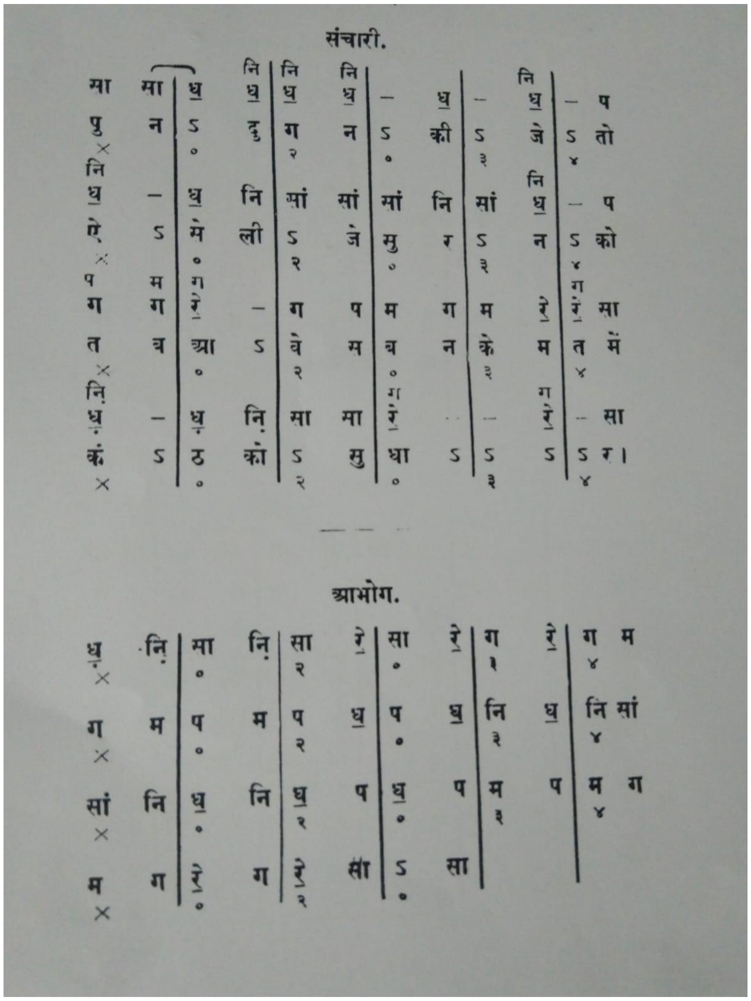
Indian Institute of Information Technology, Vadodara
(Gandhinagar Campus)
Summer Research Internship - 2022

Extraction of melodies with pitch classes From images of HCM compositions in Bhatkhande notation

under the mentorship of
Dr.Pratik Shah

Submitted by
Manjot Singh (201951090)

Abstract - Conversion of Music Notes from images of Bhatkhande Notation System into MIDI file(audio file) involving Images Processing using OpenCV, Deep convolutional neural network modeling using Tensorflow and Keras and extracting those notes to be converted into a playable audio file using music21 toolkit for computer-aided musicology. The source code and other involved files for this project can be found here.



I. Introduction

The Indian Classical Music holds a very high place in our Indian culture, to the point that it is synonymous to Indian festivals, religious customs, marriages, exotic cuisine and Traditional Clothing. Doesn't matter who you are or what part of India you belong to, the minute you hear Indian music, the nostalgia and the amusement kicks in. Even if you are not from India, it would still create a sense of elegance and simplistic pleasures.

But despite its legendary status in the music culture, it seems that it is not being practiced heavily by the masses. Part of the reason is that it is less accessible and not being able to be visualized from reading about it in books and online articles. We are living in the age of social media and Short Clips platforms which tend to cause low attention span. "TikTok use disorder (TTUD) is positively linked to memory loss, and it is also positively linked to depression, anxiety, and stress. Depression, anxiety, and stress are positively linked to memory loss. Furthermore, depression, anxiety, and stress have a mediating effect between TTUD and memory loss." [1]

This project tries to bridge that gap between written medium and being able to hear and picture the beauty of the melodies. To be able to just convert your favorite melodies instantly from unexciting texts in your old books to audio files in your phone or laptop almost seems like a dream come true for a music enthusiast.

