chit-7-fashion

May 6, 2025

[2]:
```
import tensorflow as tf
#from tensorflow.keras.layers.experimental import preprocessing
from sklearn.model_selection import train_test_split
from mlxtend.plotting import plot_confusion_matrix
from sklearn import metrics
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from tqdm.notebook import tqdm
import random
import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
(trainX, trainY), (testX, testY) = tf.keras.datasets.fashion_mnist.load_data()
trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
testX = testX.reshape((testX.shape[0], 28, 28, 1))
trainY_cat = tf.keras.utils.to_categorical(trainY)
testY_cat = tf.keras.utils.to_categorical(testY)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-labels-idx1-ubyte.gz
29515/29515             0s 9us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-images-idx3-ubyte.gz
26421880/26421880             9s
0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-labels-idx1-ubyte.gz
5148/5148             0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-images-idx3-ubyte.gz
4422102/4422102             3s
1us/step
```

[4]:
```
train_norm = trainX.astype('float32')
test_norm = testX.astype('float32')
train_norm = train_norm / 255.0
test_norm = test_norm / 255.0
```

[5]:
```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat','Sandal',␣
↪'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

[6]:
```
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(trainX[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[trainY[i]])
plt.show()
```

[7]:
```
model = tf.keras.models.Sequential([
    # First convolutional layer
    tf.keras.layers.Conv2D(64, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu', name='conv-layer-1'),

    # First pooling layer
    tf.keras.layers.AvgPool2D(pool_size=(2, 2), name='pooling-layer-1'),

    # Second convolutional layer
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same', name='conv-layer-2'),
```

```
    # Second pooling layer
    tf.keras.layers.AvgPool2D(pool_size=(2, 2), name='pooling-layer-2'),

    # Global average pooling
    tf.keras.layers.GlobalAveragePooling2D(name='pooling-layer-3'),

    # Fully connected layer with softmax activation for multi-class␣
 ↪classification
    tf.keras.layers.Dense(len(class_names), activation="softmax",␣
 ↪name="output-layer")
])
```

[8]:
```
model.compile(loss="categorical_crossentropy",
optimizer="adam",
metrics=["accuracy"])
```

[9]:
```
history = model.fit(train_norm, trainY_cat, epochs=10,␣
 ↪validation_data=(test_norm, testY_cat))
```

```
Epoch 1/10
1875/1875             13s 7ms/step -
accuracy: 0.4759 - loss: 1.4343 - val_accuracy: 0.7090 - val_loss: 0.8380
Epoch 2/10
1875/1875             13s 7ms/step -
accuracy: 0.7230 - loss: 0.7909 - val_accuracy: 0.7534 - val_loss: 0.7231
Epoch 3/10
1875/1875             13s 7ms/step -
accuracy: 0.7497 - loss: 0.7128 - val_accuracy: 0.7665 - val_loss: 0.6767
Epoch 4/10
1875/1875             14s 8ms/step -
accuracy: 0.7677 - loss: 0.6566 - val_accuracy: 0.7369 - val_loss: 0.7168
Epoch 5/10
1875/1875             15s 8ms/step -
accuracy: 0.7777 - loss: 0.6333 - val_accuracy: 0.7628 - val_loss: 0.6469
Epoch 6/10
1875/1875             16s 9ms/step -
accuracy: 0.7849 - loss: 0.6138 - val_accuracy: 0.7846 - val_loss: 0.6163
Epoch 7/10
1875/1875             17s 9ms/step -
accuracy: 0.7922 - loss: 0.5906 - val_accuracy: 0.7991 - val_loss: 0.5942
Epoch 8/10
1875/1875             16s 9ms/step -
accuracy: 0.7973 - loss: 0.5740 - val_accuracy: 0.7923 - val_loss: 0.5979
Epoch 9/10
1875/1875             16s 9ms/step -
accuracy: 0.8038 - loss: 0.5531 - val_accuracy: 0.8059 - val_loss: 0.5672
Epoch 10/10
```

