Machine learning based Handwritten Equation Solver

***Eesha Sougandheeka Paladi Navya Sree Pratheek R Kaushik B Nachiketh Reddy***

*ECE department ECE department ECE department ECE department*

*Pes university Pes university Pes university Pes university*

*Bengaluru, India Bengaluru, India Bengaluru, India Bengaluru, India*

***Abstract*—Robust handwritten character recognition is the most important topic in the field of image processing. In this paper, we present a system in which machines may learn to grasp problems based on their visual context. Using a handwritten text from sketchpad as input and returning a solution as output. A handwriting recognition system was built using the learning model to recognize more difficult input. A single system handles pre-processing, segmentation, and character recognition.**

**Keywords: OCR (Optical Character Recognition), CNN (Convolution Neural Network), Image Segmentation.**

1. **INTRODUCTION**

Handwriting recognition is a crucial issue in machine learning and computer vision. A variety of techniques and methodologies have been proposed but it is still an unresolved issue [1]. However, it is a challenging task especially handwriting recognition on form document. It is more complicated than only character recognition. Some noise like a bounding box on form document makes the handwriting recognition systems more complex [2]. Mathematical formulas, as is widely known, are a significant aspect of science reference and the primary language of research communication. It is significant that an automatic record of mathematical formulas be kept to speed up the circulation, storage, and retrieval of scientific references. Mathematical formulas may store a significant amount of data.[1]

Convolutional Neural Network (CNN) is one of the deep learning architectures. It extracts multiple features from low- features to high features automatically [3]

The application will be able to predict and solve handwritten mathematical equations from the sketch pad. The system can solve expressions involving arithmetic operations (addition, subtraction, multiplication, division) and solve equations of any degree (linear, quadratic, cubic and so on).

Key AI concepts used include OCR (Optical Character Recognition) and CNN (Convolutional Neural Networks). OCR is used to preprocess the image and segment characters, while CNN is used to predict the characters.

1. **DATASET**

EMNIST By Class: 814,255 characters. 62 unbalanced classes. EMNIST By Merge: 814,255 characters. 47 unbalanced classes. EMNIST Balanced: 131,600 characters. 47 balanced classes. EMNIST Letters: 145,600 characters. 26 balanced classes. EMNIST Digits: 280,000 characters. 10 balanced classes. EMNIST MNIST: 70,000 characters. 10 balanced classes.

The full complement to the NIST Special Database 19 is available in the By Class and By Merge splits. The EMNIST Balanced dataset contains a set of characters with an equal number of samples per class. The EMNIST Letters dataset merges a balanced set of the uppercase and lowercase letters into a single 26-class task. The EMNIST Digits and EMNIST MNIST dataset provide balanced handwritten digit datasets directly compatible with the original MNIST dataset. Extended MNIST (EMNIST), follows the same conversion paradigm used to create the MNIST dataset[5].

1. **METHODOLOGY**

**A) CNN MODEL:**

CNN contains 3 layers: convolution layer,ReLU layer, Pooling Layer.

