الشبكات العصبية

م. أحمد شمسان أحمد علي .

❖ مكتبة (TensorFlow):

- ### ماهي Tesorflow :

- TensorFlow هي منصة مفتوحة المصدر توفر للمطورين وعلماء البيانات الأدوات التي يحتاجونها لبناء نماذج تعلم الآلة، بداية من معالجة البيانات وتجهيزها، إلى التدريب وحتى التشغيل، وتدعم العديد من لغات البرمجة عن طريق مكتبات مخصصة لكل لغة مثل بايثون و جافاسكريبت وسي و جافا وغيرها من اللغات، وإن كانت لغة بايثون هي اللغة الأكثر استخدامًا والأكثر دعمًا [1].
- طورت شركة جوجل منصة تنسرفلو TensorFlow عن طريق فريقها Google Brain في عام 2015 كي تكون بديلًا مفتوح المصدر للنظام السابق الذي كان يُستخدم في تدريب خوارزميات تعلم الآلة والمعروف باسم ديست بليفDistBelief ، ومن أبرز مميزات تنسرفلو TensorFlow دعم مبدأ توزيع التدريب على عدة أجهزة لتعزيز كفاءة الأداء، وسهولة تعلمها، وكونها منصة مفتوحة المصدر مما يسمح للجميع بالمساهمة في تحسينها واستخدامها في مختلف المشاريع .
- كما تتميز تنسرفلو TensorFlow بقدرتها على استغلال مختلف العتاد الحاسوبي مثل وحدة المعالجة المركزية CPU ووحدة المعالجة الرسومية GPU ومسرعات التدريب المختلفة، وهذا يجلعها تتفوق على بعض المكتبات التقليدية مثل ساي كيت ليرن Scikit learn الشهيرة التي توفر العديد من خوارزميات تعلم الآلة التقليدية ولكنها لا تدعم استخدام وحدات المعالجة الرسوميةGPU ، ناهيك عن التدريب الموزع، فمن غير الممكن تدريب نماذج ذكاء اصطناعي واسعة النطاق باستخدام مكتبات لا تدعم هذه الإمكانيات المتقدمة مثل التدريب الموزع أو المسرعات.

- ### كيف يمكن تثبيت Tensorflow :

تثبيت TensorFlow على Windows [2]:

.1متطلبات النظام

- نظام التشغيل:
  Windows 10 19044أو أعلى (64 بت). يتوافق هذا مع إصدار21H2 Windows 10 .
- المستندات المطلوبة:
  - تنزيل أحدث تحديث لنظام Windows 10.
  - تثبيت WSL2.
  - إعداد دعم وحدة معالجة الرسومات NVIDIA® في WSL2.

.2إعداد وحدة معالجة الرسوميات(GPU)

يمكنك تخطي هذا القسم إذا كنت تخطط لتشغيل TensorFlow على وحدة المعالجة المركزية (CPU) فقط.

- تثبيت برنامج تشغيل وحدة معالجة الرسومات:NVIDIA

إذا لم يكن لديك برنامج تشغيل NVIDIA مثبتًا، يمكنك تنزيله من موقع NVIDIA الرسمي .

للتحقق من وجود برنامج التشغيل، استخدم الأمر التالي:

```
nvidia-smi
```

إذا عرض الأمر معلومات عنGPU ، فهذا يعني أن برنامج التشغيل مثبت بشكل صحيح.

.3تثبيتTensorFlow

- ترقية:pip

TensorFlowيتطلب إصدارًا حديثًا من .pip قم بترقية pip باستخدام الأمر التالي:

```
pip install --upgrade pip
```

- تثبيت:TensorFlow

بناءً على احتياجاتك، قم بتثبيت TensorFlow باستخدام أحد الأمرين التاليين:

- للمستخدمين الذين يعتمدون على:GPU

```
pip install tensorflow[and-cuda]
```

- للمستخدمين الذين يعتمدون على CPU فقط:

```
pip install tensorflow
```

.4التحقق من التثبيت

- التحقق من إعداد وحدة المعالجة المركزية:(CPU)

قم بتشغيل الكود التالي للتأكد من أن TensorFlow يعمل بشكل صحيح على وحدة المعالجة المركزية:

```
python3 -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```

إذا تم إرجاع موتر(Tensor) ، فهذا يعني أن TensorFlow قد تم تثبيته بنجاح.

- التحقق من إعداد وحدة معالجة الرسوميات:(GPU)

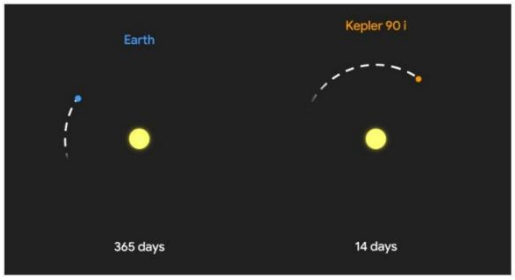
قم بتشغيل الكود التالي للتأكد من أن TensorFlow يمكنه الوصول إلى وحدة معالجة الرسوميات:

```
python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
```

إذا تم إرجاع قائمة بأجهزةGPU ، فهذا يعني أن TensorFlow قد تم تثبيته بنجاح ويمكنه استخدامGPU .

- ## مميزات تقدمها للمبرمجين [3] :

  - ### منصة قوية لتعلم الآلة



الشكل (1): نظام كبلر-90-اي

تعتبر تينسر فلو أفضل منصة مفتوحة المصدر لتعلم الآلة وخاصة عند وجود بيانات ضخمة كما في حالة مساعدة الأطباء في الكشف عن اعتلال شبكية مرضى السكري لمنع العمى، وأيضًا عند إشعار السلطات بعمليات إزالة الغابات بشكل غير قانوني للمساعدة في حمايتها، حيث تجمع بين برنامج التعلم العميق ألفا غو AlphaGoوالرؤية السحابية لجوجل Google Cloud Vision وقد قامت الوكالة الفضائية ناسا باستخدامها في اكتشاف الكوكب الثامن الذي يدور حول النجم كبلر-90 Kepler-90 خارج المجموعة الشمسية والذي يدعى كبلر-90-اي Kepler-90i.بذلك يكون كبلر-90 هو النظام الآخر الوحيد الذي نعرفه والذي يحتوي على ثمانية كواكب في مداره كما في الشكل التالي[4].

- منفّذ ايجرEager execution

```
optimizer = tf.train.MomentumOptimizer(...)
for (x,y) in dataset.make_one_shot_iterator():
    with tf.GradientTape() as g:
        y_ = model(x)
        loss = loss_fn(y,y_)
        grads= g.gradient(y_, model.variables)
        optimizer.apply_gradients(zip(grads, model.variables))
```

في حال كان هناك استياء سابق حيال تينسر فلو كونها تجبرنا على البرمجة بأسلوب أكاديمي وليس كمطورين. يمكن العودة لها الآن حيث زُودت بمنفّذ ايجر الذي سيتيح لنا التفاعل معها كمبرمج بايثون. فهو يعمل على تقييم العمليات وتصحيح الأخطاء مباشرة سطر بسطر والتبليغ عنها من خلال فحص النماذج قيد التنفيذ واختبار التغييرات وذلك لاستخدامه مصحح الأخطاء القياسي لبايثون. ويعيد قيمًا ملموسة بدلاً من إنشاء رسوم بيانية ضخمة لاعتماده أيضًا على تدفق تحكم بايثون[4].

- إمكانية بناء الشبكات العصبونية سطر بسطر

```
import tensorflow as tf

L = tf.keras.layers

model = tf.keras.Sequential([
    L.Reshape((28, 28, 1)),
    L.Conv2D(32, 5, activation=tf.nn.relu),
    L.MaxPooling2D((2, 2), (2, 2)),
    L.Conv2D(64, 5, activation=tf.nn.relu),
    L.MaxPooling2D((2, 2), (2, 2)),
    L.Flatten(),
    L.Dense(1824, activation=tf.nn.relu),
    L.Dropout(0.4),
    L.Dense(18)
])
```

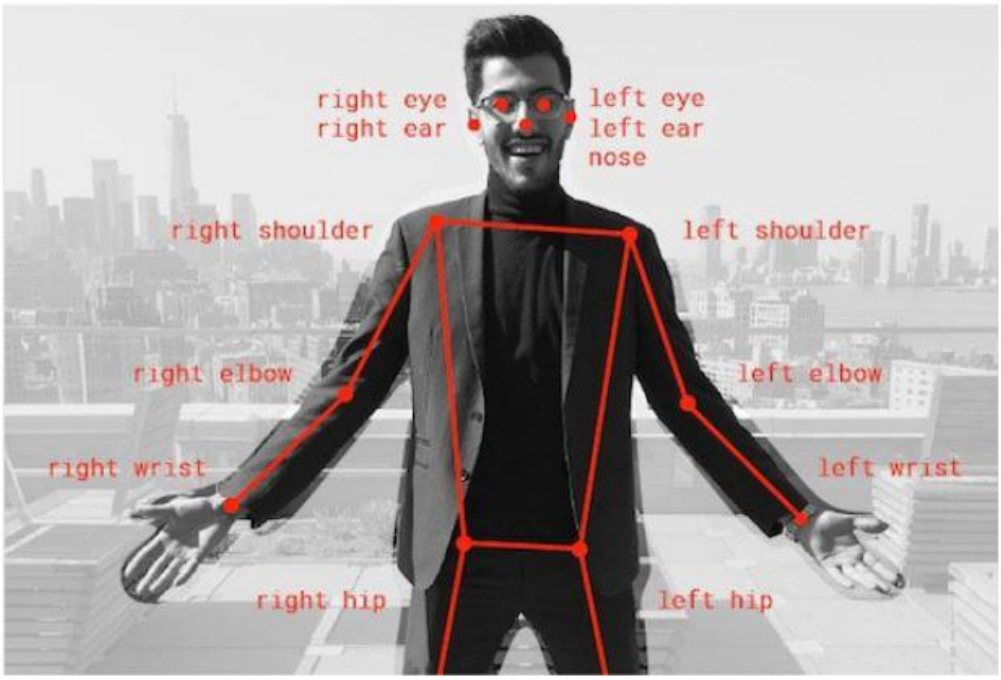
دمجت تينسر فلو مكتبة كيراس ضمن بيئتها منتجة بذلك حزمة كيراس للتعلم العميق من جوجل tf.keras والتي يمكن من خلالها بناء أي شبكة عصبونية بطريقة سهلة متسلسلة مع تحديد جميع المعاملات الضرورية وذلك بكتابة بضعة أسطر برمجية فقط في صفحة جديدة من منصة جوجل كولاب . google colab [4]

- تينسر فلو ليست فقط بايثون.

لم تعد تينسر فلو خاصة بلغة البايثون وحسب وإنما أصبحت تتضمن عدة لغات برمجية أخرى مثل جافا سكريبت JavaScript وأر R و سويفت Swift والكثير[4].

- يمكن القيام بكل شيء ضمن المتصفح

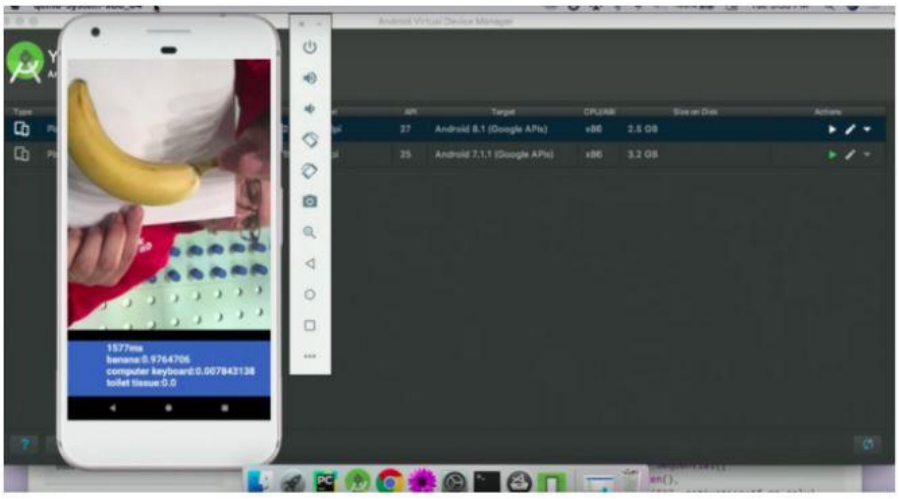
في حال أردنا استخدام جافا سكريبت Javascriptلتدريب وتنفيذ النماذج في المتصفح، يمكننا استخدام مكتبتها الخاصة بتينسر فلو TensorFlow.js. فقد تم استخدام هذه المكتبة لعمل تقنية تقوم بتخمين مفاصل جسم الإنسان الرئيسية المتحركة في الزمن الحقيقي. يمكنك تجربتها من هنا بعد فتح الكاميرا[4].



الشكل(3): تقنية تخمين مفاصل الانسان المتحركة المكتوبة باستخدام المكتبة الخاصة بالجافا سكريبت للتينسرفلو TensorFlow.js

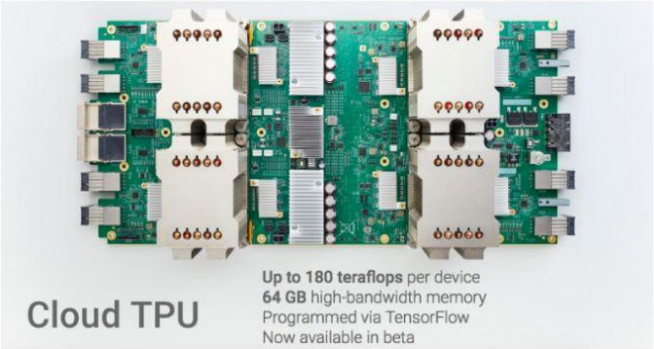
- **إصدار خفيف للأجهزة الصغيرة**

يسمح لنا الإصدار الخفيف من تينسر فلو)تينسر فلو لايت TensorFlow Lite(بتنفيذ النماذج على أنواع مختلفة من الأجهزة بما في ذلك أجهزة الهاتف النقال والجيل الجديد من الإنترنت الذي يتيح التفاهم بين الأجهزة المترابطة مع بعضها أو ما يسمى إنترنت الأشياء 'IoT' Internet of Thingsمتيحًا بذلك سرعة أكبر بثلاث مرات فى الاستدلال عن تينسر فلو العادي. أي يمكننا الآن البدء في الحصول على التعلم الآلي على هواتفنا أو على الكمبيوتر المصغر رازبيري باي .Raspberry Piوكما نرى في الصورة التالية فقد استغرق حوالي 1.6 ثانية ليستدل على أنها موزة[4].



▪ تحسين المعدات الفيزيائية

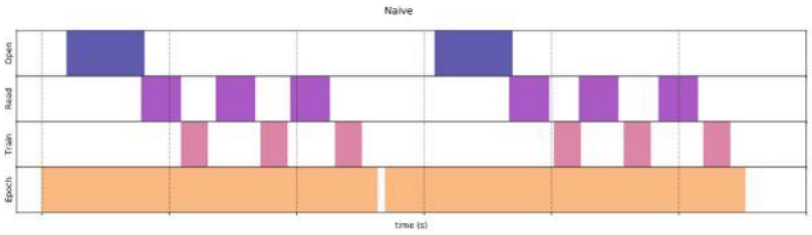
ربما يعاني الجميع من بطء وحدة المعالجة المركزية CPU عند تدريب الشبكات العصبونية. الآن أصبح بالإمكان تدريب وتشغيل نماذج تعلم الآلة بشكل أسرع من أي وقت مضى من خلال استخدام معدات فيزيائية خصصت لتعمل مع وحدة معالجة المصفوفة متعددة الأبعاد السحابية (TPU Tensor Processing Unit Cloud) والتي تم تصميمها لتشغيل نماذج تعلم الآلة مع خدمات الذكاء الاصطناعي على الحوسبة السحابية الخاصة بجوجل[4].Google Cloud



الشكل(5): وحدة معالجة تينسر السحابية

▪ مسار تدفق البيانات أكثر تطورًا

يمكن لوحدات معالجة الرسومات GPUs ووحدة معالجة المصفوفة متعددة الأبعاد السحابية TPU أن تقلل بشكل جذري من الوقت اللازم لتنفيذ خطوة تدريب واحدة. ولكن تحقيق أعلى مستوى من الأداء يتطلب وجود مسار تدفق بيانات فعال يوفر البيانات للخطوة التالية قبل انتهاء الخطوة الحالية. حققت تينسر فلو ذلك باستخدام مساحة



الشكل(6): زمن التنفيذ دون استخدام مسار تدفق بيانات فعّال هو 0.25 ثانية – مأخوذة من موقع تينسرفلو

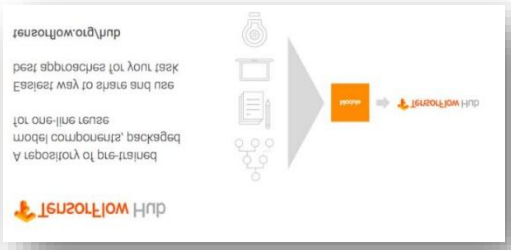
الاسم Namespace

المسمى tf.data الذي جعل معالجة الدخل في تينسر فلو أكثر تعبيرًا وفعالية، باستخدام مسارات تدفق بيانات سريعة ومرنة وسهلة الاستخدام ومتزامنة مع التدريب[4].

الشكل(7): زمن التنفيذ بعد استخدام مسار تدفق بيانات فعّال هو 0.20 ثانية – مأخوذة من موقع تينسر فلو

- **لا داعي للبدء من الصفر**

إحدى أهم الأشياء الأساسية في تطوير البرامج هي فكرة مخزن الشيفرة المشتركة. وقد وفرت تينسر فلو مخزن لمحبي تعلم الآلة يدعى تينسرفلو-هاب TensorFlow Hub والذي يحتوي على مكونات نماذج التعلم الآلي القابلة لإعادة الاستخدام والمدربة سابقا ليتم إعادة استخدامها من خلال سطر واحد فقط من الشيفرة البرمجية. كما وفرت أيضًا مدونة TensorFlow Blog وقناة على اليوتيوب خاصة بها TensorFlow Youtube Channel كمصادر جديدة للتعلم والمشاركة[4].



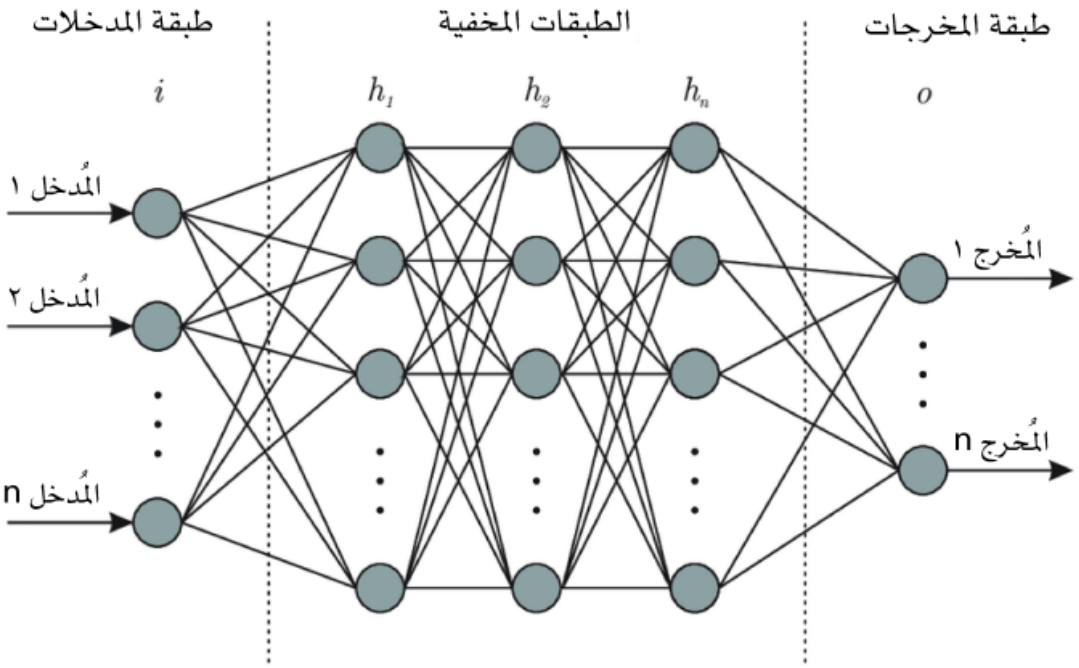
الشكل(8): مخزن تينسر فلو

- مثال يستخدم Tensorflow :