```
1875/1875              16s 9ms/step -
accuracy: 0.8126 - loss: 0.5369 - val_accuracy: 0.8096 - val_loss: 0.5543
```

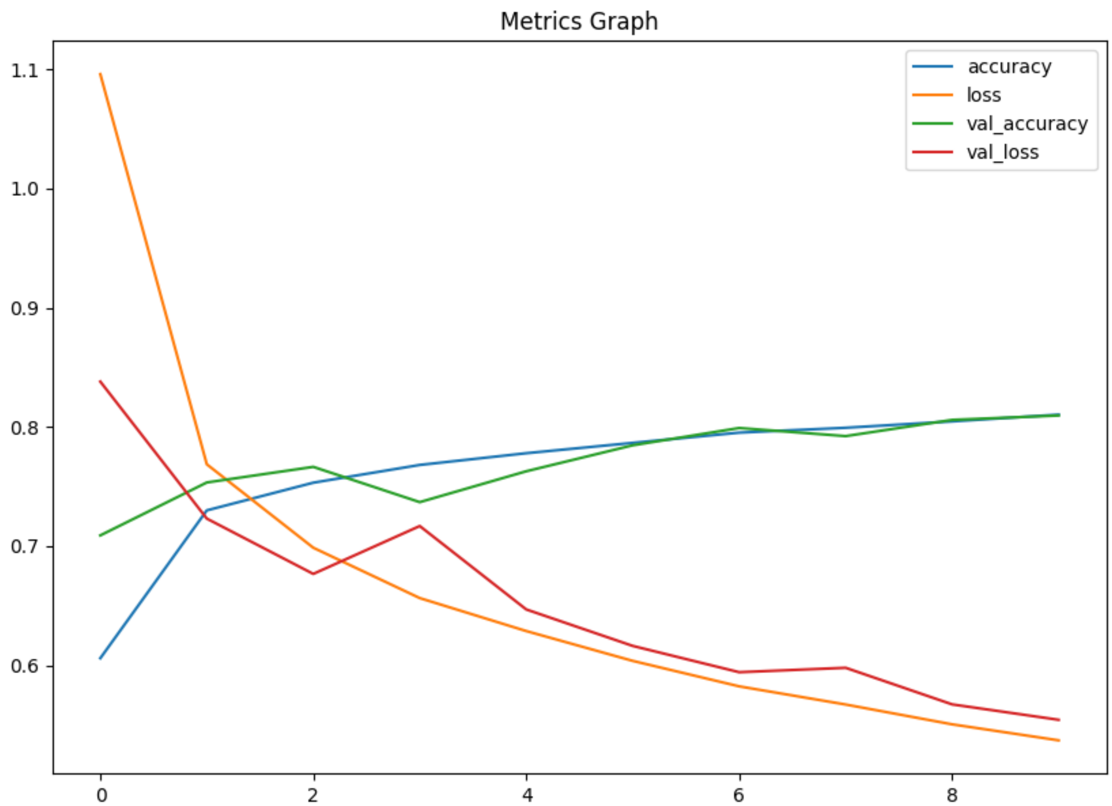
[10]:
```
tf.keras.utils.plot_model(model, show_shapes=True)
```

```
You must install graphviz (see instructions at
https://graphviz.gitlab.io/download/) for `plot_model` to work.
```

[11]:
```
model.summary()
```

```
Model: "sequential"


 Layer (type)                         Output Shape                    ␣
↪Param #

 conv-layer-1 (Conv2D)                (None, 26, 26, 64)                  ␣
↪640

 pooling-layer-1 (AveragePooling2D)   (None, 13, 13, 64)                  ␣
↪  0

 conv-layer-2 (Conv2D)                (None, 13, 13, 32)               ␣
↪18,464

 pooling-layer-2 (AveragePooling2D)   (None, 6, 6, 32)                    ␣
↪  0

 pooling-layer-3                      (None, 32)                          ␣
↪  0
 (GlobalAveragePooling2D)                                                 ␣
↪

 output-layer (Dense)                 (None, 10)                          ␣
↪330



 Total params: 58,304 (227.75 KB)


 Trainable params: 19,434 (75.91 KB)


 Non-trainable params: 0 (0.00 B)


