PACKAGE IMPORT

```
import os
import cv2
```

```
import numpy as np
```

MOUNTING DRIVE FOR COLLAB NOT REQUIRED FOR OTHER IDE

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/d
```

ACESSING FEATURES AND TARGETS

```
#Change access path of features and target as relevent to our project.
features=[]
target=[]
for x in range(0,3):
  ImagesNamesList=os.listdir("/content/drive/MyDrive/DATA" + "/" + str(x)) #Specify your location
  for y in ImagesNamesList:
    Imgarr=cv2.imread("//content/drive/MyDrive/DATA" + "/" + str(x) + "/" + y) #Specify your location
    #resize for different dimension set using .. inside try:
    try:
      Imgarr=cv2.resize(Imgarr,(100,100))    #Here 100,100 implies dimension
      features.append(Imgarr)
      target.append(x)
    except:       #this is used for exception
      pass
  print("In folder",x)
```

```
In folder 0
In folder 1
In folder 2
```

CONVERTING TO ARRAY AND OTHER FUNCTIONS

```
features=np.array(features)
```

```
target=np.array(target)
```

```
features.shape
```

```
(1500, 100, 100, 3)
```

```
target.shape
```

```
(1500,)
```

```
from sklearn.model_selection import train_test_split
```

```
features_train,features_test,target_train,target_test=train_test_split(features, target,test_size=0.2)
```

```
features_train.shape
```

```
(1200, 100, 100, 3)
```

```
target_train.shape
```

```
(1200,)
```

GRAYSCALING IMAGE FUNCTION

```
#run this without fail
def preprocessing(image):
  image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
  image=image/255
  return image
```

```
features_train=np.array(list(map(preprocessing,features_train)))
```

```
features_train.shape
```

```
(1200, 100, 100)
```

```
features_train=features_train.reshape(1200,100,100,1) #here use the number obtained from above,100,100,1
```

```
features_test=np.array(list(map(preprocessing,features_test)))
```

```
features_test.shape
```

```
(300, 100, 100)
```

```
features_test=features_test.reshape(300,100,100,1) #here use the number obtained from above,100,100,1
```

```
from keras.preprocessing.image import ImageDataGenerator
```

```
dataGen=ImageDataGenerator(rotation_range=10,width_shift_range=0.1,height_shift_range=0.1,zoom_range=0.2,sl
```

```
dataGen.fit(features_train)
```

```
batches=dataGen.flow(features_train,target_train,batch_size=20)
```

```
len(batches)
```

```
60
```

```
images,labels=next(batches)
```

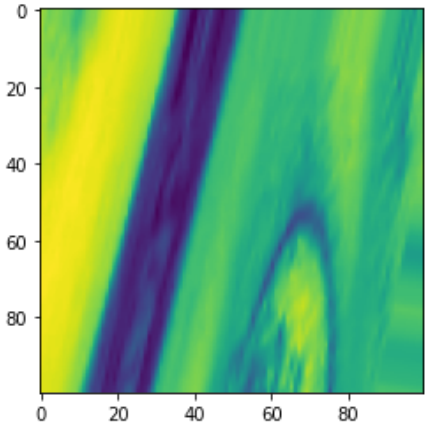
```
images.shape
```

