```
#import libraries
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import keras
from keras.datasets import fashion_mnist #
import keras.models as models
import keras.layers as layers
from keras import regularizers
from keras.layers import Dropout


#from keras.engine.sequential import Sequential
#tensor flow-> layers
import tensorflow as tf
from tensorflow.keras.models import Sequential,Model ##squnce of process
from tensorflow.keras.layers import Dense,Activation,Flatten,Dropout,Conv2D,MaxPooling2D
from keras.layers.advanced_activations import LeakyReLU
from tensorflow.keras.utils import to_categorical #for catagorical data


from google.colab import drive


drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
cd /content/drive/"MyDrive/dataset_ml/Train_set"
```

```
/content/drive/MyDrive/dataset_ml/Train_set
```

```
dataset = pd.read_csv('train.csv')
dataset.shape
```

```
(7095, 3)
```

```
dataset
```

```
data=dataset.values[:7095,0:2]#5000 image for train
print(data.shape)
data
```

```
(7095, 2)
array([['0002cc93b.jpg', 1],
       ['0007a71bf.jpg', 3],
       ['000a4bcdd.jpg', 1],
       ...,
       ['fffe98443.jpg', 3],
       ['ffff4eaa8.jpg', 3],
       ['ffffd67df.jpg', 3]], dtype=object)
```

```
#code for taking class 2 only
i=0
j=0
k=0
democlasses=[]
demoimage=[]
while(i!=245):
  if(data[j][1]==2):
    democlasses.append(data[j][1])
    demoimage.append(data[j][0])
    i+=1
    j+=1
  else:
    j+=1

#code for taking class 1 only
i=0
j=0
while(i!=245):
  if(data[j][1]==1):
    democlasses.append(data[j][1])
    demoimage.append(data[j][0])
    i+=1
    j+=1
  else:
    j+=1

#code for taking class 3 only
i=0
j=0
```

```
while(i!=245):
  if(data[j][1]==3):
    democlasses.append(data[j][1])
    demoimage.append(data[j][0])
    i+=1
    j+=1
  else:
    j+=1

#code for taking class 4 only
i=0
j=0
while(i!=245):
  if(data[j][1]==4):
    democlasses.append(data[j][1])
    demoimage.append(data[j][0])
    i+=1
    j+=1
  else:
    j+=1

#democlasses = np.array(democlasses)
print(democlasses)
print(type(democlasses))
print(len(democlasses))
```

```
[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
<class 'list'>
980
```

```
image=[]
classes=[]
for i in range(980):
  img=cv2.imread(data[i][0])#read 5000 setof image
  img2=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
  image.append(img2)#append all the image in imge
  classes.append(data[i][1])#append all the class in classes

#px.imshow(img2,binary_string=True)
print(type(image))
print(type(classes))
image=np.array(image)
classes=np.array(classes)
print(type(image))
print(type(classes))
image
```

```
<class 'list'>
<class 'list'>
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
array([[[ 70,  70,  68, ...,  48,  48,  50],
        [ 66,  68,  68, ...,  48,  49,  51],
        [ 61,  64,  65, ...,  49,  51,  54],
```

```
       ...,
       [155, 133, 131, ...,  51,  51,  50],
       [160, 111, 100, ...,  55,  54,  48],
       [155, 114,  98, ...,  58,  58,  50]],

      [[ 47,  49,  49, ...,  65,  67,  63],
       [ 49,  51,  52, ...,  64,  66,  67],
       [ 49,  51,  51, ...,  61,  62,  67],
       ...,
       [106, 109, 100, ...,  98,  86,  85],
       [103, 110, 106, ...,  86,  85,  85],
       [103, 111, 107, ...,  83,  90,  90]],

      [[ 52,  51,  51, ...,  45,  45,  44],
       [ 53,  50,  49, ...,  48,  48,  47],
       [ 54,  51,  50, ...,  47,  47,  47],
       ...,
       [ 77,  78,  78, ...,  76,  75,  79],
       [ 72,  79,  78, ...,  76,  75,  78],
       [ 69,  79,  78, ...,  74,  74,  78]],

      ...,

      [[ 62,  69,  64, ...,  59,  57,  56],
       [ 69,  68,  59, ...,  58,  58,  58],
       [ 68,  69,  63, ...,  55,  54,  56],
       ...,
       [ 72,  75,  78, ...,  68,  69,  68],
       [ 70,  71,  75, ...,  66,  65,  63],
       [ 74,  71,  71, ...,  65,  65,  64]],

      [[ 51,  53,  54, ...,   0,   0,   0],