II. Problem Statement

Now that we understand the importance of the project, we need to understand and break down what are the necessary steps we need to take to solve this problem.

Task: From images of HCM compositions in Bhatkhande notation, extract the melodies with pitch classes.

So, initially this image needs to be converted into a supported audio file.

III. Proposed Solution

A lot of approaches had to be considered for this project. The final approach ended up being diving this task into 3 parts:

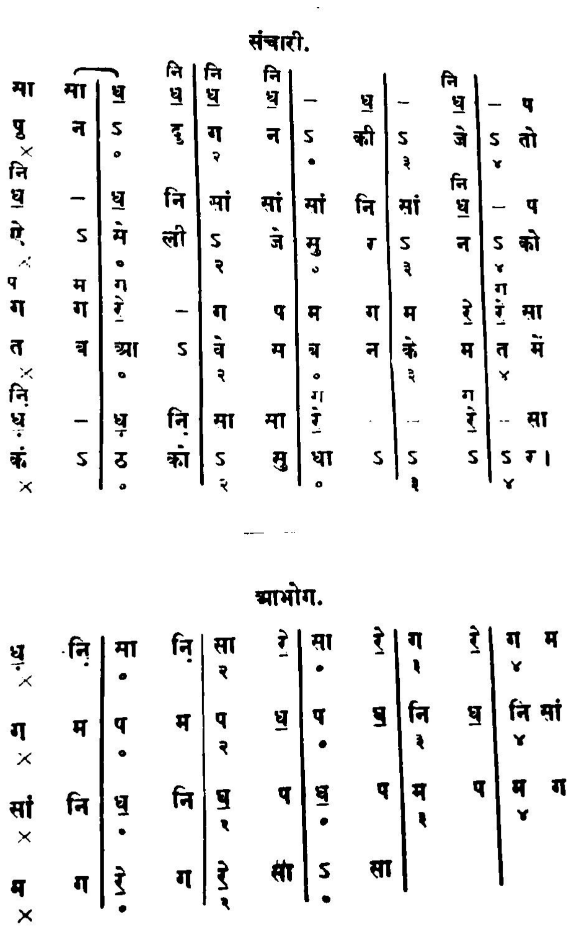
- NOTATION RECOGNITION AND PROCESSING
- NOTATION CLASSIFICATION
- AUDIO EXTRACTION

Or in simpler terms, this is divided as:

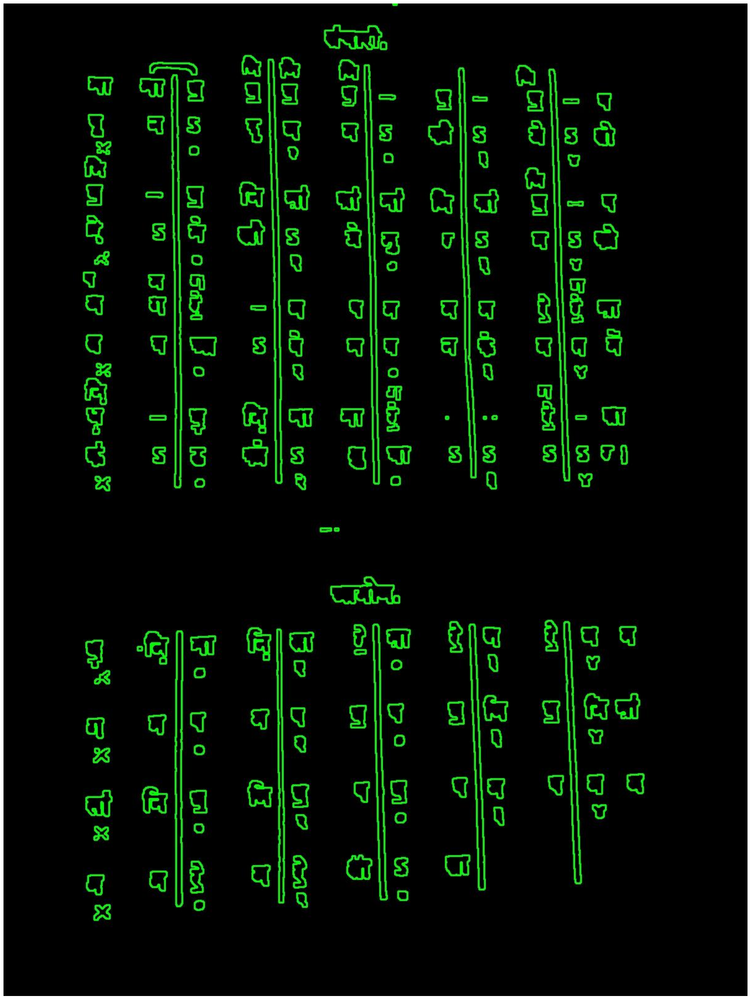
- What to read in this image?
- What is the meaning of the part that I just read in the image?
- What sound to produce in response to what I just read.

PART A - NOTATION RECOGNITION AND PROCESSING

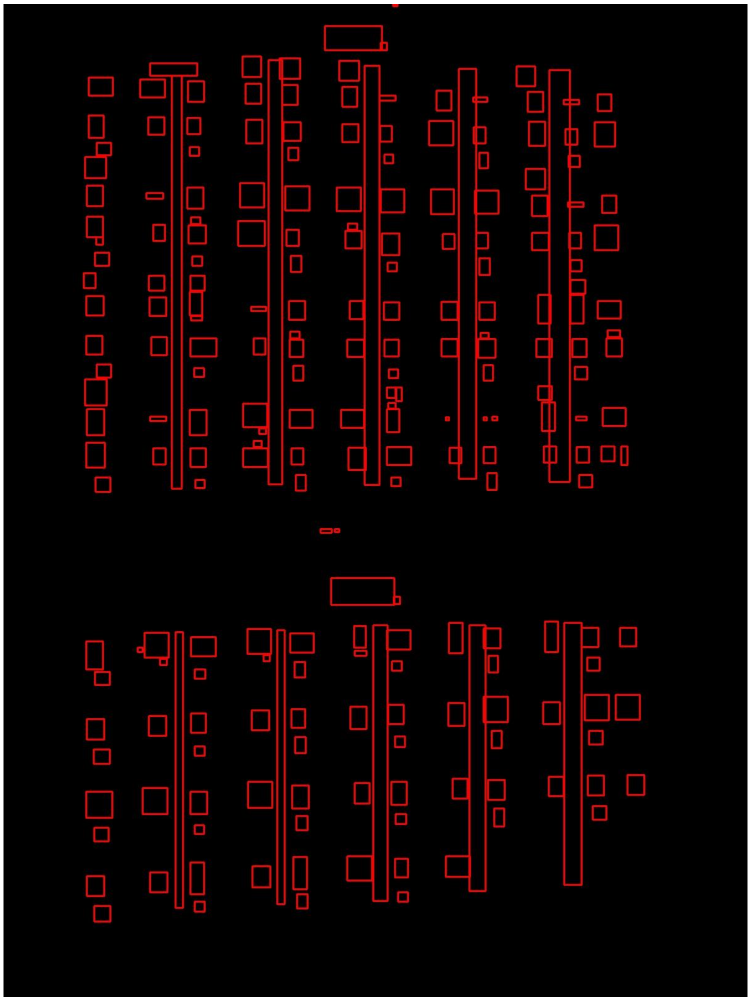
For this, OpenCV is used to convert the notation image to a data array of the image. First comes Preprocessing of the image in which all the extra noise on the image is removed, like the color of the paper, different shades of color etc, so that it would not interfere with the detection. After this we end up with a binary version of the image where only two colors exist, Black (0,0,0) and White(255,255,255).



