Recognition of Handwritten Mathematical Expressions

A Project Report submitted
in partial fulfilment of the requirements for the degree of

B. Tech.

By:

Apoorva
[LCS2019018],
Akanksha
[LCS2019019],
Roopam Jain
[LCS2019050]

Under the supervision of:
Dr. Soumendu
Chakraborty

to

Indian Institute of Information Technology, Lucknow

भारतीय सूचना प्रौद्योगिकी संस्थान, लखनऊ

Department of Computer Science

May 8, 2022

Declaration of Authorship

We, Apoorva [LCS2019018],
Akanksha [LCS2019019],
Roopam Jain [LCS2019050], declare that this project titled, "Recognition of Handwritten Mathematical Expressions" and the work presented in it are our own. We confirm that:

- This work was done wholly or mainly while in candidature for a B.Tech at this Institute.
- Where any part of this project has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this Project is entirely our own work.
- We have acknowledged all main sources of help.
- Where the project is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signature: ______________________________

Date: May 8, 2022
Place: Lucknow, India

[Recognition of Handwritten Mathematical Expressions]

by [Apoorva, Akanksha, Roopam Jain]

Abstract

Recognition of handwritten text has been a prominent problem in the field of image processing and recognition of handwritten mathematical equations is even more challenging due to a large variety of mathematical symbols and the complex structures of the equations. Also, mathematical equations are used in all kinds of scientific studies and by using a calculating software we can solve these equations easily and quickly with no errors. There are various kinds of software present online that detect mathematical equations written using a digital pen on the go. [4] But there might be people who do not have access to a digital pen. To the best of our knowledge, there are no software present online that detect handwritten mathematical equations containing subscripts and superscripts by taking their images as input and predicting their corresponding notations.

In this project, we made a model that can process an image of a handwritten mathematical equation containing subscripts and superscripts, captured by a suitable sensor, recognise the equation and form the corresponding digital form of the equation. This digital form can be used as an input to various computational software.

Acknowledgements

We would like to express our gratitude to Dr. Soumendu Chakraborty for his valuable suggestions and encouragement in completion of our project. Without his timely guidance and insights on image segmentation techniques, this project would be nowhere compared to where it is today. We would also like to extend our thanks to Dr. Deepshikha Agarwal and Dr. Mainak Adhikari and our honourable Dean, Dr. Dhananjoy Dey for their support in the accomplishment of our project. We would also be thankful to our director Dr.Arun Mohan Sherry for providing all the required facilities in completion of this project.

Lastly, we would like to extend our deep appreciation to our group members, without whose support and co-ordination the project would not have come to fruition.

Contents

Declaration of Authorship **iii**

1 Introduction **1**
1.1 Background 1
1.2 Motivation 1
1.3 Objective 1
1.4 Contributions 2

2 Literature Review **3**

3 Problem Formulation **5**

4 Proposed Methodology **7**

5 Experiments and Results **11**

6 Conclusion and Future Work **15**

A Contribution of Group Members **17**
A.1 Contribution of Group Member 1 17
A.2 Contribution of Group Member 2 17
A.3 Contribution of Group Member 3 17

B Appendix B **19**

C Appendix C **23**

Bibliography **25**

List of Figures

5.1 Input image being processed and segmented 11
5.2 Y coordinates of each element is checked 11
5.3 Symbol delta being recognized 12
5.4 Symbol gamma being recognized 12
5.5 LATEX format output is given 13

C.1 In this page you need to select the type of expression which you want to recognize 23
C.2 In this page you need to upload the image 24
C.3 In this page you will get the output string 24

Chapter 1

Introduction

1.1 Background

Recognition of handwritten text means interpreting the handwritten text present on papers or written by using digital pen. Detecting and interpreting the text present in images is called optical character recognition. A lot of work has been done in handwritten text recognition field. Most of the research were focused on recognizing English letters and numbers. Later, researchers started developing systems to recognize simple mathematical expressions containing basic math operations, such as +, -, / and x. However, not much research has been done on recognition of complex mathematical expressions that involve exponents and subscripts in offline mode. We dig into this problem in our work.

1.2 Motivation

While using mathematical software we faced problem in inputting the mathematical expressions into the software. It is easier to draw various symbols and digits on paper rather than typing them. By being able to accurately recognize and predict expressions and mathematical expressions, we would be able to change the way students learn and share knowledge online. There are various kinds of software present online that detect mathematical expressions written using a digital pen on the go. But there might be people who do not have access to a digital pen. To the best of our knowledge, there are no software present online that detect handwritten mathematical expressions containing subscripts and superscripts by taking their images as input and predicting their corresponding digital notations. We wanted to foray into the field of image processing and realized that the above mentioned problem could be solved by it. This forms the basis for our motivation behind the project.

1.3 Objective

The major objective of this project is to learn and implement CNNs to solve the problem of recognizing handwritten mathematical expressions. We do this by researching the work done previously by various authors in this field, collate that knowledge and try to build something of our own in the process. We have tried to develop software that can take images of handwritten mathematical expressions containing subscripts and superscripts as input and recognize the expressions. Another objective of this project to build a basic web application that showcases the work done by us in a simple format.

1.4 Contributions

Our project investigates the problem of recognizing handwritten mathematical expressions. Our contribution is in creating an end-to-end system using a well-trained CNN model to detect handwritten symbols and expressions correctly and creating a basic flask application to showcase the functionality of the model. We observed that we can differentiate between superscript, base script and subscript present in an equation by checking their y-coordinates. We generated three algorithms based on our observations in order to identify which element of the equation is superscript, base script or subscript.

Chapter 2

Literature Review

A lot of research has been done on recognition of handwritten numbers and English words [5]. But not much research has been done around the recognition of handwritten mathematical expressions containing exponents and subscripts.

MyScript (previously Vision Objects) won the 2013 Competition on Recognition of Online Handwritten Mathematical Expressions (CHROHME), with a 60% accuracy at the expression level. The second place holders had built a model that could predict the equations with an accuracy of 26% only. In the software developed by MyScript, the user draws a mathematical expression on the canvas using their mouse and gets back a valid TeX markup and its visualization [4]. The poor results in CROHME reflect the general state of handwritten expression recognition public research.

An independent researcher built a system that could only recognize the first digit after the power sign [1]. Ole Kroger et. al. managed to develop a framework of constructing LaTeX equations out of handwritten images [2]. Catherine Lu et. al. and Karanveer Mohan et. al. ended up creating an end-to-end system using a welltrained CNN model to go from strokes to symbols to a LATEX expression [3].

Dmitry Zhelezniakov et. al., Viktor Zaytsev et. al. and Olga Radyvonenko et. al. published a research paper in which they surveyed the work of over 170 researchers around the world and accumulated the results of their findings over the last 40 years [6].

Several research papers were based on detecting only linear equations with limited set of digits and symbols. Most of them were detecting the equations in online (live) mode but a lot of people might not have access to digital pen. We took it upon ourselves to build such a system that can take images of different kinds of handwriiten mathematical expressions as input, identify them and convert them into a suitable digital format (preferably LATEX) and which can then be fed into other computational software.

Chapter 3

Problem Formulation

To the best of our knowledge, there were no software available which could take image of handwritten mathematical expressions containing superscript and subscript as input and then recognize the corresponding expression present in the image.

Chapter 4

Proposed Methodology

In our project we would enter an image containing handwritten mathematical expression and our model would detect the expression and print its equivalent $\LaTeX$ string. After carefully going through various resources, we have decided to use the "Handwritten Math Symbols" dataset present on Kaggle and MNIST dataset. Our solution involves two steps:
(1) First part of our solution requires the segmentation of the image.
(2) Secondly we need to recognize the symbols present in the image.

Image Segmentation

We would be using the threshold method for image segmentation where based on the intensity, the pixels in an image get divided by comparing the pixel's intensity with a threshold value. Currently we are just doing simple threshold. The reason for using the threshold method is that there is a stark difference between background and foreground pixels and by using the threshold method, we can easily select digits and symbols while ignoring the black background.

