DL Lab Exp No.3

[1]:
```
import tensorflow as tf
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")
```

[2]:
```
(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)
```

[3]:
```
train_norm = trainX.astype('float32')
test_norm = testX.astype('float32')

train_norm = train_norm / 255.0
test_norm = test_norm / 255.0
```

[4]:
```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

[5]:
```
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()
```

[6]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, kernel_size=(3, 3), input_shape=(28, 28, 1),␣
 ↪activation='relu', padding='same', name='conv-layer-1'),
    tf.keras.layers.AvgPool2D(pool_size=(2, 2), name='pooling-layer-1'),
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same', name='conv-layer-2'),
    tf.keras.layers.AvgPool2D(pool_size=(2, 2), name='pooling-layer-2'),
    tf.keras.layers.GlobalAveragePooling2D(name='pooling-layer-3'),
```

```
    tf.keras.layers.Dense(len(class_names), activation="softmax",␣
  ↪name="output-layer")
])
```

[7]:
```
model.compile(loss="categorical_crossentropy", optimizer="adam",␣
  ↪metrics=["accuracy"])
```

[8]:
```
history = model.fit(trainX, trainY_cat, epochs=10, validation_data=(testX,␣
  ↪testY_cat))
```

```
Epoch 1/10
1875/1875              80s 41ms/step -
accuracy: 0.6166 - loss: 1.2695 - val_accuracy: 0.7968 - val_loss: 0.5972
Epoch 2/10
1875/1875              48s 25ms/step -
accuracy: 0.8084 - loss: 0.5460 - val_accuracy: 0.8129 - val_loss: 0.5320
Epoch 3/10
1875/1875              46s 25ms/step -
accuracy: 0.8274 - loss: 0.4839 - val_accuracy: 0.8433 - val_loss: 0.4574
Epoch 4/10
1875/1875              47s 25ms/step -
accuracy: 0.8415 - loss: 0.4441 - val_accuracy: 0.8323 - val_loss: 0.4671
Epoch 5/10
1875/1875              47s 25ms/step -
accuracy: 0.8473 - loss: 0.4259 - val_accuracy: 0.8508 - val_loss: 0.4301
Epoch 6/10
1875/1875              56s 30ms/step -
accuracy: 0.8624 - loss: 0.3952 - val_accuracy: 0.8601 - val_loss: 0.3975
Epoch 7/10
1875/1875              49s 26ms/step -
accuracy: 0.8679 - loss: 0.3760 - val_accuracy: 0.8607 - val_loss: 0.3909
Epoch 8/10
1875/1875              51s 27ms/step -
accuracy: 0.8735 - loss: 0.3594 - val_accuracy: 0.8621 - val_loss: 0.3979
Epoch 9/10
1875/1875              53s 28ms/step -
accuracy: 0.8766 - loss: 0.3504 - val_accuracy: 0.8757 - val_loss: 0.3546
Epoch 10/10
1875/1875              45s 24ms/step -
accuracy: 0.8834 - loss: 0.3331 - val_accuracy: 0.8786 - val_loss: 0.3491
```

[9]:
```
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.
```

[10]:
```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv-layer-1 (Conv2D)	(None, 28, 28, 64)	640
pooling-layer-1 (AveragePooling2D)	(None, 14, 14, 64)	0
conv-layer-2 (Conv2D)	(None, 14, 14, 32)	18,464
pooling-layer-2 (AveragePooling2D)	(None, 7, 7, 32)	0
pooling-layer-3 (GlobalAveragePooling2D)	(None, 32)	0
output-layer (Dense)	(None, 10)	330

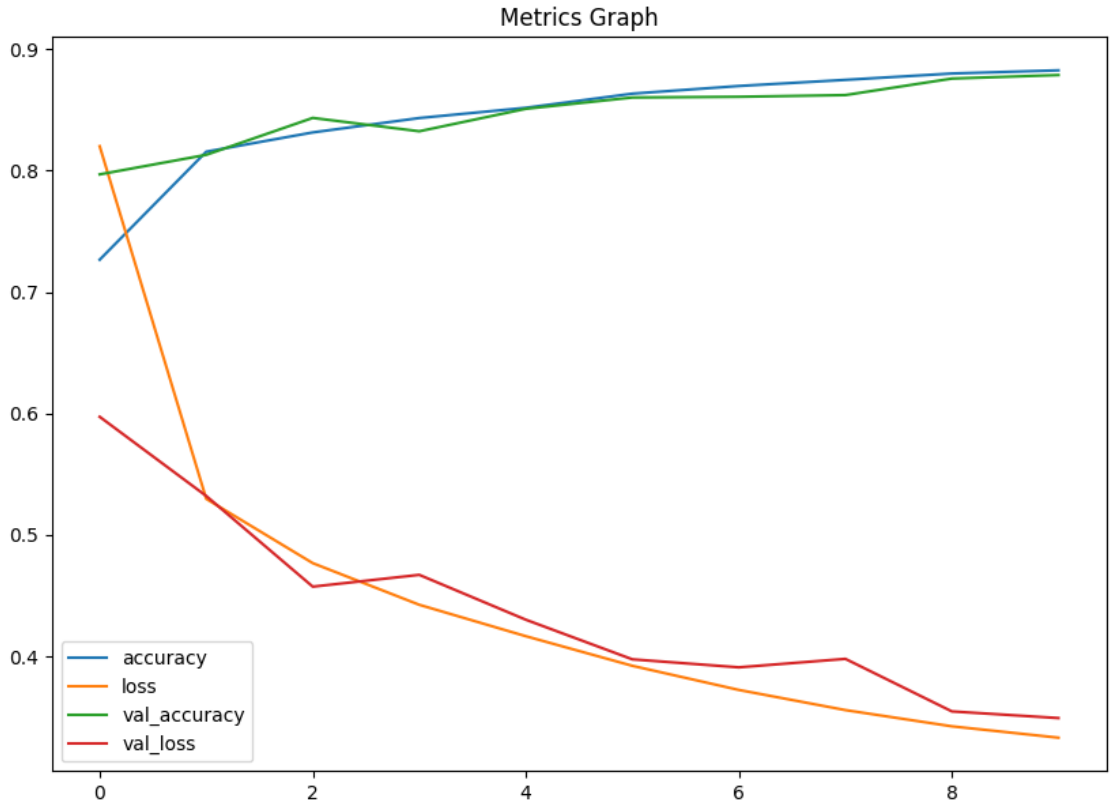
```
Total params: 58,304 (227.75 KB)

Trainable params: 19,434 (75.91 KB)

Non-trainable params: 0 (0.00 B)

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

[11]:
```
pd.DataFrame(history.history).plot(figsize=(10,7))
plt.title("Metrics Graph")
plt.show()
```



[12]:
```
model.evaluate(testX, testY_cat)
```

```
313/313             3s 9ms/step -
accuracy: 0.8810 - loss: 0.3418
```

[12]:
```
[0.3491450846195221, 0.878600001335144]
```

[13]:
```
predictions = model.predict(testX)
```

```
313/313             3s 7ms/step
```

[14]:
```
predictions = tf.argmax(predictions, axis=1)
```

[15]:
```
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.
```

[16]:
```
model.evaluate(testX, testY_cat)
```

```
313/313             3s 10ms/step -
accuracy: 0.8810 - loss: 0.3418
```

[16]: [0.3491450846195221, 0.878600001335144]

[17]:
```
predictions = model.predict(testX)
predictions = tf.argmax(predictions, axis=1)
y_test = tf.argmax(testY_cat, axis=1)
y_test = tf.Variable(y_test)
```

```
313/313             2s 8ms/step
```

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

```
Accuracy:  0.8786
```

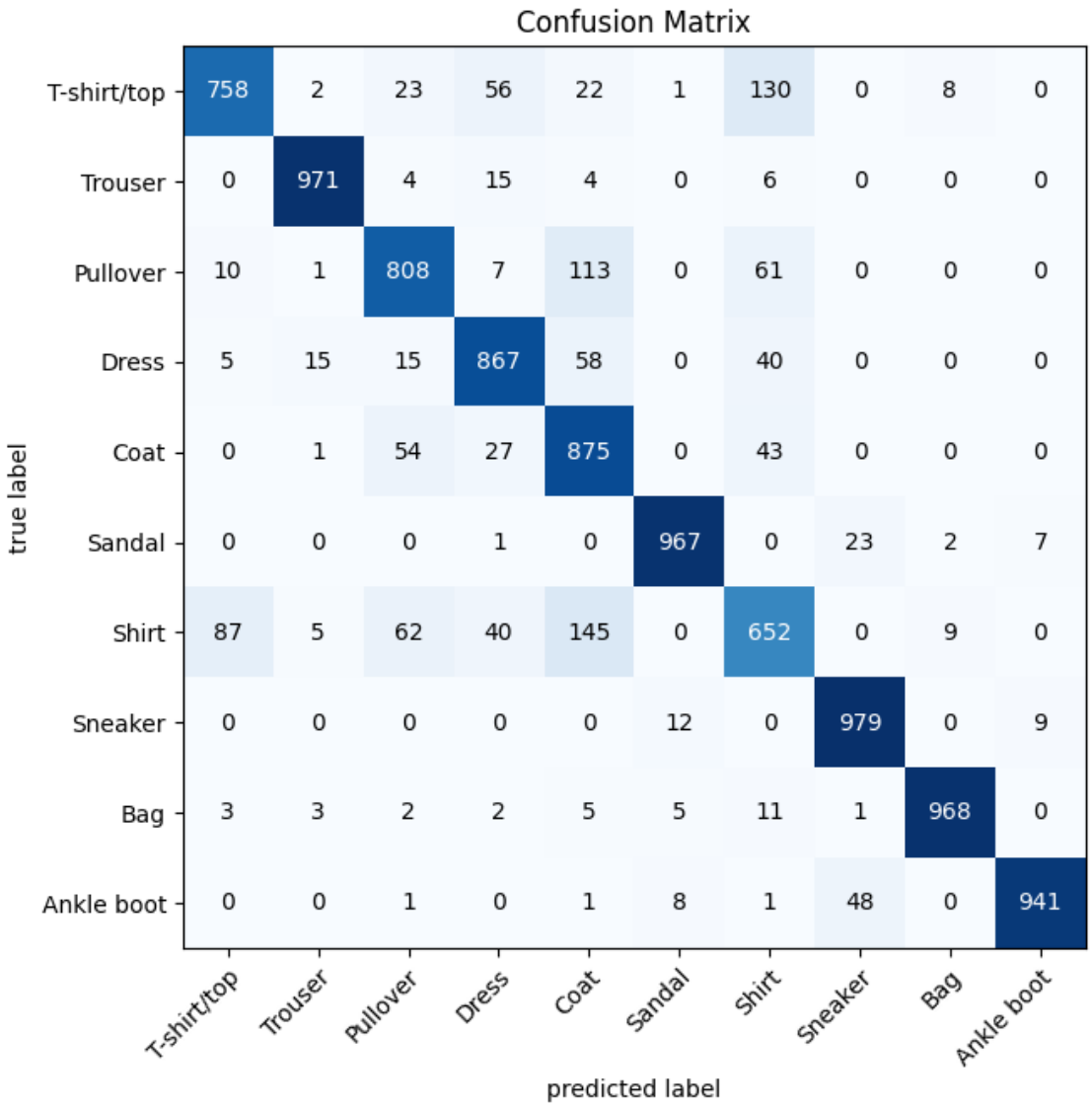
[19]:
```
print(metrics.classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.88      0.76      0.81      1000
           1       0.97      0.97      0.97      1000
           2       0.83      0.81      0.82      1000
           3       0.85      0.87      0.86      1000
           4       0.72      0.88      0.79      1000
           5       0.97      0.97      0.97      1000
           6       0.69      0.65      0.67      1000
           7       0.93      0.98      0.95      1000
           8       0.98      0.97      0.97      1000
           9       0.98      0.94      0.96      1000

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     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
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 108ms/step
1/1             0s 101ms/step
1/1             0s 99ms/step
1/1             0s 160ms/step
1/1             0s 124ms/step
1/1             0s 136ms/step
1/1             0s 101ms/step
1/1             0s 106ms/step
1/1             0s 94ms/step
1/1             0s 93ms/step
```