After that, contours(outline along the boundary) of the image are detected, extracted and drawn onto another blank image.



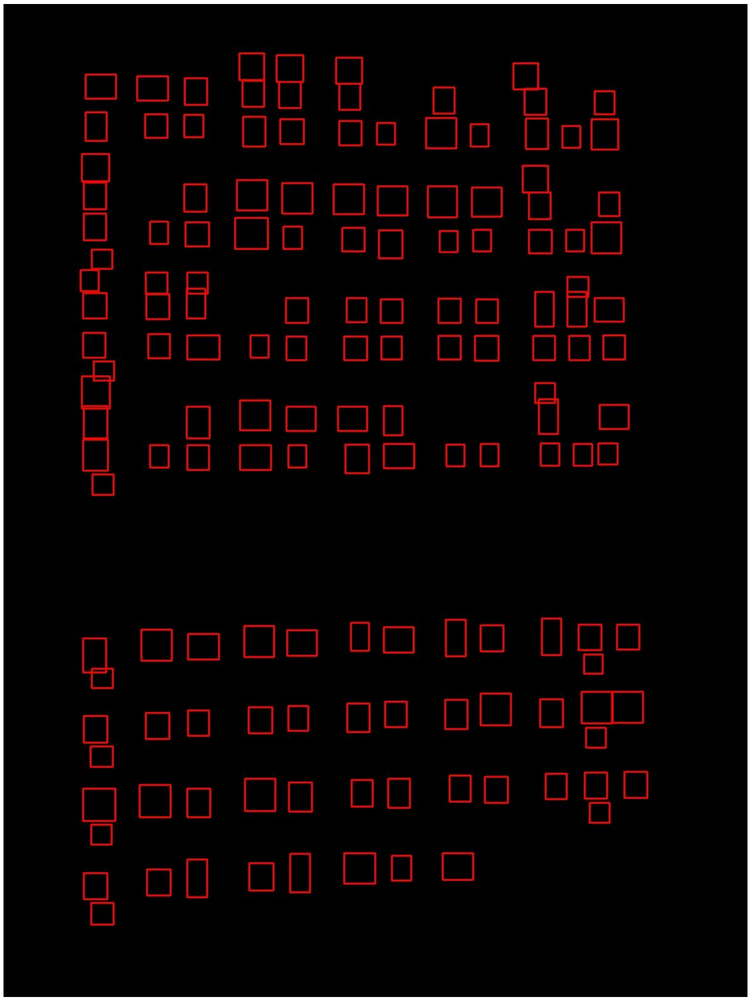
Using these contours, Bounding Rectangles are drawn on the co-ordinates of these contours and saved on another blank image.