We would input an image containing a handwritten expression and apply some image pre-processing, and obtain contours from left to right of the image. We first find the contours related to the symbols present in the image. Sometimes, we may get two or more contours for the same digit/symbol. To avoid that, we can check if the bounding rectangle of those two contours overlaps or not. If they overlap, then discard the smaller rectangle. We can then store each contoured image in an array which can be further fed into our model to predict each digit and symbol.
The above method works well for expressions that are only linear but mathematical expressions are much more than that. The main idea behind detecting the characters and recognising whether they are superscript/subscript/base of an expression is experimenting with their y-coordinates. Three algorithms were generated for the following three types of expressions :

- **Expressions containing only characters in superscript and base :**
  Here, each character is being contoured individually whether it is superscript or base-script. Therefore, we would be noting down the y-coordinates of each character and we would be storing them in an array. Now we would be noting down the maximum of them all. This would be the character at the lowest level(hence given that it would be in the base of the expression). Similarly a character with the lowest y-coordinate would be in the superscript. After observing the y-coordinates, a after a little trial and error, it was concluded that all the y-coordinates of characters in the superscript followed the following :

y_{super}	=	y-coordinates of characters in the superscript
max	=	maximum y-coordinate present in the array.

  $$y_{super} < 0.6 * (y_{max})$$

  By using this criteria, we would be classifying the characters as superscript or base.This is stored in an array called 'power_rect' and it is stored as :

0	:	The character is base-script
1	:	The character is super-script

  This information would be stored in an array and would be passed onto the next step of the algorithm along with array of segmented images.

- **Expressions containing only characters in subscript and base :**
  Here, each character is being contoured individually whether it is subscript or in base-script. Therefore, we would be noting down the y-coordinates of each character and we would be storing them in an array. Now we would be noting down the maximum of them all. This would be the character at the lowest level(hence given that it would be in the subscript of the expression). Similarly a character with the lowest y-coordinate would be in the base-script. After observing the y-coordinates, and after a little trial and error, it was concluded that all the y-coordinates of characters in the base-script followed the following :

y_{base}	=	y-coordinates of characters in the base-script
max	=	maximum y-coordinate present in the array.

  $$low = 0.4 * (max)$$

  $$high = 0.8 * (max)$$

  $$(y_{base} \geq low) and (y_{base} \leq high)$$

  By using this criteria, we would be classifying the characters as subscript or base-script.This is stored in an array called 'power_rect' and it is stored as :

1	:	The character is base-script
2	:	The character is sub-script

  This information would be stored in an array and would be passed onto the next step of the algorithm along with array of segmented images.

- **Expressions containing characters in superscript, subscript and base :**
  Here, each character is being contoured individually whether it is superscript, subscript or in base-script. Therefore, we would be noting down the y-coordinates of each character and we would be storing them in an array. Now we would be noting down the maximum of them all. This would be the character at the lowest level(hence given that it would be in the subscript of the expression). Similarly a character with the lowest y-coordinate would be in the super-script. After observing the y-coordinates, and after a little trial and error, it was concluded that all the y-coordinates of characters in the base-script and superscript followed the following :

y_{base}	=	y-coordinates of characters in the base-script
y_{super}	=	y-coordinates of characters in the superscript
max	=	maximum y-coordinate present in the array.

  $$low = 0.4 * (max)$$

  $$high = 0.8 * (mx)$$

  $$base = 0.6 * (mx)$$

  $$low_{super} = 0.2 * (base)$$

  $$high_{super} = 0.6 * (base)$$

  $$(y_{base} \geq low)\ and\ (y_{base} \leq high)$$

  $$(y_{super} \geq low_{super})\ and\ (y_{super} \leq high_{super})$$

  By using this criteria, we would be classifying the characters as superscript, subscript or base-script. This is stored in an array called 'power_rect' and it is stored as :

0	:	The character is superscript
1	:	The character is base-script
2	:	The character is sub-script

  This information would be stored in an array and would be passed onto the next step of the algorithm along with array of segmented images.

Symbol Recognition

We used Convolutional Neural Networks(CNNs) for the recognition of different digits and symbols. There is no need for manual feature extraction in CNNs as the system itself learns to do feature extraction and they can also develop internal representation of 2D images.
We used 2D convolution layers to classify the images from 45 classes as it gives us relationships between low-level features in the X-Y dimension. We used three 2D convolutional layers in sequential order each with 5*5 kernel size

having 32, 64, 128 filters respectively.After every convolutional layer there is a maxpooling 2d layer with 2*2 kernel size to reduce the dimensionality. It reduces the number of parameters, which both shortens the training time and combats overfitting.Then the output is sent to drop-out function. We used drop-out to avoid the problem of over-fitting. It means during training time, at each iteration, a neuron is temporarily "dropped" or disabled with probability p. Output of the drop-out function is first flattened and then fed into the three fully connected layers. Finally the output of the fully connected layers is then fed into the final output layer which predicts the symbol in the input image.

