TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A PROJECT REPORT ON

Handwritten Maths Expression Recogniser

BY:

GOPAL BAIDAWAAR CHETTRI (PUL075BCT036)

JIWAN PRASAD GURAGAIN (PUL075BCT041)

KUSHAL SHRESTHA (PUL075BCT045)

MANJEET PANDEY (PUL075BCT048)

Laboratory for ICT Research and Development

LALITPUR, NEPAL

March 14, 2023

ACKNOWLEDGEMENTS

This report Handwritten Mathematical Expression Recogniser has been prepared for the **LICT research project** assigned to us in our 8th semester. We would like to thank **Dr. Basanta Joshi**, who guided us on every aspect of the Artificial Intelligence course and helped visualize this project in reality.
We are also very grateful to **Mr. Sushant Regmi** and **Ms. Anusha Acharya** from NsDevil, for their guidance and helpful comments on our weekly meetings.

We would also like to extend our sincere gratitude to our friends who gave us valuable suggestions on every aspect of our project and helped to shape this project from users' perspective.

Also, we would like to express our appreciation to the **LICT-Pulchowk** for providing us this wonderful opportunity.

Sincerely,
Gopal Baidawaar Chhetri(075BCT036),
Jiwan Prasad Guragain(075BCT041),
Kushal Shrestha(075BCT045),
Manjeet Pandey(075BCT048)

ABSTRACT

Optical Character recognition is one of the most researched fields in deep learning and computer vision. Many researchers and tech giants are spending a lot of resources for the advancement of this field. A subset of this field is Maths Expression Recognition that has many applications in the field of academics. This report presents the detailed explanation of algorithms and methodology we adopted to complete our LICT project. titled ‘Handwritten Mathematical Equation Recognizer’. Our system integrates a draw pad ui in a rich text editor, that recognises handwritten equations with an accuracy around 95% on the training dataset. We use an end to end model Pix2Tex that we fine tuned on a handwritten dataset. An end to end model can infer its own rules regarding how to interpret the layout and form an operations tree intrinsically. So it is quite robust to writing and layout variations in handwritten expressions.

This report revolves around a review of various approaches to the task of recognizing handwritten expressions. We also present several challenges we faced while working on this project as undergraduate students.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 Problem Statement

Mathematical expressions constitute an essential part in most scientific and engineering disciplines. The input of mathematical expressions into computers is often more difficult than that of plain text, because mathematical expressions typically consist of special symbols and Greek letters in addition to English letters and digits. The layout of mathematical expressions also can complicate their input into computers because the relative positions of symbols in their two dimensional structure are used extensively to convey information; for example, subscripts and superscripts commonly denote some type of correspondence or function application, and stacking is used to denote division. The large number of characters and symbols, and the two dimensional nature of mathematical notation complicates entry and editing on a computer, since the ways of input for the computers only support a limited number of characters, and most computer interfaces are optimized to accommodate the linear text of natural language. The extra symbols may be entered through a more complex/specialized way for input some extra keys in the keyboard (e.g., function keys) along with a set of unique key sequences for representing other special symbols, or by defining a set of keywords to represent special characters and symbols, as in LATEX, and all mathematical expressions, even those with 2D layout, can be represented in a linear form. But such notations are difficult to understand and use, and require intensive training and practice. Alternatively, by taking advantage of pen-based computing technologies, one could simply write mathematical expressions on an electronic tablet for the computer to recognize them automatically. In situations where the expressions are already in some printed form, we could just scan in the document for the computer to recognize the expressions directly from the image. Such two dimensional representations allow easy understanding and modification of existing expressions.

1.2 Background

Mathematical equations are an important part of computing. Writing mathematical expressions and equations on the computer is quite hard and difficult. It is easier to write the equations by hand, on a board than to write on the computer. While writing the mathematical equations on the computer, we must have realized that it is a tedious job. All of this suggests that a proper tool for the identification of handwritten equations and expressions is of the essence. That's why, we have thought of developing a web application which is user friendly for the recognition of handwritten mathematical equations and expressions. And since

1.3 Objectives

- To recognize the numbers, variables, and operators/symbols in a handwritten mathematical expression
- To research and know more about the current state of the area of Handwritten Mathematical Expression Recognition.
- To create an easy to use user interface and integrate it with a rich text editor.
- To export the recognized expression into appropriate formats for use in electronic documents or programs.

1.4 Scope of the project

This project is being done by a group of four students assigned by LICT IOE Pulchowk. The project will include researching applicable algorithms and tools, and creating an app that fulfill the objectives of the project. Handwritten Equation Recognizer is a tool that can be used to basically convert the handwritten mathematical equations into the standard mathematical formats(like latex expressions). The user can write in the designed drawpad, the designated mathematical equations which will be converted into the digital form. The domain of this project is strictly mathematics. The deliverables for the project include an easy to use draw pad in a rich text editor that fulfills our objectives.

2 LITERATURE REVIEW

A preliminary literature review shows that plenty of studies has been done reviewing various approaches to each of the aspects of recognizing an equation from pen strokes, or vector image or document image input. A survey by Zanibbi, et. al.[1] provides a survey of various techniques on the subject of expression detection, symbol recognition, symbol layout analysis and mathematical content interpretation. Following its citations, the works of Anderson[2], and Labahn, et. al.[3] which describe the implementation of recognition in equations input through pen-based systems and document images were known to exist. Recently transformer networks have become popular, and ViT[8] are increasingly replacing CNNs in many tasks. An open source model pix2tex[9] implements an end to end model for math recognition purpose.

Since this project consists of creating a UI integrated into a rich text editor we researched several options for open source web-editors.We found that there were multiple options such as, using a react framework, where third party editor components are available, summernote, and ckeditor. The last two are open source projects.We used ckeditor[10] because it was the most popular open source alternative. We also searched the internet to try out existing applications that fulfill this project's objectives. In collecting these resources from internet we have found that various systems are available online and on mobile devices for performing desired mathematical calculations from handwriting input like a web app on the url https://webdemo.myscript.com/views/math/index.html , and the app Microsoft math solver from Microsoft. We will also be referencing tutorials from YouTube that cover the mentioned subjects.

3 THEORETICAL BACKGROUND

3.1 Mathematical Expression Recognition

The aim of mathematical expression recognition is to generate an accurate operator tree representation of expressions contained in a source which may range from pictures to hand drawn vector style files. Such operator tree representations then open the possibilities for multitude of applications like automatic graders, easy inclusions of mathematical expressions in neat digitized forms into academic papers, mathematical information retrieval and so on.

As figure 1 shows, the complete procedure for generating the operator tree representa- tion of the expressions contained in either rastor or vector style graphics can be divided into several stages. A brief overview of each of these stages follows next.

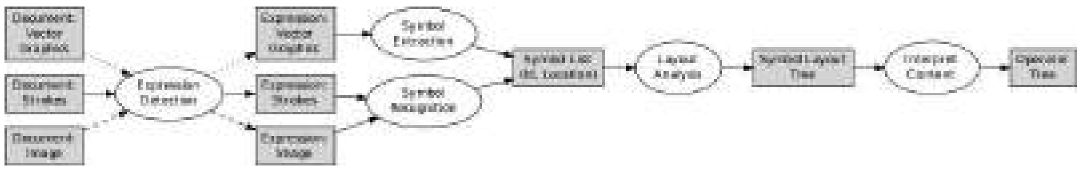


Figure 1: Stages in Maths Expression Recognition

3.1.1 Expression Detection

Expressions must be first identified and segmented. Methods for detecting offset ex- pressions are fairly robust, but the detection of expressions embedded in text lines remains a challenge. Another challenging aspect of expression detection is when there are multiple expressions that aren’t directly related by some operator in those ex- pressions (like in solutions to problems where latter expressions follow from previous ones). Such expressions may contain symbols which are ambiguougly located and it is challenging to deduce which expression they lie in.

3.1.2 Symbol Extraction or Symbol Recognition

In vector-based representations, such as PDF, symbol locations and labels can be recovered through the vector segments, though some handling of special cases is needed (e.g. root symbols are often typeset with the upper horizontal bar represented sep-

arately from the radical sign, $\surd$). In raster image data and pen strokes, detecting symbol location and identity is challenging. There are hundreds of alphanumeric and mathematical symbols used, many so similar in appearance that some use of context is necessary for disambiguation (*e.g. c, (, C*).

3.1.3 Layout Analysis

Analysis of the spatial relationships between symbols is challenging. Spatial structure is often represented using a tree, which can be termed a symbol layout tree. Symbol layout trees represent information similar to LATEX math expressions; they indicate which groups of horizontally adjacent symbols share a baseline (writing line), along with subscript, superscript, above, below, and containment relationships. Simple techniques like gridwise separation is often hindered by various writing styles for expressions which need extra logic or processing to be grouped with other symbols properly(see figure).

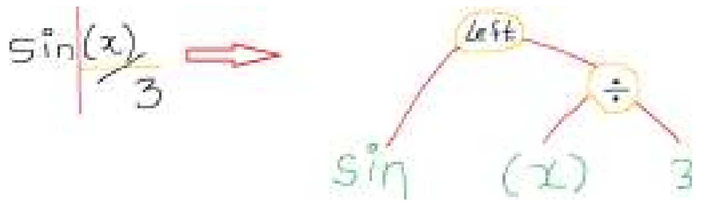


Figure 2: Incorrect Layout Breakdown by a simple gridwise separator

3.1.4 Mathematical Content Interpretation

Symbol layout is interpreted, mapping symbols and their layout in order to recover the variables, constants, operands and relations represented in an expression, and their mathematical syntax and semantics. This analysis produces a syntax tree for an ex- pression known as an operator tree.

3.2 StreamLit

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. It provides built in components that allow us to build and deploy powerful data apps in minimum amount of time.

3.3 Dart

Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks. Dart is designed for a technical envelope that is particularly suited to client development, prioritizing both development (sub-second stateful hot reload) and high-quality production expe- riences across a wide variety of compilation targets (web, mobile, and desktop). Dart also forms the foundation of Flutter. Dart provides the language and runtimes that power Flutter apps, but Dart also supports many core developer tasks like formatting, analyzing, and testing code.

3.4 Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase. The major components of Flutter include:

- Dart platform
- Flutter engine
- Foundation library
- Design-specific widgets
- Flutter Development Tools (DevTools)

3.5 DeepLearning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.Multi-layers neural networks attempt to simulate the behavior of the human brain allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks with- out human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

3.6 Neural Networks

A neural network works similarly to the human brain’s neural network. A “neuron” in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is a known as perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

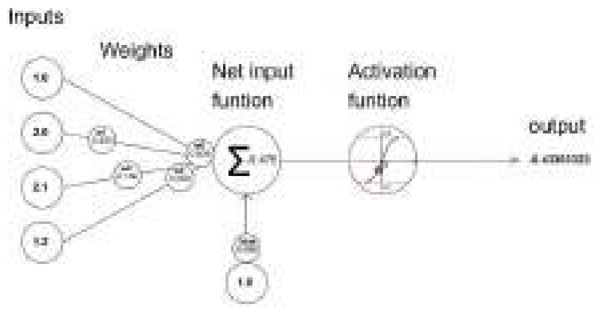


Figure 3: Neural Network

3.7 Vision Transformers

The Vision Transformer (ViT) is a deep learning model for image recognition that was introduced in a research paper [8] by Dosovitskiy et al. in 2020.

