1. INTRODUCTION

Discovery, that is the word we associate with finding something new. The reason for this is that this universe already has everything it requires, it's just in a form that mankind has yet to comprehend or discover. Mathematics is the simplest way to understand the universe's transformations and intricacies. The language of mathematics which can describe the nature of the universe to the shapes and structure and pattern of almost all objects that we perceive through our senses. And today we have reduced the time gap to get answers for tedious methods and procedures with the help of new technologies and gadgets. The most general example would be a calculator which is used instead of manually working out mathematical methods thus reducing the percentage of human error associated with it. But using the calculator to solve the humongous equations and find answers for them is a tedious task since it requires the equations to be entered accurately.

Therefore, another need for innovation came to life and thus technology which could scan documents and interpret what is written and further find the answer for it to ease the daily humdrum. Mathematical equations form an integral part of most research work, so researchers use such mathematical tools to save time and boost efficiency keeping in mind the complexities and syntax of the chosen tool. If the syntax rules are not followed properly, one might not get the desired output. This is precisely the tool we have developed and discussed throughout this paper. Computers can be made to think and act like a human would, by forming a neural network and also retaining their computational superiority. In our project, we have attempted to simplify the interaction between humans and computers pertaining to the processing and solving mathematical equations.

We have focused on building a model which is simple to understand, by not restricting the user with various syntax rules and complexities. In today’s world there are many tools, simulators which are available for mathematical purposes. However, more the features, more the awareness about the particular tool is required from the user. To shorten this gap of excessive awareness on the user’s side, the machine could be trained to understand the user’s needs more accurately with very less effort from the concerned user. This being the ideology we have tried to build a machine learning model which will take the image input of the mathematical expression written by the user and identify the input expression and provide the answer with accuracy achieved while building the model.

1.1 Motivation of The Work

Mathematical equation recognition is one of the challenging tasks in the field of computer vision. In single digit or symbol recognition, a dense layer can give a test score of 99% when trained on 70,000 numbers between 0-9 which is the MNIST dataset.

We have chosen the project mathematical equation solver so that students would get facility to check the answer of their equations without depending on others. If they can check that the solution is correct or not they can move to solve other equations with confident. If the students are not sure about the correct solution then they might get demotivated and leave the work.

1.2 Aim

The aim of the project is to create the system to solve the simple mathematical equation. It intends to make student learn themselves even if there is no guidance and check their answers. The system is about to recognize the digits and symbol and provide the final answer.

Collect the required dataset. Research on algorithms and the similar system that are developed. To create a system that recognize handwritten digits and operators. To calculate the problem and display the final solution in screen.

1.3 Objective

Writing mathematical expressions in digital form in a difficult task because of the combination of strokes making up a mathematical expression. Therefore, the objective of this project is to digitize and evaluate the handwritten mathematical expressions.

The objectives of the project are following:

- Segmenting the digits, operators and variables from the input image of a mathematical expression. These characters will be sorted and stored in the same order as they appear in the image of the expression.
- The segmented characters need to be recognized in order to be digitized. This will be achieve using Convolutional Neural Network because of their state performance is the classification of images. of the art. The digitized expression needs to be evaluated.
- Evaluation of the expressions will be done and result is displayed.

2. LITERATURE SURVEY

The Internet is the key to connecting with the world and helping us learn. It is one of the greatest pillars to educate many in some way or the other. But, in the field of mathematics, considering the use of Greek letters and special symbols in mathematical equations, it is difficult to give these as an input to the internet. This issue has been addressed as we have Latex to represent these equations, however using Latex requires some training and intensive practice. On the other-hand, some development is done in recognition of these expressions using machine learning. Mathematical expression recognition typically consists of two major stages: symbol recognition and structural analysis. Both symbol recognition and structure analysis of two-dimensional patterns have been extensively studied for decades.

- **Methods for Lines and Matrices Segmentation in RNN-based Online Handwriting Mathematical Expression Recognition Systems**
  by Oleg Yakovchuk 2020

Shallow Neural Networks (RNN) uses single Hidden layer. RNN is able to achieve this with the help of a hidden layer. It is heavily used in speech recognition, video tagging, generating image descriptions, but it also gives a satisfying performance with a test accuracy of 96% and an error rate of 0.13% when trained on 8,00,000 data points while predicting the solution for simple mathematical operations. Because of the Markov assumption, RNNs with LSTM blocks can integrate the influences of previous nodes without suffering from the gradient vanishing problem, whereas MRFs can only consider neighbouring points.

- **Stroke Extraction for Offline Handwritten Mathematical Expression Recognition**
  By Chungkwong Chan 31 March 2020

A stroke extraction algorithm is required to convert a bitmap image to a sequence of strokes, so that online recognition engines are applicable afterward. The set of strokes is reconstructed by finding a set of paths on a graph-based representation of skeleton, in which vertexes and edges represent junctions and segments respectively. Astroke extraction algorithm is required to convert a bitmap image to a sequence of strokes, so that online recognition engines are applicable afterward. The proposed stroke extraction method consists of three stages: pre-processing, tracing, and postprocessing. Furthermore, given a trainable online recognition system, retraining it with extracted strokes resulted in an offline recognizer with the same level of accuracy. On the other hand, the speed of the entire pipeline was fast enough to facilitate on-device recognition on mobile phones with limited resources. To conclude, stroke extraction provides an attractive way to build optical character recognition software.

- **Online Handwritten Mathematical Expression Recognition and Applications: A Survey**

The research is now centred on DML techniques, ensuing in third generation end to end solutions. Although these solutions have taken the lead in lots of areas, in handwritten Mathematical equation recognition, these techniques are nonetheless at an early phase of implementation and improvement with a few limitations. This survey demonstrates that interest in online HME recognition has been growing over the past 40 years. The development of integrated solutions made it possible to combine various methods (grammatical, statistical) minimizing the accumulation of recognition errors.

- **Handwritten Mathematical Expressions Recognition using Back Propagation Artificial Neural Network**

In the proposed system, recognition of the straight-line equation $y = mx + c$ has been done with better accuracy and recognition rate. the neural network trained using gradient descent with momentum and adaptive learning. Neural Network developed using one input layer, two hidden layers and one output layer. By repeatedly presenting data set to the network performance of the network is increased. The system presented an approach to recognize handwritten MEs such as quadratic equations with improvement in recognition rate, processing time and accuracy. Centroid and bounding box are the key features extracted from each character and uprightness of the system using back propagation neural network.

- **Recognition of Online Handwritten Mathematical Expressions Using Convolutional Neural Networks**

The author discovered that CNNs are an effective method for recognising handwritten expressions. With more time and greater computational resources, it may be able to improve overall expression accuracy dramatically. Because symbol accuracies are compounded at the expression level, small percent increases in accuracy at the symbol level have dramatic effects at the expression level. This approach relies heavily on CNNs, which are widely used for a variety of vision recognition problems. pipeline has five distinct phases data set enrichment, image segmentation, data extraction, character-level classification, and expression-level classification The CNN-based system outperforms SVM-based system by 3-4%.

- **A Survey on Image Classification Approaches and Techniques**

worldwide Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 1, January 2013 Various categorization strategies are taken into account in this study; Decision Tree (DT), Support Vector Machine (SVM), and Fuzzy Classification are all examples of artificial neural networks. It classifies different techniques for image recognition and image classification.

- **A Review of Various Handwriting Recognition Method**

Convolutional Neural Network (CNN), Semi-Incremental, Incremental, Lines and Words Segmentation, Parts based method, Slope and Slant Correction, Ensemble, Zoning. In virtually every situation, CNN's accuracy is excellent. The Slope and Slant Correction Method has the lowest accuracy. In writing recognition, zonation approaches, line and word segments, ensemble methods, and part-based algorithms are all quite reliable. After CNN has been trained, image recognition will be accurate. It has been demonstrated to be effective in a variety of handwriting and computer recognition applications. The % accuracy of each technique may be determined using the technologies, reinforcing the author's conclusion that CNN has the best accuracy.

3. SYSTEM ANALYSIS

3.1 System Study

The focus in this project has been on segmentation and recognition of arithmetic expressions and polynomial equations from an image and then evaluating these successfully recognized expressions. Handwritten mathematical expression recognition is of two types: online and offline. The former form consists of recognizing the characters by their strokes while they are being written on a tablet or smartphone. While the latter form consists of recognizing the characters from an image of a handwritten mathematical equation. The segmentation of '=' and components instead of a '÷' characters results in the detection of separate ingle symbol. This problem is solved by vertically combining the components for the height of the image resulting in a single segmented image of the symbol.