After recognizing the symbols of each element of the image array a predicted array is formed in which the predicted value of each element is stored order wise.To form the output string each element is checked if it is superscript or base or subscript by using the power_rect array. After checking if the element is base, superscript or power they are formatted according to the rules of LaTeX and added to the output string. The output generated can be run as a mathematical expression on LaTeX software.

Chapter 5

Experiments and Results

- Image Segmentation

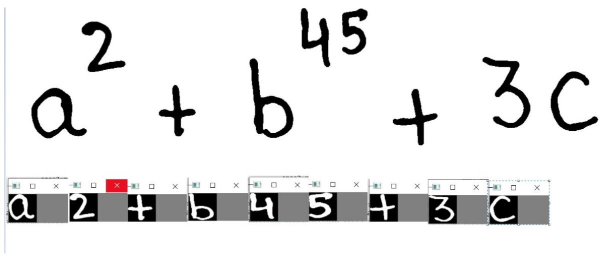
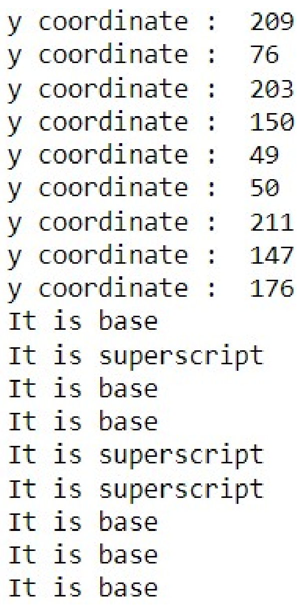


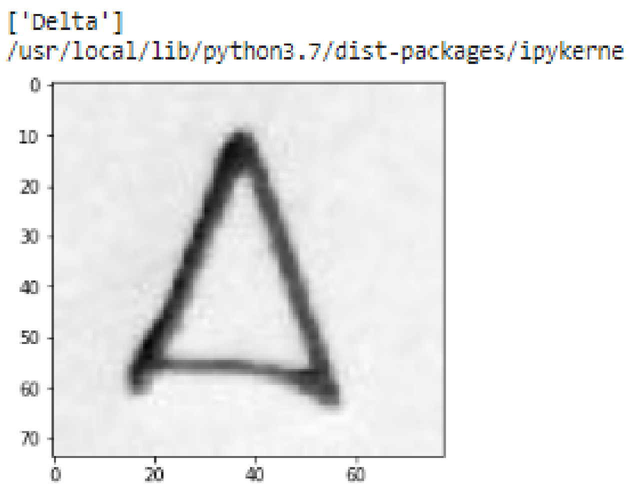
FIGURE 5.1: Input image being processed and segmented



```
y coordinate :  209
y coordinate :  76
y coordinate :  203
y coordinate :  150
y coordinate :  49
y coordinate :  50
y coordinate :  211
y coordinate :  147
y coordinate :  176
It is base
It is superscript
It is base
It is base
It is superscript
It is superscript
It is base
It is base
It is base
```

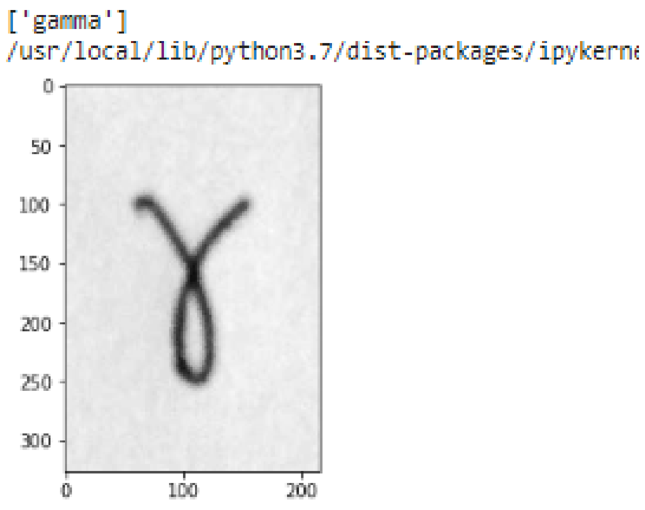
FIGURE 5.2: Y coordinates of each element is checked

