Build a DNN using Keras with `RELU` and `ADAM`

Load tensorflow

```
import tensorflow as tf
import numpy as np
```

Collect Fashion mnist data from tf.keras.datasets

```
from keras.datasets import fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

Change train and test labels into one-hot vectors

```
y_train = tf.keras.utils.to_categorical(y_train, num_classes=10)
y_test = tf.keras.utils.to_categorical(y_test, num_classes=10)
```

Build the Graph

Initialize model, reshape & normalize data

```
y_train.shape
```

```
(60000, 10)
```

```
y_test.shape
```

```
(10000, 10)
```

```
x_test.shape
```

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

```
x_train.shape
```

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

```
x_train.dtype
```

```
dtype('uint8')
```

```
x_test=x_test.astype(np.int32)
```

```
x_train=x_train.astype(np.int32)
```

```
y_train=y_train.astype(np.int32)
```

```
y_test=y_test.astype(np.int32)
```

```
x_train=x_train.reshape(x_train.shape[0],28,28,1).astype('float32')
x_test=x_test.reshape(x_test.shape[0],28,28,1).astype('float32')
```

```
x_train.shape
```

```
(60000, 28, 28, 1)
```

```
x_train=x_train/255
```

```
x_test=x_test/255
```

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

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten, Reshape
from keras.layers import Convolution2D, MaxPooling2D


# Define model
    model2 = Sequential()

    # 1st Conv Layer
    model2.add(Convolution2D(32, 3, 3, input_shape=(28, 28, 1)))
    model2.add(Activation('relu'))

    # 2nd Conv Layer
    model2.add(Convolution2D(32, 3, 3))
    model2.add(Activation('relu'))

    # Fully Connected Layer
    model2.add(Flatten())
    model2.add(Dense(200))
    model2.add(Activation('relu'))



    # Prediction Layer
    model2.add(Dense(10))
    model2.add(Activation('softmax'))
```

```
# Loss and Optimizer
model2.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Store Training Results
early_stopping = keras.callbacks.EarlyStopping(monitor='val_acc', patience=5, verbose=1,
callback_list = [early_stopping]

# Train the model2
model2.fit(x_train, y_train, batch_size=32, nb_epoch=10,
          validation_data=(x_test, y_test), callbacks=callback_list)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: UserWarning: Update your
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: UserWarning: Update your

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:30: UserWarning: The `nb_ep
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 10s 168us/step - loss: 0.3729 - acc: 0.86
Epoch 2/10
60000/60000 [==============================] - 10s 161us/step - loss: 0.2328 - acc: 0.91
Epoch 3/10
60000/60000 [==============================] - 10s 162us/step - loss: 0.1656 - acc: 0.93
Epoch 4/10
60000/60000 [==============================] - 10s 162us/step - loss: 0.1138 - acc: 0.95
Epoch 5/10
60000/60000 [==============================] - 10s 162us/step - loss: 0.0727 - acc: 0.97
Epoch 6/10
60000/60000 [==============================] - 10s 167us/step - loss: 0.0482 - acc: 0.98
Epoch 00006: early stopping
<keras.callbacks.History at 0x7fbcea1a8eb8>
```

```
# Define model
model2 = Sequential()

# 1st Conv Layer
model2.add(Convolution2D(32, 3, 3, input_shape=(28, 28, 1)))
model2.add(Activation('relu'))

# 2nd Conv Layer
model2.add(Convolution2D(32, 3, 3))
model2.add(Activation('relu'))

# Fully Connected Layer
model2.add(Flatten())
model2.add(Dense(100))
model2.add(Activation('relu'))


# Prediction Layer
model2.add(Dense(10))
```

```
model2.add(Activation('softmax'))

# Loss and Optimizer
model2.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Store Training Results
early_stopping = keras.callbacks.EarlyStopping(monitor='val_acc', patience=5, verbose=1,