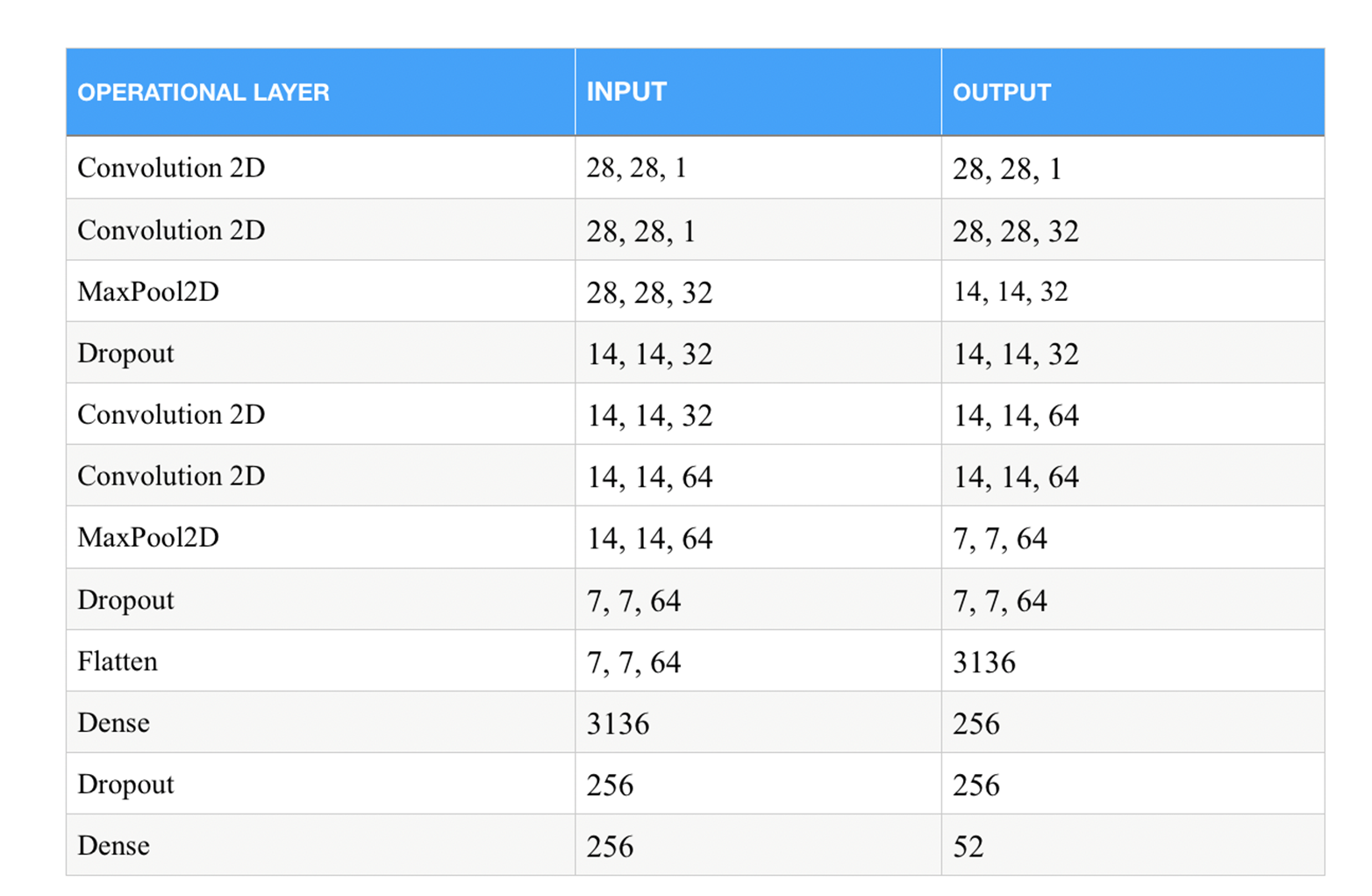
Convolution Layer: Multiplying each pixel of the image and adding (convolution) within the window and sliding the window.

ReLU: It is an activation function which retain values above threshold and 0 below threshold.

Pooling: Shrinking of image is done, highest pixel among neighboring pixels is updated.

The CNN model used has 128 layers for flattening, 64 layers for normalization with a learning rate of 0.01 and a loss function of categorial cross entropy.

Relu activation function and SoftMax, RMSprop optimizer are used.



We trained a model using EMNIST dataset and stored it in model.h5 file.

**B) PRE-PROCESSING:**

The major steps include: Noise Removal, Binarization, Thresholding and Image Segmentation.

Image read is converted to grayscale.

