Build a DNN using Keras with RELU and ADAM

Load tensorflow

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

```
The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %tensc
```

Collect Fashion mnist data from tf.keras.datasets

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.
11493376/11490434 [==============================] - 0s 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)
```

Build the Graph

Initialize model, reshape & normalize data

```
#Initialize model, reshape & normalize data
tf.keras.backend.clear_session()
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Reshape((784,),input_shape=(28,28,)))
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())

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))
```

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

```
#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
_________________________________________________________________
batch_normalization_1 (Batch (None, 200)               800
_________________________________________________________________
Layer_2 (Dense)              (None, 100)               20100
_________________________________________________________________
batch_normalization_2 (Batch (None, 100)               400
_________________________________________________________________
dropout (Dropout)            (None, 100)               0
_________________________________________________________________
Output (Dense)               (None, 10)                1010
=================================================================
Total params: 182,446
Trainable params: 180,278
Non-trainable params: 2,168
_________________________________________________________________
```

```
model.fit(trainX,trainY,
          validation_data=(testX,testY),
          epochs=5,
          batch_size=32)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 12s 197us/sample - loss: 0.1831 - acc: 0.
Epoch 2/5
60000/60000 [==============================] - 11s 189us/sample - loss: 0.1282 - acc: 0.
Epoch 3/5
60000/60000 [==============================] - 11s 187us/sample - loss: 0.1003 - acc: 0.
Epoch 4/5
60000/60000 [==============================] - 11s 191us/sample - loss: 0.0896 - acc: 0.
Epoch 5/5
60000/60000 [==============================] - 11s 188us/sample - loss: 0.0745 - acc: 0.
<tensorflow.python.keras.callbacks.History at 0x7f4cea3665c0>
```