Build a DNN using Keras with RELU and ADAM

Load tensorflow

```
!pip install -U tensorflow==2.0 --quiet
```

```
     |████████████████████████████████| 86.3MB 26kB/s
     |████████████████████████████████| 450kB 22.5MB/s
     |████████████████████████████████| 3.8MB 32.4MB/s
     |████████████████████████████████| 81kB 7.9MB/s
ERROR: tensorboard 2.0.2 has requirement grpcio>=1.24.3, but you'll have grpcio 1.15.
ERROR: google-colab 1.0.0 has requirement google-auth~=1.4.0, but you'll have google-
```

```
import tensorflow as tf
```

```
tf.__version__
```

```
'2.0.0'
```

Collect Fashion mnist data from tf.keras.datasets

```
import keras
```

```
Using TensorFlow backend.
```

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

```
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/trai
32768/29515 [================================] - 0s 4us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/trai
26427392/26421880 [==============================] - 2s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k
8192/5148 [===============================================] - 0s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k
4423680/4422102 [==============================] - 1s 0us/step
```

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)
```

```
testY[0]
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 1.], dtype=float32)
```

```
print(trainY.shape)
print('First 5 examples now are: ', trainY[0:5])
```

```
(60000, 10)
First 5 examples now are:  [[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Build the Graph

Initialize model, reshape & normalize data

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

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

#Normalize the data - getting numbers between 0 to 1
model.add(tf.keras.layers.BatchNormalization())
```

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

```
#Hidden layers
model.add(tf.keras.layers.Dense(200, activation='relu', name='Layer_1'))
model.add(tf.keras.layers.Dense(100, activation='relu', name='Layer_2'))
#Dropout layer
model.add(tf.keras.layers.Dropout(0.25))
```

Add the output layer with a fully connected layer with 10 neurons with `softmax` `categorical_crossentropy` loss and `adam` optimizer and train the network. An

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

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

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
_________________________________________________________________
Layer_2 (Dense)              (None, 100)               20100
_________________________________________________________________
dropout (Dropout)            (None, 100)               0
_________________________________________________________________
Output (Dense)               (None, 10)                1010
=================================================================
Total params: 181,246
Trainable params: 179,678
Non-trainable params: 1,568
_________________________________________________________________
```

```
#Train the model
model.fit(trainX,trainY,
          validation_data=(testX,testY),
          epochs=40,
          batch_size=32)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/40
60000/60000 [==============================] - 13s 219us/sample - loss: 0.5141 - accu
Epoch 2/40
60000/60000 [==============================] - 13s 217us/sample - loss: 0.3966 - accu
Epoch 3/40
60000/60000 [==============================] - 10s 175us/sample - loss: 0.3612 - accu
Epoch 4/40
60000/60000 [==============================] - 11s 189us/sample - loss: 0.3349 - accu
Epoch 5/40
60000/60000 [==============================] - 11s 186us/sample - loss: 0.3157 - accu
Epoch 6/40
60000/60000 [==============================] - 12s 203us/sample - loss: 0.3022 - accu
Epoch 7/40
60000/60000 [==============================] - 12s 196us/sample - loss: 0.2894 - accu
Epoch 8/40
60000/60000 [==============================] - 13s 216us/sample - loss: 0.2763 - accu
Epoch 9/40
60000/60000 [==============================] - 12s 202us/sample - loss: 0.2664 - accu
Epoch 10/40
60000/60000 [==============================] - 11s 190us/sample - loss: 0.2589 - accu
Epoch 11/40
60000/60000 [==============================] - 12s 196us/sample - loss: 0.2472 - accu
Epoch 12/40
60000/60000 [==============================] - 11s 180us/sample - loss: 0.2388 - accu
Epoch 13/40
60000/60000 [==============================] - 11s 180us/sample - loss: 0.2329 - accu
Epoch 14/40
60000/60000 [==============================] - 11s 177us/sample - loss: 0.2259 - accu
Epoch 15/40
60000/60000 [==============================] - 11s 175us/sample - loss: 0.2187 - accu
Epoch 16/40
60000/60000 [==============================] - 10s 168us/sample - loss: 0.2145 - accu
Epoch 17/40
60000/60000 [==============================] - 10s 174us/sample - loss: 0.2084 - accu
Epoch 18/40
60000/60000 [==============================] - 10s 170us/sample - loss: 0.2039 - accu
Epoch 19/40
60000/60000 [==============================] - 9s 157us/sample - loss: 0.1993 - accur
Epoch 20/40
60000/60000 [==============================] - 10s 164us/sample - loss: 0.1941 - accu
Epoch 21/40
60000/60000 [==============================] - 10s 161us/sample - loss: 0.1929 - accu
Epoch 22/40
60000/60000 [==============================] - 10s 164us/sample - loss: 0.1845 - accu
Epoch 23/40
60000/60000 [==============================] - 10s 173us/sample - loss: 0.1818 - accu
Epoch 24/40
60000/60000 [==============================] - 10s 166us/sample - loss: 0.1827 - accu
Epoch 25/40
60000/60000 [==============================] - 10s 168us/sample - loss: 0.1761 - accu
Epoch 26/40
60000/60000 [==============================] - 10s 165us/sample - loss: 0.1708 - accu
Epoch 27/40
60000/60000 [==============================] - 10s 174us/sample - loss: 0.1700 - accu
Epoch 28/40
60000/60000 [==============================] - 10s 165us/sample - loss: 0.1692 - accu
Epoch 29/40
60000/60000 [==============================] - 10s 171us/sample - loss: 0.1635 - accu
Epoch 30/40
60000/60000 [==============================] - 9s 157us/sample - loss: 0.1630 - accur
```

```
Epoch 31/40
60000/60000 [==============================] - 11s 178us/sample - loss: 0.1581 - accu
Epoch 32/40
60000/60000 [==============================] - 10s 173us/sample - loss: 0.1530 - accu
Epoch 33/40
60000/60000 [==============================] - 10s 175us/sample - loss: 0.1527 - accu
Epoch 34/40
60000/60000 [==============================] - 10s 173us/sample - loss: 0.1514 - accu
Epoch 35/40
60000/60000 [==============================] - 10s 170us/sample - loss: 0.1495 - accu
Epoch 36/40
60000/60000 [==============================] - 10s 167us/sample - loss: 0.1493 - accu
Epoch 37/40
60000/60000 [==============================] - 10s 164us/sample - loss: 0.1488 - accu
Epoch 38/40
60000/60000 [==============================] - 10s 171us/sample - loss: 0.1415 - accu
Epoch 39/40
60000/60000 [==============================] - 10s 165us/sample - loss: 0.1399 - accu
Epoch 40/40
60000/60000 [==============================] - 10s 174us/sample - loss: 0.1375 - accu
<tensorflow.python.keras.callbacks.History at 0x7fbdd073b748>
```

```
model.save('dnn_1.h5')
```

```
model = tf.keras.models.load_model('dnn_1.h5')
```