FINAL REPORT: DIGITAL IMAGE RECOGNITION OF HANDWRITTEN MATHEMATICAL EXPRESSIONS

Maeesha Biswas
Student# 1003931392
maeesha.biswas@mail.utoronto.ca

Muhammad Ahmed
Student# 1007364825
mm.ahmed@mail.utoronto.ca

Tracy Sun
Student# 1006981438
tracy.sun@mail.utoronto.ca

1 INTRODUCTION

Mathematics is relevant to every industry. As the world is becoming more technology-driven, it has become essential to automate the digitization of handwritten mathematical expressions to reduce the manual and redundant nature of this work.

The main goal of our project is to create a model that can take handwritten mathematical equations from an image upload or canvas and convert the equation to LaTeXformat. The model takes the input and goes through a symbol segmentation process using OpenCV in Python. Then it uses a pre-trained model, which consists of architecture similar to Convolutional Neural Networks (CNN), and an Artificial Neural Network (ANN) classifier used for flattening. This allows the model to carry out feature extraction. Once each character of the equation is identified, the model recombines and outputs the equation in LaTeXformat. The CNN model has a 93% testing accuracy. To demonstrate this model, a web application was built using the Python Flask web framework.

As the application of mathematics increases, it is important to expand its readability. Our model will allow the digitization of handwritten mathematics. This introduced use cases such as allowing students to export handwritten notes into neater and searchable digital notebooks, assisting the visually impaired with reading handwritten math (Shinde et al., 2017), and extracting information from old mathematical texts.

Deep Learning (DL) allows the programmer to create an ANN. This ANN can be used to learn from data that has multiple layers of abstraction (LeCun et al., 2015). Computer Vision (CV) is hard-coded, while DL is not; Therefore, DL can consider different writing styles and extract certain patterns through digital image processing (O'Mahony et al., 2019). It also results in more accuracy in image classification and semantic segmentation (O'Mahony et al., 2019). Our model required character classification from a dataset containing varying handwritten mathematical equations. It needed to learn and predict regardless of the abstraction from the data. This is why DL was a reasonable approach for our model.

2 PROJECT ILLUSTRATION

The project illustration can be seen in Figure 1 on the next page.

3 BACKGROUND & RELATED WORK

Several existing applications can convert handwritten text to LaTeX. For example, detexify Kirsch & Radford (2020) can recognize individual symbols drawn on a digital canvas and show the LaTeXrepresentation of the 5 most relevant results. Also, Mathpix Mathpix (2022) is capable of converting images to various digital formats including LaTeXand Docx.

