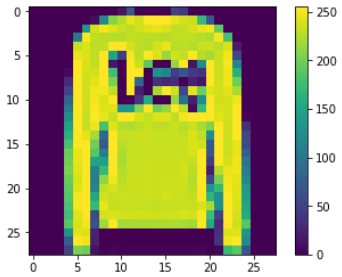
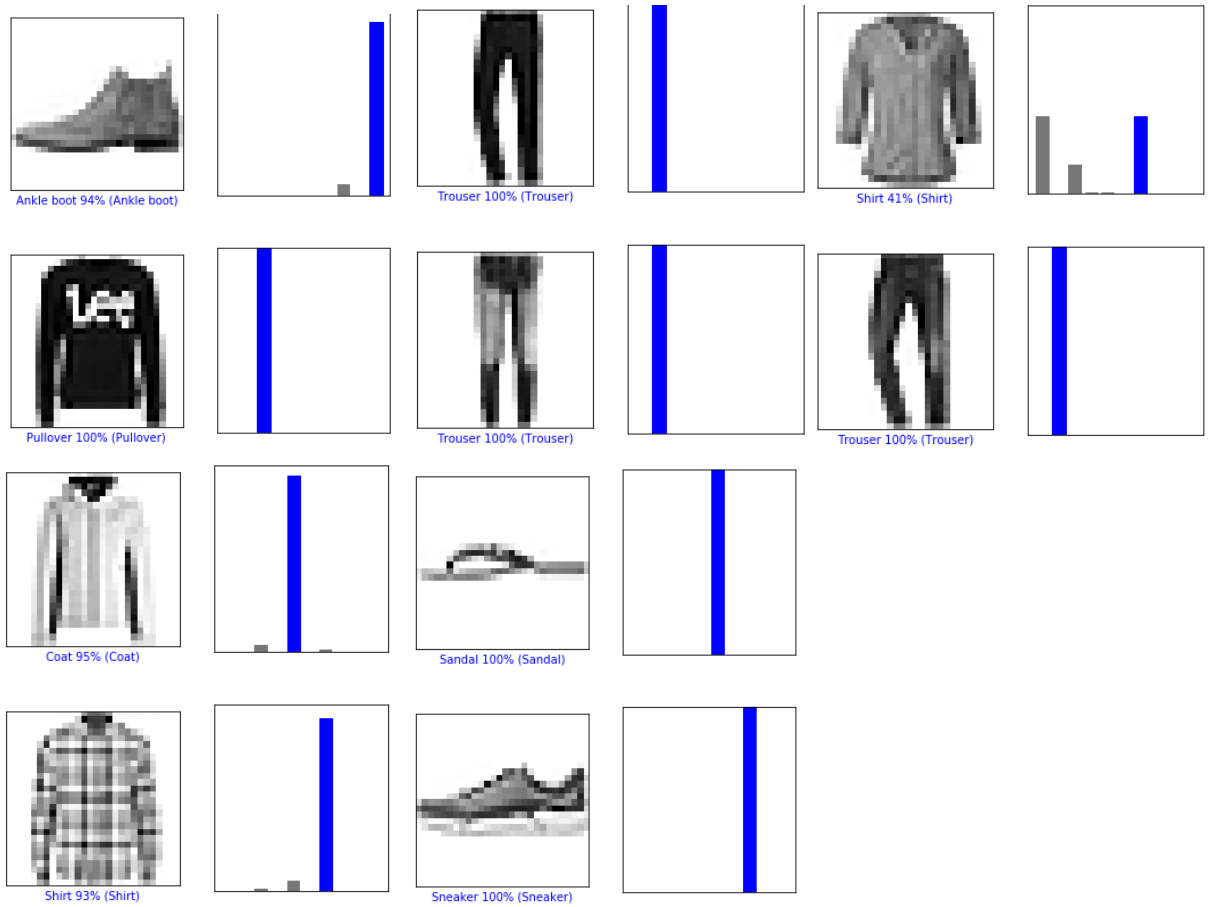
VM581 HW1

P1. Build and train a DNN.

Questions (1)– (5) will shown in the code. And some preprocess and visualization results are presented as followed. Code was run in Spyder.



```
WARNING:tensorflow:From D:\anaconda\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435:
colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
Epoch 1/5
60000/60000 [==============================] - 3s 43us/sample - loss: 0.5075 - acc: 0.8202
Epoch 2/5
60000/60000 [==============================] - 2s 40us/sample - loss: 0.3797 - acc: 0.8634
Epoch 3/5
60000/60000 [==============================] - 2s 38us/sample - loss: 0.3415 - acc: 0.8759
Epoch 4/5
60000/60000 [==============================] - 2s 37us/sample - loss: 0.3148 - acc: 0.8850
Epoch 5/5
60000/60000 [==============================] - 2s 38us/sample - loss: 0.2973 - acc: 0.8900
10000/10000 [==============================] - 0s 21us/sample - loss: 0.3775 - acc: 0.8690
Test accuracy: 0.869
a= [7862 6838 7377 4776  353 6054 1097 7387 1059 9742]
```

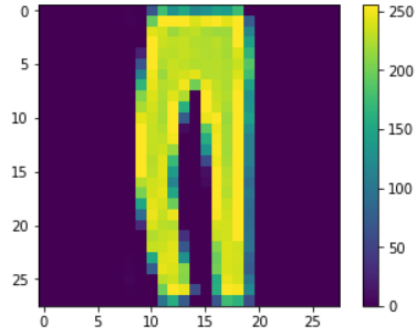


(6) Using DNN, the test accuracy is about 0.869. And results of prediction based on 10 randomly selected testing images can be seen above.

P2. Build and train a CNN.

Questions (1)– (5) will shown in the code. And some preprocess and visualization results are presented as followed. Code was run in Spyder.

```
Num of figures: 60000
Size of figure: 28x28
Num of figures: 10000
Size of figure: 28x28
```

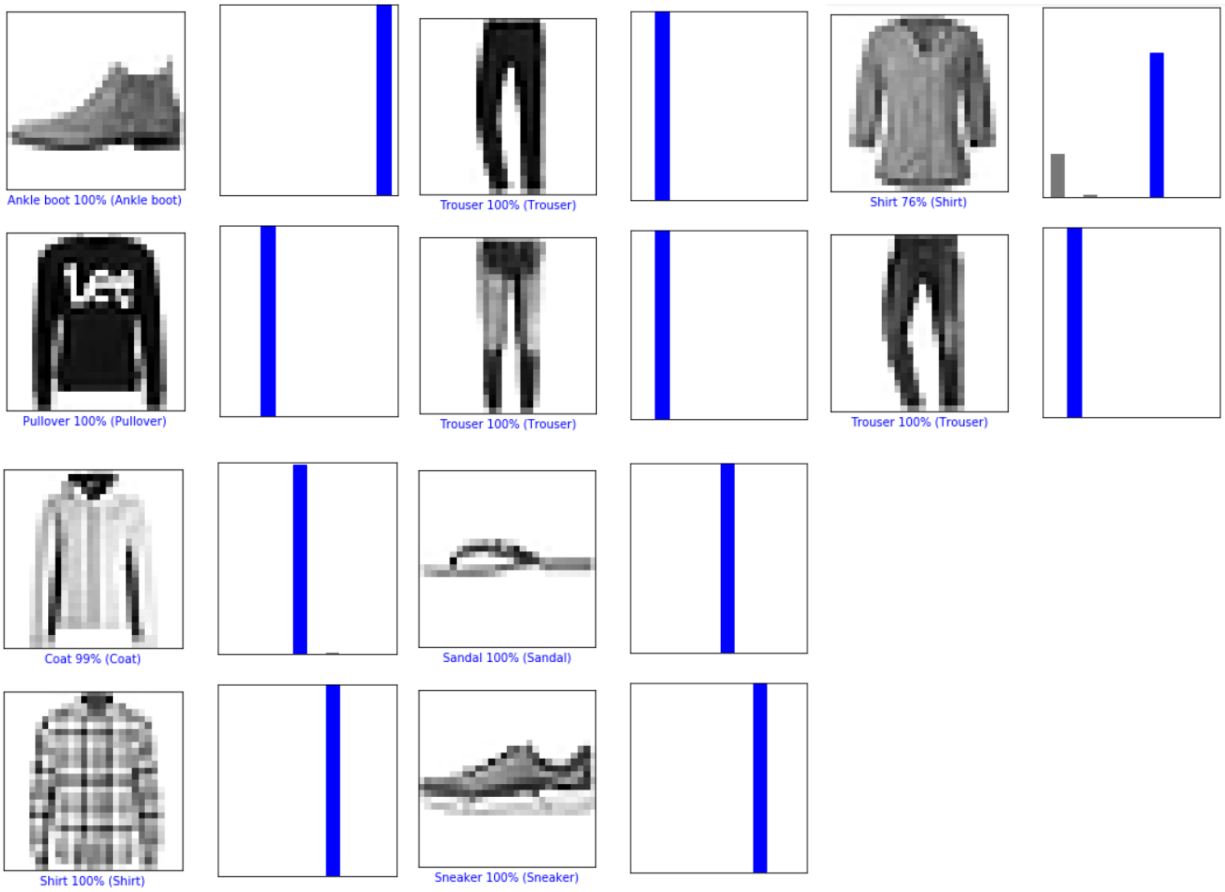



```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
batch_normalization (BatchNo	(None, 28, 28, 1)	4
conv2d (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_1 (Batch	(None, 28, 28, 32)	128
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
dropout (Dropout)	(None, 14, 14, 64)	0
batch_normalization_2 (Batch	(None, 14, 14, 64)	256
conv2d_2 (Conv2D)	(None, 14, 14, 96)	55392
max_pooling2d_1 (MaxPooling2	(None, 7, 7, 96)	0
dropout_1 (Dropout)	(None, 7, 7, 96)	0
flatten (Flatten)	(None, 4704)	0
dense (Dense)	(None, 128)	602240
activation (Activation)	(None, 128)	0
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1290
activation_1 (Activation)	(None, 10)	0

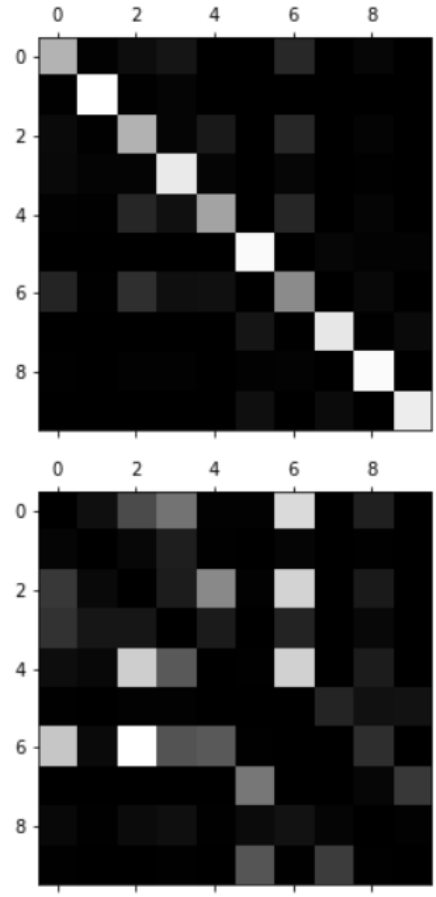
```
Total params: 678,126
Trainable params: 677,932
Non-trainable params: 194
```

