Izmir Institute of Technology

Electronics and Communications Dep.

Final Project Proposal

Handwritten and Digital Mathematical Expression Recognition and Processing

This project aims to recognize and convert handwritten or digitally written mathematical expressions into a string format using computer vision and machine learning techniques. The developed system will analyze these expressions and perform mathematical operations such as differentiation and integration with limits etc.

Writing mathematical expressions by hand is a common practice in education, research, and daily use. This project aims to automate the solving process, saving time and reducing errors.

Project Requirements

Functional Requirements

- Recognition of handwritten mathematical expressions and conversion into digital format.
- Syntactic analysis of expressions and execution of mathematical operations (differentiation, integration, etc.).
- Production of accurate results as output.

Technical Requirements

- Software: Python, various python libraries (such as numpy, open-cv etc.), PyTorch
- A large dataset containing handwritten mathematical symbols.

Performance Requirements

- High accuracy rate (95%+).
- Fast processing time (Producing recognition within 1 seconds)

The project encompasses the following steps:

- Finding contours of handwritten expressions using OpenCV or custom algorithms and segmenting them into symbols.
- Creating an image classification model to recognize the segmented symbols (e.g., CNN-based)
- Determining mathematical meanings of symbols based on their positions
- Performing operations such as differentiation and integration.

System Architecture

This section outlines the design and components of the system for recognizing and processing handwritten mathematical expressions.

Input Module:

- Handles the initial input of the handwritten mathematical equation image.
- Python, with potential use of libraries for image handling (e.g., OpenCV, Pillow).

Preprocessing Module:

- Enhances the quality of the input image to improve symbol recognition (e.g., noise removal, thresholding)
- After converting the input image to a more usable form, it identifies the boundaries of individual symbols within the preprocessed image. (Finding contours)
- Python, with potential use of libraries like OpenCV for contour detection algorithms.
- Separates the identified contours into individual symbols.

Symbol Classification Module:

- Recognizes each segmented symbol using machine learning techniques.
- PyTorch for implementing a Convolutional Neural Network (CNN).

Mathematical Expression Interpretation and Output Module:

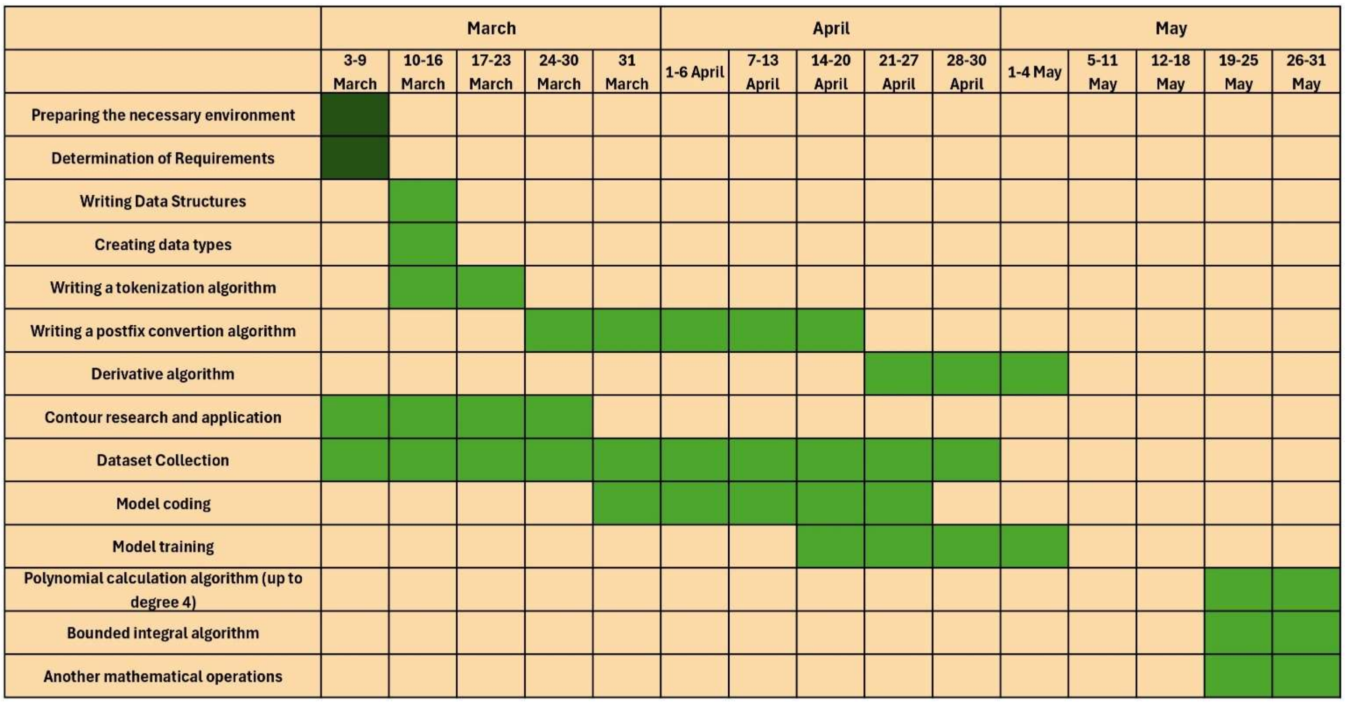
- Determines the spatial relationship between symbols, performs mathematical operation on the recognized expression, and presents the result in a readable format
- The module has tasks as follows: Positional analysis (e.g., identifying superscripts), string conversion of the expression, tokenization, postfix conversion, mathematical operations

