A MNIST-like fashion product database

In this, we classify the images into respective classes given in the dataset. We use a Neural Net ar and check the accuracy scores.

Load tensorflow

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

```
tf.__version__
```

```
'1.15.0'
```

Collect Data

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

```
     |████████████████████████████████| 86.3MB 30kB/s
     |████████████████████████████████| 3.8MB 38.0MB/s
     |████████████████████████████████| 450kB 55.5MB/s
     |████████████████████████████████| 81kB 9.7MB/s
ERROR: google-colab 1.0.0 has requirement google-auth~=1.4.0, but you'll have google-
ERROR: tensorboard 2.0.1 has requirement grpcio>=1.24.3, but you'll have grpcio 1.15.
```

```
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 8us/step
40960/29515 [==========================================] - 0s 7us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/trai
26427392/26421880 [==============================] - 6s 0us/step
26435584/26421880 [==============================] - 6s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k
16384/5148 [===============================================================================
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k
4423680/4422102 [==============================] - 2s 0us/step
4431872/4422102 [==============================] - 2s 0us/step
```

```
print(testY[0:5])
```

```
[9 2 1 1 6]
```

```
trainX.shape
```

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

```
testX.shape
```

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

```
testY.shape
```

```
(10000,)
```

```
trainY.shape
```

```
(60000,)
```

```
testY[0]
```

```
9
```

Convert both training and testing labels into one-hot vectors.

Hint: check **tf.keras.utils.to_categorical()**

```
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: ', array([[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.]], dtype=float32))
```

Visualize the data

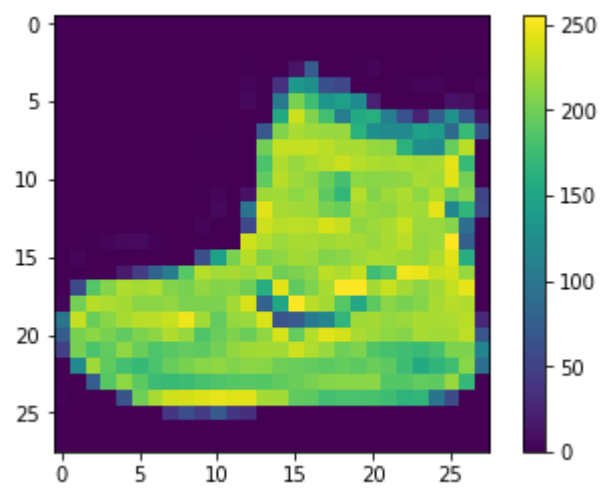
Plot first 10 images in the triaining set and their labels.

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

```
import matplotlib.pyplot as plt
```

```
plt.figure()
```

```
plt.imshow(trainX[0])
plt.colorbar()
plt.grid(False)
plt.show()
```




```
trainX = trainX / 255.0
testX = testX / 255.0
```

```
trainY
```

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

```
%matplotlib inline
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for i in range(10):
plt.subplot(5,5,i+1)
plt.xticks([])
plt.yticks([])
plt.grid(False)
plt.imshow(trainX[i], cmap=plt.cm.binary)
plt.xlabel(class_names[trainY[i]])
plt.show()
```

Build a neural Network with a cross entropy loss function and sgd optimizer in neurons as we have 10 classes.

```
trainX.dtype
```

```
dtype('float64')
```

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Reshape((784,),input_shape=(28,28,)))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

```
W1110 13:00:04.724555 139628989691776 deprecation.py:506] From /usr/local/lib/python2
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

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

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 10)                7850
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_________________________________________________________________
```