```
Epoch 1/10
60000/60000 [==============================] - 23s 384us/sample - loss: 0.5938 - acc: 0.7970
Epoch 2/10
60000/60000 [==============================] - 22s 366us/sample - loss: 0.3286 - acc: 0.8813
Epoch 3/10
60000/60000 [==============================] - 22s 366us/sample - loss: 0.2843 - acc: 0.8982
Epoch 4/10
60000/60000 [==============================] - 22s 372us/sample - loss: 0.2618 - acc: 0.9056
Epoch 5/10
60000/60000 [==============================] - 22s 364us/sample - loss: 0.2460 - acc: 0.9104
Epoch 6/10
60000/60000 [==============================] - 22s 365us/sample - loss: 0.2334 - acc: 0.9153
Epoch 7/10
60000/60000 [==============================] - 22s 363us/sample - loss: 0.2206 - acc: 0.9192
Epoch 8/10
60000/60000 [==============================] - 22s 366us/sample - loss: 0.2127 - acc: 0.9228
Epoch 9/10
60000/60000 [==============================] - 22s 364us/sample - loss: 0.1999 - acc: 0.9273
Epoch 10/10
60000/60000 [==============================] - 22s 362us/sample - loss: 0.1915 - acc: 0.9308

10000/10000 [==============================] - 2s 164us/sample - loss: 0.2171 - acc: 0.9264
Test accuracy: 0.9264

 a= [5058 2768 4768 4276 6954 1162 2432 4260 4428 9921]
```



(6) Using the CNN, the test accuracy is about 0.9264. And 10 randomly selected testing images results can be seen after running the code. Obviously, CNN is more accurate than DNN.

(7) Below are the analysis of error using Confusion Matrix. The diagonal was filled with zeros to keep only the errors



To improve the model's performance, I think using more layers and higher parameters will help it.