Train your first neural network: basic classification

Jisen Fang
CS550 - Machine Learning and Business Intelligence
2023 Spring
Github: https://github.com/JFang2023/JF/blob/main/Machine%20Learning/Neural%20Network/Basic%20Classification/basic_classification.ipynb

Table of contents

Setup

```
[1]  1 # TensorFlow and tf.keras
     2 import tensorflow.compat.v1 as tf
     3
     4 from tensorflow import keras
     5
     6 # Helper libraries
     7 import numpy as np
     8 import matplotlib.pyplot as plt
     9
    10 print(tf.__version__)
```

```
2.11.0
```

Import the Fashion MNIST dataset

```
[2]  1 fashion_mnist = keras.datasets.fashion_mnist
     2
     3 (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

```
[3]  1 class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
     2               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Explore the data

Explore the data

```
[7]  1 train_images.shape #60,000 images with each 28 * 28 pixels
```

0s

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

```
[8]  1 len(train_labels) #6000 labels
```

0s

```
60000
```

```
[9]  1 train_labels #each label is an integer between 0 and 9
```

0s

```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

```
[10]  1 test_images.shape #10,000 images with each 28 * 28 pixels
```

0s

```
(10000, 28, 28)
```

```
[11]  1 len(test_labels)
```

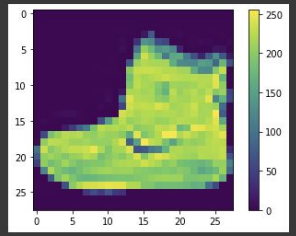
0s

```
10000
```

Preprocess the data

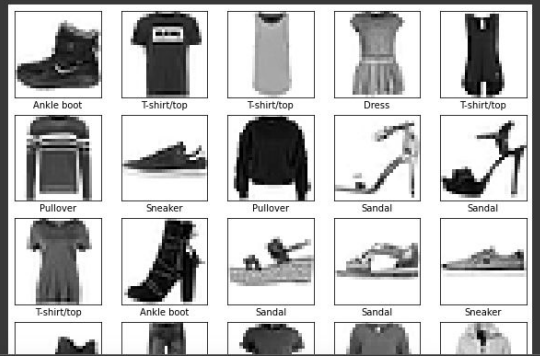
Preprocess the data

```
1 plt.figure()
2 plt.imshow(train_images[0])
3 plt.colorbar()
4 plt.grid(False)
5 plt.show()
```



```
[15] 1 train_images = train_images / 255.0 #scale these values to a range of 0 to 1 by divide 255
     2
     3 test_images = test_images / 255.0
```

```
[14] 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()
```



Build the model

Build the model

setup the layers

```
[18]  1 model = keras.Sequential([
      2     keras.layers.Flatten(input_shape=(28, 28)), #28 by 28 pixels
      3     keras.layers.Dense(128, activation=tf.nn.relu), #The first Dense layer has 128 nodes
      4     keras.layers.Dense(10, activation=tf.nn.softmax) #The second one is a 10-node softmax layer and returns an array of 10 probability scores that sum to 1.
      5 ])
```

compile the model

