□□ الآلة تعلم إلى مقدمة

امب اننأ متعلن Python، الف دب أن دثحتن اضًأي عن متعلت الآلة. تعلم ل أن ديدعلا من اهتابتكم ةبوتكم ةغلب قليلاً. بها والعلب بتثبيتها لنبدأ .□□□□□□

□□□□□□□□□□

بالتثبيت. قم

```
$ pip install tensorflow
ERROR: Could not find a version that satisfies the requirement tensorflow
ERROR: No matching distribution found for tensorflow
```

ترجمة:

```
$ pip install tensorflow
:                            tensorflow
:                         tensorflow

$ type python
python             `/usr/local/Cellar/python@3.9/3.9.1_6/bin/python3'
```

3.9. الإصدار نحن نستخدم بينما .3.8□3.5 Python إصدارات فقط يدعم Tensorflow 2 فإن ذلك، ومع

```
% type python3
python3    /usr/bin/python3
% python3 -V
Python 3.8.2
```

من الإصدار لهذا المقابل pip تثبيت يتم أين نظامي. على مثبت python3 من 3.8.2 الإصدار أن لاحظت □□□□□□؟

```
% python3 -m pip -V
pip 21.0.1   /Users/lzw/Library/Python/3.8/lib/python/site-packages/pip (python 3.8)
```

الخاص shell بتغيير قمت مؤخرًا .zprofile. ملف بتعديل سأقوم لذا هنا. موجودة pip من المقابلة النسخة سطر. بإضافة سأقوم السابق. bash_profile. ملف يعادل zprofile. ملف بي.

```
alias pip3=/Users/lzw/Library/Python/3.8/bin/pip3
```

ملاحظة: ص النص أعلاه هو أمر في سطر الأوامر يستخدم لإنشاء اسم مختصر لـ pip3 أمر لتنفيذ البرمجية. التعليمات من جزء أنه حيث ترجمة إلى النص هذا يحتاج لا محدد. مسار من pip3 مع Tensorflow.

نستخدم الطريقة، بهذه python3 و pip3 بعلل

```
% pip3 install tensorflow
...
      :    absl-py-0.12.0 astunparse-1.6.3 cachetools-4.2.1 certifi-2020.12.5 chardet-4.0.0 flatbuffers-
```

الرسمي. الموقع من مثالًا واستخدمت المكتبات. من العديد تثبيت تم

```
import tensorflow as tf
```

تحميل وتهيئة مجموعة بيانات MNIST

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
```

لتمثيل النموذج التسلسلي باستخدام مكتبة Keras و TensorFlow. النموذج يتكون من:

1. **طبقة تسوية**: تقوم بتحويل البيانات ذات الشكل (28, 28) إلى مصفوفة أحادية البعد (784,).
2. **طبقة كثيفة**: تستخدم دالة التنشيط relu وتحتوي على 128 عصبونة طبقة عصبية كاملة الاتصال.
3. **طبقة تسوية**: تقوم بإسقاط 20% من الوحدات العصبية لكل عينة تدريب أثناء الإفراط في التخصيص.
4. **طبقة كثيفة**: تحتوي على 10 عصبونات طبقة عصبية كاملة الاتصال تستخدم لتصنيف المدخلات إلى 10 فئات.

هذا النموذج يمكن استخدامه لتصنيف الصور في مجموعة بيانات MNIST.

```
predictions = model(x_train[:1]).numpy()
print(predictions)
```

.

```shell
$ /usr/bin/python3 tf.py
              https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 10s 1us/step
[[ 0.15477428 -0.3877643   0.0994779   0.07474922 -0.26219758 -0.03550266
   0.32226565 -0.37141111  0.10925996 -0.0115255 ]]
```

تم لزيزنت عةوموجم تانايبلا، حاجنب مث تم ضرع. جئاتنلا

قِلنل رظنة نآلا ىلع لاثم فينصت. روصلا

```
# TensorFlow   tf.keras
import tensorflow as tf
```

مساعدة مكتبات

[illegible]

```
print(tf.__version__)
```

.خطأ

```
ModuleNotFoundError:            'matplotlib'
```

.قم بالتثبيت

```
% pip3 install matplotlib
```

.تم التصحيح

```
$ /usr/bin/python3 image.py
2.4.1
```

لكود: ولصق نسخ لكيفية على مثال إليك

```
#                    Python

def greet(name):
    print(f"    {name}!")

