Mathematical Equation Solver Using Image Recognition

Keiry Gao (kg368), Derek Cao (dgc68), Vernetta Huang (vjh8)

AI Keywords: Computer vision

Application setting: Solve a written math equation

Section 1: Overview

We want to create a math solver that takes in an image of a math equation (handwritten or printed) and outputs the solution. It's often difficult and tedious to input your mathematical expressions into websites like WolframAlpha and Symbolab or input them into a calculator. Calculators return syntax errors and the UIs are small. A program that solves integrals, expectation problems, and uses mathematical constants such as Euler's number or Pi through image recognition replaces the usage of physical calculators, calculator websites, and speeds up calculations.

Section 2: Core AI

Our project roughly breaks up into two parts, both of which rely on computer vision. The first part would be image recognition for recognizing numbers and symbols in an image. This would be the first step that will be taken in working towards our project. In order to solve equations, it is first required that we have a system that can recognize where the numbers will be in the image, as well as the symbols that may also be part of the image. From there, we will be able to tie this into the second part of our project.

The second part of our project will be an image segmentation problem that groups equations together in the image and parses the equation to be put into an equation solve api like Wolfram Alpha. This part of the project will utilize the first part to recognize the different parts of the equation (the numbers, symbols, and equations) and then group the different equations that might be in the image. Once each equation has been identified and separated from each other, it will then parse each equation to be solved.

Section 3: Evaluating Our System

We will know how successful our project is by comparing the parsed output to our intended input whether it be printed math expressions or handwritten math expressions. We are going to test how the program parses poor handwriting, mathematical constants (Euler's number, Pi), correct evaluation of expressions, and edge cases. Testing the program against Euler's number and the number 9 is one case of bad handwriting testing. Edge cases include but are not limited to the following: empty inputs, empty integrals, proper fractions, complex numbers, and divide by 0.

We are going to use sample expressions ranging from elementary algebra to more complex math like trigonometry and calculus to test general correct evaluation of expressions. The difficulty of the problems can be organized by topics taught at each school grade level. It's also important for us to test speed because the whole point of this project is to beat the speed of inputting numbers and symbols into online or physical calculators. We will time how long it takes to handwrite simple and complex expressions and compare it to how long it takes for our program to process the input image and calculate the solution. Furthermore, we will also test in comparison to existing programs to see how ours compares (accuracy and speed).

Of course, we would also have to consider where and how our program is hosted if we intend to test how the program works on different machines. However, we will save this for the latter part of the project because cross-functionality is more of a stretch milestone.

Section 4: Timeline

- **Milestone 2: Project Proposal -** 2/16
- **Number Recognition -** 3/10
  - Be able to recognize both handwritten and printed numbers from images
  - We will test as we implement, being sure to test various handwritings and fonts
- **Forming Basic Equations -** 3/24
  - Be able to recognize both handwritten and printed arithmetic operations and symbols
  - Be able to parse basic equations
    - Elementary math and algebra
  - We will test as we implement, being sure to test various handwritings and fonts as well as all ways to represent an operation (for example / and ÷ for division)
- **Milestone 3: Status Report -** 3/27
  - We plan to have a draft done two days before the due date
- **Advanced Equations -** 4/28
  - Be able to recognize and solve more advanced equations such as integrals
  - This milestone will be done in sequence of problem difficulty and continue to add until the final submission
  - Our first goals would be trigonometry then calculus
  - We will test as we implement, being sure to test various handwritings and fonts as well as a wide variety of questions
- **Final Testing** - 5/12
  - Continue to build upon Advanced Equations milestone while implementing even more throughout testing
  - We will find volunteers to try our project
- **Presentation and Write Up -** 5/16
  - Complete both project presentation and final submission documents.
- **Milestone 4: Project Presentation -** 5/18
  - We plan to have a draft done two days before the due date
- **Milestone 5: Final Submission -** 5/18
  - We plan to have a draft done two days before the due date
  - By each milestone above, we will also document our work for our final report. We will continue to document everything in a living document throughout the semester.

Section 5: Resources

WolframAlpha: WolframAlpha is an existing platform that has similar functionality to what we wish our project to do. We hope to rely on this as a software resource regarding our solving methodology and math calculations.

MNIST: MNIST is a large collection of handwritten digits with a training set of 60,000 examples, and a test set of 10,000 examples. We hope to use this existing data resource as a start for our number recognition.

Section 7: References

We referenced WolframAlpha and Photomath as similar existing platforms when formulating our project.