```
[20]  1 model.compile(optimizer='adam', #optimizer – This is how the model is updated based on the data it sees and its loss function.
      2               loss='sparse_categorical_crossentropy', #loss - measures how accurate the model is during training
      3               metrics=['accuracy']) #metrics – monitor the training and testing steps
```

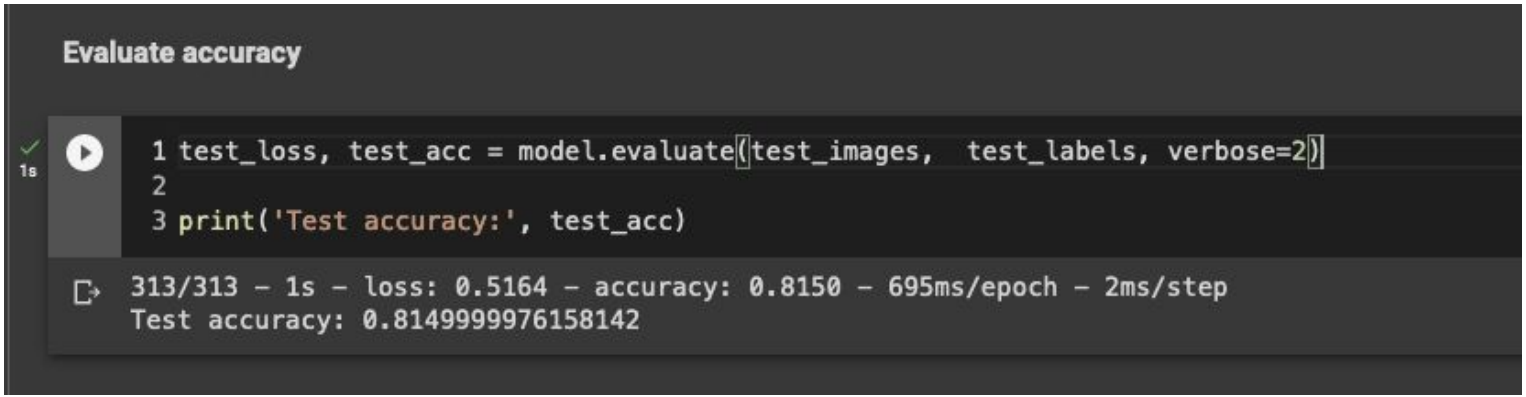
Train the model

Train the model

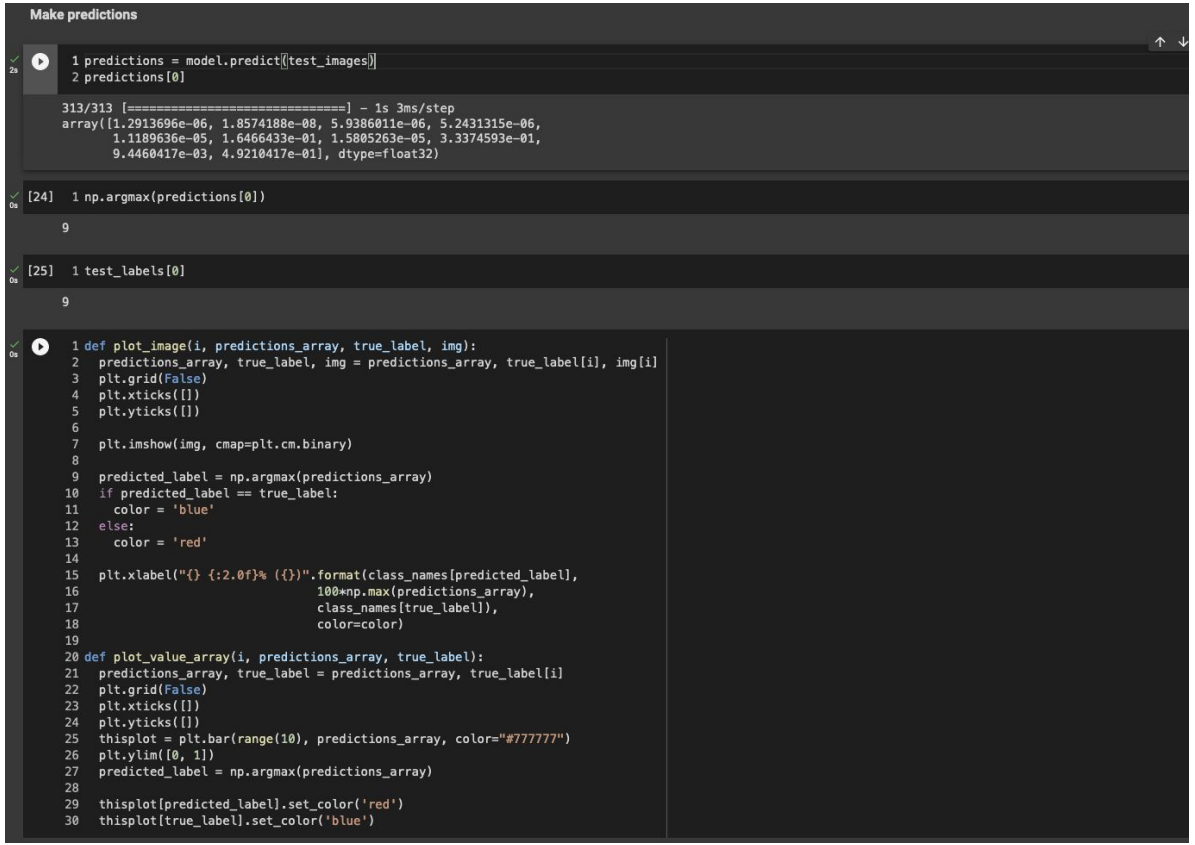
```
1m
1 model.fit(train_images, train_labels, epochs=5) #call the model.fit method–the model is "fit" to the training data