callback_list = [early_stopping]

# Train the model2
model2.fit(x_train, y_train, batch_size=32, nb_epoch=10,
          validation_data=(x_test, y_test), callbacks=callback_list)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: UserWarning: Update your
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: UserWarning: Update your

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:30: UserWarning: The `nb_ep
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 9s 152us/step - loss: 0.3812 - acc: 0.862
Epoch 2/10
60000/60000 [==============================] - 9s 148us/step - loss: 0.2353 - acc: 0.913
Epoch 3/10
60000/60000 [==============================] - 9s 145us/step - loss: 0.1709 - acc: 0.937
Epoch 4/10
60000/60000 [==============================] - 9s 146us/step - loss: 0.1197 - acc: 0.955
Epoch 5/10
60000/60000 [==============================] - 9s 145us/step - loss: 0.0816 - acc: 0.969
Epoch 6/10
60000/60000 [==============================] - 9s 148us/step - loss: 0.0554 - acc: 0.979
Epoch 00006: early stopping
<keras.callbacks.History at 0x7fbcea05c710>
```

```
# Define Model
model3 = Sequential()

# 1st Conv Layer
model3.add(Convolution2D(32, 3, 3, input_shape=(28, 28, 1)))
model3.add(Activation('relu'))

# 2nd Conv Layer
model3.add(Convolution2D(32, 3, 3))
model3.add(Activation('relu'))

# Max Pooling
model3.add(MaxPooling2D(pool_size=(2,2)))

# Dropout
model3.add(Dropout(0.25))

# Fully Connected Layer
```

```
model3.add(Flatten())
model3.add(Dense(128))
model3.add(Activation('relu'))

# More Dropout
model3.add(Dropout(0.5))

# Prediction Layer
model3.add(Dense(10))
model3.add(Activation('softmax'))

# Loss and Optimizer
model3.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Store Training Results
early_stopping = keras.callbacks.EarlyStopping(monitor='val_acc', patience=7, verbose=1,
callback_list = [early_stopping]

# Train the model
model3.fit(x_train, y_train, batch_size=32, nb_epoch=10,
          validation_data=(x_test, y_test), callbacks=callback_list)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: UserWarning: Update your
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: UserWarning: Update your

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:38: UserWarning: The `nb_ep
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 9s 148us/step - loss: 0.5050 - acc: 0.822
Epoch 2/10
60000/60000 [==============================] - 9s 142us/step - loss: 0.3471 - acc: 0.875
Epoch 3/10
60000/60000 [==============================] - 9s 142us/step - loss: 0.3012 - acc: 0.890
Epoch 4/10
60000/60000 [==============================] - 9s 143us/step - loss: 0.2689 - acc: 0.901
Epoch 5/10
60000/60000 [==============================] - 9s 150us/step - loss: 0.2484 - acc: 0.907
Epoch 6/10
60000/60000 [==============================] - 9s 142us/step - loss: 0.2321 - acc: 0.914
Epoch 7/10
60000/60000 [==============================] - 9s 143us/step - loss: 0.2188 - acc: 0.918
Epoch 8/10
60000/60000 [==============================] - 8s 140us/step - loss: 0.2080 - acc: 0.922
Epoch 9/10
60000/60000 [==============================] - 8s 141us/step - loss: 0.1962 - acc: 0.927
Epoch 10/10
60000/60000 [==============================] - 9s 143us/step - loss: 0.1886 - acc: 0.929
<keras.callbacks.History at 0x7fbce10ef9b0>
```

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

```
loss_and_metrics = model3.evaluate(x_test, y_test)
print(loss_and_metrics)
```

```
10000/10000 [==============================] - 1s 52us/step
[2.6105039825439453, 0.1]
```

```
loss_and_metrics = model3.evaluate(x_train, y_train)
print(loss_and_metrics)
```

```
60000/60000 [==============================] - 3s 50us/step
[0.1047543773808827, 0.9632666666666667]
```