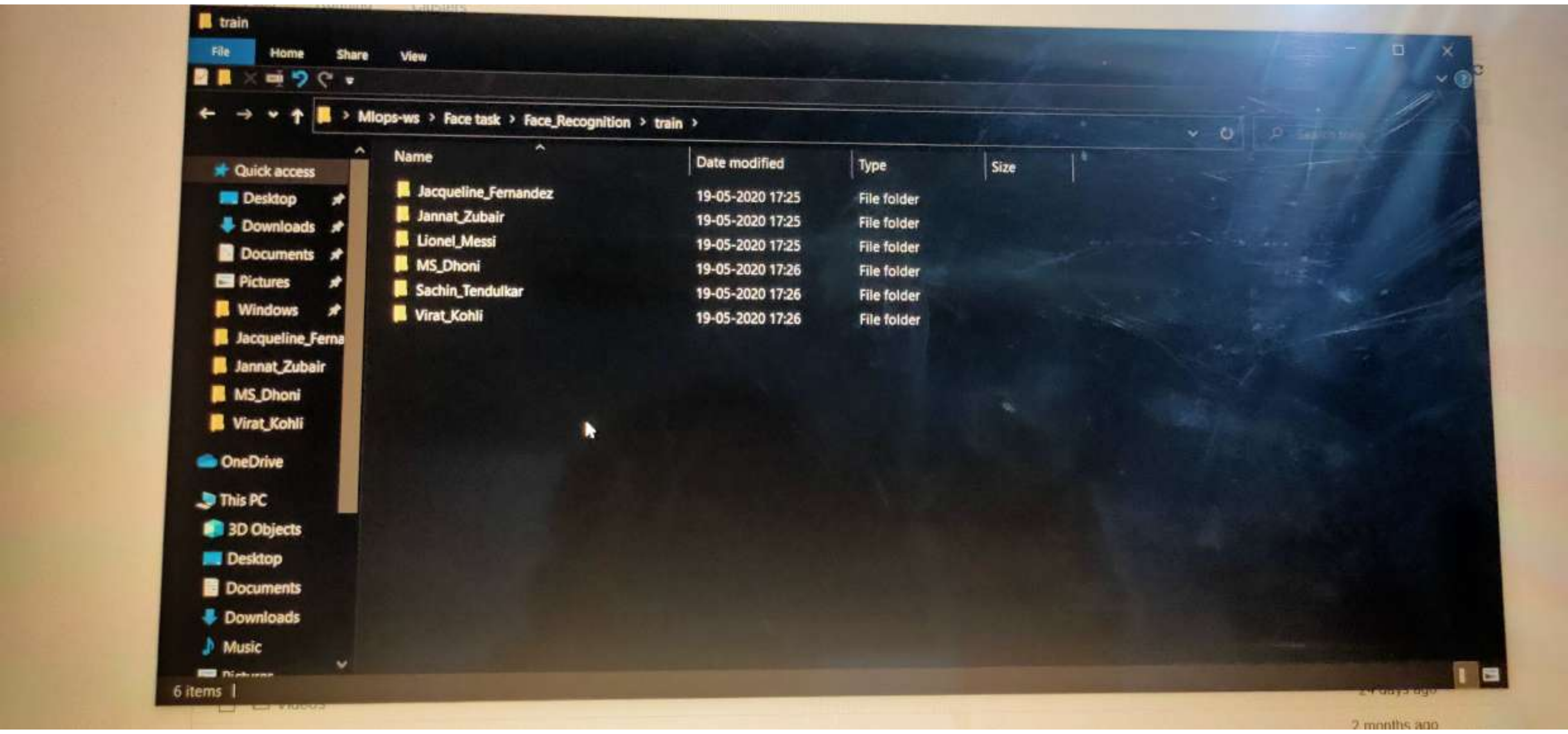
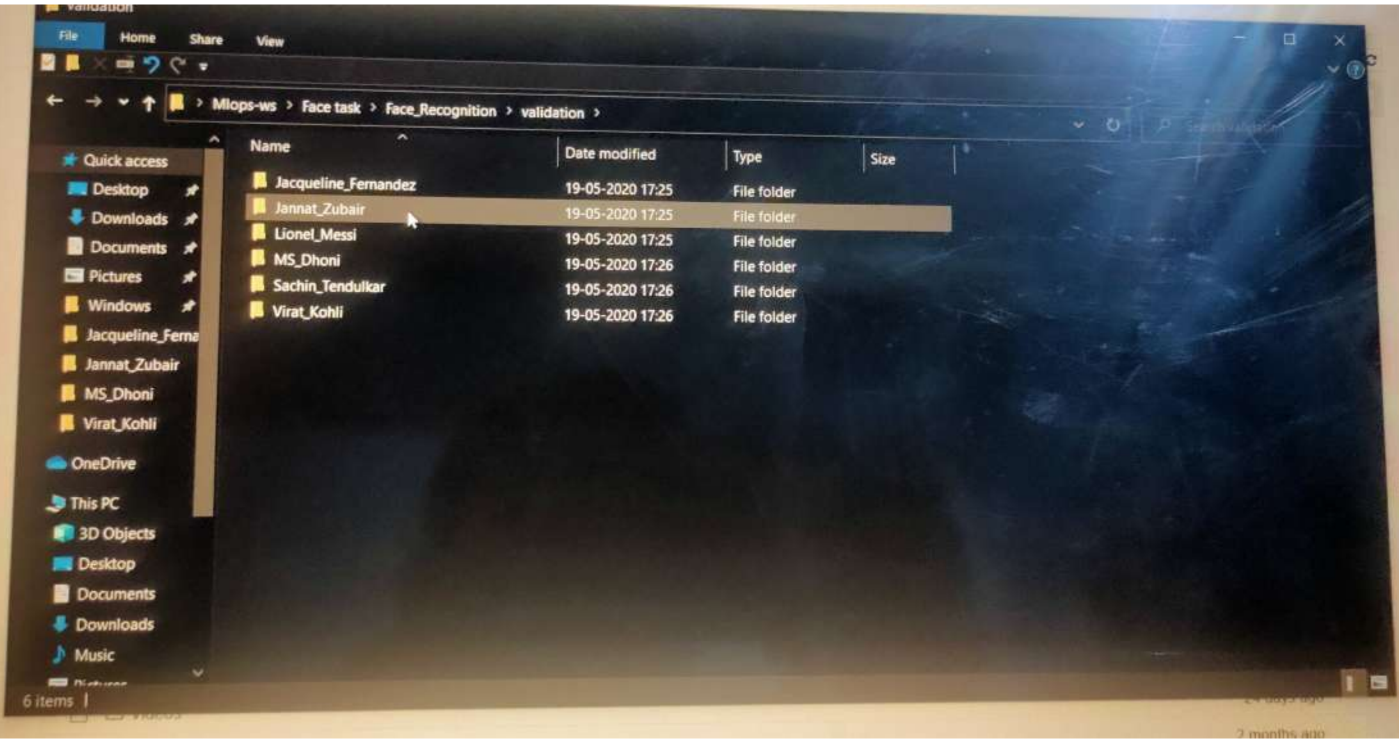


