Q-Sol

As the name suggests: “an Equation Solver”

INTERIM REPORT

Project Group no: 5

Guide:

Prof. Divya Prasad

Group Members:

Omkar R (31)
Adhithya T (1)
Akshay TA (5)
Karun Krishnan (21)

CONTENTS

ABSTRACT

Robust handwritten character recognition is a tricky job in the area of image processing. Among all the problems, handwritten mathematical expression recognition is one of the complicated issues in the area of computer vision research. In this paper a group of handwritten arithmetic equations are considered to recognize and make a solution for those equations. For classification of specific characters we apply Convolutional Neural Network. Each of the correct detection, character string operations is used for the solution of the equation. Finally the experimental results show the great effectiveness of our proposed system.

The main objective of this project is to develop a Website which calculates a Handwritten Equation Solver trained by handwritten digits and mathematical symbols using Convolutional Neural Network with some image processing techniques to achieve a decent accuracy of 89.9%.

INTRODUCTION

Convolutional Neural Network(CNN) is one of the mostly used classification model in computer vision area. Above all existing models, CNN is one of the most popular models and has been providing state-of-the-art recognition accuracy on object recognition, segmentation, human activity analysis, image super resolution ,object detection , scene understanding, tracking, and image captioning. Our job is to recognize the handwritten quadratics from the image and for each successful detection finding the solution of that equation. Neural networks are particularly useful for solving problems that cannot be expressed as a series of steps, such as recognizing patterns, classifying them into groups, series prediction and data mining.

OBJECTIVE

To develop a website ‘EQ-SOL’ wherein the user can give a mathematical equation as input (image) and will get the answer as output.

REQUIREMENTS

OS	Windows 10
RAM	8 GB
LANGUAGE	Python
HARDWARE	Webcam
LIBRARIES USED	Keras,openCv,flask

IMPLEMENTATION

A. Dataset preparation

We prepared some dataset by ourselves and we also used some modified version of the NIST dataset which is similar to the popular EMINIST dataset for digits. Image size we used is 45x45.

B. Pre-Processing

1. Invert the image and then convert it to a binary image because contour extraction gives the best result when the object is white and surrounding is black.
2. To find contours use the ‘findContour’ function. For features, we obtain the bounding rectangle of contour using ‘boundingRect’ function (Bounding rectangle is the smallest horizontal rectangle enclosing the entire contour)
3. Since each image in our dataset contains only one symbol/digit, we only need the bounding rectangle of maximum size. For this

purpose, we calculate the area of the bounding rectangle of each contour and select the rectangle with maximum area.

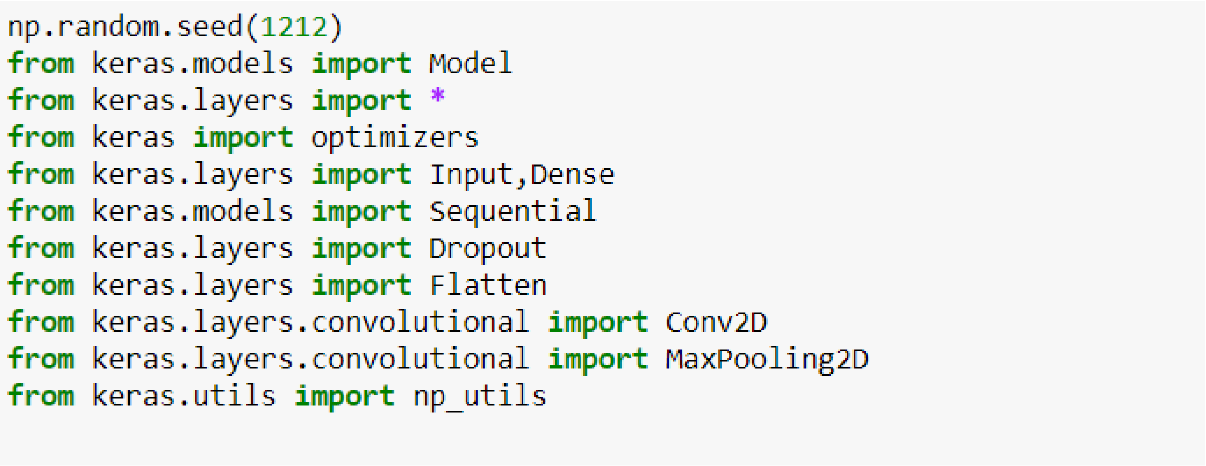
4. Resize the maximum area bounding rectangle to 28 by 28. Reshape it to 784 by 1. So there will be now 784-pixel values or features. Now, give the corresponding label to it. So our dataset contains 784 features columns and one label column. After extracting features, save the data to a CSV file.

