
_________________________________________________________________
flatten (Flatten)            (None, 311904)            0
_________________________________________________________________
dropout (Dropout)            (None, 311904)            0
_________________________________________________________________
dense (Dense)                (None, 2)                 623810
=================================================================
Total params: 769,778
Trainable params: 769,778
Non-trainable params: 0
_________________________________________________________________
```

فيما يلي الاختلافات الرئيسية فيما يتعلق بالشبكة المستخدمةفي التعرف على الأرقام المكتوبة بخط اليد:

- لدينا اثنين من الخلايا العصبيةفي آخر طبقة softmax بدلاً من 10، حيث لدينا الآن فئتان.
- صور الإدخال لدينا أكبر بكثير، 256 × 256 بدلاً من 28 × 28. لاحظ أن شكل الإدخال للطبقة الأولى يجب أن يتطابق مع الشكل المعطى للمولد.
- لقد أضفت طبقة تلافيفية ثالثة، وأقوم باستخراج المزيد من الميزاتفي كل طبقة.
- تم زيادة معدل التسرب dropout rate من 0.4 إلى 0.9 للمساعدةفي تقليل الضبط الزائد.

النقطتان الأوليان تقنيتان نوعًا ما، نحتاج فقط إلى القيام بذلك وإلا فلن تعمل الشبكة.

النقطتان الأخيرتان بعيدتان عن الوضوح. تأتي هذه الاختيارات من تحسين طويل الأمد. لقد بدأت بطبقتين تلافيفيتين وميزات أقل، لكن دقة التدريب كانت ثابتة، وهي علامة على أن الشبكة كانت تعاني من الضبط الناقص underfitting . هذا يعني أن الشبكة لم يكن لديها معلمات كافية لوصف مجموعة بيانات التدريب. ناهيك عن مجموعة بيانات التحقق.

لذلك قمت بزيادة التعقيد بإضافة طبقة، وزيادة عدد الميزات المستخرجةفي كل طبقة حتى تصل دقة التدريب إلى ما يقرب من 100٪.

في تلك المرحلة، كانت الشبكة تعاني من الضبط الزائد overfitting : كانت دقة التحقق أقل بكثير من دقة التدريب. لذلك قمت بزيادة معدل التسرب ببطء من 0.4 إلى 0.9 لتقليل الضبط الزائد، ووصلت بالفعل إلى هذه القيمة المرتفعة إلى حد ما. وهذا يعني أنه قبل الدخول إلى آخر طبقة كثيفة للتصنيف، تسقط طبقة التسرب 90٪ من المتغيرات من المراحل السابقة للشبكة بشكل عشوائي. هذا كثير!

لتدريب شبكة عصبية، نحتاج إلى استخدام مُحسِّن optimizer. تحدد هذه الأداة كيفية تغيير معلمات الشبكة بعد كل دفعة لتقليل المسافة بين إخراج الشبكة والحقيقة. من بين المحسّنين المطبقينفي Keras، غالبًا ما يختار الناس Adam أو RMSProp ، غالبًا على سبيل العادة.

لكنفي هذه الحالة، لا يعمل هؤلاء المحسنون جيداً. غالبًا ما تبدأ الشبكة التدريبفي تكوين سيئ بعيداً جداً عن مجموعة المعلمات المثلى، ولا يمكنها التعلم. تبدأ الخسارة حوالي 8، ولا تتحسن. نتيجة لذلك، تظل دقة التدريب عند 50٪، وهو ما يعادل تخمينًا عشوائيًا.

لذا عدت إلى التدرج الاشتقاقي العشوائي (SGD). إنه يعمل بشكل جيد والشبكة تتعلم. لكن التدريب طويل جدا. وبالمناسبة، هذا هو السببفي أن الناس اخترعوا محسنات معززة مثل Adam أو RMSProp.

في كل هذه الدراسات، حاولت تغيير معدل التعلم بشكل كبير learning rate، ولكن دون جدوى.

ثم قرأت الورقة حول Adam ، طريقة للتحسين العشوائي ، وقررت أن أجرب AdaMax ، وهو البديل من Adam:

```
model.compile(loss='binary_crossentropy',
              optimizer=keras.optimizers.Adamax(lr=0.001),
              metrics=['acc'])
```

بعد تجميع compiling النموذج، نقوم بتدريبه على مجموعة بيانات التدريب، والتحقق من صحة النتائجفي نهاية كل فترة epoch باستخدام مجموعة بيانات التحقق من الصحة. أنا أستخدم 10 نوى من وحدة المعالجة المركزية الخاصة بي للتعامل مع مهام ImageDataGenerator، واثنان من GeForce GTX 1080 Ti لـ TensorFlow.

يجب أن تستغرق كل فترة حوالي دقيقة واحدة. إذا استغرق الأمر أكثر من ذلك بكثير، مثل عشرين دقيقة، فيجب عليك التأكد من أنك تستخدم بالفعل وحدة معالجة الرسوميات الخاصة بك. تحقق من تثبيت برامج تشغيل nvidia و TensorFlow على Linux أو Windows.

```
history = model.fit_generator(
    train_dataset,
    validation_data = val_dataset,
    workers=10,
    epochs=20,
)
```

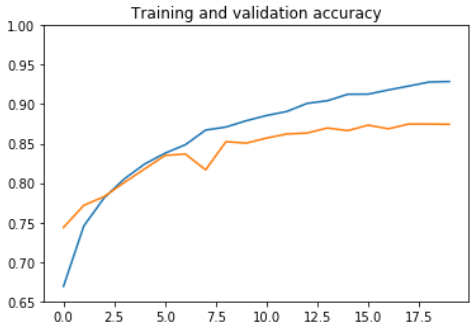
```
Epoch 14/20
667/667 [==============================] - 49s 73ms/step - loss: 0.2323 - acc: 0.9041
Epoch 15/20
667/667 [==============================] - 48s 73ms/step - loss: 0.2159 - acc: 0.9127
Epoch 16/20
667/667 [==============================] - 48s 72ms/step - loss: 0.2145 - acc: 0.9124
Epoch 17/20
667/667 [==============================] - 48s 72ms/step - loss: 0.2038 - acc: 0.9182
Epoch 18/20
667/667 [==============================] - 47s 71ms/step - loss: 0.1917 - acc: 0.9227
Epoch 19/20
667/667 [==============================] - 48s 72ms/step - loss: 0.1812 - acc: 0.9282
Epoch 20/20
667/667 [==============================] - 48s 71ms/step - loss: 0.1771 - acc: 0.9289
```

الآن وبعد الانتهاء من التدريب، نحتاج إلى طريقة لمعرفة كيفية عمل التدريب. لهذا، سنكتب دالة صغيرة لرسم الخطأ والدقة لكل من مجموعات بيانات التدريب والتحقق من الصحة، كدالة للفترة:

```
def plot_history(history, yrange):
    '''Plot loss and accuracy as a function of the epoch,
    for the training and validation datasets.
    '''
    acc = history.history['acc']
    val_acc = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    # Get number of epochs
    epochs = range(len(acc))

    # Plot training and validation accuracy per epoch
    plt.plot(epochs, acc)
    plt.plot(epochs, val_acc)
    plt.title('Training and validation accuracy')
    plt.ylim(yrange)

    # Plot training and validation loss per epoch
    plt.figure()

    plt.plot(epochs, loss)
    plt.plot(epochs, val_loss)
    plt.title('Training and validation loss')

    plt.show()
```

وهنا النتائج:

```
plot_history(history, (0.65, 1.))
```



يمكننا أن نستنتج ما يلي:

- دقة التدريب غير قادرة على الاقتراب من 100٪ بسبب معدل التسرب الكبير.
- تبدأ الشبكةفي الضبط الزائدفي الفترة 6، حتى مع معدل التسرب الكبير.
- تبلغ دقة التحقق من الصحة حوالي 87٪، وهذا ليس سيئًا للغاية.

لا يمكننا زيادة التسرب أكثر، لذلك سنحتاج إلى المزيد من البيانات للقيام بعمل أفضل.في القسم التالي، سنرى كيفية استخدام زيادة البيانات data augmentation لإنشاء المزيد من الصور التدريبية من تلك الموجودة لدينا بالفعل. سيكون ذلك أسهل بكثير وأسرع بكثير من جمع صور القطط والكلاب الجديدة ووضع تسميات عليها.

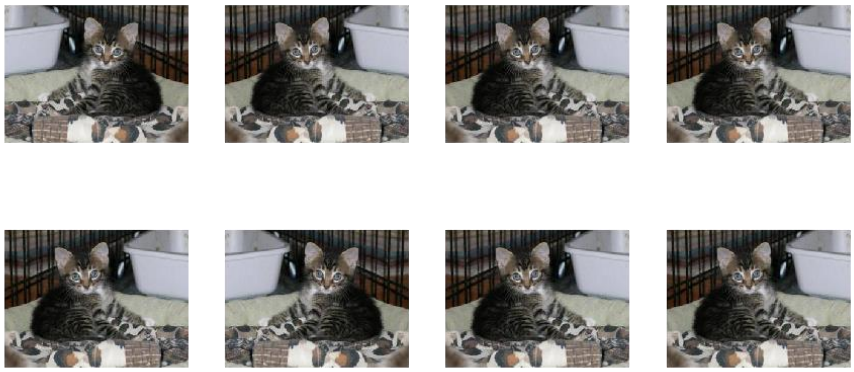
زيادة البيانات

تتمثل زيادة البياناتفي إنشاء أمثلة تدريبية جديدة من الأمثلة التي لدينا بالفعل، بطريقة تؤدي إلى زيادة حجم عينة التدريب بشكل مصطنع. من السهل جدًا القيام بذلك باستخدام ImageDataGenerator. على سبيل المثال، يمكننا أن نبدأ بالتقليب العشوائي randomly flipping لليسار واليمينفي صورنا:

```
imgdatagen = ImageDataGenerator(
    rescale = 1/255.,
    horizontal_flip = True,
    validation_split = 0.2,
)
```

دعونا نرى تأثير هذا التحول على صورة معينة:

```
image = img.imread('cats/cat.12.jpg')

def plot_transform():
    '''apply the transformation 8 times randomly'''
    nrows, ncols = 2,4
    fig = plt.figure(figsize=(ncols*3, nrows*3), dpi=90)
    for i in range(nrows*ncols):
        timage = imgdatagen.random_transform(image)
        plt.subplot(nrows, ncols, i+1)
        plt.imshow(timage)
        plt.axis('off')
```

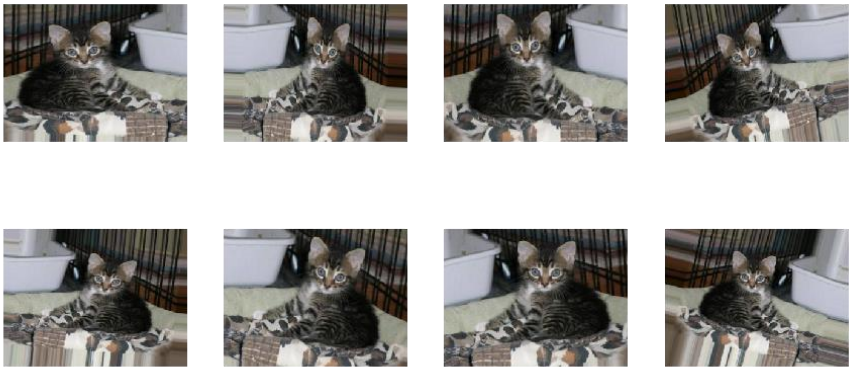


```
plot_transform()
```

يجب أن تكون قادرًا على رؤية تأثير التقليب الأفقي، إلا إذا كنت غير محظوظ حقًا!

الآن، لنجعل التحويل أكثر تعقيدًا. هذه المرة، سيقوم ImageDataGenerator بقلب الصور وتكبيرها / تصغيرها وتدويرها على أساس عشوائي:

```
imgdatagen = ImageDataGenerator(
    rescale = 1/255.,
    horizontal_flip = True,
    zoom range = 0.3,
    rotation_range = 15.,
    validation_split = 0.1,
)

plot_transform()
```



لقد رأينا أن هذه التحولات transformations تنتج صورًا جديدة مقبولة تمامًا. لذلك دعونا نعيد تدريب الشبكة مع زيادة البيانات. من المهم ملاحظة أنه نظرًا للطبيعة العشوائية للتحولات، سترى الشبكة كل صورة مرة واحدة فقط. لذلك يمكننا أن نتوقع أنه سيكون من الصعب على الشبكة أن تزداد.

```
batch_size = 30
height, width = (256,256)

train_dataset = imgdatagen.flow_from_directory(
    os.getcwd(),
    target_size = (height, width),
    classes = ('dogs','cats'),
    batch_size = batch_size,
    subset = 'training'
)

val_dataset = imgdatagen.flow_from_directory(
    os.getcwd(),
    target_size = (height, width),
    classes = ('dogs','cats'),
    batch_size = batch_size,
    subset = 'validation'
)
```

```
Found 22481 images belonging to 2 classes.
Found 2496 images belonging to 2 classes.
```

مع زيادة البيانات، لم تعد هناك حاجة لمعدل تسرب كبير بعد الآن. لقد خفضته من 0.9 إلى 0.2 لكنني لم أحاول ضبط هذه المعلمة:

```
model = keras.models.Sequential()

initializers = {

}
model.add(
    keras.layers.Conv2D(
        24, 5, input_shape=(256,256,3),
        activation='relu',
    )
)
model.add( keras.layers.MaxPooling2D(2) )
model.add(
    keras.layers.Conv2D(
        48, 5, activation='relu',
    )
)
model.add( keras.layers.MaxPooling2D(2) )
model.add(
    keras.layers.Conv2D(
        96, 5, activation='relu',
    )
)
model.add( keras.layers.Flatten() )
model.add( keras.layers.Dropout(0.2) )

model.add( keras.layers.Dense(
    2, activation='softmax',
    )
)

model.summary()
```

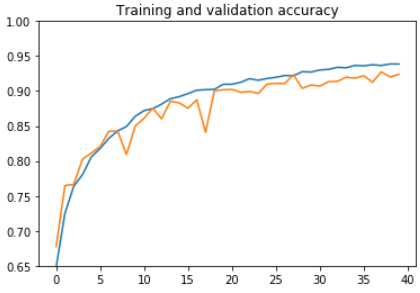
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 252, 252, 24)      1824
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 126, 126, 24)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 122, 122, 48)      28848
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 61, 61, 48)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 57, 57, 96)        115296
_________________________________________________________________
flatten (Flatten)            (None, 311904)            0
_________________________________________________________________
dropout (Dropout)            (None, 311904)            0
_________________________________________________________________
dense (Dense)                (None, 2)                 623810
=================================================================
Total params: 769,778
Trainable params: 769,778
Non-trainable params: 0
_________________________________________________________________
```

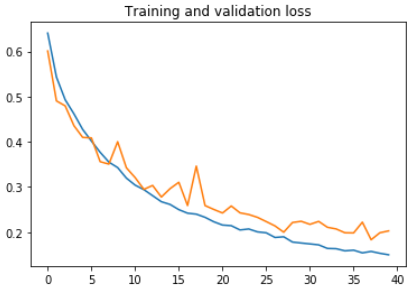
```
model.compile(loss='binary_crossentropy',
              optimizer=keras.optimizers.Adamax(lr=0.001),
              metrics=['acc'])
```

```
history_augm = model.fit_generator(
    train_dataset,
    validation_data = val_dataset,
    workers=10,
    epochs=40,
)
```

```
750/750 [==============================] - 153s 204ms/step - loss: 0.1742 - acc: 0.9301
Epoch 32/40
750/750 [==============================] - 153s 204ms/step - loss: 0.1718 - acc: 0.9311
Epoch 33/40
750/750 [==============================] - 153s 203ms/step - loss: 0.1639 - acc: 0.9340
Epoch 34/40
750/750 [==============================] - 152s 203ms/step - loss: 0.1635 - acc: 0.9333
Epoch 35/40
750/750 [==============================] - 151s 202ms/step - loss: 0.1587 - acc: 0.9366
Epoch 36/40
750/750 [==============================] - 151s 202ms/step - loss: 0.1601 - acc: 0.9360
Epoch 37/40
750/750 [==============================] - 153s 204ms/step - loss: 0.1541 - acc: 0.9377
Epoch 38/40
750/750 [==============================] - 152s 203ms/step - loss: 0.1573 - acc: 0.9366
Epoch 39/40
750/750 [==============================] - 154s 206ms/step - loss: 0.1532 - acc: 0.9388
Epoch 40/40
750/750 [==============================] - 153s 204ms/step - loss: 0.1502 - acc: 0.9386
```

```
plot_history(history_augm, (0.65, 1))
```





كما ترى، مع زيادة البيانات، يستغرق التدريب وقتًا أطول، ولكن يتم تقليل الضبط الزائد كثيرًا، ويمكننا الوصول إلى دقة تصنيف تبلغ 92٪.

يمكننا الاستمرارفي ضبط الشبكة للحد من الضبط الزائد عن طريق زيادة معدل التسرب قليلاً، وتدريب المزيد. لكن أعتقد أننا لن نتمكن من الوصول إلى 95٪في مجموعة البيانات هذه.

لحسن الحظ، هناك طرق أخرى، كما سنرى.

استخدام نموذج مدرب مسبقًا: ResNet50

يعمل العديد من الأشخاص الأذكياء على التعرف على الصور باستخدام أجهزة قوية ومجموعات بيانات كبيرة جدًا. لا أعرف عنك، لكن ليس لدي ذلك.

ومع ذلك، ما يمكننا فعله هو استخدام شبكاتهم مباشرة. تسمى هذه الشبكات بالنماذج المدربة مسبقًا pre-trained models. نظرًا لأنه يمكن تدريب هذه النماذج على مجموعات بيانات كبيرة جدًا، فإنها عادةً ما تتمتع ببُنية عميقة ومعقدة، كما أنها قوية للغاية. يمكن تنزيل هذه النماذج مع ضبط معلماتها على القيم التي كانت لديهافي نهاية التدريب. بهذه الطريقة، يمكننا بسهولة الحصول على أداء ممتاز دون تدريب نموذج جديد بأنفسنا.

فيما يلي قائمة بالموديلات المدربة مسبقًا المتوفرةفي keras.

قررت استخدام ResNet50 ، وهو نموذج تم تدريبه على مجموعة بيانات ImageNet ، والتي تحتوي على 14 مليون صورة تنتمي إلى 1000 فئة.

أولاً، نقوم بتنزيل النموذج وإنشائهفي بضعة أسطر من التعليمات البرمجية:

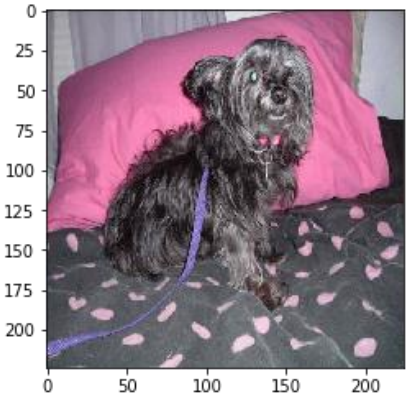
```
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input,
decode_predictions
import numpy as np

model = ResNet50(weights='imagenet')
```

بعد ذلك، نحدد دالة مساعدة صغيرة لتقييم النموذج على صورة إدخال، ونسمي هذه الدالة على بضع صورفي مجموعة البيانات الخاصة بنا:

```
def evaluate(img_fname):
    img = image.load_img(img_fname, target_size=(224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    preds = model.predict(x)
    # print the probability and category name for the 5 categories
    # with highest probability:
    print('Predicted:', decode_predictions(preds, top=5)[0])
    plt.imshow(img)
```

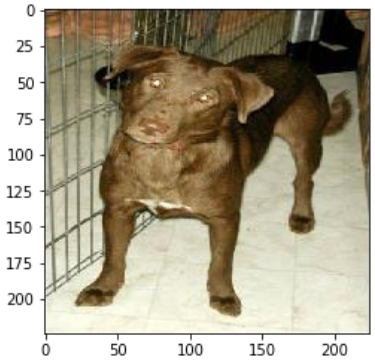
```
evaluate('dogs/dog.0.jpg')
```

```
Predicted: [('n02102318', 'cocker_spaniel', 0.29664052), ('n02097298',
'Scotch_terrier', 0.14396854), ('n02097130', 'giant_schnauzer',
0.14393643), ('n02110627', 'affenpinscher', 0.10783979), ('n02088094',
'Afghan_hound', 0.04753604)]
```



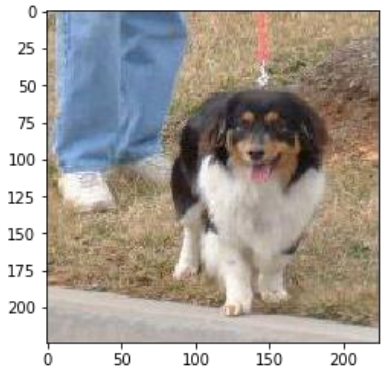

```
evaluate('dogs/dog.1.jpg')
```

```
Predicted: [('n02099849', 'Chesapeake_Bay_retriever', 0.87790024),
('n02105412', 'kelpie', 0.06544642), ('n02099712',
'Labrador_retriever', 0.008923257), ('n02106550', 'Rottweiler',
0.005405719), ('n02099429', 'curly-coated_retriever', 0.004976345)]
```




```
evaluate('dogs/dog.2.jpg')
```

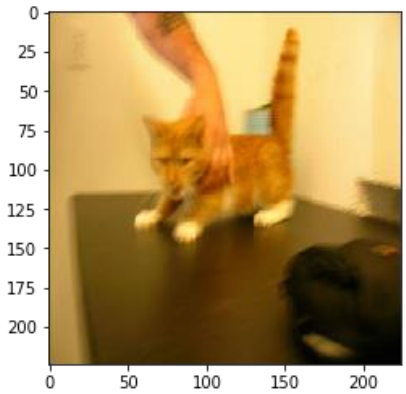
```
Predicted: [('n02108551', 'Tibetan_mastiff', 0.2795359), ('n02097474',
'Tibetan_terrier', 0.21642059), ('n02106030', 'collie', 0.21163173),
('n02106166', 'Border_collie', 0.06243812), ('n02108000',
'EntleBucher', 0.040746134)]
```



كما ترون، فإن الشبكة تعطي الكلاب في أعلى الفئات، بل إنها تتنبأ إلى حد كبير بسباق الكلب!

ماذا عن القطط؟

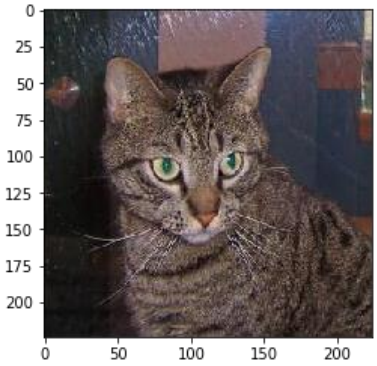
```
evaluate('cats/cat.0.jpg')
Predicted: [('n04404412', 'television', 0.10631036), ('n02094258',
'Norwich_terrier', 0.10413598), ('n02085620', 'Chihuahua',
0.075974055), ('n02093991', 'Irish_terrier', 0.07585583),
('n02123045', 'tabby', 0.07348687)]
```



هنا لا يعمل بشكل جيد. الفئة الأولى هي التلفاز television، ويفترض أن ذلك يرجع إلى تدني جودة الصورة. ثم نحصل على كلاب من هذا اللون، وأخيراً قطة. في الصورة التالية، الشبكة تتصرف بطريقة أفضل:

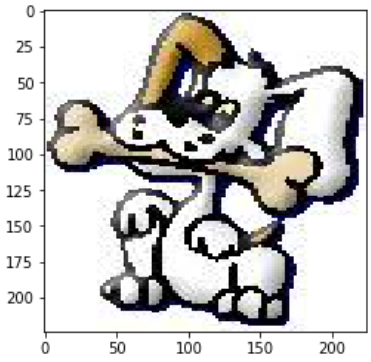
```
evaluate('cats/cat.1.jpg')
```

```
Predicted: [('n02123045', 'tabby', 0.6946943), ('n02123159',
'tiger_cat', 0.18619044), ('n02124075', 'Egyptian_cat', 0.06427032),
('n02127052', 'lynx', 0.00819175), ('n03958227', 'plastic_bag',
0.004613503)]
```



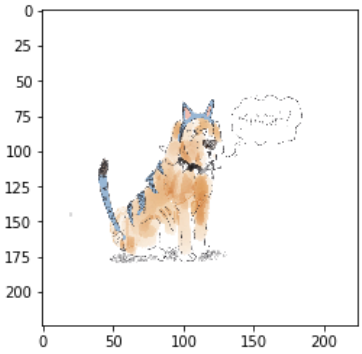
الآن دعنا نحاول مع الرسوم الكاريكاتورية للكلاب:

```
evaluate('Trash/dogs/dog.9188.jpg')

Predicted: [('n02088466', 'bloodhound', 0.11124672), ('n03000684',
'chain_saw', 0.1000548), ('n03814639', 'neck_brace', 0.04567196),
('n03825788', 'nipple', 0.035572648), ('n03803284', 'muzzle',
0.030304618)]
```



فئة الاحتمال الأعلى هي بالفعل كلب! ... يليه عن كثب منشار السلسلة chain saw. الآن ماذا عن صورة غلاف منشور المدونة هذا؟

```
# download the image from my github repository
import urllib.request as req
url =
'https://raw.githubusercontent.com/cbernet/maldives/master/dogs_vs_cat
s/datafrog_chien_chat.png'
req.urlretrieve(url, 'dog_cartoon.jpg')

evaluate('dog_cartoon.jpg')

Predicted: [('n02106662', 'German_shepherd', 0.27522734),
('n02113023', 'Pembroke', 0.17356005), ('n03803284', 'muzzle',
0.12873776), ('n02109047', 'Great_Dane', 0.06609615), ('n02114712',
'red_wolf', 0.060678747)]
```



لم تنخدع الشبكة بالتنكر: إنه كلب!

حسنًا، لقد استمتعنا مع ResNet50، لكن الشبكة لا تفعل ما نريده بالضبط، وهو تصنيف الصور على أنها كلب أو قطة. للقيام بذلك باستخدام نموذج يعتمد على ImageNet، سنحتاج إلى أن نكون قادرين على اكتشاف أن فئة الحبيبات الدقيقة مثل Great_Dane تنتمي بالفعل إلى فئة dog الخشنة. حتى نقوم بذلك، لا يمكننا تحديد أداء هذا النموذجفي سياق مشكلتنا.

يمكننا القيام بذلك يدويًا، لكننا سنتعلم بدلاً من ذلك حلاً أكثر أناقةفي المقالة التالية.

الخلاصة

في هذا المنشور، تعلمت كيفية:

- بناء وضبط شبكة تلافيفية مع keras لتصنيف الصور.
- اختر المحسن المناسب للتأكد من أن شبكتك قادرة على التعلم.
- استخدم برنامج keras ImageDataGenerator لزيادة مجموعة البيانات الخاصة بك والحد من الضبط الزائد.
- استخدم نموذج ResNet50 المدرب مسبقًا على مجموعة بيانات ImageNet.
- لقد وصلنا إلى دقة 92٪ من خلال شبكتنا التلافيفية البسيطة، لكننا رأينا أنه سيكون من الصعب المضي قدمًا.

لزيادة تحسين الأداء، سنحتاج إلى استخدام بُنية أكثر تعقيدًا. لكننا نرى أنه حتى مع شبكتنا التلافيفية الصغيرة وزيادة البيانات الجادة، فقد تأثرنا بالفعل بالضبط الزائد overfitting. سيكون الأمر أكثر صعوبة مع شبكة عصبية متطورة تحتوي على العديد من المعلمات.

ولكنفي المنشور التالي ، التعرف على الصور باستخدام التعلم الانتقالي ، سنرى كيفية استخدام التعلم الانتقالي لاستخدام أحدث الشبكات التلافيفية لتصنيف الكلاب والقطط بدقة 98.5٪!

المصدر:

/https://thedatafrog.com/en/articles/dogs-vs-cats

5) التعرف على الصور مع نقل التعلم Image Recognition with Transfer Learning

في هذه المقالة، ستتعلم كيفية استخدام نقل التعلم transfer learning للتعرف على الصور بشكل فعال، باستخدام keras وTensorFlow والشبكات العصبية المدربة مسبقًا: VGG16 وVGG19 وResNet50.

في هذه العملية سوف تفهم ما هو نقل التعلم، وكيفية القيام ببعض الأشياء التقنية:

- إضافة طبقات إلى شبكة عصبية حالية مدربة مسبقًا لتكييفها مع احتياجاتك.
- احفظ نموذج keras بحيث يمكنك إعادة استخدامه لاحقًا، دون الحاجة إلى إعادة التدريب.
- تقييم أداء النموذج وإلقاء نظرة على الصور التي تم التعرف عليها بشكل خاطئ.

(يُرجى الانتباه إلى أن هذا المنشور لا يقدم مقارنة للأداء بين النماذج الثلاثة. سيكون من الصعب جدًا القيام بذلك بطريقة مناسبة وعادلة).

لكن اولا ...

ما هو نقل التعلم؟

في المقال السابق التعرف على الصور: الكلاب مقابل القطط!، لقد رأينا كيفية بناء شبكة تلافيفية بسيطة من البداية لتصنيف صور الكلاب والقطط بدقة 92٪.

الشبكات العصبية التلافيفية الحديثة مثل VGG أو ResNet أو Inception، ستكون قادرة على أداء هذه المهمة بدقة تزيد عن 99٪. لكن هذه النماذج عميقة ومعقدة. لذلك من الصعب تدريبهم، وهناك عدد كبير جدًا من الصور ضرورية لتدريب هذه الشبكات دون تعرضها للضبط الزائد overfitting.

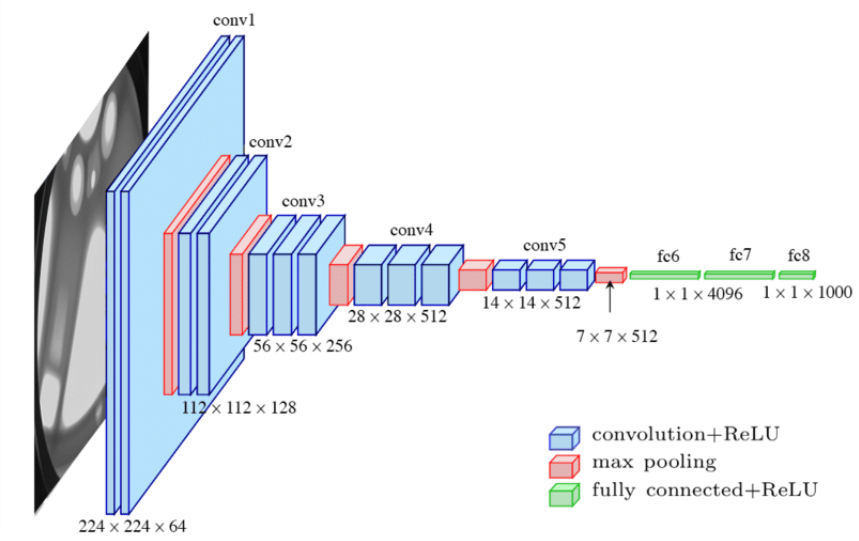
في الواقع، يتم تدريب هذه النماذج الآن على مجموعة بيانات ImageNet، والتي تضم أكثر من 14 مليون صورة مصنفةفي 1000 فئة. بالمقارنة مع ذلك، فإن مجموعة بيانات الكلاب والقطط، التي تحتوي على 25000 صورة، صغيرة بشكل يبعث على السخرية. وقد رأينافي المقالة السابقة أنه حتى مع شبكتنا البسيطة، فنحن مضطرون إلى استخدام زيادة قوية للبيانات للحد من الضبط الزائد. لذا فإن تدريب نموذج معقد على مجموعة البيانات هذه أمر غير وارد.

إذن كيف يمكننا تحسين أداء التصنيففي مجموعة البيانات الصغيرة الخاصة بنا إذا لم تكن هناك طريقة لتدريب نموذج معقد على مجموعة البيانات هذه؟

الحل هو نقل التعلم ، ومن السهل جداً تنفيذه تقنيًا كما سنرى!

ها هي الفكرة.

أولاً، ضعفي اعتبارك بنية الشبكة التلافيفية VGG16، الموضحة أدناه.



معمارية VGG16

في الجزء الأول من الشبكة، نرى خمس كتل تلافيفية convolutional blocks (من 1 إلى 5)، والتي تتكونفي طبقات تلافيفية convolutional layers مكدسة تليها طبقة تجميع قصوى max pooling layer (يمكنك العثور على شرح حول هذه الطبقات هنا). لذلك سنسمي هذا الجزء الجزء التلافيفي convolutional part.

ينتج هذا الجزء موترًا بقيم 7x7x512 لكل صورة. يتماشى البعدان الأولان (7،7) مع أبعاد الصورة الأصلية، ويمكننا التفكيرفي هذاكنسخة خشنة للغاية من الصورة، مع 7 × 7 = 49 بكسل فقط. ولكن لكل بكسل، بدلاً من وجود 3 قنوات ملونة، لدينا 512 ميزة تصف ما تراه الشبكةفي هذا البكسل (وأيضًا حوله).

لذلك يحتوي هذا الموتر على 25 0887 = 7x7x512 رقمًا وهي الميزات المستخرجة بواسطة الشبكة للصورة.

ولكن ما هي بالضبط هذه "الميزات features"؟

في منشوري حول Real Time Human Detection with OpenCV، استخدمنا خوارزمية ذكية (ومعقدة إلى حد ما)، الرسوم البيانية للتدرجات الموجهة (HOG)، لاستخراج ميزات الصورة (أساسًا الحواف في الصورة). يتم بعد ذلك تفسير هذه الميزات بواسطة Support Vector Machine (SVM) لتحديد ما إذا كان هناك إنسان في نافذة الكشف أم لا.

هنا، وبشكل عام في التعلم العميق، ندع الشبكة تكتشف الميزات من تلقاء نفسها أثناء التدريب! لا حاجة لتصميم وترميز خوارزمية استخراج ميزة معقدة، فنحن فقط نعطي الشبكة بُنية ذات مرونة كافية ونقوم بتدريبها بالأمثلة.

فكر في طفل.

لا أعرف الكثير عن الدماغ البشري ولا عن الرؤية البشرية، ولدي خبرة محدودة فقط مع الأطفال! (حصلت على اثنين.) على أي حال هنا ما أعتقد. اعتبرها تشبيهًا، ربما يكون صحيحًا، ربما يكون خاطئًا، لكنني أجده مثيرًا للاهتمام.