 Optimizer params: 38,870 (151.84 KB)
```

[12]:
```
pd.DataFrame(history.history).plot(figsize=(10,7))
plt.title("Metrics Graph")
plt.show()
```



[13]:
```
model.evaluate(testX, testY_cat)
```

```
313/313              1s 3ms/step -
accuracy: 0.6350 - loss: 115.7895
```

[13]: [119.34942626953125, 0.6317999958992004]

[14]:
```
predictions = model.predict(testX)
```

```
313/313              1s 3ms/step
```

[15]:
```
predictions = tf.argmax(predictions, axis=1)
```

[16]:
```
y_test = tf.argmax(testY_cat, axis=1)
```

[17]:
```
y_test = tf.Variable(y_test)
```

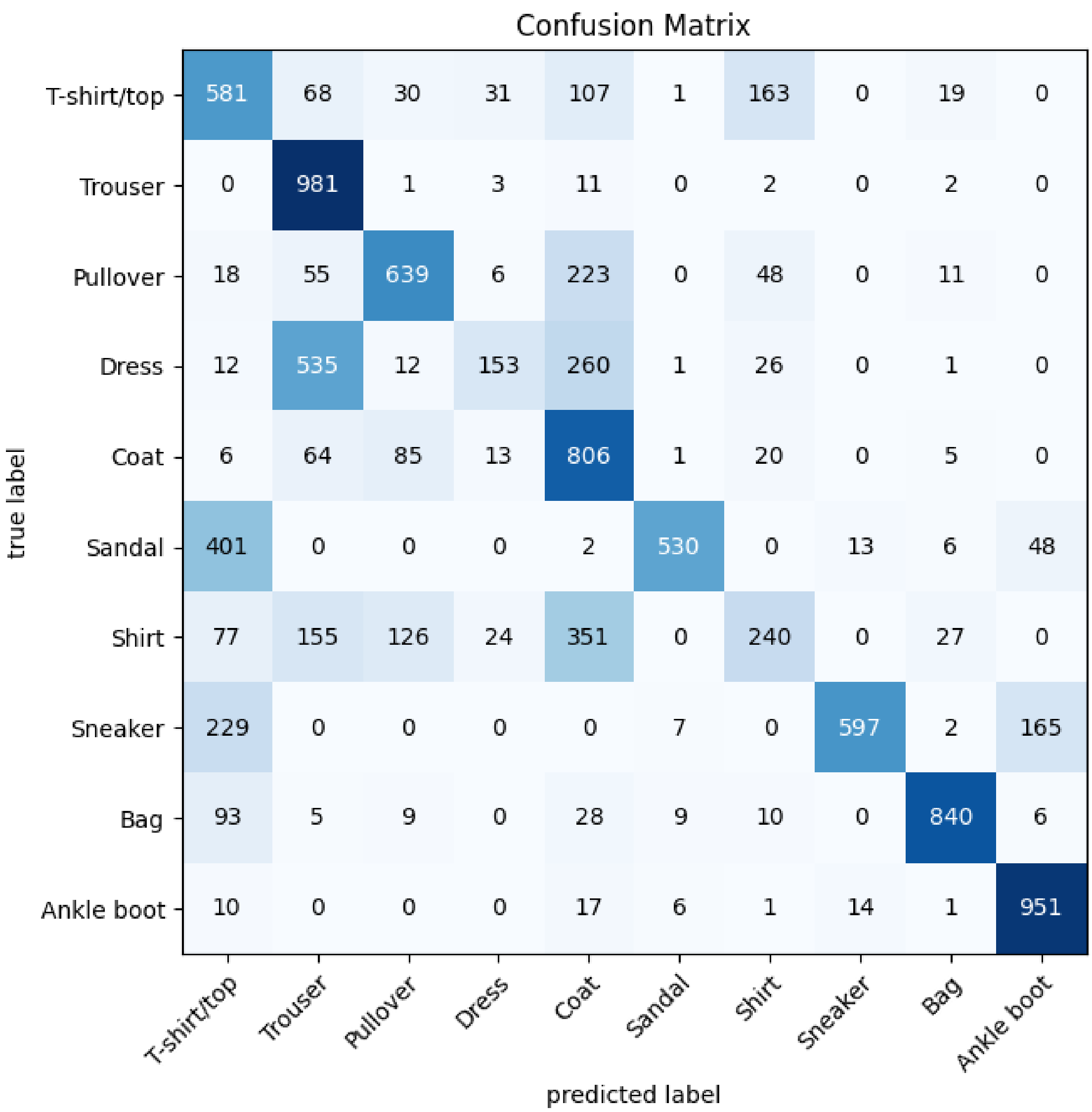
[18]:
```
print(metrics.accuracy_score(y_test, predictions))
```

