Working of the Project

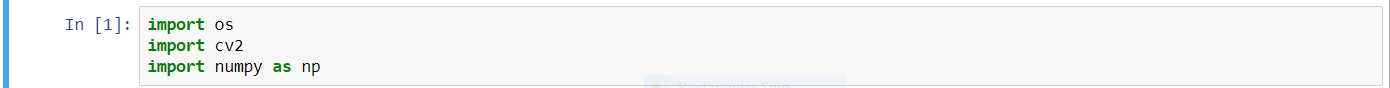


Fig 1: Importing all the modules

Import os module

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open (), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the file input module. For creating temporary files and directories see the temp file module, and for high-level file and directory handling see the shutil module.

Import cv2 module

OpenCV is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

* **core** - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
* **imgproc** - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
* **video** - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
* **calib3d** - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
* **features2d** - salient feature detectors, descriptors, and descriptor matchers.
* **objectdetect** - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
* **highgui** - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.
* **gpu** - GPU-accelerated algorithms from different OpenCV modules.

Import NumPy module

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

* a powerful N-dimensional array object
* sophisticated (broadcasting) functions
* tools for integrating C/C++ and Fortran code
* useful linear algebra, Fourier transform, and random number capabilities



Fig 2: Resizing the image

This is used to set the dimensions that is 50 height and 50 width of the image.

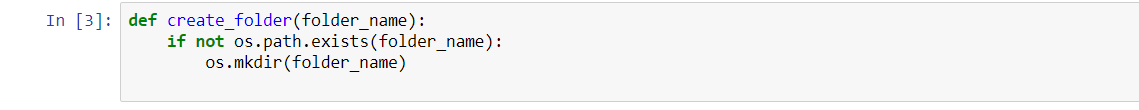


Fig 3: Creating the folder in OS

This is a function that is used to create the folder if the folder is not there in the current directory.

Function - **os.path.exists()** method in Python is used to check whether the specified path exists or not. This method can be also used to check whether the given path refers to an open file descriptor or not.

**os.mkdir()** method in Python is used to create a directory named path with the specified numeric mode. This method raise FileExistsError if the directory to be created already exists.

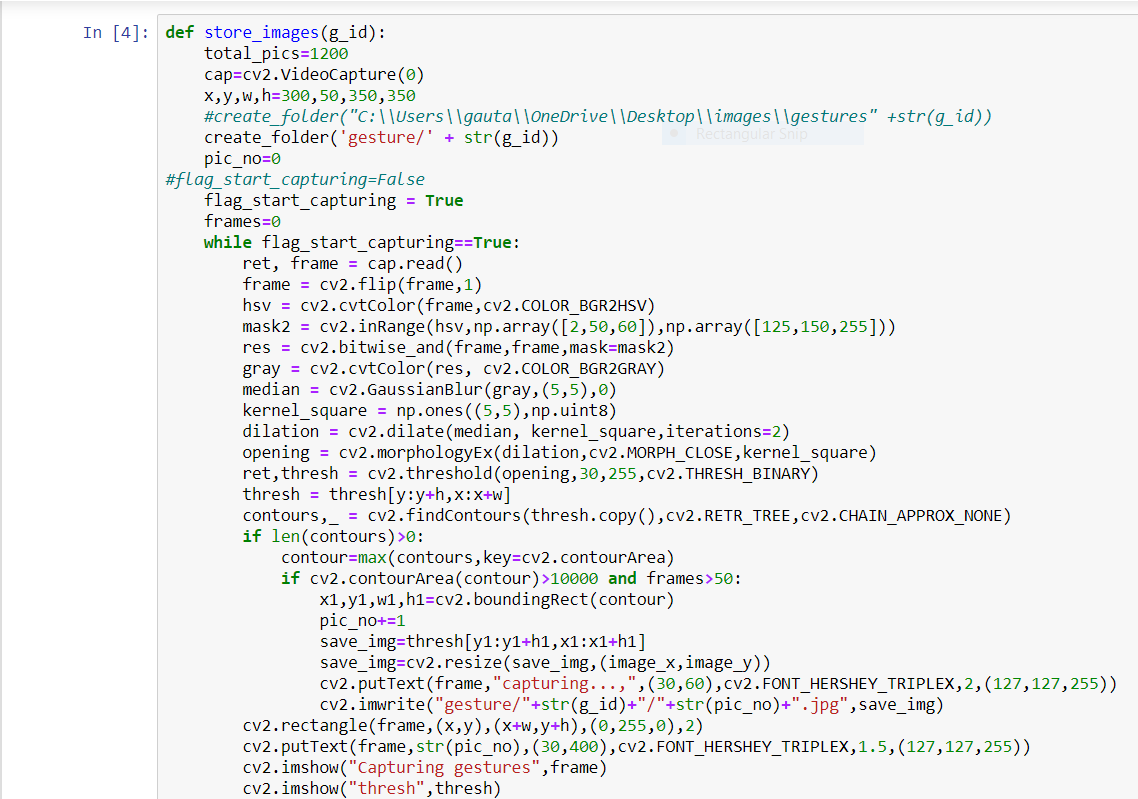


Fig 4: Basic Image processing operations using opencv

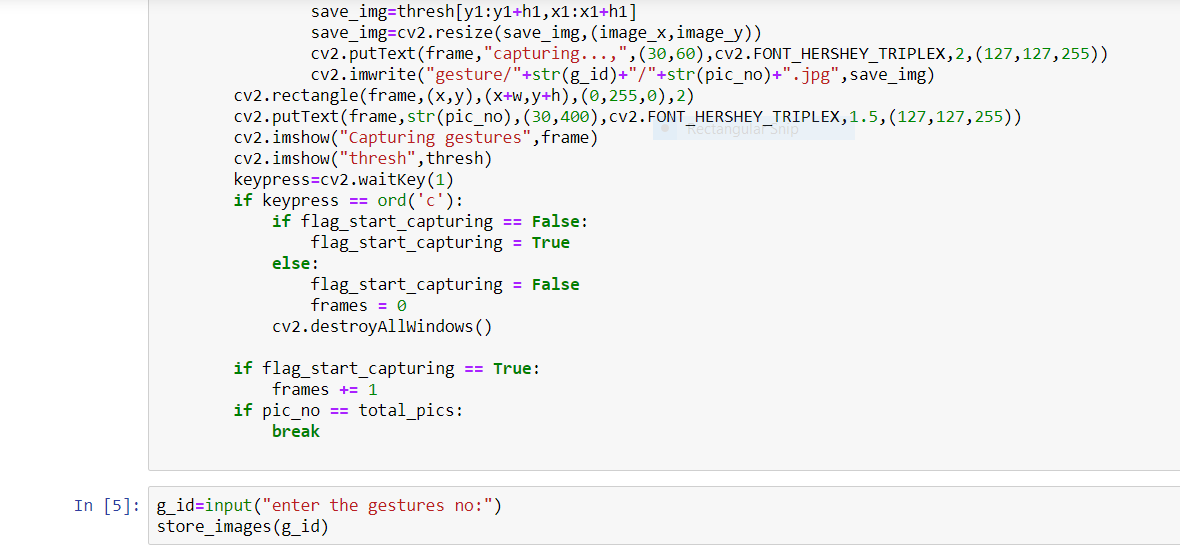


Fig 5: Storing all the images after processing

1.Def store images – function is used to store the images with the g\_id in the same directory folder that is created with the help of create\_folder function. So g\_id will be stored inside the folder.

2.cap=cv2.VideoCapture(0) - To capture a video, you need to create a **VideoCapture** object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected (as in my case). So, I simply pass 0 (or -1). You can select the second camera by passing 1 and so on.

3.create\_folder ('gesture/' + str(g\_id))- This will create a folder with the g\_id passed I the same directory.

4. cap.read() returns a bool (True/False). If frame is read correctly, it will be True. So you can check end of the video by checking this return value.

5. frame = cv2.flip(frame,1) this is used to flip image capturing left right with respect to frame, since when we place our hand our hand direction is captured in opposite direction so flip function is used to flip it to left and right.

6. hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)- This convert BGR image to HSV, we can use this to extract a colored object. In HSV, it is easier to represent a colour than RGB color-space.

7. mask2 = cv2.inRange(hsv,np.array([2,50,60]),np.array([125,150,255])) This is used for masking only the color in range that is specified in array. Since this will only capture the color in range with our skin.

8. res = cv2.bitwise\_and(frame,frame,mask=mask2) This is used in bitwising and with the frame and the mask.

9. gray = cv2.cvtColor(res, cv2.COLOR\_BGR2GRAY)- For BGR \rightarrow Gray conversion we use the flags cv2.COLOR\_BGR2GRAY

10. median = cv2.GaussianBlur(gray,(5,5),0)- In this, instead of box filter, gaussian kernel is used. It is done with the function, **cv.GaussianBlur()**. We should specify the width and height of kernel which should be positive and odd. We also should specify the standard deviation in X and Y direction, sigmaX and sigmaY respectively. If only sigmaX is specified, sigmaY is taken as same as sigmaX. If both are given as zeros, they are calculated from kernel size. Gaussian blurring is highly effective in removing gaussian noise from the image.

11. dilation = cv2.dilate(median, kernel\_square,iterations=2) -This is used in processing the image , this convulate the brighter region . It causes the brighter region to grow.

12. opening = cv2.morphologyEx(dilation,cv2.MORPH\_CLOSE,kernel\_square)- The morphologyEx takes two parameters that is cv2.MORPH\_OPEN, cv2.MORPH\_CLOSE. Since cv2.MORPH\_CLOSE is dialation followed by erosion.

13. ret,thresh = cv2.threshold(opening,30,255,cv2.THRESH\_BINARY)- If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The function used is **cv2.threshold**. First argument is the source image, which **should be a grayscale image**. Second argument is the threshold value which is used to classify the pixel values. Third argument is the maxVal which represents the value to be given if pixel value is more than (sometimes less than) the threshold value. OpenCV provides different styles of thresholding and it is decided by the fourth parameter of the function.

14. contours,\_ =cv2.findContours(thresh.copy(),cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_NONE)

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same colour or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

* For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection.
* findContours function modifies the source image. So if you want source image even after finding contours, already store it to some other variables.

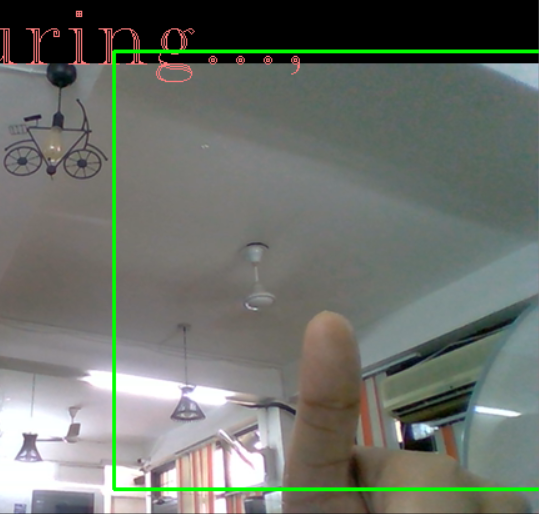


Figure 6: Capturing the image



Figure 7: Storing the captured image

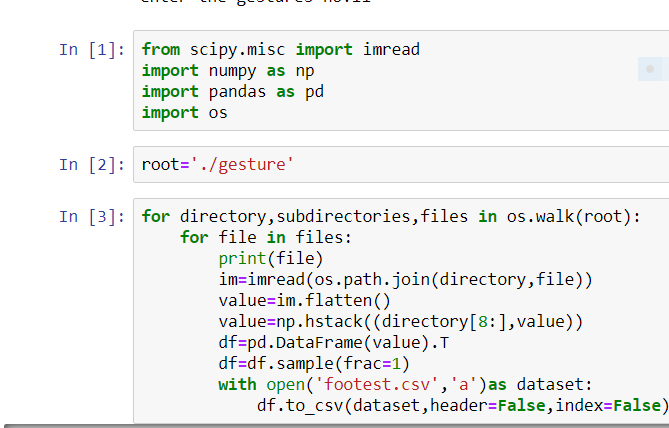


Fig 8: Creating the csv file containing pixels of image

Imread is used to read images and os.walk generate the file names in a directory tree by walking the tree either top-down or bottom-up. Root prints out the directories in the same path.

value=im.flatten() – This is used to convert the n-d array into one-d array and then return the original array and if we modify any value of an array, the original array is not affected.

value=np.hstack((director[8:],value))- Stack arrays in sequence horizontally (column wise).

This is equivalent to concatenation along the second axis, except for 1-D arrays where it concatenates along the first axis. Rebuilds arrays divided by hsplit.

This function makes most sense for arrays with up to 3 dimensions. For instance, for pixel-data with a height (first axis), width (second axis), and r/g/b channels (third axis). The functions concatenate, stack and block provide more general stacking and concatenation operations.

df=pd.DataFrame(value).T - DataFrame.Transpose index and columns. Reflect the Data Frame over its main diagonal by writing rows as columns and vice-versa. The property [**T**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.T.html#pandas.DataFrame.T) is an accessor to the method transpose().

df=df.sample(frac=1)- This is used to return the random values of the sample. This return random item of samples from an axis of objects.

with open('footest.csv','a’) as dataset: - This is used to append the data in the csv file and the append the data in the csv file.

df.to\_csv(dataset,header=False,index=False)- Save a Pandas Data Frame as a CSV file using to\_csv() method.

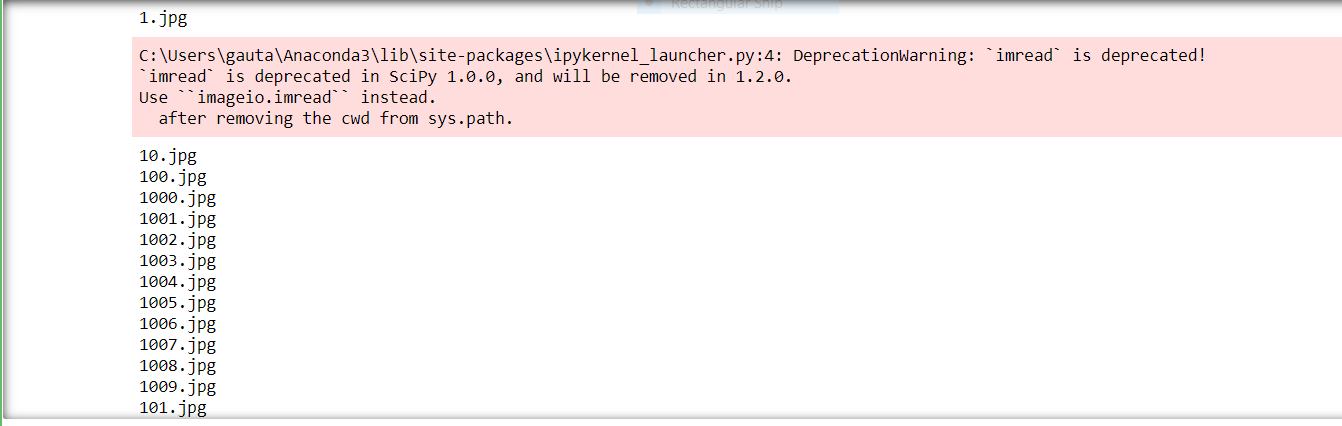


Fig 9: Printing the images as datasets



Fig 10: Optimizing the size of datasets

We have used the keras and TensorFlow as backend for developing convolution neural network. The data is taken from the csv file and reshaped and divided by 255 so that we can get all pixels in the range of 0 and 1.

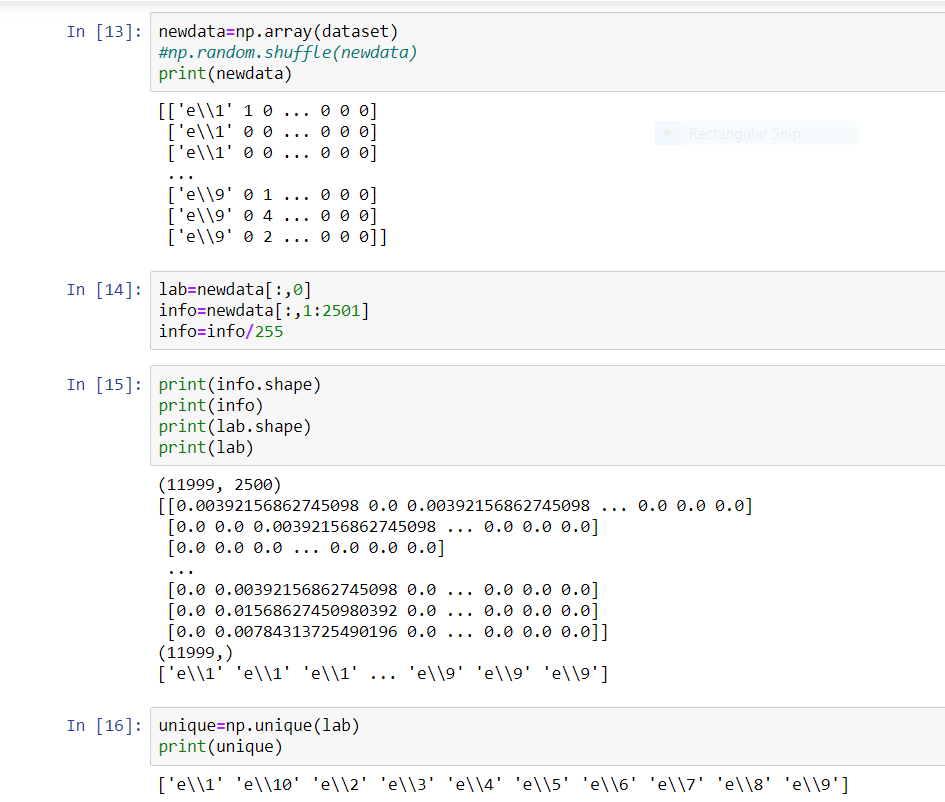


Fig 11: Processing on datasets

After that when the data is taken and reshaped the folders labels are in the string form since we cannot perform one hot encoding in string data so we have to convert into integer with the help of dictionary

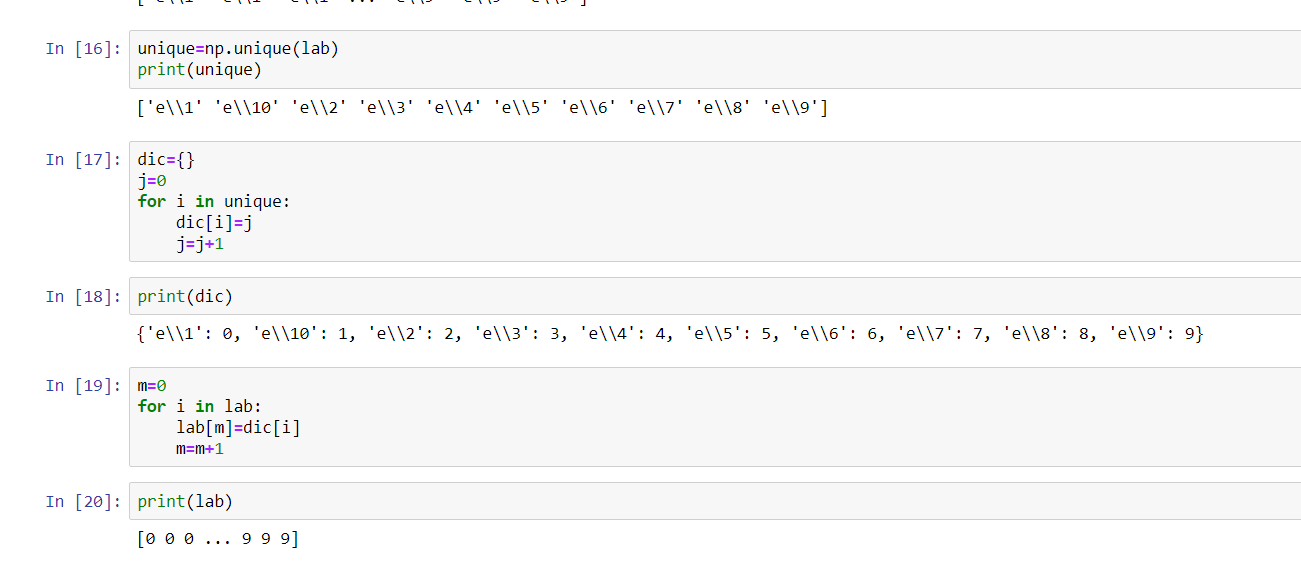


Fig 12: Encoding the datasets

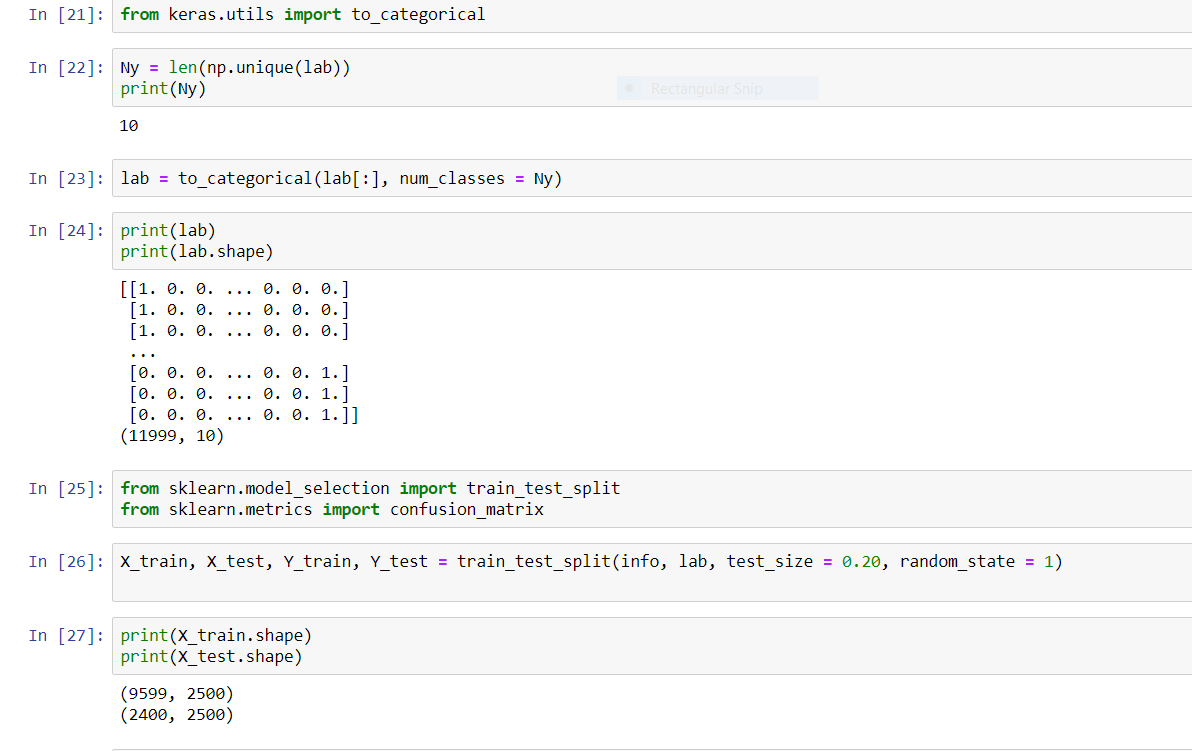


Fig 13: Splitting the datasets into train and test sets

This is importing the categorical and perform one hot encoding on the labels datasets and then splitting the datasets into test and train datasets with the help of sklearn train\_test\_split.

