Build a DNN using Keras with RELU and ADAM

Load tensorflow

```
!pip install -U tensorflow==2.0.0 --quiet
```

```
     |████████████████████████████████| 86.3MB 26kB/s
     |████████████████████████████████| 3.8MB 50.4MB/s
     |████████████████████████████████| 450kB 52.9MB/s
     |████████████████████████████████| 81kB 10.9MB/s
ERROR: tensorboard 2.0.2 has requirement grpcio>=1.24.3, but you'll have grpcio 1.15.0 w
ERROR: google-colab 1.0.0 has requirement google-auth~=1.4.0, but you'll have google-aut
```

```
import tensorflow as tf
import keras
tf.set_random_seed(42)
```

Collect Fashion mnist data from tf.keras.datasets

```
(trainX, trainY),(testX, testY) = tf.keras.datasets.fashion_mnist.load_data()
```

```
trainX.shape
```

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

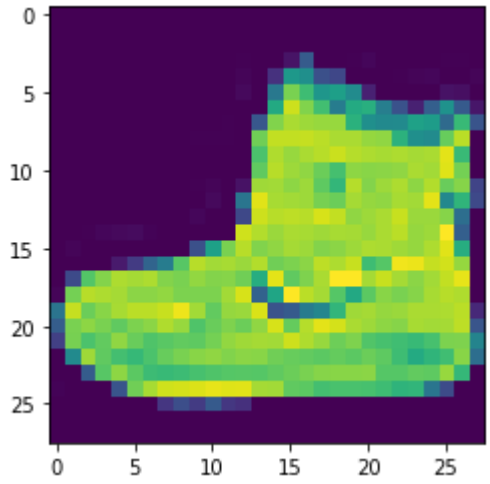
```
testX.shape
```

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

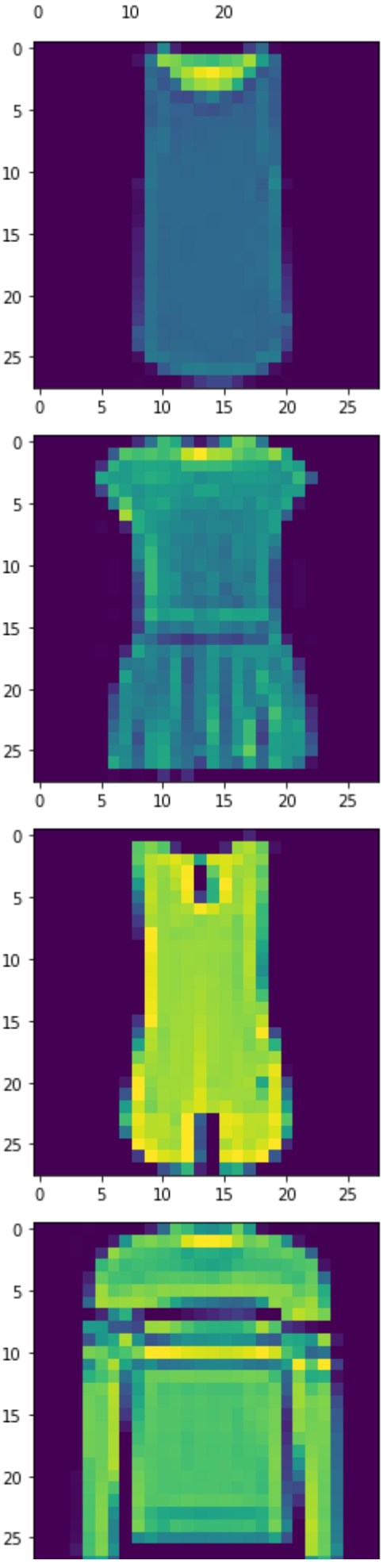
```
testY[0]
```

```
9
```

```
import matplotlib.pyplot as plt
first_array=trainX[0]
plt.imshow(first_array)
plt.show()
```




```
figure =plt.figure(figsize=(3,3))
for i in range(1, 6):
    first_array=trainX[i]
    plt.imshow(first_array)
    plt.show()
```




```
trainX = trainX.reshape(trainX.shape[0], 28, 28, ).astype('float32')
testX = testX.reshape(testX.shape[0], 28, 28, ).astype('float32')
trainX /= 255
testX /= 255

print('--- THE DATA ---')
print('trainX shape:', trainX.shape)
print(trainX.shape[0], 'train samples')
print(testX.shape[0], 'test samples')
```

```
--- THE DATA ---
trainX shape: (60000, 28, 28)
60000 train samples
10000 test samples
```

```
import pandas as pd



## Checking the numbers of unique classification

pd.DataFrame(trainY).nunique()
```

```
0    10
dtype: int64
```