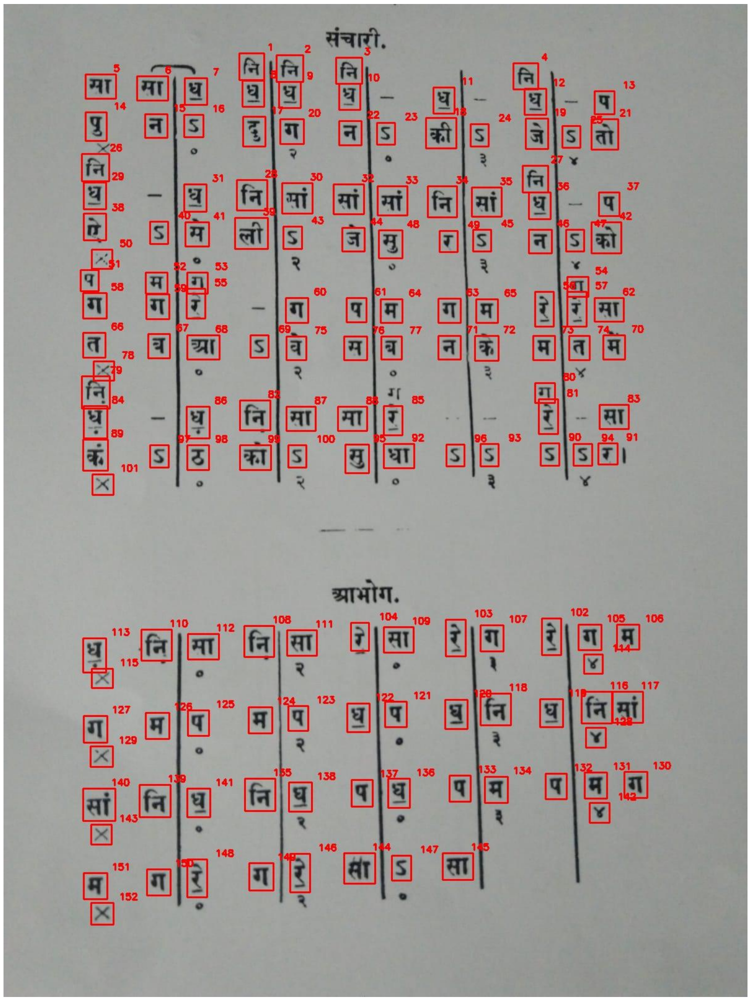
After getting all the bounding rectangles of all the contours out of the image, selective filtering of the contours is done for classification and thus removing the unnecessary clutter like vertical lines, heading, numberings etc.

We get x, y, h, w from contours as properties. For filtering, 4 conditions were set:

- **h / w < 3**, i.e. *Height / Width should be less than 3* (This removes all the tall lines which have a lot more height than width).
- **w / h < 2**, i,e. *Width / Height should be less than 2* (This removes all the horizontal lines and heading which have a lot more width than height).
- **h > 20,** i.e. *h should be greater than 20px* (This removes smaller elements such as numbers or dots)
- **w> 18,** i.e. *w should be greater that 18px* (This removes marks and numberings)



And Voila, we ended up with the desired elements needed for the classification. These elements are cut out of the original binary images in order for the classification.