3.2 Existing System

A stroke extraction algorithm is required to convert a bitmap image to a sequence of strokes, so that online recognition engines are applicable afterward. The set of strokes is reconstructed by finding a set of paths on a graph-based representation of skeleton, in which vertexes and edges represent junctions and segments respectively.

The proposed stroke extraction method consists of three stages:

- **pre-processing**
  In pre-processing binarization and thinning are applied to increase the signal-to-noise ratio.
- **Tracing**
  The set of strokes is reconstructed by finding a set of paths on a graph based representation of skeleton, in which vertexes and edges represent junctions and segments respectively
- **Postprocessing**
  In postprocessing stroke direction detection and stroke order normalization are applied to recover remaining temporal information.

3.2.1 Disadvantages

1. Traditional systems of handwriting recognition have relied on handcrafted features and a large amount of prior knowledge.
2. Another disadvantage is BODMAS not applied.
3. Existing System is completely offline.
4. There is no available public dataset of handwritten MEs.
5. Ambiguity of handwriting production, which can be considered as overlapping between classes.
6. Existing System is time Consuming.

3.3 Proposed System

Our approach to the problem can be broken down into three broad steps.

1. The input image provided by the user is pre-processed and segmented to recognize and predict each individual character, number and symbols
2. Recognizing the handwritten equation using CNN and by doing so, digitizing it
3. After recognition the mathematical expression is formed which is further processed using python libraries to provide the calculations.

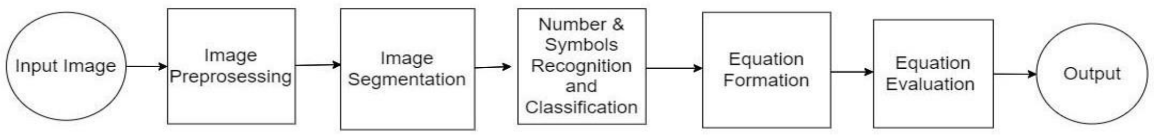


Figure 3.3: Block Diagram

The Block Diagram represents the procedure in which entire processing is carried out.

For Recognition Dense layers are used to extract features from input images with relu as an activation function for the hidden layers and softmax for the final layer for predicting numbers and operators is a multi-class classification problem. This model is trained with an adam optimizer which is reported to have a faster convergence rate than normal stochastic gradient descent (SGD) with momentum.

3.3.1 Convolution Neural Networks

CNN (Convolutional Neural Networks) can be used in handwritten mathematical equation recognition and solving by treating the handwritten equation image as an input and processing it through the layers of the network to extract relevant features and classify the equation components. CNN is successfully applied for natural language processing, recommender system and many other. Detecting the important feature is the major advantage of CNN without human supervision.

The image are classified by taking input of image, processing it and classify it. Input of images are in the form of array of pixel when given input to computer depending upon the image resolution. CNN consists of layers and they areconvolutional layers with filter, pooling layer and fully connected layer. Each image passes through these layers.

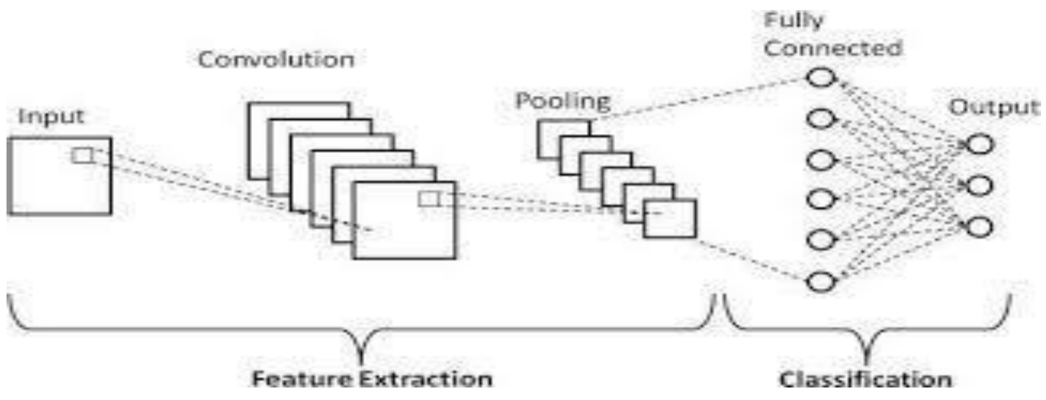


Figure 3.3.1.1: Convolutional Neural Network

Here's a general overview of how CNN can be used in this context:

1. **Pre-processing:** The handwritten equation image is pre-processed to remove any noise, distortion, or background interference.

   The image is then normalized and resized to a fixed size for input into the CNN.

2. **Feature extraction:** The CNN is designed to extract features from the image that are relevant to the recognition and solving of mathematical equations. The layers of the CNN perform convolution, pooling, and activation functions to extract high-level features that can differentiate between different equation components, such as numbers, operators, and symbols.

3. **Classification:** The CNN then classifies the equation components based on the extracted features. The output of the CNN is a sequence of identified components that represent the handwritten equation.

Overall, CNN can be used in handwritten mathematical equation recognition and solving by leveraging its ability to extract relevant features from complex image data and classify components with high accuracy.

Max pooling layer:

Pooling layer Pooling layer is the next layer added after the convolutional layer. Repeated layer for one or more time is ordered with the pooling layer in CNN. To create the new set of pooled feature map of same number operation of pooling take place upon each feature map separately. Pooling operation like filter to be applied is selected in the pooling.

Fully Connected Layer and Output:

First, a Flatten layer is use to convert the final feature maps into a single 1D vector (necessary for Dense layer input). Next, a fully-connected (Dense) layer that acts as an Artificial Neural Network.

Output layer:

The CNN model's final layer stores the results of the labels determined for classification and assigns a class to the images.

3.4 Software Environment

The document contains the functional and non-functional requirements. How the user is going to interact to system is explained. How the system interacts with internal modules, hardware, communication with other program and human user interactions.

3.4.1 Functional Requirements

Functional Requirement defines a function of a system component. A function is depicted as a set of inputs, the conduct, and yields. Functional requirements maybe calculations, technical details, data manipulation and processing and other specific functionality that define what a system should achieve. Behavioural necessities depicting all the situations where the framework utilizes the functional requirements are captured in use cases. In this project some of the functional requirements are:

Upload image: should be able to select and upload image from the file explorer.

Canvas: should be able to draw the equation on canvas and save it.

Undo: should be able to undo the drawn equation by one step.

Clear: should be able to clear the contents to start freshly.

Display: should display the image selected or canvas drawing for cross checking.

Predict: predict the handwritten Mathematical equation and solve it to display on the interface.

3.4.2 Non-Functional Requirements

A Non-Functional requirement is a requirement that determines criteria that can be utilized to judge the operation of a framework, as opposed to particular practices. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture. Some of the non-functional requirements in this project are:

- **Throughput**: Throughput is the maximum rate of production or the maximum rate at which something can be processed.
- **Efficiency**: Efficiency is the ratio of useful work to resources like processor and storage expended.

3.5 Input Design

In this system, input is the Handwritten Equation Image that is processed to produce output. Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −

- It should serve specific purpose effectively such as storing, Displaying, and retrieving the images

- It ensures proper completion with accuracy.

In this project the input is taken through a web interface through uploading an image i.e., offline image selected from the system through an upload Button. The image can be drawn on the canvas board also provided in the interface.

3.6 Output Design

The Design of output is done by providing a button to predict the handwritten equation and a solve button to solve the equation. The recognition and classification of each number is done using CNN and the solution for equation is provided through sympy.

3.7 System Requirements

3.7.1 Software Requirements

- **Operating System:** Microsoft Windows 10 or higher.
- **Front end:** HTML, CSS, JAVASCRIPT
- **Back end:** Flask
- **Language:** Python
- **Tools:** Visual Studio Code
- **Modules Required:**
  1. Tensorflow
  2. OpenCv,
  3. Python Imaging Library
  4. NumPy
  5. Pandas
  6. Matplotlib
  7. Sympy

3.7.2 Hardware Requirements:

CENTRAL PROCESSING UNIT (CPU) : Intel Core i3 8th Generation processor or higher.