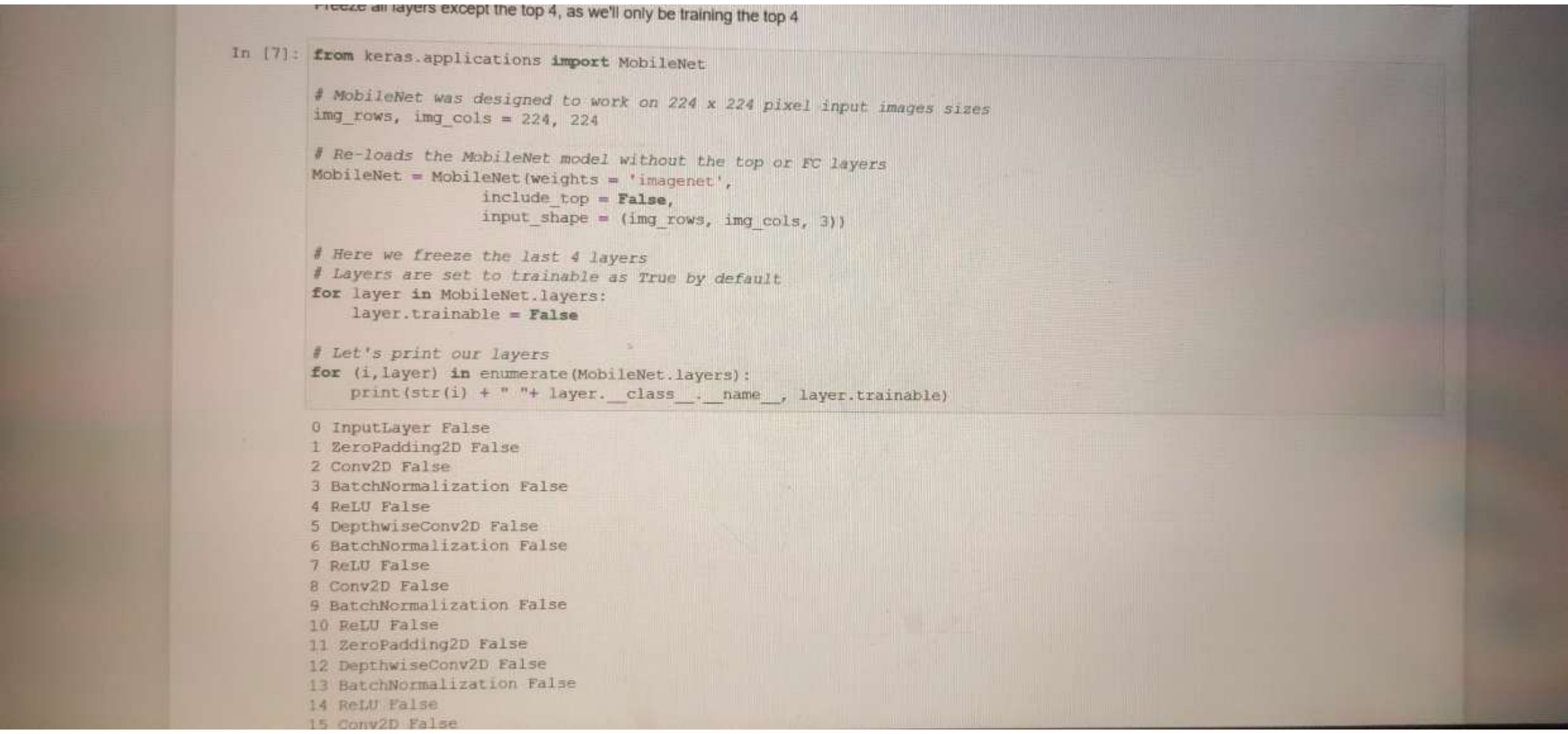
Mlops-ws › Face task › Face_Recognition › train ›