- Symbol Recognition



```
['Delta']
/usr/local/lib/python3.7/dist-packages/ipykerne
```

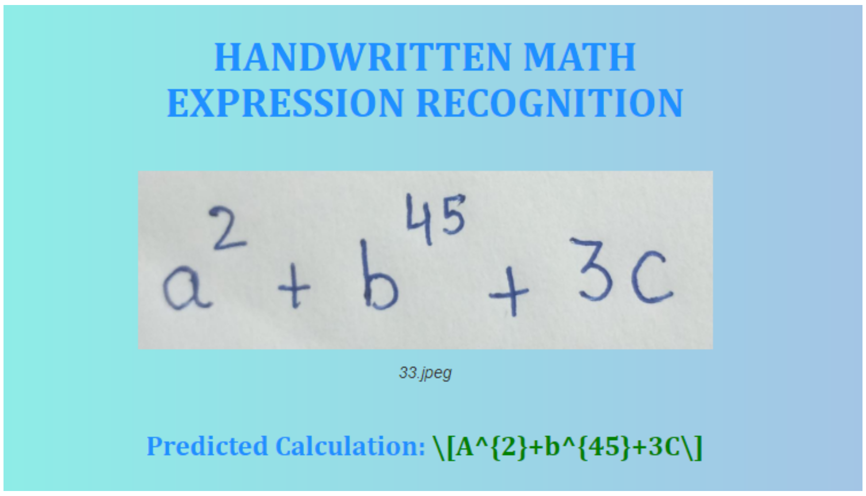
FIGURE 5.3: Symbol delta being recognized



```
['gamma']
/usr/local/lib/python3.7/dist-packages/ipykerne
```

FIGURE 5.4: Symbol gamma being recognized

- Output Generated


HANDWRITTEN MATH
EXPRESSION RECOGNITION

Predicted Calculation: \[A^{2}+b^{45}+3C\]

FIGURE 5.5: LATEX format output is given

Chapter 6

Conclusion and Future Work

In this project we were able to make three algorithms to differentiate between subscript, base script and superscript of an expression. We generated three algorithms – one for expressions containing only superscript, one for expressions containing both superscript and subscript and one for expressions containing only subscript. We were able to make a CNN model for symbol classification with 92.69 percent accuracy when tested on a test set containing 87,986 images. We were able to integrate our model on a flask web app to make it easy to use.

We have built a model in which the user needs to manually choose the kind of equation they need to input, and based on that our model is able to accurately predict expressions with subscripts, superscripts and expressions containing both subscripts and superscripts separately. In the future, we would like to build upon the existing model and try to train an aggregate model in which there is no need of specifying the kind of equation being inputted manually and the model can identify and predict these different expressions on its own.
Another limitation is that we are able to give LaTeX output of a limited class of symbols only. This problem arises due to the non-availability of higher computational prowess and CUDA GPUs needed to train and test larger amount of data. Availability of machines with better computation capacity would allow us to significantly bring down model simulation time and train a larger class of symbols, enabling us to expand the scope of this project to even more kinds of mathematical expressions.

Appendix A

Contribution of Group Members

A.1 Contribution of Group Member 1

Member Name - Apoorva [LCS2019018]

- Wrote the code for image segmentation
- Wrote the code classification of characters on the basis of whether they are superscript, subscript or base-script
- Made the front-end of the flask app

A.2 Contribution of Group Member 2

Member Name - Akanksha [LCS2019019]

- Wrote the code for CNN model for symbol recognition
- Made the backend of the flask app.

A.3 Contribution of Group Member 3

Member Name - Roopam Jain [LCS2019050]

- Deploying a basic web app built using flask on the back-end and HTML, CSS and JavaScript on the front-end. The app displays basic functionality of the trained machine learning model
- Collecting and cleaning the dataset and performing further computations