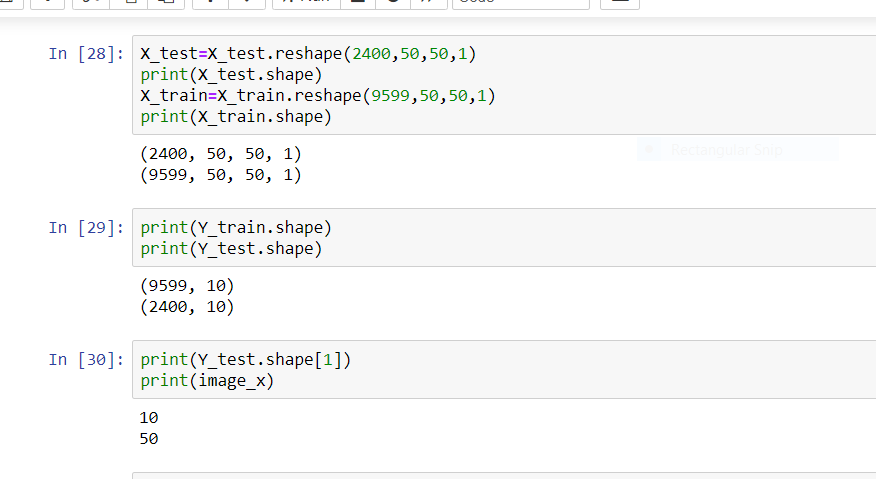


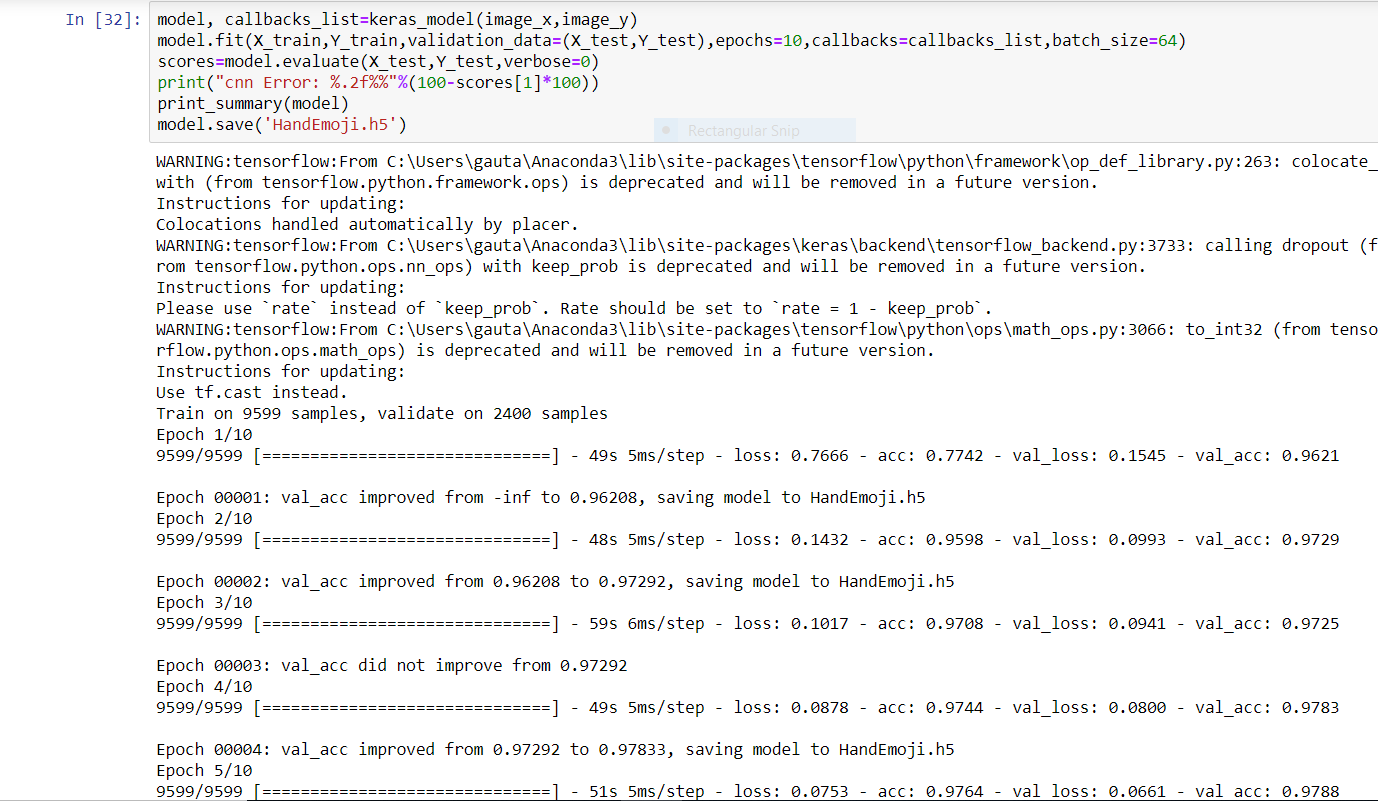
Fig 14: Reshaping and Printing of train and test sets

This is reshaping of test and train datasets. Since we cannot pass these values directly we have to reshape in the filters



Fig 15: Creating the Keras model

This is keras model that is convolution neural network. The convolution neural network consists of hidden layers, input layers and output layers. It acts as a filter and in this process, the convolution conv2D layers will act as a input layer and maxpolling layers act as a finding the maximum filters characteristics. While the flattening is done to convert all of them as a flatten array. Since densening and categorical cross entropy is used to calculate the losses and after this it get stored as handemoji.h5 model.