Image resizing is done where width is 1320 and height is calculated using formula

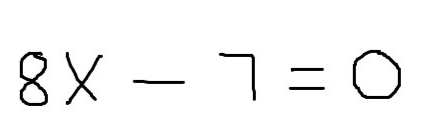
Height = width\*original height/original width

Resized image is binarized using adaptive thresholding.

Noise is removed from the binarized images using morphological closing operation using an ellipse kernel of 3\*3 size.

i)Character Segmentation

The Binarized image and the segmented images can be viewed below:





**C)SOLVING EQUATION:**

After each of the character in the image is detected, the string containing the equation is passed to this final module which solves the equation or mathematical expression.

The equation can be of two types:

A mathematical string such as ‘5+3’ or ‘66x3+2’ (String that is input to this module is of this format). This string can either be evaluated using a custom-built function or the eval () function in python.

A mathematical equation of any degree. The string ‘X2+5=0’ is interpreted as X\*\*2 + 5 since the 2 appears after the variable. Whereas 2X+5=0 is interpreted as 2\*X + 5 = 0. Since prediction of even a single character led to incorrect results/failure, simple replacements are performed on the given string to increase accuracy. These include Z -> 2, G -> 6, B -> 8 and D -> 0. The equation is solved using the SymPy library, which is a python library for symbolic computation.

The 2 types of equations are distinguished by checking if the equation contains ‘= ‘. If the equation contains ‘= ‘, it is interpreted as the 2nd type, otherwise it is interpreted as the 1st type.

The segmented images are updated to a directory called segmented from where the images for character recognition and calculation of the output are taken. The app.py file helps for the calculation of the mathematical equation by requesting the required operation from calculator.py.

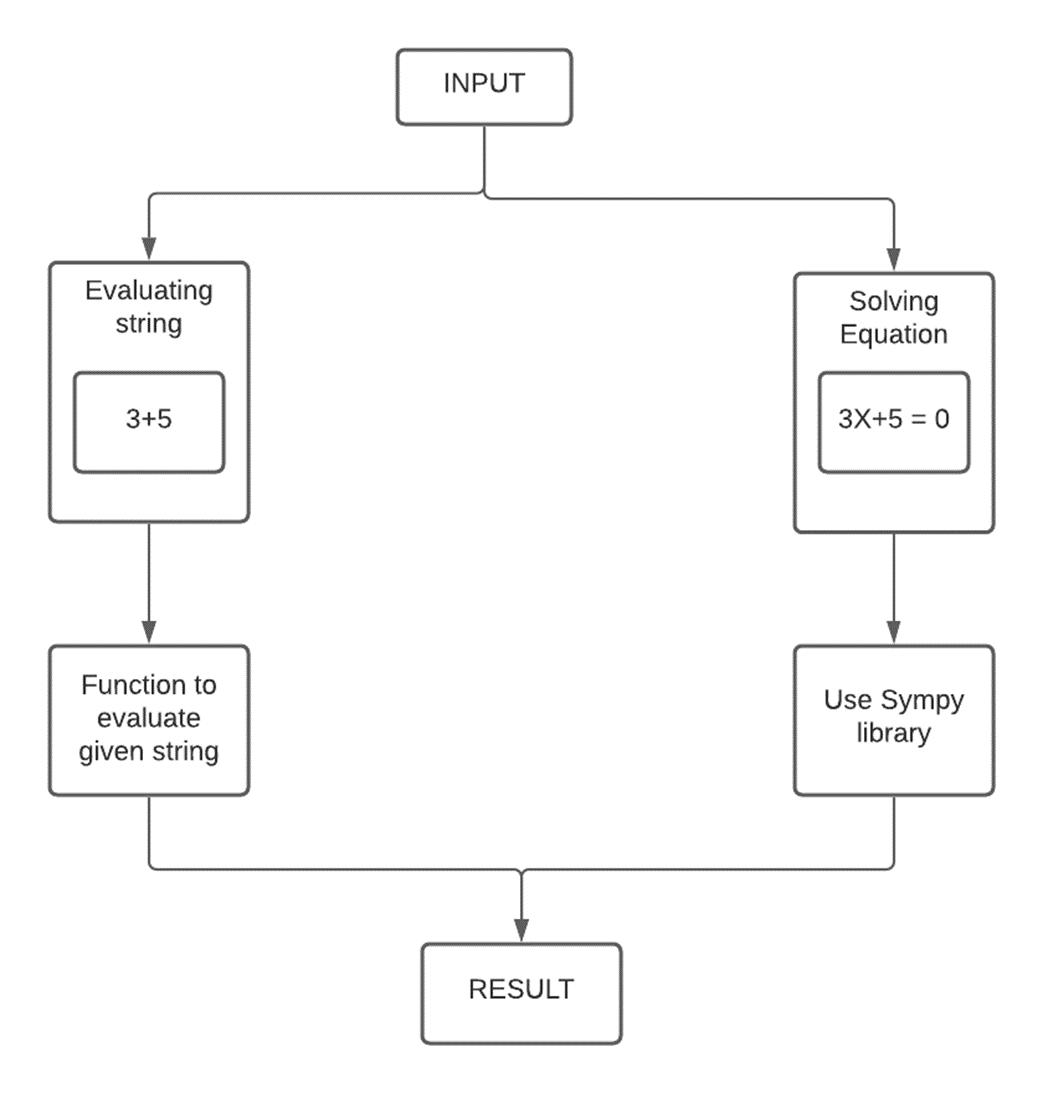
The calculator.py file consists of 2 methods