خلال الساعات الأولى من حياته، يجب أن يتعلم الطفل كيف يرى. وأعني بذلك أن أفهم تدفقات الإشارات الغامرة التي تأتي من عينيها. أظن أنه في البداية، يتعلم الطفل كيفية رؤية الخطوط، عن طريق توصيل المعلومات من قضبان الشبكية والمخاريط المجاورة. ثم تبدأ في رؤية الأشكال والتعرف على الأشياء. هذا نوع من عملية التعلم غير الخاضع للإشراف: تحتاج فقط إلى تحريك عينيك لترى أن الأشياء متصلة، ولا داعي لعرض أمثلة. وبعد ذلك بكثير، سيتمكن الطفل من استخدام التعلم الخاضع للإشراف: القط! تقول. لا، هذا نمر.

تم تدريب VGG16 على مجموعة بيانات ImageNet الكبيرة وهو قادر بالفعل على الرؤية.

ولكن تم تدريبه أيضًا على تصنيف الصور في 1000 فئة من ImageNet.

يحدث التصنيف في الجزء الثاني من النموذج، والذي يأخذ ميزات الصورة في الإدخال ويختار فئة. يحتوي جزء المصنف هذا على:

- طبقتان مخفيتان متصلتان بالكامل fully connected (أو كثيفتان dense)، تحتوي كل منهما على 4096 خلية عصبية.
- طبقة كثيفة تحتوي على 1000 خلية عصبية، واحدة لكل فئة من فئات ImageNet. يتم استخدام دالة تنشيط softmax لهذه الخلايا العصبية، بحيث تصل القيم الألف التي يصدرونها إلى الوحدة، ويمكن اعتبارها احتمالات.

ما سنفعله، بالنسبة لـ VGG16 والنماذج الأخرى المدربة مسبقًا، هو تنزيل النموذج بالأوزان الناتجة عن التدريب على ImageNet. بعد ذلك، سنقوم باستبدال جزء المصنف بمصنف بسيط

خاص بنا، يتكيف مع مشكلتنا. على سبيل المثال، سيحتوي هذا المصنف على خليتين عصبيتين فقطفي الطبقة الأخيرة، واحدة للكلب والأخرى للقطة. أخيرًا، سنقوم بتجميد جميع طبقات الجزء التلافيفي، بحيث يكون علينا فقط تدريب معلمات المصنف الخاص بنا على مجموعة البيانات الصغيرة الخاصة بنا.

لذلك دعونا نرى كيفية القيام بذلك تقنيًا باستخدام keras.

تنفيذ هذا البرنامج التعليمي

لتشغيل هذا البرنامج التعليمي، سوف تحتاج إلى:

- جهاز كمبيوتر يعمل بنظام Linux أو Windows مزود بوحدة معالجة رسوميات GPU.
- حزم بايثون محددة للتعلم العميق (Keras، TensorFlow) ولتحليل النتائج (matplotlib ،numpy)
- مجموعة بيانات الكلاب والقطط.

إذا كنت ترغبفي إعداد هذا، يرجى الرجوع إلى التعليمات الواردةفي منشوري الأولى التعرف على الصور: الكلاب مقابل القطط! (92٪).

عند الانتهاء، حدد الخلية الموجودة أسفل موقع دليل مجموعة البيانات، أي الخلية التي تحتوي على الدلائل الفرعية للكلاب والقطط. ثم قم بتنفيذ الخلية لاستيراد الحزم المطلوبة.

```
# define and move to dataset directory
datasetdir = '/data2/cbernet/maldives/dogs vs cats'
import os
os.chdir(datasetdir)

# import the needed packages
import matplotlib.pyplot as plt
import matplotlib.image as img
import tensorflow.keras as keras
import numpy as np
```

عدة أدوات

سنبدأ بتحديد دالتي سنحتاجهما لاحقًا.

تقوم الدالة الأولى، المولدات generators، بإرجاع اثنين من مكررات الصور التي سنستخدمها لإنتاج مجموعات من الصور للتدريب والتحقق من صحة شبكاتنا العصبية.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

batch_size = 30

def generators(shape, preprocessing):
    '''Create the training and validation datasets for
```

```
    a given image shape.
    '''
    imgdatagen = ImageDataGenerator(
        preprocessing_function = preprocessing,
        horizontal_flip = True,
        validation_split = 0.1,
    )

    height, width = shape

    train_dataset = imgdatagen.flow_from_directory(
        os.getcwd(),
        target_size = (height, width),
        classes = ('dogs','cats'),
        batch_size = batch_size,
        subset = 'training',
    )

    val_dataset = imgdatagen.flow_from_directory(
        os.getcwd(),
        target_size = (height, width),
        classes = ('dogs','cats'),
        batch_size = batch_size,
        subset = 'validation'
    )
    return train_dataset, val_dataset
```

تحتوي الدوال على معلمتين، الشكل shape والمعالجة المسبقة preprocessing، والتي تعتمد على النموذج المستخدم مسبقًا.

تقوم iterators load بتحميل صور القطط والكلاب من القرص وتحويل هذه الصور إلى مصفوفات بالشكل المحدد. إذا كان الشكل خاطئًا، فلن يتم تكييف الصور مع النموذج، وسوف يتعطل الكود. لذلك علينا توخي الحذر لاختيار الشكل الصحيح لكل نموذج مدرب مسبقًا سنستخدمه. تلتقط كل من VGG16 وVGG19 وResNet50 صورًا ذات شكل (224،224،3)، لذلك مع ثلاث قنوات ملونةفي 224 × 224 بكسل. لكن InceptionV3، على سبيل المثال، قد يلتقط صورًا للشكل (299،299،3).

يتم إنشاء التكرارات بواسطة ImageDataGenerator من kerasالذي يقوم بما يلي:

- لديها فرصة 50٪ لقلب اليسار واليمينفي الصورة. يوفر هذا زيادة أساسية للبيانات دون تكلفة كبيرة من حيث الحوسبة.في الواقع، يتم تنفيذ التقليب بسهولة تحت الغطاء بواسطة numpy بطريقة فعالة للغاية.
- ثم يقوم بتطبيق دالة preprocessingعلى الصورة. يجب تكييف هذه الدالة مع النموذج المدرب مسبقًا المستخدم، ويتم تمريرها إلى دالة generators كوسيطة. في الواقع،في لغة بايثون، الدالة هي كائن، يمكن أن تنتقل بسهولة إلى دوال أخرى).

- يحتفظ بـ 90٪ من الصور للتدريب، ويحجز 10٪ من الصور للتحقق من صحتها. نظرًا لأن لدينا حوالي 25000 صورةفي مجموعة البيانات الخاصة بنا إجمالاً، فإن هذا يترك 2500 صورة للتحقق الدقيق إلى حد ما (المزيد حول هذا لاحقًا).

سيتم استخدام الدالة الثانية لرسم الدقة accuracy والخطأ loss كدالة للفترة epoch، حتى نتمكن من رؤية كيفية عمل التدريب. للحصول على شعور بالضبط الزائد، سيتم رسم هذه الكميات لكل من مجموعات بيانات التدريب training والتحقق من الصحة validation:

```
def plot_history(history, yrange):
    '''Plot loss and accuracy as a function of the epoch,
    for the training and validation datasets.
    '''
    acc = history.history['acc']
    val_acc = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    # Get number of epochs
    epochs = range(len(acc))

    # Plot training and validation accuracy per epoch
    plt.plot(epochs, acc)
    plt.plot(epochs, val_acc)
    plt.title('Training and validation accuracy')
    plt.ylim(yrange)

    # Plot training and validation loss per epoch
    plt.figure()

    plt.plot(epochs, loss)
    plt.plot(epochs, val loss)
    plt.title('Training and validation loss')

    plt.show()
```

نحن الآن جاهزون للبدء بأول نموذج مدرب مسبقًا.

VGG16

تم إنشاء أول نماذج VGG بواسطة Karen Simonyan و Andrew Zisserman، وتم تقديمها لأول مرةفي ورقة شبكات تلافيفية عميقة للغاية للتعرف على الصور على نطاق واسعفي عام 2015. VGG16 بها 16 طبقة مع أوزان ، و VGG99 بها 19 طبقة مع أوزان.

في ذلك الوقت، كانت طرازات VGG بمثابة اختراق حقيقي لعدد من الأسباب. أولاً، كان المؤلفون قادرين على التفوقفي الأداء على المنافسة بنسبة كبيرةفي تحدي التعرف المرئي على نطاق واسع Image Net (ILSVRC). بعد ذلك، أظهروا أنه مع نقل التعلم، فإن نماذجهم تعمم جيدًا على مهام التعرف على الصور الأخرى على مجموعات بيانات أصغر (انظر الملحق B في المقالة)، مما يحقق أداءً متطورًا على مجموعات البيانات هذه أيضًا. أخيرًا، أتاحوا أفضل شبكاتهم أداءً للجمهور لمزيد من البحث والتطبيقات العملية.

من المثير للدهشة أن بُنية VGG واضحة تمامًا وتشبه إلى حد كبير الشبكات التلافيفية الأصلية. كانت الفكرة الرئيسية وراء VGG هي جعل الشبكة أعمق من خلال تكديس المزيد من الطبقات التلافيفية. وقد أصبح هذا ممكنًا من خلال قصر حجم النوافذ التلافيفية على 3 × 3 بكسل فقط.

استخراج الميزة مع VGG16

لذلك دعونا نلقي نظرة على بُنية VGG16. لهذا، قمنا بإنشاء مثيل من طراز VGG16 باستخدام keras، ونطبع الملخص:

```
vgg16 = keras.applications.vgg16
vgg = vgg16.VGG16(weights='imagenet')



vgg.summary()
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808

```
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
```

نرى بوضوح الجزء التلافيفي وجزء المصنف. بين الاثنين، تقوم الطبقة المسطحة Flatten layer بتحويل موتر الميزة للشكل (7،7،512) إلى مصفوفة D1 مع 250887 = x7x512 قيم، والتي يمكن إرسالها كمدخلات إلى أول طبقة كثيفة Dense layer من المصنف.

تم تكييف المصنف مع 1000 فئة من ImageNet. ومع ذلك، فإن مهمتنا هي تصنيف صور الكلاب والقطط، لذلك لدينا فئتان فقط.

ماذا نستطيع ان نفعل؟ باستخدام keras، من السهل استيراد الجزء التلافيفي فقط من VGG16، عن طريق تعيين المعلمة include_top على False:

```
vgg16 = keras.applications.vgg16


conv_model = vgg16.VGG16(weights='imagenet', include_top=False)
conv_model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, None, None, 3)     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
_________________________________________________________________
```

```
_________________________________________________________________
block3_conv3 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, None, None, 512)   0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

يمكنك التحقق في الملخص من أن المصنف قد تمت إزالته بالفعل.

يمكن استخدام النموذج التلافيفي بالفعل لاستخراج الميزات لصورة معينة:

```
from keras.preprocessing import image

img_path = 'dogs/dog.1.jpg'

# loading the image:
img = image.load_img(img_path, target_size=(224, 224))
# turn it into a numpy array
x = image.img to array(img)
print(np.min(x), np.max(x))
print(x.shape)
# expand the shape of the array,
# a new axis is added at the beginning:
xs = np.expand_dims(x, axis=0)
print(xs.shape)
# preprocess input array for VGG16
xs = vgg16.preprocess_input(xs)
# evaluate the model to extract the features
features = conv_model.predict(xs)
```

```
print(features.shape)
```

```
Using TensorFlow backend.
0.0 255.0
(224, 224, 3)
(1, 224, 224, 3)
(1, 7, 7, 512)
```

دعونا نلقي نظرة فاحصة على هذا الكود.

أول شيء مهم يجب ملاحظته هو أن طريقة التنبؤ predictلنموذجنا مصممة للعمل على عدة صور. من المفترض أن يتم تخزين هذه الصورفي مصفوفة numpyذات شكل (n,224,224,3))، حيث n هو عدد الصور المراد معالجتها. لذلك أولاً، قمنا بتحميل صورة، وقمنا بتحويلها إلى مصفوفة متعددة الأشكال (224,224,3). لمطابقة توقيع طريقة predict، أنشأنا بعد ذلك مصفوفة من الأشكال (1,224,224,3) باستخدام np.expand_dims .

النقطة المهمة الأخرى هي أن VGG16 قد تم تدريبه على الصور المعالجة مسبقًا. نقلا عن مقالةVGG .

"المعالجة الوحيدة التي نقوم بها هي طرح متوسط قيمة RGB ، المحسوبةفي مجموعة التدريب من كل بكسل"

للوصول إلى أقصى أداء، من المهم تطبيق نفس المعالجة المسبقة بالضبط قبل تقييم الشبكة. يدعو Keras إلى استخدام vgg16.preprocess_inputs لهذا الغرض، لذلك هذا ما سنفعله.

يمكنك طباعة موتر الميزة feature tensor إذا كنت ترغبفي ذلك، لكن هذا لن يخبرك كثيرًا. هذهفي الحقيقة مجرد مجموعة (كبيرة) من الأرقام. لفهم هذه الأرقام، نحتاج إلى إنشاء المصنف الخاص بنا.

يمكن أن يكون أحد الاحتمالات تخزين الميزاتفي مصفوفات البيانات لكل صورة. بعد ذلك، يمكننا تدريب شبكة عصبية صغيرة على هذه المصفوفات. سيكون هذا النهج قابلاً للتطبيق تمامًا. ومع ذلك، هذا ليس ما سنفعله. بدلاً من ذلك، سنقوم بتمديد VGG16 باستخدام المصنف الخاص بنا. هذا الحل أسهلفي التنفيذ وهو أيضًا أكثر مرونة.

التصنيف المخصص مع VGG16

في وثائق Keras لـ VGG16، وكذلكفي المقال الأصلي، نرى أن إدخال VGG16 يجب أن يكون صورًا بدقة 224 × 224 بكسل. ونعلم أيضًا أنه يجب معالجة الصور بالطريقة الصحيحة

لهذا النموذج. لذلك قمنا بإنشاء مكررات للتدريب والتحقق من الصحة لإنتاج مثل هذه الصور، مع الدالة التي حددناهافي بداية هذا المنشور:

```
train_dataset, val_dataset = generators((224,224),
preprocessing=vgg16.preprocess_input)
```

```
Found 22481 images belonging to 2 classes.
Found 2496 images belonging to 2 classes.
```

كما ترون، ليس لدي كل 25000 صورة لمجموعة بيانات الكلاب والقطط. هذا لأنني قمت بتنظيف مجموعة البيانات لإزالة بعض الأمثلة السيئة حقًا، كما هو موضحفي التعرف على الصور : الكلاب والقطط .إذا لم تكن قد فعلت ذلك، فلا تقلق، فسيعمل هذا البرنامج التعليمي بشكل جيد.

نقوم بإنشاء الجزء التلافيفي مرة أخرى، حيث نحتاج إلى تحديد شكل الإدخال input_shape هذه المرة حتى نتمكن من إنشاء النموذج الكامل:

```
conv_model = vgg16.VGG16(weights='imagenet', include_top=False,
input_shape=(224,224,3))
```

إذا لم تحدد input_shape، تظل أبعاد الشبكة غير محددة، وينتهي بك الأمر برسالة الخطأ التالية عندما تحاول إنشاء أول طبقة كثيفة من المصنف أدناه.

```
ValueError: The last dimension of the inputs to `Dense` should be defined. Found `None`.
```

ثم نقوم بتوصيل ناتج الجزء التلافيفي بالمصنف:

```
# flatten the output of the convolutional part:
x = keras.layers.Flatten()(conv_model.output)
# three hidden layers
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dense(100, activation='relu')(x)
# final softmax layer with two categories (dog and cat)
predictions = keras.layers.Dense(2, activation='softmax')(x)



# creating the full model:
full_model = keras.models.Model(inputs=conv_model.input,
outputs=predictions)
full_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
```

```
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 100)               2508900
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 202
=================================================================
Total params: 17,243,990
Trainable params: 17,243,990
Non-trainable params: 0
_________________________________________________________________
```

نقوم بإغلاق جميع طبقات الجزء التلافيفي:

```
for layer in conv_model.layers:
    layer.trainable = False
```

ونتحقق من أن الطبقات الوحيدة التي سيتم تدريبها هي طبقات المصنف الكثيف dense classifier:

```
full_model.summary()
```

```
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 202
=================================================================
Total params: 17,243,990
Trainable params: 17,243,990
Non-trainable params: 0
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 100)               2508900
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 202
=================================================================
Total params: 17,243,990
Trainable params: 2,529,302
Non-trainable params: 14,714,688
```

في الواقع، نرى أن عدد المعلمات القابلة للتدريب هو العدد الإجمالي للمعلماتفي آخر 4 طبقات كثيفة:

```
2508900+10100*2+202
```

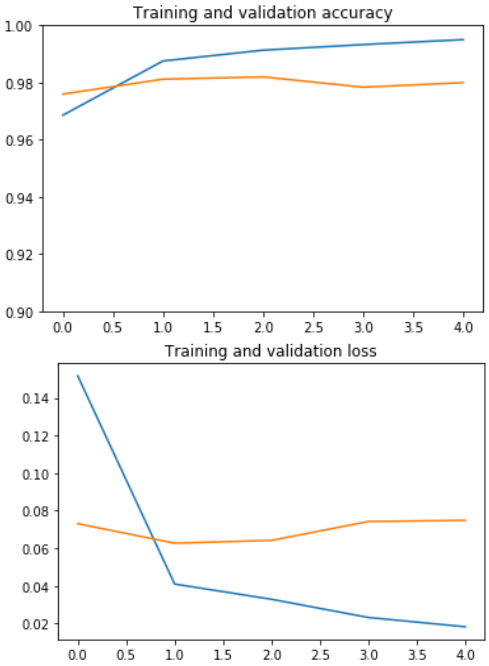
```
2529302
```

يمكننا الآن تجميع النموذج وتدريبه:

```
full_model.compile(loss='binary_crossentropy',
                   optimizer=keras.optimizers.Adamax(lr=0.001),
                   metrics=['acc'])
history = full_model.fit_generator(
    train_dataset,
    validation_data = val_dataset,
    workers=10,
    epochs=5,
)
```

```
Epoch 1/5
750/750 [==============================] - 73s 97ms/step - loss: 0.1518 - acc: 0.9685 - v
Epoch 2/5
750/750 [==============================] - 71s 94ms/step - loss: 0.0409 - acc: 0.9876 - v
Epoch 3/5
750/750 [==============================] - 72s 96ms/step - loss: 0.0329 - acc: 0.9913 - v
Epoch 4/5
750/750 [==============================] - 71s 94ms/step - loss: 0.0232 - acc: 0.9932 - v
Epoch 5/5
750/750 [==============================] - 72s 96ms/step - loss: 0.0182 - acc: 0.9950 - v
```

```
plot_history(history, yrange=(0.9,1))
```



نرى أن النموذج سريع جدًافي التدريب، حيث نحتاج فقط إلى تدريب جزء المصنف. فترة واحدة تكفيفي الواقع للوصول إلى دقة تحقق تبلغ حوالي 98٪، وهي نسبة أعلى بكثير من 92٪ التي حصلنا عليها عندما قمنا بتدريب شبكة تلافيفية بسيطة من الصفر بمفردنافي التعرف على الصور: الكلاب مقابل القطط!.

بعد ذلك، الضبط الزائد overfitting معتدل ويحد من الأداء. ربما يمكننا العمل على الضبط الزائد عن طريق تبسيط المصنف، أو عن طريق إضافة طبقة تسرب dropout layer قبل المصنف مباشرة. لكن هذا خارج نطاق هذا المنشور.

استطرادية موجزة حول عدم اليقين الإحصائي

لاحظ أن قيمة الدقة لا ينبغي أن تؤخذ على محمل الجد. تعتمد دقة التحقق على عينة التحقق المحددة التي اختارها المولد، ويتأثر هذا الرقم بعدم اليقين الإحصائي statistical uncertainty. بشكل تقريبي، لدينا 2500 صورةفي مجموعة بيانات التحقق من الصحة. عدم دقة 2٪ يعني أننا أخطأنافي تصنيف 50 صورة.

نظرًا لأن هذا الرقم صغير إلى حد ما، فإنه يتأثر بعدم اليقين الإحصائي الكبير إلى حد ما. ويمكننا تقدير عدم اليقين النسبي على هذا الرقم على أنه $1/\sqrt{50} \sim 15\%$. كما أن عدم اليقين النسبي بشأن عدم الدقة يبلغ 15٪. لذلك بالنسبة لعدم الدقة بنسبة 2٪، لدينا عدم يقين مطلق قدره $2 \times 0.15 = 0.3$٪.

لذلك، عندما نتحدث عن دقة 98٪، يجب أن تتذكر أن الدقة الحقيقية يجب أن تكونفي حدود $98 \pm 0.3\%$.

يمكننا استخدام مجموعة بيانات تحقق أكبر لتقليل عدم اليقين، لكن هذا سيترك صورًا أقل لتدريب شبكاتنا العصبية.

VGG19

VGG19 هو أحدث إصدار من طرازات VGG وهو مشابه جدًا لـ VGG16. إذا قارنت ملخص النموذج أدناه بملخص VGG16، فسترى أن البُنية هي نفسها، ولا تزال تعتمد على خمس كتل تلافيفية.

ومع ذلك، فقد تم زيادة عمق الشبكة عن طريق إضافة طبقة تلافيفيةفي الكتل الثلاثة الأخيرة.

لا يزال الإدخال عبارة عن صورة RGB للشكل (224،224،3)، والإخراج هو ميزة موتر للشكل (7،7،512). يوفر Keras دالة معالجة مسبقة محددة لـ VGG19، ولكن إذا نظرت إلى الكود،

فسترى أنها نفس الدالة تمامًا مثل VGG 16. لذلك لا نحتاج إلى إعادة تعريف مكررات مجموعة البيانات الخاصة بنا.

لنقم الآن ببناء النموذج الكامل والتحقق منه:

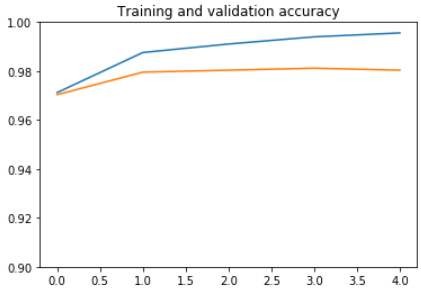
```
vgg19 = keras.applications.vgg19
conv_model = vgg19.VGG19(weights='imagenet', include_top=False,
input_shape=(224,224,3))
for layer in conv_model.layers:
    layer.trainable = False
x = keras.layers.Flatten()(conv_model.output)
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dense(100, activation='relu')(x)
predictions = keras.layers.Dense(2, activation='softmax')(x)
full_model = keras.models.Model(inputs=conv_model.input,
outputs=predictions)
```

```
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_4 (Dense)              (None, 100)               2508900
_________________________________________________________________
dense_5 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_6 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 202
=================================================================
Total params: 22,553,686
Trainable params: 2,529,302
Non-trainable params: 20,024,384
```

```
full_model.summary()
full_model.compile(loss='binary_crossentropy',
                   optimizer=keras.optimizers.Adamax(lr=0.001),
                   metrics=['acc'])
history = full_model.fit_generator(
    train_dataset,
    validation_data = val_dataset,
    workers=10,
    epochs=5,
)
```

```
Epoch 1/5
750/750 [==============================] - 83s 110ms/step - loss: 0.1480 - acc: 0.9712 -
Epoch 2/5
750/750 [==============================] - 84s 112ms/step - loss: 0.0441 - acc: 0.9876 -
Epoch 3/5
750/750 [==============================] - 83s 111ms/step - loss: 0.0320 - acc: 0.9910 -
Epoch 4/5
750/750 [==============================] - 83s 111ms/step - loss: 0.0251 - acc: 0.9940 -
Epoch 5/5
750/750 [==============================] - 83s 111ms/step - loss: 0.0172 - acc: 0.9956 -
```

```
plot_history(history, yrange=(0.9,1))
```



ResNet50

تم تقديم ResNet لأول مرةفي عام 2015 بواسطة Kaiming He و Xiangyu Zhang و Shaoqing Ren و Jian Sunفي ورقتهم الممتازة Deep Residual Learning للتعرف على الصور.

من بين الإنجازات الأخرى، تمكن المؤلفون من تأمين المركز الأولفي تحدي ILSVRC 2015!

كما أشار مؤلفو VGG بالفعل، فإن عمق التمثيل مهم للغاية.في حالة VGG، يمكن بناء شبكات أعمق باستخدام مرشحات تلافيفية أصغر.

من ناحية أخرى، كان لمؤلفي ResNet فكرة ذكية أخرى: يتم تدريب كتل من بضع طبقات مكدسة على تعلم دالة متبقية residual function فيما يتعلق بإدخال الكتلة، بدلاً من تعلم دالة عامة دون الرجوعفي سياق هندسة الشبكات.

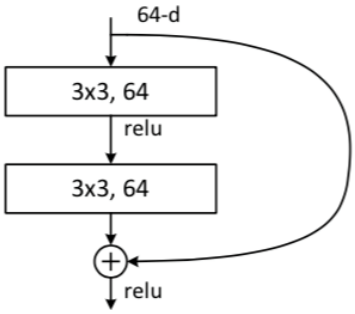
طيب ولكن ماذا يعني هذا؟

من أجل فهم جيد لشبكة ResNet، أنصح بشدة بقراءة الورقة البحثية المكتوبة جيدًا والتعليمية والمفصلة. إذا كنت ترغبفي بدء قراءة الأوراق حول التعلم الآلي (وربما حتى الأوراق العلمية بشكل عام)، فهذا مرشح جيد!

لكن هذا ليس ضروريًافي الواقع لمجرد استخدام ResNet. سألخص هنا وأبسط قليلًا.

أولاً، تذكر أن الشبكات العصبية هي مجرد دوال، ويمكنك العثور على مناقشة حول ذلكفي The 1-Neuron Network: Logistic Regression. إنها متعددة الأبعاد (كل رقمفي صورة الإدخال هو بُعد) وقد تحتوي على ملايين المعلمات والعديد من قيم الإخراج (هنا واحد لكل فئة). ومع ذلك، فهي دوال. يعني تدريب الشبكة ملاءمة الدالة للبيانات عن طريق ضبط معلماتها.

بدلاً من التفكيرفي شبكة ResNet بأكملها، دعنا نركز على إحدى اللبنات الأساسية الخاصة بها مع هذه الصورة المستخرجة من الورقة:



التعلم المتبقي Residual learning، هو وحدة بناء. تأخذ الطبقة التلافيفية الأولىفي الجزء العلوي خريطة معالم بها بكسلات nxn ، ولكل منها 64 معلمًا. يقوم اختصار الهوية identity shortcut بنسخ بيانات الإدخال، والتي يتم جمعها مع إخراج الطبقة التلافيفية الثانية.

إذا كانت صورة الإدخال x، فيمكننا التفكيرفي دالة الشبكة $G(x)$ بأكملها كتكوين لدوال وحدات البناء m،

$$G(x) = h_m(h_{m-1}(\dots h_1(x)))$$

في الشبكات الكلاسيكية، لا يوجد اختصار للتعريف، ويجب على الكتلة i أن تتعلم الدالة $h_i(x)$ مباشرة.في ResNet ، مع اختصار الهوية ، لدينا:

$$h_i(x) = f_i(x) + x$$

حيث f_i (x) هي الدالة المتبقية. بهذه الطريقة، يجب أن تتعلم الكتلة فقط الدالة المتبقية، والتي توفر انحرافات صغيرة فيما يتعلق بالمدخلات. لا يتعين على الكتلة إعادة إنتاج مدخلاتها بعد الآن، بالإضافة إلى نمذجة الانحرافات الصغيرة.

علاوة على ذلك، لا تؤدي إضافة اختصار الهوية إلى زيادة عدد معلمات الكتلة، لأن اختصار الهوية لا يحتوي على معلمة! انها مجرد نسخة.

كما هو موضح في الورقة، فإن هذا التغيير هو الذي جعل من الممكن إنشاء شبكات عميقة للغاية. على سبيل المثال، يحتوي الجزء التلافيفي لشبكة ResNet50 التي سنستخدمها على 50 طبقة، بينما يحتوي VGG19 على 22 طبقة. وهناك أيضًا إصداران أعمق من ResNet وResNet101 وResNet152.

في الوقت نفسه، يتم الاحتفاظ بعدد المعلمات في ResNet50 إلى 34 مليون يمكن التحكم فيها، مقارنة بـ 23 مليون معلمة من VGG19! هذا ما يجعل من الممكن تعلم تمثيل عميق جداً بسهولة من مجموعة بيانات "محدودة" تضم حوالي مليون صورة.

لذلك دعونا نبني شبكة ResNet50 باستخدام keras. أولاً، نقوم بإنشاء مكررات مجموعة البيانات الخاصة بنا، مع شكل الإدخال الصحيح ودوال المعالجة المسبقة:

```
resnet50 = keras.applications.resnet50
train_dataset, val_dataset = generators((224,224),
preprocessing=resnet50.preprocess_input)
```

```
Found 22481 images belonging to 2 classes.
Found 2496 images belonging to 2 classes.
```

بعد ذلك، نقوم بإنشاء نموذجنا الكامل، وهو الجزء التلافيفي من ResNet50 متبوعًا بمصنفنا البسيط، تمامًا مثل VGG ، ونقوم بتدريبه.

```
conv_model = resnet50.ResNet50(weights='imagenet', include_top=False,
input_shape=(224,224,3))
for layer in conv_model.layers:
    layer.trainable = False
x = keras.layers.Flatten()(conv_model.output)
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dense(100, activation='relu')(x)
predictions = keras.layers.Dense(2, activation='softmax')(x)
full_model = keras.models.Model(inputs=conv_model.input,
outputs=predictions)


full_model.summary()
```

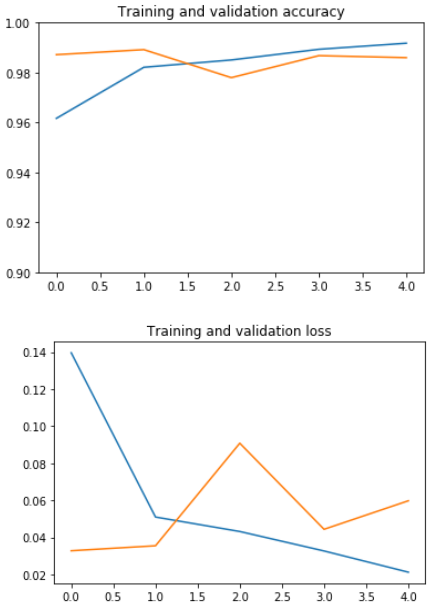
```
_____
flatten_4 (Flatten)          (None, 100352)        0           activation_146[0][0]
_____
dense_16 (Dense)             (None, 100)           10035300    flatten_4[0][0]
_____
dense_17 (Dense)             (None, 100)           10100       dense_16[0][0]
_____
dense_18 (Dense)             (None, 100)           10100       dense_17[0][0]
_____
dense_19 (Dense)             (None, 2)             202         dense_18[0][0]
==========================================================================================
Total params: 33,643,414
Trainable params: 10,055,702
Non-trainable params: 23,587,712
```

```
full_model.compile(loss='binary_crossentropy',
```

```
                    optimizer=keras.optimizers.Adamax(lr=0.001),
                    metrics=['acc'])
history = full_model.fit_generator(
    train_dataset,
    validation_data = val_dataset,
    workers=10,
    epochs=5,
)
```

```
Epoch 1/5
750/750 [==============================] - 64s 85ms/step - loss: 0.1398 - acc: 0.9616 - v
Epoch 2/5
750/750 [==============================] - 60s 80ms/step - loss: 0.0509 - acc: 0.9821 - v
Epoch 3/5
750/750 [==============================] - 60s 80ms/step - loss: 0.0431 - acc: 0.9851 - v
Epoch 4/5
750/750 [==============================] - 61s 82ms/step - loss: 0.0326 - acc: 0.9893 - v
Epoch 5/5
750/750 [==============================] - 61s 81ms/step - loss: 0.0212 - acc: 0.9918 - v
```

```
plot_history(history, yrange=(0.9,1))
```



حفظ وتحميل نموذج Keras

لنفترض أنك حصلت على أداء ممتاز باستخدام نموذج معين. أنت بالتأكيد تريد حفظ هذا النموذج للاستخدامفي المستقبل. لنفعل هذا الآن مع نموذجنا القائم على ResNet50، وهو آخر نموذج قمنا بتدريبه.

باستخدام Keras ، من الممكن جعل النموذج الكامل ثابتًا على القرص ، ولكن قد تصبح النماذج غير قابلة للقراءة عند استخدام طبقات غير قياسية.

أجد حفظ أوزان النموذج (المعلمات) أسهل:

```
full_model.save_weights('resnet50.h5')
```

لقراءتها مرة أخرى، نقوم بإنشاء نموذج جديد مطابق للنموذج الذي قمنا بتدريبه:

```
conv_model = resnet50.ResNet50(weights='imagenet', include_top=False,
input_shape=(224,224,3))
for layer in conv_model.layers:
    layer.trainable = False
x = keras.layers.Flatten()(conv_model.output)
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dense(100, activation='relu')(x)
predictions = keras.layers.Dense(2, activation='softmax')(x)
full_model = keras.models.Model(inputs=conv_model.input,
outputs=predictions)
```

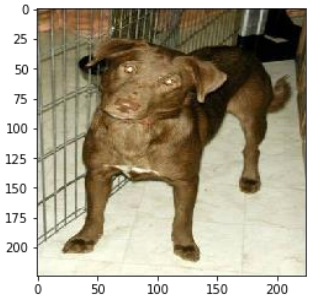
ونقوم بتحميل الأوزان:

```
full_model.load_weights('resnet50.h5')
```

تقييم النموذج

يمكننا البدء بتقييم صورة واحدة (لاحظ أننا نستخدم الآن النموذج الذي تم حفظه وإعادة تحميله):

```
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input

img_path = 'dogs/dog.1.jpg'
img = image.load_img(img_path, target_size=(224,224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
print(full_model.predict(x))
plt.imshow(img)
```



تعطي شبكتنا العصبية هذه الصورة احتمالية هائلة بنسبة 99.993٪. هذا في الواقع ليس مفاجئًا للغاية: هذا كلب نموذجي، لذلك لا توجد صعوبة هنا.

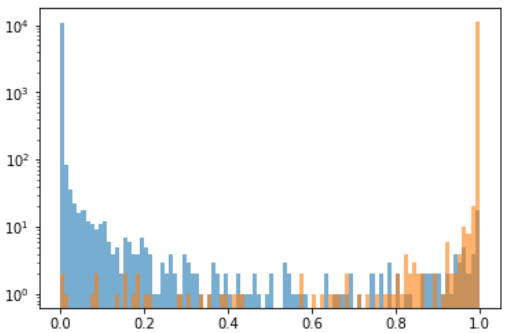
ولكن ماذا عن الصور الأخرىفي مجموعة البيانات؟ سيكون من الممتع للغاية النظر إلى الصور التي تم التعرف عليها بشكل خاطئ، لمعرفة ما يحدث معها. لنبدأ بتقييم النموذج لجميع الصور في مجموعة بيانات التدريب training dataset:

```
import sys

def true_and_predicted_labels(dataset):
    labels = np.zeros((dataset.n,2))
    preds = np.zeros_like(labels)
    for i in range(len(dataset)):
        sys.stdout.write('evaluating batch {}\r'.format(i))
        sys.stdout.flush()
        batch = dataset[i]
        batch_images = batch[0]
        batch_labels = batch[1]
        batch_preds = full_model.predict(batch_images)
        start = i*batch_size
        labels[start:start+batch_size] = batch_labels
        preds[start:start+batch_size] = batch_preds
    return labels, preds

train_labels, train_preds = true_and_predicted_labels(train_dataset)
```

الآن بعد أن أصبح لدينا تنبؤات النموذج، يمكننا توضيح كيف يمكن للنموذج فصل الفئتين. لهذا، سننظر فقطفي درجة القط، مع تذكر أن درجة الكلب تساوي واحداً مطروحًا منه درجة القط. وسنقوم برسم نقاط القط للفئتين. بالنسبة للقطط، نتوقع أن تكون درجة القط قريبة من واحدة. بالنسبة للكلاب، ستكون قريبة من الصفر.