RAM : 4 GB minimum, 8 GB or higher is recommended.

4. SYSTEM DESIGN

4.1 Architecture Diagram

The System architecture consists of various components interacting with each other and flow of processes. The proposed system is for educational use where the recognition and solving process of mathematical equations will be done by machine. In this system for recognition of equations, we use a Convolutional neural network (CNN) model. The system is capable of performing fast calculations on human handwritten mathematical expressions.

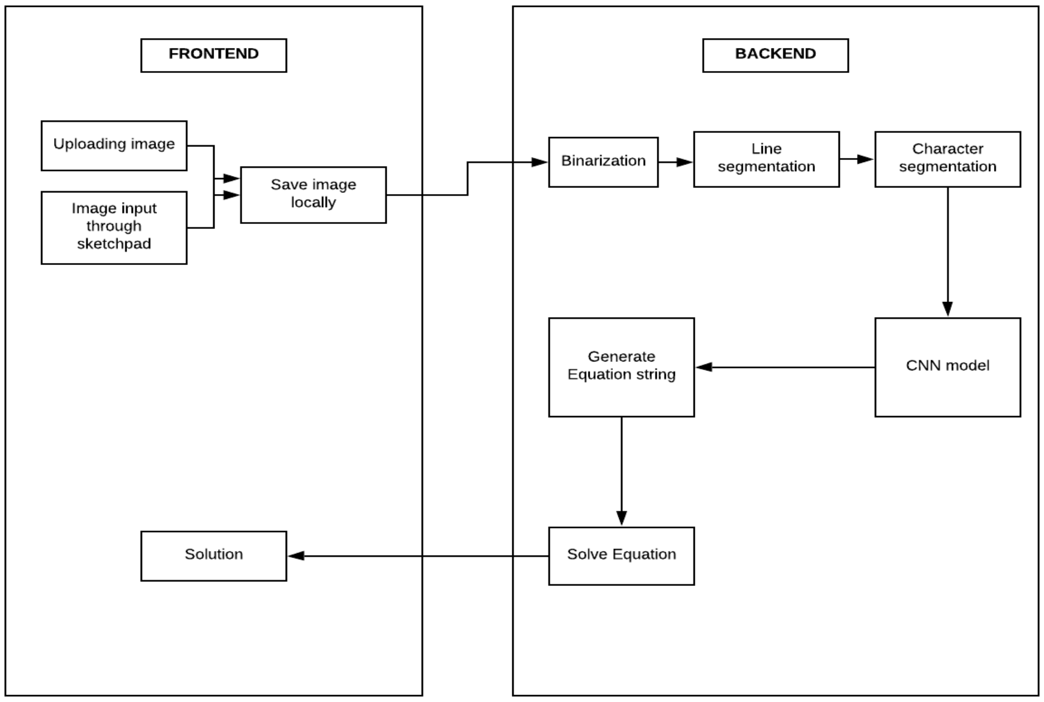


Figure 4.1.1: Architecture Diagram

Frontend:

There are two ways to provide input for the system.

- Uploading image
- Image input through canvas

Uploading image:

- The user can take a picture of handwritten mathematical equation and save the image locally.

- The choose file button is clicked then the user should get the interface of local system option.
- Select the image from local system should and click on the upload button.
- The image displayed on the screen turned into grayscale.
- Then click on the predict button which display the handwritten mathematical equation in the system form.
- Finally click on the solve button it displays the solution of the equation.

Image input through Canvas:

- The user can write the equation on the canvas board and then save the equation.
- Click on the predict button which shows the system recognized equation in the system form.
- Click on the solve button it displays the solution of the equation.

Backend:

- In **Binarization** 1s indicate black pixels and 0s represent white pixels in the horizontal projection computation.
- The **line segmentation** is to separate out the line of text from the image.
- **Character segmentation** helps in separating out the individual characters from the image which is already divided into lines of text in line segmentation
- **Convolutional Neural Network** (CNN), a common deep neural network architecture, may be used to accomplish HME Recognition. Traditional CNN classifiers are capable of learning and classifying essential 2D characteristics in pictures, with the classification accomplished using the soft-max layer.
- After the recognition of the characters and symbols, the expression is passed to the function to solve and display the solution.

4.2 Data Flow Diagrams

A Data Flow Diagram is a traditional visual representation of the information flows within a system. The DFD also provides information about the outputs and inputs of each entity and the process itself.

4.2.1 Level 0 DFD:

- The level 0 DFD shows the overall system as a single process with two external entities: Handwritten Mathematical Equation Recogniser and solver and interface
- The image is taken from the interface and return the equation and solution

The level 0 DFD can only be seen between the components,

- Handwritten Mathematical Equation Recogniser and solver
- Interface

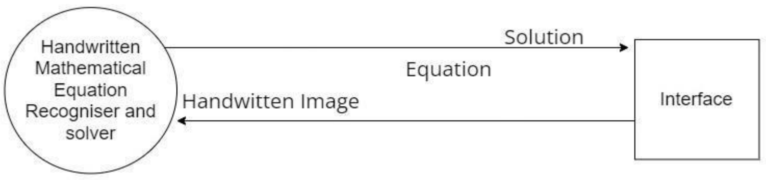


Figure 4.2.1.1: DFD Level 0 Diagram

4.3 UML Diagrams

UML, which stands for Unified Modeling Language, is a way to visually represent the architecture, design, and implementation of complex software systems. UML diagrams divide that software system into components and subcomponents.

4.3.1 Use-Case Diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

Here the actor is the user who interacts with the system by either using the canvas to write a mathematical equation or by selecting the mathematical equation from the system files.

The actors in the Use case diagram are,

- User
- System files
- System

The actors interact with the system which pertains to similar use case generated in the below figure. The user either draws or uploads an image whose result is displayed on the systems screen. Here you can see the user when selects the option to draw he doesn't need any access to the device's local storage, but when the option for uploading image is selected the user invokes the System Files in order to get the image. Never the less, at the end the answer is directly displayed on the System.

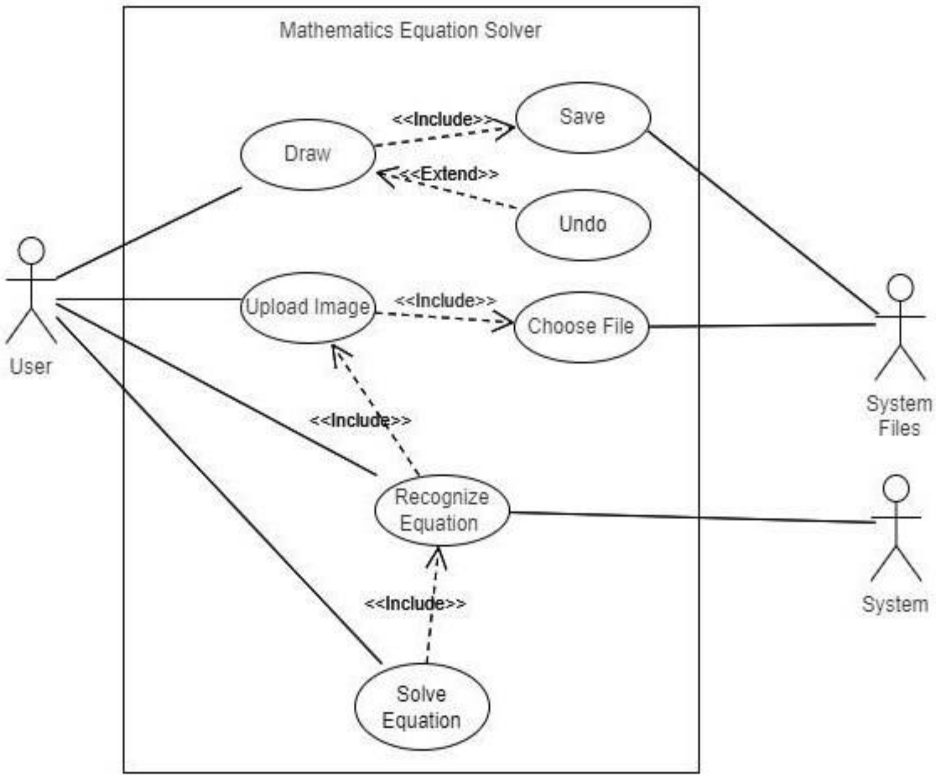


Figure 4.3.1.1: Use Case Diagram

4.2.2 Activity Diagram:

An activity diagram is a type of UML (Unified Modeling Language) diagram that shows the workflow or process of a system or business process. An activity diagram for a handwritten mathematical equation recognizer and solver using CNN can help to illustrate the overall flow of the system. The process starts with an input image of a handwritten mathematical equation.

The below figure depicts how the activity proceeds in the handwritten mathematical equation recognizer and solver using CNN.

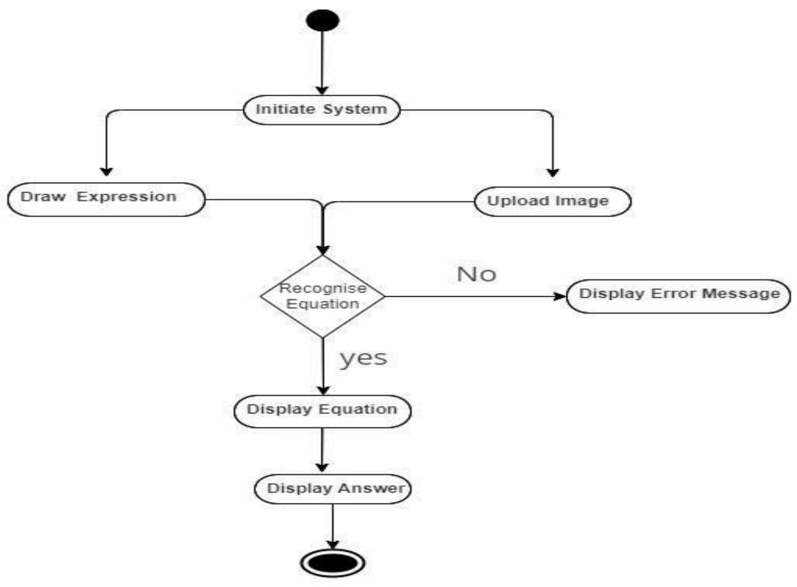


Figure 4.3.2.1: Activity Diagram

4.3.1 Sequence Diagram:

A sequence diagram is a type of interaction diagram that shows the interactions between objects or components in a system over time. The sequence diagram illustrates the interactions between the user and the handwritten equation solver system. It also shows the interactions between the user, image recognition model, pre-processing component, and equation solving component during the process of recognising and solving a handwritten mathematical equation.

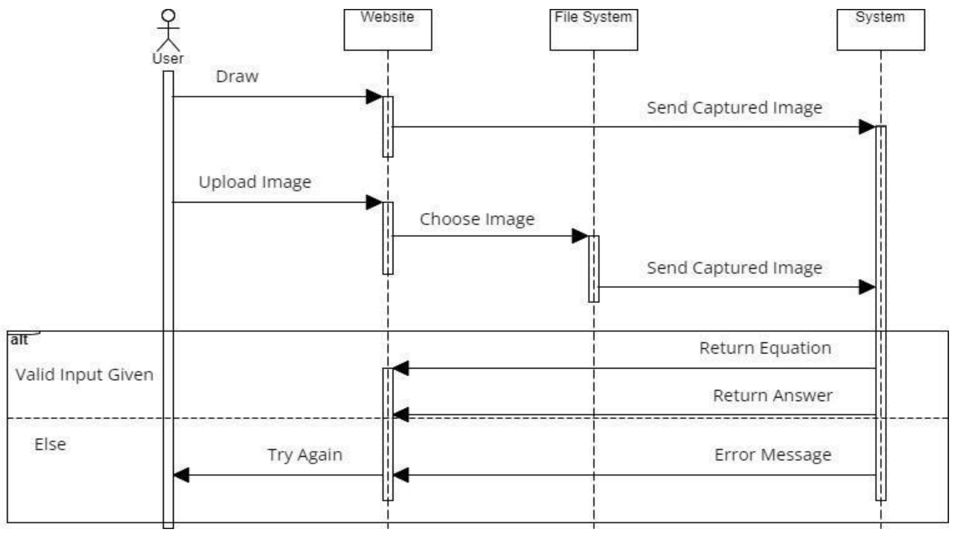


Figure 4.3.3.1: Sequence Diagram

Various components in the sequence diagram are:

- User
- Website
- File system
- System

4.3.2 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

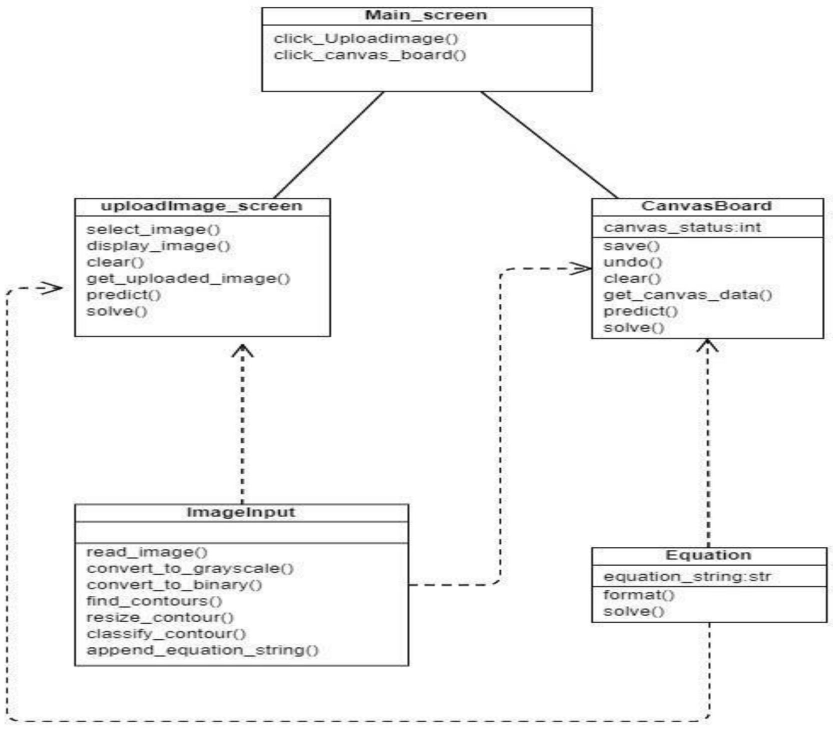


Figure 4.3.4.1: Class Diagram

In our application, the classes are as follows,

- Main_screen
- Uplad_screen
- canvasBoard
- ImageInput
- Equation

5. SYSTEM IMPLEMENTATION

Modules

- Handwritten Mathematical Equation Recognition
- Expression Evaluation
- Web Application

5.1 Handwritten Mathematical Equation Recognition

It consists of the following steps

5.1.1 Data Collection

The dataset was used from Kaggle and it consisted images of handwritten digits. Every specific element has more than 100 images for the model to train on and predict. Dataset is fairly diverse since it contains handwritten samples of different individuals which helps the CNN model to train on various parameters. Therefore, only a subset of these mathematical symbols is considered in this paper which are digits (0-9), arithmetic operators (+, -, *, ÷), characters(y). Took 45055 training images belonging to 16 classes. Save all the data folder in the same path as your programs are.

5.1.2 Pre-Processing

Binarization: The conversion of the input image into binarized image is necessary. **Binarization** is the method of converting any grayscale image, RGB Image into a black-white image. In a simple example, transforming an image's gray-scale from the 0-255 spectrum to a 0-1 spectrum is binarization. This is Achieved using OpenCv method cv2.adaptiveThreshold()

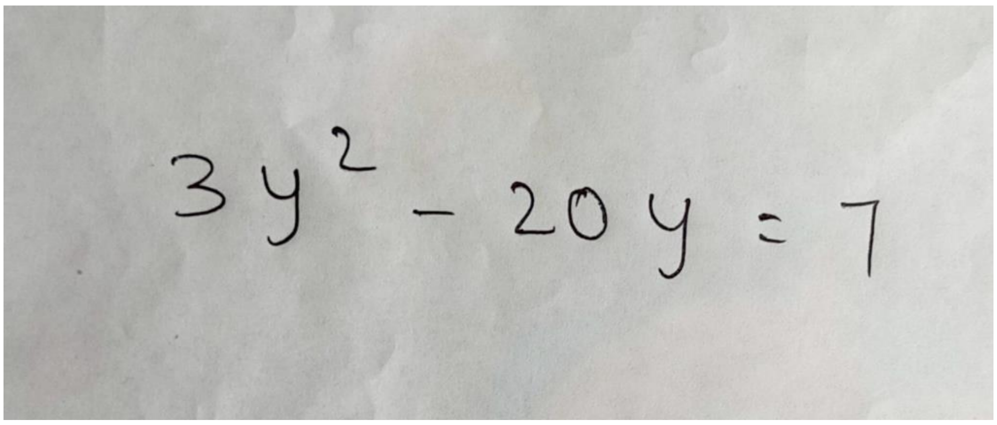


Figure 5.1.2.1: Handwritten Image

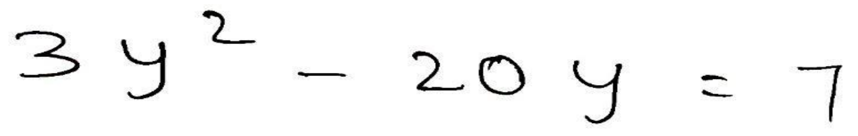


Figure 5.1.2.2: Threshold Image

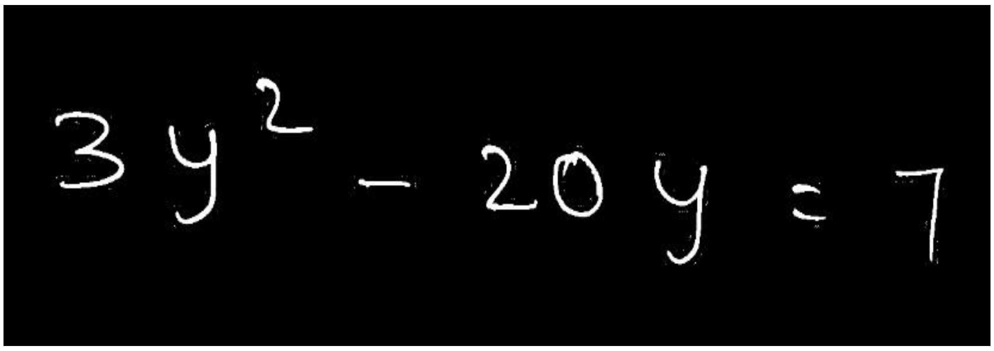


Figure 5.1.2.3: Binarized Image

The same goes for the canvas image too but it is not converted into threshold and is directly binarized.

5.1.3 Segmenting Pre-Processed Image

5.1.3.1 Contour Based Segmentation

Contours are defined as the line connecting all the points along an image's borders that have the same intensity. Contours are useful for shape analysis, determining the size of an object of interest, and object detection. The findContours() function in OpenCV aids in the extraction of contours from images. It is most effective with binary images. These contours segregate the handwritten characters, symbols and numbers based on their intensities in binarized image which are than further used for recognition.

5.1.3.2 Bounding Box

In digital image processing, the bounding box is merely the coordinates of the rectangular border that fully encloses a digital image. This is done using OpenCV method cv2.rectangle().

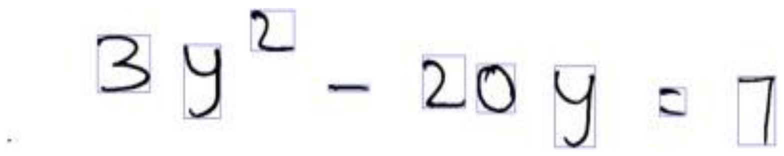


Figure 5.1.3.2.1: Bounding Box Around Each Contour

5.1.3.3 Resizing Each Contour:

The training images are of 45 by 45 pixels and each contour may vary in size so in order to make the model recognise the features of each contour we resize it before we feed them to the model to predict.

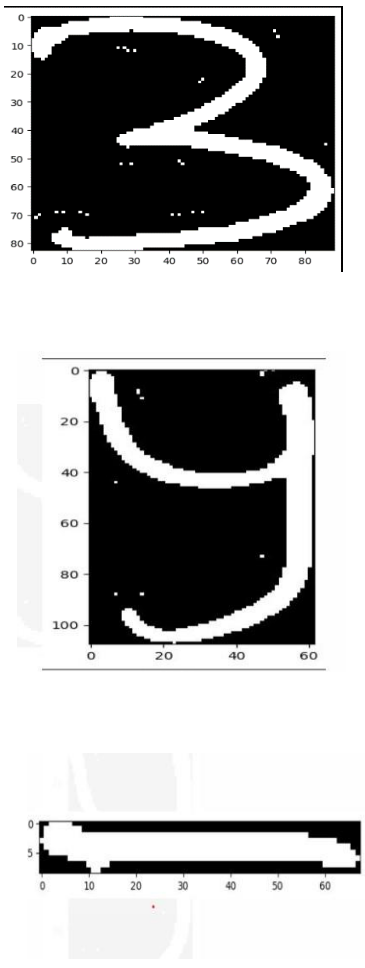


Figure 5.1.3.3.1: Before Resize

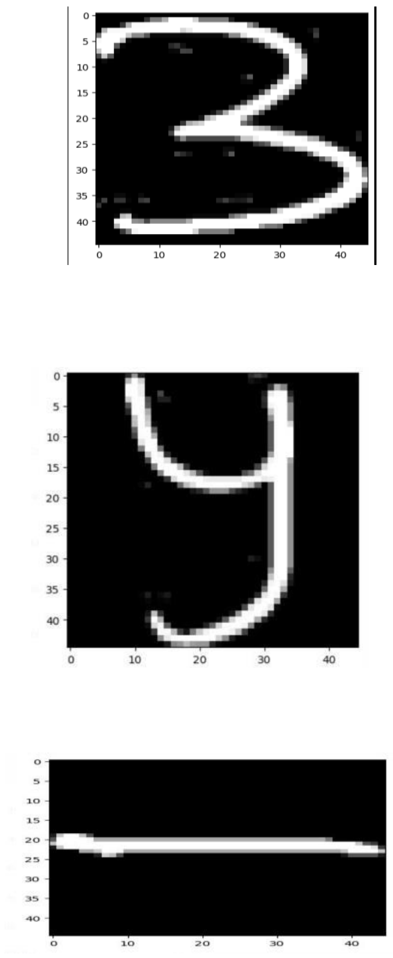


Figure 5.1.3.3.2: After Resize

5.1.4 Building Model

The model is built using Keras. Necessary modules and layers from keras are imported which would form the model. The model created is a Sequential model.

Input and First Hidden layer:

First, a set of two Convolutional layers (32 filters and ReLU activation function) to identify the low level image patterns (lines, edges, etc.).Next, the Max Pooling layer simply downsamples the filters to reduce computational load, memroy usage and number of parameters.

Second Hidden Layer:

A set of two Convolutional layers (32filters and ReLU activation function) to identify complex patterns that are a combination of lower-level patterns detected by the first layer.Max Pooling layer for downsampling.

Fully Connected Layer and Output:

First, a Flatten layer is use to convert the final feature maps into a single 1D vector (necessary for Dense layer input). Once the layers are added, the training parameters (loss function, score function and optimzation function) are defined.The loss function measures the error rate between the observed and predicted labels. For this model, categorical_crossentropy is used as the loss function. Next, the Adam is used as the optimizer algorithm to iteratively improves various model parameters.

This optimization algorithm is a further extension of stochastic gradient descent to update network weights during training. Unlike maintaining a single learning rate through training in SGD, Adam optimizer updates the learning rate for each network weight individually.The metric function “accuracy” is used as the score function to evaluate the performance our model.Finally the model is trained on the dataset with 3 epoches and a batch size of 32.