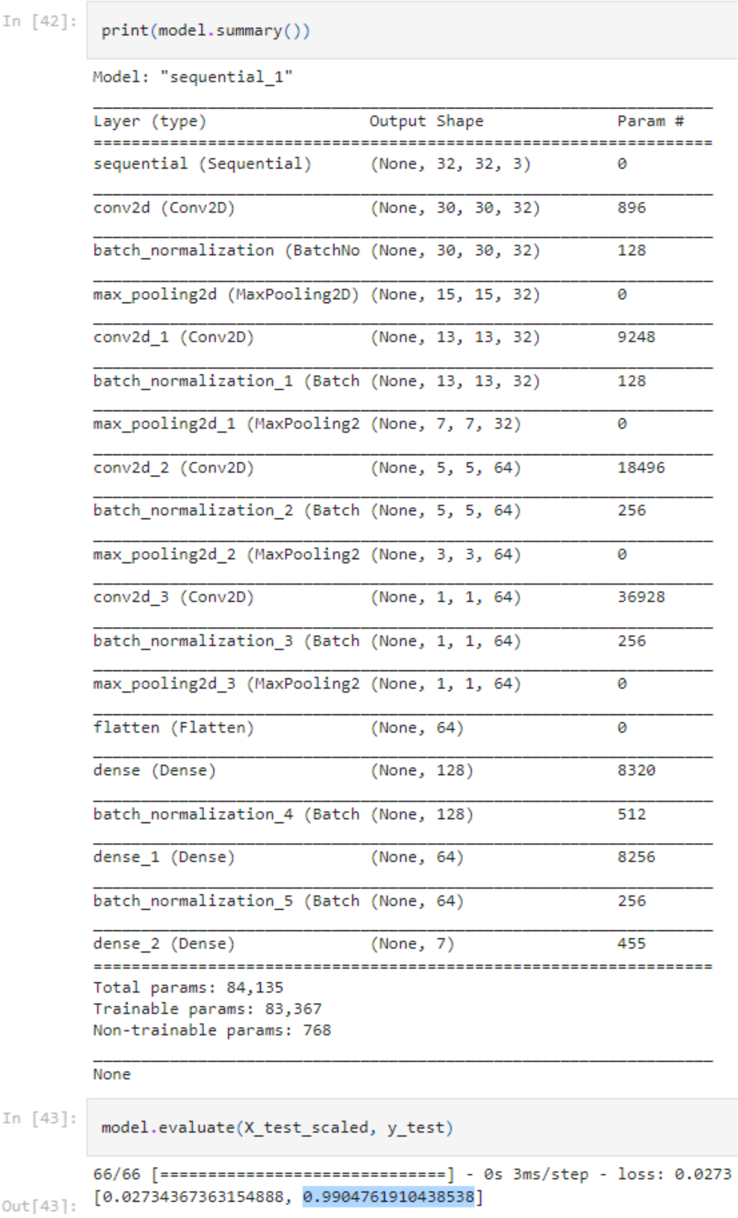


After this, we need to pass these bounded cutout elements in an array for further classification in the Deep CNN Model.

PART B - NOTATION CLASSIFICATION

A lot of approaches had to be considered for this part, since there was no dataset available for the hindi music notation, Devnagri Hindi Text Classification ended up being used and only selective elements, namely (सा रे गा मा प ध) were ended up getting passed to the model. These images are resized to 32px X 32px for uniformity, converted to binary as well and normalized.

Data Augmentation is also used to add random flips, random rotation and random zoom to counter overfitting of the model. in addition to the normal Deep CNN model with max pooling, batch normalization, and adding multiple dense layers.



```
In [42]:  print(model.summary())

          Model: "sequential_1"
          _________________________________________________________________
          Layer (type)                Output Shape              Param #
          =================================================================
          sequential (Sequential)     (None, 32, 32, 3)         0
          _________________________________________________________________
          conv2d (Conv2D)             (None, 30, 30, 32)        896
          _________________________________________________________________
          batch_normalization (BatchNo (None, 30, 30, 32)       128
          _________________________________________________________________
          max_pooling2d (MaxPooling2D) (None, 15, 15, 32)       0
          _________________________________________________________________
          conv2d_1 (Conv2D)           (None, 13, 13, 32)        9248
          _________________________________________________________________
          batch_normalization_1 (Batch (None, 13, 13, 32)       128
          _________________________________________________________________
          max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)         0
          _________________________________________________________________
          conv2d_2 (Conv2D)           (None, 5, 5, 64)          18496
          _________________________________________________________________
          batch_normalization_2 (Batch (None, 5, 5, 64)         256
          _________________________________________________________________
          max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)         0
          _________________________________________________________________
          conv2d_3 (Conv2D)           (None, 1, 1, 64)          36928
          _________________________________________________________________
          batch_normalization_3 (Batch (None, 1, 1, 64)         256
          _________________________________________________________________
          max_pooling2d_3 (MaxPooling2 (None, 1, 1, 64)         0
          _________________________________________________________________
          flatten (Flatten)           (None, 64)                0
          _________________________________________________________________
          dense (Dense)               (None, 128)               8320
          _________________________________________________________________
          batch_normalization_4 (Batch (None, 128)              512
          _________________________________________________________________
          dense_1 (Dense)             (None, 64)                8256
          _________________________________________________________________
          batch_normalization_5 (Batch (None, 64)               256
          _________________________________________________________________
          dense_2 (Dense)             (None, 7)                 455
          =================================================================
          Total params: 84,135
          Trainable params: 83,367
          Non-trainable params: 768
          _________________________________________________________________
          None

In [43]:  model.evaluate(X_test_scaled, y_test)

          66/66 [==============================] - 0s 3ms/step - loss: 0.0273
Out[43]:  [0.02734367363154888, 0.9904761910438538]
```