```python

import tensorflow as tf
import numpy as np

# توليد 100 نقطة عشوائية باستخدام numpy
x_data = np.random.rand(100)
# الصيغة أدناه مكافئة لخط مستقيم به ميل 0.1 وإزاحة 0.2
y_data = x_data * 0.1 + 0.2

# بناء نموذج خطي
b = tf.Variable(0.)
k = tf.Variable(0.)
y = k * x_data + b

# وظيفة التكلفة التربيعية
loss = tf.reduce_mean(tf.square(y_data-y))
# تحديد طريقة النسب التدرج للمحسن التدريب
optimizer = tf.train.GradientDescentOptimizer(0.2)  # المعلمة تشير إلى كفاءة التحسين
# تقليل وظيفة التكلفة
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    for step in range(201):
        sess.run(train)
        if step % 20 == 0:
            print(sess.run([k, b]))
```

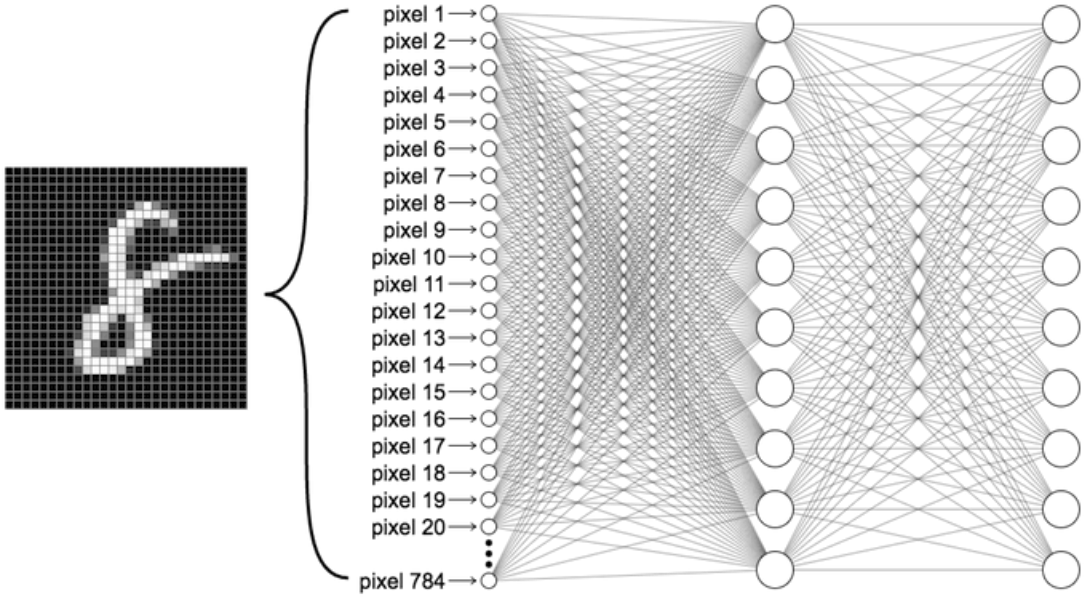
❖ شبكة عصبية بسيطة[5]:

يُعد تصميم الشبكة العصبية من أكثر الأمور المحيّرة في مجال التعلم العميق، عند تصميم الشبكة عادة ما تواجهنا العديد من الأسئلة مثل: ماهي أفضل قيمة لمعدل التعلم؟ كم عدد الطبقات المخفية (الوسطى) التي يجب أن تحتويها الشبكة؟ هل إسقاط الخلايا العصبية (dropout) مفيد للشبكة؟ لماذا تتلاشى مشتقة الخطأ عبر الشبكة(Vanishing Gradient) ؟
هذا المقال سيزيح الستار عن هذه الأسئلة المحيرة التي قد تواجهك عند تصميم الشبكة العصبية، وسيقوم بإرشادك خطوة بخطوة في كيفية اتخاذ قرارات ذكية بشأن بُنية الشبكة العصبية الخاصة بك.
١.البُنية الأساسية للشبكة العصبية



١.١ مُدخَلات الشبكة العصبية

- يُقصد بعدد مُدخلات الشبكة العصبية عدد الخصائص/السمات (Features) التي تستخدمها الشبكة العصبية لاستنتاج المُخرجات.
- سنحتاج لخلية عصبية واحدة لكل خاصية موجودة في المُدخلات، إذا كانت البيانات على شكل جدول فإن عدد الخلايا العصبية المطلوب يمثل عدد الخصائص (الأعمدة) في مجموعة البيانات الخاصة بك، لابد من تحديد هذه الخصائص بعناية و إزالة أي خصائص قد تحتوي على أنماط لا يمكن تعميمها خارج مجموعة التدريب) حتى لا تتسبب في الوصول لحالة فرط التخصيص((overfitting) ، إذا كانت البيانات عبارة عن صور فعدد الخلايا العصبية المطلوب في طبقة الإدخال يمثل أبعاد الصورة، فمثلاً في مجموعة بيانات MNIST أبعاد الصور هو 28 * 28 = 784 ، إذن سنحتاج إلى 784 خلية عصبية في طبقة الإدخال.



مثال على عدد الخلايا العصبية في طبقة الإدخال لمجموعة بياناتMNIST

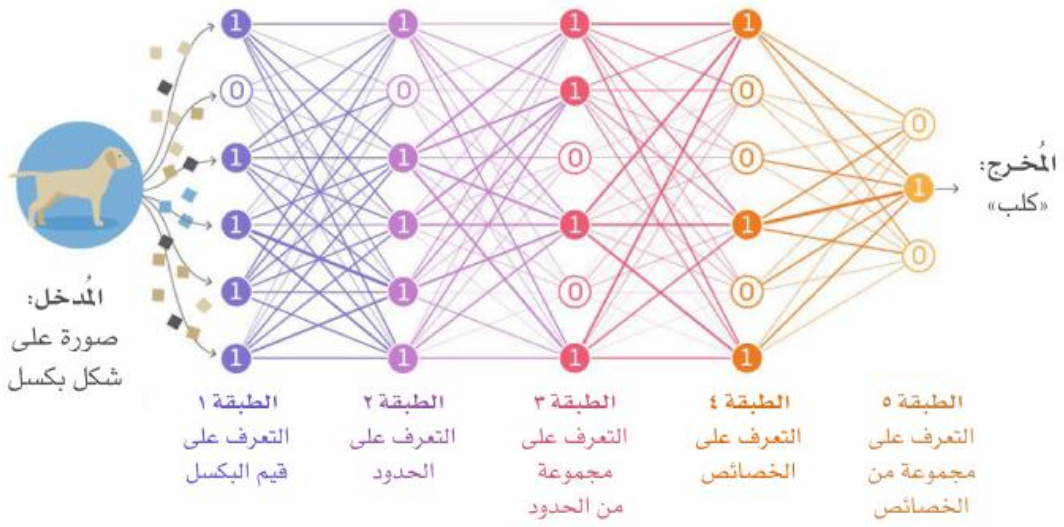
٢.١ مُخرَجات الشبكة العصبية

- عدد المُخرجات يُمثل عدد النتائج أو التنبؤات التي تود الحصول عليها من الشبكة.
- الانحدار :(Regression) في مشاكل الانحدار التي يكون فيه مُخرَج الشبكة قيمة واحدة (مثل: سعر المنزل) سنحتاج إلى خلية عصبية واحدة، أما في حالات الانحدار متعدد المُخرجات سنحتاج لخلية عصبية واحدة لكل قيمة متوقعة (مثلاً في مشكلة تحديد موقع كائن في صورة ما سنحتاج إلى ٤ خلايا عصبية في طبقة الإخراج لتحديد مكان المستطيل الذي يحيط بالكائن في الصورة، خلية عصبية لكل من (الإحداثي س، الإحداثي ص، عرض المستطيل، طول المستطيل.(

- التصنيف :(Classification) في حالة التصنيف الثنائي (مثل: تصنيف البريد إلى مزعج وغير مزعج) سنحتاج لخلية عصبية واحدة بحيث يُمَثّل الناتج احتمال انتماء المُدخَل للفئة الإيجابية. بالنسبة للتصنيف متعدد الفئات (على سبيل المثال: التعرف على كائنات مختلفة مثل سيارة أو كلب أو منزل وما إلى ذلك)، سنحتاج لخلية عصبية واحدة لكل فئة، سنستخدم دالة التنشيط سوفت ماكس في طبقة الإخراج/النهائية لضمان أن يكون مجموع جميع الخلايا العصبية في طبقة الإخراج يساوي واحد.

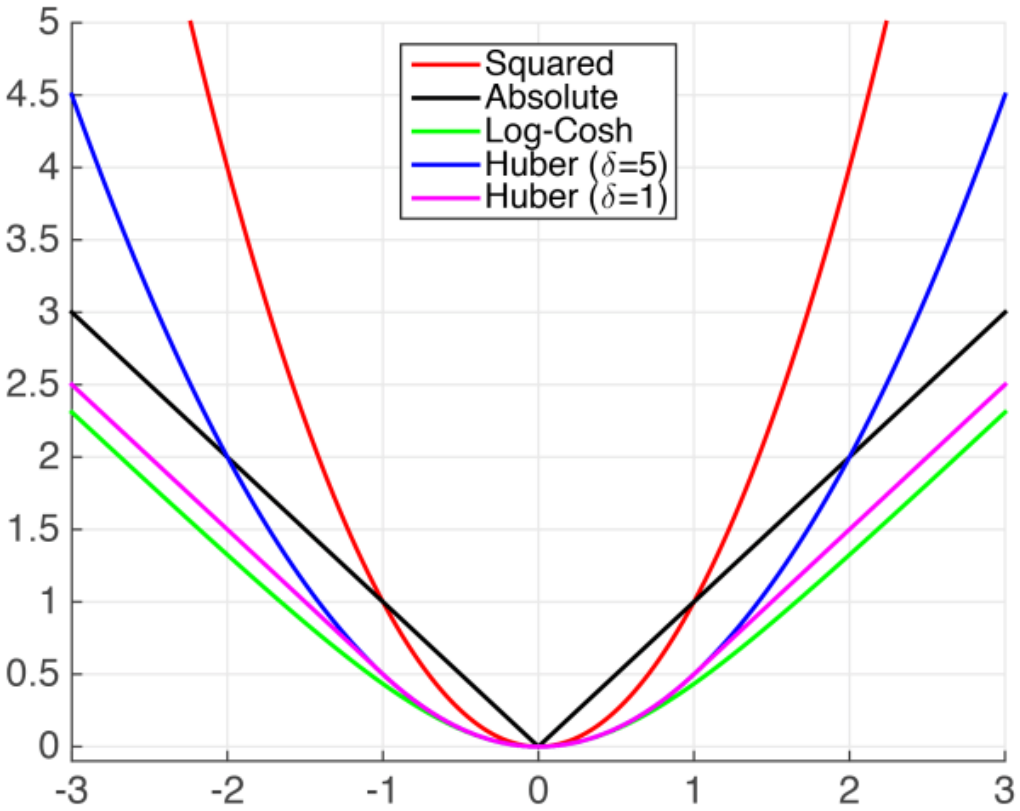
  ٣.١ الطبقات المخفية/الوسطى والخلايا العصبية في كل طبقة

- يعتمد عدد الطبقات المخفية/الوسطى وعدد الخلايا العصبية في كل طبقة بشكل كبير على المشكلة المُراد حلها وبنية الشبكة العصبية الخاصة بك. الهدف الأساسي هو الوصول إلى الشبكة العصبية المثالية بحيث لا تكون كبيرة جداً ولا صغيرة جداً بل مناسبة تماماً لمشكلتك.
- بشكل عام فإن استخدام طبقة واحدة إلى خمس طبقات مخفية/وسطى سوف يخدمك بشكل جيد في معظم المشاكل. عند العمل على بيانات على شكل صور أو صوت، قد تحتاج لأن تحتوي شبكتك ما بين عشرات إلى مئات من الطبقات، والتي قد لا تكون جميعها متصلة بشكل كامل. في هذه الحالة يمكنك استخدام نماذج مدربة مسبقاً مثل YOLO :و ResNetو VGG والتي تسمح لك باستخدام أجزاء كبيرة منها، ووضع النموذج الخاص بك على رأس هذه الشبكات ليتعلم الخصائص العليا المرتبطة بالمشكلة، في هذه الحالة سيكون عليك تدريب عدد أقل من الطبقات.
- بشكل عام فإن استخدام نفس العدد من الخلايا العصبية لجميع الطبقات المخفية سيكون كافياً. في بعض مجموعات البيانات وجود طبقة أولى كبيرة بمدخلات كبيرة ثم متابعتها بطبقات أصغر سيؤدي إلى أداء أفضل حيث يمكن للطبقة الأولى أن تتعلم الخصائص البسيطة للبيانات والتي يمكن أن تتجمع لتمثل خصائص ذات مستوى أعلى في الطبقات اللاحقة.



- في العادة إضافة عدد طبقات أكثر يؤدي إلى تحسين أكبر في الأداء مقارنة بإضافة المزيد من الخلايا العصبية في كل طبقة.
- يُوصَى بالبدء بعدد من إإلى ٥ طبقات ومن إإلى١٠٠ خلية عصبية في كل طبقة، ثم إضافة المزيد من الطبقات والخلايا العصبية بالتدريج حتى تبدأ في الوصول لحالة فرط التخصيص. بعد ذلك يمكنك تتبع دالة الخسارة (Loss function) والدقة لمعرفة أي عدد من الطبقات والخلايا يعطي أفضل النتائج.
- من الأمور التي يجب مراعاتها عند اختيار عدد قليل من الطبقات أو الخلايا العصبية أنه إذا كان الرقم صغيراً جداً، فلن تتمكن الشبكة من التعرف على الأنماط الأساسية الكامنة في بياناتك وبالتالي لن تكون ذات فائدة. هناك طريقة لمواجهة هذا الأمر وهو البدء بعدد كبير من الطبقات المخفية والخلايا العصبية ثم استخدام تقنية إسقاط الخلايا العصبية (Dropout) والتوقف المبكر للتدريب(early stopping) ؛ وذلك للسماح للشبكة العصبية باختيار الحجم المناسب لها بطريقة ذاتية، يُفضّل تجربة إعدادات وأرقام مختلفة وتتبع الأداء لتحديد حجم الشبكة المثالي لمشكلتك.
- يوصِي أندريه كاباثاي باتباع أسلوب الإفراط ثم المعادلة حاول أولاً الحصول على نموذج كبير بما يكفي ليصل لحالة فرط التخصيص (أي قم بالتركيز فقط على تقليل خطأ النموذج أثناء عملية التدريب)، بعد ذلك قم بضبط النموذج بشكل مناسب (أي تخلى عن بعض الدقة في النموذج مقابل تحسين تعميم النموذج وتحسين أداءه على بيانات التحقق).

  ٤.١ دالة الخسارة(Loss function)



- الانحدار :(Regression) دالة متوسط الخطأ التربيعي (Mean Squared Error) من أكثر دوال الخسارة استخداماً في خوارزميات التحسين في مشاكل الانحدار، إذا كان هناك عدد كبير من القيم المتطرفة في هذه الحالة يُفضّل استبدالها بدالة متوسط الخطأ المطلق (Mean Absolute Error)أو خطأ هوبر.
- التصنيف :(Classification) دالة الإنتروبيا المتقاطعة (Cross-entropy) هي خيار جيد في معظم الحالات.

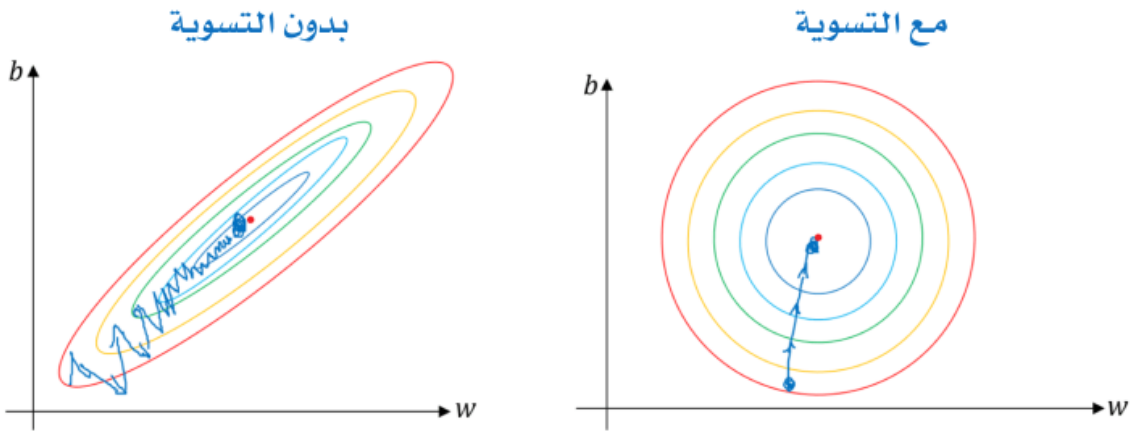
٥.١ حجم الحزمة التدريبية(Batch Size)

- استخدام أحجام كبيرة من الحزم التدريبية قد يكون خياراً جيداً لأنه يتيح استغلال قوة وحدات المعالجة الرسومية (GPUs) كما أنه يتيح معالجة أكبر عدد ممكن من عينات التدريب في كل مرة OpenAI .وجدت أن استخدام حزم تدريبية كبيرة (عشرات الآلاف من العينات في تصنيف الصور ونمذجة اللغة، وملايين العينات في التعليم التعزيزي) يُسهّل من تطوير النموذج والمعالجة المتوازية.
- هناك حالات يُفضّل فيها استخدام حزم تدريبية أصغر. وفقًا لهذه الورقة العلمية التي أعدها ماسترز ولوستشي فإن المميزات المكتسبة من تكبير حجم العينة مثل زيادة المعالجة المتوازية ومعالجة مجموعة أكبر من البيانات، يقابلها مميزات أخرى عند استخدام حزم أصغر من البيانات مثل زيادة تعميم النموذج واستخدام أقل للذاكرة العشوائية. أظهرت الورقة العلمية أن الأحجام الكبيرة للحزم التدريبية تقلل من نطاق قيم معدل التعلم (learning rate) للحصول على تقارب مستقر .(convergence) خلاصة الورقة هي أن استخدام الحزم التدريبية الأصغر أفضل، وأنه يمكن الحصول على أفضل أداء من خلال استخدام أحجام حزم تدريبية صغيرة تتراوح مابين ٢ إلى ٣٢.
- إذا كنت لا تعمل على مشكلة بحجم ونطاق كبير، فإنه من الأفضل البدء بحزم تدريبية صغيرة وزيادة حجم الدفعات بالتدريج ومراقبة الأداء لتحديد الحجم الأنسب.

٦.١ عدد الدورات التدريبية

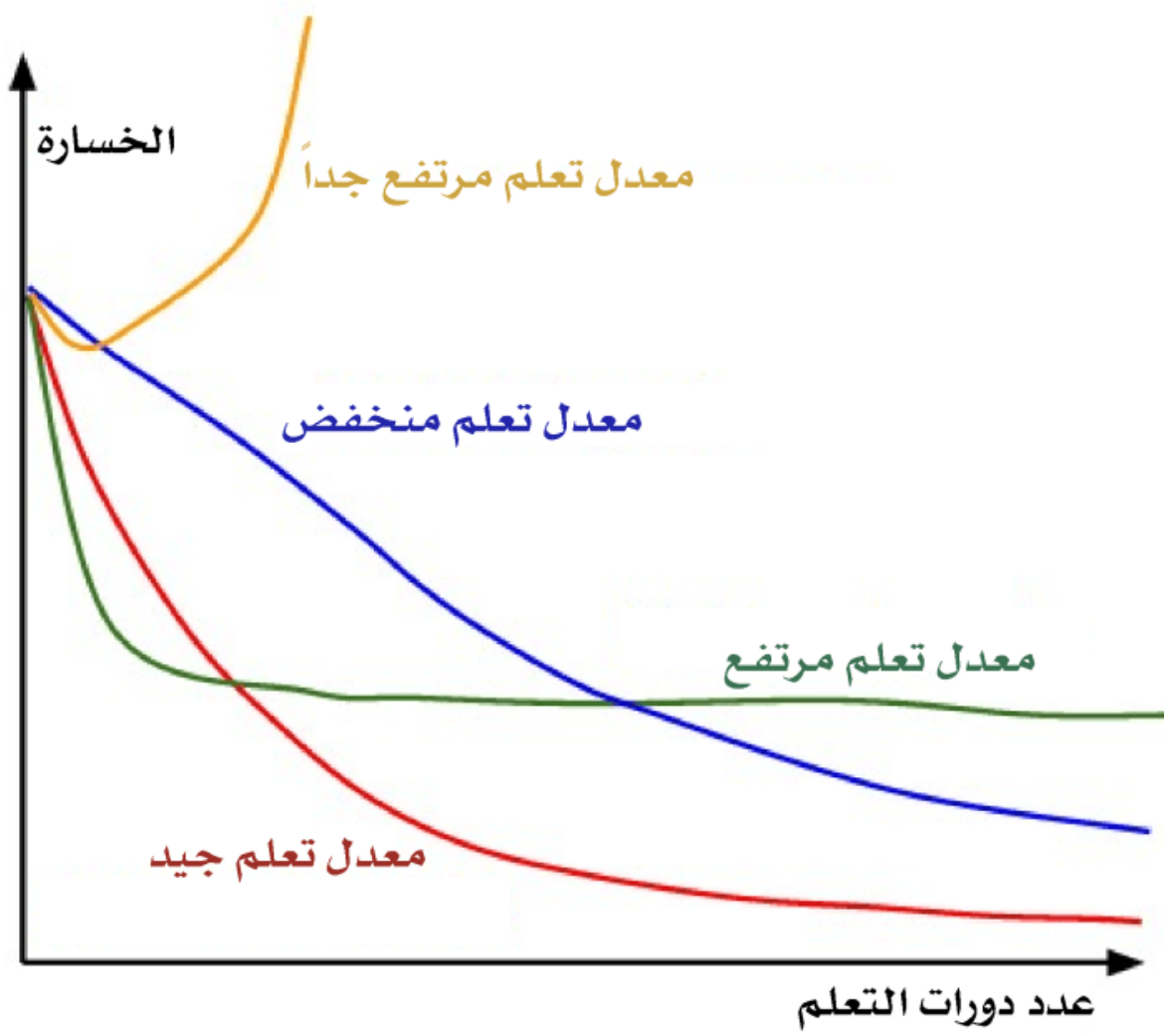
- من الأفضل البدء بعدد كبير من الدورات التدريبية واستخدام التوقف المبكر أثناء التدريب عندما يتوقف الأداء عن التحسن (انظر القسم الرابع تلاشي وانفجار مشتقة الخطأ).

٧.١ تسوية الخصائص



- قبل استخدام بياناتك كمدخلات للشبكة العصبية تحقق من أن جميع الخصائص على مقياس واحد فهذا يضمن الحصول على تقارب أسرع. عندما يكون لخصائصك مقاييس مختلفة (على سبيل المثال رواتب الموظفين بالآلاف وعدد سنوات الخبرة بالعشرات)، سيكون شكل دالة التكلفة أشبه بشكل الوعاء الممتد (الشكل على اليسار). هذا يعني أن خوارزمية التحسين الخاصة بك ستستغرق وقتًا طويلاً للتقارب مقارنة باستخدام الخصائص بعد التسوية (الشكل على اليمين).

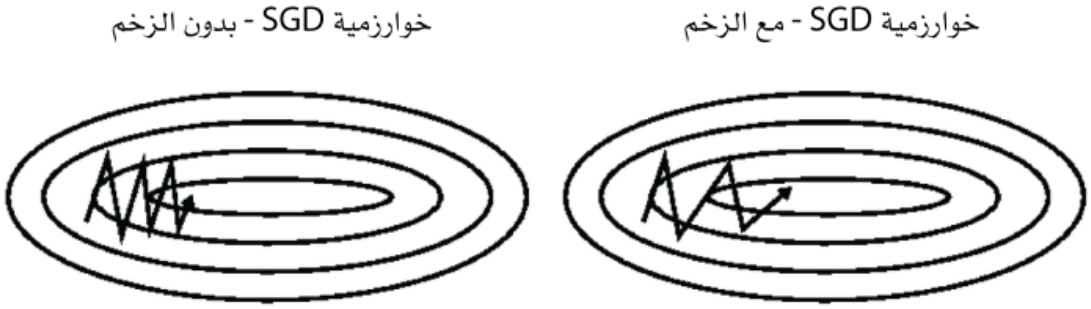
٢ .معدل التعلم(Learning Rate)



- اختيار قيمة معدل التعلم من الأمور المهمة عند تدريب أي شبكة عصبية، لاختيار قيمته بشكل صحيح قم بتغيير القيمة في كل مرة تقوم فيها بتعديل قيم مُدخلات الضبط (hyper-parameters) الأخرى لشبكتك.
- للوصول لأفضل قيمة لمعدل للتعلم ابدأ بقيمة منخفضة جداً (10^6-)، واضربها بقيمة ثابتة بشكل تصاعدي حتى تصل قيمة معدل التعلم إلى قيمة عالية (على سبيل المثال 10). قم بقياس أداء النموذج الخاص بك مقابل كل قيمة من قيم معدل التعلم التي جربتها لتحديد معدل التعلم المثالي لمشكلتك، يمكنك بعد ذلك إعادة تدريب نموذجك باستخدام معدل التعلم الأفضل.

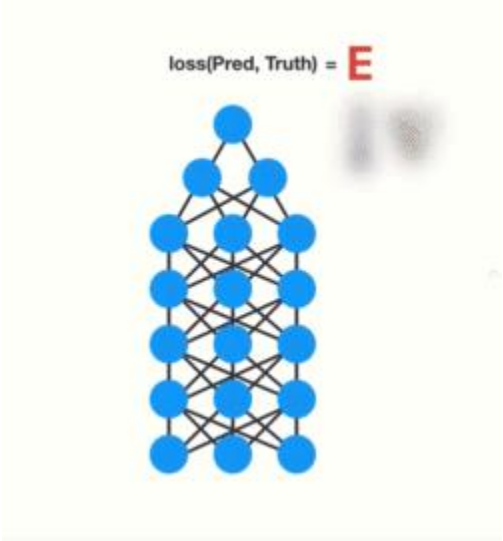
- أفضل قيمة لمعدل التعلم عادة ما تكون نصف قيمة معدل التعلم الذي يبدأ بعدها النموذج في التباعد. حاول تجربة قيم مختلفة لمعدل التعلم ولاحظ كيف تؤثر على أداء النموذج لتطوير حدسك حول معدلات التعلم.
- يُوصَى أيضًا باستخدام مكتشف معدل التعلم الذي قدمته ليزلي سميث. هذه الطريقة ممتازة للعثور على معدل تعلم جيد لمعظم خوارزميات التحسين SGD) وأشكالها المختلفة (وتعمل مع معظم معماريات الشبكة.
- أنظر أيضاً القسم الخاص بجدولة معدل التعلم أدناه.

٣ .الزخم(Momentum)



- تتخذ خوارزمية النزول الاشتقاقي (Gradient Descent) خطوات ثابتة للوصول للحد الأدنى، عندما تكون المشتقة صغيرة قد تستغرق الخطوات الصغيرة الكثير من الوقت للوصول للتقارب(Convergence) ، في المقابل يتيح الزخم لخوارزمية النزول الاشتقاقي الأخذ بعين الاعتبار الخطوات السابقة للخطوة الحالية، هذا يتيح للخوارزمية تجاوز النقاط الصغرى المحلية من خلال أخذ خطوات كبيرة والدفع بشكل أسرع عند الوصول إلى هذه النقاط.
- بشكل عام، من الأفضل أن تكون قيمة الزخم قريبة جداً من الرقم 1، وتعتبر 0.9 قيمة جيدة للبدء لمجموعات البيانات الصغيرة. حاول الاقتراب تدريجياً من القيمة واحد (0.999) كلما زاد حجم مجموعة البيانات التي لديك.

٤ .تلاشي وانفجار مشتقة الخطأ



- طبقات الشبكة العصبية لا تتعلم جميعها بنفس الدرجة، عندما تنشر خوارزمية الانتشار العكسي (Backpropagation) مشتقة الخطأ من طبقة المخرجات إلى الطبقات الأولى في الشبكة، تصبح المشتقات أصغر وأصغر حتى تصبح لا تكاد تذكر عندما تصل للطبقات الأولى. هذا يعني أن أوزان الطبقات الأولى لا يتم تحديثها بشكل ملحوظ مع كل خطوة تدريبية مقارنة بالطبقات الأقرب لنهاية الشبكة.
- هذه المشكلة تسمى مشكلة تلاشي المشتقة(Vanishing Gradient) ،)هناك مشكلة مشابهة لها تُعرف بانفجار المشتقة، تحدث هذه المشكلة عندما تصبح مشتقة الخطأ لبعض الطبقات كبيرة جداً وبشكل تدريجي، مما يؤدي إلى تحديث الأوزان فيها إلى قيم ضخمة مقارنة بالطبقات الأخرى.
- هناك عدة طرق لمواجهة مشكلة تلاشي مشتقة الخطأ، وهي:

١.٤ دوال التنشيطActivation Functions

- بشكل عام يتحسن الأداء عند استخدام دوال التنشيط في الطبقات المخفية/الوسطى وفق الترتيب التالي (أداء أفضل → أداء أدنى):

logistic→ tanh → ReLU → Leaky ReLU → ELU → SELU

- دالة ReLU هي دالة التنشيط الأكثر شيوعاً، إذا كنت لا ترغب بتجربة العديد من دوال التنشيط فإن دالة ReLU ستكون خياراً جيداً لك، ولكن ضع في اعتبارك أن ReLU أصبحت أقل فعالية من دوال مثل ELU أوGELU
- إذا كنت ترغب في تجربة دوال أخرى، فيمكنك تجربة الدوال التالية:

•لمقاومة مشكلة فرط التخصيص في الشبكة العصبية استخدمRReLU

•لتقليل التأخير في وقت التشغيل استخدمLeaky ReLU
•لمجموعات التدريب الضخمة استخدمPReLU
•للحصول على استدلال سريع استخدمLeaky ReLU
•إذا كانت شبكتك لا تقوم بتسوية البيانات استخدمELU
•للحصول على دالة تنشيط قوية وشاملة استخدمSELU

- كما هو الحال دائماً لا تخف من تجربة دوال تنشيط مختلفة، وراقب قيم الأوزان والانحياز (Bias) لمساعدتك في اختيار الدالة التي تناسبك بشكل أفضل.
- هذه الورقة العلمية تقارن بعمق بين دوال التنشيط المختلفة للشبكات العصبية.
- دوال التنشيط في طبقة المخرج/النهائية
  - الانحدار :(Regression) لا تتطلب مشاكل الانحدار دالة تنشيط للخلايا العصبية في طبقة المخرج إذا كان يمكن للناتج أن يأخذ قيمة غير محددة بنطاق معين. في الحالات التي نريد فيها أن تكون القيم ضمن نطاق معين، يمكننا استخدام دالة tanh لتُخرج لنا قيم بين -1 و 1 أو الدالة اللوجستية لتُخرج لنا قيم بين 0 و 1، في الحالات التي نبحث فيها عن نتائج إيجابية فقط، يمكننا استخدام دالة التنشيط softplus.
  - التصنيف :(Classification) تستخدم دالة التنشيط Sigmoid للتصنيف الثنائي للتأكد من أن الناتج يقع بين القيمتين 0 و1. وتستخدم دالة softmax للتصنيف متعدد الفئات لضمان أن يكون مجموع كل احتمالات الإخراج يساوي 1.

٣.٤ طريقة تهيئة الأوزان(initialization)

- من الممكن أن تؤدي التهيئة الصحيحة للقيم الأولية للأوزان إلى تسريع وقت التقارب إلى حد كبير. اختيار طريقة التهيئة تعتمد على دالة التنشيط التي تم اختيارها. هذه بعض الأمور التي يمكنك تجربتها:
- عند استخدام دالة ReLU أو Leaky RELU، استخدم تهيئةHe
- عند استخدام دالة SELU أو ELU ، استخدم تهيئةLeCun
- عند استخدام دالة softmax أو Sigmoid أو tanh ، استخدم تهيئةGlorot
- أغلب طرق التهيئة هي أشكال مختلفة من دالة التوزيع الطبيعي.(normal distribution)

٣.٤ التسوية الحزميةBatchNorm

- التسوية الحزمية تتعلم القيمة المتوسطة المثلى لكل حزمة تدخل للطبقة وتقوم بتسوية المدخلات بناء عليها. تتم التسوية من خلال طريقة التوسيط الصفري (جعل الصفر هو المتوسط الحسابي للقيم). وتسوية المدخلات بحيث تكون ضمن نفس النطاق. تعمل التسوية الحزمية أيضاً كضابط يمنع فرط التخصيص، مما يعني أنه قد لا نحتاج لاستخدام تقنية إسقاط الخلايا أو ضبط L2 لمنع فرط التخصيص.
- يتيح استخدام التسوية الحزمية استخدام معدلات تعلم أكبر (والتي بدورها تؤدي إلى تقارب أسرع) كما أن لها دور في تحسين أداء الشبكات العصبية عن طريق الحد من مشكلة تلاشي مشتقة الخطأ. الجانب السلبي الوحيد للتسوية الحزمية هو أنها تزيد من وقت تدريب الشبكة بسبب الحسابات الإضافية المطلوبة في كل طبقة.

٤.٤ اقتصاص مشتقة الخطأ(Gradient Clipping)

- من الطرق الفعالة لتقليل انفجار مشتقة الخطأ خاصة عند تدريب الشبكات العصبية التكرارية (RNNs) هي ببساطة اقتصاص قيمتها عندما تتجاوز حد معين.
- جرّب تغيير حد قيم الاقتصاص للعثور على أفضل قيمة مناسبة.

٥.٤ التوقف المبكر

•يسمح التوقف المبكر بتدريب نموذج يحتوي على العديد من الطبقات المخفية والكثير من الخلايا العصبية المخفية لدورات تدريبية أكثر مما تحتاجه، مع إيقاف التدريب عندما يتوقف الأداء عن التحسن بعد دورة تدريبية معينة، كما أن باستطاعته حفظ النموذج الأفضل أداء بالنسبة لك.

٥ .الإسقاط(Dropout)

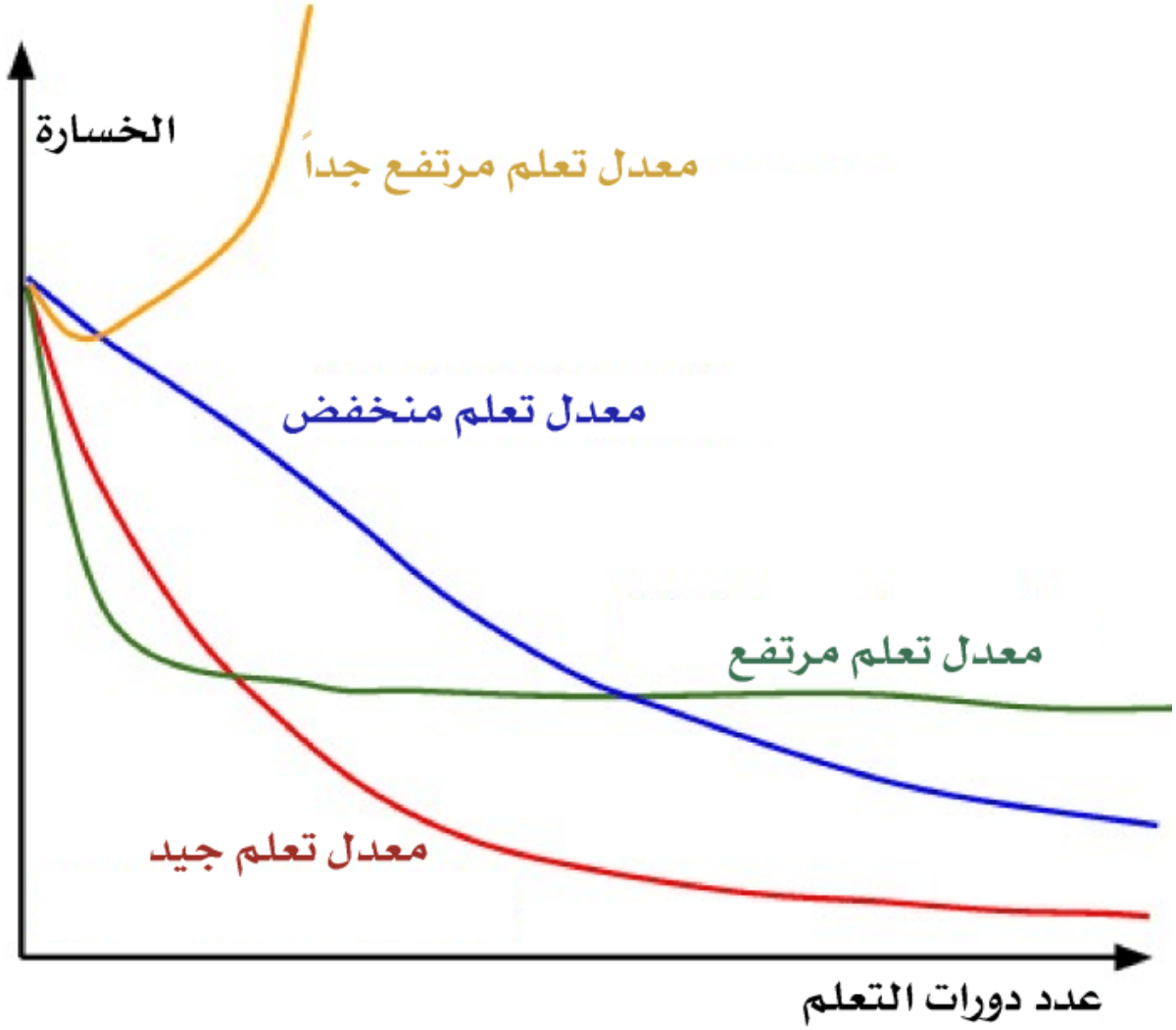
- يعتبر الإسقاط طريقة تنظيم رائعة تمنحك تحسيناً كبيراً في الأداء مقارنة بمدى سهولة تطبيق هذه التقنية (يحسّن الأداء بنسبة ~ 2٪ في النماذج الحديثة). تقوم جميع طرق الإسقاط على نفس المبدأ إذ أنها تقوم بإيقاف نسبة من الخلايا العصبية بشكل عشوائي في كل طبقة مع كل خطوة تدريب. هذا يجعل الشبكة أكثر قوة لأنها لا تستطيع الاعتماد على مجموعة معينة من الخلايا العصبية لإخراج النتائج. بالتالي يتم توزيع تعلم المعرفة للشبكة بأكملها. عند استخدام تقنية الإسقاط يتم توليد ما يقارب 2^n شبكة عصبية مختلفة حيث n يمثل عدد الخلايا العصبية الموجودة أثناء عملية التدريب والتي يتم تجميعها لاحقاً لإخراج النتائج.
- معدل الإسقاط الجيد يتراوح ما بين 0.1 إلى 0.5، استخدم قيمة 0.3 لشبكات العصبية التكرارية (RNN) و 0.5 لشبكاتCNN ، قم بزيادة القيم كلما زاد عدد الطبقات في الشبكة. زيادة قيمة معدل الإسقاط يقلل من فرط التخصيص، وإنقاص قيمته يساعد في التغلب على نقص التخصيص.
- من الممكن تجربة معدلات مختلفة لقيم الإسقاط في الطبقات الأولى من شبكتك والتحقق لاختيار أفضل أداء، بالتأكيد لن تحتاج لاستخدام تقنية الإسقاط في طبقات الإخراج.
- قم بقراءة هذه الورقة إذا أردت استخدام تقنية الإسقاط مع التسوية الحزمية(BatchNorm)

٦ .خوارزميات التحسين(Optimizers)

- خوارزمية النزول الاشتقاقي ليست الخوارزمية الوحيدة الموجودة في الساحة إذ أن هناك خوارزميات أخرى يمكن اختيارها. يقدم هذا المقال وصفاً جيداً لبعض خوارزميات التحسين التي يمكنك الاختيار منها.
- النصيحة العامة استخدم خوارزمية النزول الاشتقاقي العشوائي (Stochastic Gradient Descent) إذا كانت الأولوية والاهتمام الأكبر هو لجودة الوصول لنقطة التقارب وكان الوقت الذي يستغرقه التقارب ليس بالأهمية القصوى.

- إذا كانت الأولوية لوقت التقارب وليس لجودة نقطة التقارب أي أن الوصول إلى أي نقطة قريبة من نقطة التقارب المثلى يعتبر كافياً بالنسبة لك، جرب خوارزميات مثل Adam و Nadam و RMSProp و Adamax.
- Adamو Nadam عادة ما تكون نقاط بداية جيدة وتميل إلى التجاوز عن الاختيارات الخاطئة لقيمة معدل التعلم وغيرها من مُدخلات الضبط (hyperparameters)غير المثالية.
- وفقاً لأندريه كارباثي أداء SGD الذي يتم ضبطه بشكل جيد أفضل من أداء Adam في حالة الشبكات العصبية الالتفافية.ConvNets

V .جدولة معدل التعلم(Learning Rate Scheduling)



- تحدثنا عن أهمية اختيار معدل التعلم الجيد، لا نريد أن تكون قيمته مرتفعة جداً، حتى لا تتأرجح دالة التكلفة حول القيمة المثلى وتتباعد. ولا نريد أن تكون القيمة منخفضة جداً إذ أن ذلك يعني أن التقارب سيستغرق وقتاً طويلاً جداً.
- اختيار قيمة معدل التعلم مهمة صعبة لأن كل من معدلات التعلم المرتفعة والمنخفضة لها مزاياها. الجيد في الأمر هو أنه ليس علينا الالتزام بمعدل تعلم واحد، فمن خلال جدولة معدل التعلم يمكننا البدء بمعدلات تعلم ذات قيمة عالية تسمح لنا بالتحرك بشكل أسرع خلال منحدرات المشتقة، وإبطائها عندما نصل إلى الوادي الذي يحتوي نقطة التقارب والذي يتطلب اتخاذ خطوات أصغر.
- هناك العديد من الطرق لجدولة معدلات التعلم بما في ذلك خفض معدل التعلم بشكل أسي(exponentially) ، أو باستخدام دالة الخطوة (Step function)، أو تعديلها عندما يبدأ الأداء في الانخفاض أو استخدام جدولة الدورة الواحدة.one-cycle
- استخدم معدل تعلم ثابت حتى تقوم بتحديد جميع مدخلات الضبط الأخرى(Hyper-parameters) ، ثم قم بتنفيذ جدولة معدل التعلم في النهاية.
- يُوصى بإجراء تجارب مختلفة مع استراتيجيات مختلفة مع جدولة معدل التعلم لاختيار الاستراتيجية التي تؤدي إلى أفضل نموذج.

عمليًا في بايثون[6] :

1.إعداد المشروع

ستحتاج أولًا لتثبيت بعض التبعيات، وذلك لإنشاء مساحة عملٍ للاحتفاظ بملفاتنا قبل أن نتمكن من تطوير برنامج التعرف على الصور، وسنستخدم بيئة بايثون 3.8 الافتراضية لإدارة التبعيات الخاصة بمشروعنا.

سَنُنشئ مجلدًا جديدًا خاصًا بمشروعنا وسندخل إليه هكذا:

```
mkdir tensorflow-demo
```

```
cd tensorflow-demo
```

سننفذّ الأوامر التالية لإنشاء البيئة الافتراضية:

```
python -m venv tensorflow-demo
```

ومن ثم الامر التالي في Linuxلتنشيط البيئة الافتراضية:

```
source tensorflow-demo/bin/activate
```

أما فيWindows ، فيكون أمر التنشيط:

```
"tensorflow-demo/Scripts/activate.bat"
```

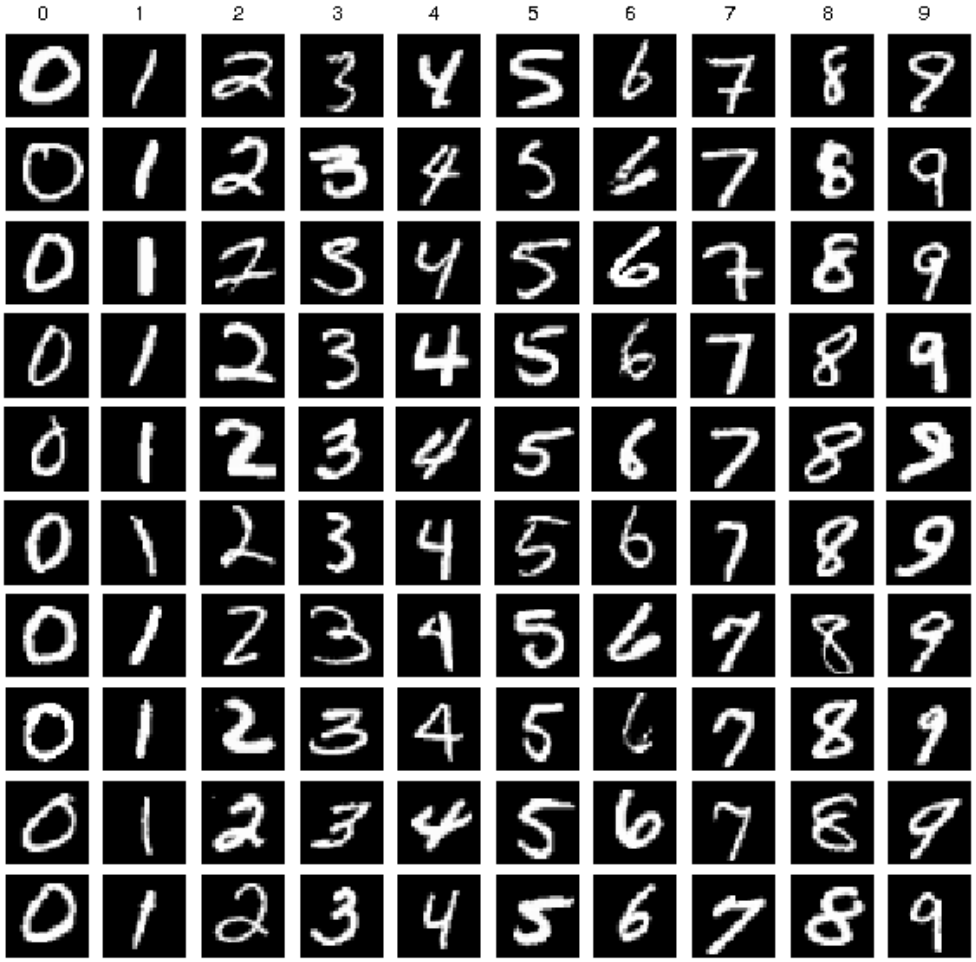
بعد ذلك، سنُثبتّ المكتبات التي سنستخدمها.
سنستخدم إصداراتٍ محددةٍ من هذه المكتبات، من خلال إنشاء ملف requirements.txtفي مجلد المشروع، وسيُحدِّد هذا الملف المتطلبات والإصدارات التي سنحتاج إليها.
نفتح الملف requirements.txtفي محرر النصوص، ونُضيف الأسطر التالية، وذلك لتحديد المكتبات التي نريدها وإصداراتها:

```
keras==2.6.0
numpy==1.19.5
Pillow==8.4.0
scikit-learn==1.0
scipy==1.7.1
sklearn==0.0
tensorflow==2.6.0
```

سنحفظ التغييرات التي طرأت على الملف وسنخرج من محرر النصوص، ثم سنُثَبت هذه المكتبات بالأمر التالي:

```
(tensorflow-demo) $ pip install -r requirements.txt
```

بعد تثبيتنا لهذه التبعيات، سنُصبح جاهزين لبدء العمل على مشروعنا.
.2استيراد مجموعة بياناتMNIST
تُسمى مجموعة البيانات التي سنستخدمها، بمجموعة بيانات MNIST، وهي مجموعةٌ كلاسيكيةٌ في مجتمع مُطوري تعلم الآلة، وتتكون من صورٍ لأرقامٍ مكتوبةٍ بخط اليد، بحجم 28×28 بكسل. ونستعرض فيما يلي بعض الأمثلة للأرقام المُتضمنة فيها:



لاحظ أنه ينبغي أن نستخدم ملفًا واحدًا لجميع أعمالنا في هذا المقال، ولنُنشئ برنامج بايثون يتعامل مع مجموعة البيانات هذه، سننشئ ملفًا جديدًا باسم main.py، وسنفتح هذا الملف بأي محرر شيفرات لدينا -مثل VS code- وسنُضيف هذه الأسطر البرمجية لاستيراد المكتبات اللازمة:

```
import tensorflow as tf
import numpy as np
from sklearn.preprocessing import OneHotEncoder
#مكتبة معالجة الصور
from PIL import Image
#التوافقية مع إصدار سابق
tf.compat.v1.disable_v2_behavior()
```

وسنضيف أيضًا هذه الأسطر من الشيفرات البرمجية لملفك لاستيراد مجموعة بيانات MNIST وذلك باختيار صور التدريب المتاحة من Tensorflow ومن ثم نُنزلها ونقسمها إلى جزئين: الأول للتدريب والثاني للاختبار:

```
#اختيار بيانات التدريب

mnist = tf.keras.datasets.mnist
#تنزيل بيانات التدريب والاختبار
(x_train, y_train), (x_test, y_test) = mnist.load_data()
#طباعة عدد بيانات التدريب والاختبار
print (len(x_train))
print (len(x_test))
#تحجيم البيانات بين 0 و1
x_train, x_test = x_train / 255.0, x_test / 255.0
#الترميز الأحادي النشط
y_train = [[i] for i in y_train]
y_test = [[i] for i in y_test]
enc = OneHotEncoder(sparse=True)
enc.fit(y_train)
y_train =  enc.transform(y_train)
y_test = enc.transform(y_test)
```

تقوم الدالة 'mnist.load_data' بتنزيل البيانات وتقسيمها إلى مجموعتين واحدة للتدريب (60000 صورة) والمجموعة الثانية للاختبار (10000 صورة).

وعند قراءة البيانات سنستخدم الترميز الأحادي النشط One-Hot Encodingلتمثيل التصنيفات للصور. حيث يَستَخدم الترميز الأحادي النشط One-Hot Encoding متجهًا vector مُكونٌ من قيمٍ ثنائيةٍ لتمثيل القيم الرقمية أو الصنفية. ونظرًا لأن أصنافنا مخصصةٌ لتمثيل الأرقام من 0 إلى 9، فإن المتجه سيحتوي على 10 قيمٍ، واحدةٌ لكلّ رقمٍ ممكنٍ. وتُسنَد إحدى هذه القيم بوضع القيمة 1، وذلك لتمثيل الرقم في هذا المؤشر للمتجه، كما ستُسنَد القيم الباقية بالقيمة 0. فمثلًا، سيُمثلُ الرقم 3 من خلال المتجه هكذا[0] : 0، 0، 1، 0، 0، 0، 0، 0، 0، 0.]وسنلاحظ وجود القيمة 1 في الفهرس 3، لذلك فإن المتجه سيُمثِلُ الرقم 3.

ولتمثيل الصور الفعلية والتي تكون بحجم 28 28xبكسل، يتوجب علينا تسويتها في المتجه 1 Dبحجم 784 بكسل، وهو ناتج ضرب 28×28. وسنخزن هذه البكسلات والتي ستُشكل الصورة لاحقًا، وذلك في قيمٍ تتراوح بين 0 و255، حيث ستحدّد هذه القيم تدرج اللون الرمادي للبكسل، وستُعرَض صورنا باللونين الأبيض والأسود فقط. لذلك سيُمثلُ البكسل الأسود بالقيمة 255، والبكسل الأبيض بالقيمة 0، وذلك مع التدرجات المختلفة للون الرمادي بينهم.

والآن بعد استيرادنا للبيانات، حان الوقت للتفكير في كيفية بناء الشبكة العصبية.

.3تحديد بنية الشبكة العصبية

يُشير مصطلح بنية الشبكة العصبية لعناصرٍ متنوعةٍ، مثل عدد الطبقات في الشبكة وعدد الوحدات في كلّ طبقةٍ، كما يشير إلى كيفية توصيل هذه الوحدات بين الطبقات المختلفة. ونظرًا لأن الشبكات العصبية مستوحاةٌ من كيفية عمل الدماغ البشري، فسنستخدم مصطلح الوحدة ليُمثّل ما يُمكن تسميته بيولوجيًا بالخلايا العصبية.

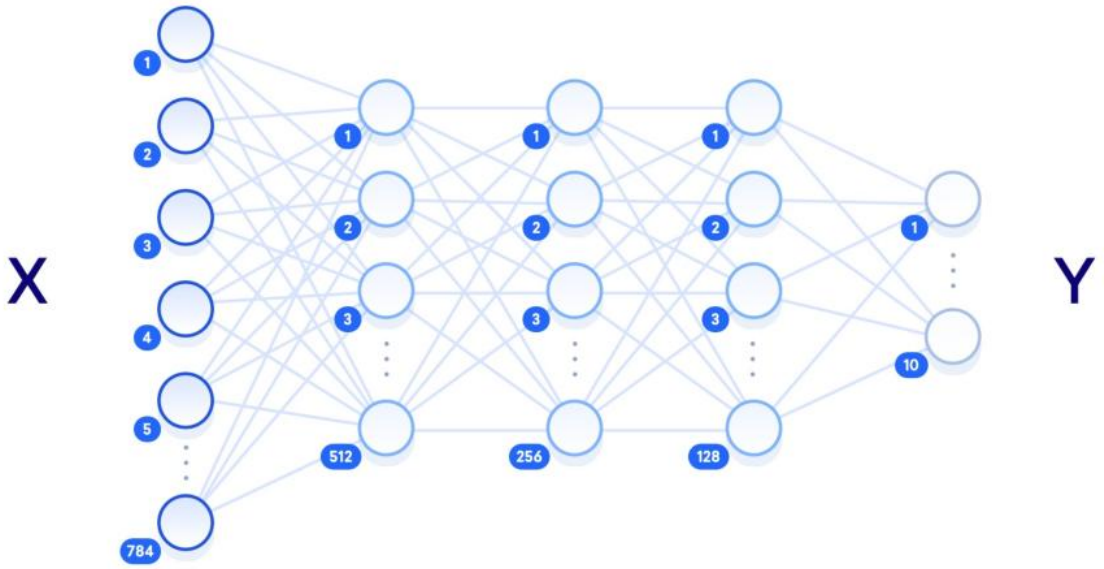
تأخذ الوحدات بعض القيم من الوحدات السابقة مثل مُدخلاتٍ لها، حيث تتشابه مع الخلايا العصبية التي تُمرر إشاراتها حول الدماغ، ثمّ تُجري عمليةً حسابيةً، وتُمرر القيمة الجديدة مثل مُخرجاتٍ إلى وحداتٍ أخرى، وهكذا. تُوضع هذه الوحدات على شكل طبقاتٍ متراكبةٍ فوق بعضها البعض مشكّلةً الشبكة العصبية، بحيث يمكن للشبكة أن تتألف كحدٍ أدنى من طبقتين، طبقةٌ لإدخال القيم، وطبقةٌ أخرى لإخراج القيم. يُستخدم مصطلح الطبقة المخفية لجميع الطبقات الموجودة بين طبقات المُدخلات وطبقات المُخرجات الخارجية، أي أن تلك الطبقات تكون مخفيةً عن العالم الحقيقي.

تحقق البُنى المختلفة للشبكة نتائجًا مختلفةً عن بعضها البعض، ويمكن اتخاذ الأداء مثل معيارٍ للحكم على هذه البُنى المختلفة، كما يمكن اتخاذ عناصر أخرى معيارًا للحكم، مثل الوسطاء والبيانات ومدة التدريب.

سنضيف هذه الأسطر البرمجية التالية لملفك، وذلك لتخزين عدد الوحدات المُخصصة لكلّ طبقةٍ ووضعها في متغيّراتٍ عامةٍ. وهذه الطريقة ستسمح لنا بتغيير بِنية الشبكة بمكانٍ واحدٍ، وفي نهاية هذا المقال يمكنك اختبار مدى تأثير الأعداد المختلفة من الطبقات والوحدات على نتائج نموذجنا:

```
n_input = 784 # input layer (28x28 pixels)
n_hidden1 = 512 # 1st hidden layer
n_hidden2 = 256 # 2nd hidden layer
n_hidden3 = 128 # 3rd hidden layer
n_output = 10 # output layer (0-9 digits)
```

يُوضح الرسم البياني التالي تصورًا للبِنية التي صمّمناها، مع توصيل كلّ طبقةٍ بالطبقات المحيطة بها توصيلًا كاملًا:



ويرتبط مصطلح الشبكة العصبية العميقة Deep Neural Network بعدد الطبقات المخفية، وعادةً ما تُشير كلمة السطحية في مصطلح الشبكة العصبية السطحية إلى وجود طبقةٍ مخفيةٍ واحدةٍ، بينما تُشير كلمة العميقة إلى وجود طبقاتٍ مخفيةٍ متعددةٍ.
ونظريًا إذا أُعطِيت الشبكة العصبية السطحية ما يكفي من بياناتٍ للتدريب، فيجب أن تَقدِر على تمثيل أي وظيفةٍ يمكن للشبكة العصبية العميقة أن تؤدّيها. ولكن من ناحية الفعالية الحسابية، فغالبًا ما يكون نتائج استخدام شبكةٍ عصبيةٍ عميقةٍ ذات حجمٍ صغيرٍ أفضل من النتائج التي تُعطيها الشبكة العصبية السطحية ذات العدد الكبير من الوحدات المخفية، وذلك عند تأديتهم لنفس المَهمة.
كما أن الشبكات العصبية السطحية غالبًا ما تواجه مشكلة فَرط التخصيصOverfitting ، إذ يكون هدف الشبكة الأساسي هو حفظ بيانات التدريب التي شاهدتها، ولكنها لن تستطيع تعميم المعرفة التي اكتسبتها على البيانات الجديدة، وهذا هو السبب في كون استخدام الشبكات العصبية العميقة أكثر شيوعًا، إذ أنها تسمح للطبقات المتعددة الموجودة بين البيانات المُدخلة الأولية والبيانات المُصنفة الناتجة، بتعلم الميزات على مستوياتٍ متنوعةٍ، مما يُعزز قدرة الشبكة على التعلم وتعميم الفكرة.
ومن العناصر الأخرى للشبكة العصبية التي يجب تعريفها هنا هي الوسطاء الفائقةHyperparameters ، فعلى عكس الوسطاء العادية التي تُحدث قيمها أثناء عملية التدريب، سنُسند قيم الوسطاء الفائقة في البداية وسنثبتها طوال العملية.
أسنِد المتغيّرات بالقيم التالية في ملفك:

```
learning_rate = 1e-4
n_iterations = 1000
batch_size = 128
dropout = 0.5
```

يمثل معدل التعلم Learning Rate مدى تعديل الوسطاء في كلّ خطوةٍ من عملية التعلّم، إذ تُعَد هذه التعديلات مكونًا رئيسيًا للتدريب، فبعد كلّ عملية مرورٍعبر الشبكة، سنضبط أوزان الطبقات قليلًا لأهمية ذلك في محاولةٍ لتقليل الخسارة، حيث يمكن لمعدل التعلم المرتفع أن يتحقق بسرعة، ولكن يمكن كذلك أن تتجاوز القيم المثلى عند تحديثها في كلّ مرة.
يشير مصطلح عدد التكرارات Number Of Iterations إلى عدد مرات مرورنا على خطوة التدريب، ويشير حجم الدفعة Batch Size لعدد أمثلة التدريب التي نستخدمها في كل خطوة، كما ويمثل المتغير dropoutالموضع الذي نحذف عنده بعضًا من الوحدات عشوائيًا. وسنستخدم المتغير dropoutفي الطبقة النهائية المخفية لإعطاء كلّ وحدة من الوحدات احتمالًا بنسبة 50٪ للتخلص منها في كلّ خطوة تدريبٍ، وهذا سيساعد على منع ظهور مشكلة فرط التخصيص.Overfitting
حددنا الآن بِنية شبكتنا العصبية والوسطاء الفائقة التي ستُؤثر على عملية التعلّم، والخطوة التالية هي بناء الشبكة مثل مخططٍ بيانيٍ من خلال مكتبة.TensorFlow
.4بناء مخطط بياني من خلال مكتبةTensorFlow
لبناء شبكتنا، لابد لنا من إعداد الشبكة مثل مخططٍ بيانيٍ حسابي من خلال مكتبة TensorFlow لتنفيذه. والمفهوم الأساسي لمكتبة TensorFlowهو tensor، وهو بنية بياناتٍ مشابهةٍ لبِنية المصفوفةArray ، أو القائمة .List وهذا المتغير سيهيئ ويُعالَج عند مروره عبر المخطط البياني للشبكة عبر عملية التعلّم.
وسنبدأ بتحديد ثلاثة متغيراتٍ tensorsمن نوع placeholders، وهو نوع tensorتُسندُ قيمته لاحقًا. والآن سنضيف الشيفرة البرمجية التالية إلى الملف الذي نعمل عليه:

```
X = tf.compat.v1.placeholder("float", [None, n_input])
Y = tf.compat.v1.placeholder("float", [None, n_output])
keep_prob = tf.compat.v1.placeholder(tf.float32)
```

إنّ الوسيط الوحيد الذي يتوجب علينا تحديده عند التعريف هو حجم البيانات التي سنُسندها لاحقًا، وبالنسبة للمتغير Xسنستخدم شكل None]، 784[، إذ ستمثل القيمة Noneكميةً غير محددةٍ، وسنُسند كميةً غير محددةٍ من الصور ذات حجم 784 بكسل. بحيث يصبح شكل المتغير Yهو None]، 10[، وستمثل Noneعددًا غير محددٍ من التصنيفات الناتجة، مع وجود 10 أصنافٍ محتملةٍ.

وسنستخدم في المتغير keep_prob tensor من نوع placeholders للتحكم في معدل dropout، وسنجعله من نوع placeholders وذلك لجعله متغيرًا من نوعٍ قابلٍ للتعديل، بدلًا من كونه متغيرًا من نوعٍ غير قابلٍ للتعديل immutable variable ، وذلك لأننا نريد استخدام نفس tensor التدريب عند إسناد dropout بالقيمة 0.5، ونفس tensor الاختبار عند إسناد dropout بالقيمة 1.0.

والوسطاء التي ستُحدث قيمها الشبكة العصبية في عملية التدريب هي القيم الخاصة بوزن كلّ طبقةٍ، والتي تُعبر عن الأهمية وقيم التحيز bias values ، لذلك سنحتاج لإسنادهم بقيمٍ إبتدائيةٍ بدلًا من قيمٍ فارغةٍ. وهذه القيم هي الأساس الذي ستبدأ الشبكة رحلة التعلم انطلاقًا منها، إذ ستُستخدم في تفعيل دوال الشبكة العصبية، والتي تُمثِلُ قوة الاتصالات بين الوحدات.

ونظرًا لاستمرار تحسين القيم أثناء عملية التدريب، يمكننا ضبطها حاليًا بالقيمة 0. لاحظ أن القيمة الأولية في الواقع لها تأثيرٌ كبيرٌ على الدقة النهائية للنموذج. وسنستخدم التوزيع الاحتمالي الطبيعي المنقطع Truncated normal distribution لتوليد قيمٍ عشوائيةٍ لأوزان الطبقات، بحيث يكونون قريبين من الصفر حتى يتمكنوا من التعديل إما باتجاهٍ إيجابيٍ أو سلبيٍ، كما يكونون مختلفين قليلًا، وذلك ليُنتِجوا أخطاءً مختلفةً، وبهذه الطريقة سنضمن بأن يتعلم النموذج شيئًا مفيدًا.

والآن سنضيف هذه الأسطر البرمجية التالية لملفنا الذي نعمل عليه:

```
weights = {
    'w1': tf.Variable(tf.random.truncated_normal([n_input, n_hidden1],
                                    stddev=0.1)),
    'w2': tf.Variable(tf.random.truncated_normal([n_hidden1, n_hidden2],
                                    stddev=0.1)),
    'w3': tf.Variable(tf.random.truncated_normal([n_hidden2, n_hidden3],
                                    stddev=0.1)),
    'out': tf.Variable(tf.random.truncated_normal([n_hidden3, n_output],
                                     stddev=0.1)),
}
```

بالنسبة للتحيز Bais ، سنستخدم قيمةً ثابتةً صغيرةً لضمان تنشيط جميع tensors المراحل الأولية، وبالتالي المساهمة في الانتشار. وستُخزن الأوزان وجميع tensors التحيزات في objects قواميس Dictionary لسهولة الوصول إليها.

أضف هذه الشيفرة البرمجية للملف الذي نعمل عليه وذلك لتعريف التحيز وقِيَمه:

```
biases = {
    'b1': tf.Variable(tf.constant(0.1, shape=[n_hidden1])),
    'b2': tf.Variable(tf.constant(0.1, shape=[n_hidden2])),
    'b3': tf.Variable(tf.constant(0.1, shape=[n_hidden3])),
    'out': tf.Variable(tf.constant(0.1, shape=[n_output]))
}
```

والآن جهّز طبقات الشبكة العصبية من خلال تعريف العمليات التي ستتعامل مع tensors المرحلة الحالية. وأضف هذه الشيفرة البرمجية للملف الذي نعمل عليه:

```
layer_1 = tf.add(tf.matmul(X, weights['w1']), biases['b1'])
layer_2 = tf.add(tf.matmul(layer_1, weights['w2']), biases['b2'])
layer_3 = tf.add(tf.matmul(layer_2, weights['w3']), biases['b3'])
layer_drop = tf.nn.dropout(layer_3, keep_prob)
output_layer = tf.matmul(layer_3, weights['out']) + biases['out']
```

ستنفذ كلّ طبقةٍ مخفيةٍ عملية ضربٍ للمصفوفة على نتائج الطبقة التي سبقتها وعلى أوزان الطبقة الحالية، وسيُضاف التحيز لهذه القيم. في الطبقة المخفية الأخيرة، سنُطبّق عملية التسرب dropout بالقيمة 0.5 للمتغير Keep_prob الخاص بنا.

الخطوة الأخيرة في بناء المخطط البياني، هي تحديد دالة الخسارة التي نريد تحسينها. والاختيار الشائع لدالة الخسارة في المكتبة البرمجية TensorFlow هو الانتروبي المشترك Joint Antropy ، والمعروف كذلك باسم فقدان السجل log-loss ، وهو الذي يُحدد الفرق بين التوزيعين الاحتماليين لكلّ من التنبؤات والتصنيف. ويمكن أن تكون قيمة الانتروبي المشترك 0، وذلك في أفضل الأحوال عند التصنيف المثالي، وذلك مع انعدام الخسارة تمامًا.

سنحتاج كذلك إلى اختيار خوارزمية التحسين المناسبة، والتي سنستخدمها لتقليل الناتج من دالة الخسارة. وتُسمى هذه العملية بعملية تحسين الانحدار التدريجي، وهي طريقةٌ شائعةٌ للعثور على الحد الأدنى للدالة، من خلال اتخاذ خطواتٍ تكراريةٍ على طول التدرج في الاتجاه السلبي التنازلي. وهناك العديد من الخيارات لخوارزميات تحسين الانحدار التدريجي المُطبقة في المكتبة البرمجية TensorFlow، إلا أننا سنستخدم في هذا المقال خوارزمية المُحسِّن أدم Adam optimizer، الذي يعمتد على عملية تحسين الانحدار التدريجي باستخدام الزخم أو كمية الحركة Momentum ، وذلك بتسريع عملية التنعيم من خلال حساب متوسطٍ مُرجَّحٍ بكثرة للتدرجات، واستخدام ذلك في التعديلات مما يؤدي لتقاربٍ أسرع. وسنضيف هذه الشيفرة للملف الذي نعمل عليه:

```
cross_entropy = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits(
        labels=Y, logits=output_layer
    ))
train_step = tf.compat.v1.train.AdamOptimizer(1e-4).minimize(cross_entropy)
```

عرّفنا حتى الآن الشبكة وبنيناها باستخدام المكتبة البرمجية TensorFlow ، والخطوة التالية هي إرسال البيانات عبر المخطط البياني لتدريبها، ومن ثَمّ اختبارها للتحقق فيما إن كانت تعلمت شيئًا بالفعل أم لا.

.5التدريب والاختبار

تتضمن عملية التدريب تغذية المخطط البياني للشبكة بمجموعة بيانات التدريب، وتحسين نتيجة دالة الخسارة، إذ أن في كلّ مرةٍ تمر فيها الشبكة عبر مجموعةٍ إضافيةٍ من صور التدريب، فستُحدثُ الوسطاء لتقليل الخسارة، وذلك بهدف تحسين دقة التنبؤ للأرقام؛ أما عملية الاختبار، فتتضمن تشغيل مجموعة بيانات الاختبار الخاصة بنا عبر المخطط البياني المدرّب، كما ستتتبع عدد الصور التي صح التنبؤ بها، حتى نحسب الدقة جيدًا.

قبل البدء في عملية التدريب، سوف نحدد دالة تقييم الدقة لكي نتمكن من طباعتها على مجموعاتٍ صغيرةٍ من البيانات أثناء التدريب. هذه البيانات المطبوعة ستسمح لنا بالتحقق من انخفاض الخسارة وزيادة الدقة، وذلك بدءًا من المرور الأول عبر المخطط البياني، وحتى المرور الأخير؛ كما ستسمح لنا بتتبّع ما إذا نفذنا عمليات مرورٍ كافيةً عبر المخطط البياني للوصول لنتيجةٍ مناسبةٍ ومثاليةٍ أم لا:

```
correct_pred = tf.equal(tf.argmax(output_layer, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
```

سنستخدم الدالة arg_maxفي المتغيّر right_predللموازنة بين الصور التي صنّفَ توقُّعها، وذلك بالنظر لقيمة التنبؤات output_layerوالتصنيفاتY ، وسنستخدم الدالة equalلإعادة هذه النتائج مثل قائمةٍ مؤلفةٍ من قيمٍ بوليانيةٍ. ويمكننا بعد ذلك تحويل هذه القائمة للنوع float، وذلك لحساب المتوسط للحصول على درجة الدقة الإجمالية.

الآن نحن جاهزون لتهيئة الجلسة لتشغيل المخطط البياني، إذ سنرسل للشبكة أمثلة التدريب الخاصة بنا، وبمجرد انتهاء التدريب، سنرسل أمثلة اختبارٍ جديدةٍ عبر المخطط البياني نفسه لتحديد دقة النموذج.

أضف هذه الشيفرة للملف الذي نعمل عليه:

```
init = tf.compat.v1.global_variables_initializer()
sess = tf.compat.v1.Session()
sess.run(init)
```

إن جوهر عملية التدريب في التعلم العميق هو تحسين ناتج دالة الخسارة. ونحن هنا سنهدف إلى تقليل الفرق بين التصنيفات المُتوقعة للصور والتصنيفات الحقيقية لها. وستتضمن هذه العملية أربع خطواتٍ تتكرر لعددٍ محددٍ من مرات المرور عبر المخطط البياني، وهي:

- دفع القيم إلى الأمام عبر الشبكة.
- حساب الخسارة.
- دفع القيم للخلف عبر الشبكة.
- تحديث الوسطاء.

ففي كلّ خطوة تدريب، سنعدّل الوسطاء قليلًا في محاولةٍ لتقليل نتائج دالة الخسارة. وفي الخطوة التالية مع تَقدُّم عملية التعلّم، يجب أن نشاهد انخفاضًا في الخسارة، حيث سنوقِف التدريب في النهاية، وسنستخدم الشبكة مثل نموذجٍ لاختبار بياناتنا الجديدة.

سنضيف هذه الشيفرة البرمجية للملف الذي نعمل عليه:

#التدريب على دفعات صغيرة

```
for i in range(n_iterations):
    startbatch = (i*batch_size) % len(x_train)
    endbatch = ((i+1)*batch_size) % len(x_train)
    batch_x = np.array(x_train[startbatch:endbatch])
    batch_x = batch_x.reshape(batch_size, -1)
    batch_y = y_train[startbatch:endbatch].toarray()
    if batch_x.shape != (128, 784):
        continue
    sess.run(train_step, feed_dict={
        X: batch_x, Y: (batch_y), keep_prob: dropout
    })
# طباعة الخسارة والدقة لكل دفعة صغيرة
    if i % 100 == 0:
        minibatch_loss, minibatch_accuracy = sess.run(
            [cross_entropy, accuracy],
            feed_dict={X: batch_x, Y: batch_y, keep_prob: 1.0}
        )
        print(
            "Iteration",
            str(i),
            "\t| Loss =",
            str(minibatch_loss),
            "\t| Accuracy =",
            str(minibatch_accuracy)
        )
```

بعد 100 عملية مرورٍ لكلّ خطوة تدريبٍ والتي أرسلنا فيها مجموعةً صغيرةً من الصور عبر الشبكة، سنطبع نتائج دالة الخسارة والدقة لتلك الدفعة. وينبغي ألّا نتوقع هنا انخفاض معدل الخسارة وزيادة الدقة، لأن القيم لكل دفعةٍ صغيرةٍ. إذ أن النتائج ليست للنموذج

بأكمله. فنحن نستخدم مجموعاتٍ صغيرةٍ من الصور بدلًا من إرسال كلّ صورةٍ بمفردها، وذلك لتسريع عملية التدريب والسماح للشبكة برؤية عددٍ من الأمثلة المختلفة قبل تحديث الوسطاء.
وبمجرد اكتمال التدريب، يمكننا تشغيل الجلسة على الصور المخصصة للاختبار. وهذه المرة سنستخدم القيمة 1.0 مثل مُعدل تَسرّب dropoutللمتغيّر Keep_prob، وذلك للتأكد من أن جميع الوِحدات نشطةٌ في عملية الاختبار.
أضف هذه الشيفرة البرمجية للملف الذي نعمل عليه:
إعداد صور الاختبار كمتجهات أحادية طول كل منها# 28*28

```
x_test = x_test.reshape(-1,784)
test_accuracy = sess.run(accuracy, feed_dict={X: x_test, Y: y_test.toarray(), keep_prob: 1.0})
print("\nAccuracy on test set:", test_accuracy)
```

والآن سنشغّل برنامجنا، لنعرف مدى دقة شبكتنا العصبية في التعرف على الأرقام المكتوبة بخط اليد. وسنحفظ التغييرات في الملف main.pyالذي نعمل عليه.
نَفّذ الأمر التالي في الوِحدة الطرفية لتنفيذ الشيفرة البرمجية:

```
(tensorflow-demo) $ python main.py
```

سترى نتيجةً مشابهةً لما يلي، ويمكن أن تختلف قليلًا نتائج الخسارة والدقة الفردية:

```
Iteration 0 | Loss = 3.67079 | Accuracy = 0.140625
Iteration 100 | Loss = 0.492122 | Accuracy = 0.84375
Iteration 200 | Loss = 0.421595 | Accuracy = 0.882812
Iteration 300 | Loss = 0.307726 | Accuracy = 0.921875
Iteration 400 | Loss = 0.392948 | Accuracy = 0.882812
Iteration 500 | Loss = 0.371461 | Accuracy = 0.90625
Iteration 600 | Loss = 0.378425 | Accuracy = 0.882812
Iteration 700 | Loss = 0.338605 | Accuracy = 0.914062
Iteration 800 | Loss = 0.379697 | Accuracy = 0.875
Iteration 900 | Loss = 0.444303 | Accuracy = 0.90625