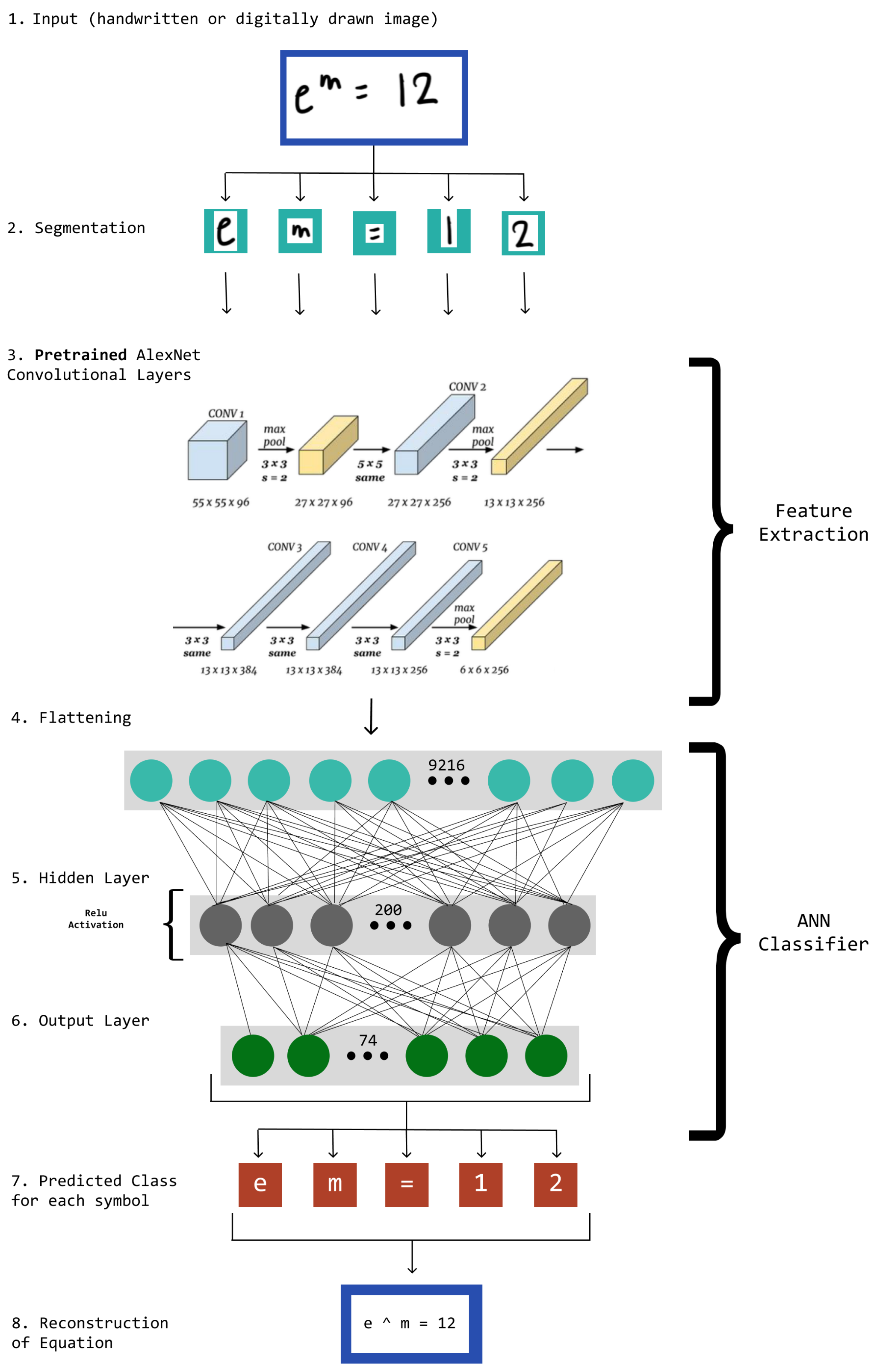


Figure 1: Illustration of Digital Image Recognition of Mathematical Expressions CNN Project

Broadly, the recognition of handwritten text is generally achieved through using neural network-based systems, and most existing approaches are based on convolutional neural networks (CNNs) and recurrent neural networks (RNNs) (Li et al., 2021). RNNs are typically used to get high-performance results and have won several handwriting recognition competitions (Voigtlaender et al., 2016). Conversely, the training time when using RNNs is normally long due to its recurrence feature (Doetsch et al., 2014). On the other hand, CNNs are efficient and reliable in image understanding, but their performance is slightly inferior to RNN-based systems (Chaudhary & Bali, 2022). However, since CNNs have high efficiency and relatively good performance on mathematical symbols and digits, they are commonly studied for tasks about handwritten math classification (Ramadhan et al., 2016).

Several studies were conducted to improve the CNN models for better performance. In 2019, Yousef et al. proposed a fully convolutional CNN architecture for text recognition (Yousef et al., 2020). This architecture makes extensive use of batch normalization and layer normalization, which allows it to achieve a high degree of regularization and effectiveness. This model has accomplished human-level performance on CAPTCHA, and won the International Conference on Frontiers of Handwriting Recognition (ICFHR) 2018 Competition on Automated Text Recognition.

In 2022, Chaudhary et al. designed an advanced version of the Efficient And Scalable TExt Recognizer (EASTER) which they published in 2020 (Chaudhary & Bali, 2020). The EASTER2.0 is a novel CNN-based architecture, and uses only a one-dimensional convolution (Chaudhary & Bali, 2022). They also proposed a new data augmentation technique called "Tiling and Corruption (T ACo)". T ACo augmentations refer to generating a new image by randomly covering up several areas on the input image, thus producing a larger training set. This augmentation makes EASTER2.0 capable of achieving high accuracy with limited training data.

There are also numerous studies on how to implement CNN specifically for better recognition of mathematical expression. In 2018, D'souza et al. proposed an offline handwritten mathematical expression recognizer (D'souza & Mascarenhas, 2018). Their model first obtains the grayscale version of input images, then the model is trained with SpNet architecture. After 100 epochs with a learning rate of 1, this model can accurately recognize 83 types of mathematical symbols (Ramadhan et al., 2016).

In 2019, Marques et al. introduced a model that can recognize handwritten polynomials (Marques et al., 2019). This model used Fractional Order Darwinian Particle Swarm Optimization (FODPSO) to separate components in the input image, then three CNN layers are used to identify each component individually in order. The first CNN layer classifies if that element is a number or symbol. Then, there is a CNN layer dedicated to number recognition, and a CNN layer for symbol recognition. The data is passed to the corresponding CNN layer after classification. After recognizing all the components on the input image, the results are merged together to produce the final output. The model achieves an average accuracy of 99% in recognizing polynomials with various handwriting styles (Marques et al., 2019).

In 2021, Sultan et al. developed a handwritten mathematical symbol classifier that can greatly improve the recognition of handwritten mathematical formulas (Sultan et al., 2021). Their model has a novel CNN architecture. It has 6 convolutional layers, after every two layers, there is a max pooling layer. This model uses different adaptive learning rates for each model parameter, thus achieving a high overall accuracy (Jepkoech et al., 2021). The average accuracy of the model is 99.19%, which is higher than most handwritten mathematical symbol classifiers (Sultan et al., 2021).

