Camera Calculator App

An investigation into the implementations of neural networks for handwritten mathematical equation recognition.

Jerry Liu

Candidate Number : Centre Number : Qualification Code :

Table of Contents

[Analysis 2](#_Toc114184194)

[Introduction 2](#_Toc114184195)

[Investigation of Existing Solutions 2](#_Toc114184196)

[Product 1 - Photomath 3](#_Toc114184197)

[Product 2 – Symbolab 5](#_Toc114184198)

[Product 3 – Math 7](#_Toc114184199)

[Proposed Solution and Limitations 8](#_Toc114184200)

[Further Research 9](#_Toc114184201)

[Modelling 9](#_Toc114184202)

[Image Processing 10](#_Toc114184203)

[Neural Networks 11](#_Toc114184204)

[Equation solving 12](#_Toc114184205)

[Objectives 12](#_Toc114184206)

[References 13](#_Toc114184207)

# Analysis

## Introduction

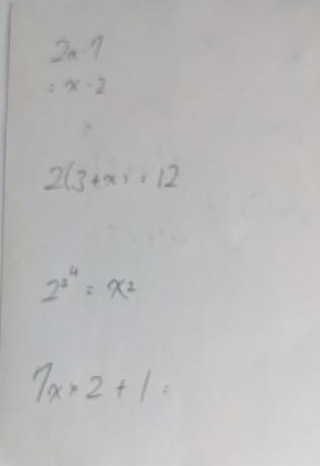
As a student, when tasked with completing math assignments and homework, seeking help from calculators is a common habit of many. However, as we progress through education the mathematical equations we face grow in complexity, where pre-installed calculators on phones and commonplace household calculators often offer little assistance. Online calculators although providing more functionality, transcribing mathematical expressions to text can often be a tedious task; some online calculators use the LaTeX language to express mathematical notation and symbols, but this requires users to learn its keywords. Other online calculators simply let users select symbols from a keyboard, which can also be time-consuming when dealing with longer expressions.

To improve the efficiency of inputting mathematical expressions and calculating them, I intend to create a mobile phone app which utilizes the camera functionality to recognize handwritten mathematical expressions and perform the desired calculations. I aim to be able to solve simple, one-variable linear equations and expressions, and gradually implement more functionality as the project progresses. The app will be designed for those who are looking for a quicker way to enter equations into their calculators.

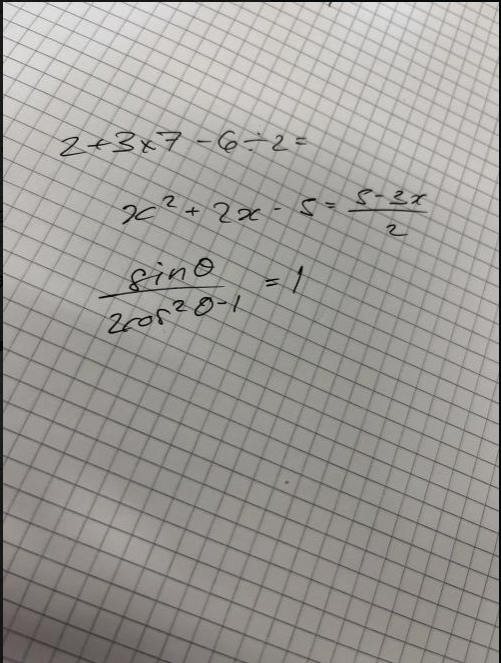
## Investigation of Existing Solutions

To better understand the functionality and uses of current solutions available to users, I’ve identified 3 calculator/equation solver apps from Google play with the features of photographic input to investigate in and collected reviews from the play store and math students within the college. The apps will be evaluated by their user interfaces relevant to the camera feature, the accuracy of results, error handling and overall functionality.