Accuracy on test set: 0.9206
```

ولمحاولة تحسين دقة نموذجنا، أو لمعرفة المزيد حول تأثير ضبط الوسطاء الفائقةhyperparameters ، يمكننا تغييرها لاختبار تأثيرها المنعكس على معدّل التعلم وعتبة التسربDropout Threshold ، وكذا حجم الدفعة من الصور في كمية الأمثلة وعدد مرات المرور عبر المخطط، كما يمكننا كذلك تغيير عدد الوِحدات في طبقاتنا المخفية وتغيير عدد الطبقات المخفية نفسها، وذلك لنرى كيف ستؤثر بنية الشبكة العصبية على النموذج سواءً بزيادة دقته أو بتخفيضها.
وللتأكد من أن الشبكة تتعرف جيدًا على الصور المكتوبة بخط اليد، فسنختبرها على صورةٍ خاصةٍ بنا، فإذا كنت تعمل على جهازك المحلي وترغب في استخدام صورٍ من جهازك، يمكنك استخدام أي محرر رسوماتٍ لإنشاء صورة بأبعاد 28 28xبكسل لأي رقمٍ تريده. مثلًا:

نزل الصورة وانقلها إلى مجلد المشروع) تأكد أنها باسم test_image.png أو غير اسمها في الشيفرة (ثم أضف في نهاية الملف main.pyهذا السطر البرمجي التالي لتحميل صورة الاختبار للرقم المكتوب بخط اليد:

```
img = np.array(Image.open("test_image.png").convert('L')).ravel()
```

إن الدالة openمن مكتبة الصور Imageتحمّل صورة الاختبار مثل مصفوفةٍ رباعية الأبعاد D4، حيث تحتوي على قنوات الألوان الثلاث الرئيسية RGB بالإضافة إلى الشفافية، ولكن هذا ليس نفس التمثيل الذي استخدمناه سابقًا عند القراءة من مجموعة البيانات باستخدام المكتبة البرمجيةTensorFlow ، لذلك سنحتاج للقيام ببعض المهام الإضافية ليتناسب تنسيق هذه الصور مع التنسيق الذي سبق واعتمدناه في الخوارزمية.
سنستخدم الدالة convertمع الوسيط Lالتقليل تمثيل D RGBA 4إلى قناة لونٍ رماديةٍ واحدةٍ، وسنُخزنها على هيئة مصفوفة .numpyوسنستدعي ravelلتسوية المصفوفة.
الآن بعد أن صححّنا بنية معلومات الصورة، يمكننا تشغيل الجلسة بنفس الطريقة السابقة، ولكن هذه المرة سنُرسل صورةً واحدةً فقط للاختبار. وسنضيف الشيفرة التالية للملف لاختبار الصورة وطباعة التصنيف الناتج، هكذا:

```
prediction = sess.run(tf.argmax(output_layer, 1), feed_dict={X: [img]})
print ("Prediction for test image:", np.squeeze(prediction))
```

وتُستدعى الدالة np.squeezeعلى المتغير predictionليُعِيد عددًا صحيحًا وفريدًا إلى المصفوفة. وسيتضح من الناتج أن الشبكة العصبية قد تعرفت على الصورة كرقم 4، هكذا:

```
Prediction for test image: 4
```

يمكنك الآن تجربة عملية اختبار الشبكة باستخدام صورٍ أكثر تعقيدًا مثل الأرقام المتشابهة مع الأرقام الأخرى، أو أرقامٍ مكتوبةٍ بخطٍ سيئٍ أو حتى خاطئةٍ، وذلك لمعرفة وقياس مدى نجاحها.

❖ متغيرات وطبقات الشبكات العصبية[7] :

في شبكة الأعصاب الاصطناعية(ANN) ، تتدفق البيانات من طبقة الإدخال، مرورًا بإحدى أو أكثر من الطبقات المخفية، إلى طبقة الإخراج. كل طبقة تتكون من خلايا عصبية (neurons) تتلقى المدخلات، وتعالجها، وتنقل الإخراج إلى الطبقة التالية. تعمل الطبقات معًا لاستخراج الميزات(features) ، وتحويل البيانات، وإجراء التنبؤات.
تتكون شبكة الأعصاب الاصطناعية عادة من ثلاثة أنواع رئيسية من الطبقات:
طبقة الإدخال(Input Layer)
الطبقات المخفية(Hidden Layers)
طبقة الإخراج(Output Layer)
كل طبقة تتكون من عقد (neurons) مترابطة. تعمل الطبقات معًا لمعالجة البيانات عبر سلسلة من التحولات.
الطبقات فيANN
الطبقات الأساسية فيANN

1. طبقة الإدخال(Input Layer)

طبقة الإدخال هي أول طبقة في شبكة الأعصاب الاصطناعية وهي مسؤولة عن استقبال البيانات الخام. تعادل خلايا هذه الطبقة الخصائص في البيانات المدخلة. على سبيل المثال، في معالجة الصور، قد تمثل كل خلية عصبية قيمة بكسل. لا تقوم طبقة الإدخال بأي عمليات حسابية، بل تقوم بتمرير البيانات إلى الطبقة التالية.
النقاط الأساسية:

- الدور :تستقبل البيانات الخام.
- الوظيفة :تمرر البيانات إلى الطبقات المخفية.
- المثال :بالنسبة للصورة، ستكون طبقة الإدخال تحتوي على خلايا عصبية لكل قيمة بكسل.

2. الطبقات المخفية(Hidden Layers)

الطبقات المخفية هي الطبقات الوسيطة بين طبقة الإدخال وطبقة الإخراج. تقوم هذه الطبقات بمعظم العمليات الحسابية التي يحتاجها الشبكة. يمكن أن يختلف عدد الطبقات المخفية وحجمها بناءً على تعقيد المهمة.
تقوم كل طبقة مخفية بتطبيق مجموعة من الأوزان (weights) والانحيازات (biases) على البيانات المدخلة، تليها دالة تفعيل (activation function)لإدخال غير خطية.

3. طبقة الإخراج(Output Layer)

طبقة الإخراج هي الطبقة الأخيرة في شبكة الأعصاب الاصطناعية. تنتج هذه الطبقة التنبؤات الإخراجية. عدد الخلايا العصبية في هذه الطبقة يتناسب مع عدد الفئات في مشكلة التصنيف أو عدد المخرجات في مشكلة الانحدار. دالة التفعيل المستخدمة في طبقة الإخراج تعتمد على نوع المشكلة:

- Softmaxللتصنيف متعدد الفئات.(multi-class classification)
- Sigmoidللتصنيف الثنائي.(binary classification)
- Linearللانحدار.(regression)

أنواع الطبقات المخفية في شبكات الأعصاب الاصطناعية

1. الطبقة الكثيفة(Dense or Fully Connected Layer)

الطبقة الكثيفة هي النوع الأكثر شيوعًا من الطبقات المخفية في شبكة الأعصاب الاصطناعية. كل خلية عصبية في طبقة كثيفة متصلة بكل خلية عصبية في الطبقات السابقة والتالية. تقوم هذه الطبقة بإجراء مجموع وزني للمدخلات وتطبيق دالة تفعيل لإدخال غير خطية. تساعد دالة التفعيل) مثلReLU ، Sigmoid، أو (Tanh الشبكة على تعلم الأنماط المعقدة.
النقاط الأساسية:

- الدور :يتعلم التمثيلات من البيانات المدخلة.
- الوظيفة :يقوم بإجراء مجموع وزني وتفعيل.
- المثال :شائع في الشبكات العصبية المتصلة بالكامل.

2. الطبقة التلافيفية(Convolutional Layer)

الطبقات التلافيفية تُستخدم بشكل رئيسي في الشبكات العصبية التلافيفية (CNNs) لمهام معالجة الصور. تقوم هذه الطبقات بتطبيق عمليات التلافيف (convolution) على المدخلات، مما يساعد في استخراج الهيكليات المكانية من البيانات. تستخدم الطبقات التلافيفية فلاتر (filters) لمسح المدخلات وإنشاء خرائط ميزات .(feature maps) يساعد ذلك في اكتشاف الحواف(edges) ، والنقوش(textures) ، والميزات البصرية الأخرى.
النقاط الأساسية:

- الدور :استخراج الميزات المكانية من الصور.
- الوظيفة :تطبيق التلافيف باستخدام الفلاتر.
- المثال :اكتشاف الحواف والنقوش في الصور.

3. الطبقة التكرارية(Recurrent Layer)

الطبقات التكرارية، مثل الذاكرة قصيرة وطويلة المدى (LSTM) والوحدات التكرارية المقفلة(GRU) ، تُستخدم في الشبكات العصبية التكرارية (RNNs) للبيانات التسلسلية مثل السلاسل الزمنية أو اللغة الطبيعية. تحتوي هذه الطبقات

على اتصالات تعود للخلف، مما يسمح للمعلومات بالاستمرار عبر الخطوات الزمنية. وهذا يجعلها مناسبة للمهام التي تكون فيها السياق والاعتمادية الزمنية مهمة.

النقاط الأساسية:

- الدور :معالجة البيانات التسلسلية مع الاعتمادات الزمنية.
- الوظيفة :الحفاظ على الحالة عبر الخطوات الزمنية.
- المثال :نمذجة اللغة، التنبؤ بالسلاسل الزمنية.

4. الطبقة الإسقاطية(Dropout Layer)

الطبقات الإسقاطية هي تقنية منتظمة تُستخدم لمنع الإفراط في التخصيص .(overfitting) تقوم بإسقاط عشوائي لجزء من الخلايا العصبية أثناء التدريب، مما يجبر الشبكة على تعلم ميزات أكثر قوة وتقليل الاعتماد على خلايا عصبية معينة. أثناء التدريب، يتم الاحتفاظ بكل خلية عصبية مع احتمال معين.(p)

النقاط الأساسية:

- الدور :منع الإفراط في التخصيص.
- الوظيفة :إسقاط الخلايا العصبية عشوائيًا أثناء التدريب.
- المثال :شائع في نماذج التعلم العميق لتحسين التعميم.

5. طبقة التجميع(Pooling Layer)

تُستخدم طبقة التجميع لتقليل الأبعاد المكانية للبيانات، وبالتالي تقليل العبء الحسابي والتحكم في الإفراط في التخصيص. الأنواع الشائعة من التجميع تشمل التجميع الأقصى (Max Pooling) و التجميع المتوسط (Average Pooling).

حالات الاستخدام :تقليص الأبعاد فيCNNs

6. طبقة تطبيع الدُفعات(Batch Normalization Layer)

طبقة تطبيع الدُفعات تقوم بتطبيع مخرجات طبقة التفعيل السابقة عن طريق طرح المتوسط للدُفعة وقسمة على الانحراف المعياري للدُفعة. يساعد ذلك في تسريع عملية التدريب وتحسين أداء الشبكة.

حالات الاستخدام :استقرار وتسريع التدريب.

❖ الشبكات العصبية التلافيفية CNN [8]:

ما هي الشبكة العصبية التلافيفية؟

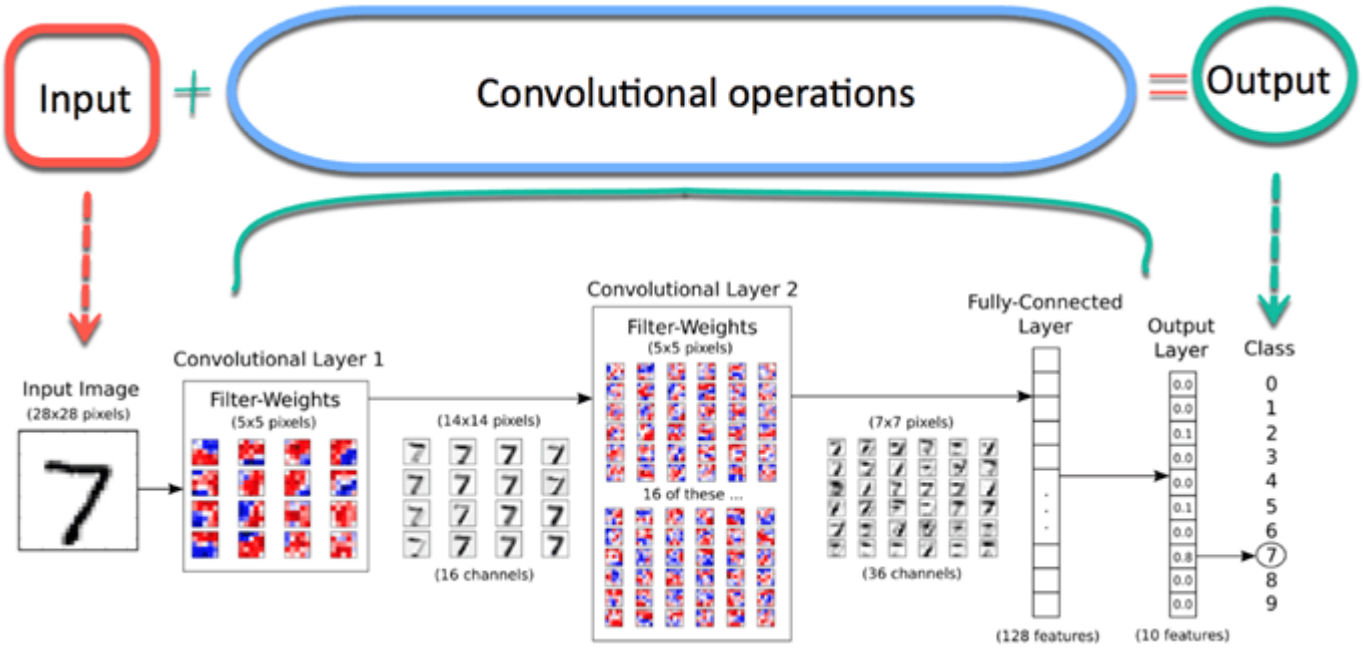
الشبكة العصبية التلافيفية : الشبكات العصبية العميقة، والمعروفة أيضًا باسم convnets أوCNN ، هي طريقة معروفة في تطبيقات الرؤية الحاسوبية. وهي فئة من الشبكات العصبية العميقة التي تستخدم لتحليل الصور المرئية. هذا النوع من الهندسة المعمارية مهيمن للتعرف على الأشياء من صورة أو فيديو. يتم استخدامه في تطبيقات مثل التعرف على الصور أو الفيديو، ومعالجة اللغة العصبية، وما إلى ذلك.

Archi بنية الشبكة العصبية التلافيفية

فكر في فيسبوك قبل بضع سنوات، بعد أن قمت بتحميل صورة إلى ملفك الشخصي، طُلب منك إضافة اسم إلى الوجه الموجود على الصورة يدويًا. في الوقت الحاضر، يستخدم Facebook شبكة convnet لوضع علامة على صديقك في الصورة تلقائيًا.

ليس من الصعب جدًا فهم الشبكة العصبية التلافيفية لتصنيف الصور. تتم معالجة الصورة المدخلة أثناء مرحلة التلافيفية ثم يتم إسناد تسمية إليها لاحقًا.

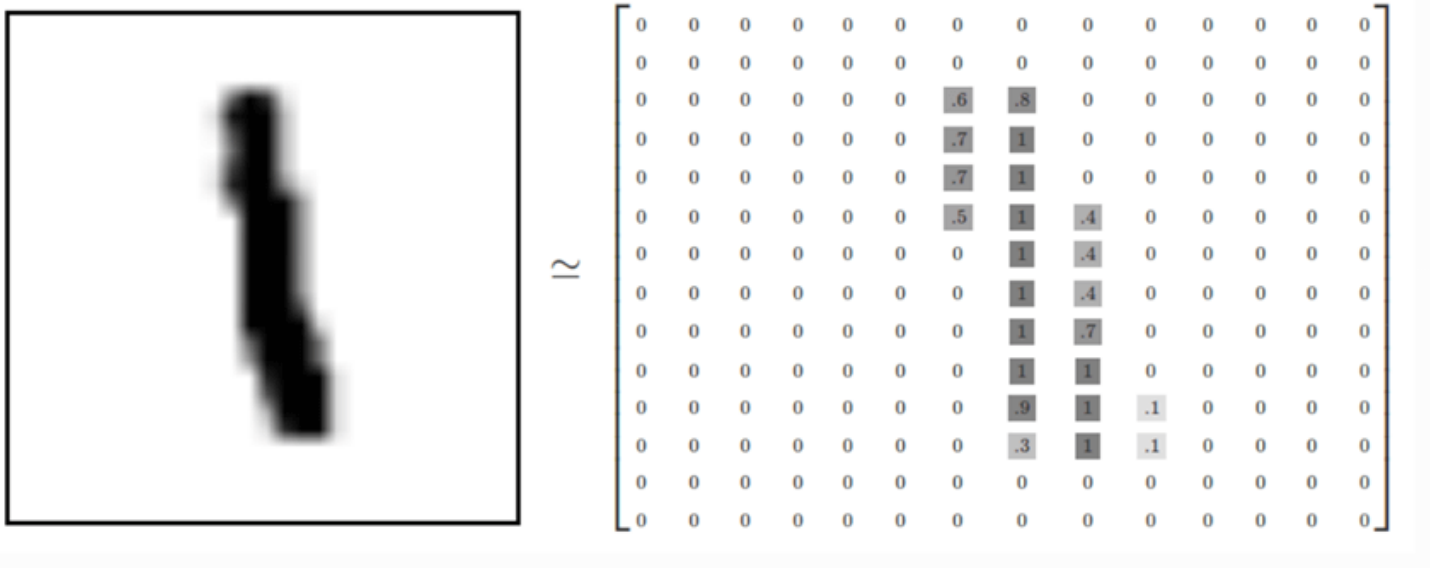
يمكن تلخيص بنية الشبكة التحويلية النموذجية في الصورة أدناه. أولاً، يتم دفع صورة إلى الشبكة؛ وهذا ما يسمى بالصورة المدخلة. بعد ذلك، تمر الصورة المدخلة بعدد لا نهائي من الخطوات؛ وهذا هو الجزء الملتف من الشبكة. وأخيرًا، يمكن للشبكة العصبية التنبؤ بالرقم الموجود في الصورة.



Archiبنية الشبكة العصبية التلافيفية(CNN)

تتكون الصورة من مجموعة من البكسلات ذات الارتفاع والعرض. تحتوي الصورة ذات التدرج الرمادي على قناة واحدة فقط بينما تحتوي الصورة الملونة على ثلاث قنوات (كل واحدة للأحمر والأخضر والأزرق). يتم تكديس القناة فوق بعضها البعض. في هذا البرنامج التعليمي، ستستخدم صورة ذات تدرج رمادي مع قناة واحدة فقط. كل بكسل له قيمة من 0 إلى 255 لتعكس شدة اللون. على سبيل المثال، البكسل الذي يساوي 0 سيظهر لونًا أبيض بينما البكسل الذي قيمته قريبة من 255 سيكون أغمق.

دعونا نلقي نظرة على الصورة المخزنة في مجموعة بيانات MNIST. توضح الصورة أدناه كيفية تمثيل الصورة على اليسار بتنسيق مصفوفة. لاحظ أنه تم توحيد المصفوفة الأصلية لتكون بين 0 و1. بالنسبة للألوان الداكنة، تبلغ القيمة في المصفوفة حوالي 0.9 بينما تبلغ قيمة وحدات البكسل البيضاء 0.



عملية التفافية

العنصر الأكثر أهمية في النموذج هو الطبقة التلافيفية. يهدف هذا الجزء إلى تقليل حجم الصورة لإجراء عمليات حسابية أسرع للأوزان وتحسين تعميمها.

أثناء الجزء التلافيفي، تحافظ الشبكة على السمات الأساسية للصورة وتستبعد الضوضاء غير ذات الصلة. على سبيل المثال، يتعلم النموذج كيفية التعرف على الفيل من صورة بها جبل في الخلفية. إذا كنت تستخدم شبكة عصبية تقليدية، فسيقوم النموذج بتعيين وزن لجميع وحدات البكسل، بما في ذلك وحدات البكسل الموجودة في الجبل، وهو أمر غير ضروري ويمكن أن يضلل الشبكة.

بدلاً من ذلك ، أ Kerasستستخدم الشبكة العصبية التلافيفية تقنية رياضية لاستخراج وحدات البكسل الأكثر صلة فقط. تسمى هذه العملية الرياضية بالالتفاف. تسمح هذه التقنية للشبكة بتعلم ميزات متزايدة التعقيد في كل طبقة. يقسم الالتفاف المصفوفة إلى أجزاء صغيرة لتعلم العناصر الأكثر أهمية داخل كل قطعة.

مكونات الشبكة العصبية التلافيفية ConvNet) أو(CNN

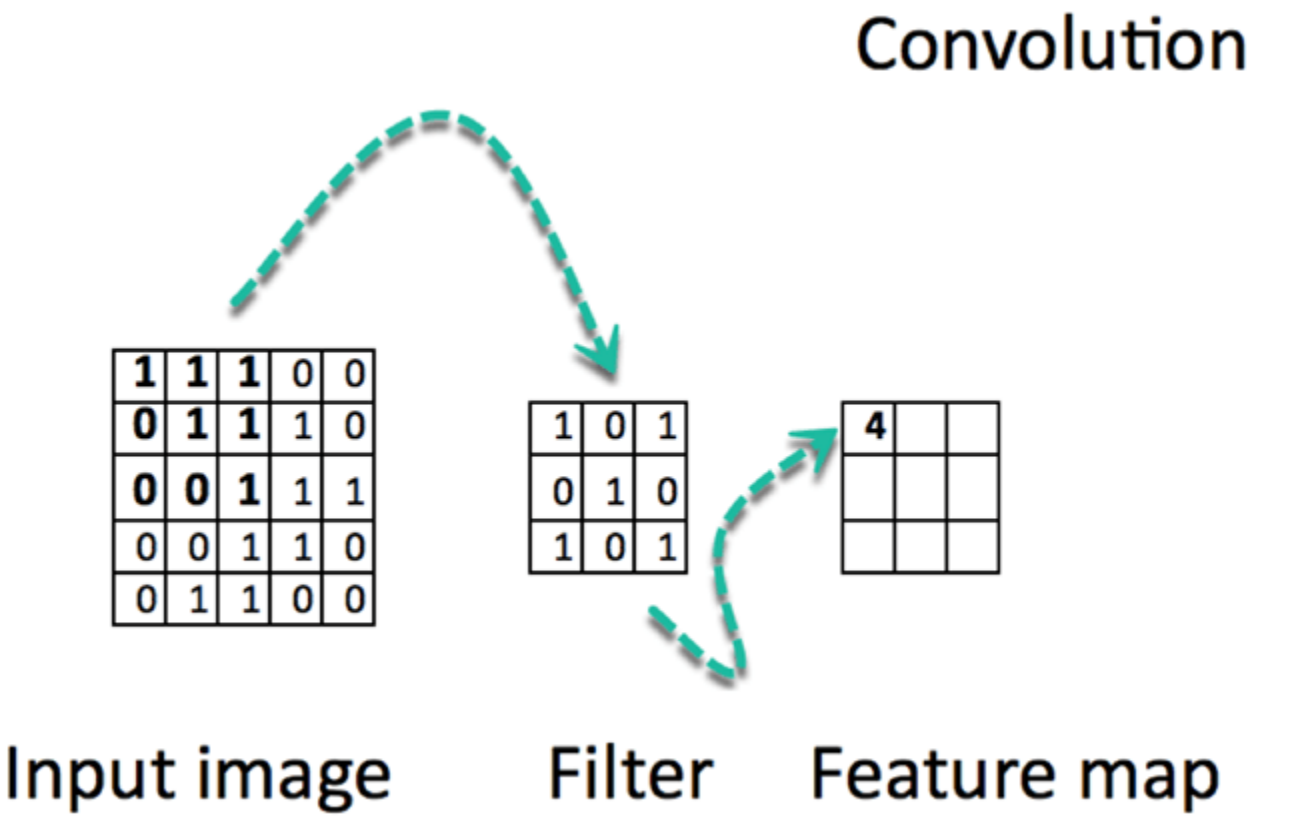
هناك أربعة مكونات لشبكةConvnets

1. التفاف
2. غير الخطية(ReLU)
3. Poolinز أو أخذ العينات الفرعية
4. التصنيف (طبقة متصلة بالكامل)

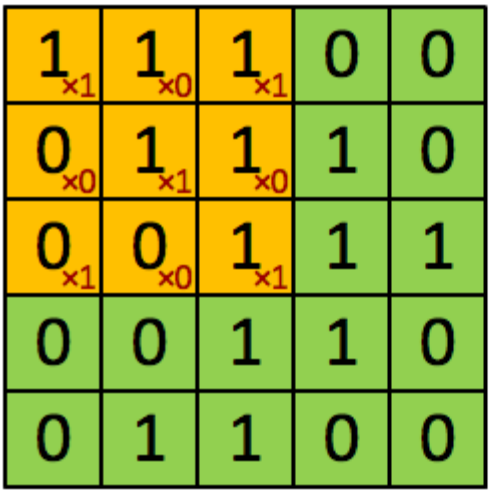
التفاف

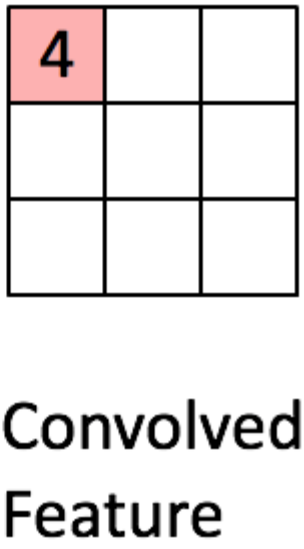
الغرض من الالتفاف هو استخراج ميزات الكائن الموجود على الصورة محليًا. وهذا يعني أن الشبكة ستتعلم أنماطًا محددة داخل الصورة وستكون قادرة على التعرف عليها في كل مكان في الصورة.

الالتفاف هو عملية ضرب على أساس كل عنصر. والمفهوم سهل الفهم. حيث يقوم الكمبيوتر بمسح جزء من الصورة، عادةً بأبعاد 3×3، ثم يقوم بضربه فى مرشح. ويطلق على ناتج عملية الضرب على أساس كل عنصر اسم خريطة الميزات. وتتكرر هذه الخطوة حتى يتم مسح الصورة بالكامل. ولاحظ أنه بعد عملية الالتفاف، يتم تقليل حجم الصورة.



يوجد أدناه عنوان URL لمعرفة كيفية عمل الالتفاف عمليًا.





هناك العديد من القنوات المتاحة. أدناه، قمنا بإدراج بعض القنوات. يمكنك أن ترى أن كل مرشح له غرض محدد. ملاحظة في الصورة أدناه؛ النواة هي مرادف للمرشح.

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

الحساب وراء الإلتواء

ستطبق المرحلة التلافيفية الفلتر على مجموعة صغيرة من وحدات البكسل داخل الصورة. سيتحرك الفلتر على طول الصورة المدخلة بشكل عام 3×3 أو 5×5. وهذا يعنى أن الشبكة ستنقل هذه النوافذ عبر الصورة المدخلة بالكامل وتحسب التلافيفية. توضح الصورة أدناه كيفية عمل التلافيفية. حجم الرقعة هو 3×3، ومصفوفة الإخراج هي نتيجة العملية على مستوى العنصر بين مصفوفة الصورة والفلتر.

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 \quad = 320$$

320					

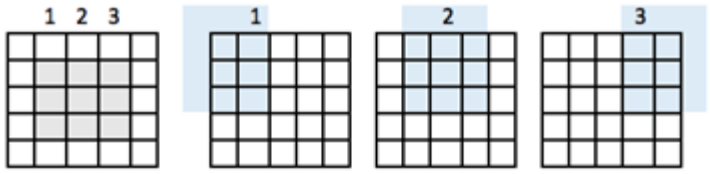
Output Matrix

Convolution with horizontal and vertical strides = 1

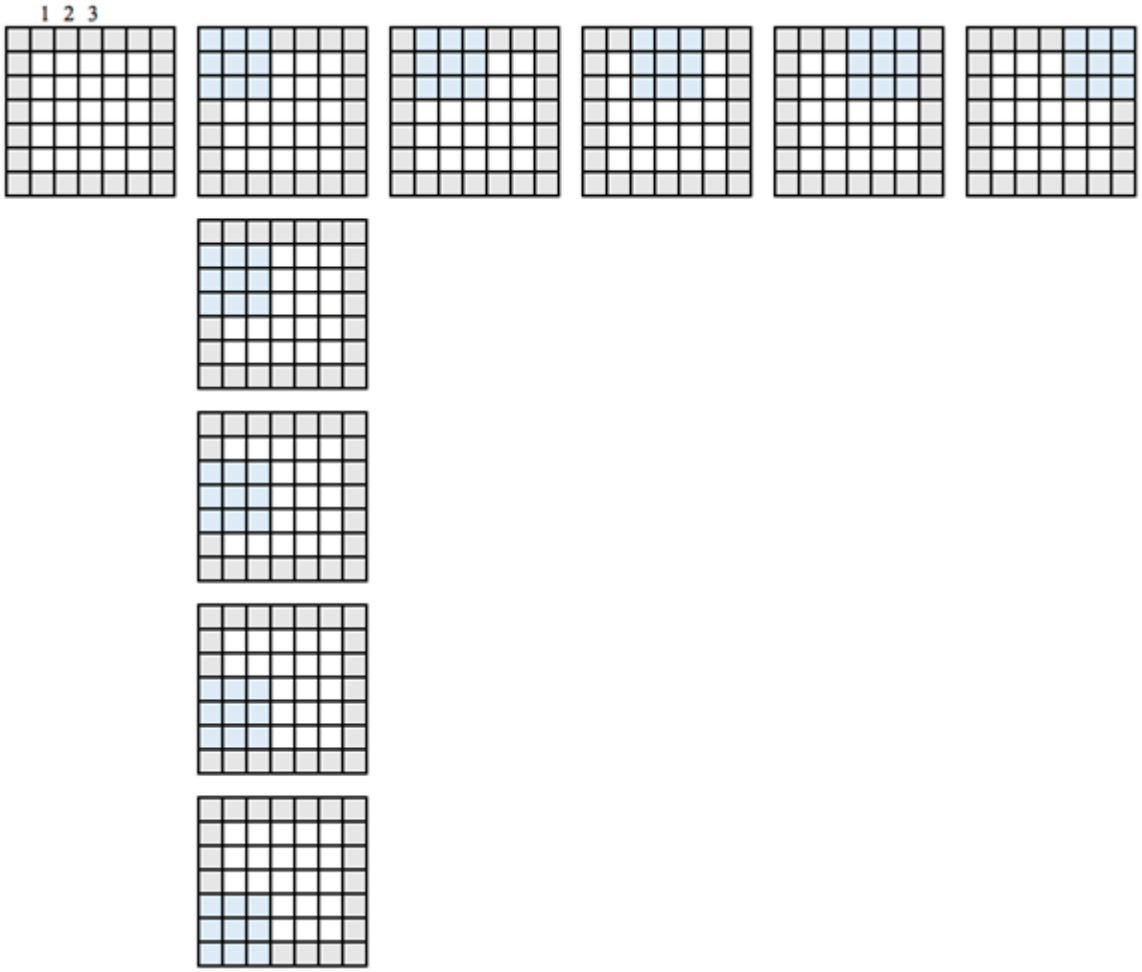
لاحظت أن عرض الإخراج وارتفاعه يمكن أن يختلفا عن عرض الإدخال وارتفاعه. يحدث ذلك بسبب تأثير الحدود.

تأثير الحدود

تحتوي الصورة على خريطة ميزات مقاس 5×5 ومرشح مقاس 3×3. توجد نافذة واحدة فقط في المنتصف حيث يمكن للمرشح فحص شبكة 3×3. سيتم تقليص خريطة ميزات الإخراج بمقدار قطعتين جنبًا إلى جنب مع بُعد 3 × 3.



للحصول على نفس البعد الناتج مثل البعد الإدخال، تحتاج إلى إضافة الحشو. الحشو يتكون من إضافة العدد الصحيح من الصفوف والأعمدة على كل جانب من المصفوفة. سيسمح للالتواء بالتوسيط ليناسب كل بلاطة إدخال. في الصورة أدناه، مصفوفة الإدخال/الإخراج لها نفس البعد 5×5



عند تحديد الشبكة، يتم التحكم في الميزات المجمعة بواسطة ثلاث معلمات:

1. عمق:يحدد عدد المرشحات التي سيتم تطبيقها أثناء الالتفاف. في المثال السابق، رأيت عمقًا قدره 1، مما يعني أنه يتم استخدام مرشح واحد فقط. في معظم الحالات، يوجد أكثر من مرشح واحد. تُظهر الصورة أدناه العمليات التي تم إجراؤها في موقف به ثلاثة مرشحات

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

$$0*0+0*-1+0*0 \\ +0*-1+105*5+102*-1 \\ +0*0+103*-1+99*0 = 320$$

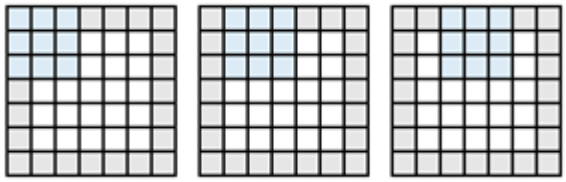
320				

Output Matrix

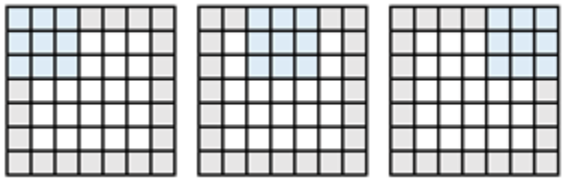
Convolution with horizontal and vertical strides = 1

2. خطوة:يحدد عدد "قفزات البكسل" بين شريحتين. إذا كانت الخطوة تساوي 1، ستتحرك النوافذ بمسافة بكسل واحدة. إذا كانت الخطوة تساوي 2، ستقفز النوافذ بمقدار XNUMX بكسل. إذا قمت بزيادة الخطوة، فستحصل على خرائط ميزات أصغر.

مثال الخطوة 1



الخطوة 2



3. حشوة صفر:الحشو هو عملية إضافة عدد مماثل من الصفوف والأعمدة على كل جانب من خرائط ميزات الإدخال. فى هذه الحالة، يكون للإخراج نفس أبعاد الإدخال.

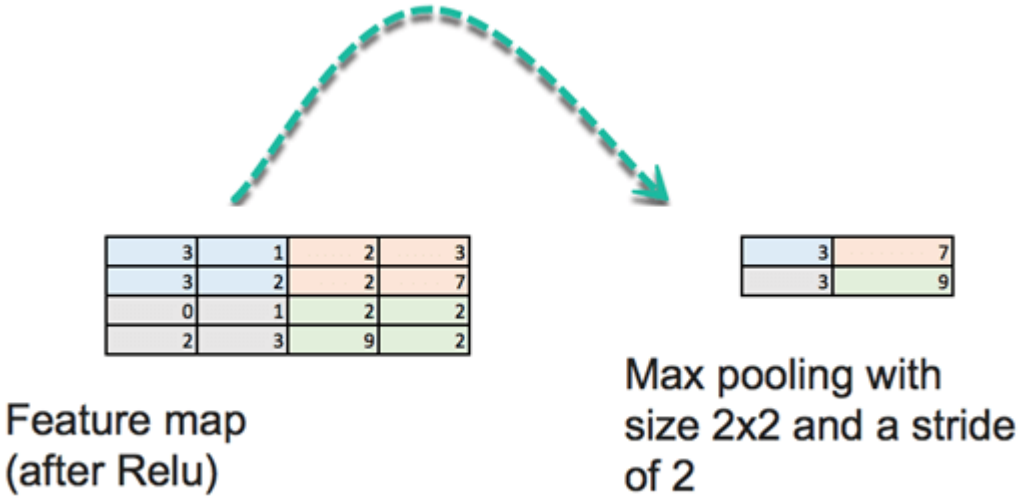
غير الخطية(ReLU)

في نهاية عملية الالتفاف، يخضع الناتج لدالة تنشيط للسماح باللاخطية. دالة التنشيط المعتادة لشبكة الالتفاف هي .Relu سيتم استبدال جميع وحدات البكسل ذات القيمة السالبة بالصفر.

Pooling Operaالإنتاج

هذه الخطوة سهلة الفهم. الغرض من التجميع هو تقليل أبعاد الصورة المدخلة. يتم تنفيذ الخطوات لتقليل التعقيد الحسابي للعملية. من خلال تقليل الأبعاد، يكون لدى الشبكة أوزان أقل للحساب، وبالتالي يمنع الإفراط في التجهيز.

في هذه المرحلة، تحتاج إلى تحديد الحجم والخطوة. الطريقة القياسية لتجميع صورة الإدخال هي استخدام القيمة القصوى لخريطة المعالم. انظر إلى الصورة أدناه. ستقوم عملية "التجميع" بفحص أربع مصفوفات فرعية من خريطة المعالم 4×4 وإرجاع القيمة القصوى. تأخذ عملية التجميع القيمة القصوى لمصفوفة 2×2 ثم تحرك هذه النوافذ بمقدار بكسلين. على سبيل المثال، المصفوفة الفرعية الأولى هي [3,1,3,2]، ستعيد عملية التجميع القيمة القصوى، وهي 3.



هناك عملية تجميع أخرى مثل المتوسط.

تعمل هذه العملية على تقليل حجم خريطة المعالم بشكل كبير

طبقات متصلة بالكامل

الخطوة الأخيرة تتكون من بناء التقليدية شبكة اعصاب صناعية كما فعلت في البرنامج التعليمي السابق. تقوم بتوصيل جميع الخلايا العصبية من الطبقة السابقة إلى الطبقة التالية. يمكنك استخدام وظيفة تنشيط softmax لتصنيف الرقم الموجود على صورة الإدخال.

خلاصة:

تقوم الشبكة العصبية التلافيفية TensorFlow بتجميع طبقات مختلفة قبل إجراء التنبؤ. تحتوي الشبكة العصبية على:

- طبقة تلافيفية
- وظيفة تفعيل ريلو
- Poolinطبقة ز
- طبقة متصلة بكثافة

تطبق الطبقات التلافيفية مرشحات مختلفة على منطقة فرعية من الصورة. تضيف وظيفة تنشيط Relu عدم الخطية، وتقلل طبقات التجميع من أبعاد خرائط الميزات.

كل هذه الطبقات تستخرج المعلومات الأساسية من الصور. أخيرًا، يتم تغذية خريطة الميزات إلى الطبقة الأساسية المتصلة بالكامل باستخدام وظيفة softmax لإجراء التنبؤ.

تدريب CNN معTensorFlow

الآن بعد أن أصبحت على دراية بالعناصر الأساسية لشبكات الاتصال، أنت جاهز لبناء واحدة باستخدامها .TensorFlowسوف نستخدم مجموعة بيانات MNIST لتصنيف صور.CNN

إن إعداد البيانات هو نفس ما تم في البرنامج التعليمي السابق. يمكنك تشغيل التعليمات البرمجية والانتقال مباشرة إلى بنية CNN.

ستتبع الخطوات أدناه لتصنيف الصور باستخدام CNN:

- ✓ الخطوة 1: تحميل مجموعة البيانات
- ✓ الخطوة 2: طبقة الإدخال
- ✓ الخطوة 3: الطبقة التلافيفية
- ✓ خطوة Poolin4 طبقة ز
- ✓ الخطوة 5: الطبقة التلافيفية الثانية وPooling طبقة ز
- ✓ الخطوة 6: طبقة كثيفة
- ✓ الخطوة 7: طبقة السجل

الخطوة 1: تحميل مجموعة البيانات

مجموعة بيانات MNIST متاحة مع scikit للتعرف عليها .URLيرجى تنزيله وتخزينه في التنزيلات. يمكنك تحميله باستخدام fetch_mldata('MNIST original').

قم بإنشاء مجموعة تدريب/اختبار

تحتاج إلى تقسيم مجموعة البيانات باستخدامTrain_test_split

مقياس الميزات

أخيرًا، يمكنك توسيع نطاق الميزة باستخدام MinMaxScaler كما هو موضح في تصنيف الصورة أدناه باستخدام مثال TensorFlow CNN.