The Vision Transformer (ViT) is a recently proposed neural network architecture for image classification tasks. It is based on the transformer architecture used in natural language processing, and is the first to demonstrate strong performance in computer vision with only self-attention mechanisms.

The ViT consists of a series of transformer blocks, each containing a multi-head self-attention mechanism and a feedforward network. The input image is first divided into patches, which are linearly embedded to obtain a sequence of 1D vectors. A learnable positional encoding is added to each vector to capture the spatial relationship between patches. The sequence of embedded patches is then passed through multiple transformer blocks, where self-attention is used to compute a weighted sum of the patch embeddings. The resulting vector, along with the original positional encoding, is passed through the feedforward network to produce the final classification output.

The attention mechanism in each transformer block computes a weighted sum of the patch embeddings based on their similarity to each other. This is done using three learned matrices: query, key, and value. For a given patch i, the attention scores with respect to all other patches are computed as follows:

$$\text{Attention}(q_i, K, V) = \text{softmax}\left(\frac{q_i K^T}{\sqrt{d_k}}\right) V$$

where $\mathbf{q_i}$ is the query vector for patch **i**, **K** and **V** are the key and value matrices for all patches, and $\mathbf{d_k}$ is the dimensionality of the key vectors. The softmax function normalizes the attention scores so that they sum to 1.

Overall, the ViT architecture achieves state-of-the-art performance on several benchmark image classification datasets, including ImageNet, with significantly fewer parameters than previous top-performing models. The success of the ViT has also inspired further research on the use of transformer-based architectures in computer vision.

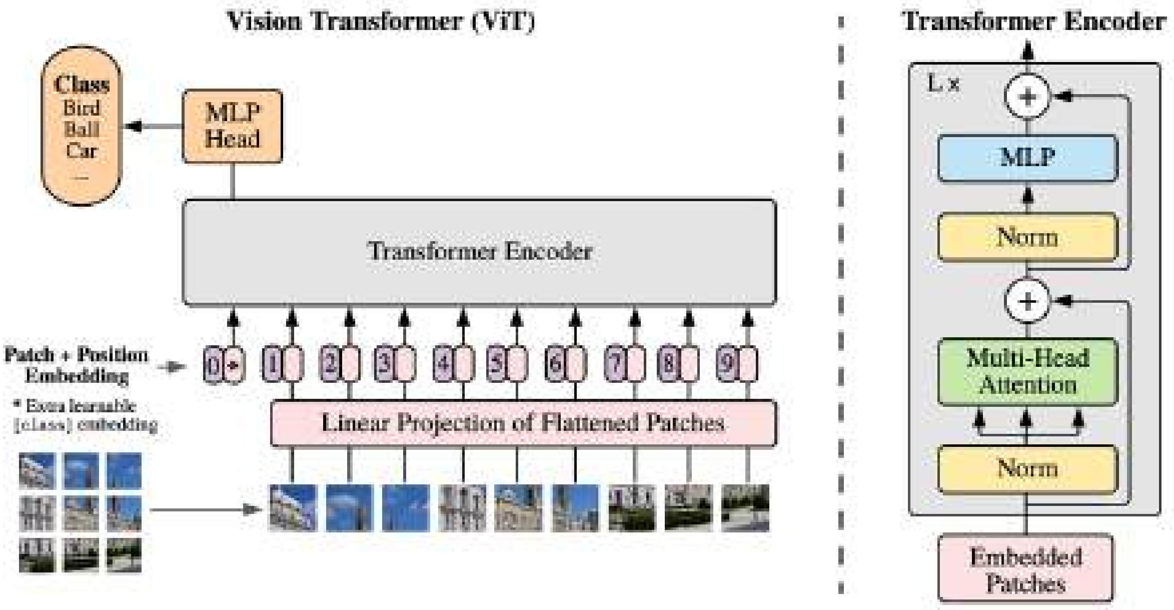


Figure 4: Block Diagram of Vision Transformer

3.8 Activation Functions:

Activation function decides whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

The choice of activation function in the hidden layer will control how well the network model learns the training dataset. The choice of activation function in the output layer will define the type of predictions the model can make.As such, a careful choice of activation function must be made for each deep learning neural network project.

3.8.1 ReLU

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

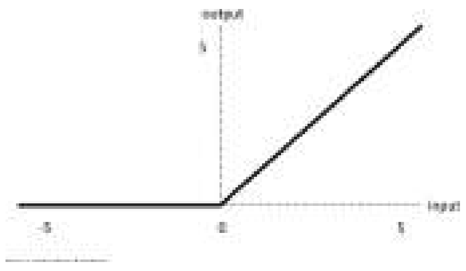


Figure 5: ReLU

3.8.2 Softmax

Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. The most common use of the softmax function in applied machine learning is in its use as an activation function in a neural network model. Specifically, the network is configured to output N values, one for each class in the classification task, and the softmax function is used to normalize the outputs, con- verting them from weighted sum values into probabilities that sum to one. Each value in the output of the softmax function is interpreted as the probability of membership for each class.

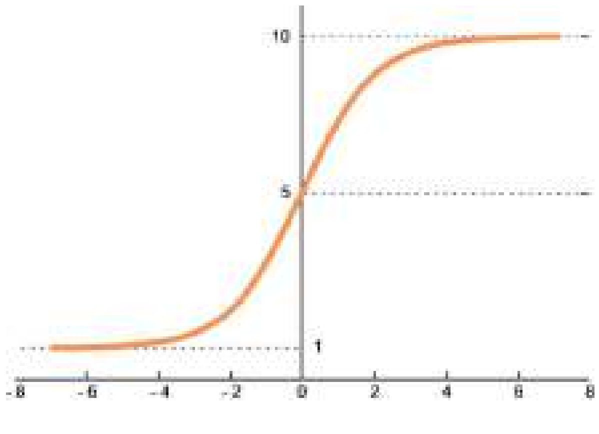


Figure 6: Softmax

3.9 Pooling Layers

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common approach used in pooling is max pooling.

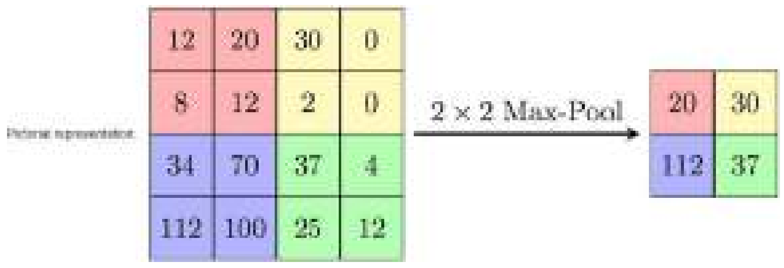


Figure 7: Softmax

3.10 Adam Optimizer

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). Specifically,the algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages.

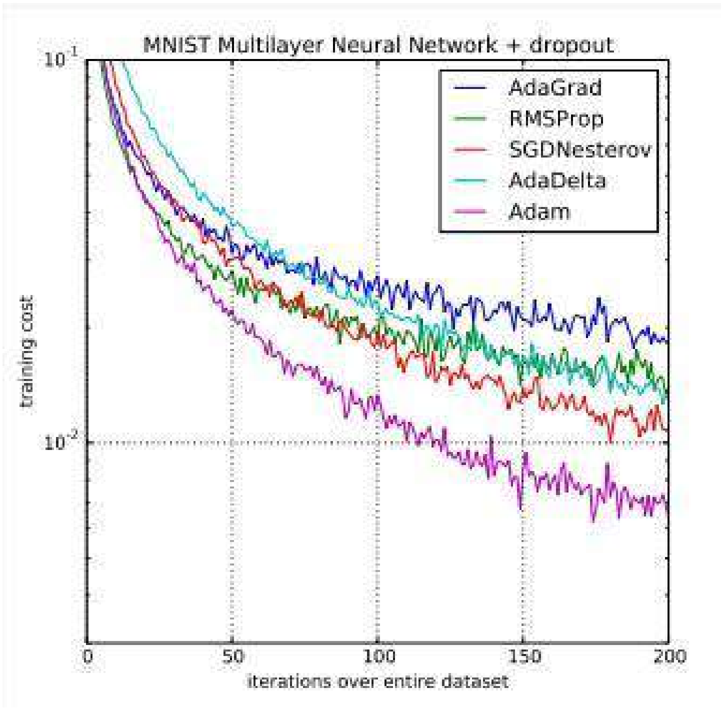


Figure 8: Adam Optimizer

4 METHODOLOGY

4.1 System design

The handwritten Equation Recognizer provides a UI integrated into ckeditor5, which provides almost all features of a rich document editor. We have added a draw pad UI in which equations are drawn and detected to give an output prediction. The approach we took to achieve this goal is to use a pre-trained model and fine tune it on a dataset consisting of handwritten characters. For the interface, we created a plugin for ckeditor5 that allows users to draw their equations and edit them after they have been recognized.

4.2 System Block Diagram

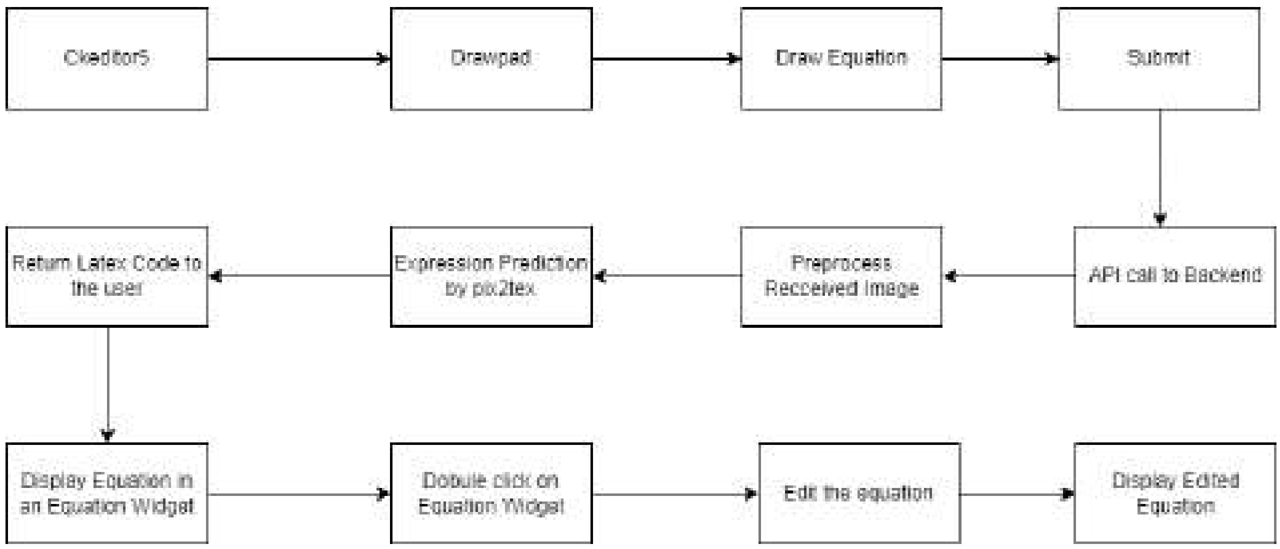

Ckeditor5
Drawpad
Draw Equation
Submit
API call to Backend
Preprocess Received Image
Expression Prediction by pix2tex
Return Latex Code to the user
Display Equation in an Equation Widget
Dobule click on Equation Widget
Edit the equation
Display Edited Equation

Figure 9: System Block Diagram for our new ckeditor based app

4.3 Image Preprocessing

When the image is uploaded to the server, it must go through phases of preprocessing before providing it as input to the model. At first, the RGB image is converted into a single channel on grayscale. All the succeeding processes start from this grayscale image. Then, we convert the size and resolution of the input image to match those of the images in the training dataset, which is necessary for good accuracy.