The architecture used for our CNN model is described as follows:

```
      Model: "sequential_1"

_______________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 45, 45, 1)         0

 conv2d_3 (Conv2D)           (None, 43, 43, 32)        320

 max_pooling2d_3 (MaxPooling  (None, 21, 21, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 19, 19, 32)        9248

 max_pooling2d_4 (MaxPooling  (None, 9, 9, 32)         0
 2D)

 conv2d_5 (Conv2D)           (None, 7, 7, 32)          9248

 max_pooling2d_5 (MaxPooling  (None, 3, 3, 32)         0
 2D)

 flatten_1 (Flatten)         (None, 288)               0

 dense_2 (Dense)             (None, 128)               36992

...
Total params: 57,485
Trainable params: 57,485
Non-trainable params: 0
```

Figure 5.1.4.1: Architecture of CNN

Saving The Model

It is important to save the model so that there is no need to run the epochs all the time. Instead we can use the saved weights of the model every time we run the web page by loading the model.

5.2 Expression Evaluation

After the recognition of the characters and symbols,they are appended to a string the expression is passed to the function to solve and display the solution. For all the mathematical processing python library called SymPy is used [21]. SymPy depends on mpath which is a python library for real and complex floating-point arithmetic with arbitrary precision. The final answer after calculation is displayed on the GUI based page.

5.3 Web Application

The final Equation and answer after calculation is displayed on the GUI based page.

5.4 Sample Code

Equation Recognition

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.preprocessing import image

def binarize(img):

    img = image.img_to_array(img, dtype='uint8')

    binarized=
np.expand_dims(cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY,11,2), -1)

    inverted_binary_img = ~binarized

    return inverted_binary_img

data_dir = 'data'

batch_size = 32

img_height = 45

img_width = 45

train_datagen = ImageDataGenerator(preprocessing_function=binarize)

 train_generator = train_datagen.flow_from_directory( data_dir,
 target_size=(img_height,   img_width),   batch_size=batch_size,    color_mode="grayscale",
class_mode="categorical", seed=123)

class_names = [k for k,v in train_generator.class_indices.items()]

class_names = ['+', '-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '=', 'div', 'times', 'y']

def detect_contours(img_path):

    input_image = cv2.imread(img_path, 0)

    input_image_cpy = input_image.copy()

binarized=cv2.adaptiveThreshold(input_image_cpy,255,cv2.ADAPTIVE_THRESH_MEAN
_C, cv2.THRESH_BINARY,11,2)

    inverted_binary_img = ~binarized

    contours_list, hierarchy = cv2.findContours(inverted_binary_img,

                                              cv2.RETR_TREE,

                                              cv2.CHAIN_APPROX_SIMPLE)

     l = []

    for c in contours_list:

        x, y, w, h = cv2.boundingRect(c)

        l.append([x, y, w, h])

    lcopy = l.copy()

    keep = []

     eqn_list = []

     for (x, y, w, h) in sorted(keep, key = lambda x: x[0]):

        img = resize_pad(inverted_binary_img[y:y+h, x:x+w], (45, 45), 0)

        first = tf.expand_dims(img, 0)

        second = tf.expand_dims(first, -1)

        predicted = new_model.predict(second)
        # print('4')
        max_arg = np.argmax(predicted)
        pred_class = class_names[max_arg]
        if pred_class == "times":
            pred_class = "*"
        if pred_class == "div":
            pred_class = chr(47)
        eqn_list.append(pred_class)
    eqn = "".join(eqn_list)
```

CNN Model

```
num_classes = 16
model = tf.keras.Sequential([
  tf.keras.layers.Input((45, 45, 1)),
      tf.keras.layers.experimental.preprocessing.Rescaling(1./255),    #    originally
tf.keras.layers.Rescaling
  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(num_classes)
])
model.compile(
  optimizer='adam',
  loss=tf.losses.CategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])
model.fit(
train_generator,
epochs=3
)
model.save('eqn-detect1 -model')
new_model = tf.keras.models.load_model('eqn-detect1 -model')
```

Equation Evaluation

```
import sympy as sym
from sympy import *
def solve_equation(equation):
   try:
        elif 'y' in equation and '=' in equation:
            y = sym.symbols('y')
            left, right = equation.split("=")
            eq = left+'-'+right
            result = sym.solve(eq, (y))
            return result
        elif 'y' in equation and '=' not in equation:
            y=sym.symbols('y')
            result = sym.solveset(equation,y)
            return result
        else:
            left, right = equation.split("=")
            result = sym.solve(left,right)
        if len(result) == 0:
            return "No solution found."
        else:
            return result
    except (ValueError, TypeError, AttributeError, RuntimeError, SyntaxError) as e:
        return str('Wrong equation prediction')
```

Web Application:

```
from flask import Flask, render_template, request, jsonify, flash,redirect, url_for
import base64
import io
from PIL import Image
from equation_calculator import equation_solver_function
import os
import cv2
from solve_equation_file import solve_equation
app = Flask(_name_)
app.config['SECRET_KEY']='abcdef'
app.config['UPLOAD_FOLDER'] = 'static'
@app.route('/', methods=['GET','POST'])
def index():
    return render_template('homepage.html')
@app.route('/upload_image', methods=['POST','GET'])
def canvas_image():
    return render_template('canvasimage.html')

@app.route('/predict_upload_image', methods=['POST','GET'])

def predict_upload_image():

    if 'equation.png'  in os.listdir('static'):

        equation = equation_solver_function('static\\equation.png')

        equation = equation.replace('**','^')

        return jsonify(equation)

    else:

        return jsonify("Please write or upload some image first"

@app.route('/solve_equation_func', methods=['POST','GET'])

def solve_equation_func():

    input_text = request.form['inputequation']

    input_text = input_text.replace('^','**')

    result= solve_equation(input_text)

    return jsonify(str(result))

@app.route('/save', methods=['POST'])

if __name__== '_main_':

