Handwriting detection using Python

**Table of Content**

1. Overview
2. Dataset
3. Model Training
4. GUI Implementation
5. Usage Instructions

**1. Overview**

The Handwriting Recognition project aims to recognize handwritten words and digits. The project utilizes the a-z-handwritten dataset from Kaggle and the MNIST dataset from Keras. The datasets are merged to create a comprehensive recognition system. The model is implemented using Keras Sequential and is saved as an HDF5 model file.

User is presented with a minimal looking GUI created using opencv, model is loaded and image processing is taken care by pillow and opencv

**2. Dataset**

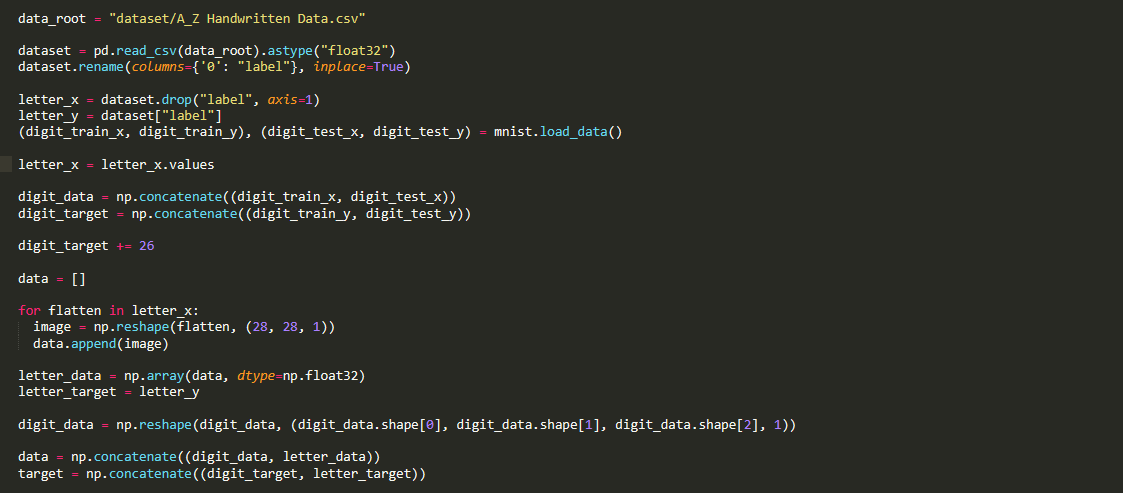
1. The a-z-handwritten dataset from Kaggle consists of handwritten letters in lowercase.

Dataset link : [The a-z-handwritten dataset](https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format)

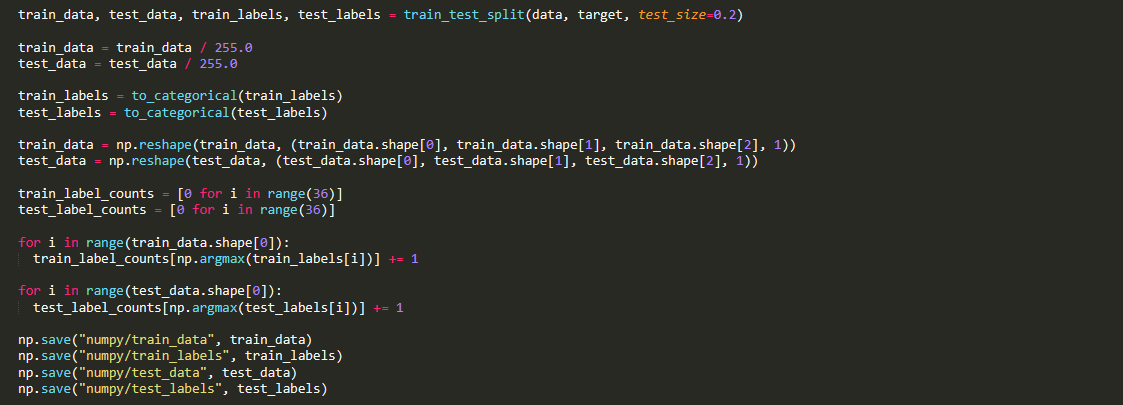
2. The MNIST dataset from Keras is a collection of handwritten digits (0-9)