4.4 Ckeditor Architecture

CKEditor 5's architecture is built around a modular and extensible framework, which allows developers to customize and extend the editor to fit the needs of their applications. Some of the key features of the architecture are:

Data model: CKEditor 5 uses a data model to represent the content being edited. The model is designed to be flexible and extensible, and can represent a wide range of content types.

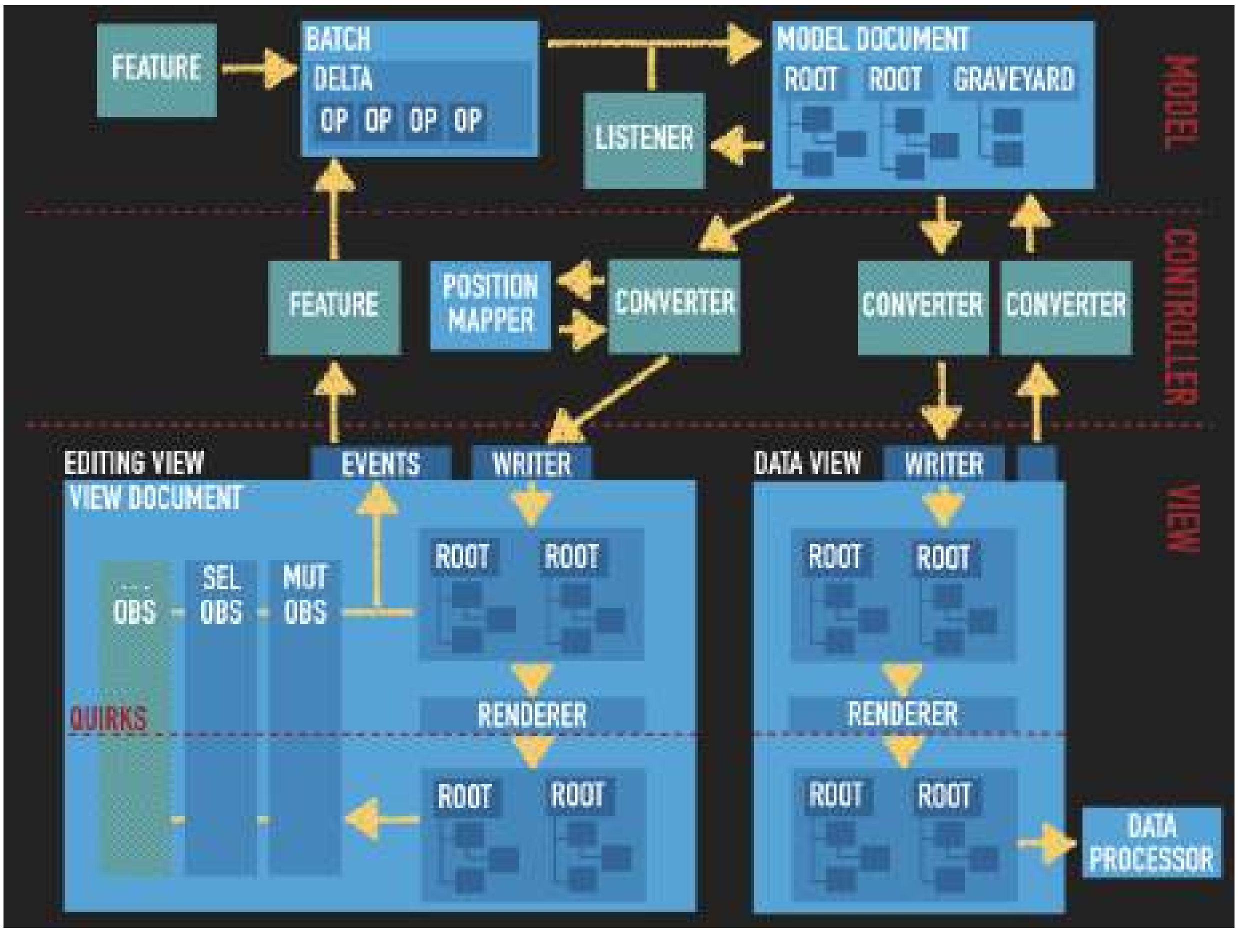
Rendering engine: CKEditor 5 uses a rendering engine to convert the content from the data model into HTML, which can be displayed in the editor and on the web page.

User interface: CKEditor 5's user interface is built using a component-based architecture, with each component responsible for a specific feature or functionality. The user interface is highly customizable, with the ability to add, remove, and rearrange toolbar buttons, menus, and other interface elements.

Plugins: CKEditor 5's functionality is implemented through a set of modular plugins, which can be easily added, removed, or customized. Plugins can be used to add features like image editing, spell checking, and real-time collaboration.

Events system: CKEditor 5's event system allows developers to listen for and respond to various events, such as when the content changes, or when the user interacts with the editor.

Ckeditor provides a highly modular interface, which makes it easy to implement any kind of functionality and integrate with it. The following figure shows its internal modules, and how Ckeditor works in general:



4.5 Model Design

For the model, we have used an open source model ‘pix2tex’, and fine tuned it on a handwritten characters dataset. ‘pix2tex’ uses resnet architecture as a backbone network for feature extraction, in the initial stage. Then it uses the transformer’s encoder-decoder network to output the latex code. The architecture of Vision transformer used was shown in figure 4 of section 3.7.

4.6 Software Requirements

4.6.1 Python

Python is a high level, general purpose, interpreted, dynamic programming language. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles.

4.6.2 Numpy

Numpy is a high performance numeric programming extension for python. It is the core library for scientific computing in python.

4.6.3 Pytorch

PyTorch is an open-source machine learning framework developed by Facebook. It is built on top of the Torch library, and provides an easy-to-use interface for building neural networks and other machine learning models. PyTorch is known for its dynamic computational graph, which allows for more flexible and efficient model development. It also offers several other features, such as built-in support for automatic differentiation, support for distributed computing, and integration with other popular libraries like NumPy and SciPy. PyTorch has gained popularity among researchers and developers due to its ease-of-use, flexibility, and the ability to rapidly prototype complex models.

4.6.4 ckeditor5

CKEditor 5 is a powerful, modern, customizable text editor that provides developers with a framework for building rich text editors in web applications. It is a complete rewrite of the previous version (CKEditor 4) and offers a modular architecture that allows developers to create custom builds by including only the features they need.

CKEditor 5 provides a range of features for formatting and styling text, including the ability to apply headings, bold, italic, underline, and strike-through styles. It also supports inline and block-level elements, tables, links, images, and more. CKEditor 5 is highly customizable and offers a range of plugins that can be used to add additional functionality.

The editor is built using modern web technologies such as ES6, React, and Redux, and is designed to work seamlessly with a wide range of modern browsers and devices. CKEditor 5 offers a clean, modern UI that is easy to use and highly intuitive, making it an excellent choice for building powerful text editors for web applications.

4.6.5 Mathtype

MathType is a software application used for creating and editing mathematical equations, formulas, and symbols. It is widely used in academic and scientific fields for creating complex mathematical expressions for research papers, textbooks, presentations, and online documents. MathType provides an intuitive graphical user interface for creating and editing equations, and supports various input modes, including keyboard shortcuts, handwriting recognition, and copy-paste from other mathematical software. It also supports integration with other applications such as Microsoft Word and Google Docs. MathType can export equations in a variety of formats, including LaTeX, MathML, and image formats such as JPEG, PNG, and GIF. The software is available for both Windows and Mac operating systems.

4.6.6 Mathlive

MathLive is an open-source JavaScript library that provides a rich user interface for mathematical input and rendering. It allows users to easily input and edit mathematical expressions and equations, as well as render them in a variety of formats, including LaTeX, MathML, and Unicode.

MathLive is designed to be flexible and customizable, allowing developers to integrate it into a wide range of web-based applications and platforms. It supports a variety of input methods, including keyboard input, handwriting recognition, and speech recognition, and can be used to create interactive math quizzes, tutorials, and other educational materials.

The library provides a number of powerful features for working with mathematical expressions, including support for symbolic manipulation, equation solving, and graphing. It also includes a built-in expression editor that allows users to easily edit and format expressions in a variety of ways.