->solver

->solve\_meThis

The solver method performs the basic arithmetic expressions whereas the solve\_meThis function solves the equations (which contains ‘=’).

Both these functions return the output to the app.py and this displays the output using the post method.



**D)WEBSITE:**

The Frontend part has been developed using ReactJS. Here the user enters the equation or expression by using the sketchpad. The image is encoded to base64 format and sent to the REST-API as a POST request.

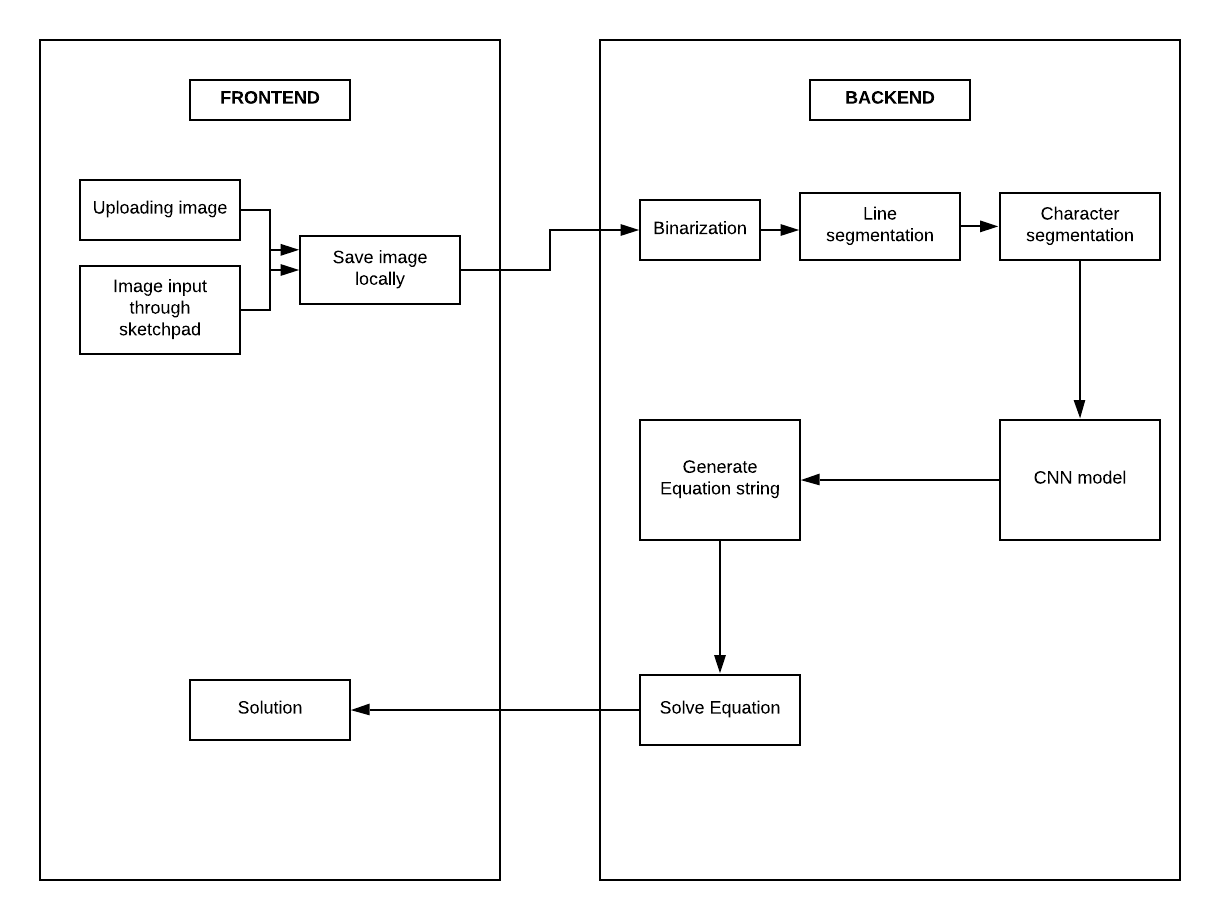
The REST-API has been implemented using Flask. The request data is decoded and saved as an image locally and this image is sent to the backend where the equation is predicted and solved.

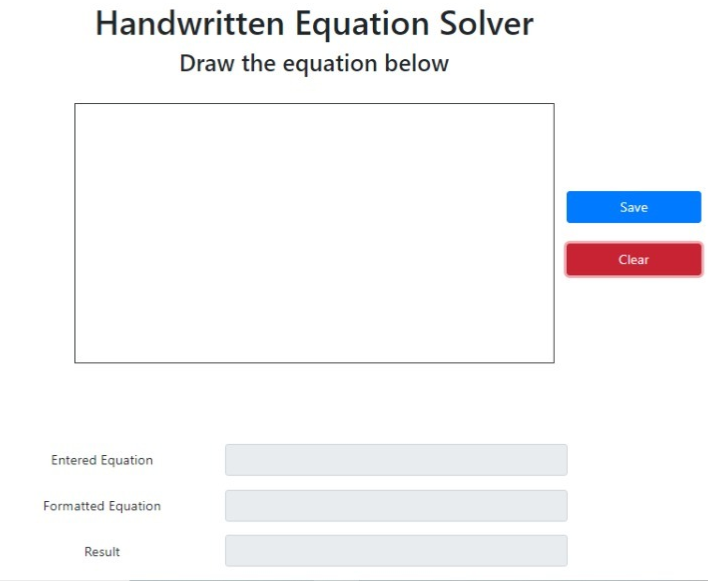
The Backend has been implemented using Python, TensorFlow and OpenCV. The backend can be seen as two separate modules : Equation Prediction and Equation Solver.

OpenCV is used to perform binarization and line and character segmentation. A TensorFlow model trained using the EMNIST (Extended MNIST) dataset is used to predict each of the segmented characters and the equation generated is passed as a string to the Equation Solver.

The Equation Solver solves the mathematical equation and passes it back to the Frontend where it can be viewed.

Block diagram:

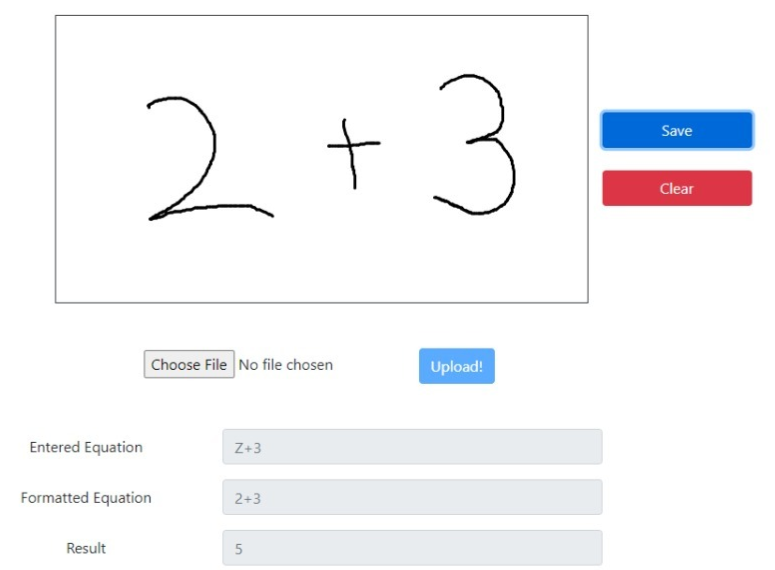




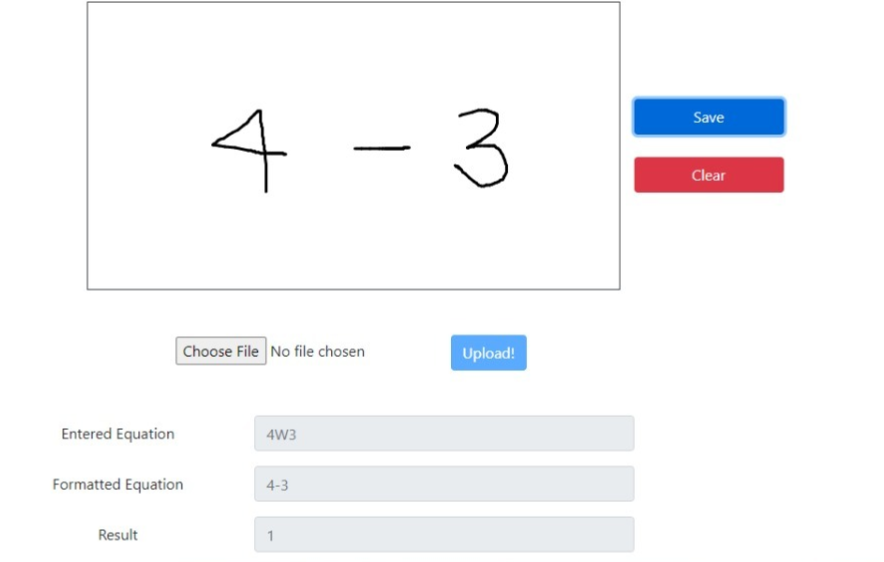
**E) RESULTS:**

Arithmetic Operations:

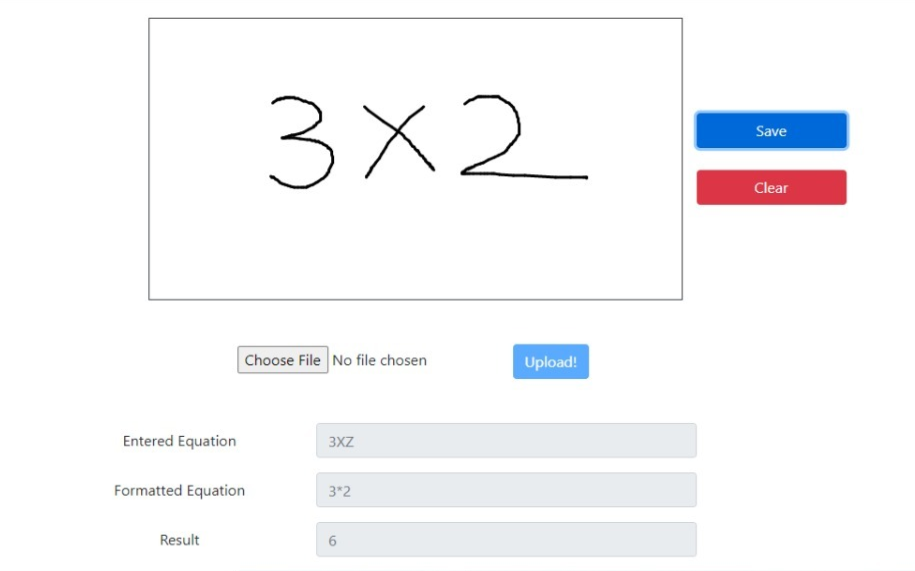
i)Addition:



ii)Subtraction:

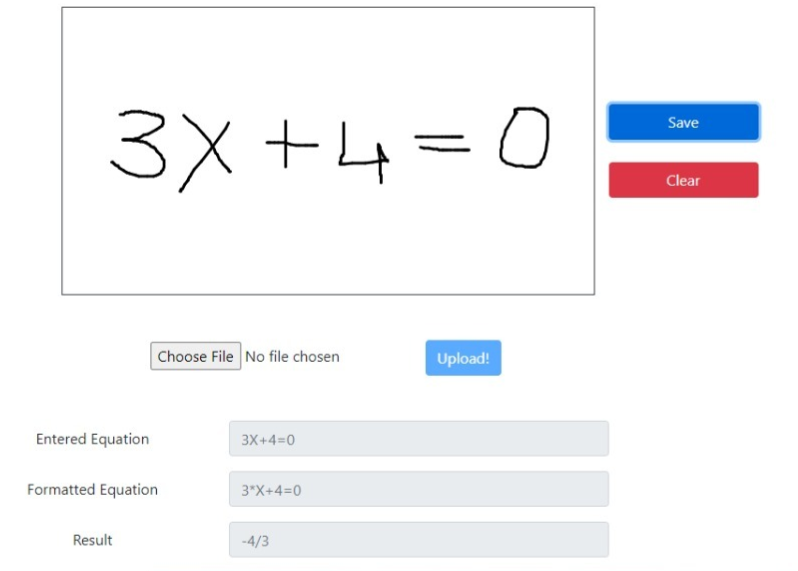


iii)Multiplication:

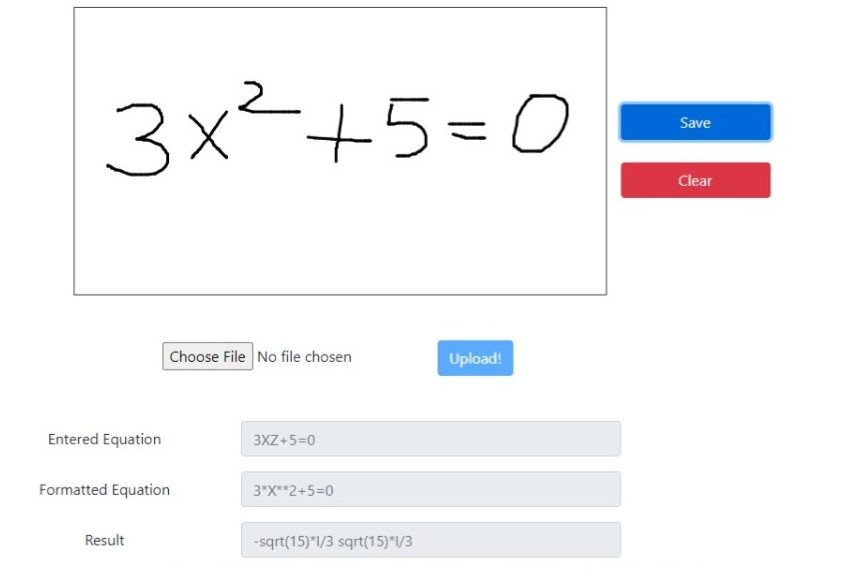


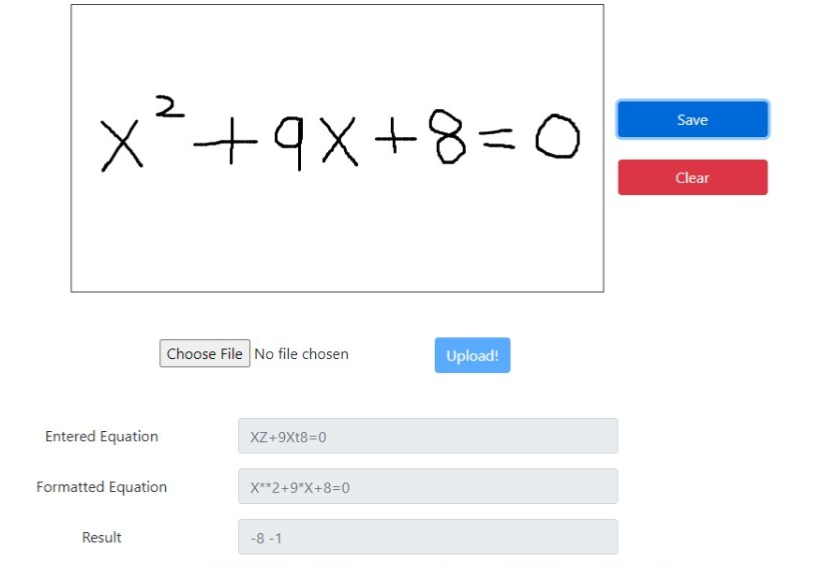
Equation solving:

i)Linear:



ii)Quadratic:





1. **CONCLUSION**

In real world applications this mathematical equation solver is convenient for evaluating and grading during exams. Students can write exams using sketch pad as it is cumbersome to type equations in the limited time provided and this equation solver becomes handy for the evaluator to evaluate marks. The accuracy we achieved using the model is 85% which can be improved using Deep CNN instead of CNN and we limited our project to solve basic arithmetic expression and equations which can be improved further for complex equations.

1. **REFERENCES**

[1] *S. N. Shuvo, F. Hasan, S. A. Hossain, and S. Abuja,* "Handwritten Polynomial Equation Recognition and Simplification Using Convolutional Neural Network," *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-6, doi: 10.1109/ICCCNT49239.2020.9225587.*

[2] Darma Tasia, Fanany, Mohamad Ivan, “Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM)”*.* (2017). *[IEEE 2017 5th International Conference on Information and Communication Technology (ICoIC7) - Melaka, Malaysia (2017.5.17-2017.5.19)] 2017 5th International Conference on Information and Communication Technology (ICoIC7), (), 1–6.* doi:10.1109/ICoICT.2017.8074699

[3] *Lecun Yann and Y. Bengio*, “Convolutional Networks for Images, Speech, and Time-Series,” *Handb. Brain Theory Neural Netw., pp. 255–258., 1998*

[4] *Yonghua Li, KeJun Wang, Wei ShangGuan and LiQun Tang, “*The Research of Mathematical Formula Recognition Method Base on Baseline Structure Analysis”, *2008 International Conference on Internet Computing in Science and Engineering* *Automation College, Harbin Engineering University, Harbin, Heilongjiang Province, China* [*tangliqun@hrbeu.edu.cn*](mailto:tangliqun@hrbeu.edu.cn)

[5] <https://arxiv.org/abs/1702.05373v1>