```
def plot_cat_score(preds, labels, range=(0,1)):
    # get the cat score for all images
    cat_score = preds[:,1]
    # get the cat score for dogs
    # we use the true labels to select dog images
    dog cat score = cat score[labels[:,0]>0.5]
    # and for cats
    cat_cat_score = cat_score[labels[:,0]<0.5]
    # just some plotting parameters
    params = {'bins':100, 'range':range, 'alpha':0.6}
    plt.hist(dog_cat_score, **params)
    plt.hist(cat cat score, **params)
    plt.yscale('log')
```

```
plot_cat_score(train_preds, train_labels)
```



يرجى ملاحظة أنني استخدمت مقياسًا لوغاريتميًا على المحور y. لقد فعلت ذلك لأنه، مع دقة التصنيف الممتازة هذه، انتهى بنا المطاف بمعظم الصور التي تحتوي إما على درجة قريبة جدًا من 1 (قطط واضحة)، أو قريبة جدًا من 0 (كلاب شفافة)، مثل المثال الذي رسمناه أعلاه. سنرى هذه فقطفي مقياس خطي.في المنتصف نرى الحالات الأكثر صعوبة وإثارة للاهتمام.

يمكننا الآن حساب الدقة.

لهذا نحتاج إلى حساب التسميات المتوقعة ومقارنتها بالتسميات الحقيقية. لحساب التسميات المتوقعة، نأخذ درجة القط، ونقرر أن الشبكة تتوقع قطة إذا كانت هذه الدرجة أكبر من عتبة threshold معينة.

يوفر Keras تقديرًا للدقة أثناء التدريب. بالنسبة لهذا التقدير، تستخدم keras عتبة 0.5، لذلك دعونا نفعل ذلك أيضًا:

```
threshold = 0.5

def predicted_labels(preds, threshold):
    '''Turn predictions (floats in the last two dimensions)
    into labels (0 or 1).'''
    pred_labels = np.zeros_like(preds)
    # cat score lower than threshold: set dog label to 1
    # cat score higher than threshold: set dog label to 0
    pred_labels[:,0] = preds[:,1]<threshold
    # cat score higher than threshold: set cat label to 1
    # cat score lower than threshold: set cat label to 0
    pred_labels[:,1] = preds[:,1]>=threshold
    return pred_labels

train_pred_labels = predicted_labels(train_preds, threshold)
print('predicted labels:')
print(train_pred_labels)
print('true labels:')
print(train_labels)
```

```
predicted labels:
[[1. 0.]
 [1. 0.]
 [0. 1.]
 ...
 [1. 0.]
 [0. 1.]
 [1. 0.]]
true labels:
[[1. 0.]
 [1. 0.]
 [0. 1.]
 ...
 [1. 0.]
 [0. 1.]
 [1. 0.]]
```

نرى أن التسميات المتوقعة تبدو مشابهة جدًا للتسميات الحقيقية. هذا لأن الدقة قريبة من 100٪، وهناك أمثلة قليلة فقط معروضةفي النسخة المطبوعة أعلاه. دعنا نحدد جزء الأمثلة المصنفة بشكل خاطئ:

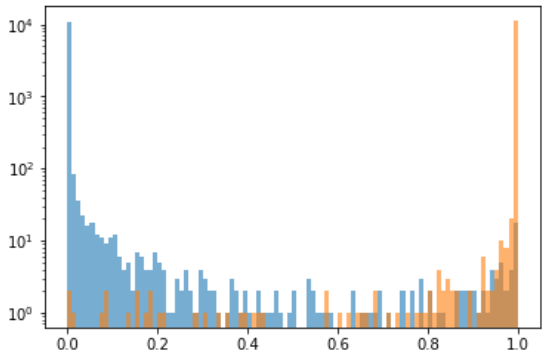
```
def misclassified(labels, pred_labels, print_report=True):
    def report(categ, n_misclassified, n_examples):
        print('{:<4} {:>3} misclassified samples ({:4.2f}%)'.format(
            categ,
            n_misclassified,
            100*(1-float(n_misclassified)/n_examples))
        )
    # total number of examples
    n_examples = len(labels)
    # total number of cats
    n_cats = sum(labels[:,0])
    # total number of dogs
    n_dogs = sum(labels[:,1])
    # boolean mask for misidentified examples
    mask_all = pred_labels[:,0] != labels[:,0]
    # boolean mask for misidentified cats
    mask_cats = np.logical_and(mask_all,labels[:,1]>0.5)
    # boolean mask for misidentified dogs
    mask_dogs = np.logical_and(mask_all,labels[:,1]<0.5)
    if print_report:
        report('all', sum(mask_all), n_examples)
        report('cats', sum(mask_cats), n_cats)
        report('dogs', sum(mask_dogs), n_dogs)
    return mask_all, mask_cats, mask_dogs

_ = misclassified(train_labels, train_pred_labels)
```

```
all  104 misclassified samples (99.54%)
cats  23 misclassified samples (99.80%)
dogs  81 misclassified samples (99.28%)
```

من خلال التدريب، نرى أن تصنيف الكلاب أكثر صعوبة. ومع ذلك، يمكن أن يؤدي كل تدريب إلى نتائج مختلفة. فعلت القليل، وحصلت على أداء تصنيف متماثل مرة واحدة فقط.

الآن، هل هناك سبب لاختيار عتبة عند 0.5 للتصنيف؟ دعنا نرسم نقاط القط مرة أخرى:

```
plot_cat_score(train_preds, train_labels)
```



من الواضح، مع عتبة 0.9 أو نحو ذلك، سنقوم فقط بتصنيف عدد قليل من القطط الأخرى، لكننا سنكتسب الكثير من الكلاب. بعد قليل من التحسين، وصلنا إلى تصنيف متماثل بحد أدنى قدره 0.85:

```
threshold = 0.85
train_pred_labels = predicted_labels(train_preds, threshold)
_ = misclassified(train_labels, train_pred_labels)
```

```
all  100 misclassified samples (99.56%)
cats  50 misclassified samples (99.56%)
dogs  50 misclassified samples (99.56%)
```

من خلال اختيار العتبة هذا، نقوم بتحسين دقة التصنيف العالمي بنسبة 0.02٪. هذا ليس مكسبًا كبيرًافي هذه الحالة، لكن تذكر أنه اعتمادًا على التدريب، قد تحصل على المزيد من التوزيعات غير المتكافئة. يمكنك محاولة إعادة تدريب ResNet50 مرة أخرى للتحقق من ذلك.

لذلك تذكر:

تحقق من درجة التصنيف وقم بضبط العتبة الخاصة بك بشكل صحيح.

الآن، قمنا بتحسين العتبة على مجموعة بيانات التدريب. ما الذي نحصل عليه من مجموعة بيانات التحقق؟

```
val_labels, val_preds = true_and_predicted_labels(val_dataset)
val_pred_labels = predicted_labels(val_preds, threshold)
_ = misclassified(val_labels, val_pred_labels)
```

```
all   35 misclassified samples (98.60%)
cats  17 misclassified samples (98.64%)
dogs  18 misclassified samples (98.56%)
```

هنا أيضًا، التصنيف متماثل إلى حد ما، ودقة التحقق من الصحة هي 98.6٪. مع عتبة تصنيف عند 0.5، سنحصل على:

```
val_pred_labels = predicted_labels(val_preds, 0.5)
_ = misclassified(val_labels, val_pred_labels)
```

```
all   38 misclassified samples (98.48%)
cats  12 misclassified samples (99.04%)
dogs  26 misclassified samples (97.92%)
```

النظر إلى الصور المصنفة بشكل خاطئ

توجد مشكلة بسيطةفي واجهة `ImageDataGenerator`: فهي لا تسمح لنا بالعثور على الصور التي تم تصنيفها بشكل خاطئ، حتى نتمكن من تحميلها من القرص والنظر إليها. لذلك نحن

بحاجة إلى إعادة تقييم الشبكة مرة أخرى، وتخزين الصور التي تم تحديدها بشكل خاطئ لعرضها لاحقًا.

```
import sys

dataset = val_dataset
misclassified_imgs = dict(dogs=[], cats=[])
for i in range(len(dataset)):
    if i%100:
        sys.stdout.write('evaluating batch {}\r'.format(i))
        sys.stdout.flush()
    batch = dataset[i]
    batch_images = batch[0]
    batch_labels = batch[1]
    batch_preds = full_model.predict(batch_images)
    batch_pred_labels = predicted_labels(batch_preds, threshold=0.85)
    mask_all, mask_cats, mask_dogs = misclassified(
        batch_labels,
        batch_pred_labels,
        print_report=False
    )
    misclassified_imgs['dogs'].extend(batch_images[mask_dogs])
    misclassified_imgs['cats'].extend(batch_images[mask_cats])
```

```
evaluating batch 83
```

فيما يلي عدد الصور المصنفة بشكل خاطئفي كل فئة:

```
print([(label, len(imgs)) for label,imgs in
misclassified_imgs.items()])
```

```
[('dogs', 14), ('cats', 18)]
```

لقد لاحظت بالتأكيد أن هذه الأرقام لا تتوافق تمامًا مع الأرقام التي رأيناها أعلاه (17 قطط مصنفة بشكل خاطئ و18 كلبًا تم تصنيفها بشكل خاطئ). أعتقد أن هذا قد يكون بسبب قدر من العشوائيةفي تقييم الشبكة، أو إلى الدقة العددية، وليس لدي أي فكرة من أين يأتي هذا.

لنكتب الآن دالة صغيرة لرسم مجموعة من الصور، حتى نتمكن من إلقاء نظرة على الصور المصنفة بشكل خاطئ:

```
def plot_images(imgs, i):
    ncols, nrows = (5, 2)
    start = i*ncols*nrows
    fig = plt.figure( figsize=(ncols*5, nrows*5), dpi=50)
    for i, img in enumerate(imgs[start:start+ncols*nrows]):
        plt.subplot(nrows, ncols, i+1)
        plt.imshow(img)
        plt.axis('off')
```

```
plot_images(misclassified_imgs['cats'],0)
```

واو ما هذا؟؟ الألوان فاسدة تمامًا ...

ويرجع ذلك إلى المعالجة المسبقة للصورة التي يقوم بها مكرر مجموعة البيانات. تذكر أننا نستخدم ResNet50 وقد طلبنا معالجة صورنا مسبقًا باستخدام `keras.applications.resnet50.preprocess_input`. هل لدينا طريقة للتراجع عن هذه العملية؟ لذلك نحتاج أولاً إلى إيجاد هذه الدالة لفهم ما تفعله.

نبدأ بالتحقق من إصدارنا من `keras_applications`:

```
import keras_applications
keras_applications.__version__
```

```
'1.0.8'
```

ثم ننظر إلى الكود المصدري لـ resnet50 ونرى أن هذه الدالة مأخوذة من `imagenet_utils`، حيث تم تعريفها هنا.

نحن ندعو الدالة دون أي حجة بصرف النظر عن الصورة إلى المعالجة المسبقة. لذلك نحنفي وضع "`caffe`". أيضًا، نوفر مصفوفة numpy للدالة، لذلك نحنفي الواقع نستدعي `_preprocess_numpy_input` ، هنا.

في وضع caffe، تقوم الدالة بما يلي:

- قم بالتبديل من تمثيل ألوان RGB إلى BGR.
- اطرح متوسط قيم BGR المحسوبة لمجموعة بيانات ImageNet بأكملها لتوسيط قيم BGR على 0.

يمكننا بسهولة كتابة دالة للتراجع عن هذه العملية:

```
def undo_preprocessing(x):
    mean = [103.939, 116.779, 123.68]
    x[..., 0] += mean[0]
    x[..., 1] += mean[1]
    x[..., 2] += mean[2]
    x = x[..., ::-1]
```

تضيف دالتنا أولاً متوسط قيم BGR مرة أخرى، نظرًا لأن هذه كانت آخر عملية للمعالجة المسبقة. ثم نعيد ترتيب مستويات اللون مرة أخرى. ما قد لا يكون واضحًا بالنسبة لك هو السطر الأخير، الذي يعيد ترتيب مستويات اللون. لذلك دعونا نلقي نظرةفي التفاصيل.

- التدوين notation ... يعني: إضافة العديد من الأبعاد حسب الضرورة. لذلك نترك الأبعاد الأولى للصور كما هي، للعمل فقط على البعد الأخير، أحد مستويات اللون (يرجى ملاحظة أنني أستخدم kerasفي الوضع الأخير للقنوات).
- يعمل التدوين `::-1` على البعد الأخير ، ويعيد ترتيب الأرقام هناك.

لنأخذ مثال بسيط. نبني مصفوفة بالشكل (2،2،3). يمكنك التفكيرفي الأمر كصورة 2 × 2 بكسل و3 مستويات لونية:

```
a = np.arange(12).reshape(2,2,3)
print(a)
```

```
[[[ 0  1  2]
  [ 3  4  5]]

 [[ 6  7  8]
  [ 9 10 11]]]
```

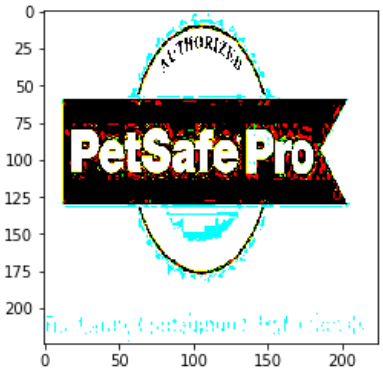
في الجزء العلوي الأيسر من البكسل، يتم تعيين مستويات الألوان الثلاثة على `[0 1 2]` على التوالي، وفي الجزء السفلي الأيمن من البكسل على `[9 10 11]` يمكننا أن نرى أن عملية الإرجاع لها التأثير المتوقع:

```
a[...,::-1]
```

```
array([[[ 2,  1,  0],
        [ 5,  4,  3]],

       [[ 8,  7,  6],
        [11, 10,  9]]])
```

لنجرب الآن دالة غير المعالجة الخاصة بنا على صورة واحدة:

```
img = misclassified_imgs['cats'][5]
plt.imshow(img)
```

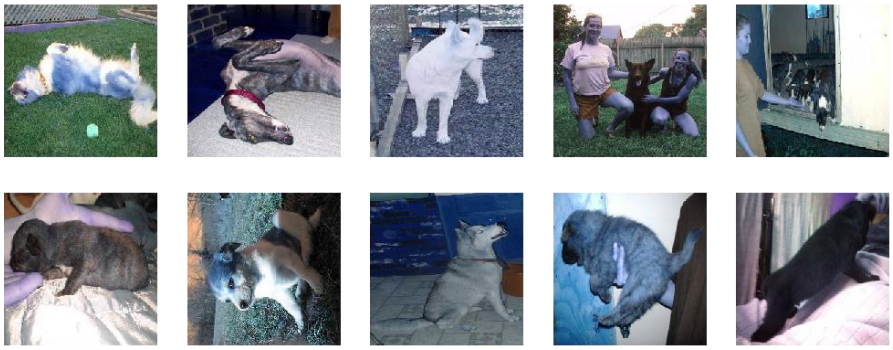



```
import copy
new_img = copy.copy(img)
undo_preprocessing(new_img)
plt.imshow(new_img.astype('int'))
```

أفضل بكثير! يبدو أن الدالة تعمل كما هو متوقع. لذلك نقوم بتعديل دالة الرسم لدينا لرسم صور غير معالجة:

```
def plot_images(imgs, i):
    ncols, nrows = (5, 2)
    start = i*ncols*nrows
    fig = plt.figure( figsize=(ncols*5, nrows*5), dpi=50)
    for i, img in enumerate(imgs[start:start+ncols*nrows]):
        img_unproc = copy.copy(img)
        undo_preprocessing(img_unproc)
        plt.subplot(nrows, ncols, i+1)
        plt.imshow(img_unproc.astype('int'))
        plt.axis('off')
```

```
plot_images(misclassified_imgs['dogs'],0)
```



في الواقع، لا تزال الألوان تبدو مضحكة بعض الشيء ... لا أعرف حقًا لماذا وأكون مصابًا بعمى الألوان، لست متأكدًا تمامًا! على أي حال، يمكننا على الأقل فهم الصور الآن.

يمكننا محاولة تخمين سبب سوء تصنيف هذه الصور. يمكن أن يكون الناس هم الذين يحملون القططفي كثير من الأحيان، مما قد يؤدي إلى تحيز الشبكة نحو إساءة تصنيف الكلاب عندما يكون الناسفي الصورة. أيضًا، ربما لا يساعد الاتجاه الخاطئ لصورة واحدة. بعد ذلك، غالبًا ما تنام القطط على ظهورها،في حين أن هذا الوضع ليس شائعًا للكلاب. أخيرًا، الجراء بالتأكيد أكثر صعوبة.

الخلاصة

في هذا المنشور، تعلمت ما هو نقل التعلم وكيفية:

- استخدم نماذج VGG16 وVGG19 وResNet50 المدربة مسبقًافي keras للوصول إلى دقة 98.5٪في تصنيف الكلاب مقابل القطط.
- إضافة طبقات إلى نموذج موجود لتكييفه مع احتياجاتك.
- احفظ نموذج keras بحيث يمكنك إعادة استخدامه لاحقًا، دون الحاجة إلى إعادة التدريب.
- قم بإنشاء أدواتك الخاصة لتقييم أداء النموذج وإلقاء نظرة على الصور التي تم التعرف عليها بشكل خاطئ، باستخدام python و numpy.

ماذا الان؟

يمكن للشبكات العصبية العميقة المدربة على مجموعة بيانات ImageNet الكبيرة رؤيتها! ويمكننا استخدامها لتصنيف الصورفي فئات مختلفة، وباستخدام مجموعة بيانات أصغر.

لنفترض أن لديك مجموعة الصور الخاصة بك التي ترغبفي تصنيفها. ربما تريد التعرف على الفطر؟ أم حشرات؟ أو معرفة ما إذا كانت الصورة الطبية مشبوهة؟ من المحتمل أن يكون نقل التعلم هو السبيل للذهاب.

بعد ذلك، سننشئ مجموعة بيانات من البداية من خلال جمع الصور على الإنترنت. باستخدام مجموعة البيانات هذه، سنتمكن من اختبار مهارات التعلم الخاصة بنا على فئات مختلفة ولكنها غير معروفة. بعد كل شيء، مميزات ImageNet للكلاب والقطط).

وسوف ندرس أيضًا نقل التعلمفي سياق معالجة اللغة الطبيعية.

المصدر:

https://thedatafrog.com/en/articles/image-recognition-transfer-learning/

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

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

حول مشروع تحديد سلالة الكلاب

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

مجموعة بيانات لمشروع تحديد سلالة الكلاب

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

معمارية النموذج

ما هو نقل التعلم؟

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

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

ما هو ResNet؟

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

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

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

متطلبات المشروع

يمكنك تثبيت جميع الوحدات النمطية لهذا المشروع باستخدام الأمر التالي:

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

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

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

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

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

هيكل المشروع

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

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

خطوات مشروع تحديد سلالة الكلاب:

الخطوة 1: استيراد المكتبات

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

الكود:

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

الخطوة 2: تحليل ملف مجموعة البيانات.

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

الكود:

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

ملف مجموعة البيانات:

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

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

الكود:

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

المخرجات:

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

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

الكود:

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

المخرجات:

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

tibetan_terrier , sussex_spaniel , standard_poodle , soft-coated_wheaten_terrier , siberian_husky

shetland_sheepdog , scottish_deerhound , schipperke , saluki , rottweiler

redbone , pomeranian , pekinese , otterhound , norwich_terrier

norfolk_terrier , miniature_schnauzer , miniature_pinscher , maltese_dog , malamute

leonberg , labrador_retriever , komondor , kelpie , japanese_spaniel

irish_wolfhound , irish_terrier , ibizan_hound , greater_swiss_mountain_dog , great_dane

golden_retriever , german_short-haired_pointer , french_bulldog , eskimo_dog , english_springer

english_foxhound , dingo , dandie_dinmont , collie , clumber

chihuahua , cardigan , bull_mastiff , briard , boxer

boston_bull , border_terrier , bluetick , blenheim_spaniel , bernese_mountain_dog

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

الخطوة 3: المعالجة المسبقة للبيانات

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

الكود:

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

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

الكود:

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

المخرجات:

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

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

الخطوة 4: ترميز البيانات وقياسها

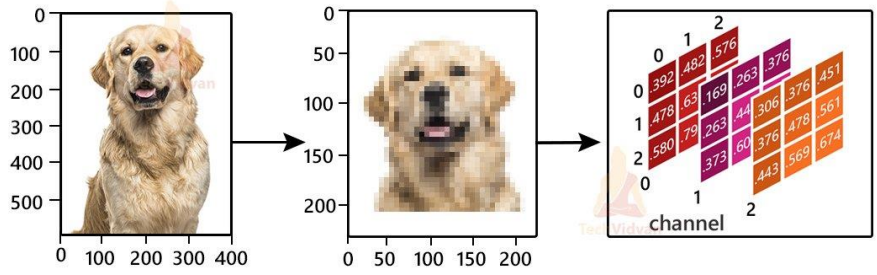
الآن أصبحت مجموعة البيانات الخاصة بنا جاهزة، سنقوم بإجراء عمليات على السجلات لأغراض التدريب والاختبار.

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

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

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

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



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

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

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

الكود:

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

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

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

الكود:

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

الخطوة 5: مجموعات التدريب والاختبار

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

الكود:

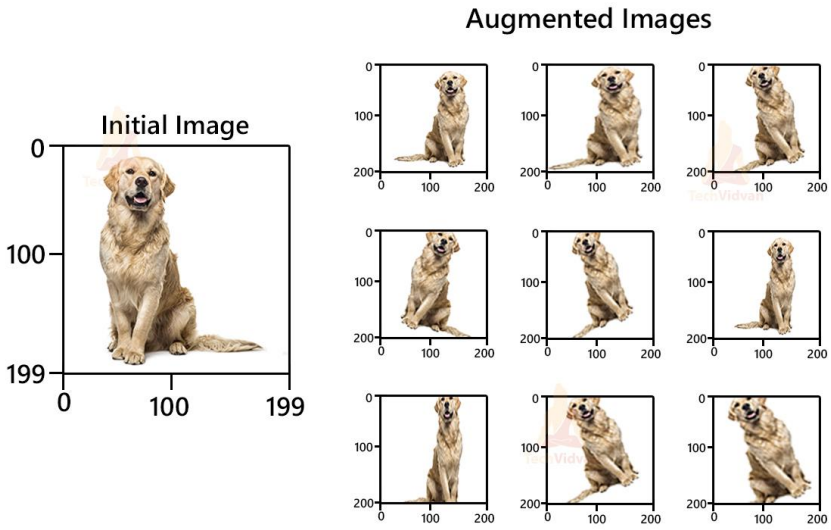
```
#split the dataset in the ratio of 80:20.
#80% for training and 20% for testing purpose
x_train, x_test, y_train, y_test =
train_test_split(train_x,train_y,test_size=0.2,random_state=42)
```

الخطوة 6: زيادة الصور

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

بعض تقنيات الزيادة الأكثر استخدامًا للصور هي:

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



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

الكود:

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

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

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

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

الخطوة 7: بناء النموذج

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

الكود:

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

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

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

الكود:

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

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

المخرجات:

تنزيل البيانات من:

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

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

الخطوة 7: تدريب النموذج

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

احفظ النموذج من أجل عملية التنبؤ.

الكود:

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

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

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

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

المخرجات:

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

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

الخطوة 8: التنبؤ

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

الكود:

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

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

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

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

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

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

مخرجات تحديد سلالة الكلاب

الملخص

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

المصدر:

https://techvidvan.com/tutorials/dog-breed-classification/

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

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

مجموعة البيانات والنماذج المستخدمة:

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

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

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

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

التنفيذ

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

```
import numpy as np

import matplotlib.pyplot as plt

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

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

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


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

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

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

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

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

base.summary()
```

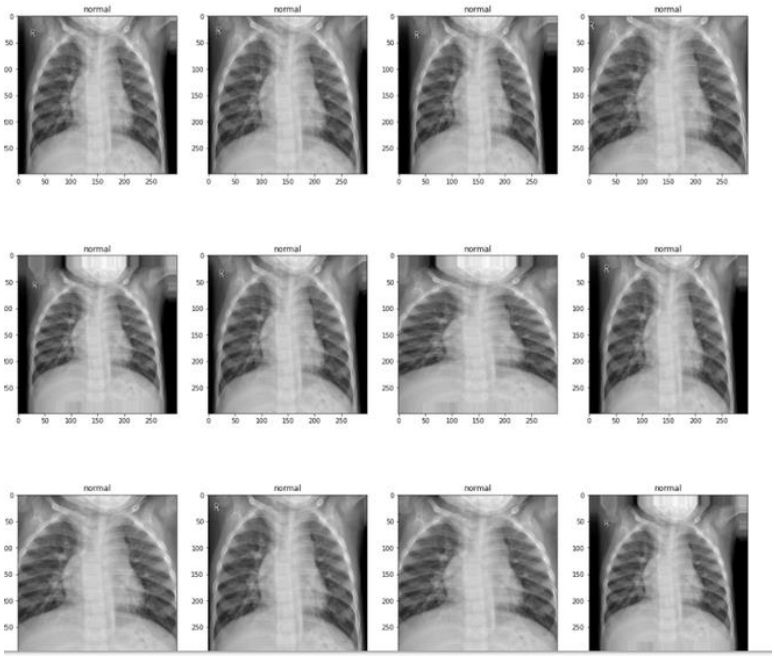
```
Downloading data from

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

xception_weights_tf_dim_ordering_tf_kernels_notop.h5

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

Model: "xception"

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

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

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

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

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

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

__________________________________________________________________________
_______________________
```

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

__________________________________________________________________________
__________________________

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

__________________________________________________________________________
__________________________

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

__________________________________________________________________________
__________________________

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

__________________________________________________________________________
__________________________

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

__________________________________________________________________________
__________________________

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

__________________________________________________________________________
__________________________

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

__________________________________________________________________________
__________________________

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

__________________________________________________________________________
__________________________

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

__________________________________________________________________________
__________________________

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

__________________________________________________________________________
__________________________

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


batch_normalization[0][0]
```

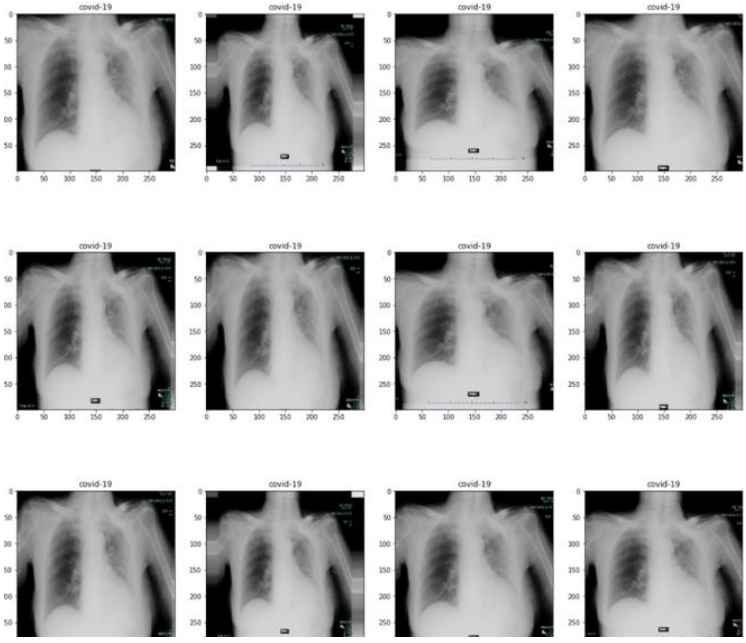
```
__________________________________________________________________________________
_____________________

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

__________________________________________________________________________________
_____________________

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

__________________________________________________________________________________
_____________________

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

__________________________________________________________________________________
_____________________

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

__________________________________________________________________________________
_____________________

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

__________________________________________________________________________________
_____________________

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

__________________________________________________________________________________
_____________________

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

__________________________________________________________________________________
_____________________

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

__________________________________________________________________________________
_____________________

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

__________________________________________________________________________________
_____________________

.............

(Trimmed model Summary)

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

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

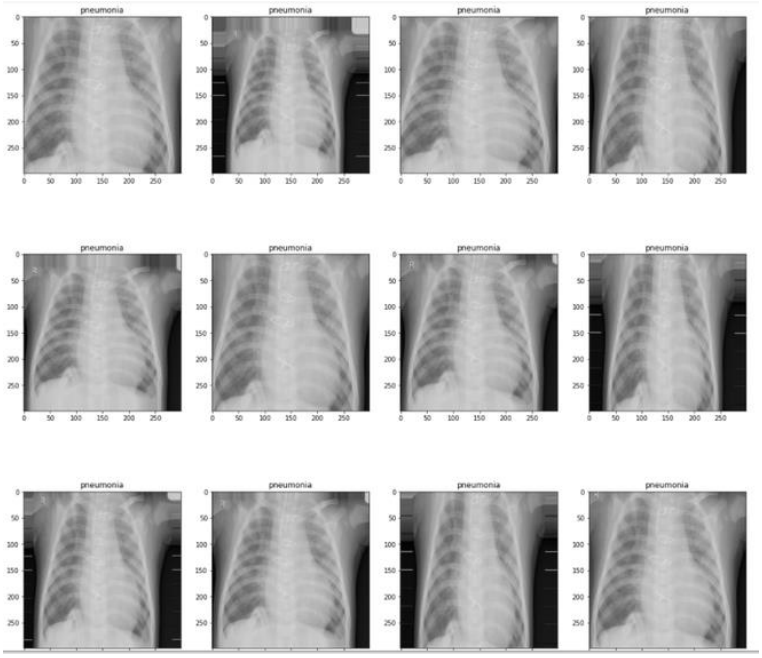
```
# Define augmentation
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        validation_split=0.25,
        horizontal_flip =True
        )

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

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

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

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

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



الأشعة السينية للرئتين الطبيعية (Normal Lungs X-ray)



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



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

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

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

```
Model: "sequential"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

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

```
xception (Functional)        (None, 10, 10, 2048)      20861480

_________________________________________________________________

dropout (Dropout)            (None, 10, 10, 2048)      0

_________________________________________________________________

flatten (Flatten)            (None, 204800)            0

_________________________________________________________________

dropout_1 (Dropout)          (None, 204800)            0

_________________________________________________________________

dense (Dense)                (None, 16)                3276816

_________________________________________________________________

dense_1 (Dense)              (None, 3)                 51

=================================================================

Total params: 24,138,347

Trainable params: 3,276,867

Non-trainable params: 20,861,480

_________________________________________________________________
```

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

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

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

```
Epoch 1/30

137/137 [==============================] - 121s 886ms/step -

loss: 5.7757 - accuracy: 0.8528 - val_loss: 3.4022 - val_accuracy: 0.8966

Epoch 2/30

137/137 [==============================] - 119s 867ms/step -

loss: 3.3137 - accuracy: 0.9028 - val_loss: 2.0748 - val_accuracy: 0.9228

Epoch 3/30

137/137 [==============================] - 119s 866ms/step -
```

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

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

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

الملخص

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

المصدر:

1. https://www.geeksforgeeks.org/pneumonia-detection-using-deep-learning/.

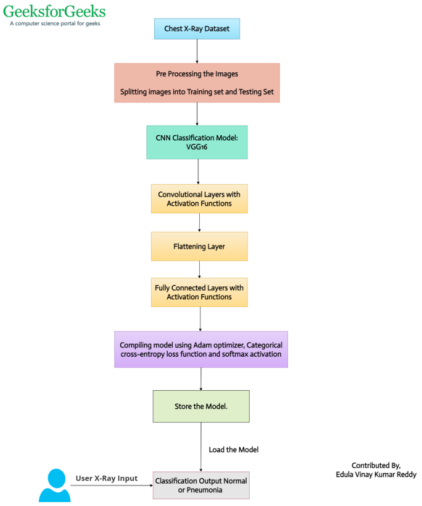
8) كشف الالتهاب الرئوي باستخدام التعلم العميق Pneumonia Detection using Deep Learning

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 النموذجفي كل مرة نقدم فيها مدخلات.

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

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

```
from keras_preprocessing import image
```

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

التنفيذ الكامل

Pneumonia.py:

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

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

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

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

Test.py:

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

المخرجات:

يتم عرض النتائج في الفيديو أدناه:

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

المصدر:

https://www.geeksforgeeks.org/pneumonia-detection-using-deep-learning/

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 البُنى 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

- تنزيل 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>
```

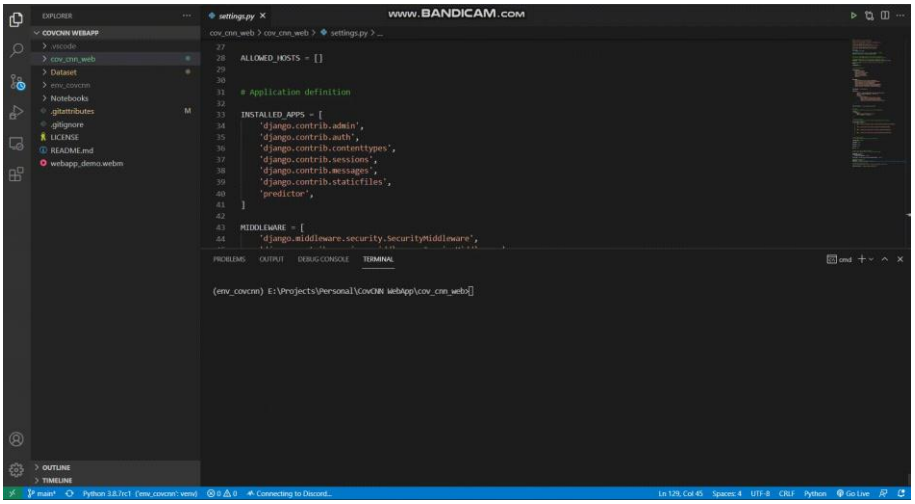
- داخل مجلد views.py، تعامل مع الصورة التي تم تحميلها. قم بتحميل ملفات النموذج وأرسل الرد مرة أخرى إلى الواجهة الأمامية.
- سيحتوي الرد على التفاصيل التالية:
  - التنبؤ النموذجي Model Prediction
  - درجات الثقة Confidence Score

- 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



المصدر:

https://www.geeksforgeeks.org/detecting-covid-19-from-chest-x-ray-images-using-cnn/

10) نظام الكشف عن الأشكال ثلاثية الأبعاد باستخدام التعلم العميق
3-D Shape Detection System using Deep learning

تعد تقنيات اكتشاف الأشكال Shape detection techniques جانبًا مهمًا من الرؤية الحاسوبية وتُستخدم لتحويل بيانات الصورة الأولية إلى تمثيلات رمزية مطلوبة للتعرف على الكائن والموقع.

في هذه المقالة، يتم تقديم نوتبوك يحتوي على تطوير نظام يكتشف أربعة أنواع من الأشكال ثلاثية الأبعاد: مكعب Cube، واسطوانة Cylinder، وشبه كروي Spheroid، وكروي Sphere.

النموذج المستخدم مبني على mobilenet v1، مع الاستفادة من فوائد نقل التعلم transfer learning من أجل بناء نموذج خفيف الوزن ولكن دقيق لشبكة CNN.

يتم تنفيذه على منصة Cainvas، التي توفر تنفيذًا سلسًا لأجهزة الكمبيوتر المحمولة من نوع بايثون لبناء أنظمة ذكاء اصطناعي يمكن نشرهافي النهاية على edge (أي نظام مضمن مثل MCUs المدمجة).

يمكن العثور على النوتبوك هنا.

Mobilenet v1 - النموذج الأساسي

تعتمد شبكات MobileNets على بُنية مبسطة تستخدم تلافيف قابلة للفصل بعمق لبناء شبكات عصبية عميقة خفيفة الوزن.

الغرض من استخدام mobilenet لحالة الاستخدام هذه هو أن هذا المشروع يهدف إلى نشره على الأجهزة المحمولة على edge، ومن ثم يكون منطقيًا تمامًا لبناء نموذج يعتمد على فئة من النماذج الفعالة (MobileNets) التي تم تدريبها مسبقًا على نشر مجموعة من نماذج DNN دقيقة الضبط لتطبيقات الرؤية المتنقلة والمدمجة.

تحميل MobileNet:

```
base_model=MobileNet(input_shape=(IMAGE_SIZE, IMAGE_SIZE,3), alpha =
ALPHA,
                     depth_multiplier = 1, dropout = 0.001,
include_top = False,
                     weights = "imagenet", classes = 4,
backend=keras.backend,

layers=keras.layers,models=keras.models,utils=keras.utils)
```

هنا يتم تعيين معلمات النموذج على النحو التالي:

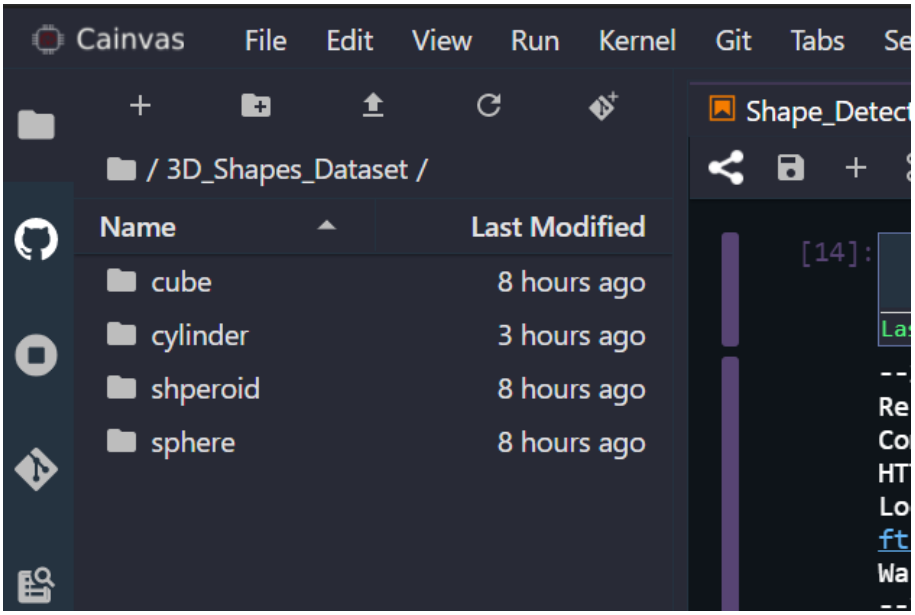
IMAGE_SIZE = 224

ALPHA = 0.75

EPOCHS=20

مجموعة بيانات الأشكال ثلاثية الأبعاد

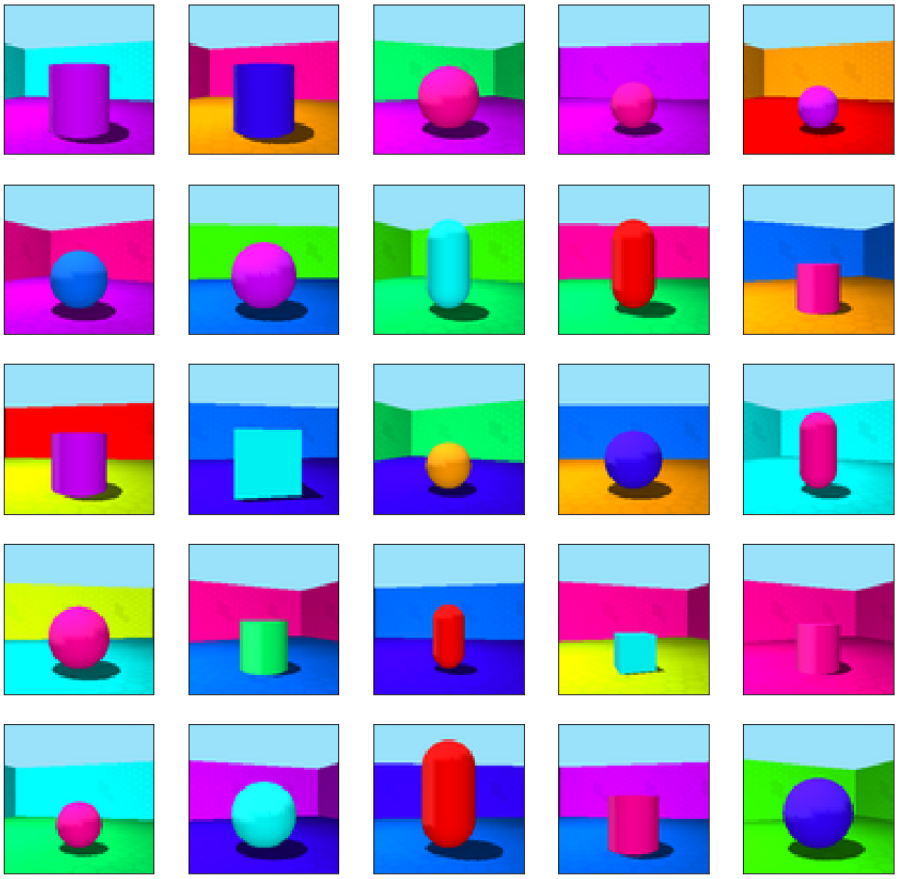
مجموعة البيانات المستخدمة هنا عبارة عن مجموعة بيانات مستخرجة مخصصة مع صور بحجم (224،224). يتكون من 4 أدلة تحتوي على الصور المقابلة لفئات الأشكال الأربعة.



تتم معالجة جميع الصور المستخدمةفي التدريب والاختبار مسبقًا على النحو التالي:

```
def prepare_image(file):
    img = image.load_img(img_path + file, target_size=(IMAGE_SIZE,
IMAGE_SIZE))
    img_array = image.img_to_array(img)
    img_array_expanded_dims = np.expand_dims(img_array, axis=0)
    return
keras.applications.mobilenet.preprocess_input(img_array_expanded_dims)
```

رسم عينة من مجموعة بيانات التدريب:



بناء النموذج – نقل التعلم

```
def build_finetune_model(base_model, dropout, fc_layers, num_classes):
    for layer in base_model.layers:
        layer.trainable = False

    x = base model.output
    x = GlobalAveragePooling2D()(x)

    for fc in fc_layers:
        # New FC layer, random init
        x = Dense(fc, activation='relu')(x)
        x = Dropout(dropout)(x)

    # New softmax layer
    predictions = Dense(num_classes, activation='softmax')(x)

    finetune model = Model(inputs=base model.input,
outputs=predictions)
```

```
    return finetune_model

FC_LAYERS = [100, 50]
dropout = 0.5

finetune_model = build_finetune_model(base_model,
                                      dropout=dropout,
                                      fc_layers=FC_LAYERS,
                                      num_classes=4)
```

النموذج المراد يضبط بشكل دقيق fine-tune مبني عن طريق إضافة بضع طبقات إضافية إلى نموذج mobilenet الأساسي.

هنا، نضيف طبقتين كثيفتين متصلتين بالكامل من 100 و 50 خلية عصبية على التوالي مع دالة تنشيط "relu" ونسبة تسرب 0.5، إلى آخر طبقة من mobileenet، وطبقة إخراج نهائية للتنبؤات– وهي طبقة كثيفة أخرى بها 4 خلايا عصبية ناتجة ودالة تنشيط "softmax". (كل خلية عصبية تتوافق مع فئة إخراج من الأشكال).

تدريب النموذج - الضبط الدقيق

الآن بعد أن تم بناء نموذج تعلم النقل الخاص بنا، يمكننا تدريبه (fine-tune) على مجموعة البيانات المذكورة سابقًا باستخدام keras ImageDataGenerator لمعالجة الصور بشكل أكبر حتى تكون مناسبة لنموذج mobienet الخاص بنا، وبالتالي إنشاء مولد تدريب. (الكود الموضح أدناه):

```
train_datagen=ImageDataGenerator(preprocessing_function=preprocess_inp
ut)

train_generator=train_datagen.flow_from_directory('3D_Shapes_Dataset',

target_size=(IMAGE_SIZE,IMAGE_SIZE),
                                                  color_mode='rgb',
                                                  batch_size=32,

class_mode='categorical', shuffle=True)
```

يتم الآن تجميع نموذج CNN الذي تم إنشاؤه مسبقًا باستخدام مُحسِّن آدم، وخطأ categorical crossentropy ومقاييسفي الاعتبار أثناء التدريب هو دقة النموذج.

يتم بعد ذلك إدخال مولد التدريب المحددفي النموذج الذي تم تجميعه كما هو موضحفي الكود أدناه.

```
finetune_model.summary()
finetune_model.compile(optimizer='Adam',loss='categorical_crossentropy
',metrics=['accuracy'])
step_size_train=train_generator.n//train_generator.batch_size
history =
finetune_model.fit_generator(generator=train_generator,steps_per_epoch
=step_size_train,epochs=EPOCHS, shuffle=True)

finetune_model.save('shape_model.h5')
```

يمكن عرض ملخص النموذج على أنه الإخراج قبل بدء التدريب داخل النوتبوك.

أخيرًا، يتم حفظ النموذج بعد اكتمال التدريب كنموذج keras (.h5).

اختبار النموذج

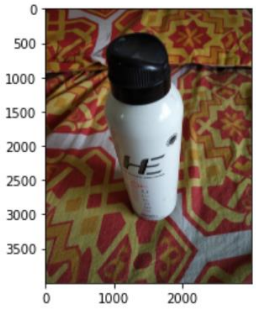
يحقق النموذج دقة تصل إلى 99٪، ومنذ تصنيفه للشكل الهندسي فقط للكائن، فإنه لا يصلح حتىفي مثل هذه المستويات العالية من الدقة.

تم اختبار النموذج على كائنات من العالم الحقيقي بالإضافة إلى صور الإنترنت لفهم قدراته بشكل أفضل.

فيما يلي النتائج:

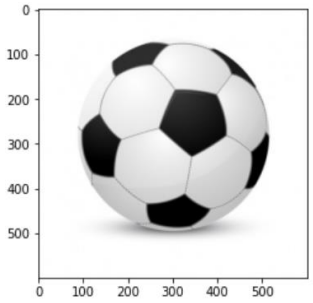
```
In [55]: # Real-Life example
         img_path="sample1.jpg"
         predict_shape(img_path)

         Shape Detected:  Cube
```


```
In [56]: # Real-Life example 2
         img_path="sample2.jpg"
         predict_shape(img_path)

         Shape Detected:  Spheroid
```



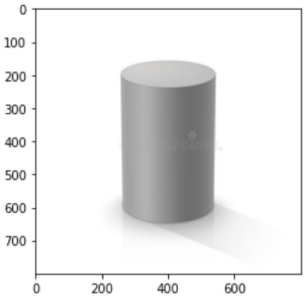
```
In [59]: # Real-Life example 3
         img_path="sample3.jpg"
         predict_shape(img_path)
```

```
Shape Detected:  Sphere
```




```
In [64]: # Real-Life example 4
         img_path="sample4.jpg"
         predict_shape(img_path)
```

```
Shape Detected:  Cylinder
```



الملخص

يمكن استخدام نظام الكشف عن الأشكال ثلاثية الأبعاد لتصنيف الكائنات حتى في الوقت الفعلي. من بين التطورات الإضافية في هذا المشروع تحويل نموذج keras CNN إلى نموذج الحد الأدنى من edge القابلة للنشر مثل tflite. أو onnx. من أجل نشر هذا على وحدة نمطية مدمجة AIoT / MCU مثل OpenMV Cam أو Raspberry Pi. هذا النشر ممكن من خلال منصة Cainvas من خلال الاستفادة من مترجمهم المسمى deepC. وبالتالي يخرج الذكاء الاصطناعي بشكل فعال على edge ـ في حالات استخدام العالم الواقعي والمادي.

المصدر:

https://medium.com/ai-techsystems/3d-shape-detection-system-d7e34286a2b1

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 دوال 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 رسم بعض الصور التي تم توفيرها لنا لبناء المصنف لكل فئة

```
path = '/lung_colon_image_set/lung_image_sets'
classes = os.listdir(path)
classes
```

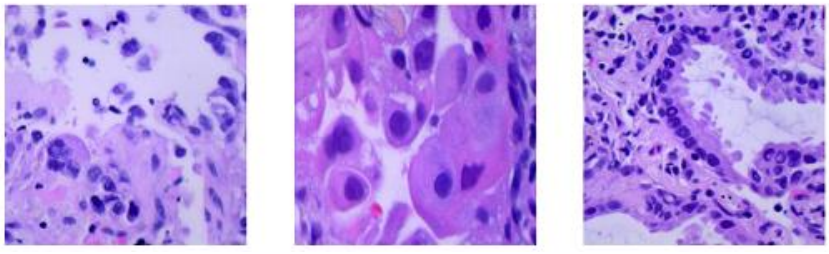
المخرجات:

```
['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()])
```

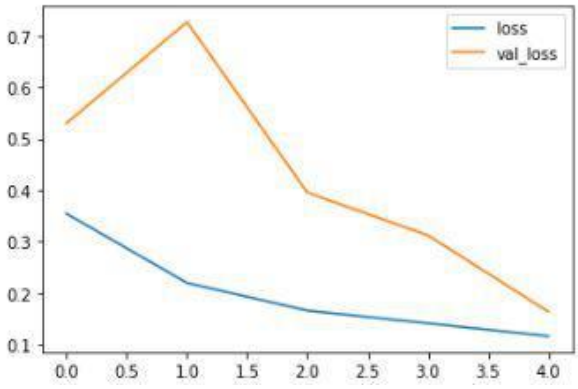
المخرجات:

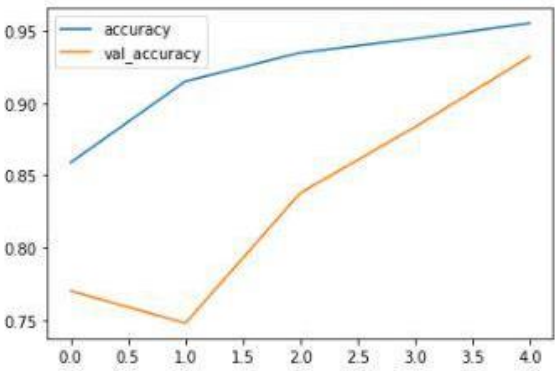
```
Epoch 1/10
188/188 [==============================] - 52s 189ms/step - loss: 0.3543 - accuracy: 0.8587 - val_loss: 0.5301 - val_accuracy: 0.7700 - lr: 0.0010
Epoch 2/10
188/188 [==============================] - 32s 169ms/step - loss: 0.2196 - accuracy: 0.9149 - val_loss: 0.7268 - val_accuracy: 0.7477 - 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.

المصدر:

https://www.geeksforgeeks.org/lung-cancer-detection-using-transfer-learning/

12)الكشف عما إذا كان الشخص يرتدي قناعا أو لا باستخدام CNN
Detecting If a Person is Wearing a Mask or Not Using CNN

المقدمة

في هذه المقالة، سننشئ مصنف Mask vs No Mask باستخدام CNN ومصنفات التعلم الآلي. سيكتشف ما إذا كان الشخص يرتدي قناعًا للوجه أم لا. سنتعلم كل شيء من البداية، وسأشرح كل خطوة. أحتاج إلى فهمك الأساسي لتعلم الآلة وعلوم البيانات. لقد قمت بتطبيقه على جهاز Windows 10 المحلي الخاص بي، ولكن إذا كنت ترغبفي ذلك، يمكنك أيضًا تنفيذه على Google Colab.

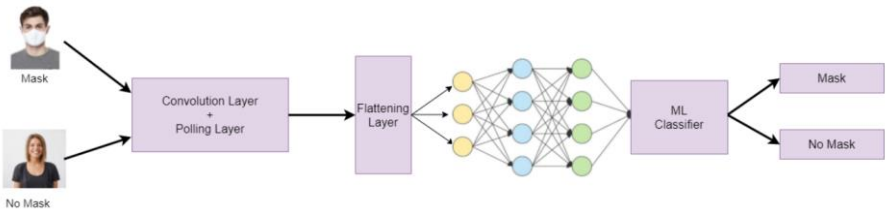
الشبكة العصبية التلافيفية (CNN) هي نوع من الشبكات العصبية الاصطناعية المصممة لمعالجة بيانات البكسل. يتم استخدامها بشكل صريحفي معالجة الصور Image Processing والتعرف على الصور Image Recognition.

Fig.(a) With Mask

Fig.(b) Without Mask

مسار نموذج CNN

أولاً، سنقوم بإدخال صور RGB بحجم 224 × 224 بكسل. ثم ستنتقل هذه الصور إلى نموذج CNN الذي سيستخرج 128 متجهًا للميزات ذات الصلة منها. ثم سنستخدم متجهات الميزات feature vectors هذه لتدريب مصنفات التعلم الآلي المتنوعة لدينا، مثل الانحدار اللوجستي Logistic Regression، والغابة العشوائية Random Forest، وما إلى ذلك، لتصنيف ما إذا كان الشخصفي تلك الصورة يرتدي قناعًا أم لا. يمكنك الرجوع إلى الرسم البياني أدناه لفهم أفضل.



عمل الكود لتدريب نموذج CNN

في هذا القسم، سنتعرف على جزء الكود. سنناقش تحميل مجموعة البيانات ومعالجتها مسبقًا، وتدريب نموذج CNN، واستخراج متجهات الميزات لتدريب مصنفات التعلم الآلي.

استيراد المكتبات الضرورية:

سنقوم باستيراد جميع المكتبات اللازمة التي نحتاجها لهذا المشروع.

سنستخدم مكتبات مثل Numpy، والتي تُستخدم لإجراء حسابات رياضية معقدة. تقوم Pandas بتحميل مجموعة البيانات ومعالجتها مسبقًا، ويتم استخدام العديد من المكتبات الأخرى.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

from itertools import cycle

from sklearn.model_selection import train_test_split

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dropout, Dense, AveragePooling2D,
Flatten ,Dense, Input

from sklearn.metrics import classification_report, confusion_matrix

import cv2
```

```
from sklearn.metrics import roc_curve, auc

from sklearn.preprocessing import label_binarize

from scipy import interp

from sklearn.ensemble import RandomForestClassifier

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.optimizers import Adam
```

التحميل والمعالجة المسبقة لمجموعة البيانات

يمكنك تنزيل مجموعة البيانات من GitHub Repo.

استنسخ مجموعة البيانات الخاصة بك من المستودع أعلاه.

تحتوي مجموعة البيانات هذه على أكثر من 1200 صورة لأشخاص مختلفين يرتدون قناع وجه أم لا. بعد تحميل مجموعة البيانات، سنقوم بمعالجتها مسبقًا. يتضمن تقسيم البيانات إلى مجموعات بيانات التدريب train والاختبار test، وتحويل قيم البكسل بين 0 إلى 1، وتحويل التسميات labels إلى تسميات مشفرة واحدة ساخنة one-hot encoded labels.

يوجد أدناه كود تحميل مجموعة البيانات ومعالجتها مسبقًا. تم التعليق عليه جيدًا حتى تتمكن من فهمه بسهولة.

Train Images	Test Images
688 (50%)	688 (50%)

في الكود أدناه، سنقرأ أولاً جميع الصور من المجلد ثم نخزنهافي مصفوفة عن طريق تغيير حجمها إلى 224 × 224 بكسل. بعد ذلك، سنقوم بتسمية هذه الصور. تحتوي الصور التي تحتوي على أقنعة على تسمية 0، والصور التي لا تحتوي على أقنعة لها تسمية 1. أخيرًا، سنقسم مجموعة البيانات هذه إلى تدريب واختبار باستخدام دالة sklearn المسماة train test split.

قائمة بجميع الصور مع قناع من الدليل الرئيسي.

```
filenames = os.listdir("observations-
master/experiements/data/with_mask")

np.random.shuffle(filenames)

print(filenames) # Read all the images from that directory and resize
them into 224x224 pixels.

with_mask_data = [cv2.resize(cv2.imread("observations-
master/experiements/data/with_mask/"+img), (224,224)) for img in
filenames]

print(len(with_mask_data))
```

يتم تنفيذ الخطوة المماثلة أعلاه للصور التي لا تحتوي على قناع.

```
filenames = os.listdir("observations-
master/experiements/data/without_mask")

np.random.shuffle(filenames)

print(filenames)
```

```
without_mask_data = [cv2.resize(cv2.imread("observations-
master/experiements/data/without_mask/"+img), (224,224)) for img in
filenames]

print(len(without_mask_data))
```

لقد قمنا بدمج كلتا المصفوفتين لإنشاء مصفوفة واحدة، وتحويل كل قيمة بكسل بين 0 و 1 بقسمتها على 255.

```
data = np.array(with_mask_data +
without_mask_data).astype('float32')/255 # Label of the image with a
mask - 0 # Label of the image without mask - 1

labels = np.array([0]*len(with_mask_data) +
[1]*len(without mask data))

print(data.shape) # Splitting the data into training and testing sets.

(training_data, testing_data, training_label, testing_label) =
train_test_split(data, labels, test_size=0.50, stratify=labels,
random_state=42)
```

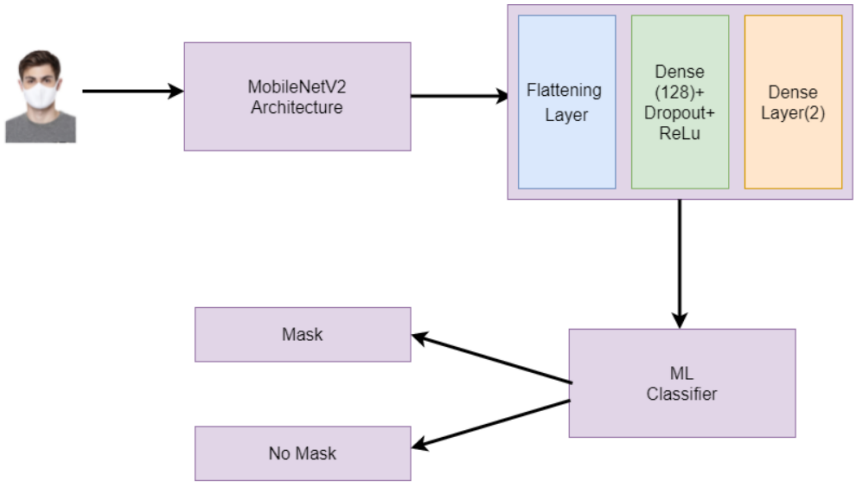
```
print(training_data.shape) # Function to Plot the Accuracy/Loss Curves

def plot_acc_loss(result, epochs):

acc = result.history['accuracy']

loss = result.history['loss']

val_acc = result.history['val_accuracy']

val_loss = result.history['val_loss']

plt.figure(figsize=(15, 5))

plt.subplot(121)

plt.plot(range(1,epochs), acc[1:], label='Train_acc')

plt.plot(range(1,epochs), val_acc[1:], label='Val_acc')

plt.title('Accuracy over ' + str(epochs) + ' Epochs', size=15)

plt.legend()

plt.grid(True)

plt.subplot(122)

plt.plot(range(1,epochs), loss[1:], label='Train_loss')

plt.plot(range(1,epochs), val_loss[1:], label='Val_loss')

plt.title('Loss over ' + str(epochs) + ' Epochs', size=15)

plt.legend()

plt.grid(True)

plt.show()
```

بناء الشبكة العصبية التلافيفية (CNN)

الآن سنقوم ببناء شبكتنا العصبية التلافيفية. أولاً، استخدمنا مولد بيانات الصورة image data generator لزيادة عدد الصورفي مجموعة البيانات الخاصة بنا. سينشئ مولد الصور هذا المزيد من الصور من هذه الصور الموجودة. يقوم ببعض التدويرفي اتجاه عقارب الساعة أو عكس اتجاه عقارب الساعة، وتغيير التباين، وإجراء تكبير أو تصغير، إلخ.

بعد ذلك، سوف نستخدم معمارية MobileNetV2 المدربة مسبقًا لتدريب نموذجنا. إنه نموذج نقل العلم transfer learning. نقل التعلم هو عندما يتم استخدام النماذج المدربة مسبقًا pre-trained models لتدريب نماذج التعلم العميق الجديدة، أي إذا قام نموذجان بأداء مهام مماثلة، فيمكننا مشاركة المعرفة. بعد تطبيق نقل التعلم، سنقوم بتطبيق طبقة تسطيح flattening layer لتحويل المصفوفة ثنائية الأبعاد إلى صفيف أحادي الأبعاد. بعد ذلك، سنقوم بتطبيق الطبقات الكثيفة Dense layers والطبقات المتسربة dropout layers لإجراء التصنيف.

أخيرًا، سنقوم بتدريب نموذجنا من خلال أخذ حجم الدُفعة batch size على أنه 32 وعدد الفترات epochs على أنه 25. يمكنك أخذ أي قيم أخرى وفقًا لقوتك الحسابية.



كود لتدريب نموذج الشبكة العصبية التلافيفية:

سنقوم ببناء نقل التعلم لمعمارية MobileNetV2، وهو نموذج CNN مدرب مسبقًا. أولاً، سننشئ المزيد من الصور من مجموعة البيانات الخاصة بنا باستخدام أداة Image Data Generator. بعد ذلك، سنقوم بتعيين معلماتنا الفائقة مثل معدل التعلم learning rate، حجم الدفعة batch size، عدد الفترات no. of epochs، وما إلى ذلك، ثم أخيرًا، سنقوم بتدريب نموذجنا والتحقق من دقتهفي مجموعة الاختبار.

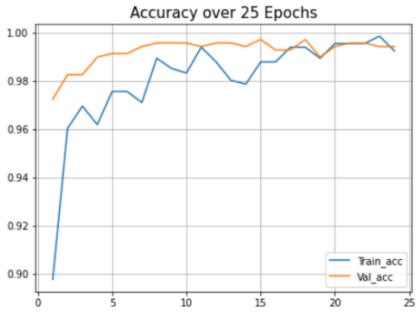
```
# Image data generator to generate more images.

generator = ImageDataGenerator(

rotation range=20,

zoom_range=0.15,

width_shift_range=0.2,

height_shift_range=0.2,

shear_range=0.15,

horizontal_flip=True,

fill_mode="nearest")
```

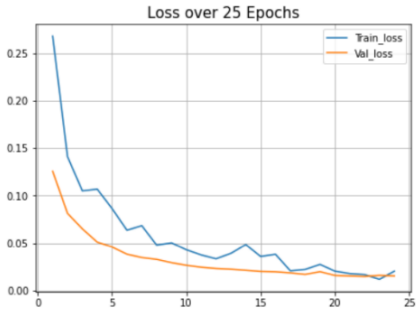
```
# Setting the hyperparameters.
```

```
learning_rate = 0.0001

epoch = 25

batch_size = 32

# Training the mobile net v2 architecture.
```

```
transfer_learning_model = MobileNetV2(weights="imagenet",
include_top=False,

input_tensor=Input(shape=(224, 224, 3)))



model_main = transfer_learning_model.output

model_main = AveragePooling2D(pool_size=(7, 7))(model_main) # Applying
the flattening layer.

model_main = Flatten(name="flatten")(model_main)

model_main = Dense(128, activation="relu",
name="dense_layer")(model_main)

model_main = Dropout(0.5)(model_main)

model_main = Dense(2, activation="softmax")(model_main)

cnn = Model(inputs=transfer_learning_model.input, outputs=model_main)

for row in transfer_learning_model.layers:

row.trainable = False



optimizer = Adam(lr=learning_rate, decay=learning_rate / epoch)
```

```
cnn.compile(loss="sparse_categorical_crossentropy",
optimizer=optimizer,

metrics=["accuracy"])

# Train the CNN model

history = cnn.fit(

generator.flow(training_data, training_label, batch_size=batch_size),

steps_per_epoch=len(training_data) // batch_size,

validation_data=(testing_data, testing_label),

validation_steps=len(testing_data) // batch_size,

epochs=epoch)


Evaluate the trained model
# Evaluate the model on the test set.

cnn.evaluate(testing_data, testing_label)

plot_acc_loss(history, 25)
```





حصلنا على دقة 99.42٪ في مجموعة الاختبار

استخدام مصنفات التعلم الآلي

الآن، سنقوم باستخراج 128 متجهًا للميزات ذات الصلة من نموذج CNN الذي تم تدريبه مسبقًا وتطبيقها على مصنفات التعلم الآلي مختلفة. سنستخدم مصنفات التعلم الآلي التالية:

تعزيز التدرج Xtreme:

Extreme Gradient Boosting (XGBoost) هي مكتبة مفتوحة المصدر تنفذ بكفاءة وفعالية خوارزمية تعزيز التدرج gradient boosting. أولاً، قم باستيراد المكتبات الضرورية ثم حدد المصنف كـ XGBClassifier. بعد تركيبه، قم بتمثيل التنبؤات ودرجات الدقة. نحصل

على الدقة accuracy ومصفوفة الارتباك confusion matrix وتقرير التصنيف classification report كإخراج. هنا، حصلنا على 98.98٪ من دقتنا.

مصنف الغابة العشوائي:

الغابة العشوائية Random Forest عبارة عن مصنف يحتوي على العديد من أشجار القرار decision trees في مجموعات فرعية مختلفة من مجموعة البيانات المحددة ويأخذ المتوسط لتحسين الدقة التنبؤية لمجموعة البيانات هذه. يؤدي العدد الأكثر أهمية من الأشجار في الغابة إلى دقة أعلى ويمنع مشكلة الضبط الزائد overfitting. أولاً، قم باستيراد المكتبات الضرورية ثم حدد المصنف كـ RandomForestClassifier. بعد تركيبه، قم بتمثيل التنبؤات ودرجات الدقة. نحصل على الدقة ومصفوفة الارتباك وتقرير التصنيف كإخراج. هنا، حصلنا على 99.41٪ كدقة لدينا، وهي أكثر من XGBoost.

الانحدار اللوجستي:

الانحدار اللوجستي Logistic regression هو خوارزمية تصنيف تعلم خاضعة للإشراف supervised learning classification تستخدم للتنبؤ باحتمالية متغير مستهدف. طبيعة المتغير المستهدف أو التابع هي ثنائية التفرع، مما يعني أنه سيكون هناك فئتان محتملتان فقط. هنا، حصلنا على 99.70٪ كدقة لدينا، وهي أكثر من XGBoost ولكنها أقل بقليل من الغابة العشوائية.

التوزيع الغاوسي:

التوزيع الاحتمالي المتماثل حول المتوسط هو التوزيع الطبيعي normal distribution، ويسمى أحيانًا التوزيع الغاوسي Gaussian distribution. يوضح أن البيانات القريبة من المتوسط تحدث بشكل متكرر أكثر من البيانات البعيدة عن المتوسط. يمثل منحنى الجرس التوزيع الطبيعي على الرسم البياني.

يوجد أدناه كود استخراج متجهات الميزات الأساسية ووضع متجهات الميزات هذه في مصنفات التعلم الآلي.

بعد تدريب نموذج CNN الخاص بنا، سنقوم الآن بتطبيق استخراج الميزات واستخراج 128 متجهًا للميزات ذات الصلة من هذه الصور. ويتم إدخال متجهات الميزات المناسبة هذه في مصنفات التعلم الآلي المتنوعة لدينا لإجراء التصنيف النهائي.

لقد استخدمنا العديد من نماذج التعلم الآلي مثل XGBoost و Random Forest و Logistic Regression و GaussianNB وما إلى ذلك. سنختار النموذج الذي يمنحنا أفضل دقة.

```
from keras.models import Model

layer_name='dense_layer' # Extracting the layer from the above CNN
model, which contains 128neurons.

new_model = Model(inputs=cnn.input,

outputs=cnn.get_layer(layer_name).output)



new model.summary()

# Get new training data that only contains these 128 features.

training_image_features = new_model.predict(training_data)

training_image_features = pd.DataFrame(data=training_image_features)



testing_image_features = new_model.predict(testing_data)

testing_image_features = pd.DataFrame(data=testing_image_features)



# Perform the classification using XGBoost Classifier. from xgboost
import XGBClassifier

from sklearn.metrics import accuracy_score

classifier = XGBClassifier()

classifier.fit(training_image_features, training_label)

predictions = classifier.predict(testing_image_features) # Getting the
accuracy score

accuracy = accuracy_score(predictions, testing_label)

print(f'{accuracy*100}%') # Getting the confusion matrix.

cf = confusion_matrix(predictions, testing_label)

print(cf)

from sklearn.metrics import classification_report

c_r = classification_report(predictions, testing_label,
output dict=True)

print(c_r)

# Perform the classification using RandomForest Classifier.

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
```

```
rfc.fit(training_image_features, training_label)

prediction = rfc.predict(testing_image_features)

accuracy = accuracy score(prediction, testing label)

print(f'{accuracy*100}%')

cf = confusion_matrix(prediction, testing_label)

print(cf)

from sklearn.metrics import classification_report

c_r = classification_report(prediction, testing_label,
output_dict=True)

print(c_r)

# Perform the classification using LogisticRegression

from sklearn.linear_model import LogisticRegression

lin_r = LogisticRegression()

lin_r.fit(training_image_features, training_label)

prediction = lin_r.predict(testing_image_features)

accuracy = accuracy_score(prediction, testing_label)

print(f'{accuracy*100}%')

cf = confusion_matrix(prediction, testing_label)

print(cf)

from sklearn.metrics import classification_report

c_r = classification_report(prediction, testing_label,
output_dict=True)

print(c_r)

# Perform the classification using GaussianNB

from sklearn.naive_bayes import GaussianNB

n_b = GaussianNB()

n_b.fit(training_image_features, training_label)

prediction = n_b.predict(testing_image_features)

accuracy = accuracy_score(prediction, testing_label)

print(f'{accuracy*100}%')

cf = confusion_matrix(prediction, testing_label)

print(cf)

from sklearn.metrics import classification_report
```

```
c_r = classification_report(prediction, testing_label,
output_dict=True)

print(c_r)
```

النتائج

في هذا القسم، سنناقش نتائج تصنيفنا. سنناقش مقدار الدقة accuracy التي حققناها وما هي الدقة precision والاستدعاء recall و f1-score. إذا كنت ترغبفي معرفة المزيد عن نتائج الأداء هذه، فهناك مقال جميل يمكنك الرجوع إليه.

الدقة Accuracy: معلمة واحدة لتقييم نماذج التصنيف هي الدقة. تُعرف النسبة المئوية للتنبؤات التي تنبأ بها نموذجنا بشكل صحيح بالدقة. فيما يلي التعريف الرسمي للدقة: عدد التخمينات الدقيقة يساوي مقدار دقة التخمينات بشكل عام.

الدقة Precision: يتم حساب الدقة بقسمة العدد الإجمالي للتنبؤات الإيجابية على نسبة الإيجابيات الحقيقية (أي عدد الإيجابيات الحقيقية بالإضافة إلى عدد الإيجابيات الخاطئة).

F1 Score: تعتبر درجة F1 واحدة من أهم مقاييس التقييمفي التعلم الآلي. الجمع بين الدقة Precision والاستدعاء recall، وهما مقياسان عادة ما يكونانفي المنافسة، ويلخصان بأناقة قدرة التنبؤ للنموذج.

فيما يلي نتائج الأداء لجميع مصنفات التعلم الآلي التي استخدمناها لتدريب نموذجنا. يعطي الانحدار اللوجستي أعلى دقة وهي 99.709٪.

Method	XGB Classifier	Random Forest	Logistic Regression	GaussianNB
Accuracy	99.127	99.273	99.418	99.2732
Precision	99.129	99.2736	99.420	99.277
Recall	99.127	99.273	99.4186	99.2732
F1-Score	99.127	99.2732	99.418	99.2732

مصفوفة الارتباك لجميع مصنفات التعلم الآلي هي:

مصفوفة الارتباك هي مصفوفة NxN تلخص النتائج المتوقعة. يحتوي على عدد التنبؤات الصحيحة وغير الصحيحة التي تكسرها كل فئة.

	TP	TN
FP	341	1
FN	4	342

GaussianNB

	TP	TN
FP	342	1
FN	3	342

Logistic Regression

	TP	TN
FP	342	2
FN	3	341

Random Forest

	TP	TN
FP	341	2
FN	4	341

XGB Classifier

الكود الكامل

في هذا القسم، قمت بمشاركة الكود الكامل المستخدمفي هذا المشروع. بالإضافة إلى الكود أعلاه، يحتوي هذا الكود أيضًا على رمز لرسم منحنيات ROC-AUC لنموذج التعلم الآلي الخاص بك.

منحنى ROC (منحنى خاصية تشغيل المستقبل) هو رسم بياني يوضح أداء نموذج التصنيففي جميع عتبات التصنيف. يرسم هذا المنحنى معاملين: المعدل الإيجابي الحقيقي True Positive Rate. المعدل الإيجابي الكاذب False Positive Rate.

أولاً قمنا بتحميل مجموعة البيانات. ثم نقرأ الصور باستخدام مكتبة OpenCV ونخزنهافي مصفوفة عن طريق تحويلها إلى أحجام 224 × 224 بكسل. بعد ذلك، يتعين علينا عمل تسميات لكلا الفئتين، أي قناع وبدون قناع. ثم ناقشنا كود Image Data Generator ومعمارية MobileNetV2. علاوة على ذلك، قمنا بتدريب نموذج CNN الخاص بنا بعد ضبط المعلمات الفائقة مثل الفترات، وحجم الدُفعة، وما إلى ذلك. وبعد الانتهاء من 25 فترة، حصلنا على دقة 99.42٪في مجموعة الاختبار.

بعد تدريب نموذج CNN، طبقنا استخراج الميزات واستخرجنا 128 متجهًا للميزات من الطبقة الكثيفة وطبقنا متجهات الميزات هذه على نموذج التعلم الآلي للحصول على التصنيف النهائي. ثم قمنا بكتابة الكود لتقييم مصفوفات الأداء المختلفة مثل Accuracyو F1-Score و Precision وما إلى ذلك. وأخيرًا، قمنا برسم منحنى ROC-AUC لنموذج التعلم الآلي الأفضل أداءً.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

from itertools import cycle

from sklearn.model_selection import train_test_split

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dropout, Dense, AveragePooling2D,
Flatten ,Dense, Input

from sklearn.metrics import classification_report, confusion_matrix

import cv2



from sklearn.metrics import roc_curve, auc

from sklearn.preprocessing import label_binarize

from scipy import interp

from sklearn.ensemble import RandomForestClassifier

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.optimizers import Adam
```

```
def plot_acc_loss(result, epochs):

acc = result.history['accuracy']

loss = result.history['loss']

val_acc = result.history['val_accuracy']

val_loss = result.history['val_loss']

plt.figure(figsize=(15, 5))

plt.subplot(121)

plt.plot(range(1,epochs), acc[1:], label='Train_acc')

plt.plot(range(1,epochs), val_acc[1:], label='Val_acc')

plt.title('Accuracy over ' + str(epochs) + ' Epochs', size=15)

plt.legend()

plt.grid(True)

plt.subplot(122)

plt.plot(range(1,epochs), loss[1:], label='Train_loss')

plt.plot(range(1,epochs), val_loss[1:], label='Val_loss')

plt.title('Loss over ' + str(epochs) + ' Epochs', size=15)

plt.legend()

plt.grid(True)

plt.show()



# filenames = glob(mypath +'with_mask/'+'*.jpg')

filenames = os.listdir("observations-
master/experiements/data/with_mask")

np.random.shuffle(filenames)

print(filenames) # 460 , 116

with_mask_data = [cv2.resize(cv2.imread("observations-
master/experiements/data/with_mask/"+img), (224,224)) for img in
filenames]

print(len(with_mask_data))



filenames = os.listdir("observations-
master/experiements/data/without_mask")
```

```
np.random.shuffle(filenames)

print(filenames) # 460 , 116

without_mask_data = [cv2.resize(cv2.imread("observations-
master/experiements/data/without_mask/"+img), (224,224)) for img in
filenames]

print(len(without_mask_data))



data = np.array(with_mask_data +
without_mask_data).astype('float32')/255

labels = np.array([0]*len(with_mask_data) +
[1]*len(without_mask_data))

print(data.shape)

(training_data, testing_data, training_label, testing_label) =
train_test_split(data, labels, test_size=0.50, stratify=labels,
random_state=42)

print(training_data.shape)



generator = ImageDataGenerator(

rotation_range=20,

zoom_range=0.15,

width_shift_range=0.2,

height_shift_range=0.2,

shear_range=0.15,

horizontal_flip=True,

fill_mode="nearest")



learning_rate = 0.0001

epoch = 25

batch size = 32



transfer_learning_model = MobileNetV2(weights="imagenet",
include_top=False,

input_tensor=Input(shape=(224, 224, 3)))
```

```
model_main = transfer_learning_model.output

model_main = AveragePooling2D(pool_size=(7, 7))(model_main)

model_main = Flatten(name="flatten")(model_main)

model_main = Dense(128, activation="relu",
name="dense_layer")(model_main)

model_main = Dropout(0.5)(model_main)

model_main = Dense(2, activation="softmax")(model_main)

cnn = Model(inputs=transfer_learning_model.input, outputs=model_main)

for row in transfer_learning_model.layers:

row.trainable = False



optimizer = Adam(lr=learning_rate, decay=learning_rate / epoch)

cnn.compile(loss="sparse_categorical_crossentropy",
optimizer=optimizer,

metrics=["accuracy"])



history = cnn.fit(

generator.flow(training_data, training_label, batch_size=batch_size),

steps_per_epoch=len(training_data) // batch_size,

validation_data=(testing_data, testing_label),

validation_steps=len(testing_data) // batch_size,

epochs=epoch)



cnn.evaluate(testing_data, testing_label)



plot_acc_loss(history, 25)



from keras.models import Model

layer_name='dense_layer'
```

```
new_model = Model(inputs=cnn.input,
outputs=cnn.get_layer(layer_name).output)


new_model.summary()


training_image_features = new_model.predict(training_data)
training_image_features = pd.DataFrame(data=training_image_features)


testing_image_features = new_model.predict(testing_data)
testing_image_features = pd.DataFrame(data=testing_image_features)


from xgboost import XGBClassifier


from sklearn.metrics import accuracy_score
classifier = XGBClassifier()
classifier.fit(training_image_features, training_label)
predictions = classifier.predict(testing_image_features)
accuracy = accuracy_score(predictions, testing_label)
print(f'{accuracy*100}%')
cf = confusion_matrix(predictions, testing_label)
print(cf)
from sklearn.metrics import classification_report
c_r = classification_report(predictions, testing_label,
output_dict=True)
print(c r)


from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(training_image_features, training_label)
```

```
prediction = rfc.predict(testing_image_features)

accuracy = accuracy_score(prediction, testing_label)

print(f'{accuracy*100}%')

cf = confusion_matrix(prediction, testing_label)

print(cf)

from sklearn.metrics import classification_report

c_r = classification_report(prediction, testing_label,
output_dict=True)

print(c_r)



from sklearn.linear_model import LogisticRegression

lin_r = LogisticRegression()

lin_r.fit(training_image_features, training_label)

prediction = lin_r.predict(testing_image_features)

accuracy = accuracy_score(prediction, testing_label)

print(f'{accuracy*100}%')

cf = confusion_matrix(prediction, testing_label)

print(cf)

from sklearn.metrics import classification_report

c_r = classification_report(prediction, testing_label,
output_dict=True)

print(c_r)



from sklearn.naive_bayes import GaussianNB

n_b = GaussianNB()

n_b.fit(training_image_features, training_label)

prediction = n_b.predict(testing_image_features)

accuracy = accuracy_score(prediction, testing_label)

print(f'{accuracy*100}%')

cf = confusion_matrix(prediction, testing_label)

print(cf)
```

```
from sklearn.metrics import classification_report

c_r = classification_report(prediction, testing_label,
output_dict=True)

print(c_r)



# Binarize the output

y = label binarize(training label, classes=[0, 1])

y_test = label_binarize(testing_label, classes=[0, 1])

n_classes = 2



# Learn to predict each class against the other

classifier = RandomForestClassifier()

classifier.fit(training_image_features, y)

y_score = classifier.predict(testing_image_features)



print(accuracy_score(y_score, y_test))



# Compute the ROC curve and ROC area for each class

fpr = dict()

tpr = dict()

roc_auc = dict()

for i in range(n_classes):

fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])

roc_auc[i] = auc(fpr[i], tpr[i])



# Compute micro-average ROC curve and ROC area

fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(),
y_score.ravel())

roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

```
# First aggregate all false positive rates

all_fpr = np.unique(np.concatenate([fpr[i] for i in
range(n classes)]))



# Then interpolate all ROC curves at these points

mean_tpr = np.zeros_like(all_fpr)

for i in range(n_classes):

mean_tpr += interp(all_fpr, fpr[i], tpr[i])



# Finally, average it and compute AUC

mean_tpr /= n_classes



fpr["macro"] = all_fpr

tpr["macro"] = mean_tpr

roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])



# Plot all ROC curves

plt.figure()

plt.plot(fpr["micro"], tpr["micro"],

label='micro-average ROC curve (area = {0:0.2f})'

''.format(roc_auc["micro"]),

color='deeppink', linestyle=':', linewidth=4)



plt.plot(fpr["macro"], tpr["macro"],

label='macro-average ROC curve (area = {0:0.2f})'

''.format(roc_auc["macro"]),

color='navy', linestyle=':', linewidth=4)



colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
```

```
for i, color in zip(range(n_classes), colors):

plt.plot(fpr[i], tpr[i], color=color,

label='ROC curve of class {0} (area = {1:0.2f})'

''.format(i, roc_auc[i]))



plt.plot([0, 1], [0, 1], 'k--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Some extension of Receiver operating characteristic to
multi-class')

plt.legend(loc="lower right")

plt.show()
```

الإعدادات التجريبية المستخدمة:

قمنا بتطبيق نظام التصنيف المقترح للتصنيف باستخدام لغة برمجة Python 3.8 مع معالج IntelR Core i5-1155G7 CPU @ 2.30GHz × 8 وذاكرة وصول عشوائي (RAM) بسعة 8 جيجا بايت تعمل على Windows 10 مع NVIDIA Geforce MX 350 مع رسومات بسعة 2 جيجا بايت.

الاستنتاج

في هذا العمل، قدمنا استخدام الشبكات التلافيفية CNN ومصنفات التعلم الآلي لتصنيف القناع وعدم وجود قناع بشكل فعال. استخدمنا أيضًا زيادة الصورةفي مجموعة البيانات الخاصة بنا لتسوية الصور. بعد استخراج ميزة الصورة من خلال CNN، يتم تطبيق خوارزميات التعلم الآلي للتصنيف النهائي مما يؤدي إلى الحصول على أفضل نتيجة حصلت عليها الشبكات العصبية التلافيفية بدقة 99.42٪ و99.21٪ للغابات العشوائية و99.70٪ للانحدار اللوجستي، وهو الأعلى بين الجميع. لذلك، يمكن أن يكون هذا النهج الخاص بالصور وتقنيات معالجة الصور طريقة تصنيف ضخمة وأسرع وفعالية من حيث التكلفة. يمكن أن يؤدي التدريب باستخدام مجموعات بيانات ضخمة والاختبارفي الميدان مع مجموعة أكبر إلى تحسين الدقة.

النقاط الرئيسية لهذا المقال:

1. لقد ناقشنا مصنفات CNN والتعلم الآلي.

2. بعد ذلك، قفزنا إلى جزء الكود وناقشنا تحميل مجموعة البيانات ومعالجتها مسبقًا.

3. علاوة على ذلك، قمنا بتدريب نموذج CNN ثم ناقشنا دقة الاختبار والتحقق من الصحة.

4. بعد ذلك، قمنا باستخراج متجهات الميزات ووضعناهافي مصنفات التعلم الآلي.

5. أخيرًا، لقد انتهينا من هذه المقالة.

المصدر

https://www.analyticsvidhya.com/blog/2022/10/detecting-if-a-person-/is-wearing-a-mask-or-not-using-cnn

13) نقل التعلم باستخدام TensorFlow Transfer learning with TensorFlow

مقدمة موجزة لنقل التعلم

ترتبط أكثر المشكلات انتشارًافي التعلم الآلي بالبيانات: فقد تكون إما غير كافية أو منخفضة الجودة. أحد الحلول الواضحة لهذه المجموعة من المشاكل هو الحصول على بيانات أكثر وأفضل. ومع ذلك، فإن هذين الاثنين لا يجتمعانفي كثير من الأحيان. علينا أن نضحي بالجودة مقابل الكمية أو العكس. لحسن الحظ، هناك حل أكثر ابتكارًا: نقل التعلم transfer learning.

نقل التعلم هو طريقة لإعادة استخدام نموذج مدرب بالفعل لمهمة أخرى. تسمى خطوة التدريب الأصلية التدريب المسبق pre-training. الفكرة العامة هي أن التدريب المسبق "يعلم teaches" النموذج ميزات أكثر عمومية،في حين أن مرحلة التدريب الأخيرة "تعلم teaches" أنها تتميز ببيانات خاصة بنا (محدودة).

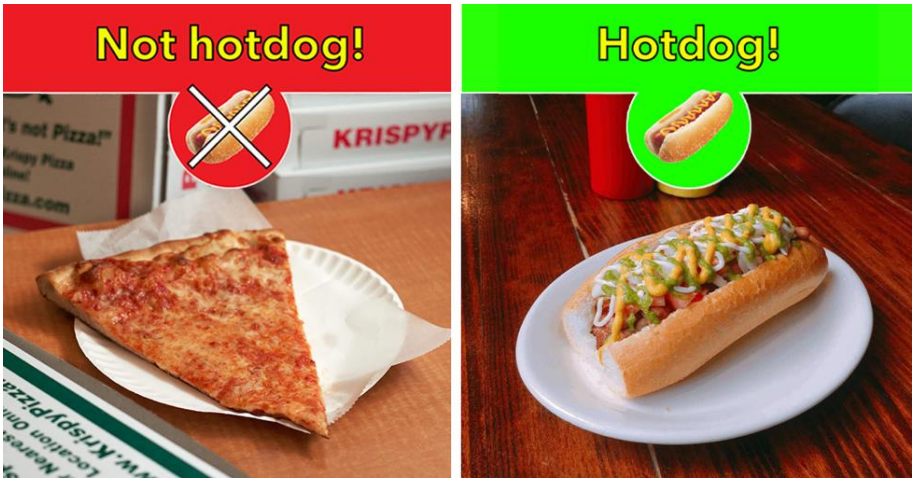
نقل التعلم مفيد بشكل خاصفي مجالات مثل الطب حيث يظل نقص البيانات مشكلة دائمة. أثبتت العديد من نماذج CNN المدربة مسبقًا على بيانات ImageNet نجاحهافي مهام طبية مختلفة. كل ما يتطلبه الأمر هو بضعة أسطر من التعليمات البرمجية لنقلها إلى البيانات الطبية.

في هذه المقالة، سوف نتعلم كيفية القيام بذلك باستخدام TensorFlow، منصة التعلم العميق الأكثر استخدامًافي العالم (اعتبارًا من عام 2021). قبل الخوضفي البرمجة، دعنا نحصل على ملخص سريع لـ TensorFlow و Keras API التي تشغله.

TensorFlow عبارة عن نظام أساسي شامل يتيح بناء نماذج التعلم الآلي ونشرها. نحن مهتمون فقط ببناء النماذج، وليس نشرها، ولهذا، نحتاج إلى استخدام Keras. Keras عبارة عن واجهة برمجة تطبيقات مصممة "للبشر ، وليس الآلات" ، على حد تعبير أنفسهم. بمعنى، تم تصميم Keras للمبرمجين مثلنا الذين يرغبونفي إنشاء نماذج مخصصة. تركيبته البسيطة وسهلة التذكر تجعله شبه إدمان.

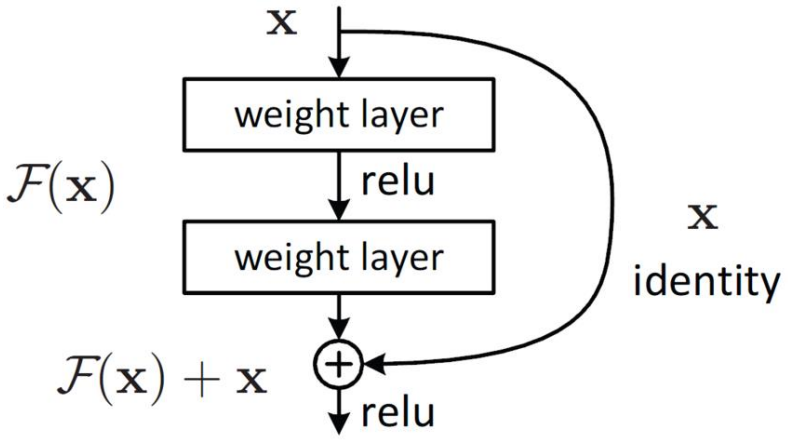
بينما يتوفر Keras API كمكتبة بايثون قائمة بذاتها، إلا أنه متاح أيضًا كجزء من مكتبة TensorFlow. يوصى باستخدام tensorflow.keras عبر Keras نفسها ، حيث يتم صيانتها بواسطة فريق TensorFlow ، مما يضمن الاتساق مع وحدات TensorFlow الأخرى.

دراسة حالة: تصنيف الصور الثنائية



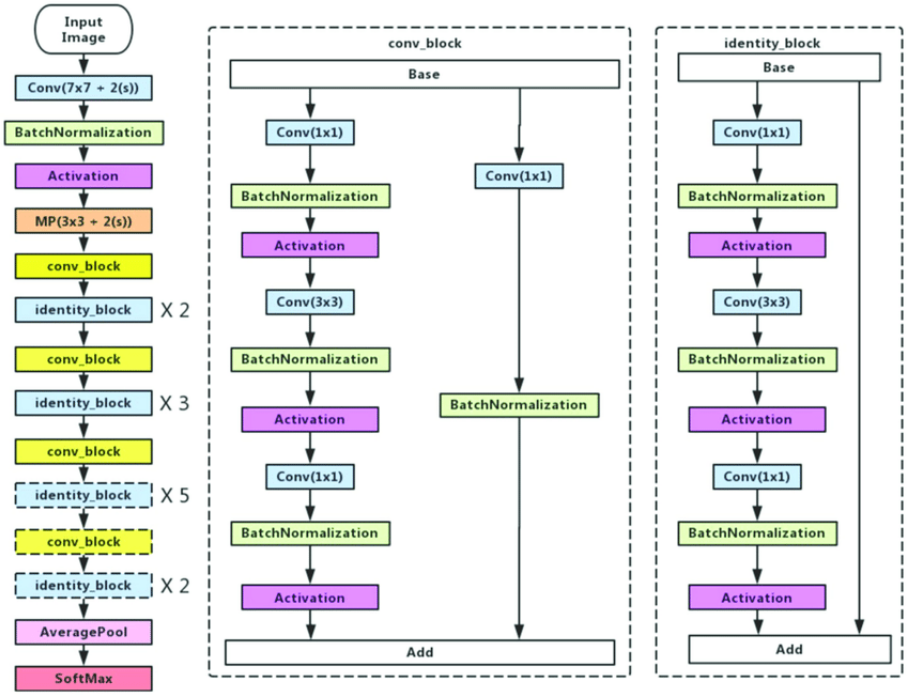
كمثال أول، سنحاول تصنيف الصور الثنائية binary image classification. ستكون مجموعة البيانات الخاصة بنا هي Hot Dog – Not Hot Dog من Kaggle وسنحاول أن نتنبأ – لقد خمنت ذلك – ما إذا كانت الصورة المقدمة عبارة عن نقانق أم لا.

لهذا، سوف نستخدم نموذج ResNet50 الذي تم تدريبه مسبقًا على مجموعة بيانات ImageNet. يشير ResNet إلى مجموعة من البُنى التي تستخدم التوصيلات المتبقية لحل مشكلة التدهور degradation problem – أي تدهور الدقة.



الشكل أعلاه يصور المطابقة المتبقية residual mapping. يتخطى هذا الاتصال طبقة واحدة (أو أكثر) وينفذ تعيين الهوية identity mapping، F (x) + x. حقق هذا التعديل الطفيف في بنية الشبكة نجاحًا هائلاً ضد مشكلة التدهور. نتيجة لذلك، يمكن أن تصل بُنى ResNet إلى

عمق 1000 طبقة. اختيارنا المحدد للنموذج، ResNet50 هو مثال ضحل نسبيًا. يمكنك أن ترى بُنيتها الشاملةفي الشكل التالي:



هناك بدائل لعائلة ResNet: أثبتت شبكات MobileNets وInception وما إلى ذلك نجاحها أيضًافي تصنيف الصور. يمكنك أيضًا اختيار واحدة من هذه الشبكات، أو شبكة مختلفة تمامًا وإجراء نقل التعلم على ذلك.

سأعمل على Google Colab، والذي أوصي به لأي شخص لا يقوم جهاز الكمبيوتر الخاص به بالمهمة، على الرغم من أنه ليس مطلبًا صارمًا. يمكنك تشغيل الكودفي أي بيئة تختارها، بمافي ذلك Jupyter Notebook أو PyChram.

فلنستعرض العملية خطوة بخطوة.

إعداد البيئة لنقل التعلم باستخدام TensorFlow

ملاحظة: قد تختلف هذه الخطوة حسب بيئتك المفضلة.

```
# Upload the kaggle API key
from google.colab import files
files.upload()
```

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

```
# Install the kaggle package
! pip install -q Kaggle
```

```
# Download the dataset from Kaggle
! kaggle datasets download -d dansbecker/hot-dog-not-hot-dog
```

```
# Import the necessary packages
import tensorflow as tf
from tensorflow import keras
from PIL import Image
import os
import numpy as np
```

تحميل البيانات من أجل نقل التعلم باستخدام TensorFlow

```
# Unzip the downloaded zip file
!unzip /content/hot-dog-not-hot-dog.zip
```

```
# Let's check size of images
for image in list(os.walk("/content/train/not_hot_dog"))[0][2]:
  a = Image.open(f"/content/train/not_hot_dog/{image}")
  print(np.asarray(a).shape)
```

```
(512, 512, 3)
(512, 384, 3)
(512, 512, 3)
(436, 512, 3)
(382, 512, 3)
(512, 382, 3)
(512, 382, 3)
(512, 382, 3)
(512, 512, 3)
(384, 512, 3)
(384, 512, 3)
(512, 384, 3)
(384, 512, 3)
(384, 512, 3)
(288, 512, 3)
(512, 382, 3)
(512, 382, 3)
(384, 512, 3)
(512, 374, 3)
(512, 289, 3)
(384, 512, 3)
(382, 512, 3)
(512, 384, 3)
(512, 512, 3)
(511, 512, 3)
(512, 382, 3)
(512, 382, 3)
(512, 382, 3)
```

هذا ليس سوى جزء من المخرجات، ولكن يمكننا أن نرى بالفعل أن أحجام الصور ليست ثابتة. ImageDataGenerator يتعامل مع هذا النوع من المشاكل ، من بين أشياء أخرى كثيرة.

بيانات الصورة هيفي الأساس مجموعة من الأرقام. يتم تمثيل الصور الملونة من خلال مجموعة من ثلاث مصفوفات ثنائية الأبعاد. تتكون كل من هذه المصفوفات من قيم بين 0 و255 (قد يختلف هذا). ثلاث من هذه القيم مجتمعة (كل واحدة من مصفوفة واحدة) تمثل اللون من البكسل.في حالتنا، صورنا لها شكل (512, 512, 3). وهذا يعني أن لدينا 512 * 512 = 262144 بكسل و 3 قنوات. (كما قلنا سابقًا، لا تتوافق جميعها مع حجم 512 * 512، لكننا سنتعامل معها.)

```
# Create ImageDataGenerator objects
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
```

```
# Assign the image directories to them
```

```
train_data_generator = train_datagen.flow_from_directory(
    "/content/train",
    target_size=(512,512)
)
```

```
test_data_generator = train_datagen.flow_from_directory(
    "/content/test",
    target_size=(512,512)
)
```

سينشئ كائن ImageDataGenerator البيانات على دفعات لنموذجنا عند الضرورة. هذا يسمح لنا بالعمل مباشرة مع البيانات المخزنة على القرص الصلب، دون زيادة التحميل على ذاكرة الوصول العشوائي. سيتم تمرير train_data_generator و test_data_generator كوسائط إلى معلمات x و validation_data على التوالي. نظرًا لأن ImageDataGenerator يحصل على الفئات من أسماء المجلدات، فإننا لا نحتاج إلى المعلمة y. (إذا حاولت تمرير وسيطة إلىy، فإن بايثون ستخطئ.)

الآن بعد أن تم تعيين بيانات التدريب والاختبار الخاصة بنا، يمكننا بناء نموذجنا وتدريبه.

بناء نموذج نقل التعلم باستخدام TensorFlow

في البداية، سنقوم بتحميل تطبيق Keras لنموذج ResNet50.

```
resnet_50 = tf.keras.applications.resnet50.ResNet50(include_top=False,
weights='imagenet')
resnet_50.trainable=False
```

include_top = False يضمن عدم تحميل الطبقة الأخيرة من نموذج ResNet50 "weights = imagenet" يحمّل أوزان ImageNet. إذا قمنا بتعيين weights=None، فسيتم تهيئة الأوزان بشكل عشوائي (في هذه الحالة، لن نقوم بإجراء نقل التعلم). من خلال تعيين سمة trainable على False، نضمن أن الأوزان الأصلية (ImageNet) للنموذج ستظل ثابتة.

نحتاج إلى مصنف ثنائي، لكن ResNet50 يحتوي على أكثر من عقدتينفي الطبقات النهائية. هذا يعني أنه يتعين علينا إضافة الطبقة النهائية يدويًا. لقد استخدمت واجهة برمجة تطبيقات وظيفية functional API، والتي يمكن أن تكون صعبة إذا كنت مستخدمًا مبتدئًا TensorFlow. (في هذه الحالة، أقترح عليك استخدام واجهة برمجة التطبيقات التسلسلية Sequential API، والتي تحتوي على بُنية أكثر وضوحًا.)

```
inputs = keras.Input(shape=(512,512,3))
x = resnet_50(inputs)
x = keras.layers.GlobalAveragePooling2D()(x)
outputs = keras.layers.Dense(2, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs, name="my_model")
model.compile(optimizer="Adam", loss="binary_crossentropy",
metrics=["accuracy"])
model.summary()
```

في هذه السطور، نحدد مدخلاتنا، ونمررها إلى نموذج resnet_50 الذي حددناه سابقًا، ونمرر مخرجاته إلى طبقة تجمع المتوسط العالمي Global Average Pooling layer، ونمرر ناتجها إلى طبقة كثيفة Dense layer مع عقد (لفئتين). يجب أن تكون دالة التنشيط softmax في هذه الحالة. مجموع قيم متجه خرج softmax يساوي دائمًا 1. بالنسبة إلى عقدتين (كل عقدة تمثل فئة)، لدينا x1 + x2 = 1 ، حيث يمثل x1 و x2 احتمالات الفئة. (خلاف ذلك، يمكن أن يكون لدينا 1 عقدة ودالة تنشيط sigmoid). بعد كل هذا، نحتاج إلى تجميع compile النموذج عن طريق اختيار مُحسِّن optimizer ودالة خطأ loss function. يمكننا أيضًا إضافة المقاييس التي يجب قياسها أثناء عملية التدريب. أخيرًا، يمكننا تدريب نموذجنا.

```
model.fit(train_data_generator, validation_data=test_data_generator,
epochs=5)
```

```
Epoch 1/5
16/16 [==============================] - 39s 2s/step - loss: 0.6494 - accuracy: 0.6466 - val_loss: 0.5793 - val_accuracy: 0.7160
Epoch 2/5
16/16 [==============================] - 32s 2s/step - loss: 0.4992 - accuracy: 0.7932 - val_loss: 0.5562 - val_accuracy: 0.6900
Epoch 3/5
16/16 [==============================] - 32s 2s/step - loss: 0.4572 - accuracy: 0.8032 - val_loss: 0.4126 - val_accuracy: 0.8500
Epoch 4/5
16/16 [==============================] - 32s 2s/step - loss: 0.3808 - accuracy: 0.8614 - val_loss: 0.3780 - val_accuracy: 0.8660
Epoch 5/5
16/16 [==============================] - 32s 2s/step - loss: 0.3375 - accuracy: 0.8855 - val_loss: 0.3561 - val_accuracy: 0.8760
<keras.callbacks.History at 0x7f05bba7ba50>
```

لقد انتهينا من جزء نقل التعلم. اختياريًا، يمكنك ضبط النموذج للحصول على نتائج أفضل.

الملاحظات النهائية

في هذه المقالة، تعلمنا كيفية تنفيذ نقل التعلم بمساعدة TensorFlow. نقل التعلم هو نهج قوي يسمح لنا بالتغلب على نقص البيانات. ومع ذلك، فهي ليست رصاصة فضية. هناك حالات عند العمل مع أي بيانات لدينا تكون أكثر منطقية وتؤدي إلى نتائج أفضل. ولها بدائل. زيادة البيانات Data augmentation أمر شائع. بالطبع، هذين ليسا حصريين. يمكن (غالبًا) دمج الأساليب المختلفة لحل مشكلة البيانات.

المصدر:

https://www.analyticsvidhya.com/blog/2021/11/transfer-learning-with-tensorflow/

14) تصنيف الأغذية باستخدام نقل التعلم و Food TensorFlow Classification Using Transfer Learning And TensorFlow

في مقالة اليوم، سنقوم بتحليل المواد الغذائية للتنبؤ بما إذا كانت طعامًا أم لا. نطبق أحدث أساليب نقل التعلم Transfer Learning وإطار عمل Tensorflow لبناء نموذج تعلم آلي لتصنيف الأطعمة.

المقدمة

تصنيف الصورة Image classification هو وظيفة حيث يتنبأ الجهاز بأن الصورة تنتمي إلى أي فئة. قبل أن يبدأ التعلم العميقفي الازدهار، لا يمكن لمهام مثل تصنيف الصور تحقيق الإنجاز على مستوى الإنسان.