Appendix B

Appendix B

(1) Code for image segmentation :

```python

img = cv2.bilateralFilter(image, 15, 20, 20)
thresh = 177
blur = cv2.GaussianBlur(img,(13,13),0)
img = cv2.threshold(blur, thresh, 255, cv2.THRESH_BINARY)[1]
cv2.imshow("image",img)
cv2.waitKey(0)
cv2.destroyAllWindows()
if img is not None:
    img=~img
    ret,thresh=cv2.threshold(img,127,255,cv2.THRESH_BINARY)
    ctrs,ret=cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    cnt=sorted(ctrs, key=lambda ctr: cv2.boundingRect(ctr)[0])
    w=int(28)
    h=int(28)
    train_data=[]
    #print(len(cnt))
    rects=[]
    for c in cnt :
        x,y,w,h= cv2.boundingRect(c)
        rect=[x,y,w,h]
        rects.append(rect)
    #print(rects)
    bool_rect=[]
    for r in rects:
        l=[]
        for rec in rects:
            flag=0
            if rec!=r:
                if r[0]<(rec[0]+rec[2]+10) and rec[0]<(r[0]+r[2]+10) and r[1]<(rec[1]+rec[3]+10) and rec[1]<(r[1]+r[3]+10):
                    flag=1
                l.append(flag)
            if rec==r:
                l.append(0)
        bool_rect.append(l)
```

(2) Code for classifying the characters as superscript or base-script :

```python
    i  = 0
    cutoff = 0.6*(mx)
    for y in coordinates :
        if y<cutoff :
            power_rect.append(1)

        else:
            power_rect.append(0)
    for img in img_wala:
        if power_rect[i]==1 :
            print("It is superscript")
            winname = "Test"
            cv2.namedWindow(winname)
            cv2.moveWindow(winname, 400,400)

            cv2.imshow(winname,img)
            cv2.waitKey(0)
            cv2.destroyAllWindows()

        else :
            print("It is base")
            winname = "Test"
            cv2.namedWindow(winname)
            cv2.moveWindow(winname, 400,400)

            cv2.imshow(winname,img)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
        i+=1
```

(3) Code for classifying the characters as subscript or base-script :

```

cutoff_range_base_low = 0.4*(mx)
    cutoff_range_base_high = 0.8*(mx)
    #y<=cutoff_range_base_high and
    for y in coordinate_y :
        if y<=cutoff_range_base_high :
            power_rect.append(1)
            print("base")
        else :
            power_rect.append(2)
            print("sub")
```

(4) Code for classifying the characters as superscript, subscript and base-script :

```

cutoff_range_base_low = 0.4*(mx)
    cutoff_range_base_high = 0.8*(mx)
    base = 0.6*(mx)
    cutoff_range_pow_low = 0.2*(base)
    cutoff_range_pow_high = 0.6*(base)

    bm = 0;
    for y in coordinate_y :
        if y<=cutoff_range_base_high and y>=
   cutoff_range_base_low :
            power_rect.append(1)
            print(str(y)+" = 1 base")

        elif y<=cutoff_range_pow_high:
            power_rect.append(0)
            print(str(y)+" = 1 power")

        else :
            power_rect.append(2)
            print(str(y)+" = 1 sub")

        bm=bm+1
```

(5) Code for the CNN Model used:

```

class Model(nn.Module):
    def _init_(self):
        super(Model, self)._init_()
        self.layer1=nn.Sequential(
            nn.Conv2d(45, 32, kernel_size=5, stride=1, padding=3),
            nn.LeakyReLU(),
            nn.MaxPool2d(kernel_size = 2, stride=2)
        )
        self.layer2=nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=5, stride=1, padding=3),
            nn.LeakyReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.layer3=nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=5, stride=1, padding=3),
            nn.LeakyReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.drop_out = nn.Dropout()
        self.ffnn1 = nn.Linear(128*7, 500)
        self.ffnn2 = nn.Linear(500, 250)
        self.ffnn3 = nn.Linear(250, 45)

    def forward(self, x):
        output = self.layer1(x)
        output = self.layer2(output)
        output = self.layer3(output)
        output = output.reshape(output.size(0), -1)
        output = self.drop_out(output)
        output = self.ffnn1(output)
        output = self.ffnn2(output)
        output = self.ffnn3(output)
        return F.log_softmax(output)

net=Model().to(device)
```

(6) Code for output generated :

```
ans1 = "\["
```

```
    i = 0
    for a in final:

        if power_rect[i]==1 and power_rect[i-1]==1 :
            ans1+=a
        elif power_rect[i]==0 and power_rect[i-1]==1 :
            temp = '}' + a
            ans1+= temp
        elif power_rect[i]==1 and power_rect[i-1]==0 :
            temp = '^{'+a
            ans1+=temp

        else :
            ans1+=a
        i+=1
    #return predicted

    ans1+="\]"
    return ans1
```