    app.run(debug=True)
```

6. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product.

There are various types of tests. Each test type addresses a specific testing requirement.

6.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation We test each and every individual component thoroughly whether they are functioning properly according to their desired operations

6.1.1 Test Strategy and Approach

Field testing will be performed manually and functional tests will be written in detail.

Test Objectives:

- All the features must work properly. Equation is recognised properly
- Equation has to be formatted placing a '^' for exponent and '*' for multiplication.
- Equation should be solved properly. Solution must be divided

6.1.2 Features to Be Tested:

- Verify that canvas is working perfectly.
- Verify Whether uploading equation is working properly.
- Whether the Handwritten Equation Recognition system is working properly.
- All links should take the user to the correct page.

Test Case Id	**Test scenario**	**Input**	**Expected Output**	**Actual output**	**Result**
UT001	Basic recognition	Image of a simple addition equation	Equation Recognised And Solved	Equation Recognized And Solved	pass
UT002	Linear Equation recognition With variable	Image of a Linear equation With One Variable ‘y’	Equation Recognized And Solved	Equation Recognised And Solved	Pass
UT003	polynomial Equation recognition With variable.	Image of a polynomial equation With One Variable ‘y’	Equation Recognised And Solved	Equation Recognised And Solved	Pass
UT004	Exponent Recognition	Exponent value less than 10	Equation Recognised And Solved	Equation Recognised And Solved	Pass
UT005	Exponent Recognition	Exponent value less than 10	Equation Recognised And Solved	Equation not Recognised And not Solved	Fail
UT006	Distorted Image Recognition	Image of Equation slightly blurred	Equation Recognised And Solved	Equation Recognised And Solved	Pass
UT007	Error handling	Input an equation that is not valid (e.g., "2 + =")	Equation Recognised And display Error Message “Wrong Equation”	“Wrong Equation “	Pass

UT008	Empty Image Recognition	Empty Image	Display “Empty Image”	Display “Empty Image”	Pass
UT009	Distorted Image Recognition	Image with lot of Noise	Equation Recognised And Solved	Equation not Recognised Properly	Fail
UT010	Multiple Equations Recognition	Batch of multiple equations	Equation Recognised And solved	Equation not Recognised Properly	Fail

Table 6.1.2.1: Unit Testing Test Cases

6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

We put the Handwritten Equation Recognition and solving system through integration test is to check that components or software applications, e.g., components in a software system or – one step up– software applications at the company level – interact without error.

The whole Handwritten Equation Recognition and solving system is integrally checked at a time and check whether the individual components are working properly and coordinating with each other such that the whole system if flawless.

Test Case Id	**Test scenario**	**Input**	**Expected Output**	**Actual output**	**Result**
IT001	Data Flow	Upload without choosing File	Error message: “No image Selected"	“No image Selected”	pass
IT002	Data Flow	Empty Canvas	Error message: “Draw Equation on canvas."	“Draw Equation on canvas."	Pass
IT003	Image Loading	Upload selected Image from directory	Display Uploaded Image	Display Uploaded image	Pass
IT004	Image Prediction	Upload Image stored inn file explorer	Display Recognized Equation	Equation	Pass
IT005	Image Prediction	Image not uploaded fromfile explorer	Error Message “upload an Image first”	“upload an Image first”	Pass
IT005	Solving	Predicted Equation	Display Solution	Solution	Pass

IT006	Solving	Without Predicting Equation	Error Message "Predict something"	"Predict something"	Pass
IT007	Model Integration with Interface	model	Smooth integration and coordination among the modules for seamless functioning	Smooth integration and coordination among the modules for seamless functioning	Pass

Table 6.2.1: Integrate Testing Test Cases

6.3 Acceptance Testing

Test Objective

To verify that the CNN-based handwritten mathematical equation recognition and solving system meets the requirements specified in the project scope.

Test Environment

The system will be tested in a controlled environment with the following hardware and software specifications:

- Operating System: Windows 10 or Higher
- Processor: Intel Core i5 or higher
- Memory: 8GB RAM or higher
- Web browser: Chrome
- Image capturing device: Smartphone camera or scanner

Test Case Id	Test scenario	Input	Expected Output	Actual output	Result
AT001	Equation recognition test	Capture a clear image of a handwritten equation and submit it to the system	The system correctly recognizes the equation and displays it on the screen.	The system correctly recognizes the equation and displays it on the screen.	Pass
AT002	Equation solving test	Submit a recognized equation to the system for solving	The system correctly solves the equation and returns the solution to the user.	The system correctly solves the equation and returns the solution to the user.	Pass
AT003	User interface test	Test the user interface with simulated user inputs	The user interface can handle and display the recognized equation and its solution correctly.	The user interface can handle and display the recognized equation and its solution correctly.	Pass
AT004	Accuracy test	Submit a set of handwritten equations with their expected solutions to the system	The system correctly recognizes and solves the equations with a high level of accuracy.	The system correctly recognizes and solves the equations with a high level of accuracy.	Pass

Table 6.3.1: Acceptance Testing Test Cases

Test results

All test cases passed, indicating that the system meets the acceptance criteria and requirements specified in the project scope. The system is ready for deployment and use.

7. RESULTS AND OUTPUTS

7.1 Results

Implementation of Handwritten mathematical Equation Recogniser and solver is done with Python, OpenCV and Tensorflow was done which includes the following steps:

1. **Pre-processing:** The handwritten equation image is pre-processed to remove any noise, distortion, or background interference. The image is then normalized and resized to a fixed size for input into the CNN using OpenCV