```
(20, 100, 100, 1)
```

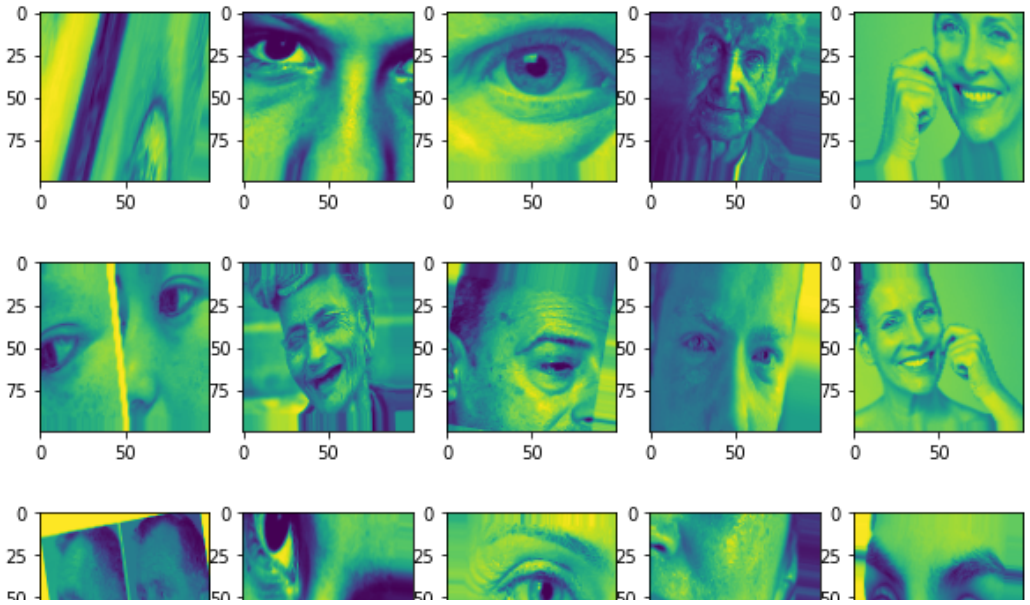
PLOTING AND DISPLAYING SOME DATASET IMAGES

```
import matplotlib.pyplot as plt

#specify the dimension according to the specified dimension as mentioned above
#i.e 100,100 if we resize it to the same
plt.imshow(images[0].reshape(100,100))
plt.show()
```




```
plt.figure(figsize=(10,10))
for i in range(0,20):
    plt.subplot(4,5,i+1)
    #specify the dimension according to the specified dimension as mentioned above
    #i.e instead of 32,32 make it 100,100 if we resize it to the same
    plt.imshow(images[i].reshape(100,100))
plt.show()
```



```
#if u run in jupitor pls change it to "keras.utils import"
from keras.utils.np_utils import to_categorical
```

```
target_train=to_categorical(target_train)
```



```
target_train.shape
```

```
(1200, 3)
```

```
features_train.shape
```

```
(1200, 100, 100, 1)
```

ARCHITECTURE

```
from keras.layers import Dense,Flatten,Conv2D,MaxPooling2D,Dropout
from keras.models import Sequential


model=Sequential()
model.add(Conv2D(60,(3,3),activation="relu",input_shape=(100,100,1)))
model.add(Conv2D(60,(3,3),activation="relu"))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(30,(3,3),activation="relu"))
model.add(Conv2D(30,(3,3),activation="relu"))
model.add(Conv2D(30,(3,3),activation="relu"))
model.add(MaxPooling2D((2,2)))
#Dropout is used to block some neurons on overfitting
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(500,activation="relu"))
model.add(Dense(3,activation="softmax"))

#HERE CHANGE THE "3" ACCORDING TO THE COLUMN SPECIFIED BY TARGET_TRAIN.SHAPE CODE. here it is 3
```

COMPILE

```
from keras.optimizers import Adam


#here "learning_rate" is used as suggested by collab. You can use lr
model.compile(Adam(learning_rate=0.001),loss="categorical_crossentropy",metrics=["accuracy"])
```

TRAIN

```
model.fit_generator(dataGen.flow(features_train,target_train,batch_size=20),epochs=5)
```

```
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py:1915: UserWarning: `Model.fit_generat
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/5
60/60 [==============================] - 93s 2s/step - loss: 0.3174 - accuracy: 0.8817
Epoch 2/5
60/60 [==============================] - 91s 2s/step - loss: 0.2937 - accuracy: 0.8817
Epoch 3/5
60/60 [==============================] - 91s 2s/step - loss: 0.3178 - accuracy: 0.8867
Epoch 4/5
60/60 [==============================] - 92s 2s/step - loss: 0.2766 - accuracy: 0.8925
Epoch 5/5
60/60 [==============================] - 91s 2s/step - loss: 0.2350 - accuracy: 0.9183
<keras.callbacks.History at 0x7f970317de50>
```

SAMPLE PREDICTION

```
predictions = model.predict(features_test)