ذلك لأن نموذج التعلم الآلي لا يمكنه تحديد معرفة الجار بالصورة. النموذج يتلقى فقط الأمر على مستوى البكسل.

بفضل إمكانات التعلم العميق، يمكن لمهام تصنيف الصور نقل الأداء على المستوى البشري باستخدام نموذج يوصف بالشبكة العصبية التلافيفية (CNN).

CNN هو نوع من نموذج التعلم العميق الذي يدرس التمثيل من الصورة. يمكن أن يحدد هذا النموذج من الميزات المسطحة إلى عالية المستوى دون مشاركة فردية.

لا يتلقى النموذج البيانات على مستوى البكسل فقط. يحصل النموذج أيضًا على بيانات الجوار من صورة بواسطة آلية تسمى الالتفاف convolution.

سيعمل الالتفاف على تجميع بيانات الجوار بضرب تجميع وحدات البكسلفي النطاق وجمعها في قيمة. سيقبل نموذج التعلم الآلي هذه الميزات لتصنيف الصورةفي مجموعة.

على الرغم من أن التعلم العميق يمكن أن يحقق الإنتاج على مستوى الإنسان، إلا أنه يحتاج إلى قدر كبير من البيانات.

ماذا لو لم يكن لدينا بيانات كافية؟ يمكننا تطبيق نظرية تسمى نقل التعلم.

نقل التعلم Transfer Learning هو أسلوب لتدريب نموذج على بيانات واسعة النطاق لاستعلامنا. وفقًا لذلك، نقوم بإعدادهم فقط من خلال ضبط النموذج fine-tuning the model. الميزة التي سنلاحظها هي أن النموذج سيتعلمفي وقت قصير.

ستقدمك المقالة لممارسة نقل التعلم لتصنيف صور الطعام باستخدام TensorFlow (Python). نظرًا لأن خطوة المعالجة المسبقة هي العملية الأساسية، فسوف أشرح أيضًا كيفية تقديم البيانات إلى نموذج التعلم العميق الخاص بنا.

دون مزيد من التأخير، فللنفذ الكود!

التنفيذ

الخطوة 1: استيراد المكتبات

الخطوة الأولى التي نطالب بها هي استيراد المكتبات import libraries. نريد TensorFlow وNumPy و os و pandas. إذا لم تكن قد نصبت الحزمة بعد، يمكنك تطبيق الأمر pip لتثبيت المكتبات المطلوبة.

يرجى ملاحظة أن TensorFlow الذي أمر باستخدامه هو الإصدار 2.4.1، لذا يرجى تثبيت هذا الإصدار.

أيضًا، إذا كنت ترغب في اعتماد GPU لتدريب نموذج التعلم العميق، فالرجاء تثبيت CUDA مع الإصدار 11.0 لأن هذا الإصدار يحمل TensorFlow مع الإصدار 2.4.1.

تنزيل برنامج CUDA.

يوجد أدناه كود لتثبيت وتحميل المكتبات المطلوبة.

```
# ! pip install tensorflow==2.4.1
# ! pip install pandas
# ! pip install numpy
import os
import numpy as np
import pandas as pd
import tensorflow as tf
```

الخطوة 2: تحضير البيانات

بعد ترتيب المكتبات، فإن الخطوة التالية هي تحضير مجموعة البيانات الخاصة بنا.في هذا المثال، سنطبق مجموعة بيانات تسمى Food-5K.

تتكون مجموعة البيانات هذه من 5000 صورة بفئتين، food وnon-food . يتم تقسيم FOOD-5K إلى تدريب training، والتحقق من الصحة validation، ومجموعة اختبار test من البيانات.

يتم تشكيل مجلد مجموعة البيانات على النحو التالي:

```
\---Food-5K
    +---evaluation
    |       0_0.jpg
    |       1_0.jpg
    |
    +---training
    |       0_0.jpg
    |       1_1.jpg
    |
    \---validation
    |       0_0.jpg
    |       1_0.jpg
```

كما تلاحظ أعلاه، يتكون كل مجلد من صور. يتضمن كل اسم ملف صورة الفئة والمعرف الخاص بها مقطوعًا بشرطة سفلية.

نحتاج إلى إنتاج إطار البيانات بأعمدة مثل اسم ملف الصورة والتسمية بترتيب المجلد هذا.

يبدو كود إصلاح مجموعة البيانات على هذا النحو:

```
def dframe(dtype):
    X = []
    y = []
    path = 'Food-5K/' + dtype + '/'
    for i in os.listdir(path):
        # Image
        X.append(i)
        # Label
        y.append(i.split('_')[0])
    X = np.array(X)
    y = np.array(y)
    df = pd.DataFrame()
    df['filename'] = X
    df['label'] = y
    return df
df_train = dframe('training')
df_val = dframe('validation')
df_test = dframe('evaluation')
```

```
df_train.head()
```

وإليك كيفية عرض إطار البيانات مثل:

	filename	label
0	0_0.jpg	0
1	0_1.jpg	0
2	0_10.jpg	0
3	0_100.jpg	0
4	0_1000.jpg	0

الخطوة التالية هي عمل كائن لوضع الصورفي النموذج. سنتدرب على كائن ImageDataGeneratorفي مكتبة tf.keras.preprocessing.image.

مع هذا الكائن، سوف ننتج دفعات من الصور. أيضًا، يمكننا زيادة صورتنا لتكبير منتج مجموعة البيانات. نظرًا لأننا نمد هذه الصور أيضًا، فقد قمنا بتعيين معلمات لتقنية زيادة الصورة.

أيضًا، نظرًا لأننا نطبق إطار بيانات كمعرفة حول مجموعة البيانات، فسنمارس طريقة flow_from_dataframe لإنتاج دفعات وزيادة الصور.

كود لما سبق يبدو مثل:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Create the ImageDataGenerator object
train_datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
)
val_datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
)
# Generate batches and augment the images
train_generator = train_datagen.flow_from_dataframe(
    df_train,
    directory='Food-5K/training/',
    x_col='filename',
    y_col='label',
    class_mode='binary',
    target_size=(224, 224),
)
val_generator = train_datagen.flow_from_dataframe(
    df_val,
    directory='Food-5K/validation/',
    x_col='filename',
    y_col='label',
    class_mode='binary',
    target_size=(224, 224),
)
```

الخطوة 3: تدريب النموذج

بعد أن ننتج الدُفعات batches، يمكننا الآن تدريب النموذج من خلال تقنية نقل التعلم. نظرًا لأننا نطبق هذه الطريقة، فإننا لا نطالب بتنفيذ بُنية CNN من البداية. بدلاً من ذلك، سوف نستخدم المعمارية الحالية والمُعدة مسبقًا.

سنقوم بتطبيق ResNet-50 كعمود فقري لنموذجنا الجديد. سنقوم بإنشاء المدخلات وضبط آخر طبقة خطية من ResNet-50 مع الطبقة الحديثة بناءً على عدد الفئات.

كود إنشاء النموذج أدناه:

```
from tensorflow.keras.applications import ResNet50

# Initialize the Pretrained Model
feature_extractor = ResNet50(weights='imagenet',
                             input_shape=(224, 224, 3),
                             include_top=False)

# Set this parameter to make sure it's not being trained
feature_extractor.trainable = False

# Set the input layer
input_ = tf.keras.Input(shape=(224, 224, 3))

# Set the feature extractor layer
x = feature_extractor(input_, training=False)

# Set the pooling layer
x = tf.keras.layers.GlobalAveragePooling2D()(x)

# Set the final layer with sigmoid activation function
output_ = tf.keras.layers.Dense(1, activation='sigmoid')(x)

# Create the new model object
model = tf.keras.Model(input_, output_)

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

# Print The Summary of The Model
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 6843s 72us/step
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
resnet50 (Functional)        (None, 7, 7, 2048)        23587712
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1)                 2049
=================================================================
Total params: 23,589,761
Trainable params: 2,049
Non-trainable params: 23,587,712
```

لتدريب النموذج، نطبق الطريقة الملائمة fit method لإعداده. ها هو الكود:

```
model.fit(train_generator, epochs=20, validation_data=val_generator)
```

الخطوة 4: اختبار النموذج

بعد تدريب النموذج، دعنا الآن نفحص النموذجفي مجموعة بيانات الاختبار. بالإضافة إلى ذلك، نحتاج إلى دمج مكتبة pillow لتحميل الصورة وتغيير حجمها وscikit-learn لتحديد أداء النموذج.

سنطبق classification_report من مكتبة scikit-Learn لإنتاج تقرير بشأن تنفيذ النموذج. أيضا، سوف نرسم مصفوفة الارتباك confusion matrix منه.

إليك الكود للتنبؤ ببيانات الاختبار وقرارها:

```
from PIL import Image
from sklearn.metrics import classification_report, confusion_matrix

y_true = []
y_pred = []

for i in os.listdir('Food-5K/evaluation'):
    img = Image.open('Food-5K/evaluation/' + i)
    img = img.resize((224, 224))
    img = np.array(img)
    img = np.expand_dims(img, 0)

    y_true.append(int(i.split('_')[0]))
    y_pred.append(1 if model.predict(img) > 0.5 else 0)

print(classification_report(y_true, y_pred))
print()
print(confusion_matrix(y_true, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       500
           1       0.99      0.96      0.97       500

    accuracy                           0.97      1000
   macro avg       0.97      0.97      0.97      1000
weighted avg       0.97      0.97      0.97      1000


[[495   5]
 [ 21 479]]
```

كما يمكنك أن تلاحظ من السابق، فإن النموذج قد تطرق بالفعل إلى أكثر من 95في المائة من حيث الأداء. وفقًا لذلك، يمكننا قبول هذا النموذجفي حالة إنشاء واجهة برمجة تطبيقات لمصنف الصور.

الخطوة 5: حفظ النموذج

إذا كنت ترغب في استخدام النموذج للاستخدام أو النشر اللاحق، يمكنك حفظ النموذج باستخدام طريقة save مثل هذا:

```
model.save('./resnet50_food_model')
```

إذا كنت تريد تحميل النموذج، فيمكنك التدرب على طريقة load_model مثل هذه:

```
model = tf.keras.models.load_model('./resnet50_food_model')
```

ماذا بعد

احسنت! أنت الآن تقر بكيفية إجراء نقل التعلم باستخدام TensorFlow. آمل أن تشجعك هذه الدراسة، خاصة أولئك الذين يطمحون إلى تدريب نموذج التعلم العميق ببيانات غير كافية.

المصدر

https://www.analyticsvidhya.com/blog/2021/05/food-classification-/using-transfer-learning-and-tensorflow

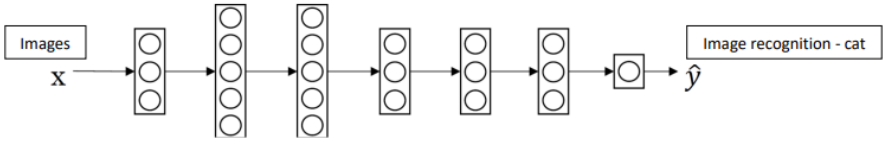
15) مقدمة في نقل التعلم باستخدام Introduction to MNIST Transfer Learning using MNIST

ما هو نقل التعلم؟

واحدة من أقوى الأدواتفي التعلم العميق هي أنهفي بعض الأحيان يمكننا أخذ المعرفة أو المعلمات التي تعلمتها الشبكة العصبية من مهمة واحدة وتطبيق تلك المعرفة على مهمة مختلفة. على سبيل المثال، ربما لدينا نموذج شبكة عصبية، تعلمنا التعرف على أشياء مثل القطط والكلاب والحيوانات الأخرى. ثم نستخدم هذه المعرفة أو نستخدم جزءًا منها للقيام بعمل أفضلفي قراءة فحوصات الأشعة السينية X-ray scans. وهذا ما يسمى نقل التعلم Transfer Learning. للحصول على تعريف أكثر واقعية،في نقل التعلم، نعيد استخدام نموذج مدرب مسبقًا pre-trained model على مشكلة جديدة. هذا مفيد بشكل خاص لأنهفي التعلم العميق يمكننا تدريب نماذج أكثر تعقيدًا، بكميات أقل من البيانات باستخدام هذه الطريقة. قد يكون هذا مفيدًافي علم البيانات لأنه،في معظم مشاكل العالم الحقيقي، هناك نقصفي نقاط البيانات المصنفة لتدريب مثل هذه النماذج المعقدة.

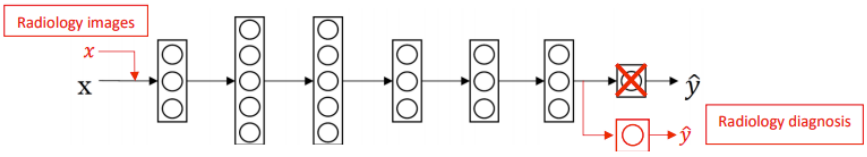
الحدس لنقل التعلم

لنفترض أنك دربت شبكتك على التعرف على الصور. لذلك أولاً تأخذ شبكة عصبية وتدربها على أزواج XY حيث X عبارة عن صورة ما، وY عبارة عن كائن مافي الصور، مثل القطط أو الكلاب.



لنفترض الآن أننا نريد أخذ هذه الشبكة العصبية وتكييفها، أو كما نقول "نقل transfer" التعلم إلى مشكلة مختلفة، مثل التشخيص الإشعاعي. ما يمكننا القيام به هو أخذ آخر طبقة إخراج للشبكة العصبية، والتي يشار إليها أحيانًا باسم "الرأس Head" وحذفها فقط، وكذلك حذف الأوزان (المعلمات) التي تغذي تلك الطبقة التي تمت إزالتها وإنشاء مجموعة جديدة من القيم المهيأة عشوائيًا فقط للطبقة الأخيرة، والتي يمكن أن تنتج التشخيص الإشعاعي.

لذا، نأخذ الآن مجموعة البيانات الجديدة مع زوج X'Y حيث يمثل X "صور الأشعة أو الأشعة السينية، و Y" هو تشخيصنا. نقوم بتدريب نموذجنا بهذه المجموعة الجديدة من البيانات. إنه يعمل بنفس قدر الدقة الذي كان يعمل به مع مجموعة البيانات السابقة.



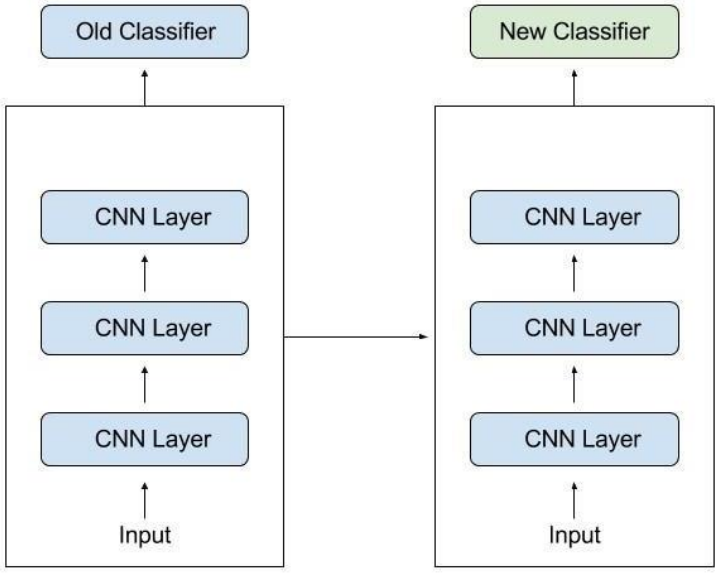
متى يكون التعلم عن طريق النقل منطقيًا؟

نفترض أن نموذجنا قد تم تدريبه مبدئيًا على المهمة A ونريد نقله إلى المهمة B.

المهمة A و B لها نفس نوع الإدخال، أي إذا كانت المهمة A لمعالجة الصور، يجب أن تكون المهمة B نوعًا من تحليل الصور أيضًا.

يجب أن تكون كمية بيانات A أكبر بكثير من B.

يمكن أن تكون ميزات المستوى المنخفض لـ A مفيدةفي تعلم الميزات عالية المستوى لـ B.



EfficientNetB0 و MNIST

في هذه المقالة، سنعمل مع مجموعة البيانات الشهيرة MNIST (المعهد الوطني المعدل للمعايير والتكنولوجيا Modified National Institute of Standards and Technology) وهي مجموعة بيانات شائعة جدًا وهي واحدة من أقدم مجموعات البيانات التي تم إنشاؤهافي عام 1998. وهي عبارة عن مجموعة ضخمة من الأرقام المكتوبة بخط اليد، وشائعة الاستخدام لأنظمة معالجة الصور. تحتوي مجموعة البيانات على 60.000 بيانات تدريب و 10000 بيانات اختبار.

من ناحية أخرى، فإن EfficientNetB0 عبارة عن شبكة عصبية تلافيفية تم تصميمها بواسطة Google ويتم تدريبها على قاعدة بيانات ImageNet. تتكون قاعدة بيانات ImageNet من 14 مليون صورة من فئات مختلفة، جميعها مشروحة يدويًا.

نظرًا لأن EfficientNetB0 عبارة عن شبكة مدربة بالفعل، فمن الناحية النظرية، يمكننا إجراء نقل التعلم والتنبؤ بتسميات البيانات من MNIST بقدر لا بأس به من الدقة. وبالتالي،في حالتنا، المهمة A هي معالجة الصور لبيانات ImageNet، والمهمة B هي التنبؤ بالأرقام باستخدام قاعدة بيانات MNIST.

يمكنك العثور على مجموعة البيانات على Kaggle هنا. لنبدأ الآن!

نقل التعلم

تركيب المكتبات والاستيراد

```
!pip install efficientnet-pytorch
```

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.optim import lr_scheduler
from torch.autograd import Variable
from torch.utils.data import DataLoader, Dataset

from sklearn.metrics import accuracy_score
from PIL import Image, ImageOps, ImageEnhance
```

```
from efficientnet_pytorch import EfficientNet
```

```
#parameters

BATCH_SIZE = 64
VALID_BATCH_SIZE = 100
TEST_BATCH_SIZE = 100
EPOCHS = 5
NUM_CLASSES = 10
SEED = 42
EARLY_STOPPING = 25
OUTPUT_DIR = '/kaggle/working/'
MODEL_NAME = 'efficientnet-b0'
```

قراءة بيانات MNIST

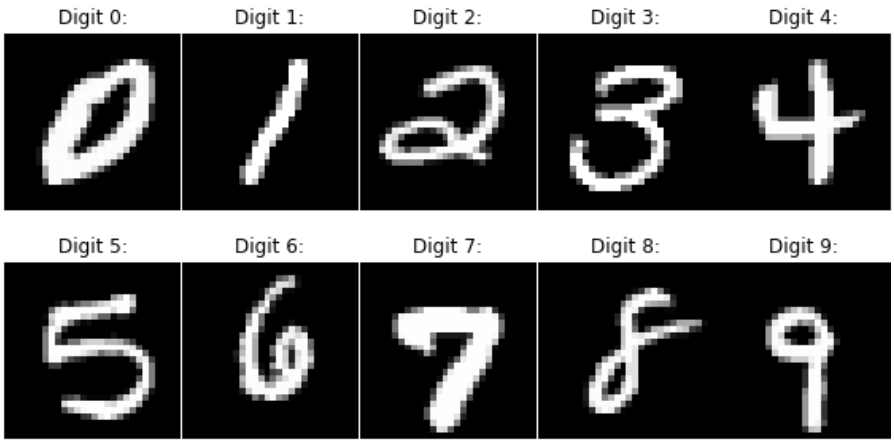
```
train = pd.<a
onclick="parent.postMessage({'referent':'.pandas.read_csv'},
'*')">read_csv('/kaggle/input/digit-recognizer/train.csv')
test = pd.<a
onclick="parent.postMessage({'referent':'.pandas.read_csv'},
'*')">read_csv('/kaggle/input/digit-recognizer/test.csv')

print('Shape of the training data: ', train.shape)
print('Shape of the test data: ', test.shape)
```

```
Shape of the training data:  (42000, 785)
Shape of the test data:  (28000, 784)
```

```
sample_df = train.groupby('label').apply(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.[996,1002]'}, '*')">lambda <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.[996,1002]..x'}, '*')">x: <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.[996,1002]..x'}, '*')">x.sample(n=1)).reset_index(drop = True)
sample_df.drop(columns=['label'], inplace=True)
```

```
nrows = 2
ncols = 5
fig, axs = plt.<a
onclick="parent.postMessage({'referent':'.matplotlib.pyplot.subplots'}
, '*')">subplots(nrows=nrows, ncols=ncols, gridspec_kw={'wspace':
0.01, 'hspace': 0.05},
                        squeeze=True, figsize=(10,12))

ind_y = 0
ind_x = 0
for i, row in sample_df.iterrows():
    if ind_y > ncols - 1:
        ind_y = 0
        ind_x += 1
    sample_digit = sample_df.values[i, :].reshape((28, 28))
    axs[ind_x, ind_y].axis('off')
    axs[ind_x, ind_y].imshow(sample_digit, cmap='gray')
    axs[ind_x, ind_y].set_title("Digit {}:".format(i))
    ind_y += 1
```

```
plt.<a
onclick="parent.postMessage({'referent':'.matplotlib.pyplot.show'},
'*')">show()
```




```
from sklearn.model_selection import train_test_split


# Perform train, validation split
train_df, valid_df = train_test_split(train, test_size = 0.2,
random_state=SEED,stratify=train['label'])
```

```
import cv2

# Define custom data loader,
# code adapted from https://www.kaggle.com/juiyangchang/cnn-with-
pytorch-0-995-accuracy

n_pixels = len(train_df.columns) - 1


class MNIST Dataset(Dataset):
    """MNIST data set"""

    def __init__(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self, <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..df'}, '*')">df
                ):

        if len(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..df'}, '*')">df.columns) == n_pixels:
            # test data
            <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.X = <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..df'}, '*')">df.values.reshape((-
1,28,28)).astype(np.<a
onclick="parent.postMessage({'referent':'.numpy.uint8'},
'*')">uint8)[:,:,:,<a
```

```
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..None'}, '*')">None]
            <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.y = <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..None'}, '*')">None

            <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.X3 = np.<a
onclick="parent.postMessage({'referent':'.numpy.full'}, '*')">full((<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.X.shape[0], 3, 28,
28), 0.0)

            for <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..i'}, '*')">i, <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..s'}, '*')">s in enumerate(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.X):
                <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.X3[<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..i'}, '*')">i] = np.<a
onclick="parent.postMessage({'referent':'.numpy.moveaxis'},
'*')">moveaxis(cv2.<a
onclick="parent.postMessage({'referent':'.cv2.cvtColor'},
'*')">cvtColor(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..s'}, '*')">s, cv2.<a
onclick="parent.postMessage({'referent':'.cv2.COLOR_GRAY2RGB'},
'*')">COLOR_GRAY2RGB), -1, 0)

        else:
            # training/validation data
            <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST Dataset.  init  ..self'}, '*')">self.X = <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..df'},
'*')">df.iloc[:,1:].values.reshape((-1,28,28)).astype(np.<a
onclick="parent.postMessage({'referent':'.numpy.uint8'},
'*')">uint8)[:,:,:,<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..None'}, '*')">None]
            <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.y = torch.<a
onclick="parent.postMessage({'referent':'.torch.from_numpy'},
'*')">from numpy(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..df'}, '*')">df.iloc[:,0].values)

            <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.X3 = np.<a
onclick="parent.postMessage({'referent':'.numpy.full'}, '*')">full((<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
```

```
93611.MNIST_Dataset.__init__..self'}, '*')">self.X.shape[0], 3, 28,
28), 0.0)

            for <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..i'}, '*')">i, <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..s'}, '*')">s in <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..enumerate'}, '*')">enumerate(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.X):
                <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..self'}, '*')">self.X3[<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..i'}, '*')">i] = np.<a
onclick="parent.postMessage({'referent':'.numpy.moveaxis'},
'*')">moveaxis(cv2.<a
onclick="parent.postMessage({'referent':'.cv2.cvtColor'},
'*')">cvtColor(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__init__..s'}, '*')">s, cv2.<a
onclick="parent.postMessage({'referent':'.cv2.COLOR_GRAY2RGB'},
'*')">COLOR_GRAY2RGB), -1, 0)

    def __len__(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__len__..self'}, '*')">self):
        return len(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__len__..self'}, '*')">self.X3)

    def __getitem__(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__getitem__..self'}, '*')">self, <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__getitem__..idx'}, '*')">idx):
        if <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__getitem__..self'}, '*')">self.y is not None:
            return <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__getitem__..self'}, '*')">self.X3[<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__getitem__..idx'}, '*')">idx] , <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__getitem__..self'}, '*')">self.y[<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__getitem__..idx'}, '*')">idx]
        else:
            return <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__getitem__..self'}, '*')">self.X3[<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.MNIST_Dataset.__getitem__..idx'}, '*')">idx]
```

```
train_dataset = MNIST_Dataset(train_df)
valid_dataset = MNIST_Dataset(valid_df)
test_dataset = MNIST_Dataset(test)
```

```
train_loader = torch.<a
onclick="parent.postMessage({'referent':'.torch.utils'},
'*')">utils.data.DataLoader(dataset=train_dataset,
                                          batch_size=BATCH_SIZE,
                                          shuffle=True)
valid_loader = torch.<a
onclick="parent.postMessage({'referent':'.torch.utils'},
'*')">utils.data.DataLoader(dataset=valid_dataset,

batch_size=VALID_BATCH_SIZE, shuffle=False)
test_loader = torch.<a
onclick="parent.postMessage({'referent':'.torch.utils'},
'*')">utils.data.DataLoader(dataset=test dataset,
                                          batch_size=TEST_BATCH_SIZE,
shuffle=False)
```

نقل التعلم باستخدام بُنية EfficientNet

```
## Load in pretrained effnet model and remove its head, replacing it
with fully connected layer
## that gives 10 outputs
def get_model(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.get_model..model_name'}, '*')">model_name='efficientnet-b0'):
    <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.get_model..model'}, '*')">model = EfficientNet.<a
onclick="parent.postMessage({'referent':'.efficientnet_pytorch.Efficie
ntNet.from_pretrained'}, '*')">from_pretrained(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.get_model..model_name'}, '*')">model_name)
    del <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.get_model..model'}, '*')">model._fc
    # # # use the same head as the baseline notebook.
    <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.get_model..model'}, '*')">model._fc = nn.<a
onclick="parent.postMessage({'referent':'.torch.nn.Linear'},
'*')">Linear(1280, NUM_CLASSES)
    return <a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.get_model..model'}, '*')">model
```

بهذه السهولة! لقد أخذنا للتو النموذج المدرب مسبقًا، وأزلنا "رأسه" وأضفنا نموذجًا جديدًا لتلبية احتياجاتنا! الآن سنقوم بتدريب النموذج لبضع فترات من خلال "رأسنا" الجديد، وسنرى النتائج.

```
import random
import os

def set_seed(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.set_seed..seed'}, '*')">seed: int = 42):
    random.seed(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.set_seed..seed'}, '*')">seed)
    np.<a onclick="parent.postMessage({'referent':'.numpy.random'},
'*')">random.seed(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.set_seed..seed'}, '*')">seed)
```

```
    os.environ["PYTHONHASHSEED"] = str(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.set_seed..seed'}, '*')">seed)
    torch.<a
onclick="parent.postMessage({'referent':'.torch.manual_seed'},
'*')">manual_seed(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.set_seed..seed'}, '*')">seed)
    torch.<a onclick="parent.postMessage({'referent':'.torch.cuda'},
'*')">cuda.manual_seed(<a
onclick="parent.postMessage({'referent':'.kaggle.usercode.11949480.435
93611.set_seed..seed'}, '*')">seed)  # type: ignore
```

```
set_seed(SEED)
device = torch.<a
onclick="parent.postMessage({'referent':'.torch.device'},
'*')">device('cuda' if torch.cuda.is_available() else 'cpu')
output_dir = OUTPUT_DIR


model = get_model(MODEL_NAME)
model = model.to(device)


# # # get optimizer
optimizer = optim.<a
onclick="parent.postMessage({'referent':'.torch.optim.Adam'},
'*')">Adam(model.parameters(), lr=0.001)

# # # get scheduler
scheduler = lr_scheduler.<a
onclick="parent.postMessage({'referent':'.torch.optim.lr_scheduler.Cos
ineAnnealingLR'}, '*')">CosineAnnealingLR(optimizer, T_max=10)

# # # get loss
loss_func = nn.<a
onclick="parent.postMessage({'referent':'.torch.nn.CrossEntropyLoss'},
'*')">CrossEntropyLoss()

if torch.<a onclick="parent.postMessage({'referent':'.torch.cuda'},
'*')">cuda.is_available():
    model = model.cuda()
    loss_func = loss_func.cuda()

best_val_accuracy = 0
min_val_loss = np.<a
onclick="parent.postMessage({'referent':'.numpy.inf'}, '*')">inf
best_epoch = 0
batches = 0
epochs_no_improve = 0
n_epochs_stop = EARLY_STOPPING
```

```
for epoch in range(EPOCHS):
    running_loss = 0.0
    targets = torch.<a
onclick="parent.postMessage({'referent':'.torch.empty'},
'*')">empty(size=(BATCH_SIZE, )).to(device)
    outputs = torch.<a
onclick="parent.postMessage({'referent':'.torch.empty'},
'*')">empty(size=(BATCH_SIZE, )).to(device)
```

```
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        batches += 1
        data, target = Variable(data), Variable(target)
        if torch.<a
onclick="parent.postMessage({'referent':'.torch.cuda'},
'*')">cuda.is_available():
            data = data.type(torch.<a
onclick="parent.postMessage({'referent':'.torch.FloatTensor'},
'*')">FloatTensor).cuda()
            target = target.cuda()
        targets = torch.<a
onclick="parent.postMessage({'referent':'.torch.cat'},
'*')">cat((targets, target), 0)
        optimizer.zero_grad()
        output = model(data)
        loss = loss_func(output, target)
        output = torch.<a
onclick="parent.postMessage({'referent':'.torch.argmax'},
'*')">argmax(torch.<a
onclick="parent.postMessage({'referent':'.torch.softmax'},
'*')">softmax(output, dim=1), dim=1)
        outputs = torch.<a
onclick="parent.postMessage({'referent':'.torch.cat'},
'*')">cat((outputs, output), 0)
        running_loss += loss.item()
        loss.backward()
        optimizer.step()
    scheduler.step()
    print('train/loss on EPOCH {}: {}'.format(epoch,
running loss/batches))
    train_acc =
accuracy_score(targets.cpu().detach().numpy().astype(int),

outputs.cpu().detach().numpy().astype(int))
    print('train/accuracy: {} for epoch {}'.format(train_acc, epoch))
```

```
val/loss: 0.11400251267921357
val/accuracy: 0.968218336483932 for epoch 0
Best val/acc: 0.968218336483932 for epoch 0, saving model---->
train/loss on EPOCH 1: 0.06466097908091849
train/accuracy: 0.9796815589353612 for epoch 1
val/loss: 0.06320315383241645
val/accuracy: 0.9801512287334594 for epoch 1
Best val/acc: 0.9801512287334594 for epoch 1, saving model---->
train/loss on EPOCH 2: 0.04791723521567204
train/accuracy: 0.9844047053231939 for epoch 2
val/loss: 0.04016698708680148
val/accuracy: 0.9873582230623819 for epoch 2
Best val/acc: 0.9873582230623819 for epoch 2, saving model---->
train/loss on EPOCH 3: 0.032390923193747284
train/accuracy: 0.9894249049429658 for epoch 3
val/loss: 0.04896201956268799
val/accuracy: 0.985586011342155 for epoch 3
train/loss on EPOCH 4: 0.029551956583294407
train/accuracy: 0.9913260456273765 for epoch 4
val/loss: 0.037788029067173955
val/accuracy: 0.9878308128544423 for epoch 4
Best val/acc: 0.9878308128544423 for epoch 4, saving model---->
```

كما ترى، نحصل على دقة تصل إلى 99٪في 5 فترات فقط!!!!

الاستنتاج

في هذه المقالة، تم تعريفنا بنقل التعلم Transfer Learning وهو مفهوم مهم جداً للتعلم العميق. مع نقل التعلم، يمكننا إعادة استخدام نموذج مبني بالفعل، وتغيير الطبقات القليلة الأخيرة، وتطبيقه على مشاكل مماثلة والحصول على نتائج دقيقة حقًا.

ثم تابعنا واستخدمنا بُنية الشبكة العصبية التي طورتها Google والتي تسمى EfficientNetB0، واستخدمنا نقل التعلم للتنبؤ بالأرقام من مجموعة بيانات MNIST وحصلت على دقة تقارب 99٪.

المصدر

https://www.analyticsvidhya.com/blog/2021/05/transfer-learning-using-mnist/

16) التعلم العميق في الطب: تصنيف خلايا الدم المصابة بالملاريا باستخدام ResNet Deep Learning in Medical: Classification of Malaria Infected Blood Cells with ResNet

التعلم العميق Deep learning هو مجموعة فرعية من التعلم الآلي machine learning وقد تم تطبيقهفي مختلف المجالات للمساعدةفي حل المشكلات الحالية. مجال واحد لتطبيق التعلم العميقفي المجال الطبي.في المجال الطبي، التعلم العميق مفيد لأخصائيي الأشعة. لأن عمل التحليل يدويًا يستغرق وقتًا طويلاً جدًا. تناقش هذه المقالة تصنيف خلايا الدم المصابة بالملاريا malaria-infected blood cells باستخدام خوارزمية التعلم العميق، Resnet.

هذا التطبيق متاح للجمهور عبر الإنترنت على العنوان التالي:

https://www.kaggle.com/muhammadarnaldo/klasifikasi-malaria-cnn/

نشأت مجموعة البيانات المستخدمةفي هذه المقالة من مجموعة البيانات العامة الرسمية للمعاهد الوطنية للصحة (NIH). يمكنك أيضًا الوصول إلى مجموعة البيانات هذه من خلال Kaggle.com.

استيراد المكتبات

```
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import
Conv2D,MaxPool2D,Flatten,Dense,Dropout,Input,AveragePooling2D
from keras.preprocessing.image import ImageDataGenerator,
img_to_array, load_img
from keras.applications import ResNet50V2
from keras.models import Model

%matplotlib inline
from matplotlib import pyplot as plt #untuk visualisasi data
from matplotlib import image as mpimg

from sklearn.metrics import confusion_matrix
import seaborn as sn
import numpy as np #struktur data
```

أول شيء يجب القيام به هو استيراد المكتبات المطلوبة. المكتبات الرئيسية المستخدمةفي هذا التطبيق هي Keras ، كمكتبة للتعلم الآلي ؛ Matplotlib ، كأداة التصوير ؛ و Sklearn كمصفوفة للتقييم.