Execute the model using model.fit()

```
model.fit(trainX,trainY,
validation_data=(testX,testY),
epochs=100,
batch_size=32)
```

```
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3893 - acc: 0
Epoch 73/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3888 - acc: 0
Epoch 74/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3887 - acc: 0
Epoch 75/100
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3882 - acc: 0
Epoch 76/100
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3880 - acc: 0
Epoch 77/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3877 - acc: 0
Epoch 78/100
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3874 - acc: 0
Epoch 79/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3871 - acc: 0
Epoch 80/100
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3867 - acc: 0
Epoch 81/100
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3864 - acc: 0
Epoch 82/100
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3862 - acc: 0
Epoch 83/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3860 - acc: 0
Epoch 84/100
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3857 - acc: 0
Epoch 85/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3855 - acc: 0
Epoch 86/100
60000/60000 [==============================] - 3s 45us/sample - loss: 0.3852 - acc: 0
Epoch 87/100
60000/60000 [==============================] - 3s 45us/sample - loss: 0.3849 - acc: 0
Epoch 88/100
60000/60000 [==============================] - 3s 45us/sample - loss: 0.3846 - acc: 0
Epoch 89/100
60000/60000 [==============================] - 3s 44us/sample - loss: 0.3846 - acc: 0
Epoch 90/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3842 - acc: 0
Epoch 91/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3841 - acc: 0
Epoch 92/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3834 - acc: 0
Epoch 93/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3836 - acc: 0
Epoch 94/100
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3833 - acc: 0
Epoch 95/100
60000/60000 [==============================] - 2s 41us/sample - loss: 0.3832 - acc: 0
Epoch 96/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3829 - acc: 0
Epoch 97/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3828 - acc: 0
Epoch 98/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3823 - acc: 0
Epoch 99/100
60000/60000 [==============================] - 3s 43us/sample - loss: 0.3820 - acc: 0
Epoch 100/100
60000/60000 [==============================] - 3s 42us/sample - loss: 0.3819 - acc: 0
<tensorflow.python.keras.callbacks.History at 0x7efdac9ff8d0>
```

In the above Neural Network model add Batch Normalization layer after the inp

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Reshape((784,),input_shape=(28,28,)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Dense(10, activation='softmax'))
model.compile(optimizer='sgd', loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
reshape_1 (Reshape)          (None, 784)               0         
_________________________________________________________________
batch_normalization (BatchNo (None, 784)               3136      
_________________________________________________________________
dense_1 (Dense)              (None, 10)                7850      
=================================================================
Total params: 10,986
Trainable params: 9,418
Non-trainable params: 1,568
_________________________________________________________________
```

Execute the model

```
model.fit(trainX,trainY,
validation_data=(testX,testY),
epochs=100,
batch_size=32)
```

```
Epoch 68/100
60000/60000 [==============================] - 4s 58us/sample - loss: 0.3978 - acc: 0
Epoch 69/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3988 - acc: 0
Epoch 70/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3989 - acc: 0
Epoch 71/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3983 - acc: 0
Epoch 72/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.4013 - acc: 0
Epoch 73/100
60000/60000 [==============================] - 3s 57us/sample - loss: 0.3999 - acc: 0
Epoch 74/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3980 - acc: 0
Epoch 75/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3988 - acc: 0
Epoch 76/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.4004 - acc: 0
Epoch 77/100
60000/60000 [==============================] - 4s 58us/sample - loss: 0.4004 - acc: 0
Epoch 78/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3997 - acc: 0
Epoch 79/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3988 - acc: 0
Epoch 80/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3968 - acc: 0
Epoch 81/100
60000/60000 [==============================] - 4s 58us/sample - loss: 0.3966 - acc: 0
Epoch 82/100
60000/60000 [==============================] - 4s 59us/sample - loss: 0.3965 - acc: 0
Epoch 83/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3951 - acc: 0
Epoch 84/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3970 - acc: 0
Epoch 85/100
60000/60000 [==============================] - 4s 59us/sample - loss: 0.3970 - acc: 0
Epoch 86/100
60000/60000 [==============================] - 4s 59us/sample - loss: 0.3965 - acc: 0
Epoch 87/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3978 - acc: 0
Epoch 88/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3970 - acc: 0
Epoch 89/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3975 - acc: 0
Epoch 90/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3962 - acc: 0
Epoch 91/100
60000/60000 [==============================] - 3s 57us/sample - loss: 0.3975 - acc: 0
Epoch 92/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3975 - acc: 0
Epoch 93/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3963 - acc: 0
Epoch 94/100
60000/60000 [==============================] - 4s 59us/sample - loss: 0.3949 - acc: 0
Epoch 95/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3960 - acc: 0
Epoch 96/100
60000/60000 [==============================] - 4s 59us/sample - loss: 0.3984 - acc: 0
Epoch 97/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3980 - acc: 0
Epoch 98/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3948 - acc: 0
```

```
Epoch 99/100
60000/60000 [==============================] - 3s 58us/sample - loss: 0.3983 - acc: 0
Epoch 100/100
60000/60000 [==============================] - 4s 59us/sample - loss: 0.3961 - acc: 0
<tensorflow.python.keras.callbacks.History at 0x7efdaf295ed0>
```

Customize the learning rate to 0.001 in sgd optimizer and run the model

```
sgd_optimizer = tf.keras.optimizers.SGD(lr=0.001)
model.compile(optimizer='sgd', loss='categorical_crossentropy',
metrics=['accuracy'])


model.fit(trainX,trainY,
validation_data=(testX,testY),
epochs=100,
batch_size=32)
```