```
0.6318
```

[19]:
```
print(metrics.classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.41      0.58      0.48      1000
           1       0.53      0.98      0.69      1000
           2       0.71      0.64      0.67      1000
           3       0.67      0.15      0.25      1000
           4       0.45      0.81      0.57      1000
           5       0.95      0.53      0.68      1000
           6       0.47      0.24      0.32      1000
           7       0.96      0.60      0.74      1000
           8       0.92      0.84      0.88      1000
           9       0.81      0.95      0.88      1000

    accuracy                           0.63     10000
   macro avg       0.69      0.63      0.61     10000
weighted avg       0.69      0.63      0.61     10000
```

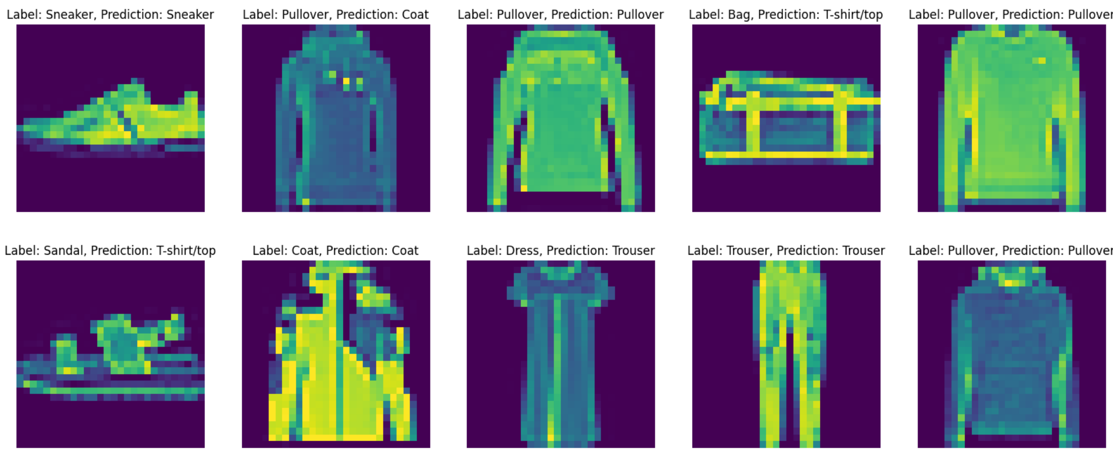
[20]:
```
cm = metrics.confusion_matrix(y_test, predictions)
plot_confusion_matrix(cm, figsize=(10,7), class_names=class_names)
plt.title("Confusion Matrix")
plt.show()
```



[21]:
```
images = []
labels = []
random_indices = random.sample(range(len(testX)), 10)
for idx in random_indices:
    images.append(testX[idx])
    labels.append(testY_cat[idx])
images = np.array(images)
labels = np.array(labels)
fig = plt.figure(figsize=(20, 8))
rows = 2
cols = 5
x = 1
```

```python
for image, label in zip(images, labels):
    fig.add_subplot(rows, cols, x)
    prediction = model.predict(tf.expand_dims(image, axis=0))
    prediction = class_names[tf.argmax(prediction.flatten())]
    label = class_names[tf.argmax(label)]
    plt.title(f"Label: {label}, Prediction: {prediction}")
    plt.imshow(image/255.)
    plt.axis("off")
    x += 1
```

```
1/1           0s 30ms/step
1/1           0s 26ms/step
1/1           0s 23ms/step
1/1           0s 23ms/step
1/1           0s 20ms/step
1/1           0s 19ms/step
1/1           0s 20ms/step
1/1           0s 19ms/step
1/1           0s 18ms/step
1/1           0s 19ms/step
```



[ ]: