JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

MINOR PROJECT REPORT

Mathematical Expression Calculator

Group Number: G21

Prateek Bhandari (18103078)
Shivansh Bansal (18103308)
Chaitanya Gautam (17103118)

SUPERVISOR – Dr. Arpita Jadhav Bhatt

ACKNOWLEDGEMENT

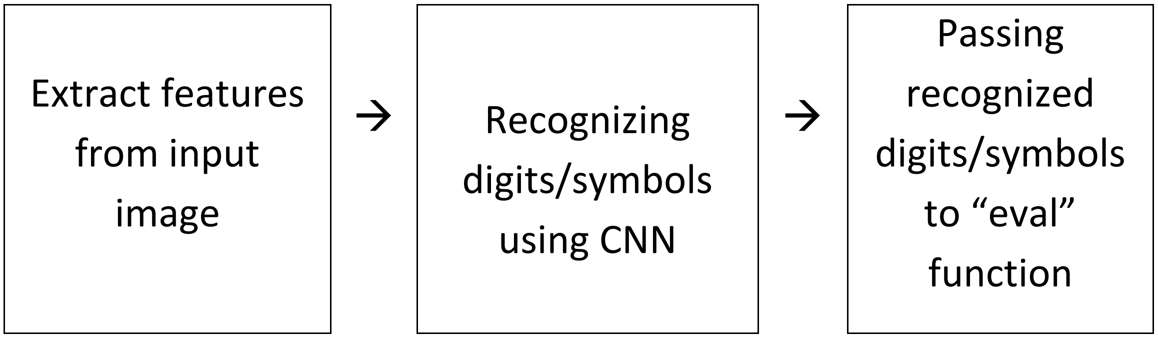
We would like to thank our supervisor *Dr. Arpita Jadhav Bhatt* for her support and encouragement during this project. At many stages in the course of this project. We benefited from her advice, particularly so when exploring new ideas. Her positive outlook and confidence in our work inspired us and gave us confidence. We consider our self very fortunate for being able to work with a very considerate and encouraging professor like her.

INDEX

BACKGROUND

The overall idea behind this project is to develop a machine learning model for recognizing digits and symbols of a mathematical expression written by freehand on a paper(or made in some software like paint), and getting the answer of the expression.

How the project must go



Extract features from input image → Recognizing digits/symbols using CNN → Passing recognized digits/symbols to “eval” function

MOTIVATION

For children in lower classes it is a very useful tool to help them learn. It can be used by them to get answers to basic math problem or what we used to call mental math.

In almost every field, we need to do some calculations either small or large those are just bare numbers but are long and time consuming. This tool can be used there also to save time and money.

PROJECT DESIGN

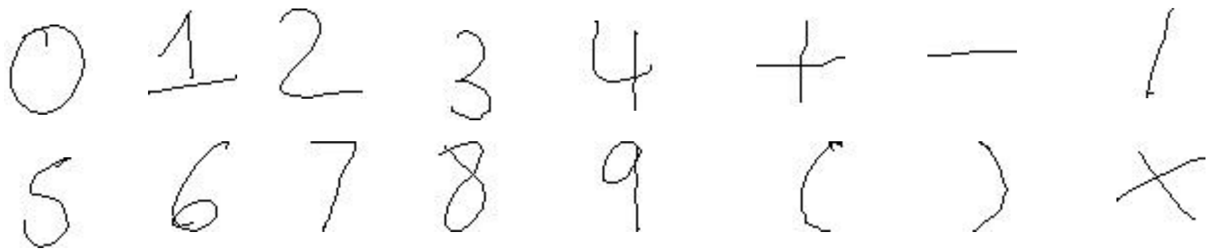
Dataset:

The dataset used in making of the model was taken from Kaggle.com that contains a large database of various data. We got our dataset of mathematical symbols and digits from the same.

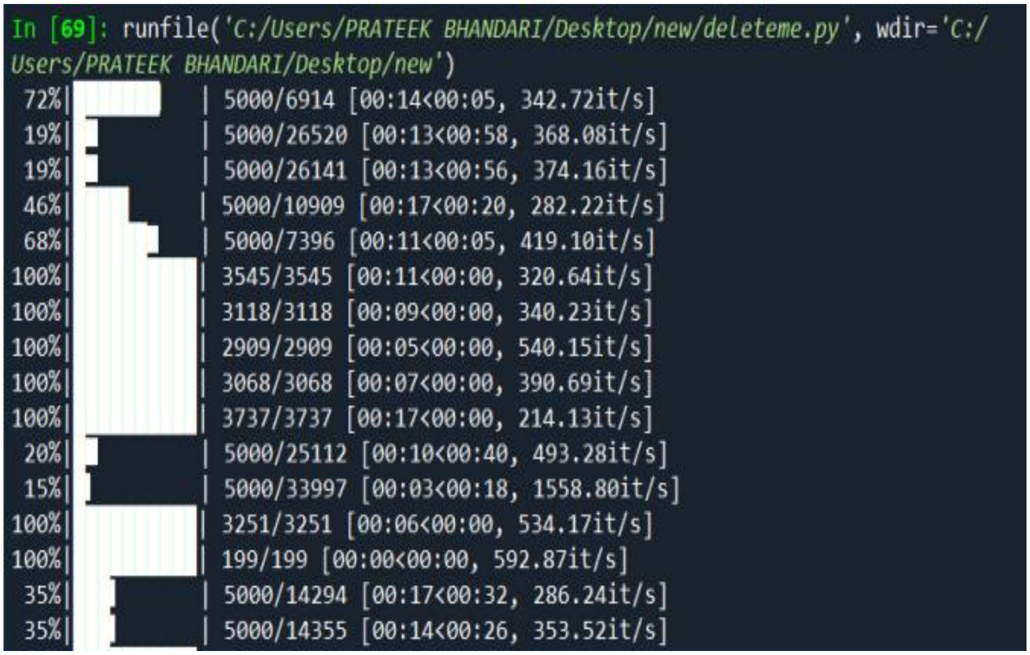
From the dataset the values we took form the project were

- No of classes = 16
- Classes = [0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , + , - , time ,forward_slash , (,)]

Samples of each class:-



The number of images taken of each class has a maximum value of 5000 each (due to less number of images in some classes the data is less in them).The adjacent image shows the number of images used in each class. The order here is same as above written in classes array



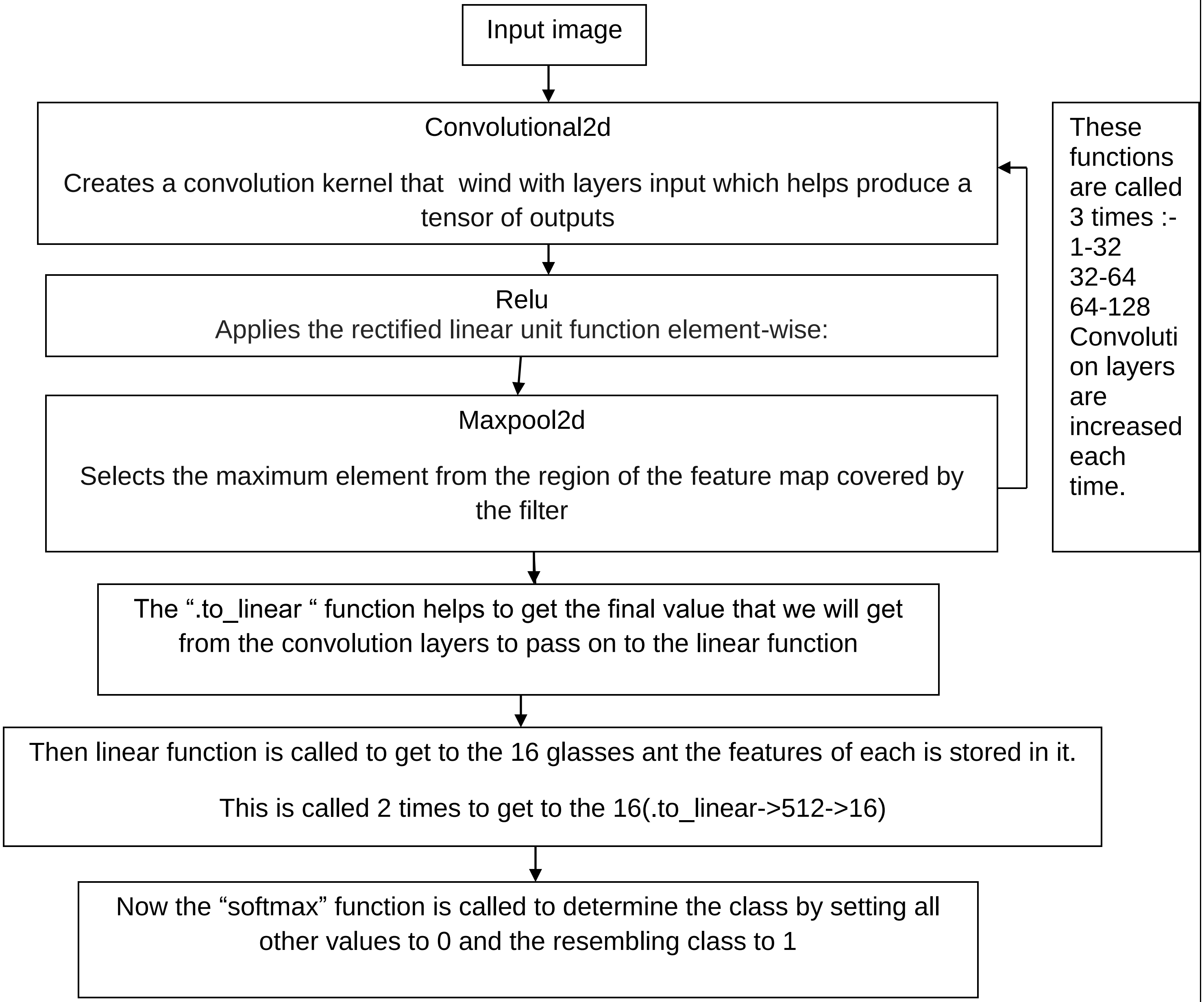
```
In [69]: runfile('C:/Users/PRATEEK BHANDARI/Desktop/new/deleteme.py', wdir='C:/
Users/PRATEEK BHANDARI/Desktop/new')
 72%|          | 5000/6914 [00:14<00:05, 342.72it/s]
 19%|          | 5000/26520 [00:13<00:58, 368.08it/s]
 19%|          | 5000/26141 [00:13<00:56, 374.16it/s]
 46%|          | 5000/10909 [00:17<00:20, 282.22it/s]
 68%|          | 5000/7396 [00:11<00:05, 419.10it/s]
100%|          | 3545/3545 [00:11<00:00, 320.64it/s]
100%|          | 3118/3118 [00:09<00:00, 340.23it/s]
100%|          | 2909/2909 [00:05<00:00, 540.15it/s]
100%|          | 3068/3068 [00:07<00:00, 390.69it/s]
100%|          | 3737/3737 [00:17<00:00, 214.13it/s]
 20%|          | 5000/25112 [00:10<00:40, 493.28it/s]
 15%|          | 5000/33997 [00:03<00:18, 1558.80it/s]
100%|          | 3251/3251 [00:06<00:00, 534.17it/s]
100%|          | 199/199 [00:00<00:00, 592.87it/s]
 35%|          | 5000/14294 [00:17<00:32, 286.24it/s]
 35%|          | 5000/14355 [00:14<00:26, 353.52it/s]
```

The dataset is broken in 9:1 ratio of training data to testing data.

Machine Learning Model:

We constructed a Convolutional Neural Network for the detection of the digits and the symbols.

Flow of the CNN model -



Input image

Convolutional2d

Creates a convolution kernel that wind with layers input which helps produce a tensor of outputs

Relu
Applies the rectified linear unit function element-wise:

Maxpool2d

Selects the maximum element from the region of the feature map covered by the filter

These functions are called 3 times :-
1-32
32-64
64-128
Convolution layers are increased each time.

The ".to_linear " function helps to get the final value that we will get from the convolution layers to pass on to the linear function

Then linear function is called to get to the 16 glasses ant the features of each is stored in it.

This is called 2 times to get to the 16(.to_linear->512->16)

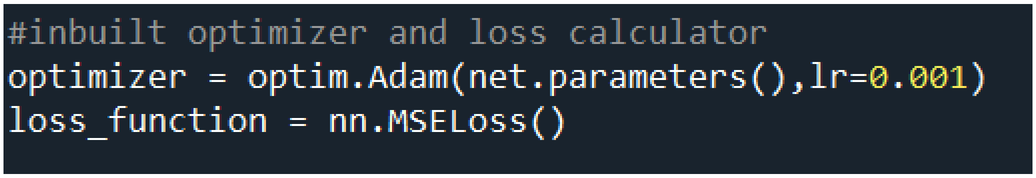
Now the "softmax" function is called to determine the class by setting all other values to 0 and the resembling class to 1

Data Preparation and Training functions:

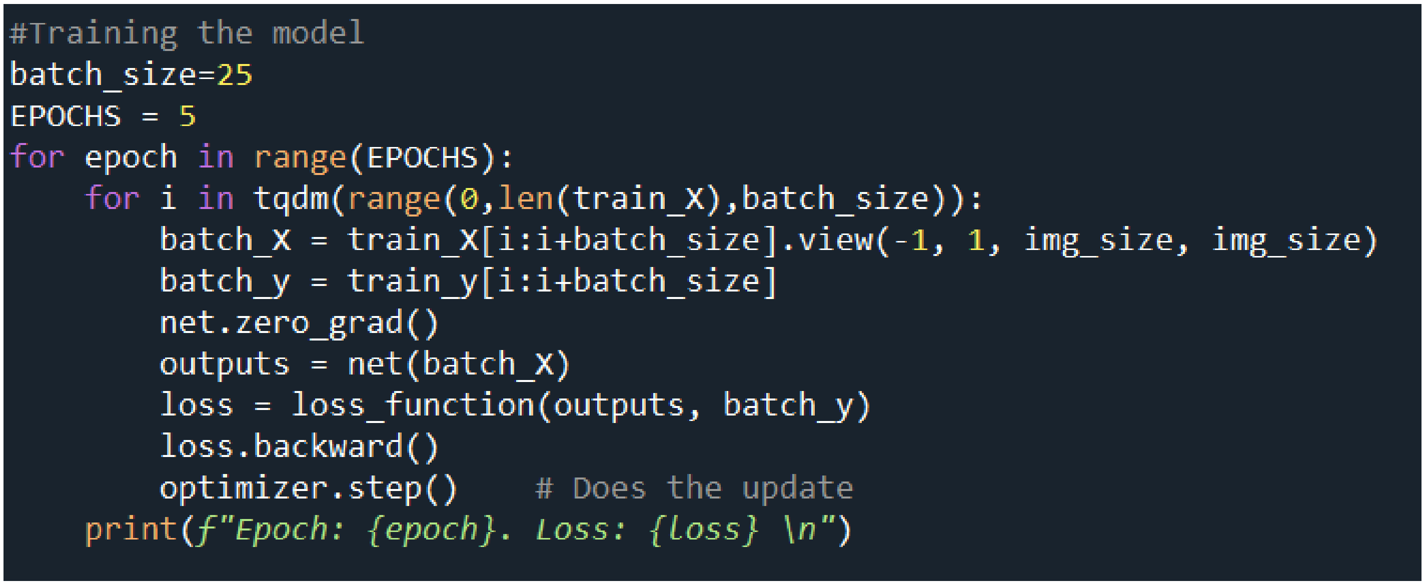
The images are converted into grayscale and resized to 75*75. Then they are stored in a list and converted to one hot vector(In natural language processing, a one-hot vector is a **1 × N**

matrix (vector) used to distinguish each word in a vocabulary from every other word in the vocabulary. The vector consists of 0s in all cells with the exception of a single 1 in a cell used uniquely to identify the word), here is used to get the specific digit form the input. Now the list is stored in a numpy array for future use.

Now during the training:-



```
#inbuilt optimizer and loss calculator
optimizer = optim.Adam(net.parameters(),lr=0.001)
loss_function = nn.MSELoss()
```



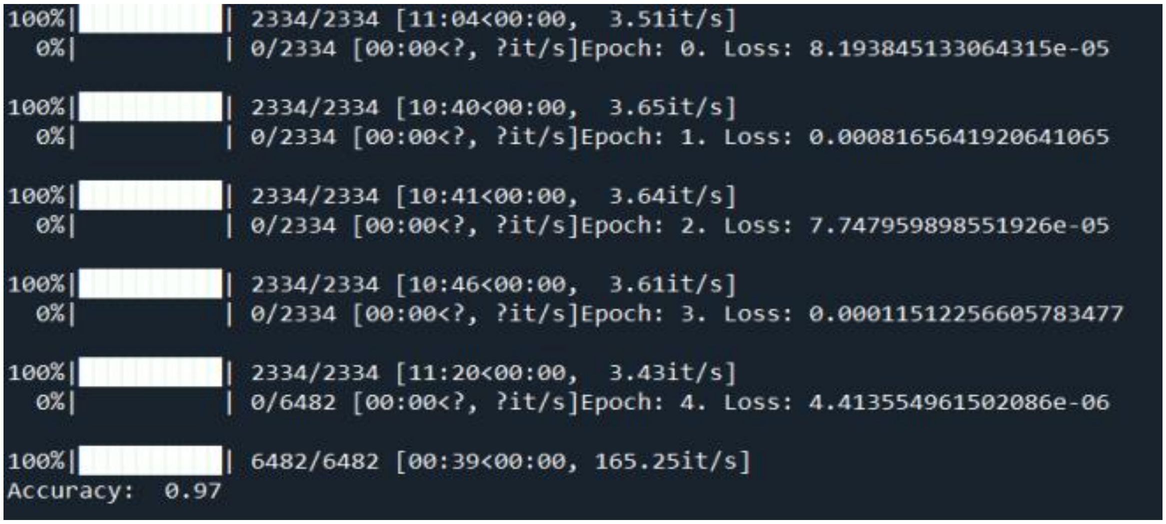
```
#Training the model
batch_size=25
EPOCHS = 5
for epoch in range(EPOCHS):
    for i in tqdm(range(0,len(train_X),batch_size)):
        batch_X = train_X[i:i+batch_size].view(-1, 1, img_size, img_size)
        batch_y = train_y[i:i+batch_size]
        net.zero_grad()
        outputs = net(batch_X)
        loss = loss_function(outputs, batch_y)
        loss.backward()
        optimizer.step()    # Does the update
    print(f"Epoch: {epoch}. Loss: {loss} \n")
```

torch.optim.Adam optimizer is used with a learning rate of 0.001.

torch.nn.MSELoss is called for loss calculation.

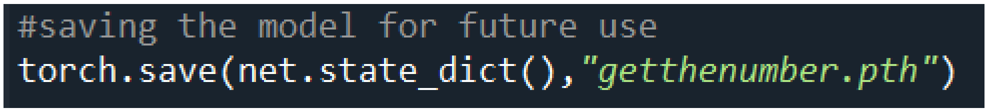
net.zero_grad is called to set all the gradients of the parameters to zero .(here net is the model name)

Result of the traning is :-



```
100%|██████████| 2334/2334 [11:04<00:00,  3.51it/s]
  0%|          | 0/2334 [00:00<?, ?it/s]Epoch: 0. Loss: 8.193845133064315e-05

100%|██████████| 2334/2334 [10:40<00:00,  3.65it/s]
  0%|          | 0/2334 [00:00<?, ?it/s]Epoch: 1. Loss: 0.0008165641920641065

100%|██████████| 2334/2334 [10:41<00:00,  3.64it/s]
  0%|          | 0/2334 [00:00<?, ?it/s]Epoch: 2. Loss: 7.747959898551926e-05

100%|██████████| 2334/2334 [10:46<00:00,  3.61it/s]
  0%|          | 0/2334 [00:00<?, ?it/s]Epoch: 3. Loss: 0.00011512256605783477

100%|██████████| 2334/2334 [11:20<00:00,  3.43it/s]
  0%|          | 0/6482 [00:00<?, ?it/s]Epoch: 4. Loss: 4.413554961502086e-06

100%|██████████| 6482/6482 [00:39<00:00, 165.25it/s]
Accuracy:  0.97
```

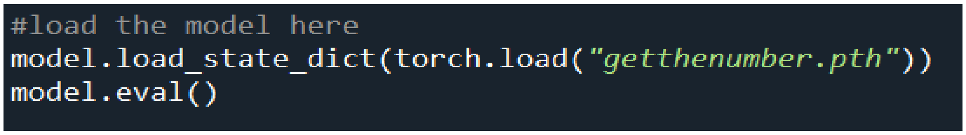
Now the model is trained and saved to be used in the detection process in the future.



```
#saving the model for future use
torch.save(net.state_dict(),"getthenumber.pth")
```

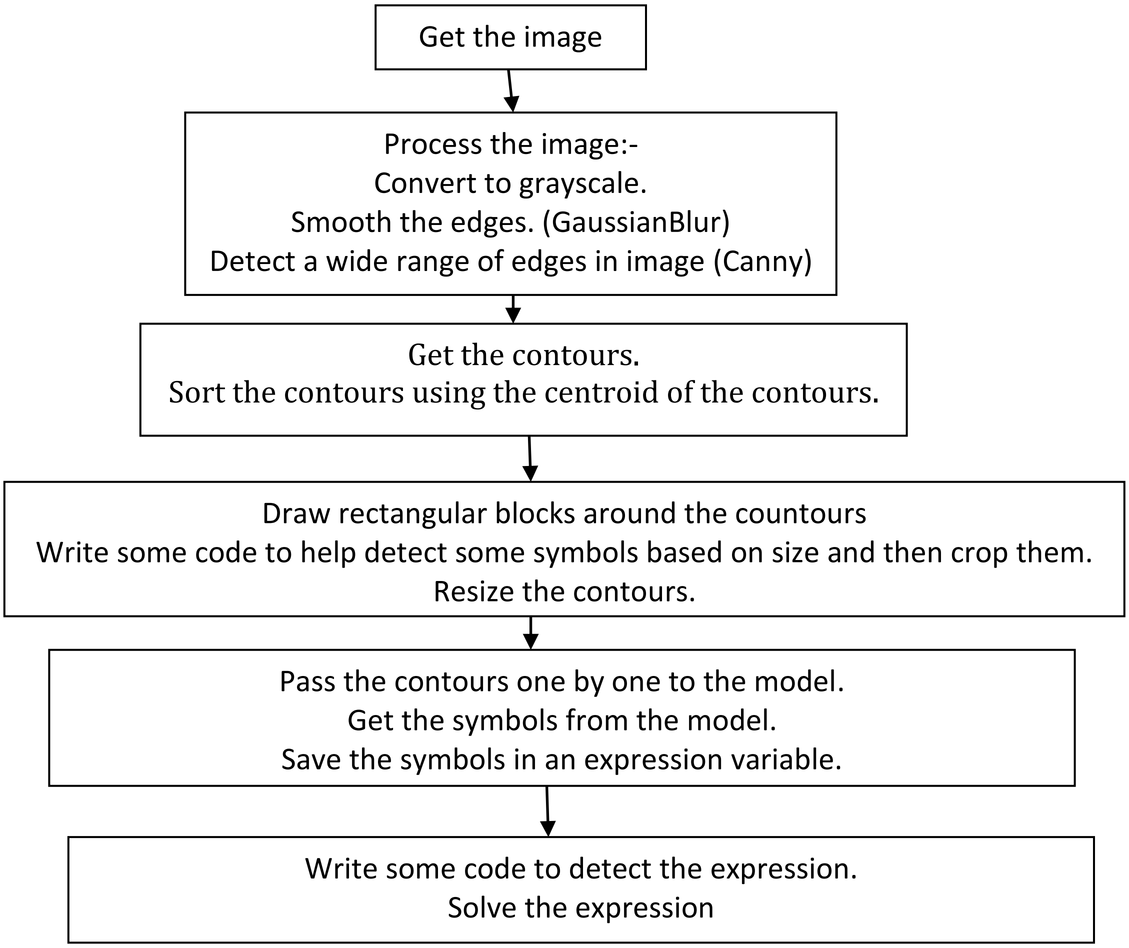
Get the answer from a image(Detecting the Expression):

First of all load the model and paste ML model made previously:-



```
#load the model here
model.load_state_dict(torch.load("getthenumber.pth"))
model.eval()
```

Flow of detection:-



Get the image

Process the image:-
Convert to grayscale.
Smooth the edges. (GaussianBlur)
Detect a wide range of edges in image (Canny)