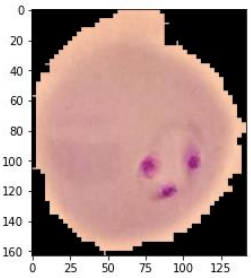
تحميل مجموعة البيانات

```
import os
data_dir = "/kaggle/input/cell-images-for-detecting-
malaria/cell_images/cell_images"
print(os.listdir(data_dir))
```

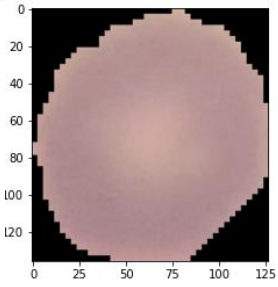
فيما يلي المرحلة الأولية، وهي استيراد المكتبات المطلوبة وتحميل مجموعة البياناتفي شكل صور إلى البيئة للمعالجة. ثم يعرض المجلدفي فئتين تصنيف، وهما غير مصاب uninfected ومتطفل parasitized. تتضمن المكتبات المستخدمة Keras وNumPy كمكتبات تعلم عميق، وSklearn كمكتبات تقييم للنموذج، وMatplotlib للمساعدةفي التصوير. هنا هو الإخراج من الكود أعلاه. يشير اسمي المجلدين إلى اسم الفئة. غير مصاب بالفئة الصحية healthy / العادية normal،في حين أن الطفيلي هو فئة للمرضى المصابين infected patients.

```
['Uninfected', 'Parasitized']
```

ثم نعرض صورة مسبقة التحميل للخلايا المصابة والصحية، للتأكد من أن البيانات المحملة صحيحة ومناسبة. نقوم بتجربة صورة عشوائية واحدة من كل فئة للعرض.



```
img_path2="/kaggle/input/cell-images-for-detecting-
malaria/cell images/cell images/Uninfected/C1 thinF IMG 20150604 10472
2_cell_60.png"
gambar2 = mpimg.<a
onclick="parent.postMessage({'referent':'.matplotlib.image.imread'},
'*')">imread(img_path2)
plt.<a
onclick="parent.postMessage({'referent':'.matplotlib.pyplot.imshow'},
'*')">imshow(gambar2)
```



المعالجة المسبقة

```
dim = 128
batch = 32

datagen =ImageDataGenerator(rescale=1/255.0, validation_split=0.3,
    rotation_range=20,
    zoom_range=0.05,
```

```
    width_shift_range=0.05,
    height_shift_range=0.05,
    shear_range=0.05,
    horizontal_flip=True)

train_data =datagen.flow_from_directory(data_dir,
target_size=(dim,dim), batch_size=batch, class_mode = 'categorical',
subset = 'training')
validation_data =datagen.flow_from_directory(data_dir,
target_size=(dim,dim), batch_size=batch, class_mode = 'categorical',
subset = 'validation', shuffle=False)
```

كمية البيانات المستخدمة هي 27558، منها 19292 بيانات (70٪) مستخدمة للتدريب، و 8266 بيانات (30٪) مستخدمة للاختبار أو التحقق من الصحة.في مرحلة المعالجة المسبقة preprocessing stage هذه، يتم تغيير حجم الصورة إلى أبعاد 128 × 128 بكسل. ثم يتم أيضًا تطبيق التدرج الرمادي. بالإضافة إلى ذلك، يتم أيضًا تطبيق زيادة الصورة image augmentation على مجموعة البيانات. الزيادات المستخدمة هي التدوير والتكبير / التصغير وتحويل العرض والارتفاع والقص والوجه أفقيًا.

إنشاء النموذج

```
baseModel =ResNet50V2(include_top=False,
                      input_tensor=Input(shape=(dim,dim, 3)))

headModel =baseModel.outputheadModel =AveragePooling2D(pool_size=(4,
4))(headModel)
headModel =Flatten(name="flatten")(headModel)
headModel =Dense(128, activation="relu")(headModel)
headModel =Dropout(0.5)(headModel)
headModel =Dense(2, activation="softmax")(headModel)

model =Model(inputs=baseModel.input, outputs=headModel)

model.summary()
```

في هذه المرحلة، يتم تشكيل النموذج. بُنية CNN الأساسية باستخدام ResNet-50. كما يوحي الاسم، فإن النموذج الذي تم تطويره يحتوي على 50 طبقة التفاف جنبًا إلى جنب مع طبقات التجميع الخاصة بهم.في نهاية النموذج، توجد طبقتان متصلتان بالكامل fully connected layers. تتكون الطبقة الأولى من 128 خلية عصبية مع تنشيط ReLU. تتكون الطبقة الثانية من 2 خلايا عصبية تمثل عدد فئات التصنيف، باستخدام تنشيط softmax. بين هذه الطبقات المتصلة بالكامل، توجد طبقة تسرب dropout layer، ويتم ضبط التكوين على 50٪. المعلمات / الأوزان الإجماليةفي هذا النموذج هي 23827330. يستخدم هذا النموذج نقل التعلم من النموذج المدرب مسبقًا على مجموعة بيانات ImageNet. لكن لم يتم عرض الكود (ضمني implicit). تُستخدم عملية نقل التعلم هذه لتسريع عملية التدريب. تنتقل صورة الإدخال إلى الالتفاف convolutionوالتجميع pooling. سيتم تكرار طبقة الالتفاف والتجميع هذه. بين الطبقات، هناك أيضًا تسوية دفعية batch normalization للتنظيم regulation وهناك طبقة تنشيط activation layer.في نموذج ResNet، توجد طبقة مهمة جدًا، وهي الطبقة

الإضافية. هذا هو مجموع المتبقيات residues من طبقة الالتفاف السابقة على طبقة الالتفاف التالية. هذه هي السمة المميزة لـ ResNet وهي أيضًا ثورةفي عالم التعلم العميق.

عملية التدريب

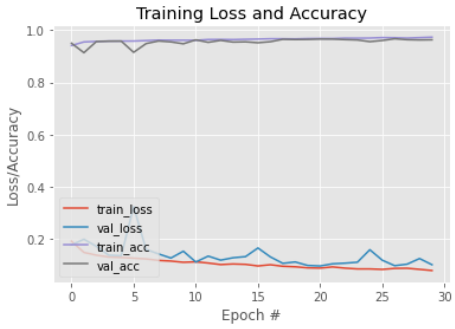
تكوينات العملية التدريبية هي كما يلي:

```
EP = 30

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

history =model.fit(train_data, validation_data=validation_data,
epochs=EP)
print("*** proses training selesai ***")
```

يتم تنفيذ عملية التدريبفي 30 فترة، باستخدام مُحسِّن Adam. دالة الخطأ loss function المستخدمة هي الانتروبيا المتقاطعة cross-entropy، لأن لدينا فئتين فقط لتصنيفهما. سيقوم model.fit بتشغيل عملية التدريب. تتضمن عملية التدريب بيانات التحقق من أجل تتبع الضبط الزائد overfittingفي خطوة التقييم.

تقييم النموذج

```
plt.style.use("ggplot")
plt.figure()

plt.plot(np.arange(0,EP),history.history["loss"], label="train_loss")
plt.plot(np.arange(0,EP),history.history["val_loss"],
label="val_loss")
plt.plot(np.arange(0,EP),history.history["accuracy"],
label="train_acc")
plt.plot(np.arange(0,EP),history.history["val_accuracy"],
label="val_acc")

plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")

plt.legend(loc="lower left")
```




```
test_loss, test_acc = model.evaluate(validation_data)
```

```
259/259 [==============================] - 44s 168ms/step - loss: 0.1036 - accuracy: 0.9619
```

نتائج الدقة التي تم الحصول عليها هي 96.19٪ من بيانات الاختبار.في المخطط السابق، يمكن ملاحظة أن خطوط التدريب والاختبار قريبة من بعضها البعض ولا تظهر أي ضبط زائد.

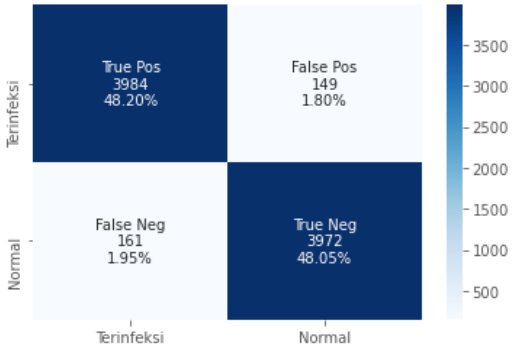
يشير الخطان الأرجواني الموجودانفي الجزء العلوي من الرسم البياني إلى الدقة accuracy، وتبدو الدقة ثابتة نسبيًا وتتزايد. بالمقارنة، تشير الخطوط الحمراء والزرقاءفي الجزء السفلي من الرسم البياني إلى الخطأ loss.

تقييم التصنيف مع مصفوفة الارتباك

```
predictions = model.predict(validation_data)
y_pred =np.argmax(predictions, axis=-1)
#y_pred = [1 * (x[0]>=0.5) for x in predictions]cf_matrix
=confusion_matrix(validation_data.classes,y_pred)

group_names = ["True Pos","False Pos","False Neg","True Neg"]
group_counts = ["{0:0.0f}".format(value)
forvalueincf_matrix.flatten()]
group_percentages = ["{0:.2%}".format(value)
forvalueincf_matrix.flatten()/np.sum(cf_matrix)]

labels = [f"{v1}\n{v2}\n{v3}"
forv1,v2,v3inzip(group_names,group_counts,group_percentages)]
labels =np.asarray(labels).reshape(2,2)

categories = ["Terinfeksi","Normal"]
sn.heatmap(cf_matrix, annot=labels, fmt='', xticklabels=categories,
yticklabels=categories, cmap='Blues')
```

توضح مصفوفة الارتباك confusion matrix التالية أن النموذج قادر على التعميم جيداً generalize well. سواء كان الأمر يتعلق بتصنيف الصور المصابة وكذلك تصنيف الصور العادية. الحساسية Sensitivity والخصوصية Specificity موزعة بالتساوي.

يبدو أيضًا أن التنبؤات الخاطئة بشأن الصور المصابة والعادية وصلت إلى حد 2٪ فقط. لذلك يمكننا أن نستنتج أن استخدام ResNet-50في تصنيف صور الملاريا حصل على نتائج جيدة.



حفظ النموذج

يمكننا حفظ النموذج بتنسيق Hierarchical Data Format h5 مع الكود التالي لاستخدامه لاحقًا.

```
model.save('./cnn_malaria_binary.h5')
```

الاستنتاج

- يمكن تنفيذ التعلم العميق في المجال الطبي، وخاصة في تصنيف الملاريا ؛
- أثبت Resnet-50 قدرته على التعميم جيداً في تصنيف الصور الطبية ؛
- نجحت بُنية التعلم العميق مع التكوين أعلاه في تحقيق دقة جيدة بنسبة 96،19٪ بدون الضبط الزائد.

ماذا بعد

- بناء تطبيق التعلم العميق في مجال آخر ؛
- أنشئ تصنيفًا باستخدام مجموعة البيانات الحالية مع خوارزمية تعلم عميق أخرى.

المصدر

https://www.analyticsvidhya.com/blog/2021/12/deep-learning-in-medical-field/

17) التنبؤ بالصورة باستخدام نموذج تم تدريبه مسبقًا Image Prediction Using a Pre-trained Model

المقدمة

يتنافس الباحثون من جميع أنحاء العالم لإنشاء أكثر أنظمة التعرف على الصور picture recognition systems دقة وفعالية. لذلك، غالبًا ما يكون من المنطقي استخدام تصميم شبكة عصبية موجود كنقطة انطلاق لمشاريعك الخاصة بدلاً من جعلهم يثنون تصميمات الشبكة العصبية الخاصة بهم من البداية. والأفضل من ذلك، شارك الباحثون إصدارات الشبكة العصبية المدربة لهياكل الشبكات هذه بعد تدريبهم على مجموعات بيانات كبيرة. لذلك، يمكننا استخدام هذه الشبكات العصبية المدربة بالفعل إما بشكل مباشر أو كنقطة انطلاق لتدريبنا.

مجموعات البيانات

يتم استخدام التسلسل الهرمي للكلمات Wordhierarchy (الآن الأسماء فقط) لتنظيم الصور في ImageNet، حيث تمثل آلاف الصور كل عقدةفي التسلسل الهرمي. حقق المشروع تقدمًا كبيرًافي مجالات التعلم العميق والرؤية الحاسوبية. يمكن للباحثين الوصول إلى البيانات مجانًا ولأغراض أكاديمية.

إن تحدي التعرف البصري على نطاق واسع على ImageNet، أو ILSVRC، هو مسابقة سنوية للتعرف على الصور تعقدها ImageNet. تتنافس الفرق الدولية من الكليات والشركات لإنشاء نماذج التعرف على الصور الأكثر دقة. تم تطوير النماذج المدربة مسبقًا المضمنةفي Keras باستخدام مجموعة بيانات أصغر مستخدمة لهذه المسابقة. يتم تضمين صور 1000 نوع مختلف من الكائنات، بمافي ذلك السلالات الغذائية والحيوانية،في مجموعة البيانات. على سبيل المثال، تفاحة Granny Smith هي أحد أنواع الكائناتفي مجموعة البيانات. يتم تضمين أكثر من 1200 صورة لهذا النوع فقط من التفاحفي مجموعة البيانات.

نماذج مُدربة مسبقًا

بعض النماذج المدربة مسبقًا لتصنيفات الصور:

VGG

تحتوي شبكة عصبية عميقة تسمى VGG على 16 أو 19 طبقة.في عام 2014، مثلت طليعة. تصميم شبكتها العصبية التلافيفية تقليدي تمامًا. نظرًا لأنه سهل الاستخدام والفهم، فإنه لا يزال يستخدم بشكل متكرر كأساس لنماذج أخرى. لكن التصميمات الأحدث عادةً ما يكون لها كفاءة أفضل.

ResNet-50

يمكن أن تكون شبكة ResNet-50، وهي شبكة عصبية مكونة من 50 طبقة تمثل أحدث ما توصلت إليه التكنولوجيا منذ عام 2015، أكثر دقة باستخدام ذاكرة أقل من بُنية VGG. تستخدم شبكة ResNet تصميمًا أكثر تعقيدًا حيث ترتبط المستويات العليا من الشبكة العصبية بطبقات عديدة تحتها بالإضافة إلى الطبقة الموجودة أسفلها مباشرةً.

Inception v3

تصميم آخر ممتاز الأداء من 2015 هو Inception v3. يحتوي على تخطيط أكثر تعقيدًا مبنيًا حول الطبقات التي تنقسم إلى العديد من المسارات المتميزة قبل العودة معًا. توضح هذه الشبكات التعقيد المتزايد وحجم الشبكات العصبيةفي البحثفي عامي 2014 و 2015 لتحسين الدقة. غالبًا ما تكون هياكل الشبكات العصبية الحديثة أكثر تخصصًا.

ما الغرض من النموذج المدرَّب مسبقًا ولأي غرض؟

1. هذه نماذج عبارة عن شبكات معقدة بها الكثير من المتغيرات.
2. عادة ما يستغرق تدريب مثل هذه الشبكة الكثير من الوقت والموارد.
3. حتى لو كان الأمر مختلفًا بعض الشيء ، فقد نخلع الطبقة العليا ونقوم فقط بتدريب وزن تلك الطبقة (نقل التعلم)

الحل

الآن، دعنا ننفذ نموذجًا مدربًا مسبقًا للتعرف على الأشياء والصور. سأستخدم مكتبة Keras حيث يتم تضمين جميع النماذج المدربة مسبقًا مع Keras. سأغطي فقط نموذج VGG.

الآن دون مزيد من اللغط، فلنبدأ.

أولاً، لنستورد جميع الحزم المطلوبة. يمكننا استيراد VGG16 من تطبيق Keras.

```
#Import modules
import numpy as np
from keras.preprocessing import image
from keras.applications import vgg16
import matplotlib.pyplot as plt
from pathlib import Path
```

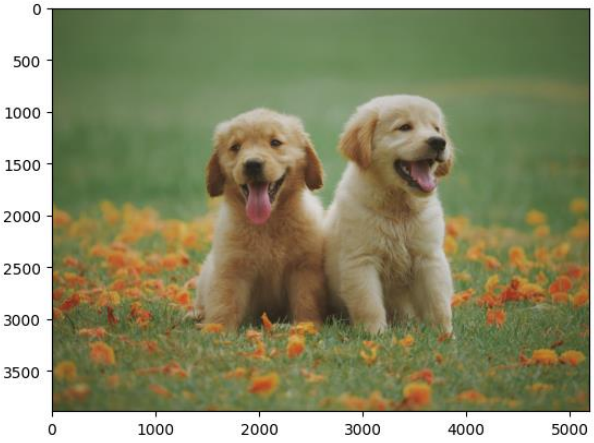
بعد ذلك، دعنا نحمل ملف الصورة لمعالجته. لقد استخدمت صورة كلب ولكن لا تترددفي استخدام أي صورة تريدها. إنه أكبر من أن نستخدم شبكة عصبية لمعالجة الصورة التي نحملها. يجب أن يتوافق حجم الصورة مع عدد عقد الإدخالفي الشبكة العصبية عند تغذيتها بالصور. يجب أن يكون حجم الصور التي نضعهافي شبكة VGGهو 224 × 224 بكسل. لذلك، نقوم بتعيين معلمة size الهدف على تلك القيمة. بالإضافة إلى ذلك، سنستخدم image.img لطريقة

المصفوفة لتحويل بيانات الصورة إلى مجموعة من الأرقام التي يمكننا إدخالها بعد ذلكفي الشبكة العصبية.

```
#Image path
path = Path("/content/dogs_pic.jpeg")
#Load the image
img = image.load_img(path, target_size=(224,224))
image_array = image.img_to_array(img)
```

الآن، نعطي الصورة البعد الرابع. هذا حتى يمكن لـ Keras تلقي مجموعة من الصورفي وقت واحد. نتيجة لذلك، تصبح صورتنا الفردية مصفوفة من عدة صور، كل منها يحتوي على عنصر واحد. يجب دائمًا تسوية الصور قبل إدخالهافي الشبكة العصبية بحيث تكون قيمة كل بكسل بين صفر وواحد. إدخال ما قبل المعالجة ودالة التسوية المضمنةفي نموذج VGG ستنجز ذلك. كل ما علينا فعله هو الاتصال بنموذج VGG هاتفيا وإرسال معلوماتنا. لتمرير بياناتنا، وهي x، سنقول vgg16.preprocessinput. الآن دعونا نرسم الصورة باستخدام مكتبة matplotlib.

```
x_train = np.expand_dims(image_array, axis=0)
#Normalize the data
x_train = vgg16.preprocess_input(x_train)
data = plt.imread(path)
plt.imshow(data)
plt.show()
```



الآن، سننشئ مثيلًا جديدًا للنموذج عن طريق إنشاء كائن vgg16 جديد.

```
model = vgg16.VGG16(weights="imagenet")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weigh
ts_tf_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 31s 0us/step
553476096/553467096 [==============================] - 31s 0us/step
```

نحن الآن على استعداد لعمل تنبؤ بالصورة باستخدام البيانات المطبعة normalized data والشبكة العصبية. باستخدام model.pred و توفير بياناتنا، يمكننا تحقيق ذلك. ستكون

التوقعات التي نتلقاها عبارة عن مصفوفة أرقام فاصلة عائمة تحتوي على 1000 عنصر. سيشير كل عنصر من عناصر المصفوفة إلى احتمال وجود كل عنصر من 1000 شيء تم تدريب النموذج على التعرف عليهفي صورتنا. يتم توفير أسماء التطابقات الأكثر احتمالاً من خلال دالة التنبؤ بفك ترميز الصورة لنموذج VGG، مما يجعل الأمور أبسط. هنا، يمكن استدعاء توقعات vgg16.decode. بعد ذلك، نرسل ببساطة كائن التنبؤ بالصورة الذي أنشأناه بالفعل. سيوفر لنا تلقائيًا أفضل خمس مطابقات على الأرجح.

```
prediction = model.predict(x_train)
pred = vgg16.decode_predictions(prediction)
print(pred)
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_ind
ex.json
40960/35363 [================================] - 0s 1us/step
49152/35363 [==========================================] - 0s 1us/step
[[('n02099601', 'golden_retriever', 0.5354275), ('n02113799', 'standard_poodle', 0.32865775), ('n021
04029', 'kuvasz', 0.050152738), ('n02099712', 'Labrador_retriever', 0.017040437), ('n02093647', 'Bed
lington_terrier', 0.013122461)]]
```

يتناسب التنبؤ بالصورة مع صورتنا جيداً،في رأيي. تعد كل من Standard poodle، و kuvasz، و Labrador Retever عددًا قليلاً من المطابقات الأخرى. كل هذه فرضيات معقولة للصورة. لا تترددفي المحاولة مرة أخرى مع الصور الخاصة بك. من الممتع مراقبة التوقعات التي ستحدثها وأنواع الصور التي ستربكها.

الاستنتاج

لقد نجحنافي تنفيذ نماذج VGG المدربة مسبقًا وتوقعنا الصورة باستخدامها. لقد قدمت لمحة عامة عن خوارزميات تصنيف الصور المدربة مسبقًا لـ VGG وكيفية استخدامها. ولكن نظرًا لأن هذا مجال يتوسع باستمرار، فهناك دائمًا نموذج جديد لتوقع آفاق جديدة لاستكشافها. أناشدك أن تختبر النماذج المذكورة أعلاهفي مجموعات بيانات مختلفة بإعدادات معلمات مختلفة وأن تبلغ النتائج التي توصلت إليهافي التعليقات أدناه!

الملخص

النماذج المدربة مسبقًا تشبه السحر؛ قد نستخدمها على الفور بدون تدريب أو بيانات بمجرد تنزيلها.

إذا اختلفت الوظيفة المصدر والمهمة المستهدفة، لكن المجالات متشابهة إلى حد ما، فقد نحتاج إلى تدريب بضع طبقات، لكنها لن تستغرق وقتًا طويلاً مثل البدء من الصفر وستتطلب بيانات أقل بكثير.

حالة الاستخدام الجيدة لاستيراد نموذج موجود وتشغيل التنبؤ بالصورة على الفور هيفي المراحل الأولى من النماذج الأولية أو مجرد تجربة نموذج. ومع ذلك، لا يزال ضبط الشبكة هو الإجراء المفضل.

لا تحتاج الشبكة العصبية عادةً إلى التدريب من البداية. بدلاً من ذلك، يمكننا استخدام شبكة عصبية موجودة وتعديلها لحل مشكلة جديدة عبر نقل التعلم.

أين الكود؟

يمكن العثور على الكود الكامل على Github هنا. لقد استخدمت Google Colaboratory ولكن لا ترددفي استخدام ما يناسبك. أثناء وجودك هناك، سيكون النجم مفيداً.

المصدر:

https://www.analyticsvidhya.com/blog/2022/09/image-prediction-using-a-pre-trained-model/

18) التعرف على صورة زجاجة كوكا كولا باستخدام نقل التعلم Coca-Cola Bottle Image Recognition Using Transfer Learning

المقدمة

تبنت شركة Coca-Cola إعادة استخدام زجاجاتها وجميع الفوائد البيئية والنقدية التي تأتي مع ذلك. عندما يشتري العملاء مشروب كوكاكولافي علب زجاجية، تتم مكافأتهم عند إعادة الزجاجة الفارغة. جعلني هذا أفكرفي جميع الزجاجات والعلب البلاستيكية التي لا تضمن مكافأة تؤدي إلى رميها وإهدارها. يجب أن تكون هناك طريقة لتحديد زجاجات Coca Cola تلقائيًا لإعادة استخدامها داخل الشركة.

يمكن تمييز زجاجات الكوكاكولا بسهولة باستخدام الملصقات التي عليها طباعة كبيرة، " Coca Cola". عادة ما تكون الطباعة باللون الأبيض. يمكننا الحصول على الملصق عن طريق عزل اللون الأبيض وتدريب النموذج على الصور المجزأة.

المكتبات

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib import colors
import os
import cv2
import PIL
from tensorflow.keras.layers import Dense,Conv2D, Dropout,Flatten,
MaxPooling2D
from tensorflow import keras
from tensorflow.keras.models import Sequential, save_model, load_model
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.applications.inception_v3 import InceptionV3,
preprocess_input
from tensorflow.keras.callbacks import ModelCheckpoint
from sklearn.decomposition import PCA
```

من الأعلى:

- يستخدم Numpy لمعالجة بيانات المصفوفة.
- يستخدم Matplotlib لتصوير الصور وإظهار مدى إمكانية تمييز اللون في نطاق معين من الألوان.
- يستخدم OS للوصول إلى بنية الملف.
- يتم استخدام CV2 لقراءة الصور وتحويلها إلى أنظمة ألوان مختلفة.
- يستخدم Keras للشبكة العصبية الحقيقية.

تحويل مخطط الألوان

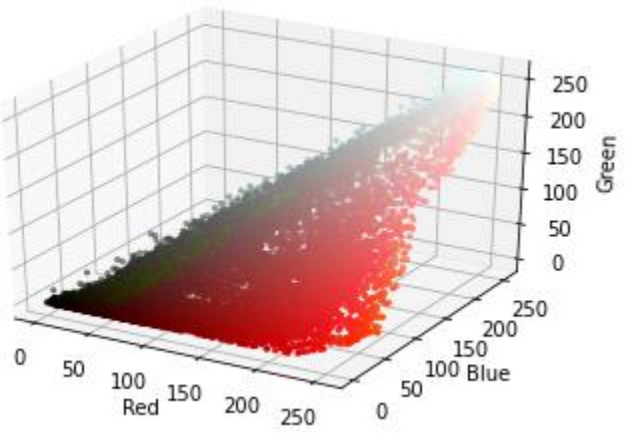
تحديد نظام الألوان المناسب:

لكي نتمكن من عزل الألوان، نحتاج إلى التحقق من مدى وضوح الألوانفي أنظمة الألوان المختلفة. يمكننا استخدام مخططات ثلاثية الأبعاد لهذا الغرض

أولاً، يمكننا تصوير الصورة بتنسيق ألوان RGBفي مساحة ثلاثية الأبعاد.

هنا، قمنا بشكل أساسي بتقسيم الصورة إلى مكوناتها التي تكونفي هذه الحالة حمراء وخضراء وزرقاء ثم قمنا بإعداد الرسم ثلاثي الأبعاد. الشيء التالي هو إعادة تشكيل الصورة وبعد ذلك نقوم بتسوية normalize الصورة مما يقلل النطاق من 0ـ255 إلى 0ـ1. أخيرًا، تُستخدم دالة scatter() لإنشاء مخطط مبعثر ثم نقوم بتسمية المحاور وفقًا لذلك.

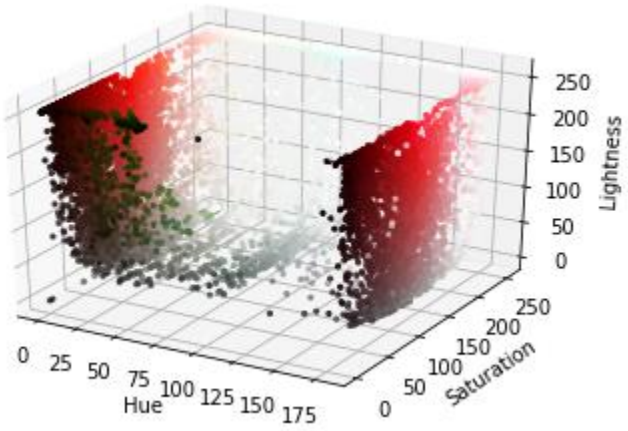
```
red, green, blue = cv2.split(img)
fig = plt.figure()
axis = fig.add_subplot(1, 1, 1, projection="3d")
pixel_colors = img.reshape((np.shape(img)[0]*np.shape(img)[1], 3))
norm = colors.Normalize(vmin=-1.,vmax=1.)
norm.autoscale(pixel_colors)
pixel_colors = norm(pixel_colors).tolist()
axis.scatter(red.flatten(), green.flatten(), blue.flatten(),
facecolors=pixel_colors, marker=".")
axis.set_xlabel("Red")
axis.set_ylabel("Green")
axis.set_zlabel("Blue")
plt.show()
```



غالبًا ما تخدم مخططات HSL وHSV بشكل أفضل لتجزئة الصور image segmentation. يمكننا رسم مخططات ثلاثية الأبعاد للصورةفي مخطط HSL.

```
hue, saturation, lightness = cv2.split(img)
fig = plt.figure()
```

```
axis = fig.add_subplot(1, 1, 1, projection="3d")
axis.scatter(hue.flatten(), saturation.flatten(), lightness.flatten(),
facecolors=pixel_colors, marker=".")
axis.set_xlabel("Hue")
axis.set_ylabel("Saturation")
axis.set_zlabel("Lightness")
plt.show()
```

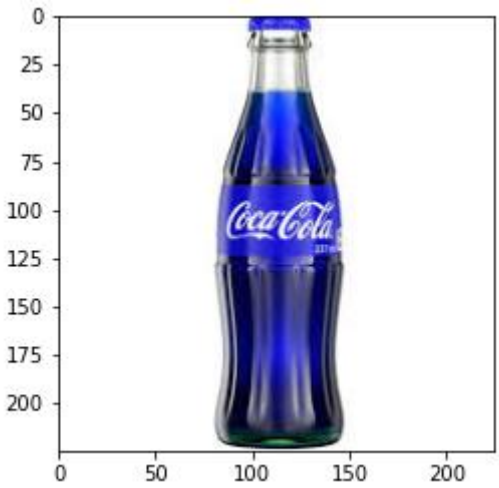


تجدر الإشارة إلى أن الألوان المعينة ليست مختلطةفي الرسم الثاني كمافي الأولى. يمكننا بسهولة تمييز وحدات البكسل ذات اللون الأبيض عن البقية.

تحويل الألوان

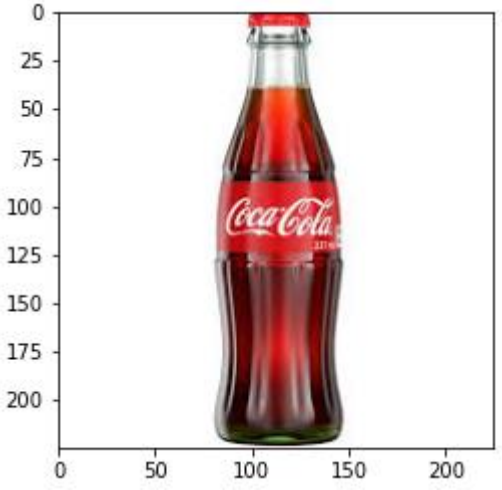
يقرأ CV2 افتراضيًا الصورفي مخطط BGR.

```
img = cv2.imread(img_path)
plt.imshow(img)
plt.show()
```



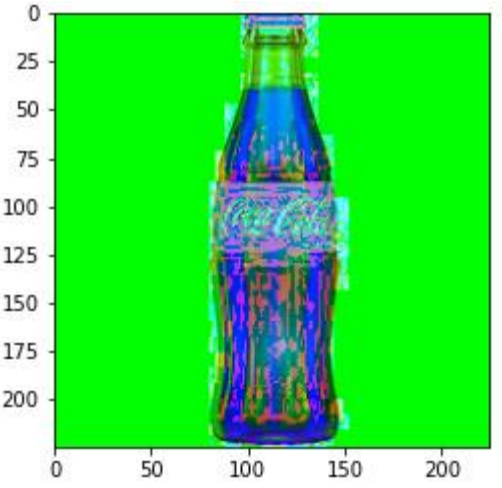
يجب تحويل هذه الصورة إلى تنسيق RGB أولاً:

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.show()
```



أخيرًا، يجب تحويل هذه الصورة إلى مخطط HLS للسماح بتمييز الألوان بسهولة. HSL هو وصف تدرج اللون Hue والتشبع Saturation والخفة Lightness للصورة.

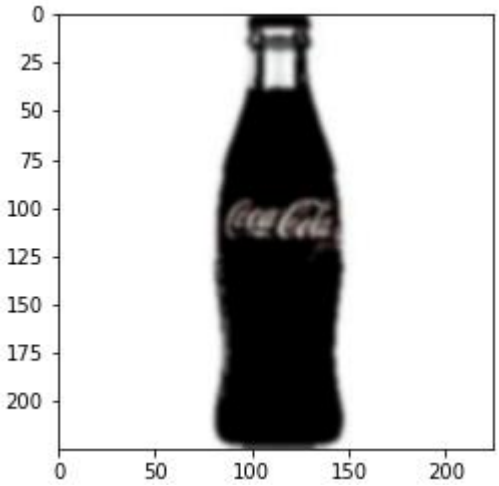
```
img = cv2.cvtColor(img, cv2.COLOR RGB2HLS)
plt.imshow(img)
plt.show()
```



عزل اللون الأبيض:

نحدد قيم الحد الأدنى والأعلى للون الأبيض ثم نحدد قناعًا باستخدام cv2.inRange() الذي يحدد صورة HSL. هذا يعيد 0 و 1. ثم نفرض القناع على صورة RGB الأصلية باستخدام دالة bitwise_and. المعنى الضمني هنا هو أنه سيحتفظ بقيمة البكسل إذا كانت القيمة المقابلة للقناع 1. يتم تطبيق Gaussian Blur لتنعيم الحواف.

```
hsl_img = cv2.cvtColor(img, cv2.COLOR_RGB2HLS)
low_threshold = np.array([0, 200, 0], dtype=np.uint8)
high_threshold = np.array([180, 255, 255], dtype=np.uint8)
mask = cv2.inRange(hsl_img, low_threshold, high_threshold)
white_parts = cv2.bitwise_and(img, img, mask = mask)
blur = cv2.GaussianBlur(white_parts, (7,7), 0)
```



نقل التعلم مع INCEPTIONV3

يمكن أن تستغرق بعض النماذج وقتًا طويلاً للتدريب، وفي النهاية لا تزال تتمتع بدقة تنبؤية غير مثيرة للإعجاب.في حين أنه من النبيل تمامًا أن ترغبفي إنشاء نموذج من البداية، فإن نقل التعلم transfer learning يستغل استخدام النماذج التي أسفرت عن نتائج رائعة على صور ImageNet.في هذا المشروع بالذات، سأوضح كيفية استخدام InceptionV3.