#
greet("  "(
```

كنكمي خسن اذه ود الكه ولصق في رحرر مص وصن أو ئيبي ريوطت ةلماكتم □□□□□. لتشغيله.

```
# TensorFlow   tf.keras
import tensorflow as tf

#
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

تم ليمحت انايب □□□□□□□ □□□□□ اهميسقتو إلى: نيتعومجم □ ةعومجم بيردتلا: □ train_images: الملقابة الصنيفات على تحتوي train_labels: □ للتدريب. المستخدمة الملابس صور على تحتوي ور□ التدريب. لصور □ ةعومجم ارباختلا: □ test_images: تحتوي على صور الملابس المستخدمة للاختبار. الصور لقابلة المصنفات على تحتوي test_labels: الاختبار.

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
print(train_images.shape)
print(len(train_labels))
```

تم إخراج النتائج. لاحظ انه وجود train_images و train_labels و test_images و test_labels. هي مسقمة إلى مجموعات بيانات التدريب واختبار البيانات.

```
(60000, 28, 28)
60000
```

الصورة. طباعة ذلك بعد لنجرب

```
print(train_images[0])
```

النتائج. على نظرة لنلقِ

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   1   0   0  13  73   0
    0   1   4   0   0   0   0   1   1   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3   0  36 136 127  62
   54   0   0   0   1   3   4   0   0   3]
 [  0   0   0   0   0   0   0   0   0   0   0   0   6   0 102 204 176 134
  144 123  23   0   0   0   0  12  10   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 155 236 207 178
  107 156 161 109  64  23  77 130  72  15]
 [  0   0   0   0   0   0   0   0   0   0   0   1   0  69 207 223 218 216
  216 163 127 121 122 146 141  88 172  66]]
  ....
```

النتائج. بعض من مقتطفات هنا

```
print(len(train_images[0][0]))
```

الطباعة. تابع 28. بعرض مصفوفة هذه أن الواضح من لذلك 28. هو الإخراج

```
    print(len(train_images[0][0][0]))
TypeError: object of type 'numpy.uint8' has no len()
```

الترجمة:

```
    print(len(train_images[0][0][0])
TypeError:           'numpy.uint8'            len()
```

وهو .□□□ قيم يحفظ المصفوفة من الأخير البعد .3*28*28 من مصفوفة هي صورة لكل جدًا. الواضح من لذلك، خاطئة. تكون قد فكرتنا أن الكتشفنا ذلك،

```
print(train_images[0][1][20])
```

```
0
```

```
print(train_images[0][1])
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

أخيرًا. السر الكتشفنا التجارب، بعض بعد .28□28 بحجم مصفوفة عن عبارة صورة لكل

النواتج. الصورة على نظرة بإلقاء أولاً لنبدأ

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```

ملاحظة: بيانات مجموعة من صورة لعرض matplotlib مكتبة ويستخدم □□□□□□ بلغة مكتوب أعلاه الكود البرمجة. لغة من جزء وهي بالإنجليزية مكتوبة والمتغيرات الأوامر أن حيث ترجمة إلى الكود يحتاج لا التدريب.

لونين؟ أي يعرف كيف لكن لونين. بين تدرج هذا الأصل، في .250 إلى 0 من اليمين؟ على الألوان شريط ترى هل بذلك؟ أخبرناه أين

أيضًا. الثانية الصورة بطباعة نقوم ثم

```
plt.imshow(train_images[1])
```

من المقدم الكود تشغيل نواصل دعنا التابعة؟ pyplot لمكتبة الافتراضي الإعداد هو هذا هل للاهتمام. مثير الرسمي. الموقع

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
```