**3. Model Training**

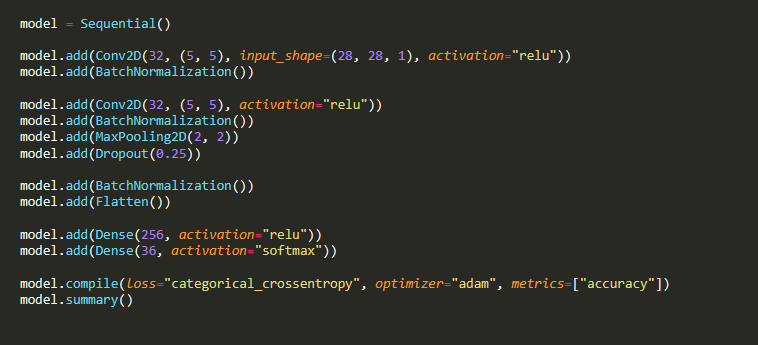
**3.1. Dataset loading and preprocessing**



Both alphabets and digit datasets are loaded and saved in numpy array, after providing proper labelling and reshaping both are concatenated to create a single array with all the data that we will further use to train, validate and test our model



We will then split our dataset into train and test, scale it and store labels into variables, finally we will save this numpy array into a npy file for quick access if needed



The model is implemented using the Keras Sequential API. The architecture includes layers for data preprocessing, feature extraction, and output prediction.



After creating model we will save model files for specific use cases(modelcheckpoints) and then train our model with our pre-processed dataset , accuracy is checked by implementing on test dataset and then finally model data is stored in a model file so that it can be loaded in Gui program

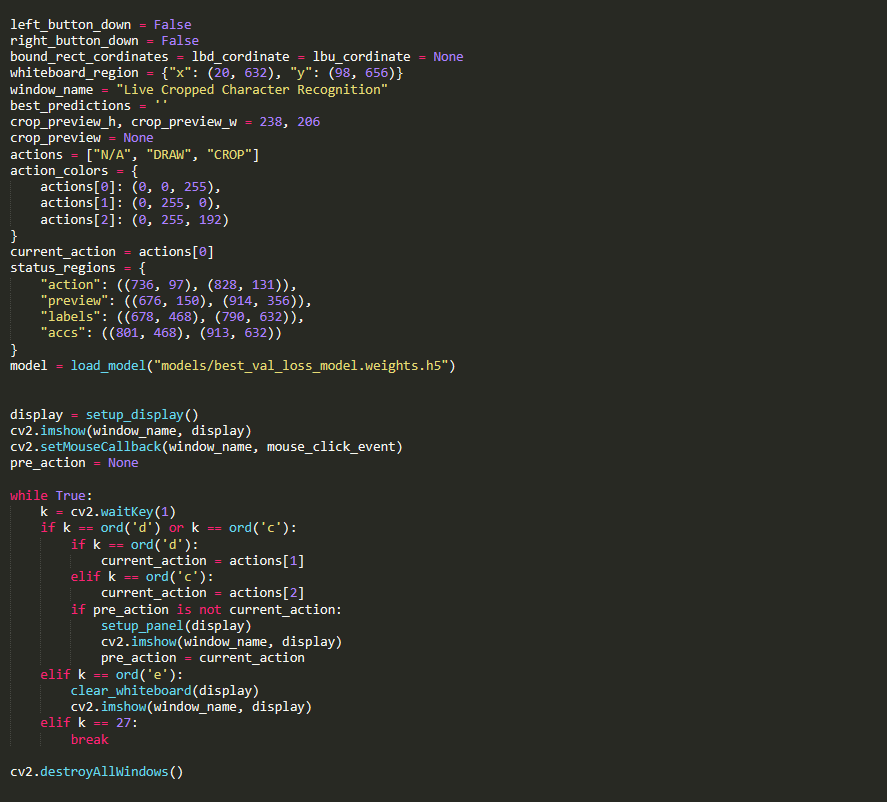
**4. GUI implementation**

Each Function has different functionality so lets discuss them one by one with no particular order