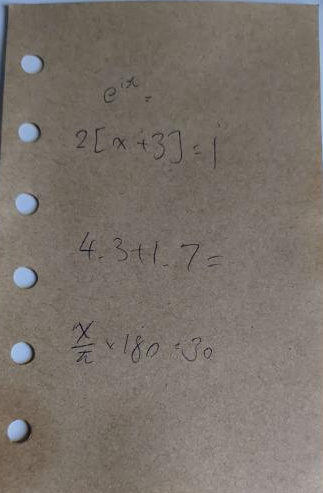
To test the accuracy of input and calculation. I’ve collected handwritten equations from different fields of math from varying handwriting on different backgrounds to test the recognition of the software.



▲*Blurred equations. Tests for recognition of equations written on multiple lines, indices and similar symbols (x and multiply sign)*



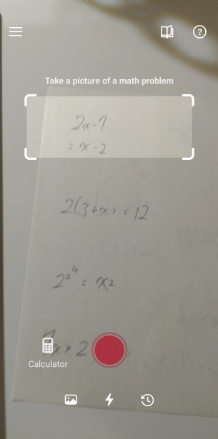
▲*Quadratics and trigonometry, test for recognition of theta and factors*



▲*Other symbols and brackets, including decimal points, pi and imaginary numbers.*

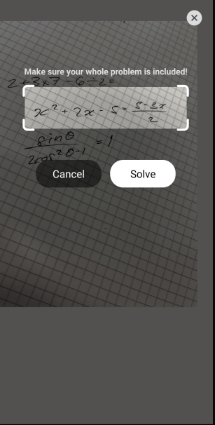
### Product 1 - Photomath

*Photomath* was able to recognize and solve most of the equations set including indices and variables such as theta; the only failure was when tasked to recognize equations written on different lines, to which an error message was outputted. I suspect that is due to the nature of how the images are broken down into individual symbols for recognition and how they are reassembled and is not due to the inability to compute the equation as the program was able to compute the result when I rewrote it on the same line.



▲*Camera mode GUI*

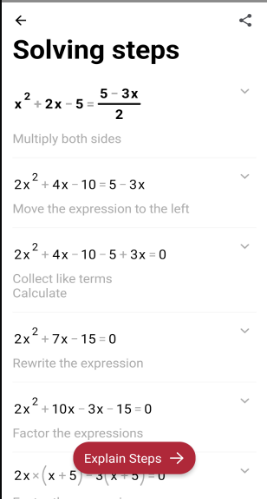
▲*Photo import GUI*



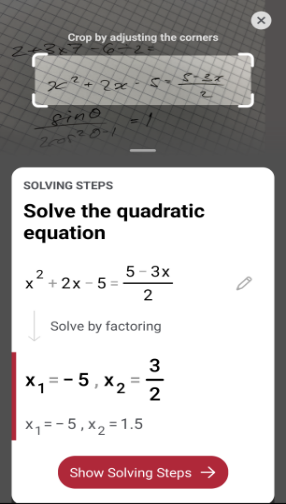
The GUI is simple; the view of the camera as the background with text prompts at the top of the screen and an adjustable white box to target the desired question. The straightforward design with minimal icons along with text prompts on the top allowed users to know what to do immediately; adjusting the size of the box as needed and capturing the image with the bright red button. The ability to adjust the white box and crop the image was particularly helpful in aligning the equations after capturing an image, along with the ability to cancel the action if an image is unclear or does not capture the entire question, reducing time wasted processing the image. Additional functionality such as allowing users to access their history using the icon in the bottom left and import images from their camera roll have proven to be very useful during testing as it allowed me to use blurry images such as the ones above.

The calculator displays the recognized equation and the type of operation performed, in this case, “Solve the quadratic equation”, along with other features such as the graphical calculator below which plots the equation if applicable. The little pen icon on the right side of the recognized equation allows users to modify the input if there has been a recognition error. The solution is then clearly written below, with a red box allowing users access to the steps used to acquire the result.