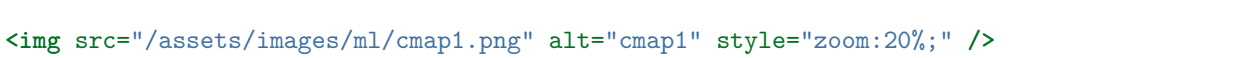
```
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

بالتأكيد فسيكون، cmp في شيء أي نكتب لم إذا .cmp معلمة عرفنا أخيرًا، معروضة. وفيئاتها الصور أن هنا لاحظ بالفعل. سابقًا. رأيناه الذي باللون

```
    plt.imshow(train_images[i])
```

المصادر. بعض وجدنا لقد .pyplot cmap عن سنبحث المرة هذه

```
    plt.imshow(train_images[i], cmap=plt.cm.PiYG)
```



المعدلة: النسخة إليك الكود. بتعديل قمت لقد

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)    ##
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.Blues)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

من بدلاً أعمدة و5 صفوف 5 على تحتوي الشبكة لجعل 5,5 إلى 2,5 بتغيير قمت تعديله، تم الذي السطر في 5□5. شبكة في صورة 25 بعرض سيسمح هذا أعمدة. و5 صفوف 2

خطأ. حدث ذلك، ومع

```
ValueError:         num    1    10    11
```

أننا من الرغم على 2؟ إلى تغييرها عند تعمل لا ولماذا بالضبط؟ السابقة 5,5,i+1 تعنيه الذي ما هذا؟ يعني ماذا num؟ معنى وما 11؟ حساب تم كيف الخطأ؟ هذا يظهر لماذا إلا أعمدة، و5 صفوف 5 تعني ربما أنها حدسي بشكل نعرف for i in range(10):، إلى تغييرها عند i=11. كانت عندما الخطأ حدث ربما لذا .2*5=10 أن لاحظ 10؟ معنى وما الثانية. النتيجة على حصلنا

أن نجد الوثائق، إلى قليلاً بالنظر .subplot(nrows, ncols, index, **kwargs)، الأمر أصبح الآن حتى حسنًا، لنا. واضحًا

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    # plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.Blues)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

مخ تلف ة. أش كال س تظ هر ال إط ار، ه ذا ت صغ ير أو ب تك بير ن قوم ع ندما ي سمى ال نوع ه ذا xticks. أن ل احظ 0 25

□□□□□□□□□□□

مخ تلف. ع رض ل ه ا س يكون xlabels و xticks وال تص غير، ال تك بير م ربع أن ل احظ

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
```

ال ع ربية: إلى ال ك ود ت رجمة

```
    = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
```

ث لاث من ي تك ون ال نموذج و□□□□□. □□□□□□□□□□□ م كتبة ب تك اس تخدام س لسلة ت سل سلي نموذج إن شاء ي تم ال كود، ه ذا ف ي ط بقات:

1. □□□□□□□: ت قوم ل يو ح تب ات ان اي ال بي ان ات ث نائ ية ال أبعاد ال□28□28 إلى ف وف ص مف وفة أحادية ال بعد.
2. □□□□□: ط بقة ع صبية ك ثيفة ي وت حت ى ل ع 128 ع صب ون وت س تخدم دالة ال ت نش يط □□□□.
3. □□□□□: ط بقة ع صبية ك ثيفة ي وت حت ى ل ع 10 ات ون ع صبية ع ادةً □ ا م دم ت س تخ دم في ف ي ن ص ت ور ال ص ور إلى 10 ف ئات□.

ه ذا نموذج ال ن بس يط ي س تخدم ع ادةً في م ا م ه ف ي ن ص ت ور ال ص ور ل ث م □□□□□.

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=10)

test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)

print('\n   :    ', test_acc)
```

ال28,28 الم عل مات: هذه ل احظ .Sequential الف ئة ب اس تخ دام النم وذج تع ري ف ا ب ه ي تم ال تي الط ري قة ه نا ل احظات ال ص ورة. ح جم ه و 28,28 أن ل احظ ال مل اءم ة. أو ال تو في ق تع ني fit. fit و compile إل ى الح اج ة ل احظ .10، relu، 128