C.Training data using CNN

Since convolutional neural networks work on two-dimensional data and our dataset is in the form of 785 by 1. Therefore, we need to reshape it. Firstly, assign the labels column in our dataset to variable y_train. Then drop the labels column from the dataset and then reshape the dataset to 28 by 28. Now, the dataset is prepared and ready.

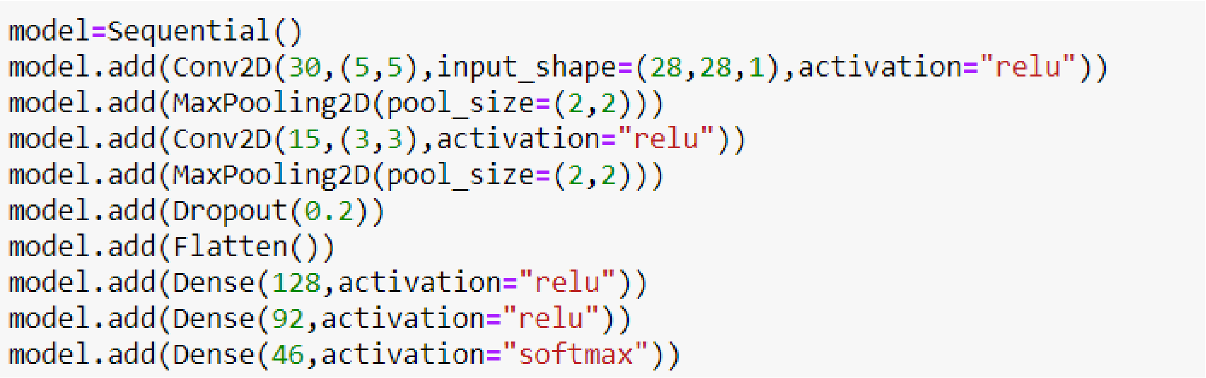
a. Building Convolutional Neural Network

For making CNN, import all the necessary libraries:



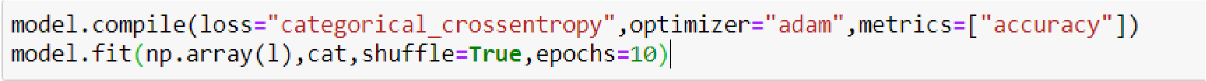
```
np.random.seed(1212)
from keras.models import Model
from keras.layers import *
from keras import optimizers
from keras.layers import Input,Dense
from keras.models import Sequential
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
```

b. Creating the model



```
model=Sequential()
model.add(Conv2D(30,(5,5),input_shape=(28,28,1),activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(15,(3,3),activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128,activation="relu"))
model.add(Dense(92,activation="relu"))
model.add(Dense(46,activation="softmax"))
```

c. Fitting Model



```
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"])
model.fit(np.array(l),cat,shuffle=True,epochs=10)
```

D. Classification model

In this research convolutional neural network is used as a classification model, a 30×30 gray scale image is used as input of the CNN input layer for the training section and also for the testing section. A 5×5 filter is used at the convolutional layer after the convolution of the input image with the filter for each input image a 28×28 feature vector is produced. It is concordant to apply a nonlinear layer (or activation layer) instantly after conv layer. Pooling layers perform down-sampling operations. Random selection of input elements to zero with a given probability is done in a dropout layer. It is a simple way to prevent overfitting in the neural network. The core idea is to randomly drop units (along with their connections) from the neural network during training. In our proposed method we use the probability 0.2 at the dropout layer during the training.