4.6.7 OpenCV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

5 IMPLEMENTATION

5.1 UML Diagrams

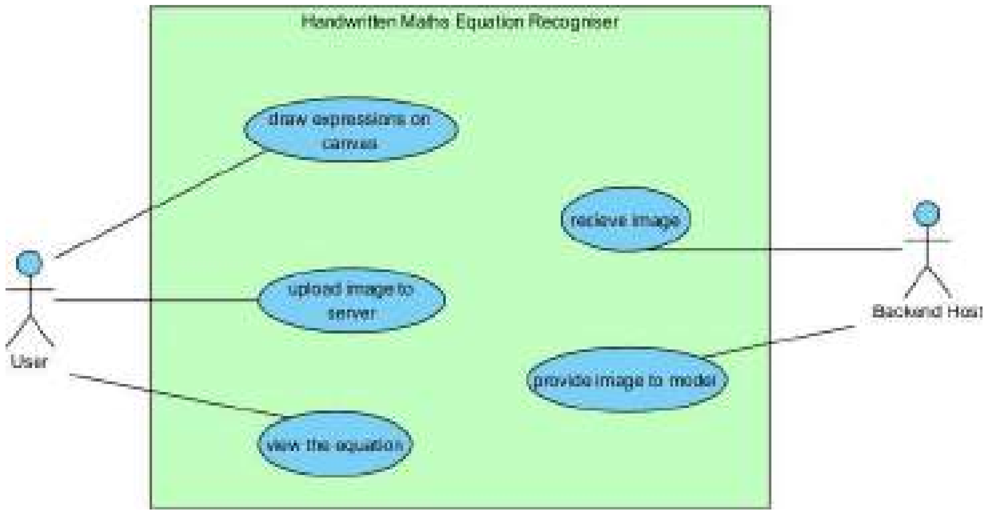


Figure 13: Use Case Diagram

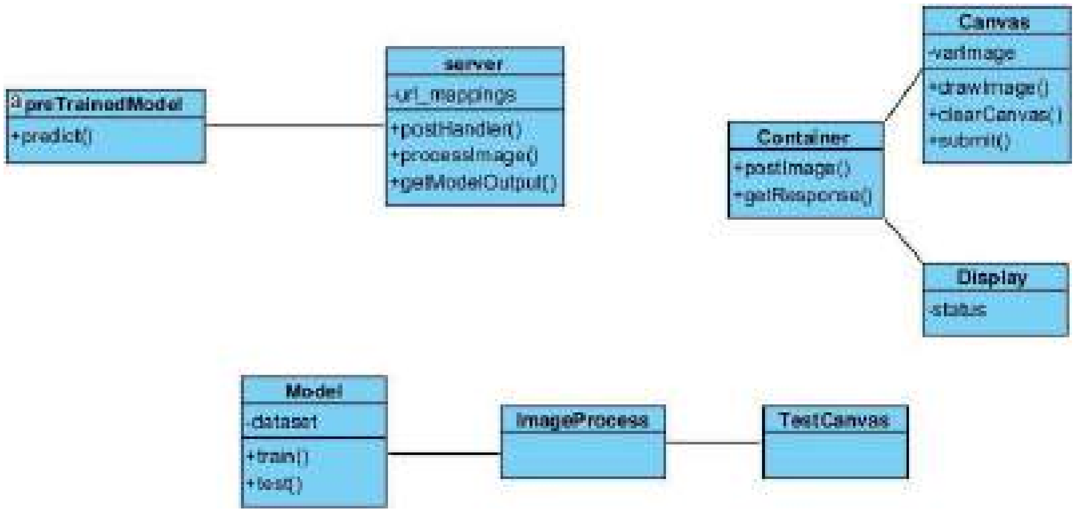


Figure 14: Class Diagram

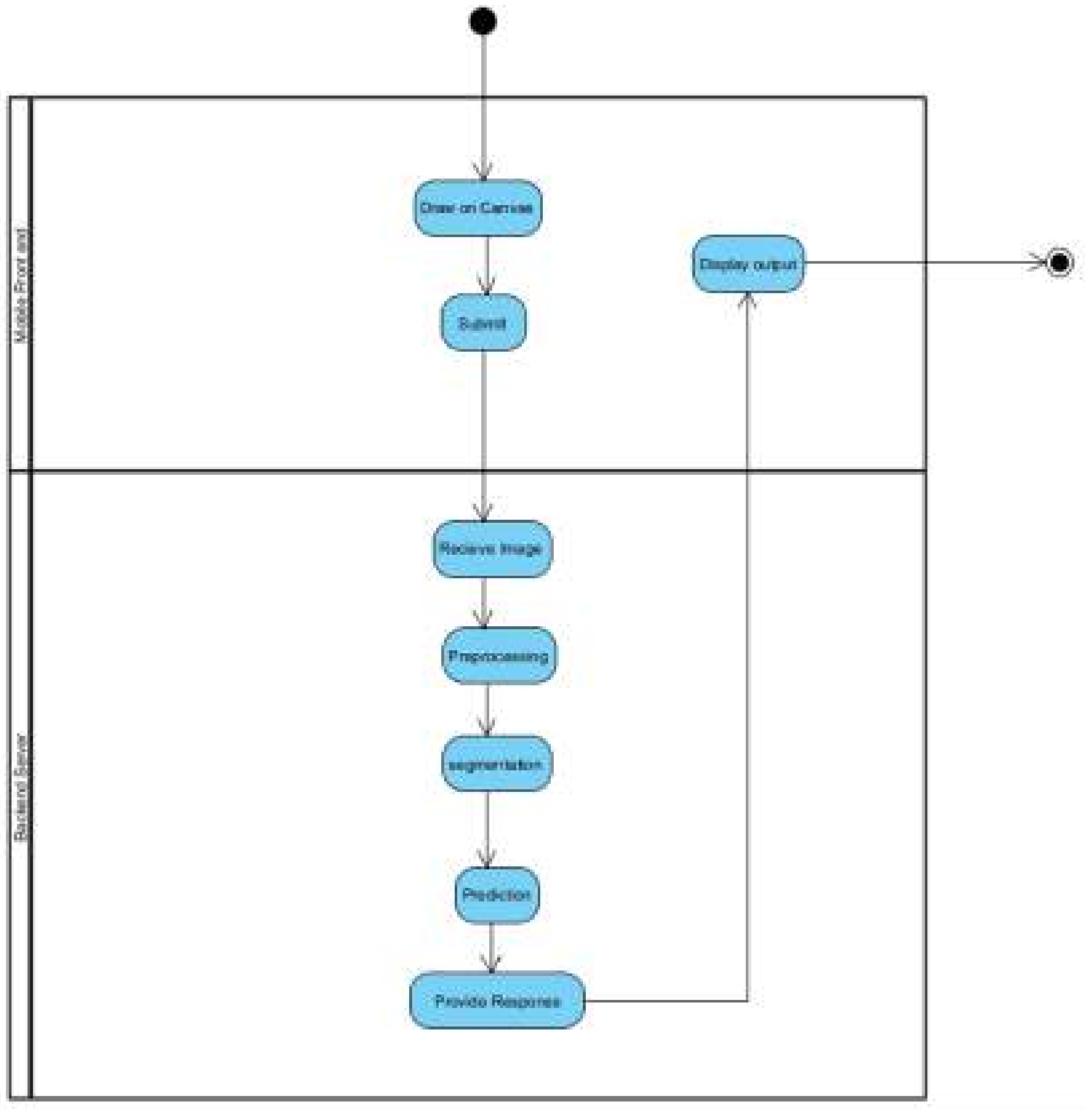


Figure 15: Activity Diagram

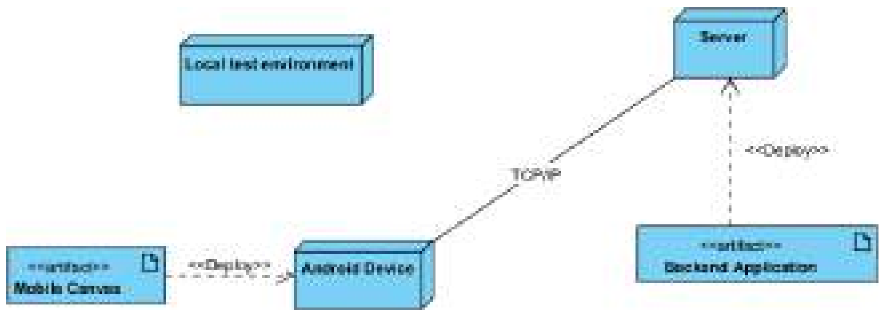


Figure 16: Deployment Diagram

5.2 Technologies Used

Our project is a mobile app with a backend server to refer services from. It uses client server based architecture for data movement over the HTTP or HTTPS protocol. For the development of the system various technologies as follows were used:

a. Programming Languages

. Python3 (For Deep Learning and Backend)

. Jupyter for easy testing during development

b. Frameworks

. Pytorch(For Deep Learning)

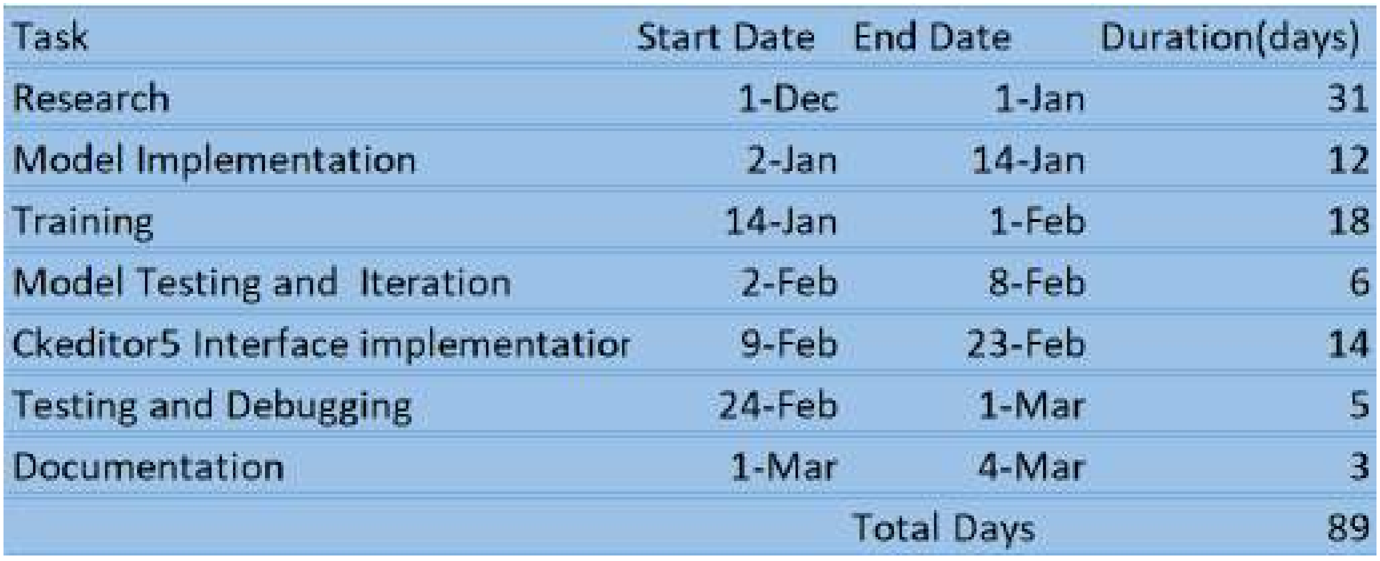
. Streamlit (For Backend Implementation)

. FastAPI for backend

. Ckeditor5 for editor

6 EPILOGUE

6.1 Project Schedule



Task	Start Date	End Date	Duration(days)
Research	1-Dec	1-Jan	31
Model Implementation	2-Jan	14-Jan	12
Training	14-Jan	1-Feb	18
Model Testing and Iteration	2-Feb	8-Feb	6
Ckeditor5 Interface implementatior	9-Feb	23-Feb	14
Testing and Debugging	24-Feb	1-Mar	5
Documentation	1-Mar	4-Mar	3
		Total Days	89

Figure 17: Project Schedule

6.2 Limitations

As we moved further with our work, because of the limited time and resources, there were some limitations

- The model has bias as sometimes it seems to mistake a new input expression with another similar expression in the training dataset
- The model has lower accuracy for single characters.

6.3 Future Enhancements

The following are the enhancements that we can do to increase the productivity of our project.

- Accuracy can be increased by using a larger dataset with both small and large expressions(based on number of terms) which can reduce the bias and the model hallucinating extra terms for small handwritten expressions.
- We can add a search functionality where the key is the drawn expression.
- The scope of our app can be expanded for math expression segmentation so that we can detect individual steps from solutions to math problems.