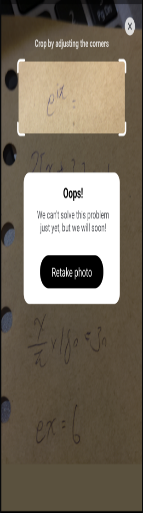
▲*Step-by-step display*



▲*Recognition of equation and calculated solution*



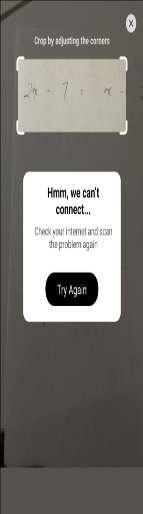
The error messages are large and clear, the prompt and proposed solution is user friendly and easy to understand:



▲*Unsupported equation error*



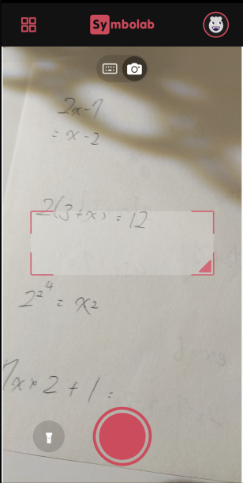
▲*Unrecognized problem error*



▲*Offline error*

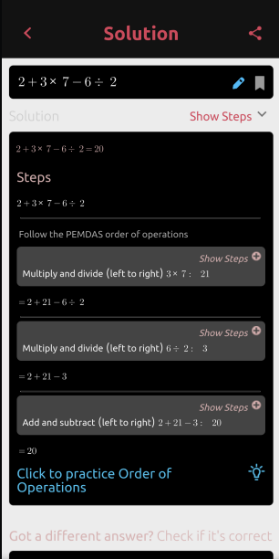
### Product 2 – Symbolab

*Symbolab* is well-known for its powerful math engine and online calculators, which let you execute a variety of mathematical operations. Their smartphone app combines those into one and adds new features such as the camera input. This allowed *Symbolab* to perform more complex operations than the other calculators tested, such as imaginary numbers and limits, and was quite difficult to find exceptions or trigger error messages. It was also able to recognize equations composed of multiple lines such as simultaneous equations by registering the two lines as two separate equations, which I suspect was solved through implementing matrix calculations.

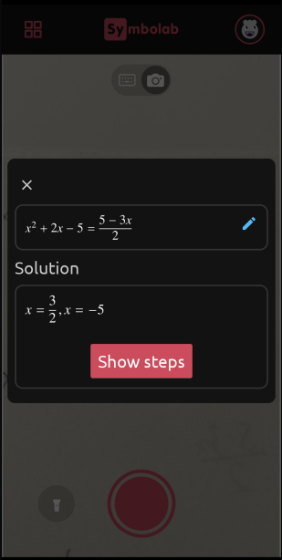


▲*Camera mode GUI*

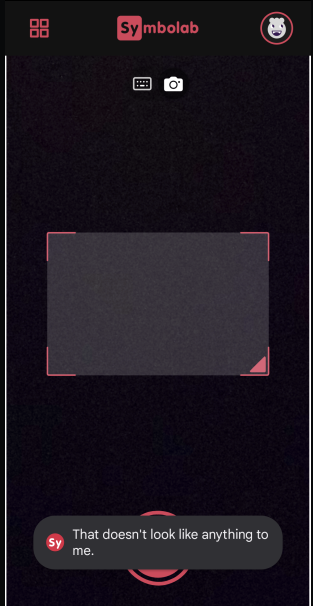
Similar to *Photomath*, *Symbolab*'s user interface features a translucent background with a target red box in the middle for alignment. It does not, however, offer access to previously typed equations or the ability to directly import photographs, unlike *Photomath*. Through interviewing maths students, it seemed that the ability to adjust the size and position of the red target box was not as clear, as many seemed to have struggled to align the box to the desired equations. It is also worth noting that *Symbolab* does not show the captured image and immediately starts to translate the targeted equation, leading to error messages if not targeted properly. After recognition, the recognized input is displayed along with the solution below it, and the option to display the steps taken. It allows the user to edit the inputted expression through the pen icon on the right of the expression, however Even after pressing capture the camera still runs in the background, which may cause distraction and take up display space that may have been used to show more information about the inputted equation. The error messages are small and often have a demanding tone and do not provide a clear solution to the errors shown, not user-friendly.