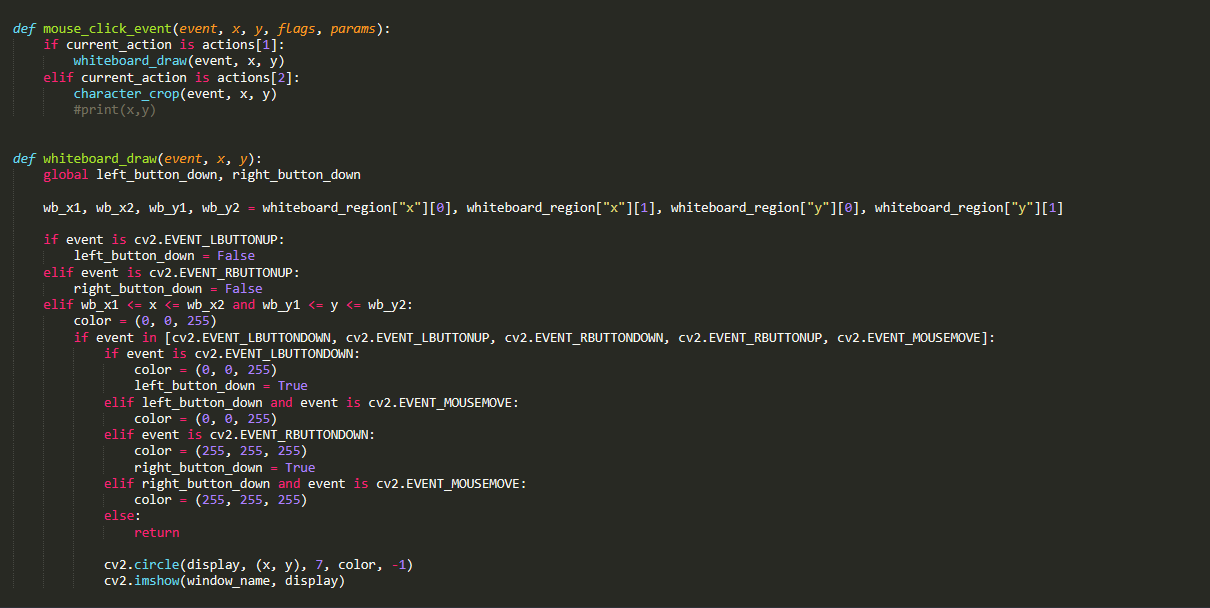
**4.1 Display setup and layout setup**

****

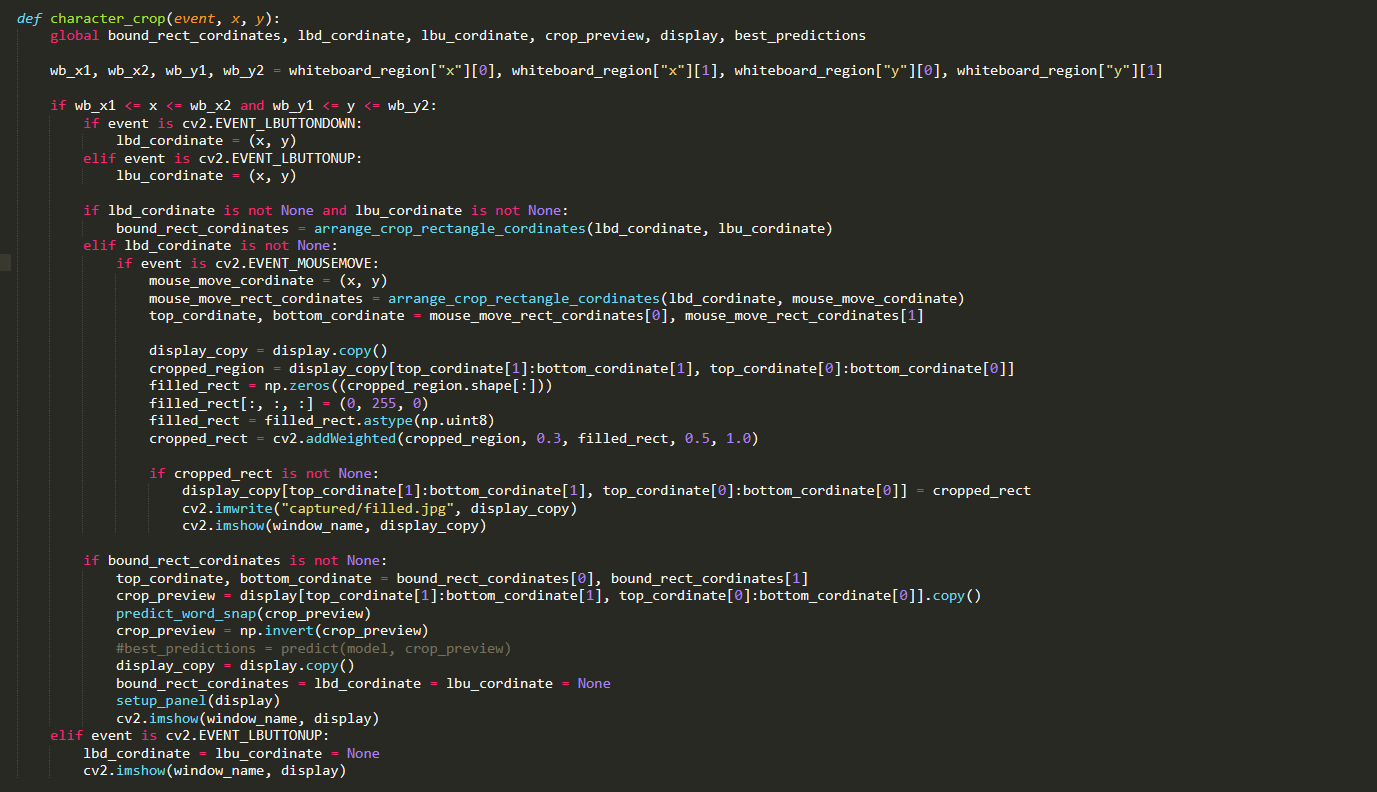
we are using opencv to create our GUI and entire display and layout setup is split into two functions setup\_display being responsible for displaying basic layout of whiteboard and panels for text display, on the other hand setup\_desplay is responsible for dynamic layout setting like preview section and output section(currently inactive and under development due to some bugs output is currently displayed in terminal using a print function )