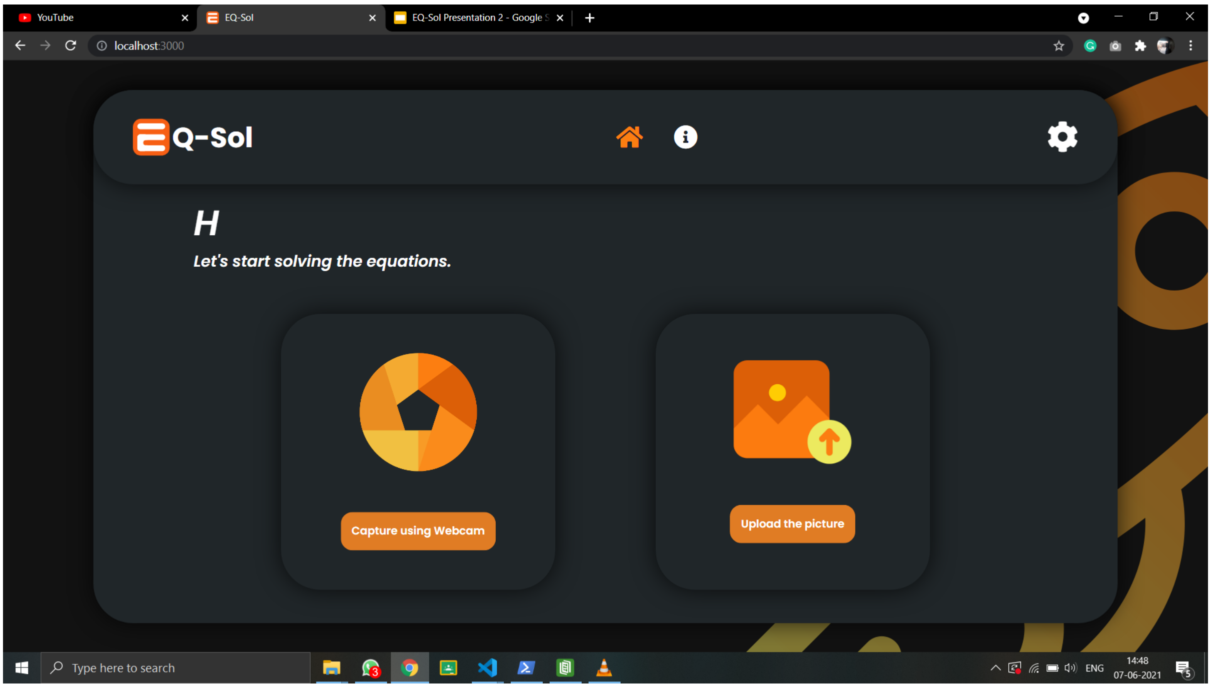
E. Front-End

On the Front-End side, we have used the **ReactJS** framework. Also used certain npm modules like react-router-dom, react-dropzone, reactstrap, react-image-crop, react-webcam. Basically frontend provides 2 options

1. **Camera**: Here we can click a picture using from the webcam and proceed for upload
2. **Upload a pic**: Here we upload a picture from our computer and proceeds for upload

Also we have added a crop image feature for selectively cropping out the equation part of the image and proceed with that cropped image. It enhances the User Experience.

Have added a decent User Interface of Dark theme for a great visual and usability of the application.



H

Let's start solving the equations.