7 RESULTS

The equation is drawn by gesture control on the canvas and the result is obtained on the text box below. Even though, the model had accuracy of about 95% the accuracy in the mobile platform is little concerning.

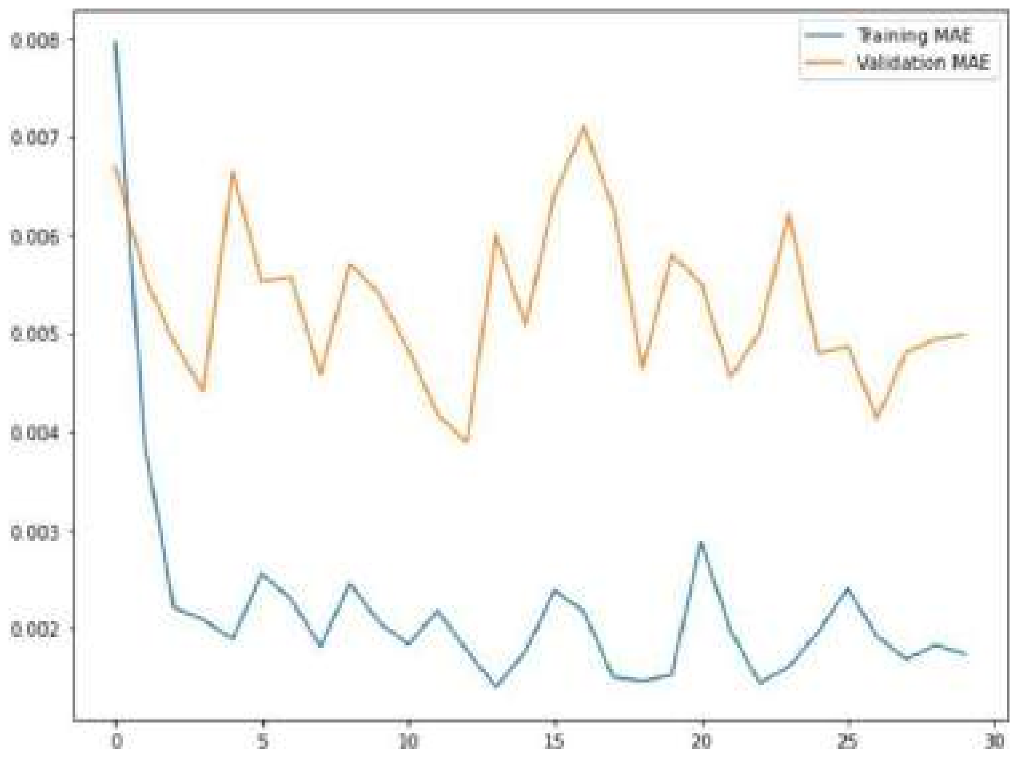


Figure 18: Training and Validation Errors

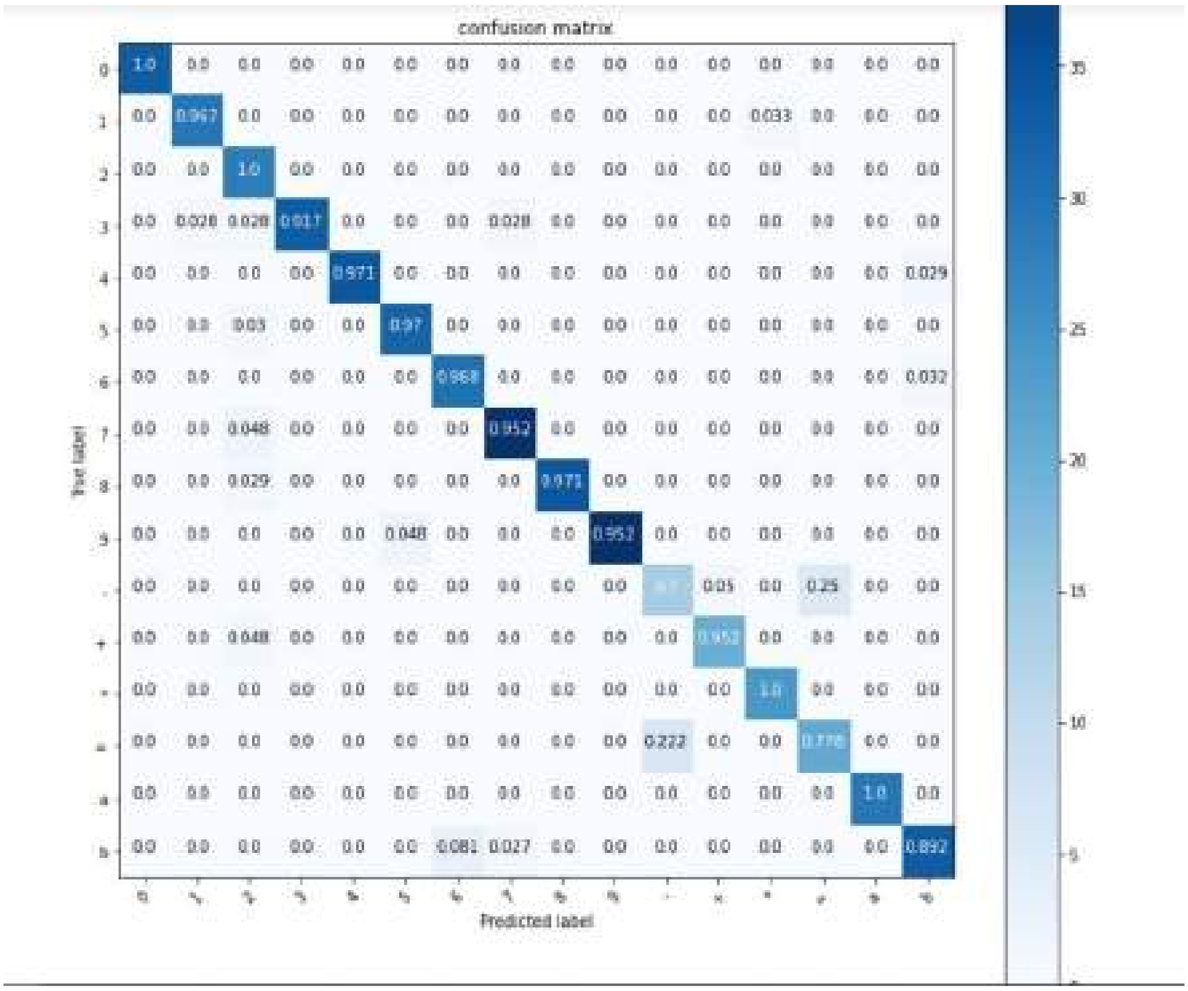


Figure 19: Confusion Matrix

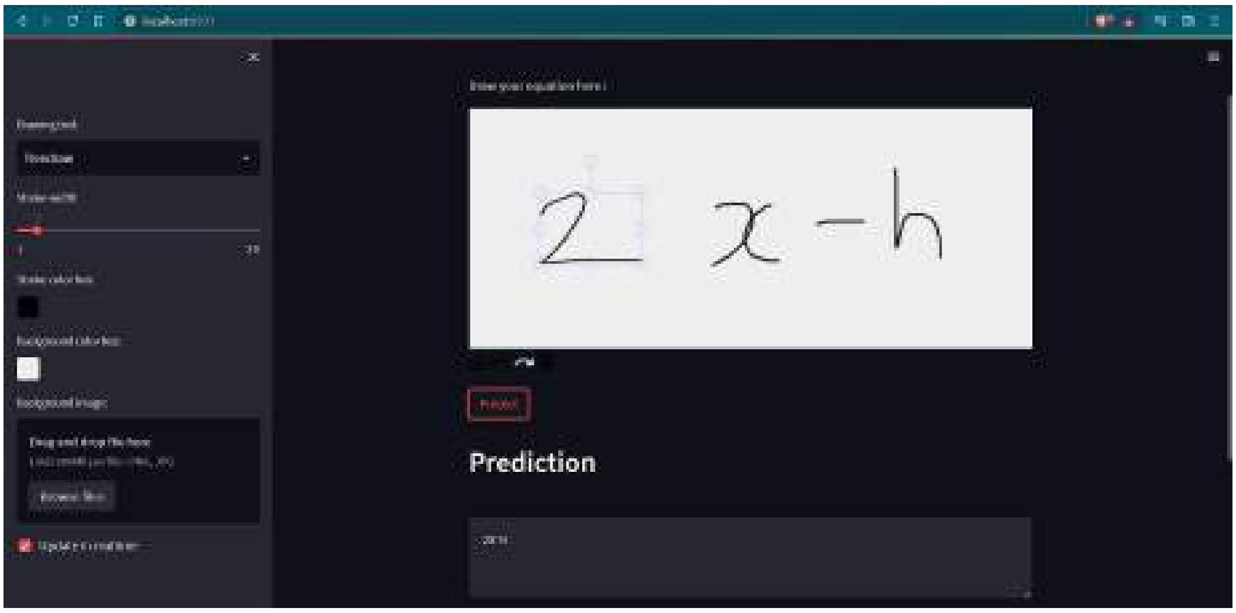
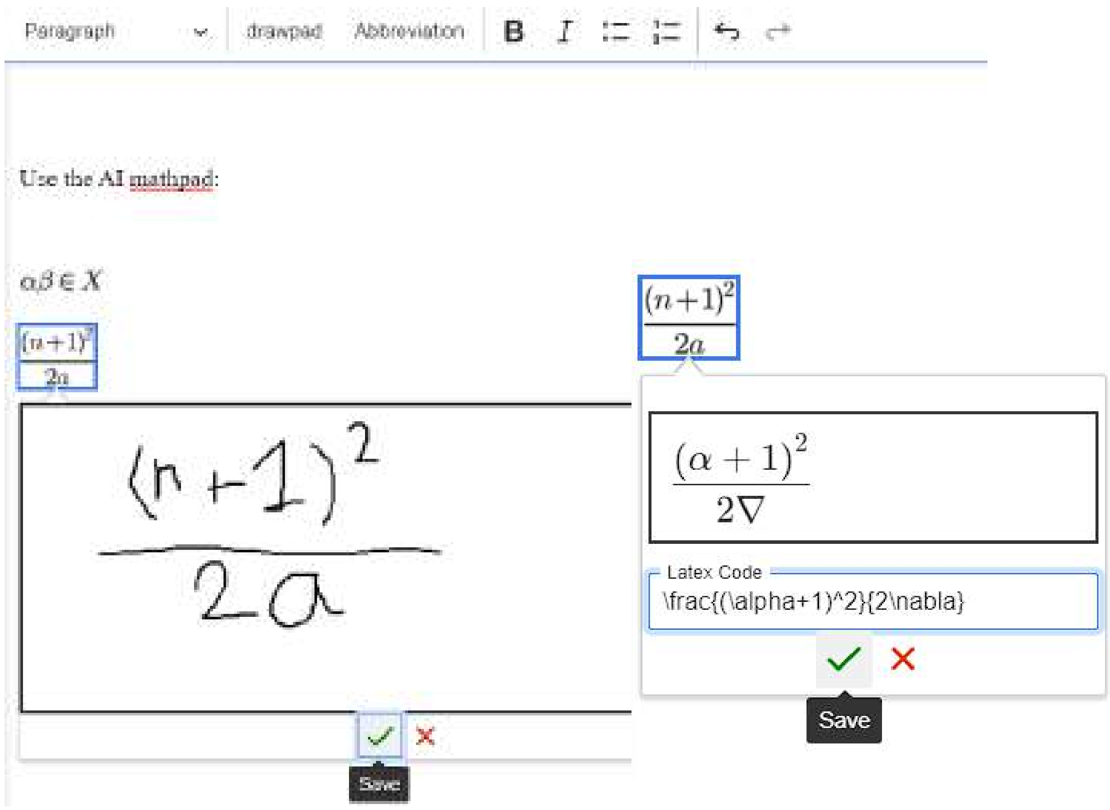


