机器学习实验一

基本图像分类

**实验步骤**

1. **环境配置**

**使用 PyCharm 集成环境可以不需要手动安装 Tensorflow 相关依赖**

1. # TensorFlow and tf.keras

2. import tensorflow as tf

3.

4. # Helper libraries

5. import numpy as np

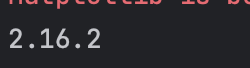
6. import matplotlib.pyplot as plt

7.

8. print(tf.\_\_version\_\_)

9.

**结果：**

****

**代表环境已经配置成功！**

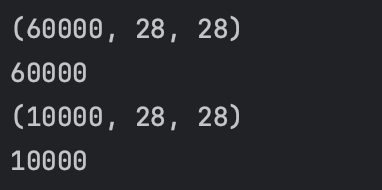
1. **导入数据集**

# TensorFlow and tf.keras  
import tensorflow as tf  
  
# Helper libraries  
import numpy as np  
import matplotlib.pyplot as plt  
  
# print(tf.\_\_version\_\_)  
class\_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',  
 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']  
fashion\_mnist = tf.keras.datasets.fashion\_mnist  
  
(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()

1. **浏览数据**

1. # 浏览数据集  
print(train\_images.shape)  
# 浏览训练集  
print(len(train\_labels))  
# 浏览测试集  
print(test\_images.shape)  
# 测试集标签  
print(len(test\_labels))

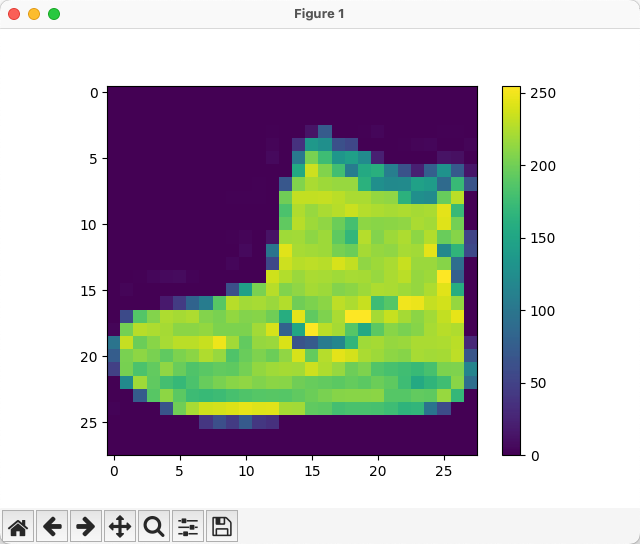
**结果**

****

1. **预处理数据**

1. plt.figure()  
plt.imshow(train\_images[0])  
plt.colorbar()  
plt.grid(False)  
plt.show()

**结果**

****

将这些值缩小至 0 到 1 之间，然后将其馈送到神经网络模型。为此，将这些值除以 255

1. train\_images = train\_images / 255.0
2. test\_images = test\_images / 255.0

为了验证数据的格式是否正确，将训练集中的前25个数据输出

1. plt.figure(figsize=(10,10))

2. for i in range(25):

3. plt.subplot(5,5,i+1)

4. plt.xticks([])

5. plt.yticks([])

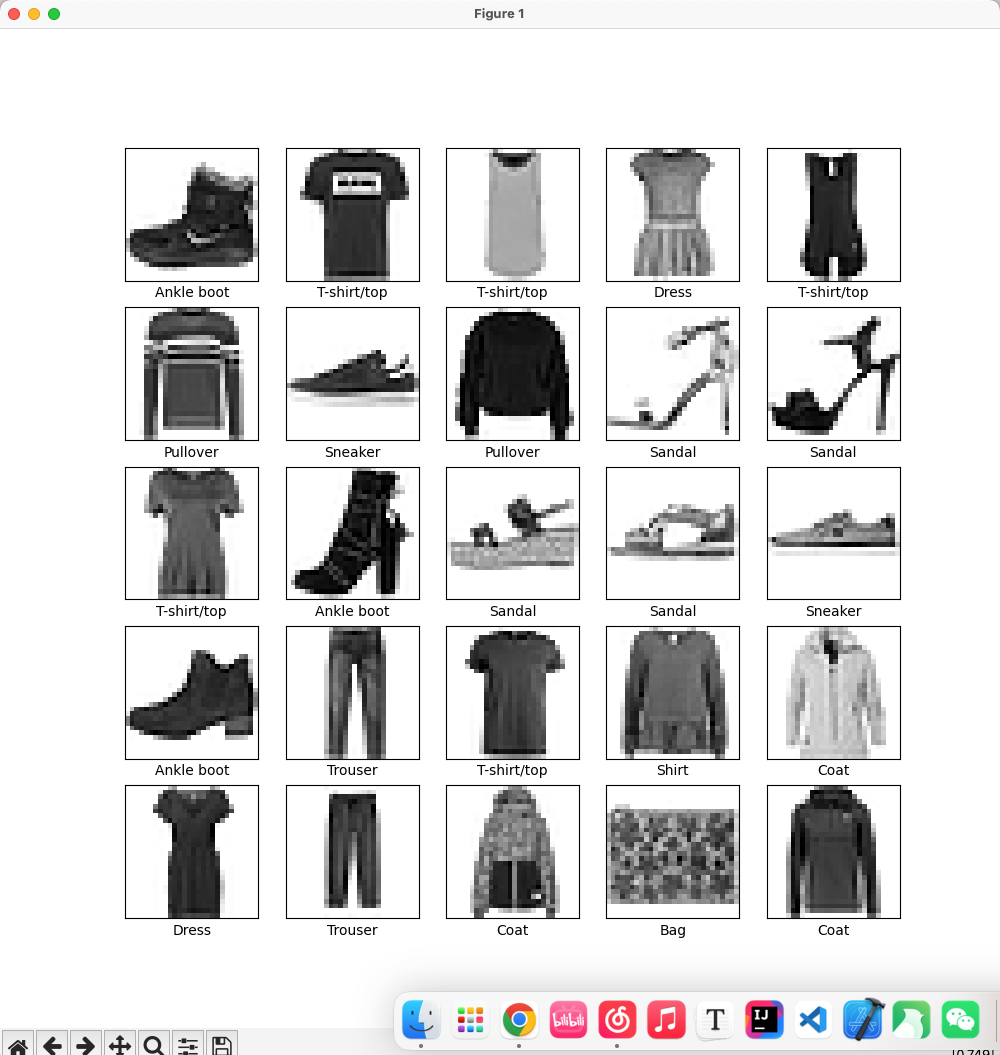
6. plt.grid(False)