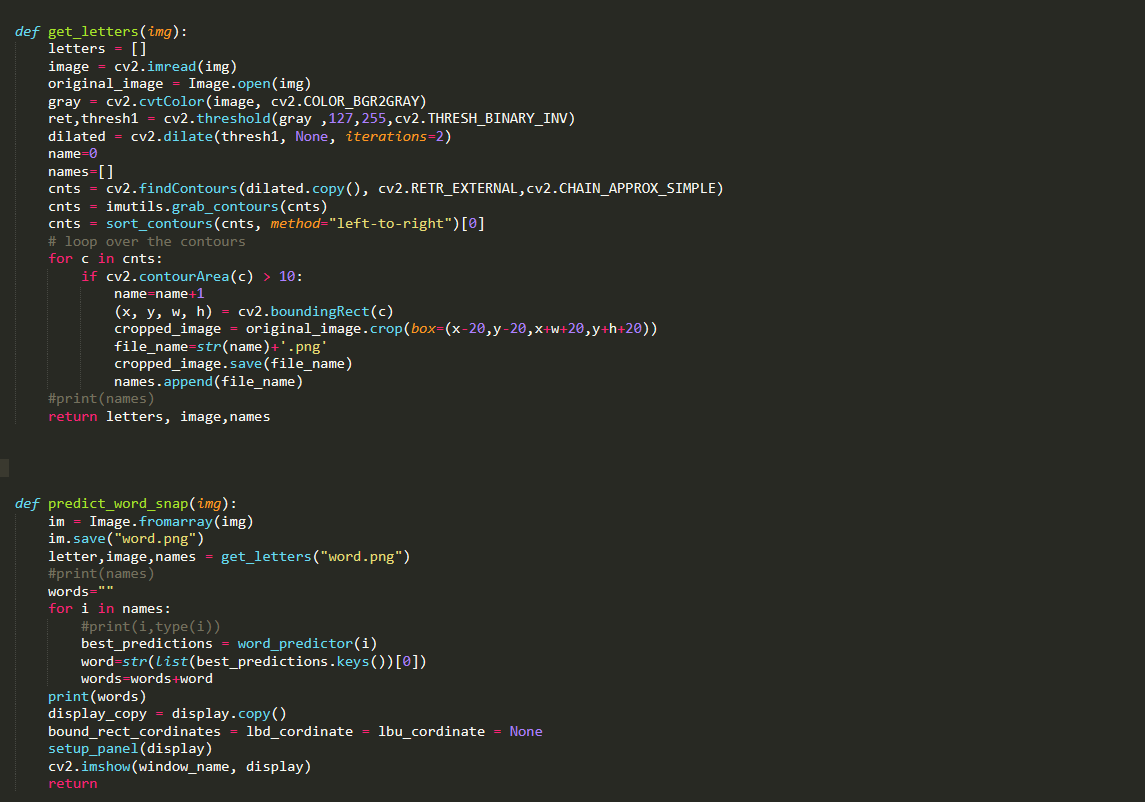
This will initialize our GUI, we have also defined some global variables that will be used in later sections, we have also load the model that we have trained and saved in the previous section, we have also defined actions that user can perform namely draw and crop, we have also defined key press for each actions like ‘e’ will erase the screen , ‘d’ to draw and soo on , we have also defined whiteboard area and coordinate and colour for some elements



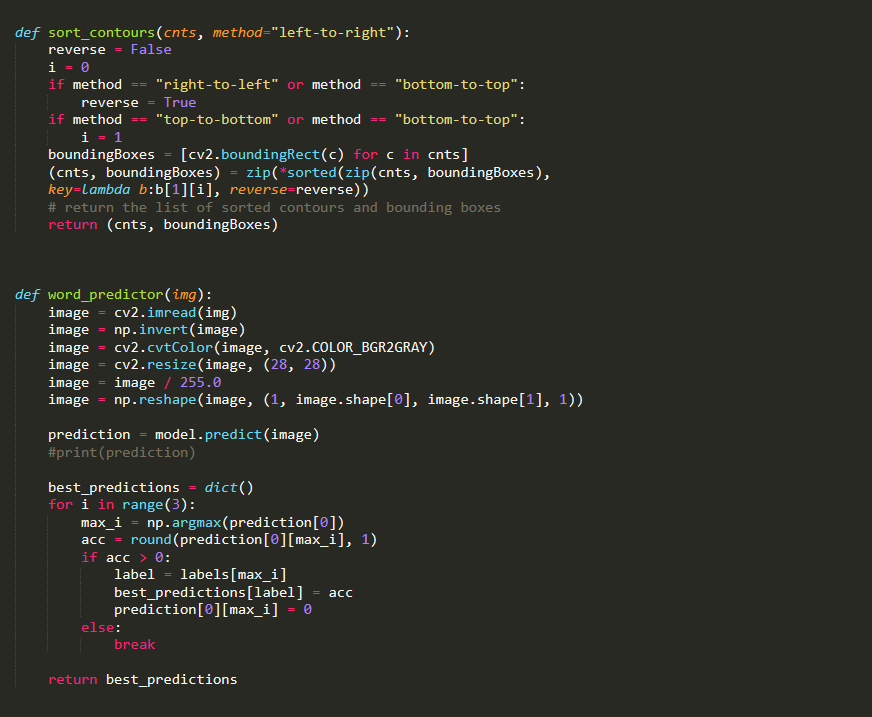
After the user presses ‘d’ we want to draw on the whiteboard, mousepressdown will trigger the pen and mousepressup will release the pen, colors and size of the pen is also defined, if we can make it dynamic or user defined we can do that by adding buttons on our gui for changing colour