Name	Date modified	Type	Size
Jacqueline_Fernandez	19-05-2020 17:25	File folder	
Jannat_Zubair	19-05-2020 17:25	File folder	
Lionel_Messi	19-05-2020 17:25	File folder	
MS_Dhoni	19-05-2020 17:26	File folder	
Sachin_Tendulkar	19-05-2020 17:26	File folder	
Virat_Kohli	19-05-2020 17:26	File folder	

6 items



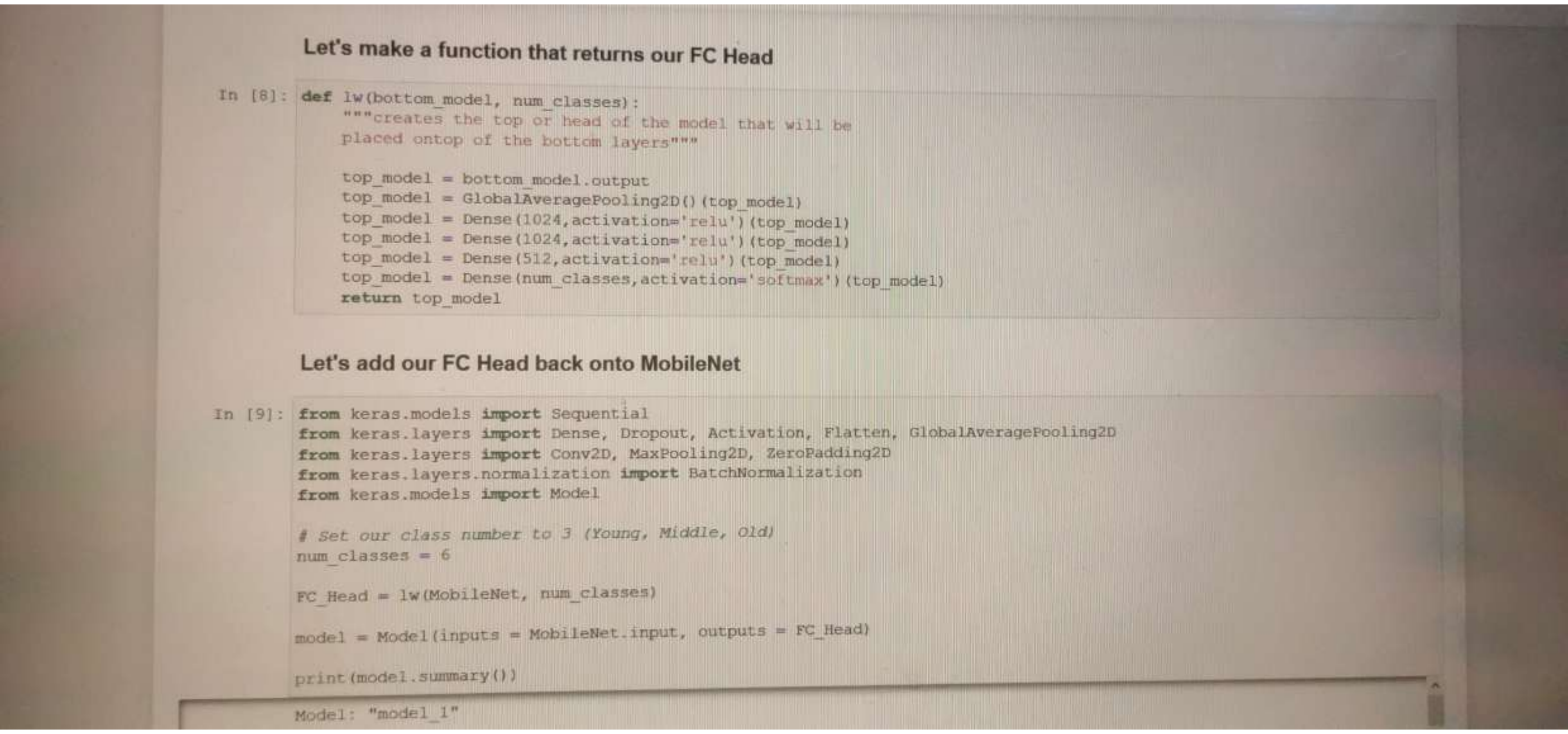
Name	Date modified	Type	Size
Jacqueline_Fernandez	19-05-2020 17:25	File folder	
Jannat_Zubair	19-05-2020 17:25	File folder	
Lionel_Messi	19-05-2020 17:25	File folder	
MS_Dhoni	19-05-2020 17:26	File folder	
Sachin_Tendulkar	19-05-2020 17:26	File folder	
Virat_Kohli	19-05-2020 17:26	File folder	



Freeze all layers except the top 4, as we'll only be training the top 4

In [7]:
```
from keras.applications import MobileNet

# MobileNet was designed to work on 224 x 224 pixel input images sizes
img_rows, img_cols = 224, 224

# Re-loads the MobileNet model without the top or FC layers
MobileNet = MobileNet(weights = 'imagenet',
                 include_top = False,
                 input_shape = (img_rows, img_cols, 3))

# Here we freeze the last 4 layers
# Layers are set to trainable as True by default
for layer in MobileNet.layers:
    layer.trainable = False

# Let's print our layers
for (i,layer) in enumerate(MobileNet.layers):
    print(str(i) + " "+ layer.__class__.__name__, layer.trainable)
```

```
0 InputLayer False
1 ZeroPadding2D False
2 Conv2D False
3 BatchNormalization False
4 ReLU False
5 DepthwiseConv2D False
6 BatchNormalization False
7 ReLU False
8 Conv2D False
9 BatchNormalization False
10 ReLU False
11 ZeroPadding2D False
12 DepthwiseConv2D False
13 BatchNormalization False
14 ReLU False
15 Conv2D False
```