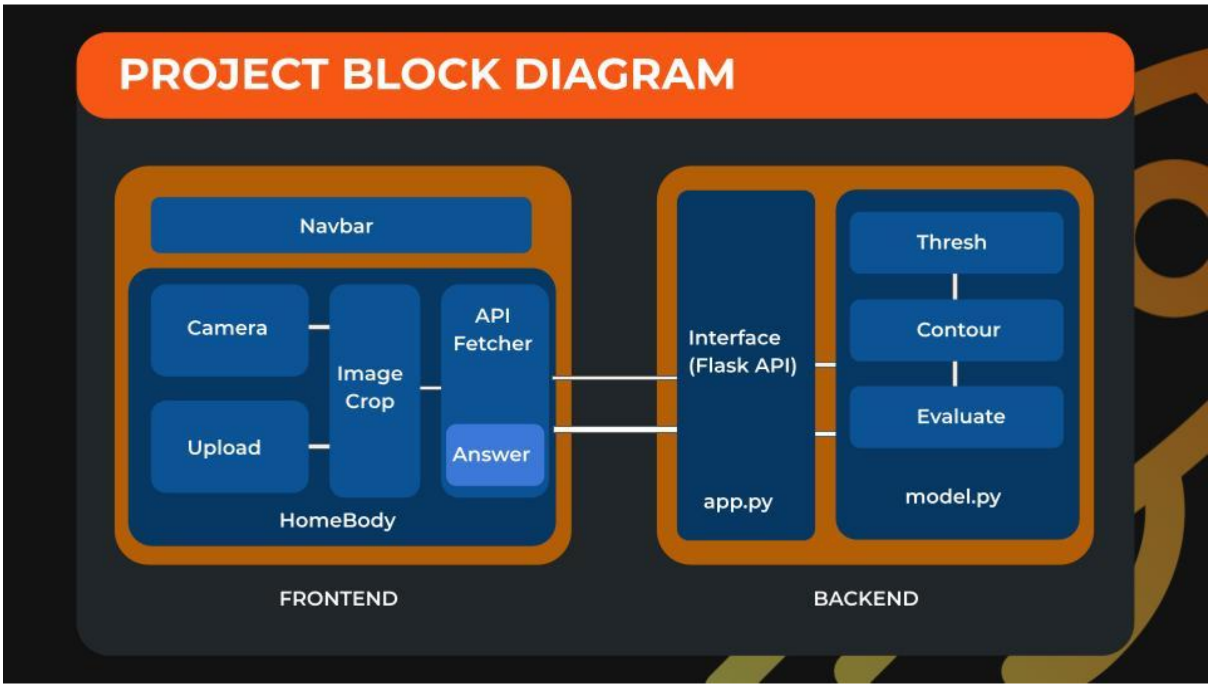
F. Interfacing

Interfacing denotes integrating Frontend with the created **ML Model** in Backend. We have used **Flask** for implementing it. Also used modules like flask-cors for making the **API RESTful** and supports any frontend framework. It generates an API which could be used by the frontend for data sending and receiving from the backend using **standard HTTP** protocol.

From the frontend part, we use the fetch function to fetch api by **POST** method with image as a **base64** encoded string to backend. In flask we declare the flask app with method POST. Here, we first of all access the parameter using request and convert that base64 image string to normal image and save it in localhost directory. Then the url of the created image is passed to the ML model function.

When it reaches the ML model, the backend code works and returns the result as a **JSON** object which contains the detected equation and its corresponding answer. Same result is passed to frontend by returning the result from the ML model function. In frontend the fetch function returns an object from the backend which is converted to normal form and displayed by accessing its attributes.