نحدد النموذج المراد استخدامه بمافي ذلك الأوزان وشكل الإدخال ثم نلائم fit قيم Xفي النموذج ونحفظ القيم الناتجةفي متغير يسمى "bottle_neck_features_train".

```
model = InceptionV3(weights='imagenet', include_top=False,
input_shape=(299, 299, 3))
bottle_neck_features_train = model.predict_generator(X, n/32,
verbose=1)
bottle_neck_features_train.shape
np.savez('inception_features_train',
features=bottle_neck_features_train)
train_data = np.load('inception_features_train.npz')['features']
```

```
train_labels = np.array([0] * m + [1] * p) // where m+p = n
```

شكل الميزات الناتجة هو ((n, 8, 8, 2048), n هو عدد الصور المستخدمة. ثم نحفظ القيم في ملف يسمى "inception_features_train.npz" لتجنب التدريب المسبق المتكرر. ما سبق هو مشكلة تصنيف ثنائية (الزجاجة إما زجاجة كوكا كولا أو ليست كذلك).

ويترتب على ذلك أن الملصقات (التسميات) يمكن أن تكون إما 0 لـ "Coke" أو 1 لـ "Not Coke" (أو أي مجموعة أخرى من أزواج الأرقام). يمكن معالجة ملصقات التدريب عن طريق إنشاء سلسلة من الأصفار مساوية لعدد صور زجاجات الكوكاكولا متبوعة بأخرى لبقية الصور.

الشبكة العصبية

يمكن وصف الشبكة العصبية بأنها سلسلة من الخوارزميات التي تحل مشكلة عن طريق محاكاة طريقة عمل الدماغ البشري. هنا، نستخدم نموذج Sequential Keras.

```
classifier = Sequential()
classifier.add(Conv2D(32, (3, 3), activation='relu',
input_shape=train_data.shape[1:], padding='same'))
classifier.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
classifier.add(MaxPooling2D(pool_size=(3, 3)))
classifier.add(Dropout(0.25))
classifier.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
classifier.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Dropout(0.50))
classifier.add(Flatten())
classifier.add(Dense(512, activation='relu'))
classifier.add(Dropout(0.6))
classifier.add(Dense(256, activation='relu'))
classifier.add(Dropout(0.5))
classifier.add(Dense(1, activation='sigmoid'))
```

طبقة الإدخال هي طبقة تلافيفية Convolutional Layer. المعلمات المستخدمة لهذه الطبقة هي عدد المرشحات filters التي تبلغ 32، وحجم النواة الذي يبلغ 3 × 3 ، ودالة التنشيط activation function وهي الوحدة الخطية المصححة ReLu ، وشكل الإدخال input shape الذي هو شكل صورة واحدة، والحشو padding. دائمًا ما يكون إدخال طبقة conv2d الأولى عبارة عن مصفوفة رباعية الابعاد. الأبعاد الثلاثة الأولى هي إدخال الصورة image input والأخيرة هي القنوات channels.

ReLu هي دالة تنشيط غير خطية على الرغم من أنها تعمل ويبدو أنها دالة خطية. ينفذ عمليات حسابية قد تسفر عن القيمة المضافة كمدخلات أو قد تسفر عن صفر.

يستخدم MaxPooling لاستخراج الميزات.في هذه الحالة، قمنا بتعيين حجم تجمع يبلغ 3،3 مما يعني أن النموذج سيستخرج صورة بحجم 3،3 بكسل ويحصل على أقصى قيمةفي هذا التجميع.

يستخدم التسرب Dropout لمنع الضبط الزائد Overfitting. يقوم بشكل عشوائي بتعيين الحواف الصادرة للوحدات المخفية على 0في كل تحديث لمرحلة التدريب. يتم توضيح الفرق بين الضبط الزائد، والضبط الناقص underfitting ، ,والضبط المناسب properly fitted أدناه.

تقوم الطبقة المسطحة Flatten layer بتحويل بيانات الإدخال إلى مصفوفة ثنائية الأبعاد. تضمن الطبقة الكثيفة Dense layer أن كل عقدةفي طبقة واحدة متصلة بالعقدة الأخرى للطبقة التالية.

نستخدم دالة التنشيط Sigmoid لطبقة الإخراج. يتم تحويل الإدخال إلى قيمة بين 0.0 و 1.0. المدخلات الأكبر بكثير من 1.0 يتم تحويلها إلى القيمة 1.0 ، وبالمثل ، فإن القيم الأصغر بكثير من 0.0 يتم قطعها إلى 0.0. شكل الدالة لجميع المدخلات الممكنة هو شكل S من صفر إلى 0.5 إلى 1.0.

لحفظ النموذج والأوزان على فترات زمنية معينة، نستخدم ModelCheckpoint مع model.fit. يضمن تعيين "save_best_only" على "true" حفظ النموذج الذي أنتج أفضل النتائج فقط. أخيرًا، نقوم بتجميع compile النموذج وتناسب fit البيانات.

```
checkpointer = ModelCheckpoint(filepath='./weights_inception.hdf5',
verbose=1, save_best_only=True)
classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics
=['binary_accuracy'])
history = classifier.fit(train_data,
train_labels,epochs=50,batch_size=32, validation_split=0.3,
verbose=2, callbacks=[checkpointer], shuffle=True)
```

أداء التنبؤات

نظرًا لأننا نحتاج إلى استخدام بيانات مشابهة للبيانات التي تم إدخالها إلى المصنف، يمكننا كتابة دالة تقوم بالتحويل ثم إجراء التنبؤات.

```
def predict(filepath):
    img = cv2.imread(filepath)
    img = cv2.resize(img,(299,299))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    hsl_img = cv2.cvtColor(img, cv2.COLOR_RGB2HLS)
    low_threshold = np.array([0, 200, 0], dtype=np.uint8)
    high_threshold = np.array([180, 255, 255], dtype=np.uint8)
    mask = cv2.inRange(hsl_img, low_threshold, high_threshold)
    white_parts = cv2.bitwise_and(img, img, mask = mask)
    blur = cv2.GaussianBlur(white_parts, (7,7), 0)
    print(img.shape)
    clf = InceptionV3(weights='imagenet', include_top=False,
input_shape=white_parts.shape)
    bottle_neck_features_predict =
clf.predict(np.array([white_parts]))[0]
    np.savez('inception_features_prediction',
features=bottle_neck_features_predict)
    prediction_data =
np.load('inception_features_prediction.npz')['features']
    q = model.predict( np.array( [prediction_data,] )  )
```

```
    prediction = q[0]
    prediction = int(prediction)
    print(prediction)
```

الخطوات التالية

رؤية كيف لدينا ما يلي:

- نموذج تنبؤي عملي.
- القيم المحفوظة للنموذج.
- صور للتنبؤ.
- دالة جاهزة لعمل التنبؤات.

يمكننا الآن محاولة إجراء تنبؤات على الصور.

كل ما نحتاج إلى فعله هو استدعاء دالة التنبؤ predict function وتمرير المسار إلى الصورة كمعامل.

```
predict("./train/Coke Bottles/Coke1.png")
```

يجب أن يوفر هذا 1 كمخرج منذ صورنا لزجاجات الكوكاكولا التي وصفناها بـ 1.

حفظ النموذج

إذا كان هذا قابلاً للتطبيق على الإطلاقفي برنامج مثل تطبيقفي الوقت الفعلي يستخدم وحدات OpenCV لعرض صورة أو مقطع فيديو وإجراء تنبؤات، فلا يمكننا أن نأمل بشكل واقعيفي تدريب نموذجنافي كل مرة يتم فيها تشغيل البرنامج.

أكثر منطقية هو حفظ النموذج وتحميله بمجرد فتح البرنامج. هذا يعني أننا بحاجة إلى الاستفادة Keras' load_model و save_model الخاص بـ Keras. نحن نستورد هذه على النحو التالي.

```
from tensorflow.keras.models import Sequential, save_model, load_model
```

الآن يمكننا حفظ النموذج عن طريق استدعاء save_model() وإدخال اسم المجلد كمعامل.

```
save_model(save_model)
```

أخيرًا، يمكننا ترتيب أي برنامج ببساطة عن طريق تحميل النموذج باستخدام load_model بدلاً من إدخال الكود لإعادة تدريب النموذج.

```
load_model("./save_model")
```

المصدر:

https://www.analyticsvidhya.com/blog/2021/01/coca-cola-bottle-image-recognition

19) تصنيف الصور متعدد الفئات باستخدام نقل التعلم Multiclass image classification using Transfer learning

يعد تصنيف الصور Image classification إحدى مشكلات التعلم الآلي الخاضعة للإشراف والتي تهدف إلى تصنيف صور مجموعة البيانات إلى فئات categories أو تسميات labels خاصة بها. يعد تصنيف صور سلالات الكلاب dog breeds المختلفة مشكلة كلاسيكيةفي تصنيف الصور. لذلك، يتعين علينا تصنيف أكثر من فئة واحدة وهذا هو سبب تصنيف الاسم متعدد الفئات multi-class classification، وفي هذه المقالة، سنفعل الشيء نفسه من خلال الاستفادة من نموذج تم تدريبه مسبقًا InceptionResNetV2 ، وتخصيصه.

دعونا أولا نناقش بعض المصطلحات.

نقل التعلم Transfer learning: يعد نقل التعلم طريقة تعلم عميقًا شائعة تتبع نهج استخدام المعرفة التي تم تعلمهافي بعض المهام وتطبيقها لحل مشكلة المهمة المستهدفة ذات الصلة. لذلك، بدلاً من إنشاء شبكة عصبية من البداية، "ننقل transfer" الميزات المكتسبة والتي تمثل أساسًا "أوزان weights" الشبكة. لتنفيذ مفهوم نقل التعلم، فإننا نستخدم "النماذج المدربة مسبقًا pre-trained models".

ضروريات نقل التعلم Necessities for transfer learning: يجب أن تكون الميزات منخفضة المستوى من النموذج A (المهمة A) مفيدة لتعلم النموذج B (المهمة B).

نموذج مدرب مسبقًا Pre-trained model: النماذج المدربة مسبقًا هي نماذج التعلم العميق التي يتم تدريبها على مجموعات بيانات كبيرة جدًا، ويتم تطويرها وإتاحتها بواسطة مطورين آخرين يرغبونفي المساهمةفي مجتمع التعلم الآلي هذا لحل أنواع مماثلة من المشكلات. يحتوي على تحيزات biases وأوزان weights الشبكة العصبية التي تمثل ميزات مجموعة البيانات التي تم التدريب عليها. الميزات التي تم تعلمها قابلة للتحويل transferrable دائمًا. على سبيل المثال، سيحتوي النموذج الذي تم تدريبه على مجموعة بيانات كبيرة من صور الزهور على ميزات تم تعلمها مثل الزوايا والحواف والشكل واللون وما إلى ذلك.

InceptionResNetV2 : هي شبكة عصبية تلافيفية بعمق 164 طبقة، مدربة على ملايين الصور من قاعدة بيانات ImageNet ، ويمكنها تصنيف الصور إلى أكثر من 1000 فئة مثل الزهور والحيوانات وما إلى ذلك. حجم إدخال الصور هو 299× 299.

وصف مجموعة البيانات:

- تتألف مجموعة البيانات المستخدمة من 120 سلالة من الكلابفي المجموع.

- كل صورة لها اسم ملف وهو معرفها الفريد.
- مجموعة بيانات التدريب (train.zip): تحتوي على 10222 صورة يتم استخدامها لتدريب نموذجنا
- مجموعة بيانات الاختبار (test.zip): تحتوي على 10357 صورة يتعين علينا تصنيفهافي الفئات أو التسميات ذات الصلة.
- labels.csv: يحتوي على أسماء تولد مطابقة لمعرف الصورة.
- sample_submission.csv: يحتوي على النموذج الصحيح لتقديم العينة المطلوب إجراؤه
- يمكن تنزيل جميع الملفات المذكورة أعلاه من هنا.

ملاحظة: للحصول على أداء أفضل، استخدم GPU.

نقوم أولاً باستيراد جميع المكتبات اللازمة.

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

from sklearn.metrics import classification_report, confusion_matrix

# deep learning libraries
import tensorflow as tf
import keras
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import applications
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D,
Flatten, Dense, Dropout
from keras.preprocessing import image

import cv2

import warnings
warnings.filterwarnings('ignore')
```

تحميل مجموعات البيانات ومجلدات الصور:

```
from google.colab import drive
drive.mount("/content/drive")

# datasets
labels = pd.read_csv("/content/drive/My Drive/dog/labels.csv")
sample = pd.read_csv('/content/drive/My
Drive/dog/sample submission.csv')

# folders paths
train_path = "/content/drive/MyDrive/dog/train"
test_path = "/content/drive/MyDrive/dog/test"
```

عرض السجلات الخمسة الأولى لمجموعة بيانات التسميات لمعرفة سماتها.

```
labels.head()
```

	id	breed
0	000bec180eb18c7604dcecc8fe0dba07	boston_bull
1	001513dfcb2ffafc82cccf4d8bbaba97	dingo
2	001cdf01b096e06d78e9e5112d419397	pekinese
3	00214f311d5d2247d5dfe4fe24b2303d	bluetick
4	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever

إضافة امتداد ".jpg" لكل معرّف

يتم ذلك لجلب الصور من المجلد نظرًا لأن اسم الصورة والمعرفات متماثلان، لذا فإن إضافة امتداد jpg سيساعدنافي استرداد الصور بسهولة.

```
def to_jpg(id):
          return id+".jpg"


labels['id'] = labels['id'].apply(to_jpg)
sample['id'] = sample['id'].apply(to_jpg)
```

زيادة البيانات:

إنها تقنية معالجة مسبقة نزيد فيها مجموعة البيانات الحالية بإصدارات محولة من الصور الحالية. يمكننا إجراء التحجيم scaling والدوران rotations وزيادة السطوع brightness والتحولات الأفينية affine transformations الأخرى. هذه تقنية مفيدة لأنها تساعد النموذج على تعميم البيانات generalize غير المرئية جيداً.

يتم استخدام فئة ImageDataGenerator لهذا الغرض والتي توفر زيادةفي الوقت الفعلي للبيانات

وصف لبعض معلماته المستخدمة أدناه:

- **rescale**: يعيد قياس القيم حسب العامل المحدد.
- **horizontal flip**: قلب المدخلات بشكل عشوائي أفقيًا.
- **Validation_split**: هذا هو جزء الصور المحجوزة للتحقق من الصحة (بين 0 و 1).

```
# Data agumentation and pre-processing using tensorflow
gen = ImageDataGenerator(
                                        rescale=1./255.,
                                        horizontal_flip = True,
```

```
                                        validation_split=0.2 # training:
80% data, validation: 20% data
                                        )

train_generator = gen.flow_from_dataframe(
          labels, # dataframe
          directory = train_path, # images data path / folder in which
images are there
          x_col = 'id',
          y_col = 'breed',
          subset="training",
          color_mode="rgb",
          target size = (331,331), # image height , image width
          class_mode="categorical",
          batch_size=32,
          shuffle=True,
          seed=42,
)


validation_generator = gen.flow_from_dataframe(
          labels, # dataframe
          directory = train_path, # images data path / folder in which
images are there
          x col = 'id',
          y_col = 'breed',
          subset="validation",
          color_mode="rgb",
          target_size = (331,331), # image height , image width
          class_mode="categorical",
          batch size=32,
          shuffle=True,
          seed=42,
)
```

المخرجات:

```
Found 8178 validated image filenames belonging to 120 classes.
Found 2044 validated image filenames belonging to 120 classes.
```

دعونا نرى كيف تبدو دفعة واحدة من البيانات.

```
x,y = next(train_generator)
x.shape # input shape of one record is (331,331,3) , 32: is the batch
size
```

المخرجات:

```
(32, 331, 331, 3)
```

دعونا نرى كيف تبدو دفعة واحدة من البيانات.

```
a = train_generator.class_indices
class_names = list(a.keys()) # storing class/breed names in a list


def plot_images(img, labels):
          plt.figure(figsize=[15, 10])
          for i in range(25):
```

```
                    plt.subplot(5, 5, i+1)
                    plt.imshow(img[i])
                    plt.title(class_names[np.argmax(labels[i])])
                    plt.axis('off')

plot_images(x,y)
```

المخرجات:

بناء النموذج

هذه هي الخطوة الرئيسية حيث يتم بناء نموذج الالتفاف العصبي.

```
# load the InceptionResNetV2 architecture with imagenet weights as
base
base_model = tf.keras.applications.InceptionResNetV2(
                                                    include top=False,
                                                    weights='imagenet',
                                                    input_shape=(331,331,3)
                                                    )

base_model.trainable=False
# For freezing the layer we make use of layer.trainable = False
# means that its internal state will not change during training.
# model's trainable weights will not be updated during fit(),
# and also its state updates will not run.

model = tf.keras.Sequential([
                    base model,
                    tf.keras.layers.BatchNormalization(renorm=True),
                    tf.keras.layers.GlobalAveragePooling2D(),
                    tf.keras.layers.Dense(512, activation='relu'),
```

```
                tf.keras.layers.Dense(256, activation='relu'),
                tf.keras.layers.Dropout(0.5),
                tf.keras.layers.Dense(128, activation='relu'),
                tf.keras.layers.Dense(120, activation='softmax')
        ])
```

BatchNormalization:

- إنها تقنية تسوية تتم على دفعات صغيرة mini-batches بدلاً من مجموعة البيانات الكاملة.
- يتم استخدامه لتسريع التدريب واستخدام معدلات التعلم العالي higher learning rates.
- يحافظ على متوسط الإخراج بالقرب من 0 والانحراف المعياري الناتج بالقرب من 1.

GlobalAveragePooling2D:

- يأخذ موترًا بالحجم (عرض الإدخال) x (ارتفاع الإدخال) x (قنوات الإدخال) ويحسب متوسط قيمة جميع القيم عبر مصفوفة (عرض الإدخال) x (ارتفاع الإدخال) لكل من (قنوات الإدخال).
- يتم تقليل أبعاد الصور عن طريق تقليل عدد البكسل في الإخراج من طبقة الشبكة العصبية السابقة.
- باستخدام هذا نحصل على موتر أحادي البعد من الحجم (قنوات الإدخال) كإخراج.
- عملية تجميع المتوسط العالمي ثنائي البعد 2D Global average pooling. هنا "العمق Depth" = "الفلاتر Filters"

طبقات كثيفة Dense Layers: هذه الطبقات عبارة عن طبقات شبكة عصبية منتظمة متصلة بالكامل fully connected بعد الطبقات التلافيفية.

طبقة التسرب Drop out layer: تُستخدم أيضًا دالتها لإسقاط بعض الخلايا العصبية بشكل عشوائي من وحدة الإدخال لمنع الضبط الزائد overfitting. تشير القيمة 0.5 إلى أنه يجب إسقاط 0.5 جزء من الخلايا العصبية.

تجميع النموذج:

قبل تدريب نموذجنا، نحتاج أولاً إلى تكوينه ويتم ذلك بواسطة model.compile() الذي يحدد دالة الخطأ loss function والمحسّنات optimizers والمقاييس metrics للتنبؤ prediction.

```
model.compile(optimizer='Adam',loss='categorical_crossentropy',metrics
=['accuracy'])
# categorical cross entropy is taken since its used as a loss function
for
```

```
# multi-class classification problems where there are two or more
output labels.
# using Adam optimizer for better performance
# other optimizers such as sgd can also be used depending upon the
model
```

عرض تقرير موجز عن النموذج

من خلال عرض الملخص summary، يمكننا التحقق من نموذجنا للتأكد من أن كل شيء على ما هو متوقع.

```
model.summary()
```

المخرجات:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 9, 9, 1536)        54336736
_________________________________________________________________
batch_normalization_203 (Bat (None, 9, 9, 1536)        10752
_________________________________________________________________
global_average_pooling2d (Gl (None, 1536)              0
_________________________________________________________________
dense (Dense)                (None, 512)               786944
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 120)               15480
=================================================================
Total params: 55,314,136
Trainable params: 969,720
Non-trainable params: 54,344,416
_________________________________________________________________
```

تحديد عمليات callbacks للحفاظ على أفضل النتائج:

Callback: هو كائن يمكنه تنفيذ إجراءاتفي مراحل مختلفة من التدريب (على سبيل المثال ، في بداية أو نهاية فترة ما ، قبل أو بعد دفعة واحدة ، إلخ).

```
early = tf.keras.callbacks.EarlyStopping( patience=10,
                                          min_delta=0.001,
                                          restore_best_weights=True)
# early stopping call back
```

تدريب النموذج Training the model: يعني أننا نجد مجموعة من القيم للأوزان والتحيزات التي لها خطأ منخفضفي المتوسط عبر جميع السجلات.

```
batch_size=32
STEP_SIZE_TRAIN = train_generator.n//train_generator.batch_size
STEP_SIZE_VALID =
validation_generator.n//validation_generator.batch_size

# fit model
history = model.fit(train_generator,

          steps_per_epoch=STEP_SIZE_TRAIN,

          validation_data=validation_generator,

          validation steps=STEP SIZE VALID,
                                                epochs=25,
                                                callbacks=[early]
```

المخرجات:

```
Epoch 1/25
255/255 [==============================] - 3267s 13s/step - loss: 1.5007 - accuracy: 0.6806 - val_loss: 0.3956 - val_accuracy: 0.8938
Epoch 2/25
255/255 [==============================] - 122s 480ms/step - loss: 0.5099 - accuracy: 0.8729 - val_loss: 0.4213 - val_accuracy: 0.8948
Epoch 3/25
255/255 [==============================] - 127s 497ms/step - loss: 0.4416 - accuracy: 0.8826 - val_loss: 0.3888 - val_accuracy: 0.8978
Epoch 4/25
255/255 [==============================] - 127s 499ms/step - loss: 0.3915 - accuracy: 0.8945 - val_loss: 0.4146 - val_accuracy: 0.8958
Epoch 5/25
255/255 [==============================] - 127s 500ms/step - loss: 0.3432 - accuracy: 0.9057 - val_loss: 0.3889 - val_accuracy: 0.9008
Epoch 6/25
255/255 [==============================] - 126s 495ms/step - loss: 0.3053 - accuracy: 0.9141 - val_loss: 0.3808 - val_accuracy: 0.9112
Epoch 7/25
255/255 [==============================] - 126s 494ms/step - loss: 0.2923 - accuracy: 0.9152 - val_loss: 0.4112 - val_accuracy: 0.8968
Epoch 8/25
255/255 [==============================] - 128s 500ms/step - loss: 0.2738 - accuracy: 0.9223 - val_loss: 0.4267 - val_accuracy: 0.8988
Epoch 9/25
255/255 [==============================] - 128s 501ms/step - loss: 0.2507 - accuracy: 0.9243 - val_loss: 0.4326 - val_accuracy: 0.9048
Epoch 10/25
255/255 [==============================] - 128s 501ms/step - loss: 0.2437 - accuracy: 0.9293 - val_loss: 0.4429 - val_accuracy: 0.9028
Epoch 11/25
255/255 [==============================] - 126s 495ms/step - loss: 0.2317 - accuracy: 0.9317 - val_loss: 0.4183 - val_accuracy: 0.9038
Epoch 12/25
255/255 [==============================] - 127s 498ms/step - loss: 0.2233 - accuracy: 0.9348 - val_loss: 0.4374 - val_accuracy: 0.9023
Epoch 13/25
255/255 [==============================] - 127s 500ms/step - loss: 0.2054 - accuracy: 0.9394 - val_loss: 0.4325 - val_accuracy: 0.9117
Epoch 14/25
255/255 [==============================] - 128s 501ms/step - loss: 0.1867 - accuracy: 0.9417 - val_loss: 0.4948 - val_accuracy: 0.9062
Epoch 15/25
255/255 [==============================] - 128s 501ms/step - loss: 0.1786 - accuracy: 0.9438 - val_loss: 0.4749 - val_accuracy: 0.8973
Epoch 16/25
255/255 [==============================] - 127s 496ms/step - loss: 0.1790 - accuracy: 0.9449 - val_loss: 0.4740 - val_accuracy: 0.9023
```

حفظ النموذج

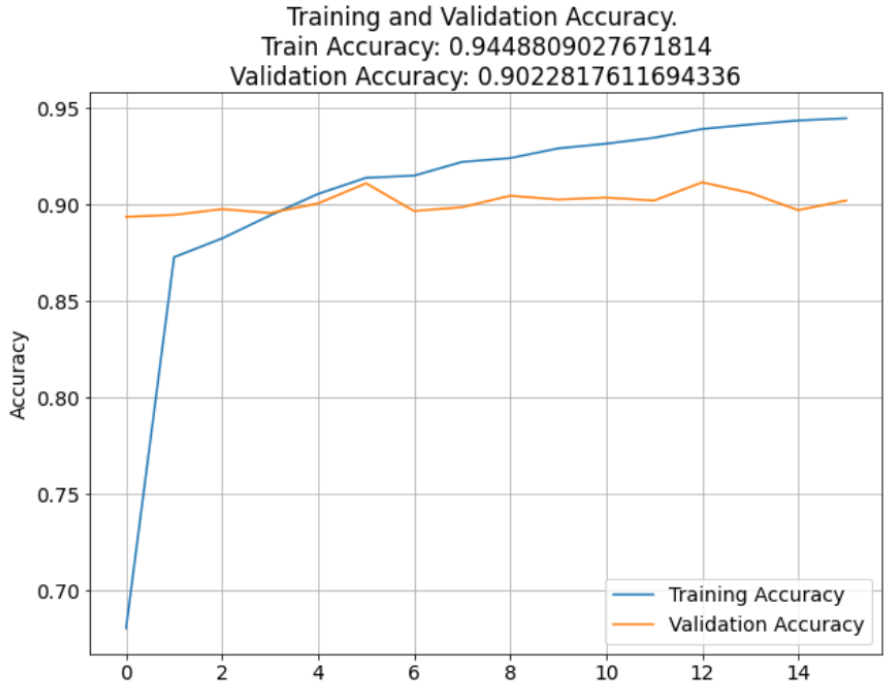
يمكننا حفظ النموذج لمزيد من الاستخدام.

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

رسم أداء النموذج

```
# store results
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']


# plot results
# accuracy
plt.figure(figsize=(10, 16))
plt.rcParams['figure.figsize'] = [16, 9]
plt.rcParams['font.size'] = 14
plt.rcParams['axes.grid'] = True
plt.rcParams['figure.facecolor'] = 'white'
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.title(f'\nTraining and Validation Accuracy. \nTrain Accuracy:
                  {str(acc[-1])}\nValidation Accuracy: {str(val_acc[-
1])}')
```

المخرجات:




```
# loss
plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
```

```
plt.title(f'Training and Validation Loss. \nTrain Loss:
                    {str(loss[-1])}\nValidation Loss: {str(val_loss[-
1])}')
plt.xlabel('epoch')
plt.tight_layout(pad=3.0)
plt.show()
```

تم أيضًا تكوين رم بياني للتدريب مقابل دقة التحقق من الصحة والخطأ. يشير الرسم البياني إلى أن دقة التحقق من الصحة والتدريب كانت متسقة تقريبًا مع بعضها البعض وأكثر من 90٪. يعد خطأ نموذج CNN رسمًا بيانيًا متخلفًا سلبيًا يشير إلى أن النموذج يتصرف كما هو متوقع مع خطأ متناقص بعد كل حقبة.

تقييم دقة النموذج

```
accuracy_score = model.evaluate(validation_generator)
print(accuracy_score)
print("Accuracy: {:.4f}%".format(accuracy_score[1] * 100))

print("Loss: ",accuracy_score[0])
```

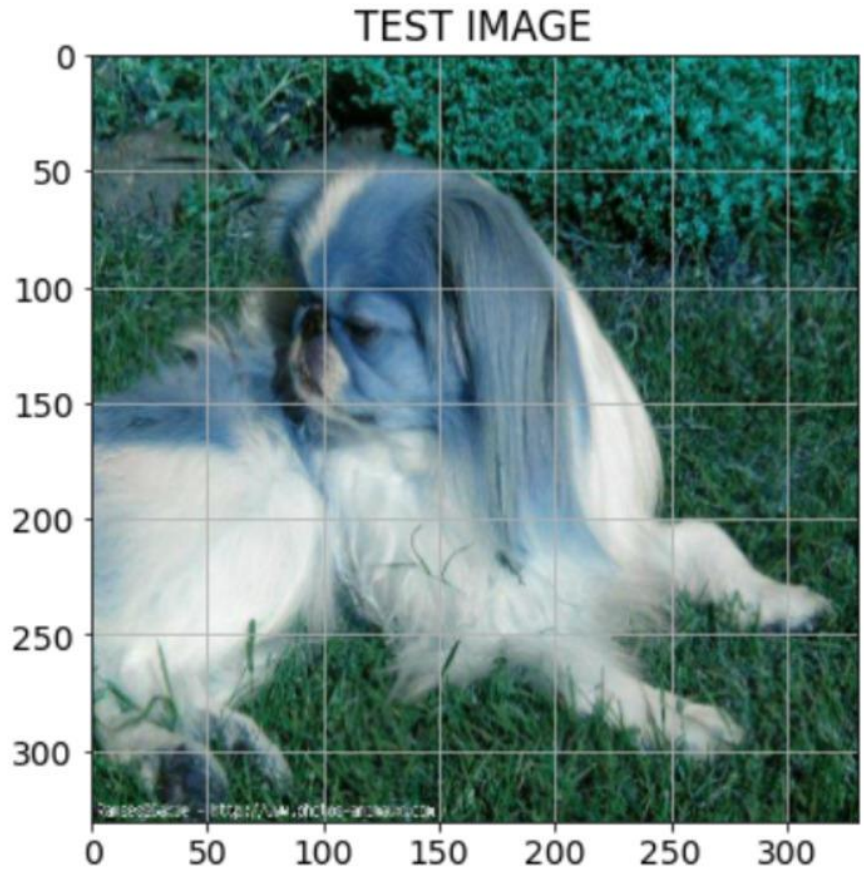
المخرجات:

```
64/64 [==============================] - 27s 412ms/step - loss: 0.3845 - accuracy: 0.9110
[0.38452133536338806, 0.9109588861465454]
Accuracy: 91.0959%
Loss:  0.38452133536338806
```

عرض صورة الاختبار

```
test_img_path = test_path+"/000621fb3cbb32d8935728e48679680e.jpg"

img = cv2.imread(test_img_path)
resized_img = cv2.resize(img, (331, 331)).reshape(-1, 331, 331, 3)/255

plt.figure(figsize=(6,6))
plt.title("TEST IMAGE")
plt.imshow(resized_img[0])
```

```
<matplotlib.image.AxesImage at 0x7f84dddde350>
```



عمل تنبؤات على بيانات الاختبار

```
predictions = []

for image in sample.id:
          img = tf.keras.preprocessing.image.load_img(test_path +'/'+
image)
          img = tf.keras.preprocessing.image.img_to_array(img)
          img = tf.keras.preprocessing.image.smart_resize(img, (331,
331))
          img = tf.reshape(img, (-1, 331, 331, 3))
          prediction = model.predict(img/255)
          predictions.append(np.argmax(prediction))

my_submission = pd.DataFrame({'image_id': sample.id, 'label':
predictions})
my_submission.to_csv('submission.csv', index=False)

# Submission file ouput
print("Submission File: \n---------------\n")
print(my_submission.head()) # Displaying first five predicted output
```

```
Submission File:
---------------

                                  image_id  label
0  000621fb3cbb32d8935728e48679680e.jpg     61
1  00102ee9d8eb90812350685311fe5890.jpg     94
2  0012a730dfa437f5f3613fb75efcd4ce.jpg     40
3  001510bc8570bbeee98c8d80c8a95ec1.jpg     88
4  001a5f3114548acdefa3d4da05474c2e.jpg     70
```

المصدر:

https://www.geeksforgeeks.org/multiclass-image-classification-using-transfer-learning/?ref=gcse

المصادر:

2. Transfer Learning in Keras with Computer Vision Models, Jason Brownlee, https://machinelearningmastery.com/transfer-learning-for-deep-learning/.
3. Transfer learning and the art of using Pre-trained Models in Deep Learning, Dishashree26 Gupta, https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/
4. Understanding Transfer Learning for Deep Learning, Pranshu Sharma, https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/
5. Transfer Learning in Keras with Computer Vision Models, Jason Brownlee, https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/
6. Image Recognition: Dogs vs Cats! (92%), https://thedatafrog.com/en/articles/dogs-vs-cats/.
7. Image Recognition with Transfer Learning (98.5%), https://thedatafrog.com/en/articles/image-recognition-transfer-learning/
8. Dog's Breed Identification using Deep Learning, https://techvidvan.com/tutorials/dog-breed-classification/
9. Pneumonia Detection using Deep Learning, https://www.geeksforgeeks.org/pneumonia-detection-using-deep-learning/.
10. Detecting COVID-19 From Chest X-Ray Images using CNN, https://www.geeksforgeeks.org/detecting-covid-19-from-chest-x-ray-images-using-cnn/
11. 3D Shape Detection System, https://medium.com/ai-techsystems/3d-shape-detection-system-d7e34286a2b1.
12. Lung Cancer Detection Using Transfer Learning, https://www.geeksforgeeks.org/lung-cancer-detection-using-transfer-learning/
13. Detecting If a Person is Wearing a Mask or Not Using CNN, Aryan Garg, https://www.analyticsvidhya.com/blog/2022/10/detecting-if-a-person-is-wearing-a-mask-or-not-using-cnn/
14. Transfer learning with TensorFlow, Jafar Isbarov, https://www.analyticsvidhya.com/blog/2021/11/transfer-learning-with-tensorflow/

15. Food Classification Using Transfer Learning And TensorFlow, Mrinal Singh Walia, https://www.analyticsvidhya.com/blog/2021/05/food-classification-using-transfer-learning-and-tensorflow/
16. Introduction to Transfer Learning using MNIST, Barney Darlington, https://www.analyticsvidhya.com/blog/2021/05/transfer-learning-using-mnist/
17. Deep Learning in Medical: Classification of Malaria Infected Blood Cells with ResNet, Muhammad Arnaldo, https://www.analyticsvidhya.com/blog/2021/12/deep-learning-in-medical-field/
18. Image Prediction Using a Pre-trained Model, Dipesh Shrestha, https://www.analyticsvidhya.com/blog/2022/09/image-prediction-using-a-pre-trained-model/
19. Coca-Cola Bottle Image Recognition (with Python code), Thomas Tsuma, https://www.analyticsvidhya.com/blog/2021/01/coca-cola-bottle-image-recognition/
20. Multiclass image classification using Transfer learning, https://www.geeksforgeeks.org/multiclass-image-classification-using-transfer-learning/?ref=gcse.