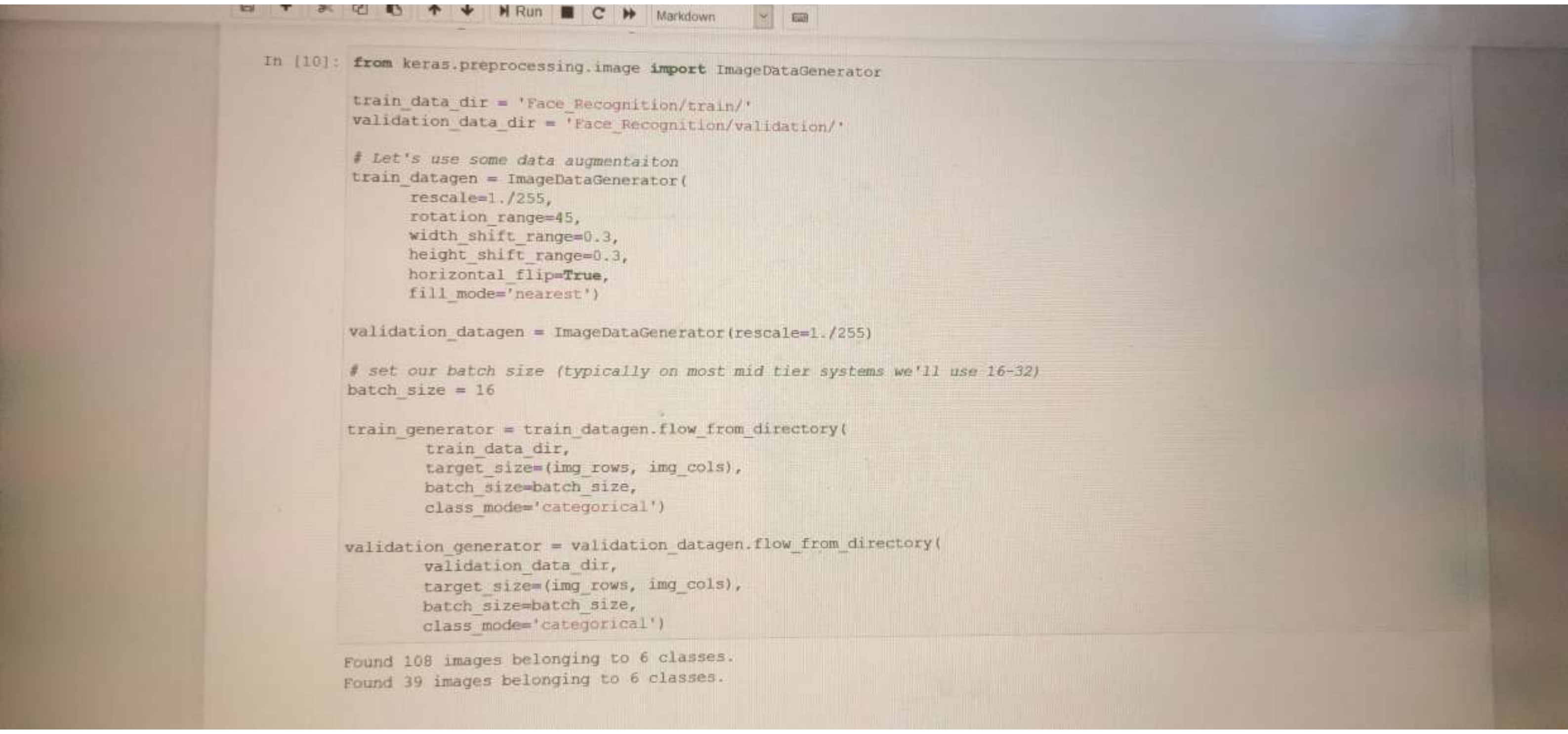
Let's make a function that returns our FC Head

```
In [8]: def lw(bottom_model, num_classes):
            """creates the top or head of the model that will be
            placed ontop of the bottom layers"""

            top_model = bottom_model.output
            top_model = GlobalAveragePooling2D()(top_model)
            top_model = Dense(1024,activation='relu')(top_model)
            top_model = Dense(1024,activation='relu')(top_model)
            top_model = Dense(512,activation='relu')(top_model)
            top_model = Dense(num_classes,activation='softmax')(top_model)
            return top_model
```