In conclusion, the proposed system architecture is modular, scalable, and designed for accuracy. It combines advanced image processing, robust symbol classification, and comprehensive mathematical expression analysis to provide a seamless user experience. This modular design ensures easy maintenance and future enhancements, making it a solid foundation for a reliable and user-friendly system.

Project Plan and Timeline

The Gantt chart below illustrates the timeline of the project, it should be noted that we might not be able to stay stick to the timeline for all the tasks to be completed.

To do 🟩 Done ■



	March					April					May				
	3-9 March	10-16 March	17-23 March	24-30 March	31 March	1-6 April	7-13 April	14-20 April	21-27 April	28-30 April	1-4 May	5-11 May	12-18 May	19-25 May	26-31 May
Preparing the necessary environment	■														
Determination of Requirements	■														
Writing Data Structures		🟩													
Creating data types		🟩													
Writing a tokenization algorithm		🟩	🟩												
Writing a postfix convertion algorithm				🟩	🟩	🟩	🟩	🟩							
Derivative algorithm									🟩	🟩	🟩				
Contour research and application	🟩	🟩	🟩	🟩											
Dataset Collection	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩					
Model coding					🟩	🟩	🟩	🟩	🟩						
Model training								🟩	🟩	🟩	🟩				
Polynomial calculation algorithm (up to degree 4)														🟩	🟩
Bounded integral algorithm														🟩	🟩
Another mathematical operations														🟩	🟩

Task Distribution

A. Ender:
   a. Contour Research
   b. Model Building
   c. Model Training
   d. Writing postfix conversion algorithm

B. Bartu:
   a. Creating Token classes, data structures
   b. Tokenize algorithm for string expressions
   c. Creating algorithms of mathematical operations

C. Bartu & Ender:
   a. Dataset collection
   b. Model & Algorithm testing

Potential Risks and Mitigation Strategies

Insufficient or lack of diversity in the dataset. This could be a major problem and requires HIGH attention. Diversity in a dataset must be arranged since it directly concerns the accuracy rate and adaptation of the model to different handwriting types. In a scenario where the model is able to identify one person’s handwriting correctly but not the others would make the program unusable and pointless.

Another problem with the model would be long learning time, which is definitely not desired. Training deep learning models, especially those handling complex image-based tasks such as handwriting recognition, can be computationally expensive and time-consuming. If the learning process takes too long, it may not only slow down development but also make it impractical to iterate and improve the model efficiently.

To address the issue of insufficient diversity in the dataset, it is essential to collect handwriting samples from a wide range of individuals with varying writing styles. Additionally, data augmentation techniques, such as altering stroke thickness, slant, and spacing, can artificially increase dataset diversity and enhance model adaptability without requiring excessive data collection.

Regarding the long learning time, several optimization techniques can be applied. Simplifying the model architecture without significantly compromising accuracy can reduce computational complexity. Furthermore, leveraging hardware acceleration, such as GPUs can significantly reduce training time and make the process more feasible for large-scale implementation.