```
Epoch 1/10
1875/1875 [==============================] - 2s 928us/step - loss: 0.6331 - accuracy: 0.7769
Epoch 2/10
1875/1875 [==============================] - 2s 961us/step - loss: 0.3860 - accuracy: 0.8615
Epoch 3/10
1875/1875 [==============================] - 2s 930us/step - loss: 0.3395 - accuracy: 0.8755
Epoch 4/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3071 - accuracy: 0.8890
Epoch 5/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2964 - accuracy: 0.8927
Epoch 6/10
1875/1875 [==============================] - 2s 985us/step - loss: 0.2764 - accuracy: 0.8955
Epoch 7/10
1875/1875 [==============================] - 2s 961us/step - loss: 0.2653 - accuracy: 0.8996
Epoch 8/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2549 - accuracy: 0.9052
Epoch 9/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2416 - accuracy: 0.9090
Epoch 10/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2372 - accuracy: 0.9086
313/313 - 0s - loss: 0.3422 - accuracy: 0.8798
```

ال اخ تب ار: دق ة 0.879800021648407

. .

```shell
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(28, activation='relu'),    # 128 -> 28
    tf.keras.layers.Dense(10)
])
```

قم لديعتب ةلمعلم الىلوألا الـ Dense.

```
Epoch 1/10
1875/1875 [==============================] - 2s 714us/step - loss: 6.9774 - accuracy: 0.3294
Epoch 2/10
1875/1875 [==============================] - 1s 715us/step - loss: 1.3038 - accuracy: 0.4831
Epoch 3/10
1875/1875 [==============================] - 1s 747us/step - loss: 1.0160 - accuracy: 0.6197
Epoch 4/10
1875/1875 [==============================] - 1s 800us/step - loss: 0.7963 - accuracy: 0.6939
Epoch 5/10
1875/1875 [==============================] - 2s 893us/step - loss: 0.7006 - accuracy: 0.7183
Epoch 6/10
1875/1875 [==============================] - 1s 747us/step - loss: 0.6675 - accuracy: 0.7299
Epoch 7/10
1875/1875 [==============================] - 1s 694us/step - loss: 0.6681 - accuracy: 0.7330
Epoch 8/10
1875/1875 [==============================] - 1s 702us/step - loss: 0.6675 - accuracy: 0.7356
Epoch 9/10
1875/1875 [==============================] - 1s 778us/step - loss: 0.6508 - accuracy: 0.7363
Epoch 10/10
1875/1875 [==============================] - 1s 732us/step - loss: 0.6532 - accuracy: 0.7350
313/313 - 0s - loss: 0.6816 - accuracy: 0.7230
```

ةقد :ارباتخالا 0.7229999899864197

`Test accuracy` . `Epoch` `fit`. 128`` `accuracy

```python
print(train_labels)
[9 0 0 ... 3 0 5]
print(len(train_labels))
60000

print(train_labels)
[9 0 0 ... 3 0 5]
print(len(train_labels))
60000
```

هذا يعني استخدام الأرقام من 0 إلى 9 ، وهذه هي التسميات. الفئات ، وبالصدفة ، فإن class_names يحتوي أيضًا على 10 عناصر.

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

دعونا نجري بعض التعديلات مرة أخرى.

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(28, activation='relu'),
    tf.keras.layers.Dense(5)   # 10 -> 5
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=10)
```

حدث خطأ.

```
tensorflow.python.framework.errors_impl.InvalidArgumentError:                9                0], 5).    :
         [[node sparse_categorical_crossentropy/SparseSoftmaxCrossEntropyWithLogits/SparseSoftmaxCrossE
```

الدوال: استدعاء مكدس [redacted]

`Sequential` `Dense` 15`` . . `Epoch` .

```python
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(28, activation='relu'),
    tf.keras.layers.Dense(15)
])
```

العربية: إلى الكود ترجمة

```
    = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(28, activation='relu'),
    tf.keras.layers.Dense(15)
])
```

في هذا الكود، تم إنشاء نموذج تسلسلي باستخدام كتبة [illegible]. نموذج يتكون من ثلاث طبقات:

1. [illegible]: تقوم هذه الطبقة بتحويل البيانات ثنائية الأبعاد [28×28] إلى بيانات أحادية البعد [28×28] = 784 عنصرًا.
2. [illegible]: طبقة عصبية كاملة تحتوي على 28 وحدة باستخدام دالة تنشيط [illegible].
3. [illegible]: طبقة عصبية كاملة تحتوي على 15 وحدة بدون استخدام دالة تنشيط.