Figure 20: Our Web App in Streamlit



Use the AI mathpad:

$\alpha\beta \in X$

$\frac{(n+1)^2}{2a}$

$\frac{(\alpha+1)^2}{2\nabla}$

Latex Code

\frac{(\alpha+1)^2}{2\nabla}

Figure 21: Our Ckeditor5 Web App

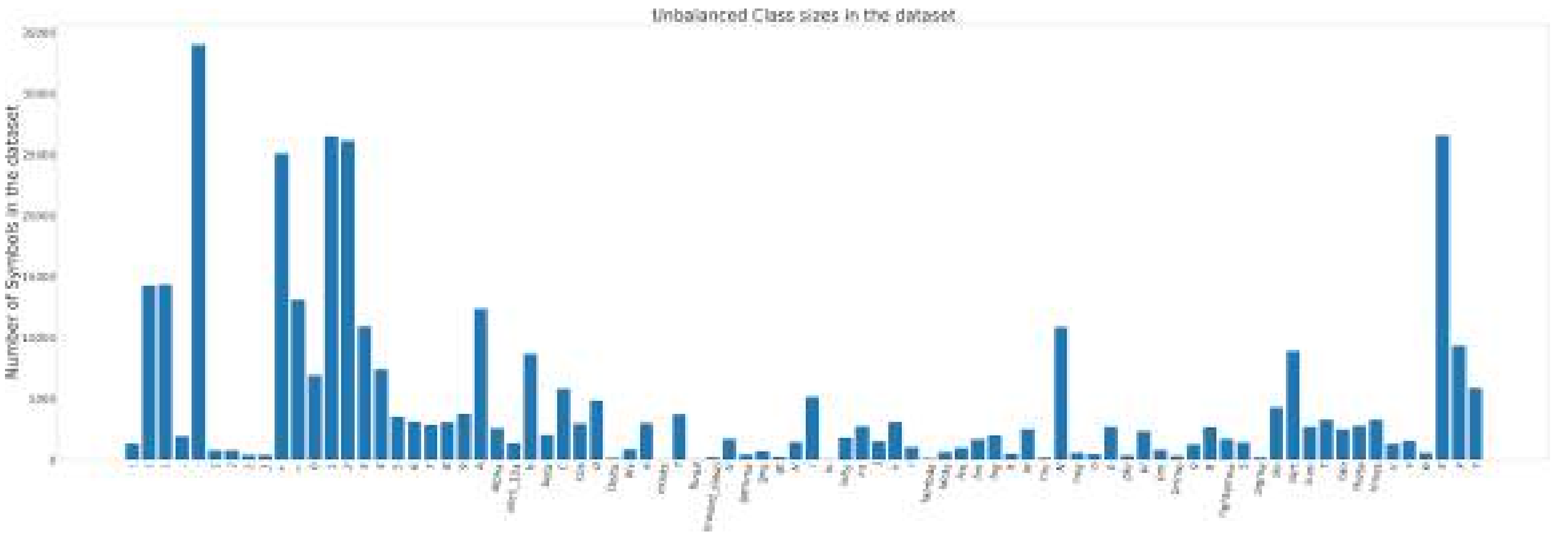


Figure 22: Numbers of classes in the Dataset

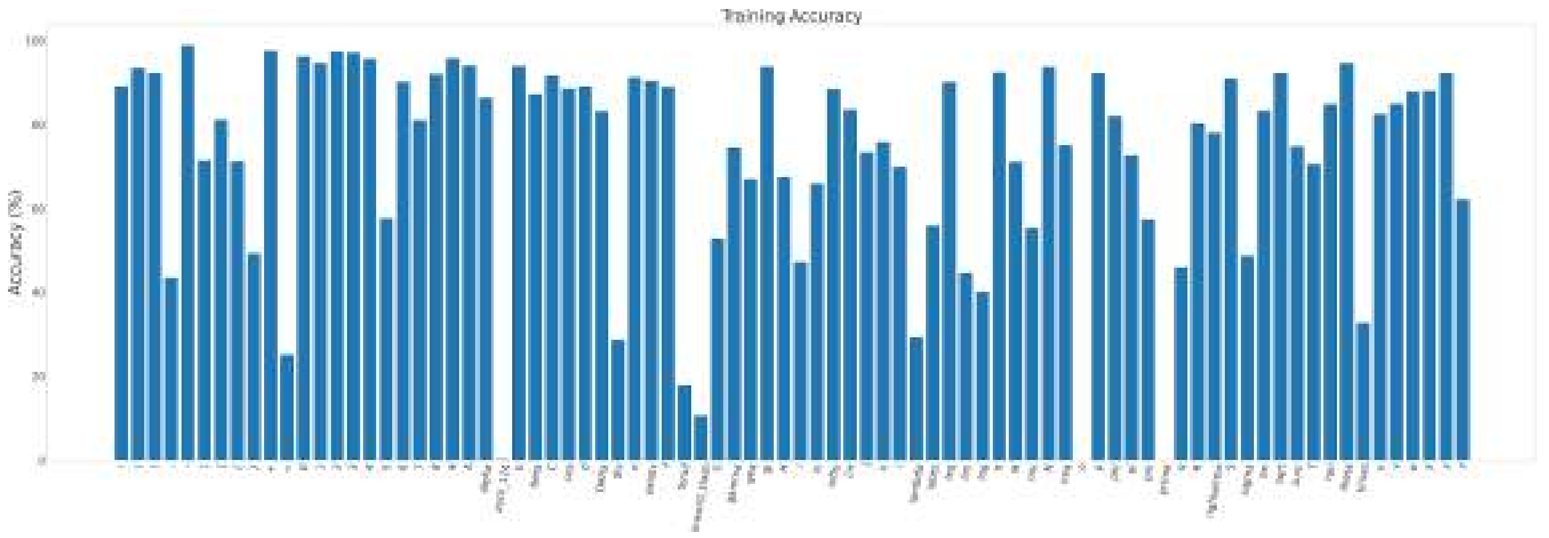


Figure 23: Training Accuracy for each class

8 CONCLUSION

The project sucessfully completed its ojectives of developing an application for recognising expressions upto the level of grade 8 mathematics.

For solving the requirements pix2tex model based on vision transformers was used. Dataset collection was done by various sources through internet. The trained model has an accuracy of 95%.

We trained our model on 82 symbols common in mathematics for finetuning the pix2tex dataset. But because of the dataset being imbalanced and our model being simple as well as due to the limited hardware we had for training our model couldn’t accurately recognize all the symbols.

We implemented a web app in streamlit and another web app consisting of a rich text editor, with an AI mathpad to interact with the model. In the editor web application, when submit is clicked after drawing the expression in the drawpad, it sends request to the model and the computation takes place. The equation obtained is shown in the editing area. The equation can also be edited to correct or change the equation, as desired.

Satisfactory output was obtained, but more refinements can be ammended for higher acccuracy and functionality.

9 REFERENCES

1 R. Zanibbi, D. Blostein. Recognition and retrieval of mathematical expres- sions. IJDAR 15, 331–357 (2012).

2 R.H. Anderson. Syntax-Directed Recognition of Hand-Printed Two-Dimensional Equations. PhD thesis, Harvard University, Cambridge, MA, 1968.

3 G. Labahn, E. Lank, S. MacLean, M. Marzouk, and D. Tausky. Mathbrush: A system for doing math on pen-based devices. In Proc. Work. Document Analysis Systems, pages 599–606, Nara, Japan, 2008.

4 A. Francisco and A.S. Joan“Comparing Several Techniques for Offline Recognition of Printed Mathematical Symbols”, IEEE,ICPR 2010.

5 M. Christopher, S. Masakazu, and U. Seiichi, “Support Vector Machines for Mathematical Symbol Recognition”, Technical Report of IEICE.

6 S. Fotini, K. Vassilis and C. George, “A Template Matching Distance for Recognition of On-Line Mathematical Symbols”, Institute for Language and Speech Processing of Athena - Research and Innovation Center in ICKT, Athens, Greece.

7 D.B. Dipak et. al,“A new approach for recognizing offline handwritten mathematical symbols using character geometry”, International Journal of Innovative Research in Science, Engineering and Technology Vol. 2, Issue 7, July 2013, ISSN: 2319-8753

8 Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. CoRR, abs/2010.11929.

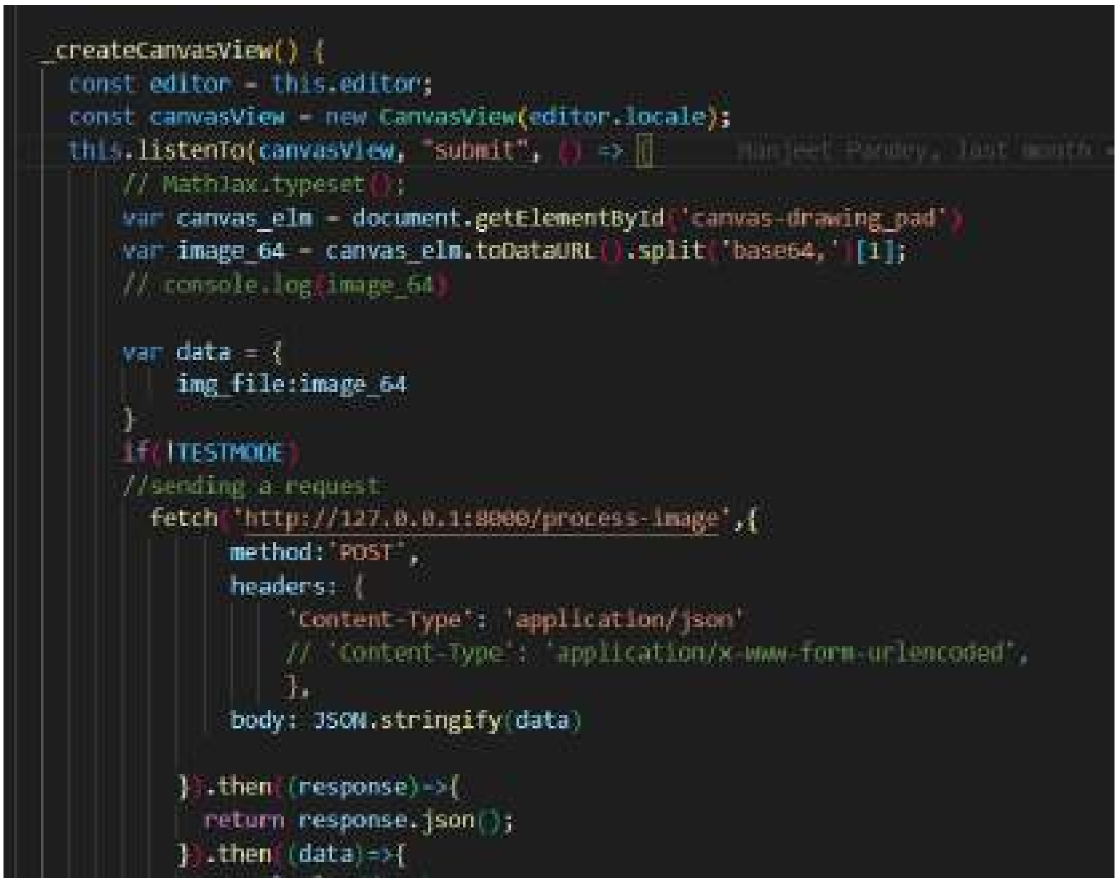
Retrieved from https://arxiv.org/abs/2010.11929

9 Blecher, Lukas. "LaTeX-OCR: Extract Text from PDFs and Images Using OCR." GitHub, 2021, https://github.com/lukas-blecher/LaTeX-OCR.