       [ 52,  53,  53, ...,   0,   0,   0],
       [ 51,  52,  52, ...,   0,   0,   0],
       ...,
       [ 62,  62,  62, ...,   1,   1,   1],
       [ 60,  59,  56, ...,   1,   1,   1],
       [ 58,  59,  58, ...,   1,   1,   1]],

      [[ 46,  46,  47, ...,  50,  49,  44],
       [ 50,  50,  50, ...,  47,  49,  48],
       [ 49,  49,  50, ...,  46,  47,  48],
       ...,
       [106, 110, 112, ..., 113, 116, 109],
       [108, 112, 116, ..., 115, 120, 110],
       [105, 108, 116, ..., 114, 119, 109]]], dtype=uint8)
```

```
plt.imshow(image[0])
print(image[0])
```

```
#model training task
#split into validation and train
from sklearn.model_selection import train_test_split
train_x,test_x,train_y,test_y=train_test_split(image,classes,test_size=0.2,random_state=13
```

```
print(train_x.shape,train_y.shape)
```

```
(784, 256, 1600) (784,)
```

```
print(test_x.shape,test_y.shape)
```

```
(196, 256, 1600) (196,)
```

```
classes=np.unique(train_y)
nclasses=len(classes)
print(classes)
print(nclasses)
```

```
[1 2 3 4]
4
```

```
classes=np.unique(test_y)
nclasses=len(classes)
print(classes)
print(nclasses)
```

```
[1 2 3 4]
4
```

```
#reshape image
train_x=train_x.reshape(-1,256,1600,1)
test_x=test_x.reshape(-1,256,1600,1)
```

```
print(train_x.shape,train_y.shape)
print(test_x.shape,test_y.shape)
```

```
(784, 256, 1600, 1) (784,)
(196, 256, 1600, 1) (196,)
```

```
train_y.shape[0]
```

```
784
```

```
#converting value 0-1
#type convertion to avoid integer
train_x=train_x.astype('float32')
test_x=test_x.astype('float32')
train_x=train_x/255
test_x=test_x/255


train_x
```

```
array([[[[0.44705883],
         [0.43137255],
         [0.4117647 ],
         ...,
         [0.21568628],
         [0.21176471],
         [0.21960784]],

        [[0.3647059 ],
         [0.3529412 ],
         [0.3254902 ],
         ...,
         [0.24313726],
         [0.23529412],
         [0.23921569]],

        [[0.33333334],
         [0.3647059 ],
         [0.36862746],
         ...,
         [0.24705882],
         [0.22745098],
         [0.22352941]],

        ...,

        [[0.56078434],
         [0.57254905],
         [0.5647059 ],
         ...,
         [0.4117647 ],
         [0.41960785],
         [0.3882353 ]],

        [[0.5294118 ],
         [0.54901963],
         [0.5254902 ],
         ...,
         [0.4509804 ],
         [0.48235294],
         [0.42745098]],

        [[0.52156866],
         [0.54901963],
         [0.49803922],
         ...,
         [0.44313726],
         [0.48235294],
         [0.41568628]]],
```

```
            [[[0.01568628],
              [0.01568628],
              [0.01568628],
              ...,
              [0.1254902 ],
              [0.12156863],
              [0.12156863]],
```

```
train_one_hot=to_categorical(train_y)
test_one_hot=to_categorical(test_y)
```

```
print(train_one_hot[777])
print(train_one_hot)
```

```
     [0. 0. 0. 0. 1.]
     [[0. 0. 0. 0. 1.]
      [0. 0. 0. 0. 1.]
      [0. 0. 0. 1. 0.]
      ...
      [0. 0. 0. 1. 0.]
      [0. 0. 0. 1. 0.]
      [0. 0. 0. 1. 0.]]
```

```
train_y_one_hot = []
#train_y_one_hot = np.append(train_y_one_hot, np.array([[11, 21, 31, 41]]), axis=0)
print(train_y_one_hot)
#train_y_one_hot=np.array(train_y_one_hot)
test_y_one_hot=[]
#test_y_one_hot=np.array(test_y_one_hot)
```

```
     []
```

```
for i in range(0,784):
  train=np.delete(train_one_hot[i],0)
  train_y_one_hot.append(train)
  #train_y_one_hot = np.append(train_y_one_hot, train, axis=0)
  #print(train)

for i in range(0,196):
  test=np.delete(test_one_hot[i],0)
  test_y_one_hot.append(test)
  #train_y_one_hot = np.append(train_y_one_hot, train, axis=0)
  #print(train)
```

```
train_y_one_hot=np.array(train_y_one_hot)
test_y_one_hot=np.array(test_y_one_hot)
```

```
print(train_y_one_hot)
print(type(train_y_one_hot))
```

```
[[0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 ...
 [0. 0. 1. 0.]
 [0. 0. 1. 0.]