2. **Feature extraction and Classification:** The CNN is designed to extract features from the image that are relevant to the recognition and solving of mathematical equations. The layers of the CNN perform convolution, pooling, and activation functions to extract high-level features. The output of the CNN is a sequence of identified components that represent the handwritten equation.

3. **Solving:** Once the components have been identified, they can be used to solve the equation using Sympy.

4. **Interface:** To easily recognise the equation an interface is created to upload images of equation or to draw on canvas using Flask.

7.2 Output

Home Page

Home page has the following buttons

- Uploading image
- Image input through canvas

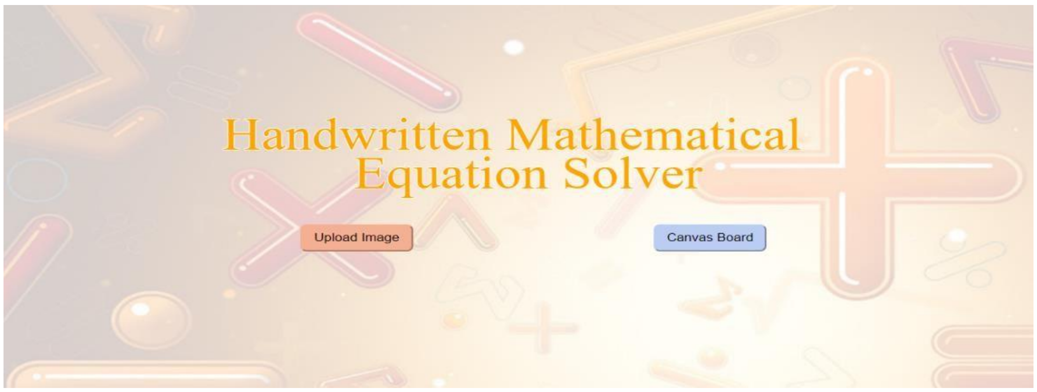


Figure 7.2.1: Home Page

Uploading Image Page

- The user can take a picture of handwritten mathematical equation and save the image locally.
- The choose file button is clicked then the user should get the interface of local system option.
- Select the image from local system should and click on the upload button.

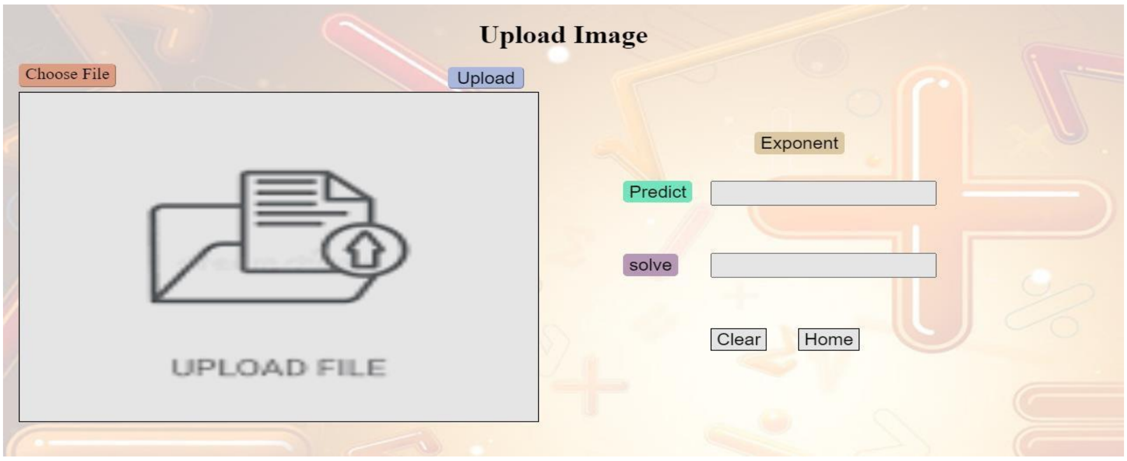
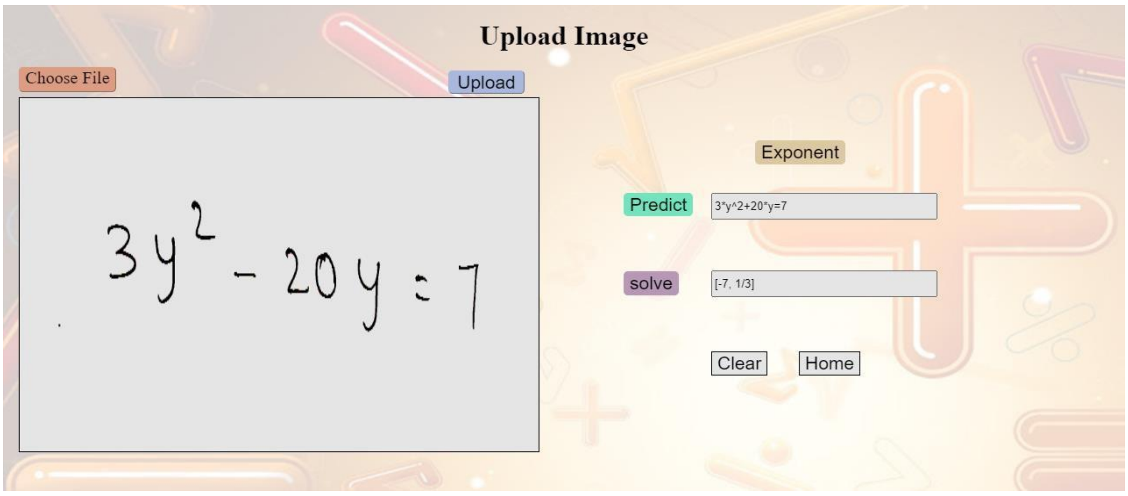


Figure 7.2.2: Before Upload



$3y^2 - 20y = 7$

Predict: 3*y^2+20*y=7

solve: [-7, 1/3]

Figure 7.2.3: Upload Image

Canvas Page:

- The user can write the equation on the canvas board and then save the equation.
- Click on the predict button which shows the system recognized equation in the system form.
- Click on the solve button it displays the solution of the equation.

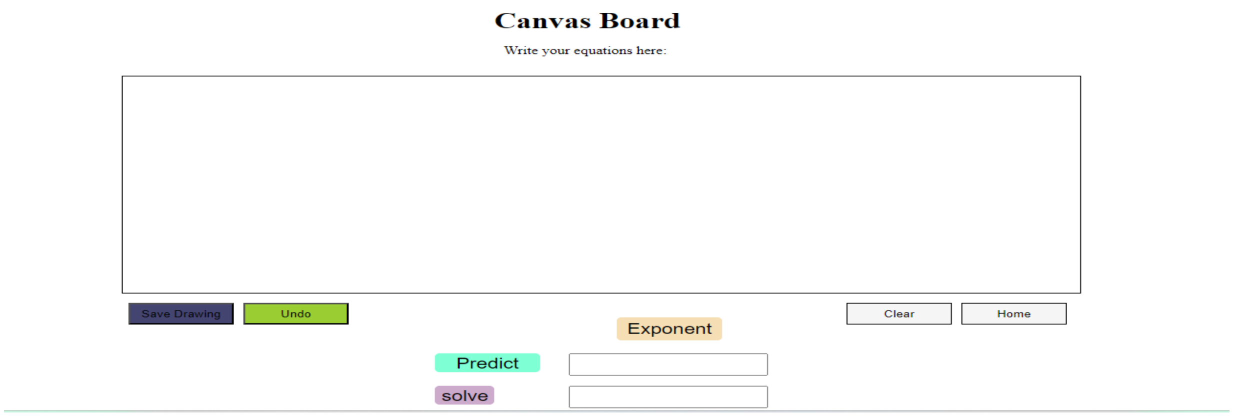
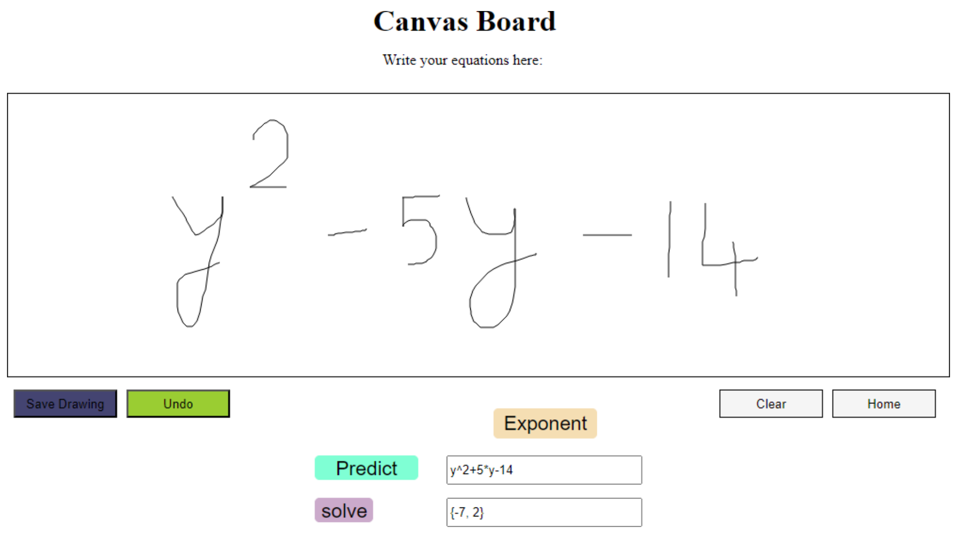


Figure 7.2.4: Before Drawing



y^2+5*y-14

{-7, 2}

Figure 7.2.5: After Drawing

8. CONCLUSION

Through this project we aim to develop a user-friendly website that captures the image of mathematical equation, recognize the equation and present the user with required solution. We aim to make the whole experience of experimenting with equations very user friendly and to remove the hassle of learning a mathematical tool just for mathematical experimentation. Since mathematics itself is a very wide field, digitizing and evaluating all of the mathematical symbols becomes a very complex and tedious task. For Recognition Dense layers are used to extract features from input images with Relu as an activation function for the hidden layers and softmax for the final layer for predicting numbers and operators is a multi-class classification problem. This model is trained with an adam optimizer which is reported to have a faster convergence rate than normal stochastic gradient descent (SGD) with momentum. Through this we got good training and testing accuracy. The motivation was to create a user friendly application which will be simple and easy to use, without feeling the need to study the usage of the tool or complex syntaxes. It reduces the task on user side as direct images can be given to the application and then get the equivalent solution for it. The discussed web application can solve all the basic arithmetic calculations, polynomial Equations.

8.1 Limitations

1. **Ambiguity:** The problem of ambiguity between '*' and 'x' symbols hence 'x' is not taken into training.so provide only equation with variable 'y'.
2. **Multiple Equation Problem**: The system Cannot Recognize multiple Equations in one Image.so, we need to make sure that the image contains only on equation.
3. **Exponent problem** The system Cannot Recognize exponent powers more than 9.so, we need to make sure that the image contains exponents less than 9.
4. **Distorted Image** The system cannot properly recognize noisy or distorted Images. we need to make sure that the image is clear.

9. FUTURE SCOPE

There are many possibilities for this project in future. Recently this project is developed to recognize limited number of symbols such as +, -,*and /. There are many other symbols used in mathematics such as brackets (), {}, []. So, the system can be trained to recognize such symbols and work on them. Also it can be trained to recognize alphabets so that the system can solve the complex equations. The system can be developed to solve the equation and also plot the graph as the solution. As Math equation solver gives the final solution to the user it can be further developed to give the solution step wise so that the user can get the knowledge of who the solution is generated. can Extend the system Recognize multiple Equations in one Image. With more rigorous training and extension of dataset the capability and accuracy of the model in general can be increased. It can further be extended to solve more complex calculations like system of linear equations, trignomatric calculations differential equations and integrations. In future days the main focus will be to try to raise the precision level and build a segmentation system that can successfully segment two connected digits, and also increase the performance level of the dataset.

10. BIBILOGRAPHY

[1] Recognition of Handwritten Mathematical Expression and Using Machine Learning Approach Prathamesh Tope1, Sarvesh Ransubhe2, Mohammad Abdul Mughni3, Chinmay Shiralkar4, Mrs. Bhakti Ratnaparkhi.

[2] Handwritten mathematical equation solver Rajwardhan Shinde, Onkar Dherange, Rahul Gavhane, Hemant Koul, Nilam Pati

[3] Salma Shofia Rosyda and Tito Waluyo Purboyo ,"A Review of Various Handwriting Recognition Methods", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 2 (2018) pp. 1155- 1164.

[4] Lyzandra D'souza and Maruska Mascarenhas, "Offline Handwritten Mathematical Expression Recognition using Convolutional Neural Network", 2018 International Conference on Information, Communication, Engineering and Technology (ICICET) Zeal College of Engineering and Research, Narhe, Pune, India.

[5] C. Lu and K. Mohan. "Recognition of Online Handwritten Mathematical Expressions" cs231n Project Report Stanford (2015).

[6] Dae Hwan Kim, Jin H. Kim , "Top-down search with bottom-up evidence for recognizing handwritten mathematical expression" ,2010 12th International Conference on Frontiers in Handwriting Recognition

[7] Dmytro Zhelezniakov, Viktor Zaytsev and Olga Radyvonenko. "Online Handwritten Mathematical Expression Recognition and Applications: A Survey" IEEE Accesd, accepted February 17, 2021, date of publication March 2, 2022

[8] https://www.kaggle.com/datasets/xainano/handwrittenmathsymbols?resource=download

[9]https://vipul-gupta73921.medium.com/handwritten-equation-solver-using-convolutional-neural-network-a44acc0bd9f8

[10] https://github.com/sabari205/Equation-Solver

[11] https://towardsdatascience.com/building-a-handwritten-multi-digit-calculator-f03cf5028052