Let's add our FC Head back onto MobileNet

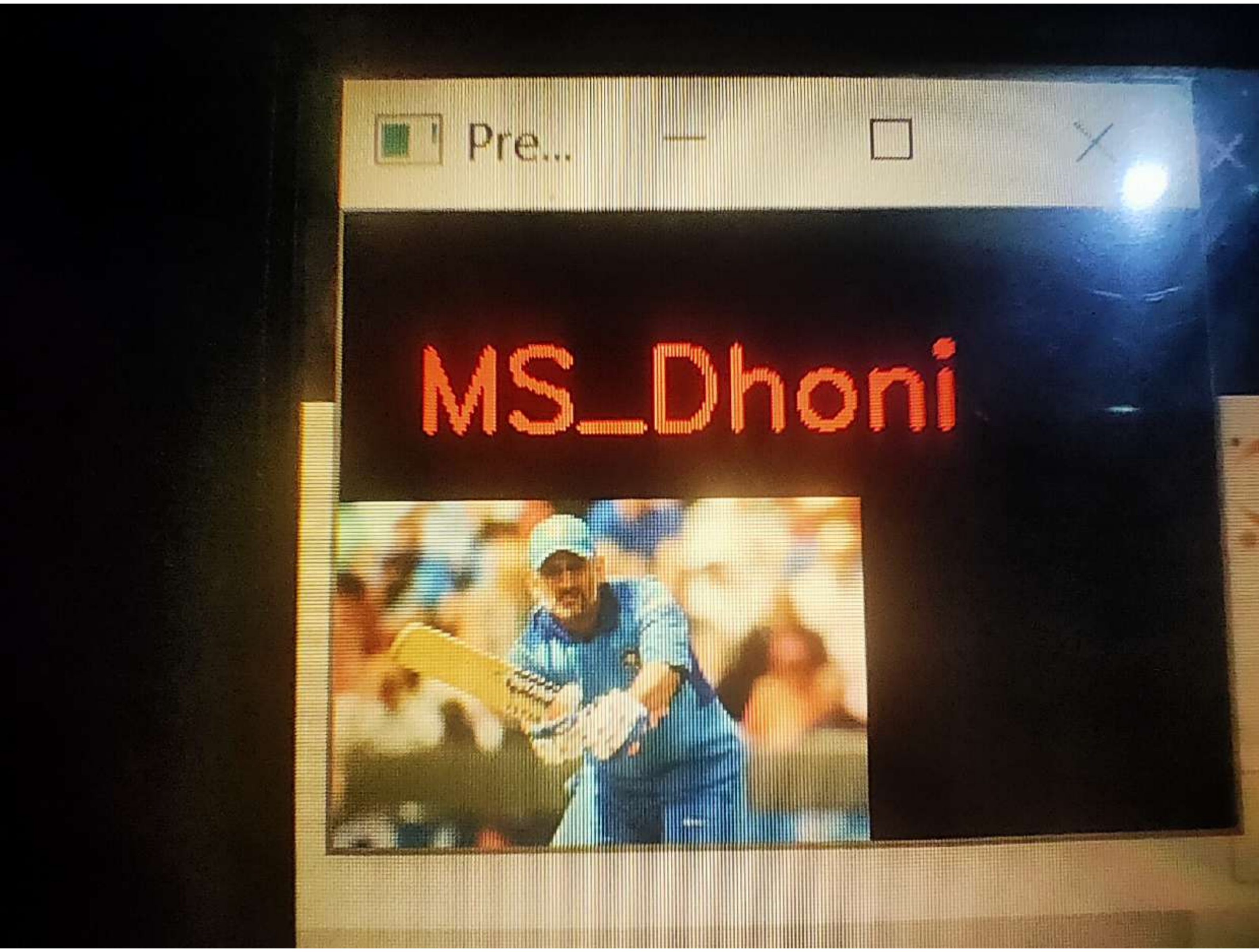
```
In [9]: from keras.models import Sequential
        from keras.layers import Dense, Dropout, Activation, Flatten, GlobalAveragePooling2D
        from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
        from keras.layers.normalization import BatchNormalization
        from keras.models import Model

        # Set our class number to 3 (Young, Middle, Old)
        num_classes = 6

        FC_Head = lw(MobileNet, num_classes)

        model = Model(inputs = MobileNet.input, outputs = FC_Head)

        print(model.summary())
```

```
        Model: "model_1"
```