Epoch 1/5
1875/1875 [==============================] - 11s 5ms/step - loss: 1.1030 - accuracy: 0.6523
Epoch 2/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.6502 - accuracy: 0.7658
Epoch 3/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.5742 - accuracy: 0.7934
Epoch 4/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.5308 - accuracy: 0.8116
Epoch 5/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.5017 - accuracy: 0.8230
<keras.callbacks.History at 0x7f76623be130>
```

Evaluate accuracy

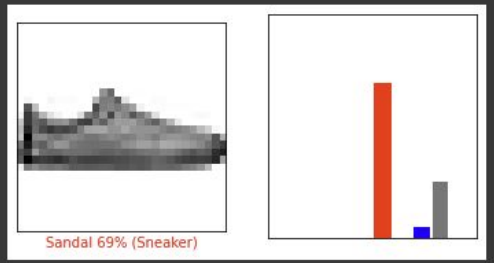


Make predictions

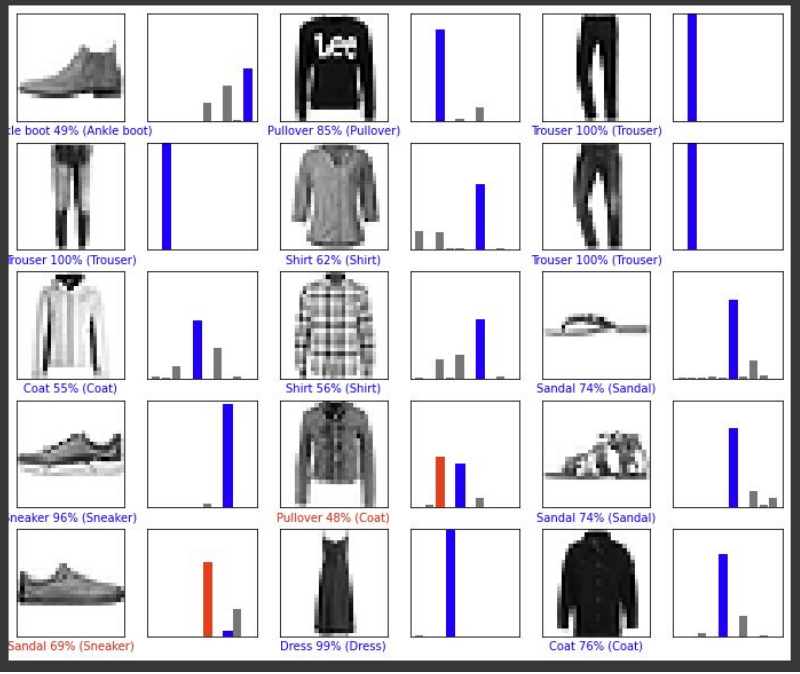


[27]
```
1 i = 0
2 plt.figure(figsize=(6,3))
3 plt.subplot(1,2,1)
4 plot_image(i, predictions[i], test_labels, test_images)
5 plt.subplot(1,2,2)
6 plot_value_array(i, predictions[i],  test_labels)
7 plt.show()
```


```
1 i = 12
2 plt.figure(figsize=(6,3))
3 plt.subplot(1,2,1)
4 plot_image(i, predictions[i], test_labels, test_images)
5 plt.subplot(1,2,2)
6 plot_value_array(i, predictions[i],  test_labels)
7 plt.show()
```



```
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.show()
```



```
[30]  1 img = test_images[1]
      2
      3 print(img.shape)
```

```
(28, 28)
```

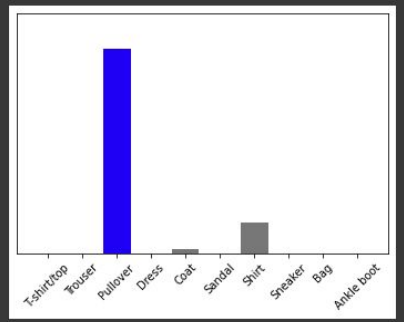
```
[31]  1 img = (np.expand_dims(img,0))
      2
      3 print(img.shape)
```

```
(1, 28, 28)
```

```
[32]  1 predictions_single = model.predict(img)
      2
      3 print(predictions_single)
```

```
1/1 [==============================] - 0s 32ms/step
[[6.3680531e-04 1.7049166e-05 8.4903616e-01 7.0577524e-05 1.9538909e-02
  7.6126838e-09 1.3051157e-01 4.7872128e-13 1.8899163e-04 2.2428961e-09]]
```

```
  1 plot_value_array(1, predictions_single[0], test_labels)
  2 plt.xticks(range(10), class_names, rotation=45)
  3 plt.show()
```




```
[34]  1 prediction_result = np.argmax(predictions_single[0])
      2 print(prediction_result)
```

```
2
```

References

- Basic classification - Colab: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/r1/tutorials/keras/basic_classification.ipynb#scrollTo=S5Uhzt6vVIB2&uniqifier=1