هذا النموذج بسيط ويستخدم عادةً في مهام تصنيف أو التنبؤ.

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

تم ترجمة الكود أعلاه إلى:

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

□□□□ الم□حسِّن تحديد تم تدريب□. □□□□□□□ النموذج لتهيئة compile الدالة استخدام تم الكود، هذا في
□□□□□□□□ على هنه أن adam، ودالة الخسارة □□□□□□ على انها SparseCategoricalCrossentropy مع المعلمة
□□□□□□□□ الدقة هو الأداء لتقييم المستخدم □□□□□□□□ والمقياس، from_logits=True□accuracy□.

```
model.fit(train_images, train_labels, epochs=15)  # 10 -> 15

test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)

print('\n   :    ', test_acc)
```

```
Epoch 1/15
1875/1875 [==============================] - 2s 892us/step - loss: 6.5778 - accuracy: 0.3771
Epoch 2/15
1875/1875 [==============================] - 2s 872us/step - loss: 1.3121 - accuracy: 0.4910
Epoch 3/15
1875/1875 [==============================] - 2s 909us/step - loss: 1.0900 - accuracy: 0.5389
Epoch 4/15
1875/1875 [==============================] - 1s 730us/step - loss: 1.0422 - accuracy: 0.5577
Epoch 5/15
1875/1875 [==============================] - 1s 709us/step - loss: 0.9529 - accuracy: 0.5952
Epoch 6/15
1875/1875 [==============================] - 1s 714us/step - loss: 0.9888 - accuracy: 0.5950
Epoch 7/15
1875/1875 [==============================] - 1s 767us/step - loss: 0.8678 - accuracy: 0.6355
Epoch 8/15
1875/1875 [==============================] - 1s 715us/step - loss: 0.8247 - accuracy: 0.6611
Epoch 9/15
1875/1875 [==============================] - 1s 721us/step - loss: 0.8011 - accuracy: 0.6626
Epoch 10/15
1875/1875 [==============================] - 1s 711us/step - loss: 0.8024 - accuracy: 0.6622
Epoch 11/15
1875/1875 [==============================] - 1s 781us/step - loss: 0.7777 - accuracy: 0.6696
Epoch 12/15
1875/1875 [==============================] - 1s 724us/step - loss: 0.7764 - accuracy: 0.6728
Epoch 13/15
1875/1875 [==============================] - 1s 731us/step - loss: 0.7688 - accuracy: 0.6767
Epoch 14/15
```

```
1875/1875 [==============================] - 1s 715us/step - loss: 0.7592 - accuracy: 0.6793
Epoch 15/15
1875/1875 [==============================] - 1s 786us/step - loss: 0.7526 - accuracy: 0.6792
313/313 - 0s - loss: 0.8555 - accuracy: 0.6418
```

0.6417999863624573 ال اخ ت ب ار: د ق ة

.15 . `tf.keras.layers.Dense(88, activation='relu')` . 128 .88 `T

```python
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
```

ت ر ج م ة:

```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
```

ال أ س اس ي ال ن م وذ ج م ن ي ت ك ون ال ذ ي tf.keras.Sequential ب اس ت خ دام اح ت م ال ي ن م وذ ج إ ن ش اء ي ت م ال ك ود، ه ذ ا ف ي اح ت م ال ات. إ ل ى ال م خ ر ج ات ل ت ح وي ل ت ُس ت خ دم ال ت ي Softmax و ط ب ق ة model

ال م ق ن ؤ ل ن ب ال ت ن ب ؤ. ب ال آ ن ظ ا ح ل ا ح ظ ة أ ن Sequential ه و ه ذ ا ن ف س ه ك ي ف. ى ل ع أ ل ا ظ ا ح ل ا ت ا م ل م ع ال model و tf.keras.layers.Softmax().

```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
```