7. plt.imshow(train\_images[i], cmap=plt.cm.binary)

8. plt.xlabel(class\_names[train\_labels[i]])

9. plt.show()

**结果**

****

**输出图片表示预处理成功！**

1. **构建模型**
   1. **设置层**

神经网络的基本组成部分是层。层会从向其馈送的数据中提取表示形式。希望这些表示形式有助于解决手头上的问题。

大多数深度学习都包括将简单的层链接在一起。大多数层（如 [tf.keras.layers.Dense](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense?hl=zh-cn)）都具有在训练期间才会学习的参数。

1. model = tf.keras.Sequential([

2. tf.keras.layers.Flatten(input\_shape=(28, 28)),

3. tf.keras.layers.Dense(128, activation='relu'),

4. tf.keras.layers.Dense(10)

5. ])

该网络的第一层 [tf.keras.layers.Flatten](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten?hl=zh-cn) 将图像格式从二维数组（28 x 28 像素）转换成一维数组（28 x 28 = 784 像素）。将该层视为图像中未堆叠的像素行并将其排列起来。该层没有要学习的参数，它只会重新格式化数据。

展平像素后，网络会包括两个 [tf.keras.layers.Dense](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense?hl=zh-cn) 层的序列。它们是密集连接或全连接神经层。第一个 Dense 层有 128 个节点（或神经元）。第二个（也是最后一个）层会返回一个长度为 10 的 logits 数组。每个节点都包含一个得分，用来表示当前图像属于 10 个类中的哪一类。

* 1. **编译模型**

1. model.compile(optimizer='adam',

2. loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

3. metrics=['accuracy'])

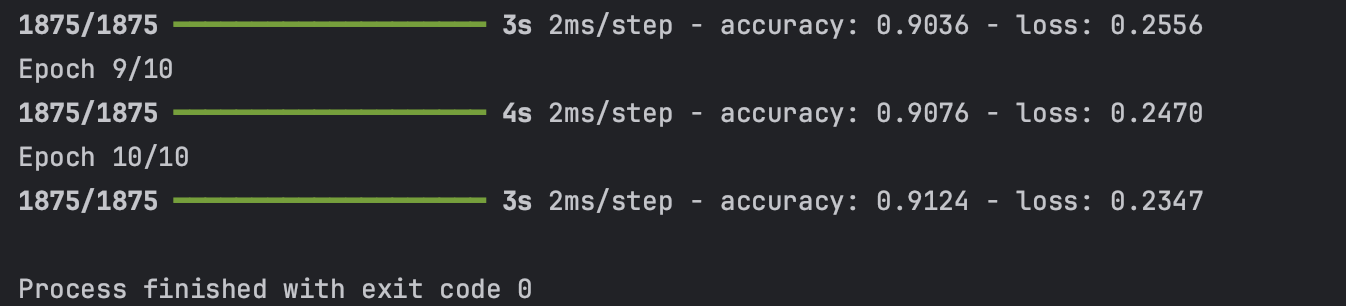
1. **训练模型**

训练神经网络模型需要执行以下步骤：

1. 将训练数据馈送给模型。在本例中，训练数据位于 train\_images 和 train\_labels 数组中。
2. 模型学习将图像和标签关联起来。
3. 要求模型对测试集（在本例中为 test\_images 数组）进行预测。
4. 验证预测是否与 test\_labels 数组中的标签相匹配。
   1. **向模型馈送数据**