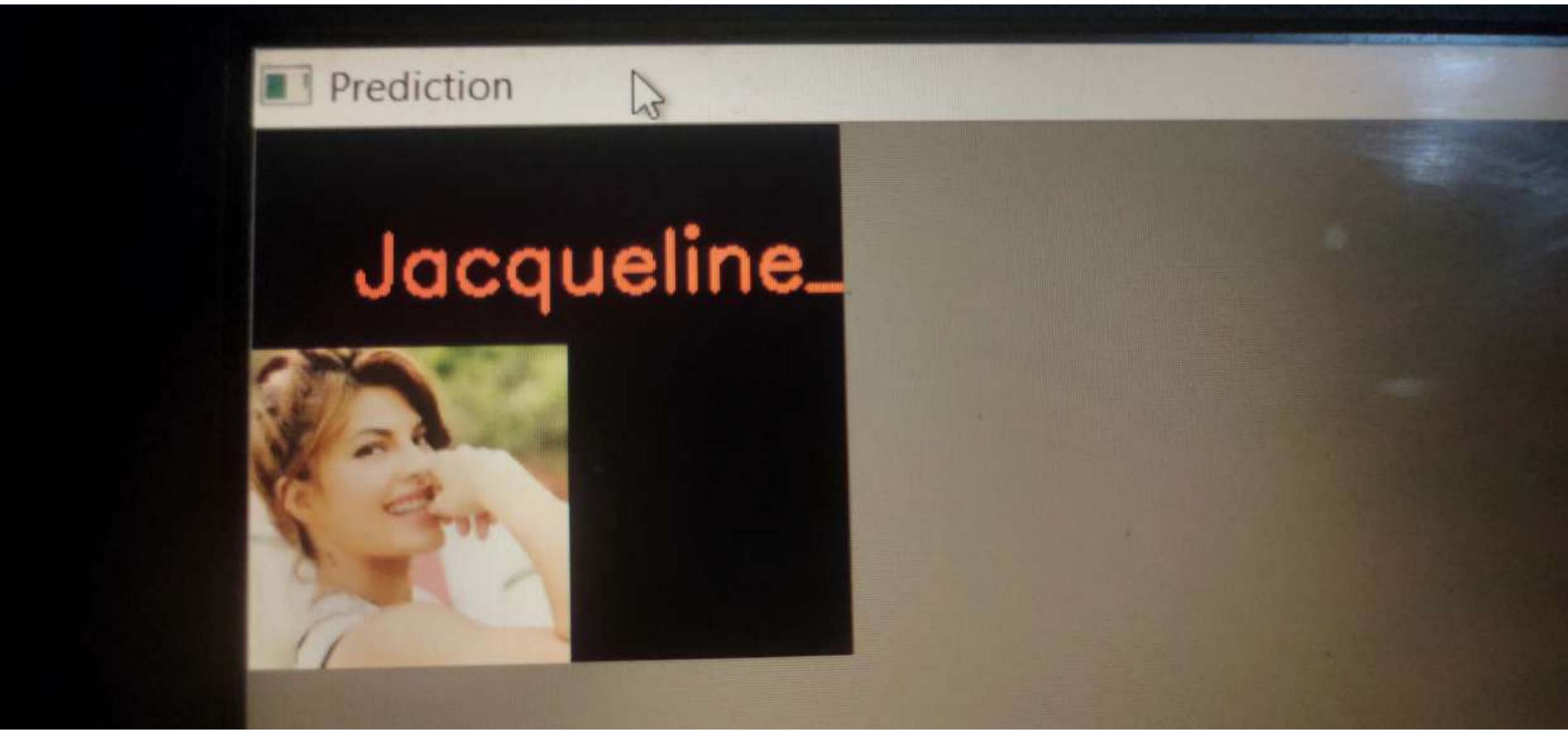


```
In [10]: from keras.preprocessing.image import ImageDataGenerator

         train_data_dir = 'Face_Recognition/train/'
         validation_data_dir = 'Face_Recognition/validation/'

         # Let's use some data augmentaiton
         train_datagen = ImageDataGenerator(
               rescale=1./255,
               rotation_range=45,
               width_shift_range=0.3,
               height_shift_range=0.3,
               horizontal_flip=True,
               fill_mode='nearest')

         validation_datagen = ImageDataGenerator(rescale=1./255)

         # set our batch size (typically on most mid tier systems we'll use 16-32)
         batch_size = 16

         train_generator = train_datagen.flow_from_directory(
                 train_data_dir,
                 target_size=(img_rows, img_cols),
                 batch_size=batch_size,
                 class_mode='categorical')

         validation_generator = validation_datagen.flow_from_directory(
                 validation_data_dir,
                 target_size=(img_rows, img_cols),
                 batch_size=batch_size,
                 class_mode='categorical')
```

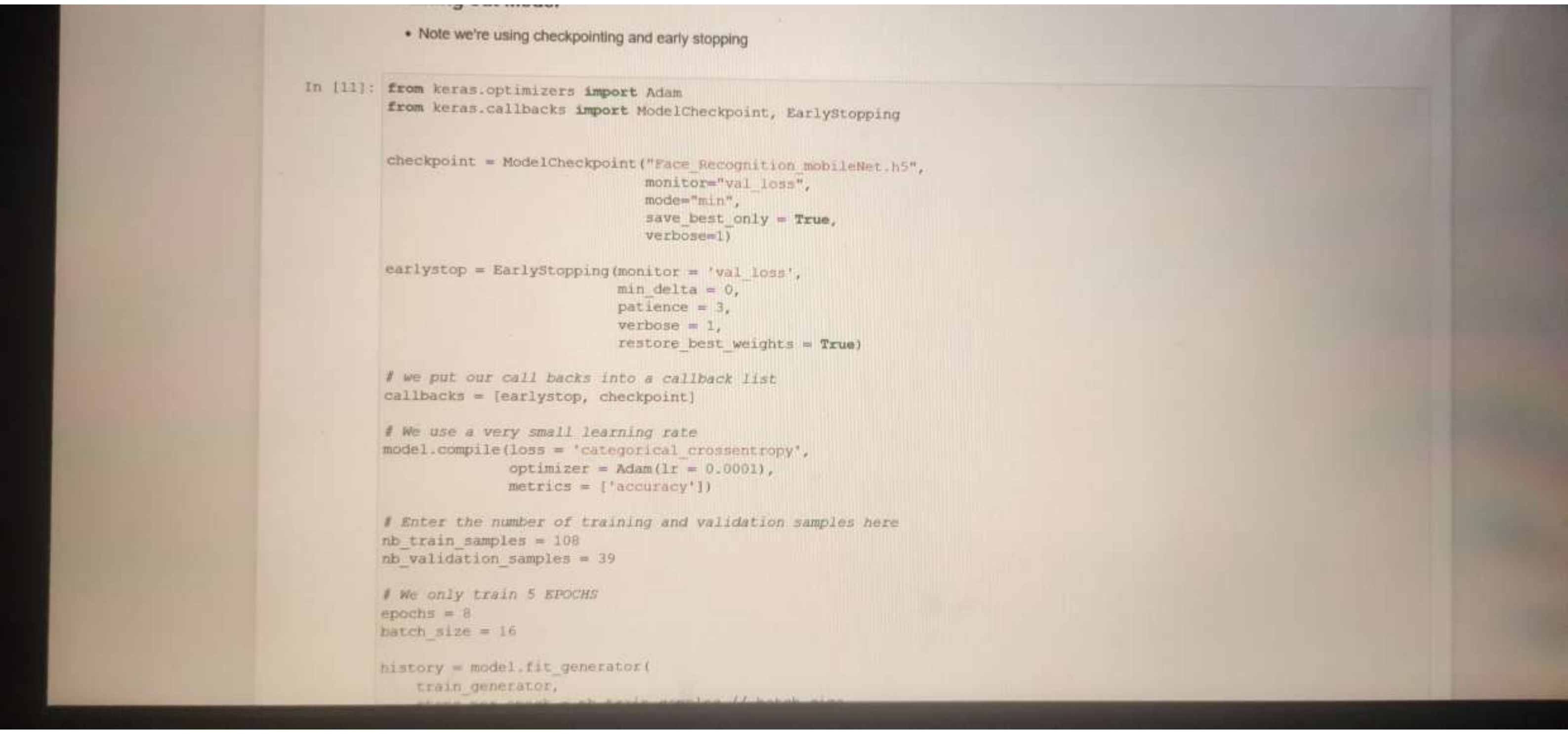
```
Found 108 images belonging to 6 classes.
Found 39 images belonging to 6 classes.
```