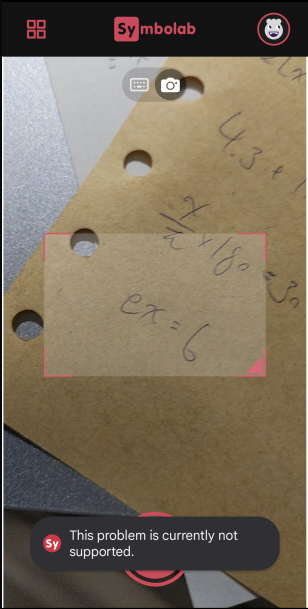
▲*Step-by-step display*



▲*Recognition of equation and calculated solution*

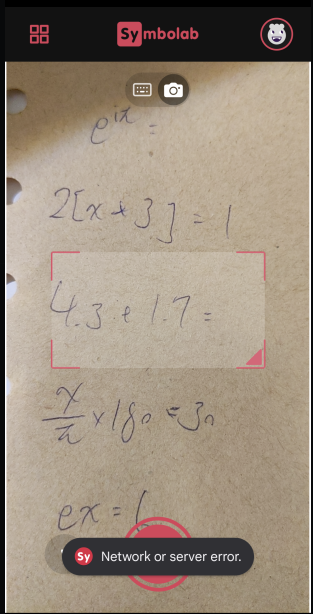


▲*Unrecognized problem error*



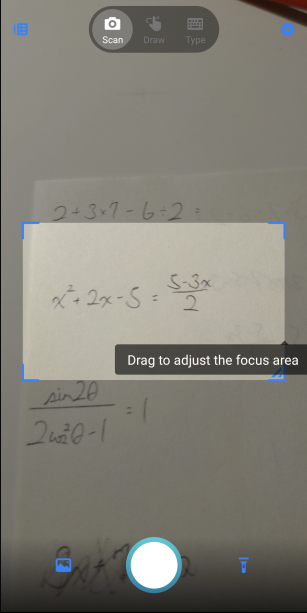
▲*Unsupported equation error*

▲*Offline error*



### *A picture containing diagram Description automatically generated*Product 3 – Math

*Math* by Microsoft is the middle ground between the two previous calculators, with a GUI and the ability to solve a variety of mathematical equations similar to that of *Symbolab*, with functionality such as importing images and history review like *Photomath*. After scanning the equations, the app was able to recognize all equations and even those the previous products failed to process. However, I did find slight errors in recognizing decimal points, which the previous two did not exhibit the same problem. Although this could be easily fixed with the editing tool accessible through the pen icon.



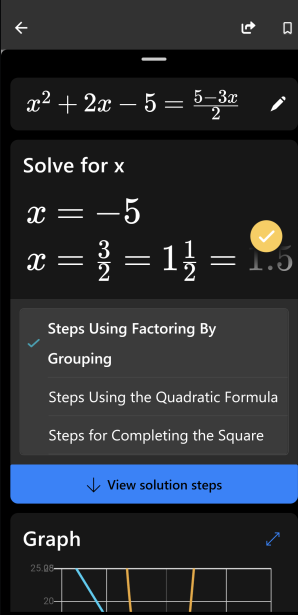
▲*Camera mode GUI*

Upon first use, the app shows text prompts showing how to navigate the interface, along with example pictures to demonstrate the functionality of the software. With a similar “capture and crop” system, *Math* prompts the user to focus on specific questions on the captured image using the blue box, sharing the advantages of *Photomath.*