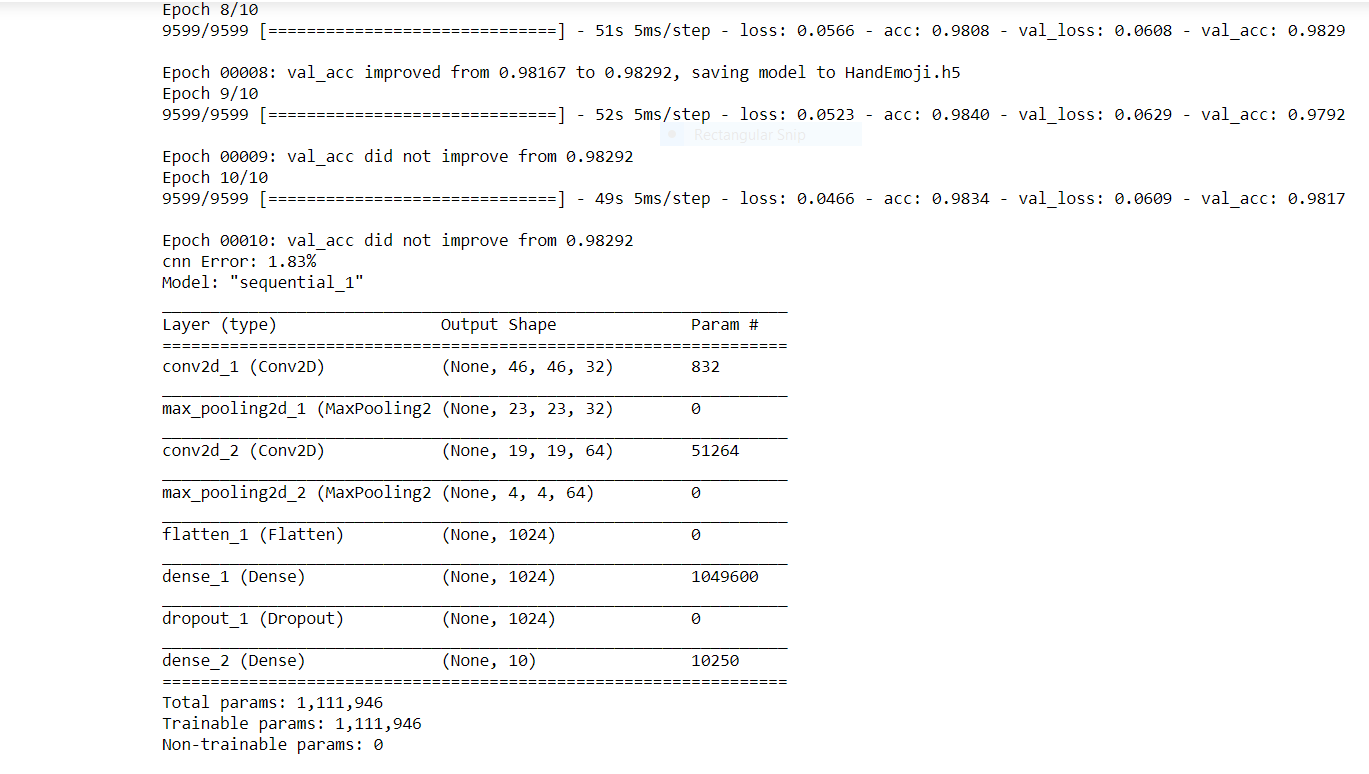


Fig 16: Printing the model summary and model layers

After this the model is trained for testing, we have supplied the different hand gestures emojis in a folder and test and comparing it with the model and when the video camera is on then it should predict the correct emojis similar to our hand on opening.