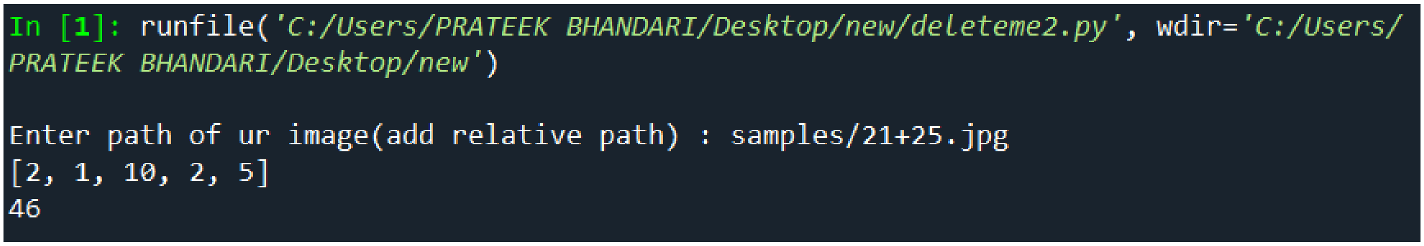
Get the contours.
Sort the contours using the centroid of the contours.

Draw rectangular blocks around the countours
Write some code to help detect some symbols based on size and then crop them.
Resize the contours.

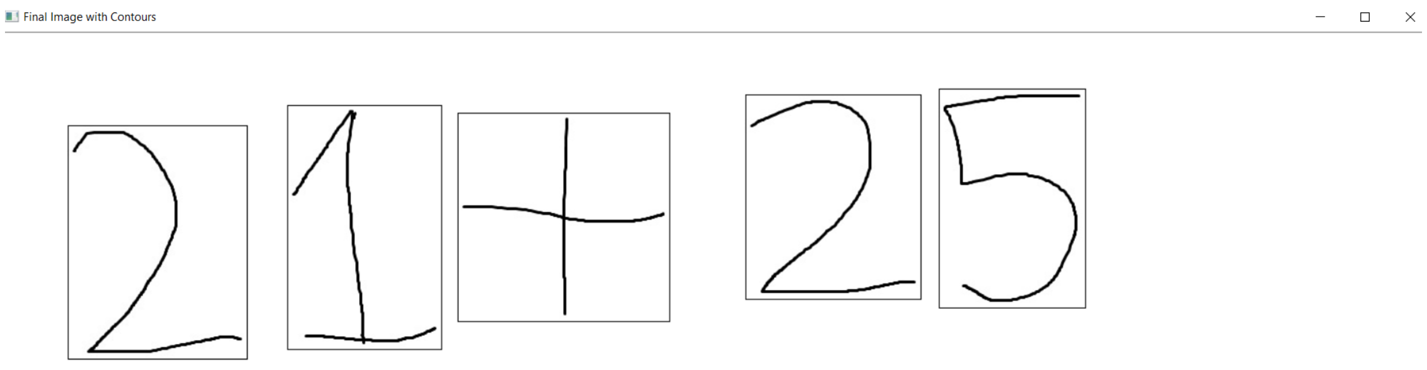
Pass the contours one by one to the model.
Get the symbols from the model.
Save the symbols in an expression variable.

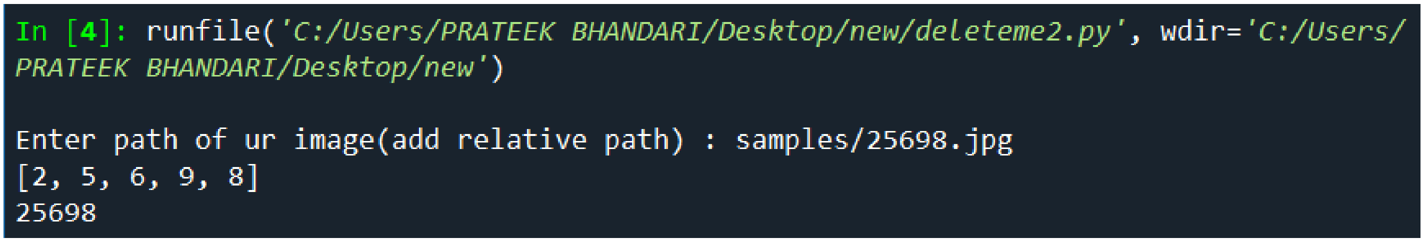
Write some code to detect the expression.
Solve the expression