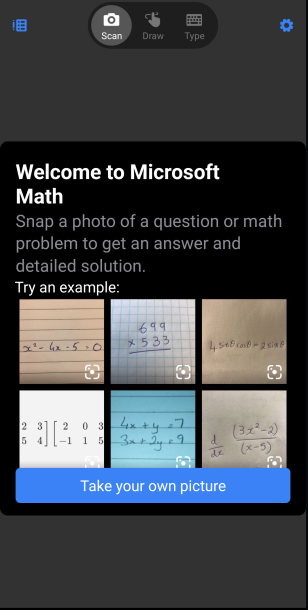
▲*Photo import GUI*

*A picture containing text

Description automatically generated*



▲*Step-by-step display*



▲*Example pictures for demonstration*

▲*Recognition of equation and calculated solution*

## Proposed Solution and Limitations

My solution will be a phone application software that uses the phone camera to take pictures of handwritten mathematical expressions, which a neural network then uses to recognize the expressions and carry out the desired operations. The application will be aimed at those who have just started learning algebra or simply needs a faster way to enter expressions into the calculator. The challenge in this project will be the recognition of handwritten mathematical expressions, for which I will be implementing a convolutional neural network to perform the task of classifying the different symbols and numbers to be inputted into the calculator. From gathering information from existing software, I have targeted some key functions and limitations of my project:

1. The application will be able to be used offline to allow users to access it anywhere
2. The initial stages of development will be aimed toward solving linear expressions and expressions; however, the implementation of trigonometric identities and quadratics will be investigated.
3. The target box will be adjustable in size, cropping the desired expression after capturing the image.

Before listing more detailed objectives, further research will be done to gain a more detailed understanding of the methods used.

## Further Research

As aforementioned, this project will heavily rely on the ability of the software to recognize handwritten digits and mathematical equations, which I will be implementing convolutional neural networks to perform optical character recognition. To research further into the topic, I have referenced “Neural Networks” by Michael Nelson which describes a similar problem in recognizing handwritten digits through training a neural network using the MNIST dataset, as well as SciML’s GitHub repository on CNNs. All resources referenced will be cited in the References section.

### Modelling

Here I will lay out the basic flow of the application to provide a basis for the objectives which need to be set and will be further investigated in the Design section of the report.

Diagram

Description automatically generated

### Image Processing

Before the classification of numbers and symbols can be performed, the image needs to be preprocessed to allow uniform inputs into the neural network. The image will be denoised and reshaped using OpenCV, and segmented into its individual symbols. During this process, the segmented parts will be labelled to allow for reconstruction of the data after recognition, in which the order may be stored as an attribute to the symbols or numbers.

### Neural Networks

For ease of designing the neural network and to maximize time in optimization, I will be converting the images obtained into NumPy arrays, as NumPy arrays use less memory, along with the ease of performing matrix transformations and other mathematical operations, all of which will be immensely useful for the development of this project. I have chosen to use Convolutional Neural Networks to perform optical character recognition due to their efficiency in image recognition as they require less preprocessing of the input images and the abundance of resources available. One of the biggest advantages of Convolutional neural networks is the ability to capture spatial dependencies; where a pixel’s value is influenced by the value of nearby pixels; by applying “filters” to the image.

Diagram

Description automatically generated

▲*An example of a CNN architecture used to classify clothing. I intend to implement a similar architecture.*

These filters, often referred to as "kernels," are typically 3 x 3 or 5 x 5 matrices that are multiplied by the image's pixel values to identify desired characteristics by sampling the same amount of pixels as their size (aka 3 x 3 kernels sample pixels in a 3 x 3 region); repeating the process across the image with a fixed “stride”, which can be thought of the distance between each sampled region. The values within the kernels are the weights which may be optimized through training where machine learning algorithms such as backpropagation may be used. The application of the kernels will be done through layers, each layer in the CNN having a different purpose. In the typical CNN, the architecture consists of a combination of convolutional layers to filter the image, a max-pooling layer to standardize and group convoluted features for ease of computing, and a fully connected layer which is responsible for classifying the collected features into their respective classes.