predictions
```

```
array([[4.08477004e-04, 4.34873611e-01, 5.64717948e-01],
       [6.17167586e-03, 9.81357992e-01, 1.24703990e-02],
       [9.64987695e-01, 3.01394630e-02, 4.87287529e-03],
       [1.55042799e-03, 2.40278859e-02, 9.74421680e-01],
       [9.00386930e-01, 5.97708933e-02, 3.98421325e-02],
       [2.29780690e-05, 9.98713255e-01, 1.26376026e-03],
       [1.40149018e-03, 1.63094103e-01, 8.35504413e-01],
       [1.63186039e-03, 2.29422934e-04, 9.98138666e-01],
       [8.59964430e-01, 1.00415312e-01, 3.96202616e-02],
       [6.75011688e-05, 7.24488199e-02, 9.27483737e-01],
       [5.22588193e-02, 5.82279146e-01, 3.65462005e-01],
       [4.83193621e-03, 2.79781017e-02, 9.67189968e-01],
       [5.47733216e-04, 9.98660445e-01, 7.91769533e-04],
       [3.93451657e-03, 5.37201822e-01, 4.58863676e-01],
       [5.04388452e-01, 4.95180309e-01, 4.31283755e-04],
       [9.43150938e-01, 5.30395433e-02, 3.80957313e-03],
       [9.36694086e-01, 2.43928423e-03, 6.08666465e-02],
       [2.41999328e-03, 9.89685833e-01, 7.89416209e-03],
       [2.15235297e-02, 5.32058766e-03, 9.73155916e-01],
       [1.85816679e-02, 2.67693073e-01, 7.13725269e-01],
       [6.58315480e-01, 6.14710562e-02, 2.80213505e-01],
       [2.96479834e-06, 9.99881744e-01, 1.15219540e-04],
       [7.65641481e-02, 6.51639879e-01, 2.71796048e-01],
       [3.44329071e-03, 1.46720201e-01, 8.49836528e-01],
       [7.22015044e-03, 5.03863215e-01, 4.88916636e-01],
```

```
       [8.98214697e-04, 9.96427715e-01, 2.67410395e-03],
       [9.80117500e-01, 1.19876936e-02, 7.89483637e-03],
       [9.89258173e-04, 9.91647601e-01, 7.36320438e-03],
       [1.94127986e-03, 2.53799200e-01, 7.44259536e-01],
       [1.43403903e-01, 3.01645026e-02, 8.26431572e-01],
       [9.47061300e-01, 2.74234340e-02, 2.55152360e-02],
       [8.41510475e-01, 6.05141409e-02, 9.79754105e-02],
       [9.65162218e-01, 2.76906509e-02, 7.14713708e-03],
       [1.73726847e-04, 9.97262239e-01, 2.56404537e-03],
       [8.81576180e-01, 1.01497926e-01, 1.69258900e-02],
       [1.08792424e-01, 8.83704722e-01, 7.50284456e-03],
       [1.42602473e-01, 8.42393041e-01, 1.50044812e-02],
       [4.40264813e-07, 9.99995828e-01, 3.66183190e-06],
       [1.82205085e-02, 6.05660432e-04, 9.81173754e-01],
       [1.83892474e-01, 5.28210923e-02, 7.63286412e-01],
       [1.19338240e-02, 4.77174670e-01, 5.10891557e-01],
       [6.02964342e-01, 2.83199400e-02, 3.68715733e-01],
       [5.70670724e-01, 2.13453293e-01, 2.15875998e-01],
       [3.28690559e-02, 3.33780888e-03, 9.63793159e-01],
       [7.12353960e-02, 4.04127151e-01, 5.24637461e-01],
       [7.74955690e-01, 1.39611796e-01, 8.54324549e-02],
       [1.61658019e-01, 7.49960065e-01, 8.83818939e-02],
       [1.42675294e-02, 1.96897477e-01, 7.88834929e-01],
       [3.34858567e-01, 4.44571972e-01, 2.20569447e-01],
       [4.22767065e-02, 3.71252179e-01, 5.86471081e-01],
       [9.84051287e-01, 5.06737037e-03, 1.08813914e-02],
       [1.77796162e-03, 3.08964904e-02, 9.67325509e-01],
       [4.57541610e-04, 1.89467400e-01, 8.10075045e-01],
       [1.12545155e-01, 8.77238691e-01, 1.02161206e-02],
       [9.25620496e-01, 3.57373022e-02, 3.86421382e-02],
       [8.65499139e-01, 7.71491462e-03, 1.26785859e-01],
       [8.43582966e-04, 9.95858133e-01, 3.29831173e-03],
       [9.54436779e-01, 3.45393568e-02, 1.10237896e-02],
       [5.37337720e-01, 4.57461238e-01, 5.20106452e-03]
```

Testing a prediction with test set

```
predictions[0]
```

```
array([4.0847700e-04, 4.3487361e-01, 5.6471795e-01], dtype=float32)
```

```
np.argmax(predictions[0])
```

```
2
```

```
target_test[0]  #This means predicted set n test set matches..........
```

```
2
```

THIS IS OPTIONAL TO STORE/SAVE THE MODEL FOR FUTURE USE WITHOUT COMPLILING

```
#THIS IS OPTIONAL
from keras.models import model_from_json


#optional
#converting model to json file to save
model_json=model.to_json()
with open("/content/drive/MyDrive/BLEH/age.json", "w") as abc:  #Specify your location
```