MS_Dhoni



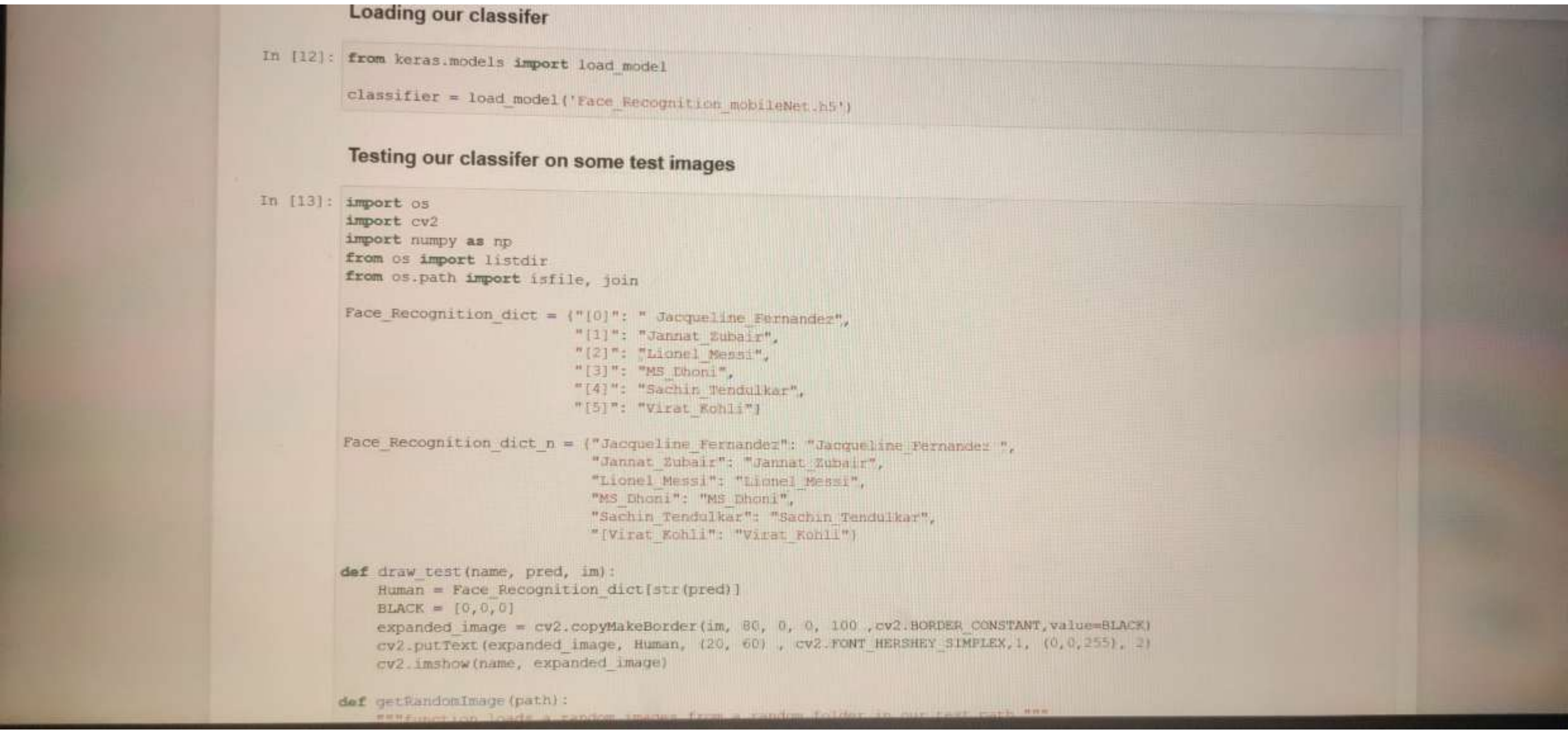
Jacqueline_



- Note we're using checkpointing and early stopping

In [11]:

```
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping


checkpoint = ModelCheckpoint("Face_Recognition_mobileNet.h5",
                             monitor="val_loss",
                             mode="min",
                             save_best_only = True,
                             verbose=1)

earlystop = EarlyStopping(monitor = 'val_loss',
                          min_delta = 0,
                          patience = 3,
                          verbose = 1,
                          restore_best_weights = True)

# we put our call backs into a callback list
callbacks = [earlystop, checkpoint]

# We use a very small learning rate
model.compile(loss = 'categorical_crossentropy',
              optimizer = Adam(lr = 0.0001),
              metrics = ['accuracy'])

# Enter the number of training and validation samples here
nb_train_samples = 108
nb_validation_samples = 39

# We only train 5 EPOCHS
epochs = 8
batch_size = 16

history = model.fit_generator(
    train_generator,
```