10 CKEditor 5 documentation. (n.d.). CKEditor. Retrieved March 16, 2023, from https://ckeditor.com/docs/ckeditor5/latest/index.html

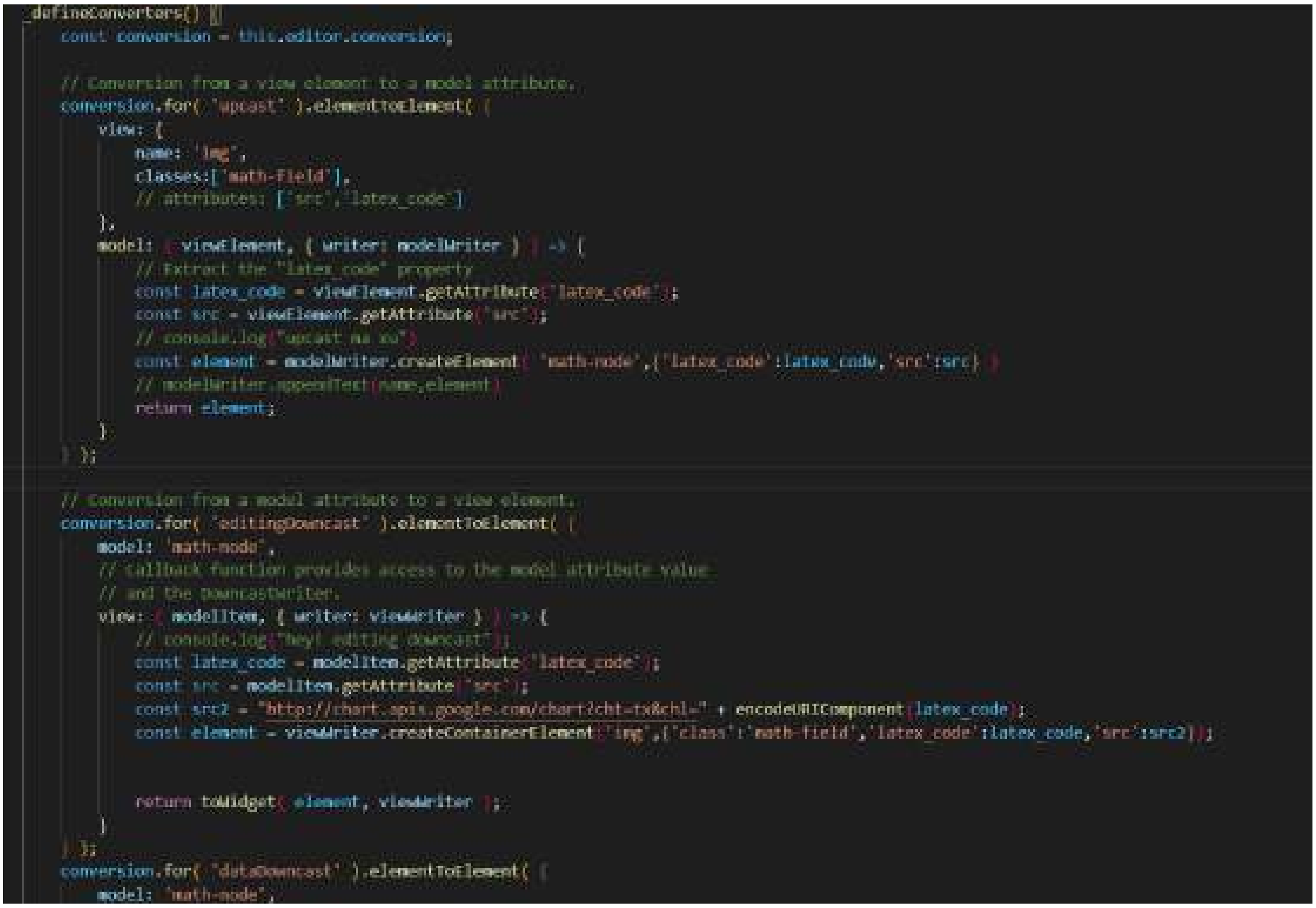
10 APPENDIX

Code Snippets:



```
_createCanvasView() {
    const editor = this.editor;
    const canvasView = new CanvasView(editor.locale);
    this.listenTo(canvasView, "submit", () => {      Manjeet Pandey, last month •
        // MathJax.typeset();
        var canvas_elm = document.getElementById('canvas-drawing_pad')
        var image_64 = canvas_elm.toDataURL().split('base64,')[1];
        // console.log(image_64)

        var data = {
            img_file:image_64
        }
        if(!TESTMODE)
        //sending a request
          fetch('http://127.0.0.1:8000/process-image',{
                method:'POST',
                headers: {
                    'Content-Type': 'application/json'
                    // 'Content-Type': 'application/x-www-form-urlencoded',
                    },
                body: JSON.stringify(data)

            }).then((response)=>{
              return response.json();
            }).then((data)=>{
```

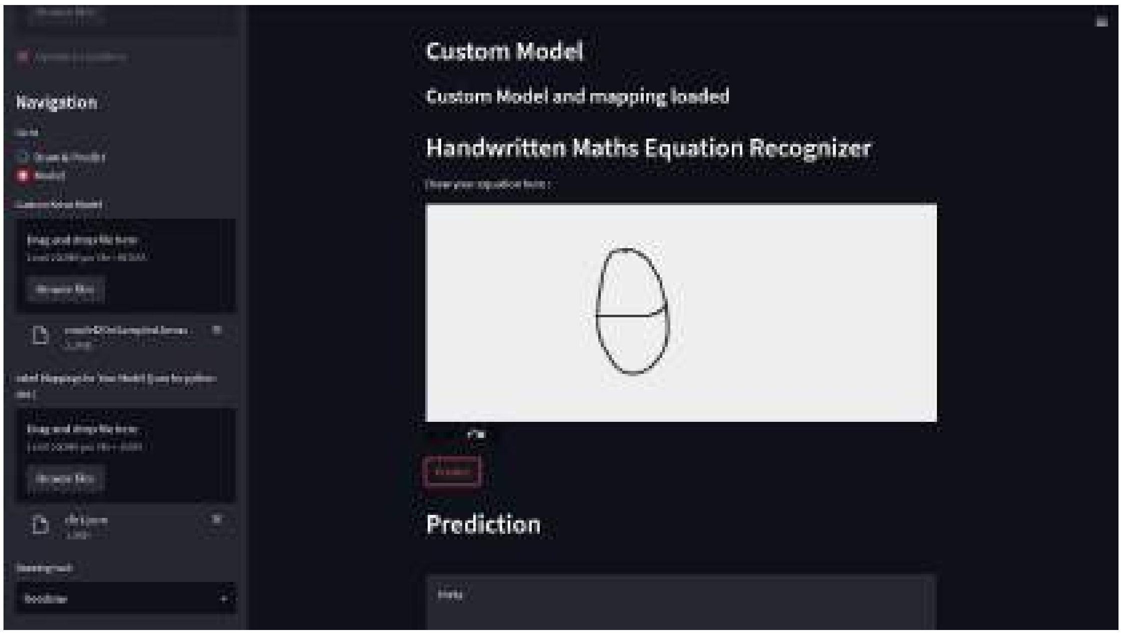
(a) Snippet of the code for communicating with backend



```
_defineConverters() {
    const conversion = this.editor.conversion;

    // Conversion from a view element to a model attribute.
    conversion.for( 'upcast' ).elementToElement( {
        view: {
            name: 'img',
            classes:['math-field'],
            // attributes: ['src','latex_code']
        },
        model: ( viewElement, { writer: modelWriter } ) => {
            // Extract the "latex_code" property
            const latex_code = viewElement.getAttribute('latex_code');
            const src = viewElement.getAttribute('src');
            // console.log("upcast ma xu")
            const element = modelWriter.createElement( 'math-node',{'latex_code':latex_code,'src':src} )
            // modelWriter.appendText(name,element)
            return element;
        }
    } );

    // Conversion from a model attribute to a view element.
    conversion.for( 'editingDowncast' ).elementToElement( {
        model: 'math-node',
        // callback function provides access to the model attribute value
        // and the DowncastWriter.
        view: ( modelItem, { writer: viewWriter } ) => {
            // console.log("hey! editing downcast");
            const latex_code = modelItem.getAttribute('latex_code');
            const src = modelItem.getAttribute('src');
            const src2 = "http://chart.apis.google.com/chart?cht=tx&chl=" + encodeURIComponent(latex_code);
            const element = viewWriter.createContainerElement('img',{'class':'math-field','latex_code':latex_code,'src':src2});


            return toWidget( element, viewWriter );
        }
    } );
    conversion.for( 'dataDowncast' ).elementToElement( {
        model: 'math-node',
```