```
  abc.write(model_json)
  abc.close
#We save the weights of the model too in h5 format
model.save_weights("/content/drive/MyDrive/BLEH/ageweights.h5") #Specify your location
print("Save the model")
```

LOAD FROM SAVED MODEL

```
#run this file to directly run it without compiling
json_file=open("/content/drive/MyDrive/BLEH/age.json","r") #Specify your location
loaded_model_json=json_file.read()
json_file.close
loaded_model=model_from_json(loaded_model_json)
loaded_model.load_weights("/content/drive/MyDrive/BLEH/ageweights.h5") #Specify your location
print("loaded the model successfully")
```

```
loaded the model successfully
```

```
#testing if the load is successful
predictions=loaded_model.predict(features_train)
```

```
len(predictions)
```

```
1200
```

LABELING THE TARGET LIST

```
#run this without fail
import numpy as np
import cv2
```

```
def getClassName(classNo):
  if classNo == 0: return 'Darkspots Face'
  elif classNo == 1: return 'Puffy eyes Face'
  elif classNo == 2: return 'Wrinkles Face'
```

PREDICTION USING CAPTURING VIDEO FROM WEBCAM **NOT REQUIRED**

```
capt=cv2.VideoCapture(0)
capt.set(3,640)
capt.set(4,480)
capt.set(10,180)
```

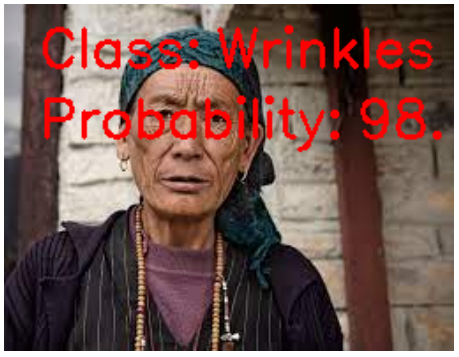
```
while True:
  message,image=capt.read()                  #collect a image from webcam
  imagearr=np.asarray(image)                 #image is converted to array
  imagearr=cv2.resize(imagearr,(100,100))      #resize the array change dimension accordingly
  imagearr=preprocessing(imagearr)           #applying preprocessing technique
  imagearr=imagearr.reshape(1,100,100,1)       #converting 2D image to 4D image cause training data was 4
  predictions=loaded_model.predict(imagearr)  #collection of probablities
  classIndex=loaded_model.predict_classes(imagearr)   #gives the index having highest Probablity value
```

```
  cv2.putText(image,"Class: ",(20,35),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
  cv2.putText(image,"Probability: ",(20,75),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
  probabilityValue=np.amax(predictions)
  if probabilityValue>0.75:
    cv2.putText(image,getClassName(classIndex),(120 , 35),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
    cv2.putText(image, str(probabilityValue * 100) + " %", (160,75), cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),
  cv2.imshow("Model Prediction", image)
```

PEDICTION USING INPUT IMAGE

```
from google.colab.patches import cv2_imshow #for collab


  image=cv2.imread('/content/drive/MyDrive/BLEH/Wrinkles_Sample1.jpg', -1)
  imagearr=np.asarray(image)                  #image is converted to array
  imagearr=cv2.resize(imagearr,(100,100))       #resize the array change dimension accordingly
  imagearr=preprocessing(imagearr)            #applying preprocessing technique
  imagearr=imagearr.reshape(1,100,100,1)        #converting 2D image to 4D image cause training data was 4
  predictions=loaded_model.predict(imagearr)  #collection of probablities
  classIndex=np.argmax(loaded_model.predict(imagearr), axis=-1) #this is for collab .. If any other IDE us
  #classIndex=loaded_model.predict_classes(imagearr)   #gives the index having highest Probablity value
  cv2.putText(image,"Class: ",(20,35),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
  cv2.putText(image,"Probability: ",(20,75),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
  probabilityValue=np.amax(predictions)
  if probabilityValue>0.75:
    cv2.putText(image,getClassName(classIndex),(120 , 35),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
    cv2.putText(image, str(probabilityValue * 100) + " %", (200,75), cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),
  cv2_imshow(image) #USE cv2.imshow() in normal IDE
```



✓ 0s completed at 1:06 PM