 [0. 0. 1. 0.]]
<class 'numpy.ndarray'>
```

```
print(train_y_one_hot[1])
print(type(train_y_one_hot))
```

```
[0. 0. 0. 1.]
<class 'numpy.ndarray'>
```

```
classes=np.unique(test_y_one_hot)
nclasses=len(classes)
print(classes)
print(nclasses)
```

```
[0. 1.]
2
```

```
test_y_one_hot[1]
```

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

```
train_y_one_hot[1]
```

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

```
#model training task
#split into validation and train
from sklearn.model_selection import train_test_split
train_x,valid_x,train_label,valid_label=train_test_split(train_x,train_y_one_hot,test_size
```

```
train_x.shape,valid_x.shape,train_label.shape,valid_label.shape
```

```
((627, 256, 1600, 1), (157, 256, 1600, 1), (627, 4), (157, 4))
```

```
batch_size=10#there is total 48000 image from that we are taking 64 student batch
epochs=20
num_classes=4
```

```
#declaration of Sequential model
model=tf.keras.Sequential()
```

```
#1 hidden layer
model.add(tf.keras.layers.Conv2D(32,(3,3),activation="linear",padding="same"))#valid->not
model.add(tf.keras.layers.LeakyReLU(alpha=0.1))#alpha is slop of line in nagative part
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2,2),padding="same"))


#2 hidden layer
model.add(tf.keras.layers.Conv2D(64,(3,3),activation="linear",padding="same"))#valid->not
model.add(tf.keras.layers.LeakyReLU(alpha=0.1))#alpha is slop of line in nagative part
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2,2),padding="same"))


#3 hidden layer
model.add(tf.keras.layers.Conv2D(128,(3,3),activation="linear",padding="same"))#valid->not
model.add(tf.keras.layers.LeakyReLU(alpha=0.1))#alpha is slop of line in nagative part
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2,2),padding="same"))


model.add(tf.keras.layers.Flatten())


model.add(tf.keras.layers.Dense(128,activation="linear"))
model.add(tf.keras.layers.LeakyReLU(alpha=0.1))
#output final layer
model.add(tf.keras.layers.Dense(num_classes,activation='softmax'))#softmax because we want


model.compile(loss=tf.keras.losses.categorical_crossentropy,optimizer=tf.keras.optimizers.


print(valid_x.shape)
print(valid_label.shape)
```

```
(157, 256, 1600, 1)
(157, 4)
```

```
model_train=model.fit(train_x,train_label,batch_size=10,epochs=10,verbose=1,validation_dat
```

```
Epoch 1/10
63/63 [==============================] - 538s 8s/step - loss: 3.2072 - accuracy: 0.64
Epoch 2/10
63/63 [==============================] - 539s 9s/step - loss: 0.8347 - accuracy: 0.72
Epoch 3/10
63/63 [==============================] - 528s 8s/step - loss: 0.8108 - accuracy: 0.71
Epoch 4/10
63/63 [==============================] - 532s 8s/step - loss: 0.7821 - accuracy: 0.72
Epoch 5/10
63/63 [==============================] - 535s 9s/step - loss: 0.7435 - accuracy: 0.73
Epoch 6/10
63/63 [==============================] - 536s 9s/step - loss: 0.7904 - accuracy: 0.71
Epoch 7/10
63/63 [==============================] - 538s 9s/step - loss: 0.7072 - accuracy: 0.73
Epoch 8/10
63/63 [==============================] - 541s 9s/step - loss: 0.6159 - accuracy: 0.74
Epoch 9/10
63/63 [==============================] - 535s 9s/step - loss: 0.5917 - accuracy: 0.77
```

```
Epoch 10/10
63/63 [==============================] - 540s 9s/step - loss: 0.5094 - accuracy: 0.79
```

```
testing_evaluation=model.evaluate(test_x,test_y_one_hot)
```

```
7/7 [==============================] - 45s 6s/step - loss: 0.8656 - accuracy: 0.6480
```

```
testing_evaluation #loss,accuracy
```

```
[0.8656001687049866, 0.6479591727256775]
```

```
accuracy=model_train.history['accuracy']
val_accuracy=model_train.history['val_accuracy']
loss=model_train.history['loss']
val_loss=model_train.history['val_loss']
epochs=range(len(accuracy))
```

```
plt.plot(epochs,accuracy,label='training accuracy')
plt.plot(epochs,val_accuracy,label='validation_accuracy')
plt.title('epochs based on accuracy')
plt.legend()
plt.show()
```

```
plt.plot(epochs,loss,label='training accuracy')
plt.plot(epochs,val_loss,label='validation_accuracy')
plt.title('epochs based on accuracy')
plt.legend()
plt.show()
```