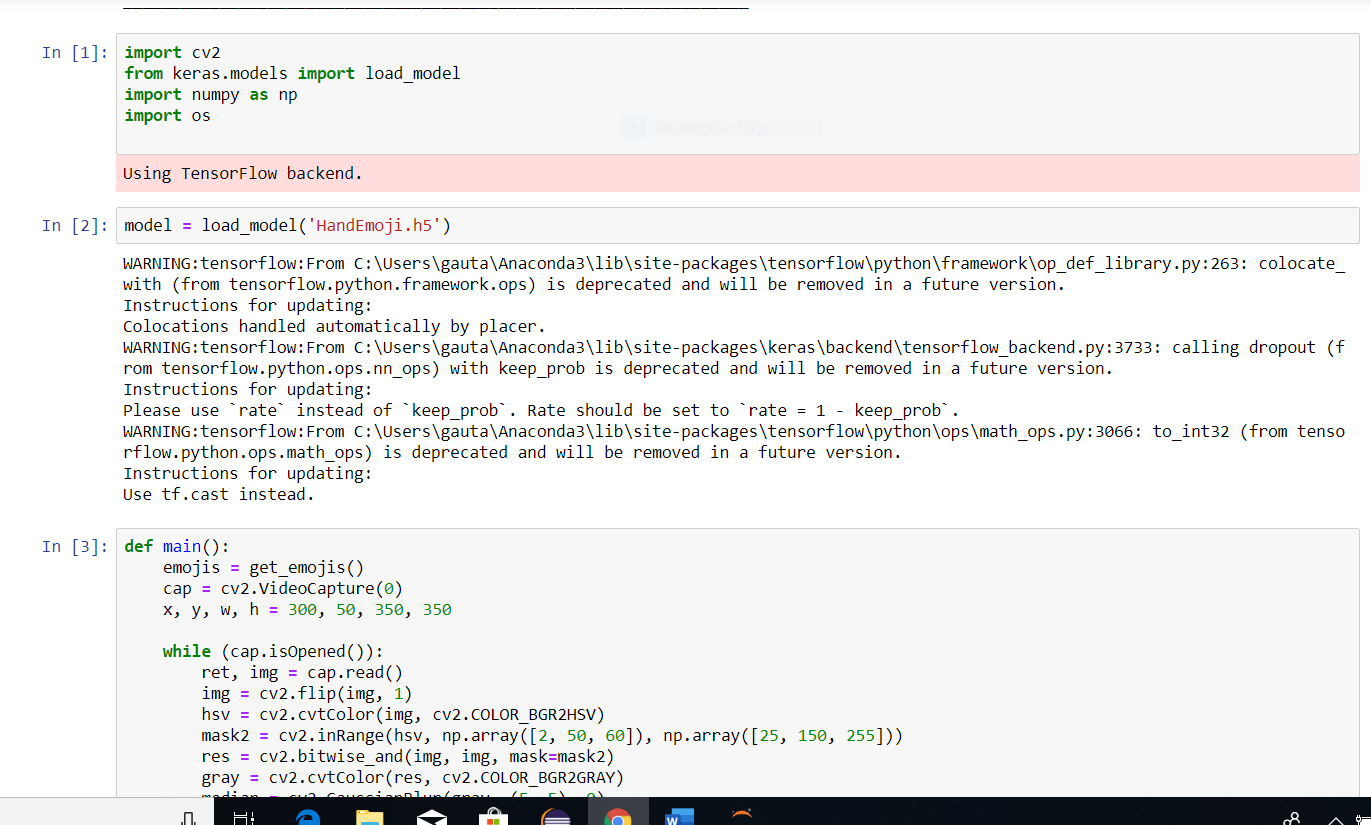


Fig 17: Loading of model

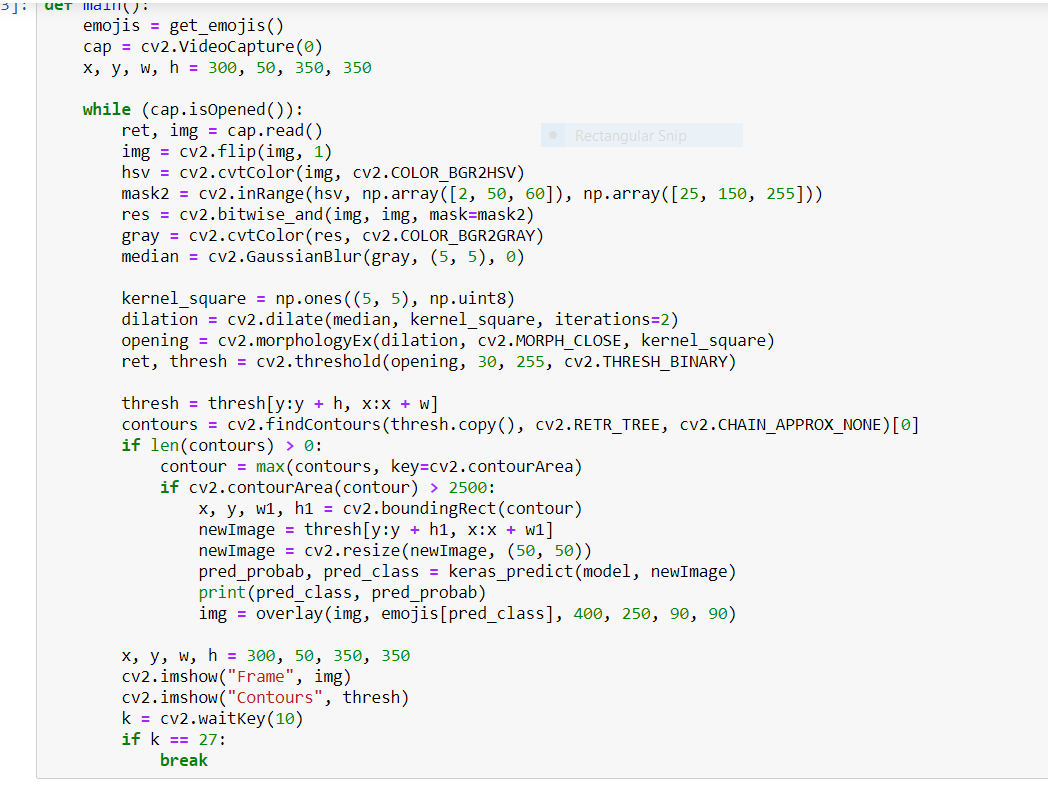
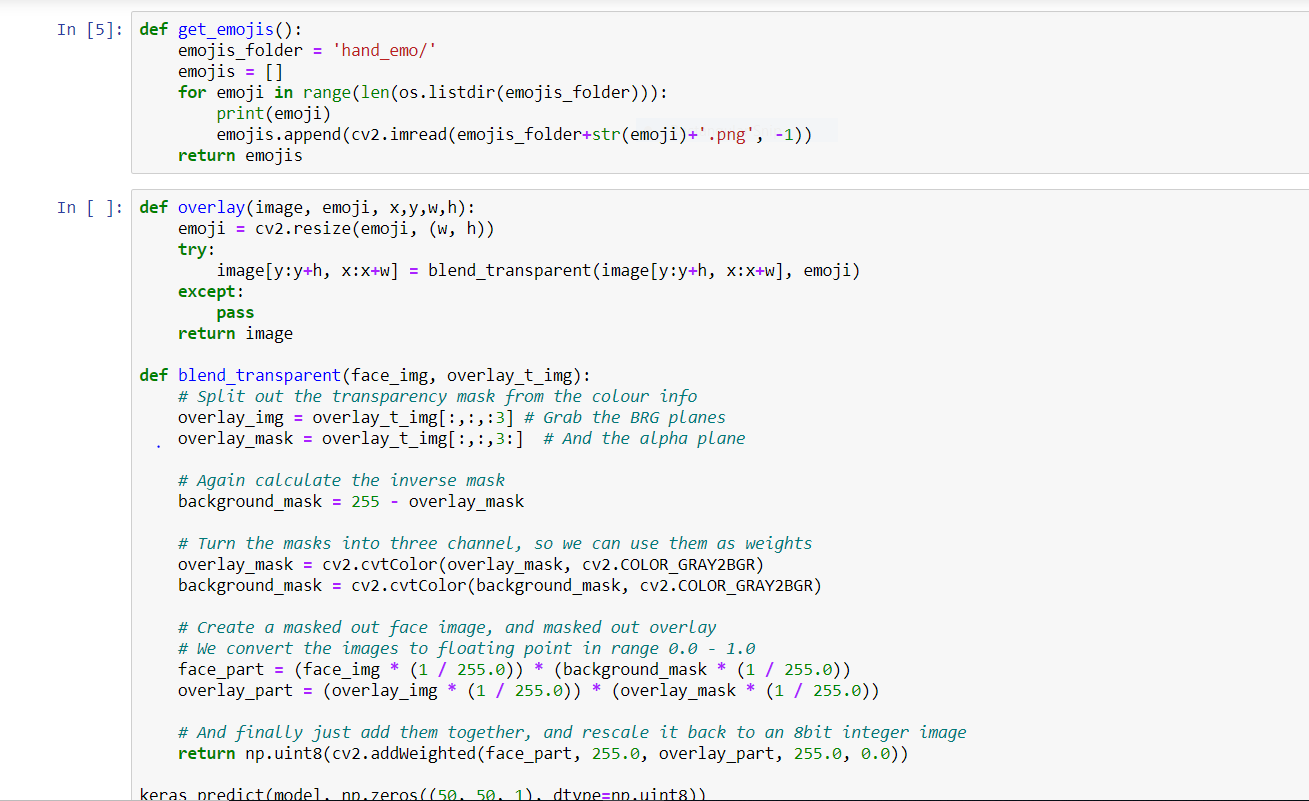
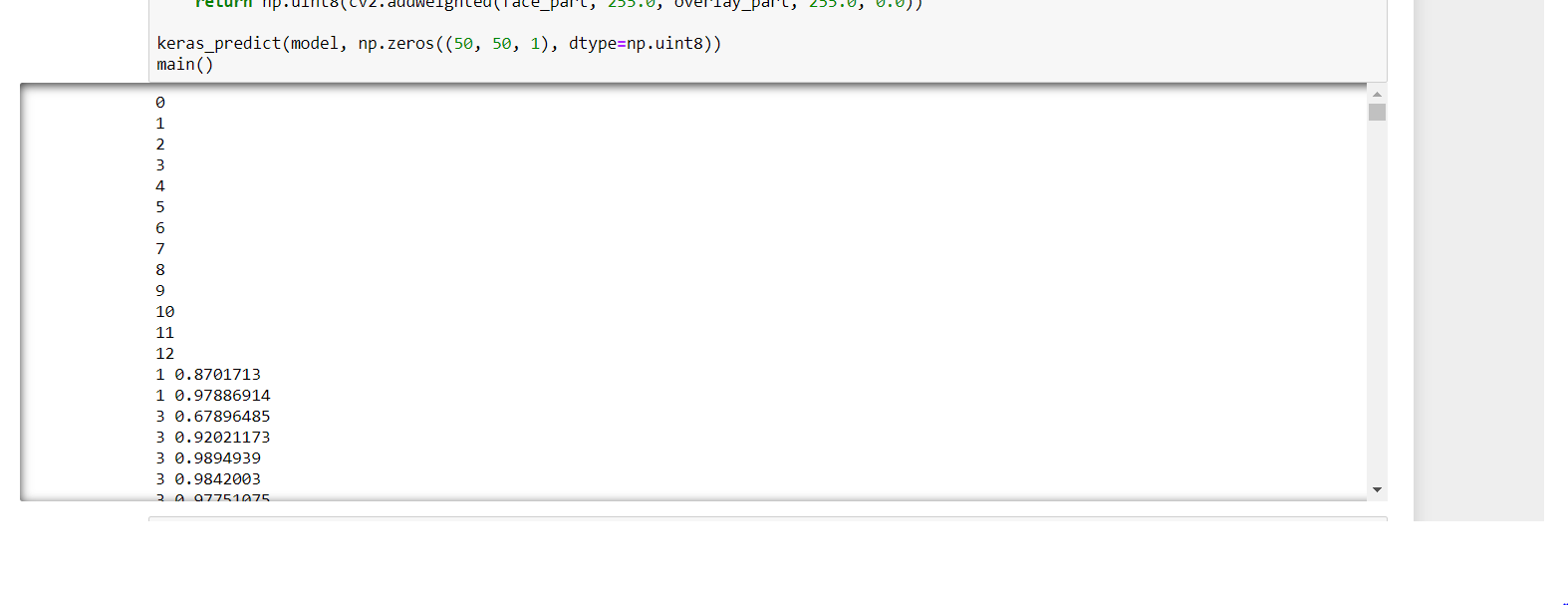


Fig 18: Matching of the input with the model

This is the same code above mentioned it is used to open the web camera and then overlaying the image to the described contours and then displaying it with our hand gestures and keras\_predict and pred\_probab are used in this.



  
 Fig 19: Printing of matched image set with the hand emoji displayed

WORKING IMAGES OF THE PROJECT



Fig 20: Display the correct hand emoji with one hand gesture



Fig 21: Display another working emoji