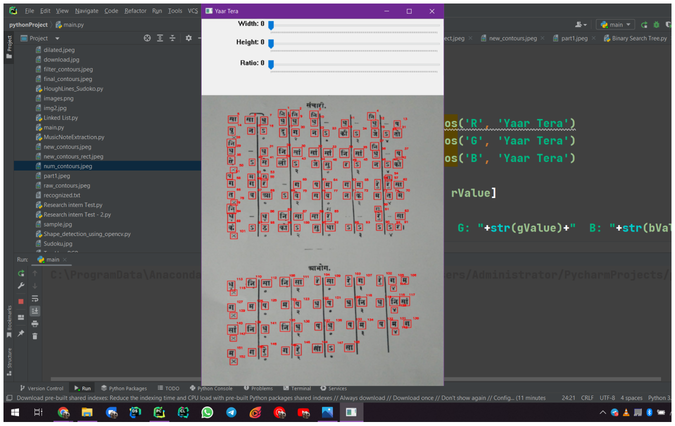
PART C - NOTATION CLASSIFICATION

The image array from PART A is used here to classify the acquired images from value 0 to 6 and then using a dictionary convert to Sa, Re, Ga, Ma, Pa, Dha, Ni and Saa. This array is then converted to stream using music21 library and converted to MIDI file and used as audio file.

IV. Experiments

- **Trackbars for customized Width, Height and Aspect Ratio selection.**

  Since this project was mainly carried out in Jupyter Notebooks, I ended up hard coding the values required for this particular image since Jupyter doesn't allow interactive windows. But it was brought to my attention that these height and width values may not work on different images with smaller or larger text. So a commercial version of this project works, say for React or Android, i tried for an interface to which includes a trackbar inorder for the correct texts to be recognized.



Yaar Tera

Width: 0

Height: 0

Ratio: 0

```
os('R', 'Yaar Tera')
os('G', 'Yaar Tera')
os('B', 'Yaar Tera')

rValue]

 G: "+str(gValue)+"  B: "+str(bVa
```

```
C:\ProgramData\Anaconda
ers/Administrator/PycharmProjects/
```

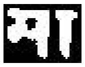
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Config... (11 minutes

- **<u>Iterative approach based on Pixel value threshold</u>**

  I ended up with an Contour detected based approach, but previously I was set on an Iterative based approach/ Sliding Window approach. I manually handpicked points on top left of the text necessarily and counted the number of pixels used to show that piece of text , i.e सा took 342 pixels on average, रे took 275 pixels on average, गा 313 pixels on average, मा took 361 on average, प took 256 and ध took 310. In short, in the 1600 X 1200 image, I planned to pick a 45 X 45 window on the top left and move it right and bottom and pass only those pixels that satisfy constraints based on these conditions. I was later made aware that my approach was fine but the implementation of it was somewhat naive. But by then, the contours based approach worked so I went with a contour based approach. This Pixels data with sliding window approach is still here.

- **<u>Improving Datasets</u>**

  Since the dataset I received was somewhat different that the same I used the model on. E.g. not being able to read the works properly even after preprocessing.



Sa getting classified as Maa.

I fiddled around with the data set but the end solution is that the model need to be trained using more of the text from the original sample to match.

V. Conclusion

This project really made me aware of a lot more concepts in Machine Learning, Deep Learning and Image Manipulation. The best part about it is how different parts of this project do different things and it all comes together in the end ever so seamlessly.

References

[1] Sha P, Dong X. Research on Adolescents Regarding the Indirect Effect of Depression, Anxiety, and Stress between TikTok Use Disorder and Memory Loss. Int J Environ Res Public Health. 2021 Aug 21;18(16):8820. doi: 10.3390/ijerph18168820. PMID: 34444569; PMCID: PMC8393543.