Change train and test labels into one-hot vectors

```
trainY = tf.keras.utils.to_categorical(trainY, num_classes=10)
testY = tf.keras.utils.to_categorical(testY, num_classes=10)
```

Initialize model, reshape & normalize data

```
#Clear out model from current memory
tf.keras.backend.clear_session()

#Initialize Sequential model
```

```
#Initialize Sequential model
model = tf.keras.models.Sequential()

#Reshape data from 2D to 1D -> 28x28 to 784
model.add(tf.keras.layers.Reshape((784,),input_shape=(28,28,)))

#Normalize the data
model.add(tf.keras.layers.BatchNormalization())
```

Add two fully connected layers with 200 and 100 neurons respectively with `relu` activatio

```
# Hidden layers
model.add(tf.keras.layers.Dense(200, activation='relu', name='Layer_1'))
model.add(tf.keras.layers.BatchNormalization())
#Dropout layer
model.add(tf.keras.layers.Dropout(0.25))

model.add(tf.keras.layers.Dense(100, activation='relu', name='Layer_2'))
model.add(tf.keras.layers.BatchNormalization())
#Dropout layer
model.add(tf.keras.layers.Dropout(0.25))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 784)               0
_________________________________________________________________
batch_normalization (BatchNo (None, 784)               3136
_________________________________________________________________
Layer_1 (Dense)              (None, 200)               157000
_________________________________________________________________
batch_normalization_1 (Batch (None, 200)               800
_________________________________________________________________
dropout (Dropout)            (None, 200)               0
_________________________________________________________________
Layer_2 (Dense)              (None, 100)               20100
_________________________________________________________________
batch_normalization_2 (Batch (None, 100)               400
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
=================================================================
Total params: 181,436
Trainable params: 179,268
Non-trainable params: 2,168
_________________________________________________________________
```

Add the output layer with a fully connected layer with 10 neurons with `softmax` ac

```
#Add OUTPUT layer
model.add(tf.keras.layers.Dense(10, activation='softmax'))



#Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',  metrics=['accuracy'])


ckpt = tf.keras.callbacks.ModelCheckpoint('mnist_v1.hdf5', save_best_only=True,
                                          monitor='val_loss', mode='min')
tboard = tf.keras.callbacks.TensorBoard(log_dir='./logs/v2')


BATCH_SIZE = 32
EPOCHS = 10


# Train the model
model.fit(trainX, trainY, batch_size=BATCH_SIZE, nb_epoch=EPOCHS,validation_data=(testX, test
```

```
WARNING:tensorflow:The `nb_epoch` argument in `fit` has been renamed `epochs`.
WARNING:tensorflow:The `nb_epoch` argument in `fit` has been renamed `epochs`.
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 10s 166us/sample - loss: 0.5554 - acc: 0.
Epoch 2/10
60000/60000 [==============================] - 10s 159us/sample - loss: 0.4294 - acc: 0.
Epoch 3/10
60000/60000 [==============================] - 10s 161us/sample - loss: 0.3934 - acc: 0.
Epoch 4/10
60000/60000 [==============================] - 10s 161us/sample - loss: 0.3687 - acc: 0.
Epoch 5/10
60000/60000 [==============================] - 9s 158us/sample - loss: 0.3514 - acc: 0.8
Epoch 6/10
60000/60000 [==============================] - 10s 160us/sample - loss: 0.3396 - acc: 0.
Epoch 7/10
60000/60000 [==============================] - 10s 163us/sample - loss: 0.3293 - acc: 0.
Epoch 8/10
60000/60000 [==============================] - 10s 164us/sample - loss: 0.3204 - acc: 0.
Epoch 9/10
60000/60000 [==============================] - 10s 163us/sample - loss: 0.3091 - acc: 0.
Epoch 10/10
60000/60000 [==============================] - 10s 161us/sample - loss: 0.3061 - acc: 0.
<tensorflow.python.keras.callbacks.History at 0x7fcf46fe49b0>
```

```
loss_and_metrics = model.evaluate(trainX, trainY)
print(loss_and_metrics)
```

```
60000/60000 [==============================] - 3s 46us/sample - loss: 0.2274 - acc: 0.91
[0.22739495115429162, 0.91326666]
```

```
##################################################################################################
##################################################################################################
```