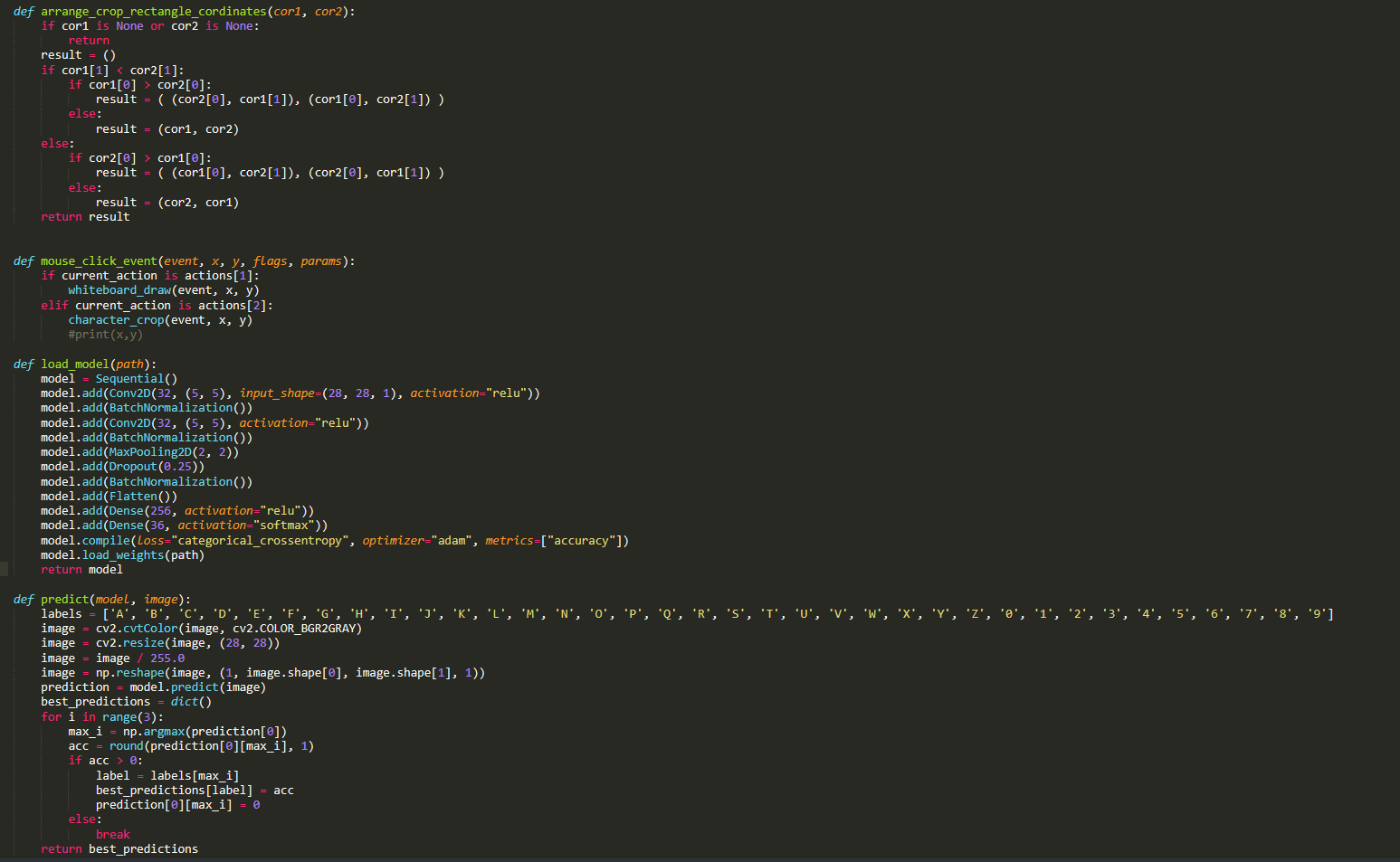
Now when the user presses ‘c’ button we want to crop what the user wants us to examine for words and digits, so we grab the mouse coordinates on the whiteboard and drag it till the mouse button is released we will also add some colour to highlight the cropped area for effects after that we will provide this same area to another function to examine or predict for word or digit



This predict\_word\_snap function will store incoming numpy array (cropped area) into an image and then call another function get\_letter which will load that image and find contours (another function will be discussed next ) , then we will iterate on each contour and store them as individual image (basically previous image was our whole word and each contour represent an alphabet or character ) after this each file name is stores in an array and returned back to previous function , back in this function we will call word\_predictor function (will be discussed ahead) for each image file, which returns a dictionary of probabilities for each characters the first key is highest so we will grab it with index and store into a variable and after all the images are done or all the contours are examines we will display the final word



Above function is used to find contours or group of elements that are together, it is used here to separate alphabets and digits from word ,word\_predictor function is called for each separated character image, it will first load it and perform some operations like resizing, normalizing etc and them feed it to our model for result, now model returns a dictionary of values , we will select top 3 of them and return them back to our previous function



Arrange\_crop\_rectangle\_coordinates will return coordinates for rectangle that is used in crop functionality this function is making sure no matter what way crop is done right coordinate is returned back, mouse\_click\_event as the name suggests switches to desired operation based on click, model function will load our sequential model

**5. Usage Instructions**

Usage is pretty straight forward, user is welcomed with 2 panels

1. First one being the whiteboard where user will write
2. Second is preview and output area(output area has encountered some bugs (model is working perfectly) so the output is displayed in terminal instead of GUI )

User can perform 3 operations

1. Press ‘d’ key and draw on the whiteboard
2. Press ‘c’ and select the area that they want to examine for word or digit
3. Press ‘e’ to erase the screen