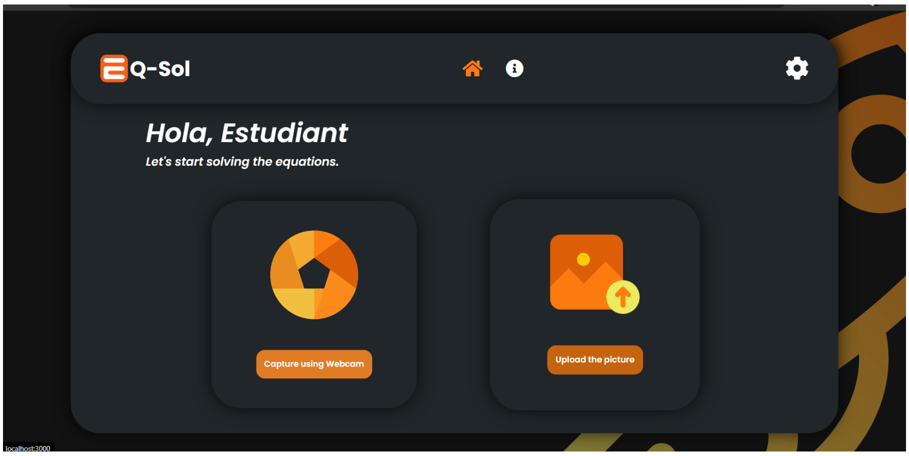
WORKING



Our website offers two options to scan the handwritten equation. It can either take a live photo using the webcam or a pic can be uploaded from the computer. It is then sent to the backend via Flask API. The characters/digits are then recognized and extracted. It undergoes several procedures like reshaping by 28x28 etc and is fed to the model. The model solves the equation and returns the answer back to the user via Flask API.

EXPERIMENTAL RESULTS

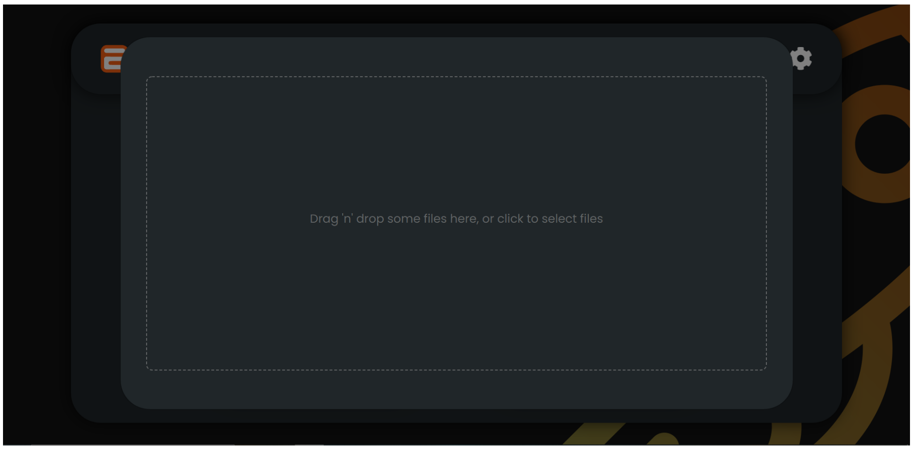
We have added screenshots which show the working of the application. It is able to detect the equation in the picture given both as upload a picture or capture a picture through the webcam and also able to show the solution of it.



Hola, Estudiant

Let's start solving the equations.

Home



Drag 'n' drop some files here, or click to select files

Image uploading

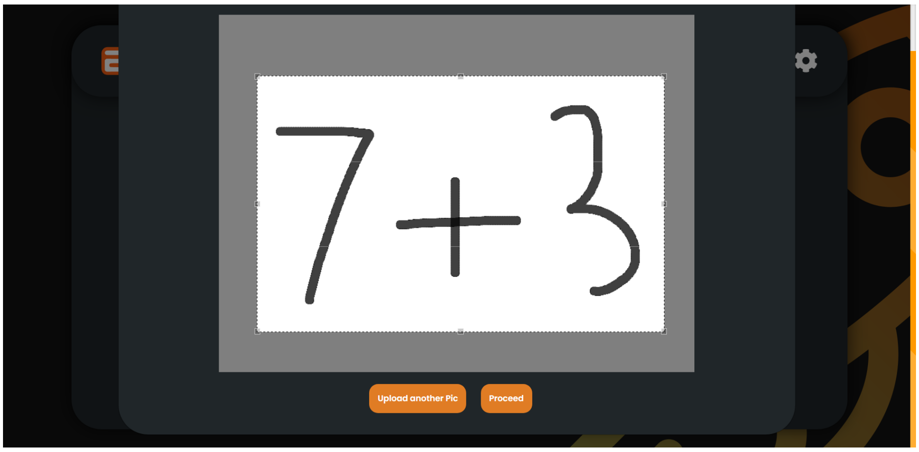
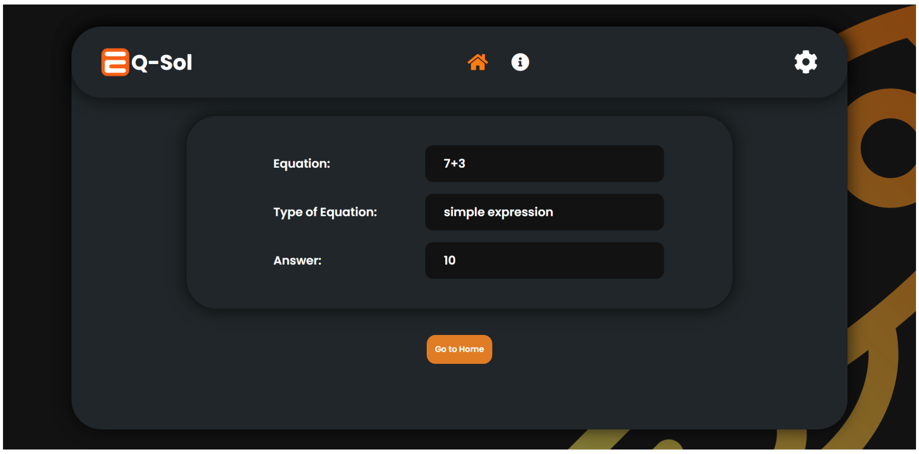