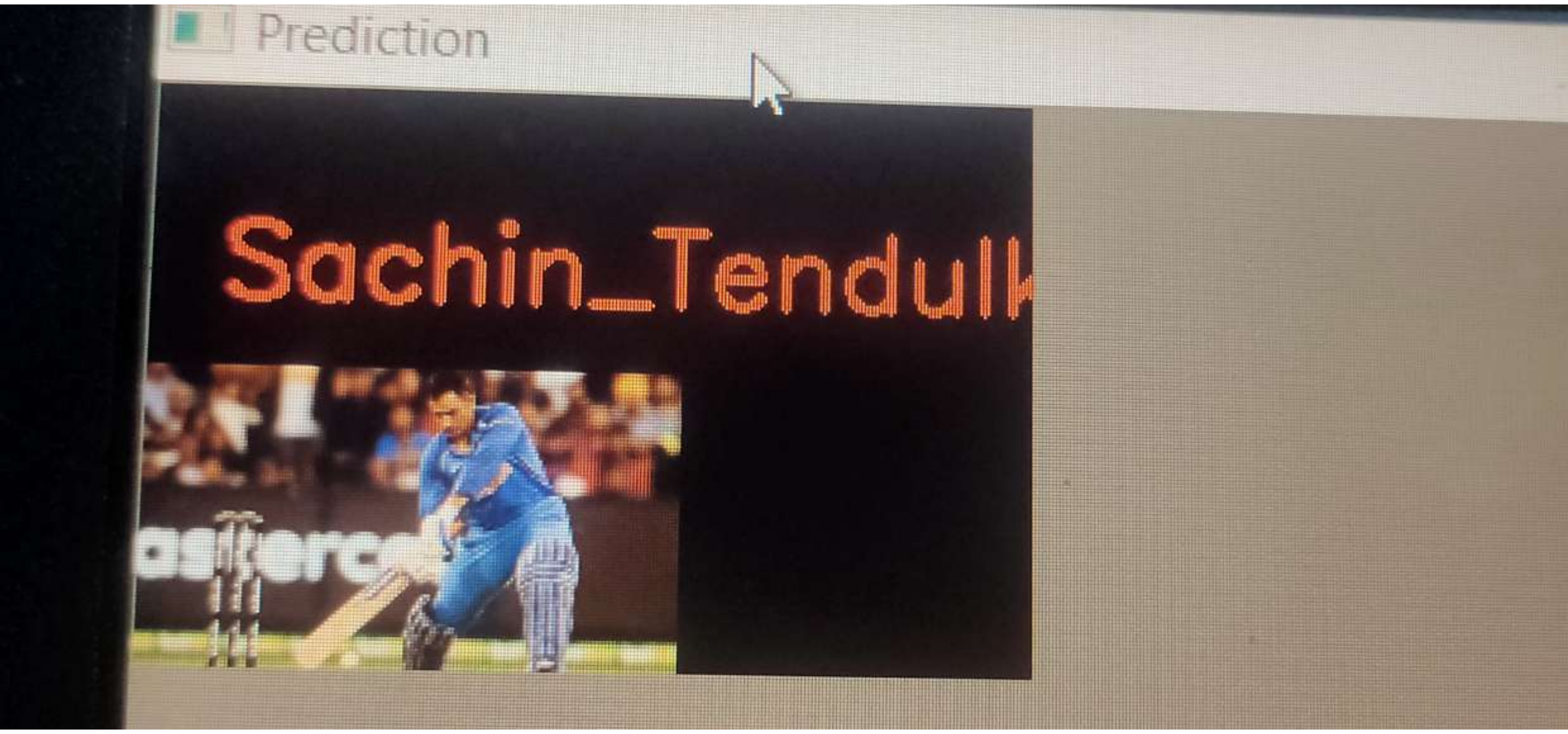


Loading our classifer

```
In [12]: from keras.models import load_model

         classifier = load_model('Face_Recognition_mobileNet.h5')
```

Testing our classifer on some test images

```
In [13]: import os
         import cv2
         import numpy as np
         from os import listdir
         from os.path import isfile, join

         Face_Recognition_dict = {"[0]": " Jacqueline_Fernandez",
                                  "[1]": "Jannat_Zubair",
                                  "[2]": "Lionel_Messi",
                                  "[3]": "MS_Dhoni",
                                  "[4]": "Sachin_Tendulkar",
                                  "[5]": "Virat_Kohli"}

         Face_Recognition_dict_n = {"Jacqueline_Fernandez": "Jacqueline_Fernandez ",
                                    "Jannat_Zubair": "Jannat_Zubair",
                                    "Lionel_Messi": "Lionel_Messi",
                                    "MS_Dhoni": "MS_Dhoni",
                                    "Sachin_Tendulkar": "Sachin_Tendulkar",
                                    "[Virat_Kohli": "Virat_Kohli"}

         def draw_test(name, pred, im):
             Human = Face_Recognition_dict[str(pred)]
             BLACK = [0,0,0]
             expanded_image = cv2.copyMakeBorder(im, 80, 0, 0, 100 ,cv2.BORDER_CONSTANT,value=BLACK)
             cv2.putText(expanded_image, Human, (20, 60) , cv2.FONT_HERSHEY_SIMPLEX,1, (0,0,255), 2)
             cv2.imshow(name, expanded_image)

         def getRandomImage(path):
             """function loads a random images from a random folder in our test path """
```



Sachin_Tendulk