ال أ س اس ي ال ن م وذ ج م ن ي ت ك ون وال ذ ي tf.keras.Sequential ب اس ت خ دام اح ت م ال ي ن م وذ ج إ ن ش اء أ ع ال ه ال ك ود ت م ث ل model[] ا ً غ و ب ت م ة ق ب ط ب Softmax. ث م ت م ي خ دام اس ت ه ذ ا ذ ج ال ن م وذ ج ال ن ب ؤ ب ال ن ت ائ ج ل ل predictions[] ب ن ا ء ى ل ع ان ات ب ي ان ال ا خ ت ب ار test_images[].

```
def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

plt.imshow(img, cmap=plt.cm.binary)
```

```
predicted_label = np.argmax(predictions_array)
if predicted_label == true_label:
    color = 'blue'
else:
    color = 'red'

plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                     100*np.max(predictions_array),
                                     class_names[true_label]),
                                     color=color)
```

إلى: أعلاه الكود ترجمة تمت

```
plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                     100*np.max(predictions_array),
                                     class_names[true_label]),
                                     color=color)
```

في هذا الكود، يتم تعيين التسمية المحورية الصحيحة باستخدام plt.xlabel. النص المعروض يتضمن اسم الفئة المتوقعة predicted_label، ونسبة الثقة في التوقع predictions_array، واسم الفئة الحقيقية true_label. اللون color يتم تعيينه أيضًا بناءً على ما إذا كان التوقع صحيحًا أم لا.

```
def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)
```

العربية: إلى الكود ترجمة

```
def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)  #
  plt.xticks(range(10))  #                    0    9
```

```
  plt.yticks([])  #
  thisplot = plt.bar(range(10), predictions_array, color="#777777")  #
  plt.ylim([0, 1])  #                    0    1
  predicted_label = np.argmax(predictions_array)  #
```

ملاحظة: تمت ترجمة الشروحات الموجودة في الكود إلى العربية مع الحفاظ على الكود الأصلي كما هو.

```
thisplot[predicted_label].set_color('red')
thisplot[true_label].set_color('blue')
```

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```

عرض plot_image أن لاحظ .Ankle boot عن عبارة الصورة هذه تكون أن []99 بنسبة احتمال هناك أن يعني هذا
plot_value_array يخرج الرسم البياني على اليمين. على الصورة، اليسار المنبأ بها

```
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```

لنا. بالنسبة لكبير حد إلى واضحة الاستخدام عملية فإن لذا، الاختبار. نتائج من المزيد يعرض فقط هذا أن لاحظ
الخلفية، في استخدامها. كيفية نعرف لكننا الخلفية. في الحساب يتم كيف الآن حتى نعرف لا نحن لذلك،
والتكامل؟ التفاضل حساب نفهم كيف والتكامل. التفاضل حساب هناك

أصغر الرقم كان إذا سأخبرك رقمًا، فيها تخمن مرة كل في تخمينه. عليك يجب و100 1 بين رقمًا هناك أن لنفترض أقول .55 تخمن أكبر. إنه لك أقول .65 تخمن أكبر. إنه لك أقول .80 تخمن أصغر. إنه لك أقول .50 تخمن أكبر. أو صحيح. بشكل خمنت لقد حسنًا، لك، أقول .58 تخمن أصغر. إنه لك

الأرقام من العديد هناك يكون قد تعقيدًا. أكثر لكنها الخلفية، في مشابهة عملية محاكاة على يعمل الآلي التعلم العمليات من العديد إجراء يتطلب تخمين كل الوقت، نفس في الأرقام. من العديد تخمين ويجب ،100 1 من كبيرًا. حسابًا ذلك يتطلب أصغر، أو أكبر الرقم كان إذا ما تحديد فيها يتم مرة كل ذلك، إلى بالإضافة الحسابية.

[illegible]

[illegible] هي [illegible] الرمجة لغة على تعتمد المصدر مفتوحة آلي تعلم مكتبة أساسي بشكل وتُستخدم، [illegible] الذكاء أبحاث فريق قبل من تطويرها تم الطبيعية. اللغة ومعالجة الحاسوبية الرؤية مثل تطبيقات في الباحثين بين شعبية العميق التعلم أدوات أكثر من واحدة وتُعتبر، [illegible] [illegible] الآن [illegible] في الاصطناعي والمطورين.

مميزات [illegible]:

1. **عالية مرونة**: حمست [illegible] إنشاء بديعة ونماذج الديناميكية بسهولة، مما هو للبحث مثالية جيجي والتجريب.
2. **حسابات تلقائية [illegible]**: دعمت [illegible] حاسب التدرجات التلقائيًا، مما يسهل عملية بدرت النماذج.
3. **توافق مع [illegible]**: تتمتت [illegible] مكتبات مع سلس بالمكتبة [illegible] الأخرى مثل [illegible] و[illegible].
4. **مجتمع نشط**: يوجد مجتمع كبير وداعم حول [illegible]، مما يجعل الحصول على الدعم والموارد التعليمية أسهل.

باستخدام [illegible] بسيط مثال:

```
import torch

# إنشاء موتر (Tensor)
x = torch.tensor([1.0, 2.0, 3.0], requires_grad=True)

# إجراء عملية
y = x * 2

# حساب
```

```
y.backward(torch.tensor([1.0, 1.0, 1.0]))

print(x.grad)  # :     tensor([2., 2., 2.])
```

ي ف ذه ،ال ث م ال م وق ن اء ش إ ب ر وت م □□□□□□□□ اء ر إ ج وإ ة ي ل م ع ة ي ا ب ا س ح ة ط ي س ب ، ه ي ل ع م ث ب س ح ن ج ر د ت ال
م ا د خ ت س ا ب ة ي ا ص خ backward.

□□□□□□□ ر ب ت ع ت ُ ا د ا ة ُ ق و ي ة ق و ي ة ا ل ع ف و أ ي ل ص خ ش ل م ع ي ف ي ال ج م ل ع ت ال ، ق ي م ع ال ء اء س ا ن ا ك ا ح ب ا أو ا ً ر و ط م.

م ق .ه ت ي ب ث ت ب ا ذه م ع د ي ر ا د ص إ 3.9 ن م □□□□□□.

```
$ pip install torch torchvision
      torch
      torch-1.8.0-cp39-none-macosx_10_9_x86_64.whl (120.6 (
     |                 | 120.6      224    /
      torchvision
      torchvision-0.9.0-cp39-cp39-macosx_10_9_x86_64.whl (13.1 (
     |                 | 13.1      549    /
          numpy       /usr/local/lib/python3.9/site-packages  ) torch) (1.20.1)
      typing-extensions
      typing_extensions-3.7.4.3-py3-none-any.whl (22 (
          pillow>=4.1.1       /usr/local/lib/python3.9/site-packages  ) torchvision) (8.0.1)
        :     typing-extensions, torch, torchvision
      torch-1.8.0   torchvision-0.9.0   typing-extensions-3.7.4.3
```

ذلك .ل ن ج ر ب

```
import torch
x = torch.rand(5, 3)
print(x)
```

خ ط أ .ح د ث

```
Traceback (most recent call last):
  File "torch.py", line 1, in <module>
    import torch
  File "torch.py", line 2, in <module>
    x = torch.rand(5, 3)
AttributeError:     'torch'                      'rand'   )            (
```

تثحب عن ةالسر أطخلا هذه ىلع [illegible]. حضتا أن ةلكشملا تانك ببسب أن انفلم ناك ىمسيُ اضًأي torch.
ريغيت مسالا لح ةلكشملا حبصأو لك ءيش لمعي لكشب .حيحص

```
tensor([[0.5520, 0.9446, 0.5543],
        [0.6192, 0.0908, 0.8726],
        [0.0223, 0.7685, 0.9814],
        [0.4019, 0.5406, 0.3861],
        [0.5485, 0.6040, 0.2387]])
```

ورثعلا ىلع .لاثم

```
# -*- coding: utf-8 -*-

import torch
import math
dtype = torch.float
device = torch.device("cpu")
# device = torch.device("cuda:0") #                    GPU
```

ءاشنإ تانايب ةيئاوشع لاخدإلل جارخإلاو

[illegible] 2000, [illegible]

ةئيهت نازوألا لكشب يئاوشع

[illegible]

```
learning_rate = 1e-6
for t in range(2000):
    #      :                  y
    y_pred = a + b * x + c * x ** 2 + d * x ** 3
```