Appendix C

Appendix C

A few screenshots of the flask application:

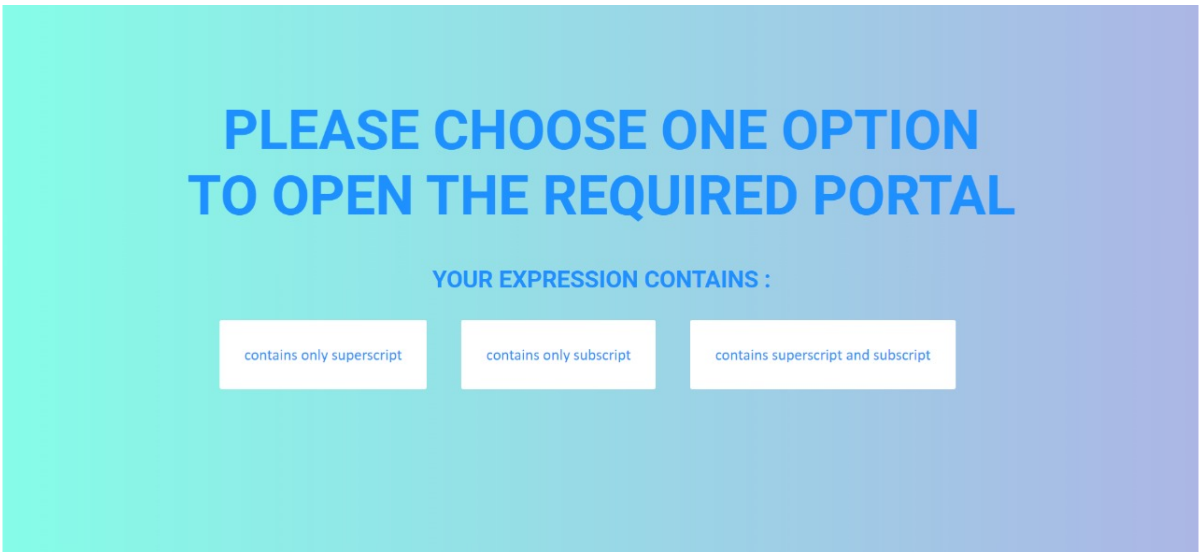

PLEASE CHOOSE ONE OPTION
TO OPEN THE REQUIRED PORTAL
YOUR EXPRESSION CONTAINS :
contains only superscript
contains only subscript
contains superscript and subscript

FIGURE C.1: In this page you need to select the type of expression which you want to recognize