Image cropping

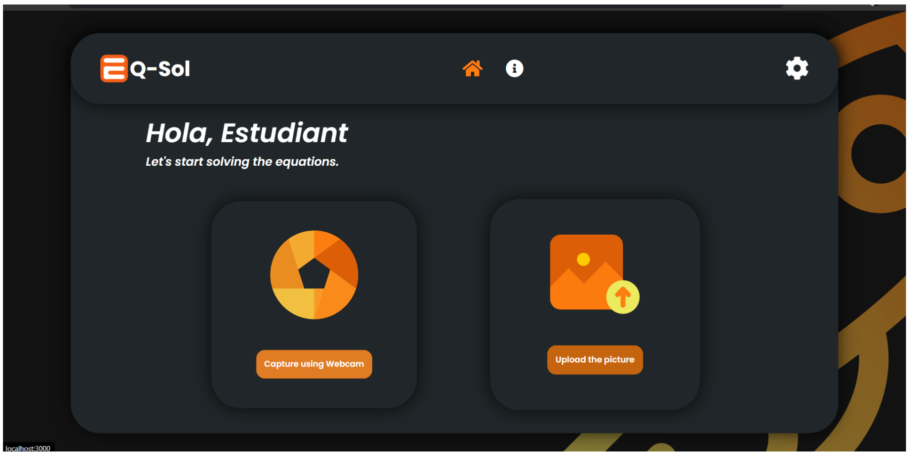


Equation: 7+3

Type of Equation: simple expression

Answer: 10

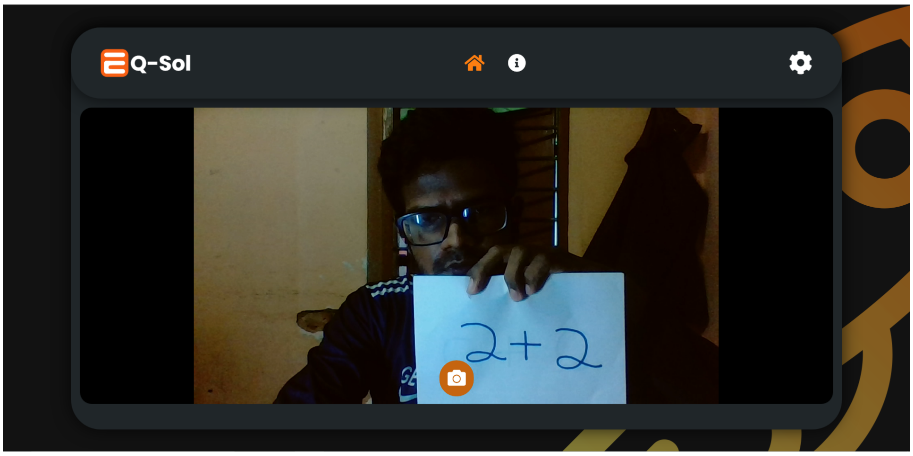
Output



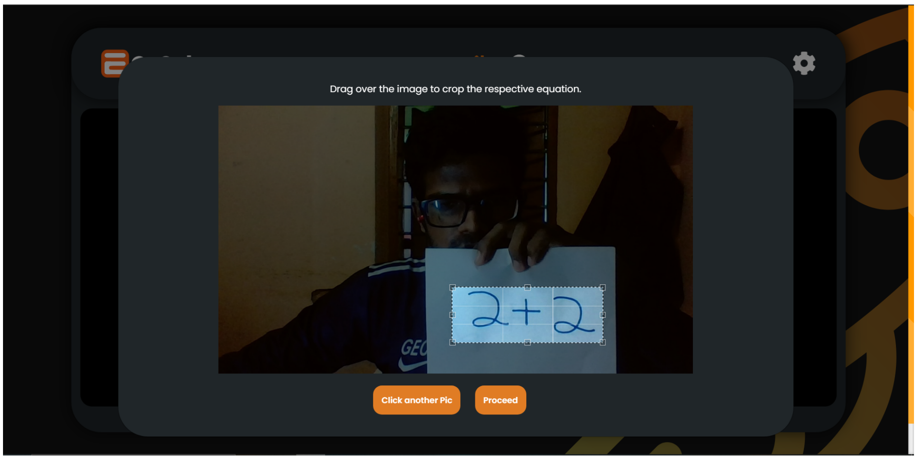
Hola, Estudiant

Let's start solving the equations.

Home

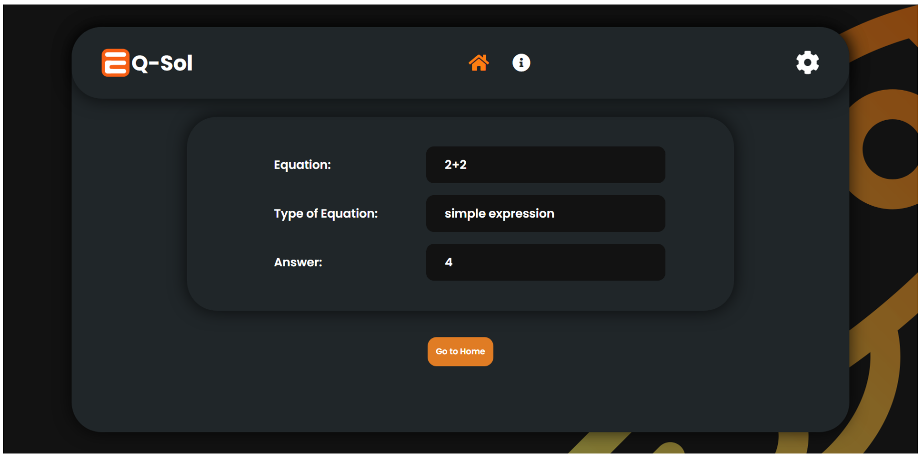


Capture image through webcam



Drag over the image to crop the respective equation.

Image cropping



Equation: 2+2

Type of Equation: simple expression

Answer: 4

Output

CONCLUSION

In this paper we are focusing on recognizing handwritten Mathematical expressions and equations. Feature extraction was the most complicated part of classification. Convolutional Neural Network the most powerful classification model is used in the classification part. Finally we obtained a state of art performance in the detection phases and also in the solution phases. To simplify the math, the main task is the feature extraction from the image and recognition with the help of the CNN model. Any person can easily use this as the process is easy and does not require any advanced knowledge of the technology.

In the future, we will try to improve the accuracy and try to make the system workable for multiple mathematical formulas simultaneously.

FUTURE WORKS

Features that include solving linear/quadratic equations, logarithms, differentiation etc.

REFERENCES

- https://www.researchgate.net/publication/326710549_Recognition_and_Solution_for_Handwritten_Equation_Using_Convolutional_Neural_Network
- https://www.nist.gov/itl/products-and-services/emnist-dataset
- https://aihubprojects.com/handwriting-recognition-using-cnn-ai-projects/
- https://www.geeksforgeeks.org/handwritten-equation-solver-in-python/
- https://github.com/sabari205/Equation-Solver

*****