```
#
loss = (y_pred - y).pow(2).sum().item()
if t % 100 == 99:
    print(t, loss)

#                   a, b, c, d
grad_y_pred = 2.0 * (y_pred - y)
grad_a = grad_y_pred.sum()
grad_b = (grad_y_pred * x).sum()
grad_c = (grad_y_pred * x ** 2).sum()
grad_d = (grad_y_pred * x ** 3).sum()

    #
    a -= learning_rate * grad_a
    b -= learning_rate * grad_b
    c -= learning_rate * grad_c
    d -= learning_rate * grad_d
print(f':     y = {a.item()} + {b.item()} x + {c.item()} x^2 + {d.item()} x^3')
```

بتشغيله. قم

```
99 1273.537353515625
199 849.24853515625
299 567.4786987304688
399 380.30291748046875
499 255.92752075195312
599 173.2559814453125
699 118.2861328125
799 81.72274780273438
899 57.39331817626953
999 41.198158264160156
1099 30.41307830810547
1199 23.227672576904297
1299 18.438262939453125
1399 15.244369506835938
1499 13.113286972045898
1599 11.690631866455078
```

```
1699 10.740333557128906
1799 10.105220794677734
1899 9.6804780960083
1999 9.39621353149414
:     y = -0.011828352697193623 + 0.8360244631767273 x + 0.002040589228272438 x^2 + -0.0903836563229560
```

قِللن لةرظن ىلع دوكلا امدختسا بةبتكم numpy .طقف

```
# -*- coding: utf-8 -*-
import numpy as np
import math
```

ءاشنإ اتانايب لادخإ جارخإو ةيئاوشع

□ □ □□.□□□□□□□□□□□□□□□□.□□, □□□□.□□, 2000□ □ □ □□.□□□□□□□

ةئيهت نازوألا لكشب يئاوشع

□ □ □□.□□□□□□□.□□□□□□□□ □ □ □□.□□□□□□□.□□□□□□□□ □ □ □□.□□□□□□□.□□□□□□□□ □ □ □□.□□□□□□□.□□□□□□□□

```
learning_rate = 1e-6
for t in range(2000):
    #      :                    y
    # y = a + b x + c x^2 + d x^3
    y_pred = a + b * x + c * x ** 2 + d * x ** 3

#
loss = np.square(y_pred - y).sum()
if t % 100 == 99:
    print(t, loss)

#                   a, b, c, d
grad_y_pred = 2.0 * (y_pred - y)
grad_a = grad_y_pred.sum()
grad_b = (grad_y_pred * x).sum()
```

```
grad_c = (grad_y_pred * x ** 2).sum()
grad_d = (grad_y_pred * x ** 3).sum()

#
a -= learning_rate * grad_a
b -= learning_rate * grad_b
c -= learning_rate * grad_c
d -= learning_rate * grad_d

print(f':     y = {a} + {b} x + {c} x^2 + {d} x^3')
```

للحساب. مختلفتان طريقتان هذه أن لاحظ

الدرجة من معادلة هي بينهما العلاقة أن افتراض تم ثم .y و x قيم من مجموعة إنشاء أولاً تم المثاليين، هذين في الخوارزميات؟ هذه تعمل كيف تكراري. بشكل المعاملات لحساب الطرق بعض استخدام تم ذلك، بعد الثالثة. الدقيقة التفاصيل سنترك أكبر. بشكل الدقة تحسين يتم مرة كل وفي مرة، 2000 العملية تكرار تم أنه لاحظ الحالي. الوقت في جانبًا العملية لهذه

الخاتمة

يمكننا الحالي. الوقت في مهمًا ليس هذا ذلك، ومع الخلفية. في الآلي التعلم حساب كيفية نفهم لا نحن حاليًا، استخدام أيضًا يمكننا أعلاه. ذكرناه لما المشابهة المعرفة باستخدام الأشياء من بالكثير القيام بالفعل المبادئ تعلم يمكننا الأمثلة، عشرات نختبر عندما ذلك. إلى وما والصوت النصوص لمعالجة الآلي التعلم لاحقًا.

التمرين

- أعلاه. موضح هو كما بالاستكشاف الطلاب يقوم