Libraries such as TensorFlow will be used for the training and development of the neural network, whereas training data will be sourced from Kaggle, which will be cited in the References section.

### Equation solving

After the recognition of numbers and symbols, the next step would be representing them in a malleable manner for it to be solved by translating them to Python’s syntax such as “÷” to “/” for example. This may be performed by setting the output parameters of the neural network to be already in Python syntax, however, this may require modification of the training data as the labels on the data may differ. We can also employ libraries to perform the translation, by setting the recognized symbols as the key and their python equivalents as the element. To reconstruct the inputted expression, the order will be arranged into order by referencing the “elements order array” defined during preprocessing.

## Objectives

1. Allow image capturing through the app’s user interface.
   1. The app will be able to request access to the phone’s camera.
   2. The captured image will be able to be collected by the software for processing.
   3. The captured image will be able to be converted into a malleable format (NumPy array or otherwise).
      1. The converted data will be able to be stored in the system for later use.
2. The app must have a graphical user interface, allowing the targeting of required math expressions.
   1. A target box will be presented to the user with the ability to be adjusted by dragging the corners, however, the centre of the box will be fixed in place to avoid awkward positioning of the target box if moved.
   2. There will be a text prompt near the target box explaining how to use the target box and how to adjust it.
   3. The target box will allow for cropping after the capture of the image.
   4. There will be options to allow users to retake the image if unsatisfied.
3. The app will be able to segment the individual symbols with sufficient accuracy for them to be distinguished.
   1. The app will be able to de-noise and filter the image for easier processing during recognition
   2. The app will be able to store the order of the segmented elements in the form of an attribute of the segmented image or otherwise to allow for later access.
4. The neural network must be able to apply changes to the converted data (as mentioned in 1.3) to allow for filters to be applicable.
   1. Matrix multiplication of kernels and the respective pixel values on the image must be correct.
      1. Kernel parameters including size and stride must be defined and changeable.
         1. The Kernel will be defined as a function with size and stride as parameters.
      2. The values within the kernel will be able to be modified during the training process through forms of backwards propagation.
      3. The amount of padding applied through each layer must be definable
         1. The padding process is to be defined as a function with the level of padding as a parameter.
5. The layers within the CNN must be able to connect with each other.
   1. Data must be able to be passed to the next layer while maintaining the integrity of the original data. There should be no unwanted changes to the ordering of the values.
6. Activation functions of the layers must be correctly implemented.
   1. Activation functions must give correct values when tested individually.
7. Output of the classification must be stored in some form in relation to the original ordering it was in before segmentation (as a reference to 3.2)
8. If insufficient evidence is gathered to deduce the equation entered, an error message will be displayed telling the user, providing the option to reactivate the camera to take another input.
9. The identified numbers and symbols must be able to be translated into a usable form (python syntax)
   1. Relationships between identified numbers and symbols and their respective mathematical notation in python must be defined.
10. The translated form of the full equation must have the same ordering or be mathematically equivalent to the original equation.
11. The translated form of the equation will be presented to the user and allow for feedback on the accuracy of the recognition
    1. It will allow users to edit the outputted equation and coefficients.
       1. Implementation of a textbox or other form of text input.
12. The equation solver will be able to correctly output the calculated solution
    1. If the equation has a variable, the solution should have “x=A”, where A is the answer.
    2. If the equation is an expression, only the value should be outputted.
13. The output should be able to be closed and the camera will reactivate to allow for input from the user, the process repeats.
    1. An implementation of a flag to check if the user is using the output box.

# 

# Design

## Hiearchy Diagram

## Class Diagram

## Prototypes

### Image Segmentation

### UI design

# References

- <http://neuralnetworksanddeeplearning.com>

- <https://github.com/stfc-sciml/sciml-workshop>

-<https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7>