Some of the outputs:-

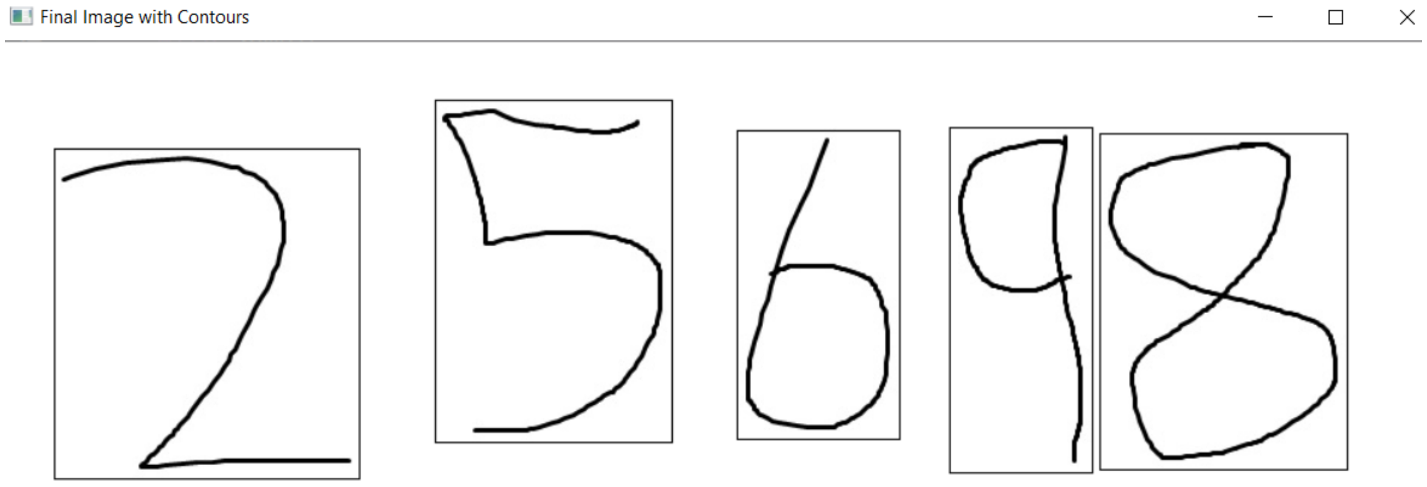


```
In [1]: runfile('C:/Users/PRATEEK BHANDARI/Desktop/new/deleteme2.py', wdir='C:/Users/
PRATEEK BHANDARI/Desktop/new')

Enter path of ur image(add relative path) : samples/21+25.jpg
[2, 1, 10, 2, 5]
46
```





```
In [4]: runfile('C:/Users/PRATEEK BHANDARI/Desktop/new/deleteme2.py', wdir='C:/Users/
PRATEEK BHANDARI/Desktop/new')

Enter path of ur image(add relative path) : samples/25698.jpg
[2, 5, 6, 9, 8]
25698
```



PROJECT SPECIFICATIONS

Features Build, Language Used

- Python as our programming language.
- Library and Function Used :-
    1. torch :- Machine Learning Library
    2. cv2 :- for image modification and processing
    3. os :- for accessing image path
    4. torch.nn :- used for accessing neural network layer(fully connected layers, loss function , linear for linear layers ,conv2d for convolutional layers ..etc)
    5. torch.nn.functional :- contains activation functionds eg. Softmax, relu(activation function for neurons) and maxpooling2d.
    6. torch.optim :- for inbuilt optimization function.
    7. numpy :- for efficiently working with arrays and to store training data that is to be loaded
- Using the concept of Counter and Bounding Rectangle to recognize digits /symbols using CNN.
- The model is unable to detect difference between 2 and 7 and / and 1 due to bad dataset, loss in contour areas and bias because of limited dataset. Resulting in some errors.
- Was trying to build front end but php is not running multiple command prompt commands at a time.

REFERENCES AND RESEARCH PAPER

- Pytorch tutorial by sentdex - YouTube.
- Multiple questions asked and tried on stack overflow
- Autograding-handwritten-mathematical-worksheets (github.com) A ML project for the same topic but done differently