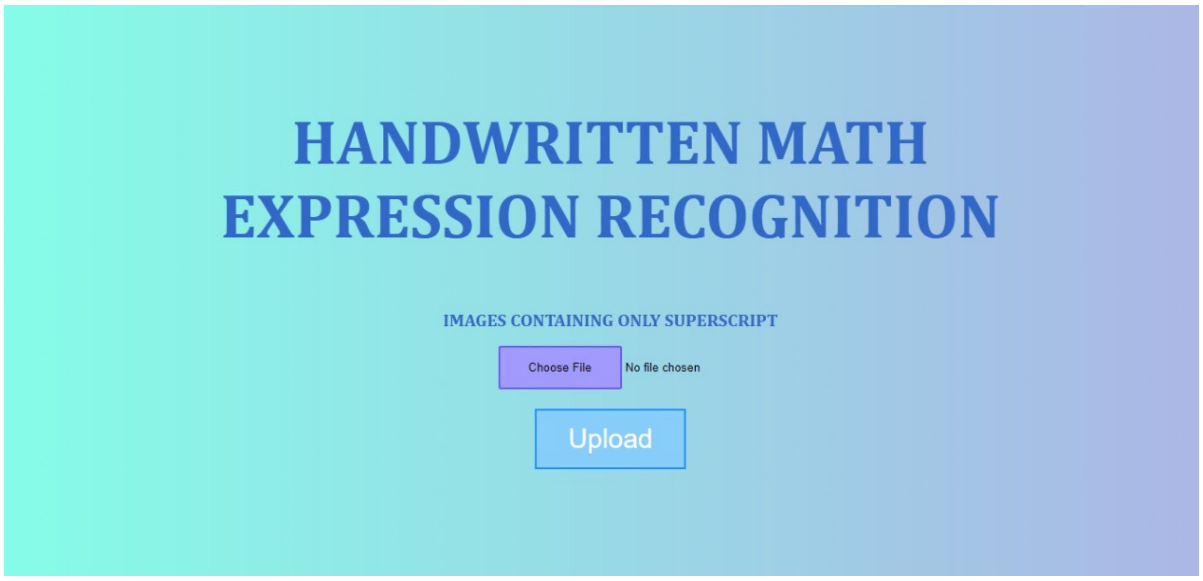
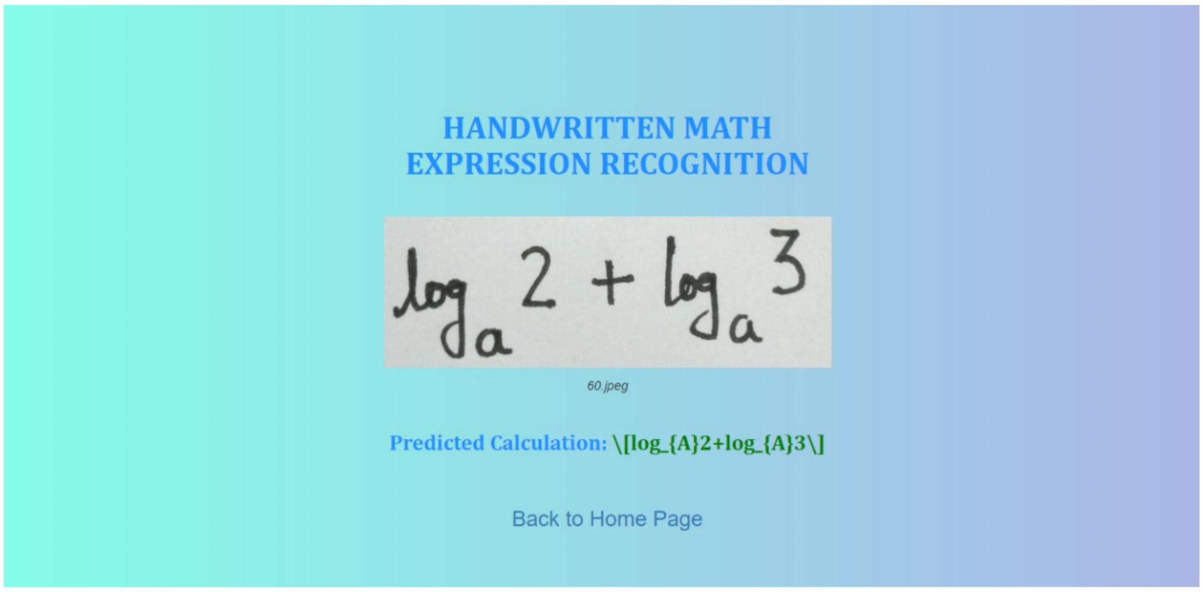

HANDWRITTEN MATH EXPRESSION RECOGNITION
IMAGES CONTAINING ONLY SUPERSCRIPT
Choose File
No file chosen
Upload

FIGURE C.2: In this page you need to upload the image



Predicted Calculation: \[log_{A}2+log_{A}3\]

FIGURE C.3: In this page you will get the output string

Bibliography

[1] 515handwritten-expression. *Hand-written Mathematical Expression Recognition System.* 2021.

[2] Ole Kroger. *Hand-written Mathematical Expression Recognition System.* 2019.

[3] Catherine Lu and Karanveer Mohan. *Recognition of Online Handwritten Mathematical Expressions Using Convolutional Neural Networks.* 2013.

[4] MyScript. *Demonstration Portal. Previously Vision Objects. url =* `http://webdemo.myscript.com/`.

[5] R. Plamondon and S. N. Srihari. "On-line and off-line handwriting recognition: A comprehensive survey". In: *IEEE Transactions on Pattern Analysis and machine intelligence* (2000).

[6] Dmytro Zhelezniakov, Viktor Zaytsev, and Olga Radyvonenko. "Online Handwritten Mathematical Expression Recognition and Applications: A Survey". In: *IEEE Access* 9 (2021), pp. 38352–38373. DOI: `10.1109/ACCESS.2021.3063413`.