```python
import numpy as np
import tensorflow as tf
from sklearn.datasets import fetch_mldata

#Change USERNAME by the username of your machine
## Windows USER
mnist = fetch_mldata('C:\\Users\\USERNAME\\Downloads\\MNIST original')
## Mac User
mnist = fetch_mldata('/Users/USERNAME/Downloads/MNIST original')

print(mnist.data.shape)
print(mnist.target.shape)
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(mnist.data, mnist.target, test_size=0.2, random_state=42)
y_train  = y_train.astype(int)
y_test  = y_test.astype(int)
batch_size =len(X_train)

print(X_train.shape, y_train.shape,y_test.shape )
## resclae
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
# Train
X_train_scaled = scaler.fit_transform(X_train.astype(np.float64))
# test
X_test_scaled = scaler.fit_transform(X_test.astype(np.float64))
feature_columns = [tf.feature_column.numeric_column('x', shape=X_train_scaled.shape[1:])]
X_train_scaled.shape[1:]
```

تعريف CNN :

تستخدم CNN مرشحات على البكسل الخام للصورة لتعلم نمط التفاصيل ومقارنته بالنمط العالمي باستخدام شبكة عصبية تقليدية. لإنشاءCNN ، تحتاج إلى تعريف:

1. طبقة تلافيفية: قم بتطبيق عدد n من المرشحات على خريطة المعالم. بعد الالتفاف، تحتاج إلى استخدام وظيفة تنشيط Reluلإضافة عدم الخطية إلى الشبكة.
2. Poolinطبقة :g الخطوة التالية بعد الالتفاف هي تقليل حجم الحد الأقصى للميزة. والغرض من ذلك هو تقليل أبعاد خريطة الميزة لمنع الإفراط في التجهيز وتحسين سرعة الحساب. تجميع الحد الأقصى هو التقنية التقليدية، التي تقسم خرائط الميزة إلى مناطق فرعية (عادةً بحجم 2×2) وتحتفظ فقط بالقيم القصوى.
3. الطبقات المتصلة بالكامل: جميع الخلايا العصبية من الطبقات السابقة متصلة بالطبقات التالية. ستقوم CNN بتصنيف العلامة وفقًا للميزات من الطبقات المتعرجة والمختزلة باستخدام طبقة التجميع.

هندسةCNN

- الطبقة التلافيفية: تطبق 14 مرشحًا 5×5 (استخراج مناطق فرعية 5×5 بكسل)، مع وظيفة تنشيطReLU
- :Pooling Layerتقوم بتنفيذ أقصى قدر من التجميع باستخدام مرشح 2×2 وخطوة 2 (التي تحدد أن المناطق المجمعة لا تتداخل)
- الطبقة التلافيفية: تطبق 36 مرشحًا 5×5، مع وظيفة تنشيطReLU
- Poolingالطبقة رقم 2: مرة أخرى، تقوم بتنفيذ أقصى تجميع باستخدام مرشح 2×2 وخطوة 2
- 1,764خلية عصبية، مع معدل تنظيم التسرب يبلغ 0.4 (احتمال 0.4 أن يتم إسقاط أي عنصر معين أثناء التدريب)
- الطبقة الكثيفة (طبقة اللوجيستات): 10 خلايا عصبية، واحدة لكل فئة مستهدفة من الأرقام.(0-9)

هناك ثلاث وحدات مهمة يمكن استخدامها لإنشاء:CNN

- .conv2d() إنشاء طبقة تلافيفية ثنائية الأبعاد تحتوي على عدد المرشحات وحجم نواة المرشح والحشو ووظيفة التنشيط كوسيطات.
- max_pooling2d() .يقوم بإنشاء طبقة تجميع ثنائية الأبعاد باستخدام خوارزمية التجميع الأقصى.
- كثيف(). يبني طبقة كثيفة بالطبقات والوحدات المخفية

الخطوة 2 طبقة الإدخال :

```
def cnn_model_fn(features, labels, mode):
    input_layer = tf.reshape(tensor = features["x"],shape =[-1, 28, 28, 1])

```

تحتاج إلى تحديد موتر مع شكل البيانات. لذلك، يمكنك استخدام الوحدة .tf.reshape في هذه الوحدة، عليك أن تعلن عن إعادة تشكيل الموتر وشكل الموتر. الوسيطة الأولى هي ميزات البيانات، والتي تم تعريفها في وسيطة الوظيفة.

الصورة لها ارتفاع وعرض وقناة. مجموعة بيانات MNIST عبارة عن صورة أحادية اللون بحجم 28 × 28. قمنا بتعيين حجم الدفعة على -1 في وسيطة الشكل بحيث تأخذ شكل الميزات .["x"] الميزة هي جعل المعلمات الفائقة لحجم الدفعة قابلة للضبط. إذا تم ضبط حجم الدفعة على 7، فسيقوم الموتر بتغذية 5,488 قيمة.(7*28*28)

الخطوة 3 الطبقة التلافيفية :

```
# first Convolutional Layer
  conv1 = tf.layers.conv2d(
      inputs=input_layer,
      filters=14,
      kernel_size=[5, 5],
      padding="same",
      activation=tf.nn.relu)

```

تحتوي الطبقة التلافيفية الأولى على 14 مرشحًا بحجم نواة 5×5 وبنفس الحشو. نفس الحشو يعني أن كلاً من موتر الإخراج وموتر الإدخال يجب أن يكون لهما نفس الارتفاع والعرض. سيضيف Tensorflow أصفارًا إلى الصفوف والأعمدة لضمان نفس الحجم.

يمكنك استخدام وظيفة التنشيط .Relu سيكون حجم الإخراج[28 ، 28، 14].

الخطوة 4 Poolin طبقة ز :

الخطوة التالية بعد الالتفاف هى حساب التجميع. سيعمل حساب التجميع على تقليل أبعاد البيانات. يمكنك استخدام وحدة max_pooling2d بحجم 2×2 وخطوة 2. يمكنك استخدام الطبقة السابقة كمدخل. سيكون حجم الإخراج[batch_size, 14, 14, 14]

```
# first Pooling Layer
pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2, 2], strides=2)
```

الخطوة 5: الطبقة التلافيفية الثانية وPooling طبقة ز

تحتوي الطبقة التلافيفية الثانية على 32 مرشحًا، بحجم إخراج .[batch_size, 14, 14, 32] تتمتع طبقة التجميع بنفس الحجم كما في السابق وشكل الإخراج هو.[batch_size, 14, 14, 18]

```
conv2 = tf.layers.conv2d(
    inputs=pool1,
    filters=36,
    kernel_size=[5, 5],
    padding="same",
    activation=tf.nn.relu)
pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=[2, 2], strides=2)
```

الخطوة 6 طبقة كثيفة :

بعد ذلك، تحتاج إلى تحديد الطبقة المتصلة بالكامل. يجب أن يتم تسوية خريطة المعالم قبل أن يتم ربطها بالطبقة الكثيفة. يمكنك استخدام إعادة تشكيل الوحدة بحجم 7*7*36.

ستربط الطبقة الكثيفة 1764 خلية عصبية. قمت بإضافة وظيفة تفعيل .Relu علاوة على ذلك، يمكنك إضافة مصطلح تسوية التسرب بمعدل 0.3، مما يعني أنه سيتم تعيين 30 بالمائة من الأوزان على 0. لاحظ أن التسرب يحدث فقط أثناء مرحلة التدريب. تحتوي الدالة cnn_model_fn على وضع وسيطة للإعلان عما إذا كان النموذج بحاجة إلى التدريب أو التقييم كما هو موضح في مثال TensorFlow لتصنيف صور CNN أدناه.

```
pool2_flat = tf.reshape(pool2, [-1, 7 * 7 * 36])

dense = tf.layers.dense(inputs=pool2_flat, units=7 * 7 * 36, activation=tf.nn.relu)
dropout = tf.layers.dropout(
    inputs=dense, rate=0.3, training=mode == tf.estimator.ModeKeys.TRAIN)
```

الخطوة 7 طبقة السجل :

وأخيرًا، في مثال تصنيف الصورTensorFlow ، يمكنك تحديد الطبقة الأخيرة مع التنبؤ بالنموذج. شكل الإخراج يساوي حجم الدفعة و 10، العدد الإجمالي للصور.

```
# Logits Layer
logits = tf.layers.dense(inputs=dropout, units=10)
```

يمكنك إنشاء قاموس يحتوي على الفئات واحتمال كل فئة. تقوم الوحدة ()tf.argmax بإرجاع أعلى قيمة إذا كانت طبقات السجل. ترجع الدالة softmaxاحتمالية كل فئة.

```
predictions = {
   # Generate predictions
   "classes": tf.argmax(input=logits, axis=1),
   "probabilities": tf.nn.softmax(logits, name="softmax_tensor")  }
```

أنت تريد فقط إرجاع تنبؤات القاموس عندما يتم ضبط الوضع على التنبؤ. يمكنك إضافة هذه الرموز لإبعاد التوقعات

```
if mode == tf.estimator.ModeKeys.PREDICT:
   return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)
```

الخطوة التالية هي حساب خسارة النموذج. في البرنامج التعليمي السابق، تعلمت أن دالة الخسارة لنموذج متعدد الفئات هي إنتروبيا متقاطعة. يمكن حساب الخسارة بسهولة باستخدام الكود التالي:

```
# Calculate Loss (for both TRAIN and EVAL modes)
```

```
loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits)
```

الخطوة الأخيرة في مثال TensorFlow CNN هي تحسين النموذج، أي العثور على أفضل قيم الأوزان. للقيام بذلك، يمكنك استخدام مُحسِّن نزول متدرج بمعدل تعلم يبلغ 0.001. الهدف هو تقليل الخسارة

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001)
train_op = optimizer.minimize(
    loss=loss,
    global_step=tf.train.get_global_step())
```

عرض مقاييس الأداء أثناء وضع التقييم. مقاييس الأداء لنموذج متعدد الفئات هى مقاييس الدقة. تم تجهيز Tensorflow بدقة وحدة مع وسيطتين، والتسميات، والقيم المتوقعة.

```
eval_metric_ops = {
    "accuracy": tf.metrics.accuracy(labels=labels, predictions=predictions["classes"])}
return tf.estimator.EstimatorSpec(mode=mode, loss=loss, eval_metric_ops=eval_metric_ops)
```

كود الخطوات السابقة كاملًا :

```python
def cnn_model_fn(features, labels, mode):
  """Model function for CNN."""
  # Input Layer
  input_layer = tf.reshape(features["x"], [-1, 28, 28, 1])

  # Convolutional Layer
  conv1 = tf.layers.conv2d(
      inputs=input_layer,
      filters=32,
      kernel_size=[5, 5],
      padding="same",
      activation=tf.nn.relu)

  # Pooling Layer
  pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2, 2], strides=2)

  # Convolutional Layer #2 and Pooling Layer
  conv2 = tf.layers.conv2d(
      inputs=pool1,
      filters=36,
      kernel_size=[5, 5],
      padding="same",
      activation=tf.nn.relu)
  pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=[2, 2], strides=2)

  # Dense Layer
  pool2_flat = tf.reshape(pool2, [-1, 7 * 7 * 36])
  dense = tf.layers.dense(inputs=pool2_flat, units=7 * 7 * 36, activation=tf.nn.relu)
  dropout = tf.layers.dropout(
      inputs=dense, rate=0.4, training=mode == tf.estimator.ModeKeys.TRAIN)

  # Logits Layer
  logits = tf.layers.dense(inputs=dropout, units=10)

  predictions = {
      # Generate predictions (for PREDICT and EVAL mode)
      "classes": tf.argmax(input=logits, axis=1),
      "probabilities": tf.nn.softmax(logits, name="softmax_tensor")
  }

  if mode == tf.estimator.ModeKeys.PREDICT:
    return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)

  # Calculate Loss
  loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits)

  # Configure the Training Op (for TRAIN mode)
  if mode == tf.estimator.ModeKeys.TRAIN:
    optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001)
    train_op = optimizer.minimize(
        loss=loss,
        global_step=tf.train.get_global_step())
    return tf.estimator.EstimatorSpec(mode=mode, loss=loss, train_op=train_op)

  # Add evaluation metrics Evaluation mode
  eval_metric_ops = {
      "accuracy": tf.metrics.accuracy(
          labels=labels, predictions=predictions["classes"])}
  return tf.estimator.EstimatorSpec(
      mode=mode, loss=loss, eval_metric_ops=eval_metric_ops)

```

....بإختصار :

أولاً، عليك تحديد مُقدِّر باستخدام نموذج CNN لتصنيف الصور.

```
# Create the Estimator
mnist_classifier = tf.estimator.Estimator(
   model_fn=cnn_model_fn, model_dir="train/mnist_convnet_model")
```

يستغرق تدريب CNN عدة مرات، لذلك يمكنك إنشاء خطاف تسجيل لتخزين قيم طبقات softmax كل 50 تكرارًا.

```
# Set up logging for predictions
tensors_to_log = {"probabilities": "softmax_tensor"}
logging_hook = tf.train.LoggingTensorHook(tensors=tensors_to_log, every_n_iter=50)
```

أنت جاهز لتقدير النموذج. قمت بتعيين حجم دفعة من 100 وخلط البيانات. لاحظ أننا قمنا بتعيين خطوات التدريب على 16.000، وقد يستغرق التدريب الكثير من الوقت. كن صبوراً.

```
# Train the model
train_input_fn = tf.estimator.inputs.numpy_input_fn(
   x={"x": X_train_scaled},
   y=y_train,
   batch_size=100,
   num_epochs=None,
   shuffle=True)
mnist_classifier.train(
   input_fn=train_input_fn,
   steps=16000,
   hooks=[logging_hook])
```

الآن بعد أن تم تدريب النموذج، يمكنك تقييمه وطباعة النتائج

```
# Evaluate the model and print results
eval_input_fn = tf.estimator.inputs.numpy_input_fn(
   x={"x": X_test_scaled},
   y=y_test,
   num_epochs=1,
   shuffle=False)
eval_results = mnist_classifier.evaluate(input_fn=eval_input_fn)
print(eval_results)
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2018-08-05-12:52:41
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from train/mnist_convnet_model/model.ckpt-15652
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2018-08-05-12:52:56
INFO:tensorflow:Saving dict for global step 15652: accuracy = 0.9589286, global_step = 15652, loss = 0.13894269
{'accuracy': 0.9689286, 'loss': 0.13894269, 'global_step': 15652}
```

مع البنية الحالية، تحصل على دقة 97%. يمكنك تغيير البنية وحجم الدفعة وعدد التكرارات لتحسين الدقة. لقد كان أداء الشبكة العصبية CNN أفضل بكثير من ANN أو الانحدار اللوجستي. في البرنامج التعليمي حول الشبكة العصبية الاصطناعية، حصلت على دقة 96%، وهي أقل من CNN.أداء CNN مثير للإعجاب مع صورة أكبر طقمسواء من حيث سرعة الحساب أو الدقة.

❖ المراجع :

[1] "ما هو TensorFlow؟ مقدمة كيف تعمل ؟ & Architecture." Accessed: Feb. 15, 2025. [Online]. Available: https://www.guru99.com/ar/what-is-tensorflow.html

[2] "Install TensorFlow with pip," TensorFlow. Accessed: Feb. 15, 2025. [Online]. Available: https://www.tensorflow.org/install/pip

[3] "تسعة أشياء يجب معرفتها عن مكتبة تينسر فلو TensorFlow للتعلّم العميق من جوجل – الذّكاءُ الإصطناعيُّ باللُّغةِ العربيّةِ." Accessed: Feb. 15, 2025. [Online]. Available: https://aiinarabic.com/9-things-you-should-know-about-tensorflow/

[4] "تسعة أشياء يجب معرفتها عن مكتبة تينسر فلو TensorFlow للتعلّم العميق من جوجل – الذّكاءُ الإصطناعيُّ باللُّغةِ العربيّةِ." Accessed: Jan. 30, 2025. [Online]. Available: https://aiinarabic.com/9-things-you-should-know-about-tensorflow/

[5] "نمذجيات؟," كيف تصمم الشبكة العصبية الخاصة بك. Accessed: Feb. 15, 2025. [Online]. Available: https://www.nmthgiat.com/%d9%83%d9%8a%d9%81-%d8%aa%d8%b5%d9%85%d9%85-%d8%a7%d9%84%d8%b4%d8%a8%d9%83%d8%a9-%d8%a7%d9%84%d8%b9%d8%b5%d8%a8%d9%8a%d8%a9-%d8%a7%d9%84%d8%ae%d8%a7%d8%b5%d8%a9-%d8%a8%d9%83%d8%9f/

[6] "بناء شبكة عصبية للتعرف على الأرقام المكتوبة بخط اليد باستخدام مكتبة TensorFlow," أكاديمية حسوب. Accessed: Feb. 15, 2025. [Online]. Available: https://academy.hsoub.com/programming/artificial-intelligence/%D8%A8%D9%86%D8%A7%D8%A1-%D8%B4%D8%A8%D9%83%D8%A9-%D8%B9%D8%B5%D8%A8%D9%8A%D8%A9-%D9%84%D9%84%D8%AA%D8%B9%D8%B1%D9%81-%D8%B9%D9%84%D9%89-%D8%A7%D9%84%D8%A3%D8%B1%D9%82%D8%A7%D9%85-%D8%A7%D9%84%D9%85%D9%83%D8%AA%D9%88%D8%A8%D8%A9-%D8%A8%D8%AE%D8%B7-%D8%A7%D9%84%D9%8A%D8%AF-%D8%A8%D8%A7%D8%B3%D8%AA%D8%AE%D8%AF%D8%A7%D9%85-%D9%85%D9%83%D8%AA%D8%A8%D8%A9-tensorflow-r1267/

[7] "Layers in Artificial Neural Networks (ANN)," GeeksforGeeks. Accessed: Feb. 15, 2025. [Online]. Available: https://www.geeksforgeeks.org/layers-in-artificial-neural-networks-ann/

[8] "مقدمة لفهم الشبكات العصبية التلافيفية | Convolutional Neural Network (CNN) - الجزء الثاني." Accessed: Feb. 15, 2025. [Online]. Available: https://fihm.ai/tutorials/%d9%85%d9%82%d8%af%d9%85%d8%a9-%d9%84%d9%81%d9%87%d9%85-%d8%a7%d9%84%d8%b4%d8%a8%d9%83%d8%a7%d8%aa-%d8%a7%d9%84%d8%b9%d8%b5%d8%a8%d9%8a%d8%a9-%d8%a7%d9%84%d8%aa%d9%84%d8%a7%d9%81%d9%8a%d9%81%d9%8a-2/