(b) Snippet of mathnode model in ckeditor

Figure 24: Code Snippets



Custom Model

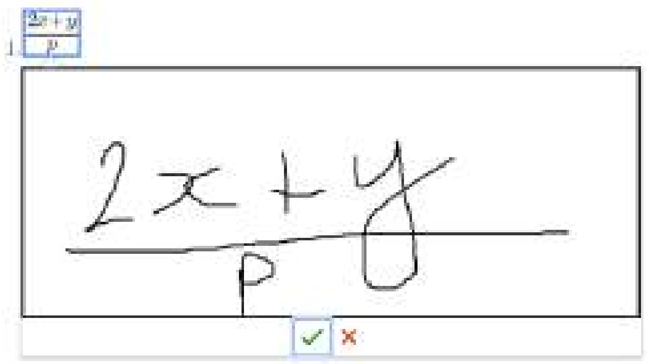
Custom Model and mapping loaded

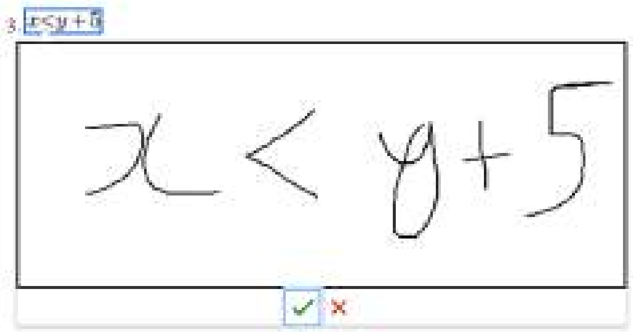
Handwritten Maths Equation Recognizer

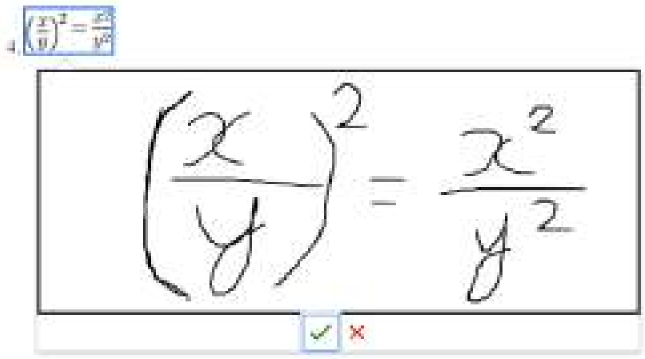
Prediction

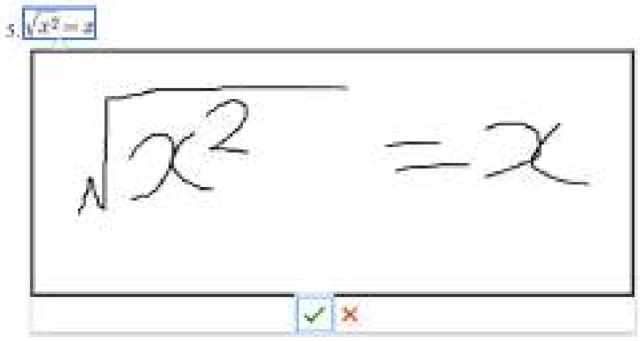
Figure 25: Using Custom Model Uploaded by User in Our Web App

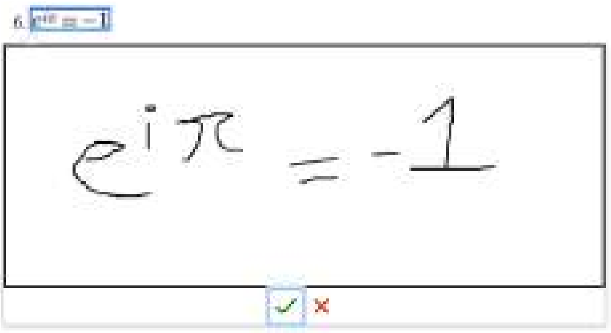
Outputs in the ckeditor interface for some tests:

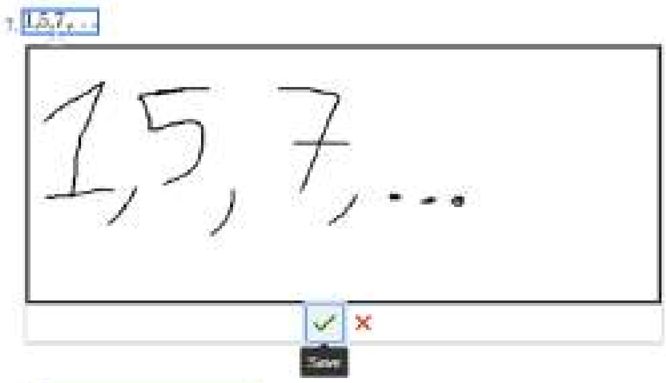

1. $\frac{2x+y}{p}$

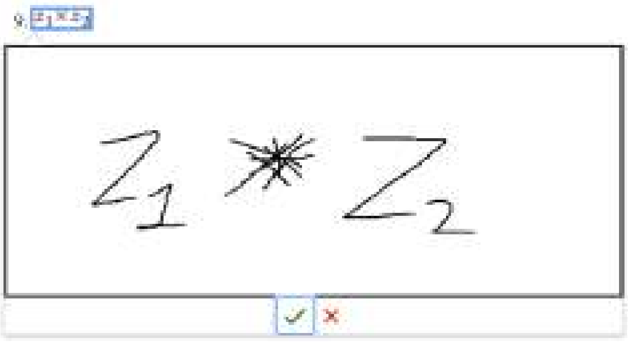

3. $x<y+5$

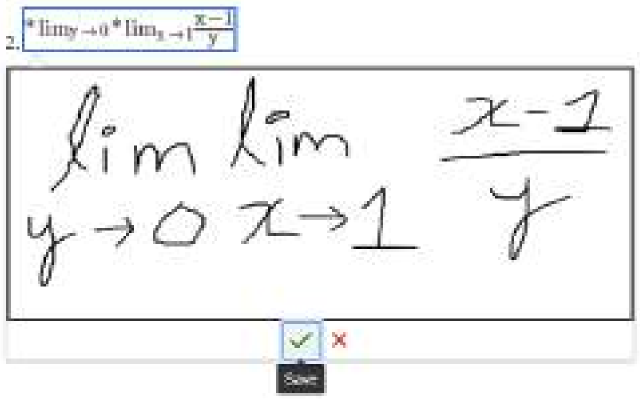

4. $\left(\frac{x}{y}\right)^2=\frac{x^2}{y^2}$

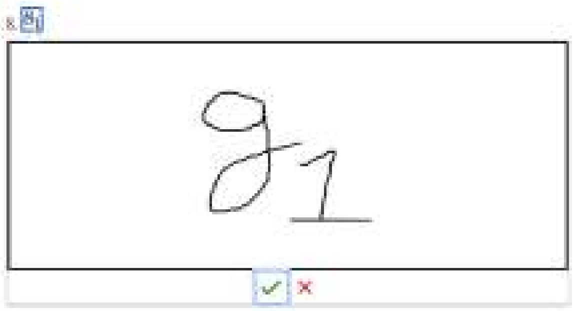

5. $\sqrt{x^2=x}$


6. $e^{i\pi}=-1$


7. $1,5,7,\ldots$

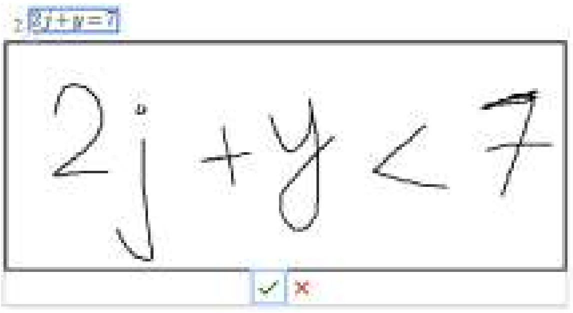

9. z_1*z_2


2. $*\mathrm{lim} y\rightarrow 0 * \mathrm{lim}_x\rightarrow 1\frac{x-1}{y}$



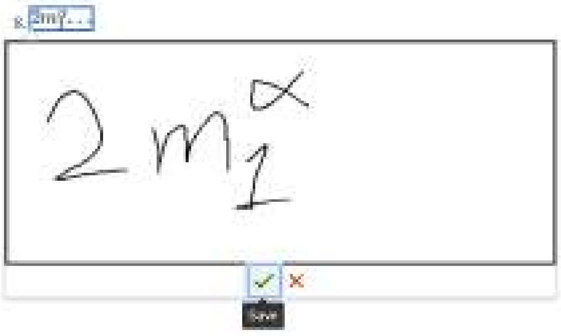
8. g_1

g_1



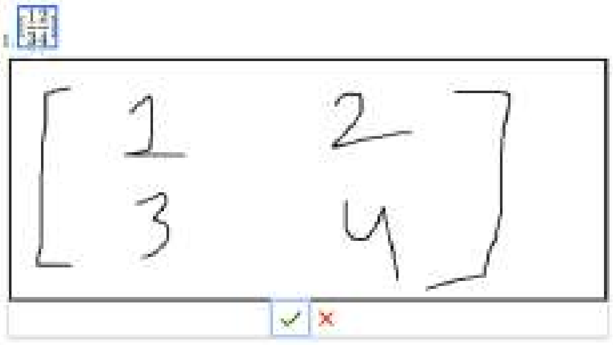
2. $2j+y=7$

$2j+y<7$



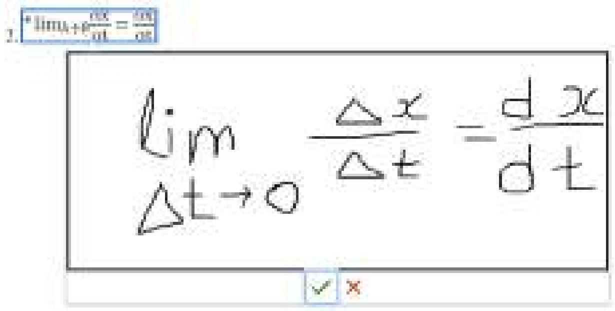
8. 2m?...

$2m_1^{\alpha}$



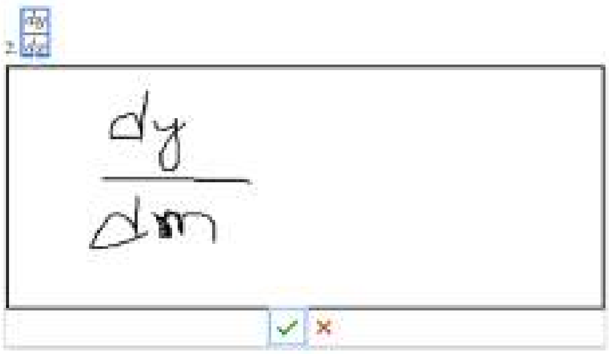
1. $\begin{bmatrix}1&2\\3&4\end{bmatrix}$

$\begin{bmatrix}1&2\\3&4\end{bmatrix}$



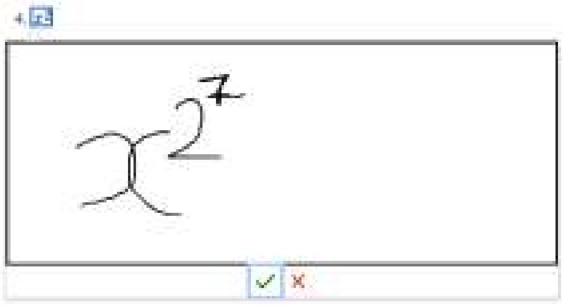
2. [illegible]

$\lim_{\Delta t\to 0}\frac{\Delta x}{\Delta t}=\frac{dx}{dt}$



2. [illegible]

$\frac{dy}{dm}$



4. [illegible]

$x^{2^{7}}$