1. model.fit(train\_images, train\_labels, epochs=10)

**结果**

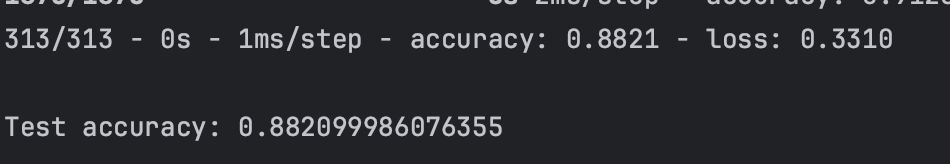
****

* 1. **评估准确率**

1. test\_loss, test\_acc = model.evaluate(test\_images, test\_labels, verbose=2)

2. print('\nTest accuracy:', test\_acc)

**结果**

****

结果表明，模型在测试数据集上的准确率略低于训练数据集。训练准确率和测试准确率之间的差距代表过拟合。过拟合是指机器学习模型在新的、以前未曾见过的输入上的表现不如在训练数据上的表现。过拟合的模型会“记住”训练数据集中的噪声和细节，从而对模型在新数据上的表现产生负面影响。

* 1. **进行预测**

模型经过训练后，您可以使用它对一些图像进行预测。附加一个 Softmax 层，将模型的线性输出 [logits](https://developers.google.com/machine-learning/glossary?hl=zh-cn#logits) 转换成更容易理解的概率。

1. probability\_model = tf.keras.Sequential([model,

2. tf.keras.layers.Softmax()])

3. redictions = probability\_model.predict(test\_images)

1. # 进行预测  
probability\_model = tf.keras.Sequential([model,  
 tf.keras.layers.Softmax()])  
predictions = probability\_model.predict(test\_images)  
print(predictions[0])  
np.argmax(predictions[0])  
print(test\_labels[0])  
  
# 将预测结果绘制成图标  
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)  
  
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)  
  
 thisplot[predicted\_label].set\_color('red')  
 thisplot[true\_label].set\_color('blue')

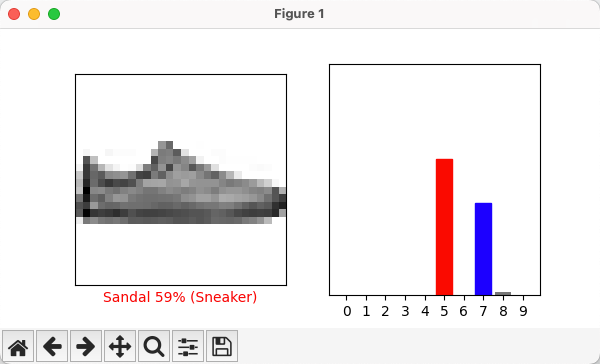
**d） 验证预测结果**

1. 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()

2.i = 12 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()

**结果**

****

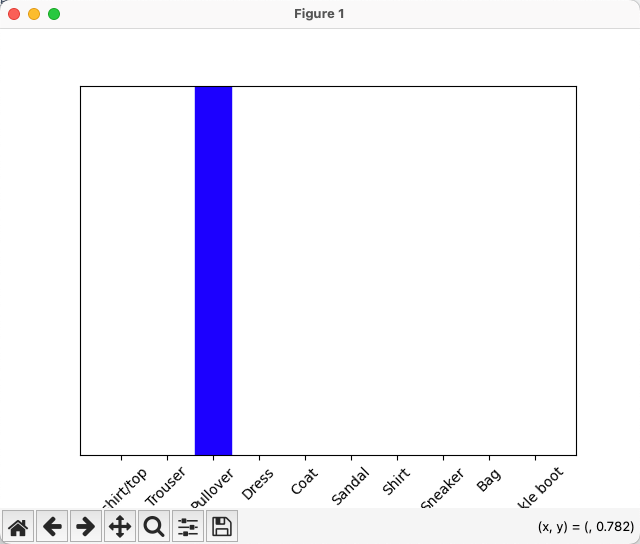
****

1. **使用模型**

最后，使用训练好的模型对单个图像进行预测。

1. # Grab an image from the test dataset.  
img = test\_images[1]  
  
print(img.shape)  
  
# Add the image to a batch where it's the only member.  
img = (np.expand\_dims(img,0))  
  
print(img.shape)  
  
predictions\_single = probability\_model.predict(img)  
  
print(predictions\_single)  
  
plot\_value\_array(1, predictions\_single[0], test\_labels)  
\_ = plt.xticks(range(10), class\_names, rotation=45)  
plt.show()  
print(np.argmax(predictions\_single[0]))

**结果**

****

输出该图片的预测类别。

****

1. **实验总结**

通过本次实验，我们完成了从环境配置到模型训练与评估的全流程操作，掌握了使用 TensorFlow 进行图像分类的基本方法。模型的准确率表明其对测试数据有良好的泛化能力，但也需要进一步优化以降低过拟合影响