4 Data Processing

4.1 Data Sources

To train our model we used an open source math symbol dataset available on Kaggle (Nano, 2016). The dataset contains individual symbols extracted from the handwritten equations provided by CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions.

4.2 Data Characteristics

The original dataset consists of over 370,000 images divided into various classes. We chose 74 classes from the dataset, including mathematical digits, the alphabet, basic operators, trigonometric functions, and some greek letters. Each image in the dataset contains one symbol written in black ink on a white background and saved as a jpeg. Figure 2 shows some sample images from the dataset with their respective labels.

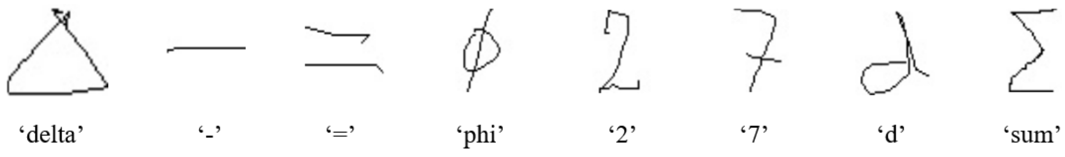


Figure 2: Sample handwritten symbols from the dataset

4.3 Data Augmentation

The images were unequally distributed among classes in the original dataset, as seen in the left-hand side bar chart in Figure 3. For example, there were about 34,000 images for '-' class but only 47 images for 'in' class. To equalize the number of images per class, we randomly deleted extra images and augmented some classes to have 800 images per class and 59,200 total images (see Figure 3). Four data augmentation techniques were used: horizontal/vertical flip, rotation, salt and pepper noise, and white padding (Gandhi, 2021). Figure 4 shows an example of each technique.

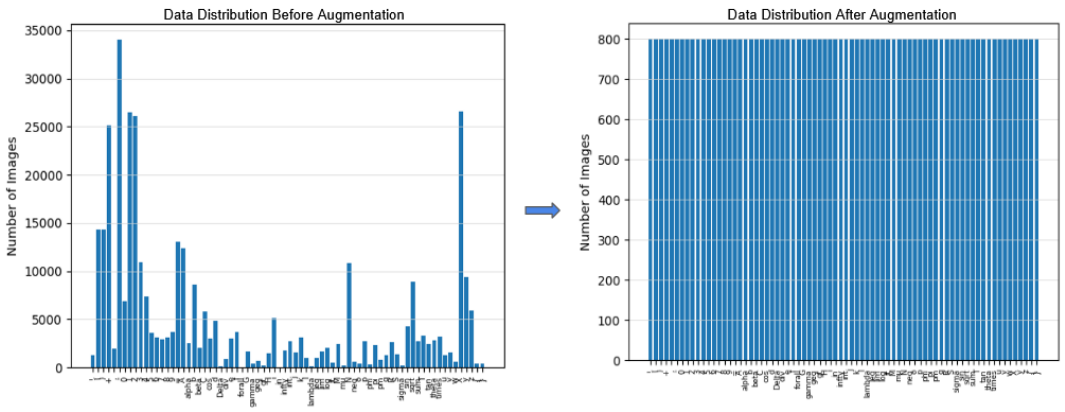


Figure 3: Data distribution before and after data augmentation

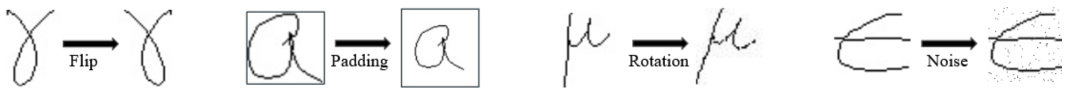


Figure 4: Different data augmentation techniques used

4.4 Data Pre-processing

The data had to be processed to be compatible with AlexNet's convolutional layers for feature extraction. For this, the images were converted from one-channel jpg to three-channel PNG files and were expanded to square size by adding white padding to the shorter dimension. Finally, all images were resized to 224×224 pixels, and following convention for large datasets, the dataset was split into training, validation, and testing data in a 60/20/20 ratio (Lee, 2018).

5 Architecture

5.1 CNN Model

We used a CNN model to classify the symbols. As Figure 1 shows, our model consists of pre-trained convolutional layers of AlexNet and an ANN classifier. The convolutional layers are used to extract features which are then used to train the ANN classifier. The ANN classifier consists of an input layer with 9216 neurons, one hidden layer with 200 neurons, ReLU activation function, and an output layer with 74 neurons: a total of 1,858,000 parameters to train.

5.2 Segmentation and Reconstruction

Segmentation and reconstruction consist of steps 2 and 8 of our multi-step classifier (Figure 1).

The OpenCV library is used to segment the digits in the equation. The original image is read as grayscale and converted to binary with inverted colour. It then passes through the `cv2.findContours()` function. Contours with similar positions and widths are combined, since they were considered to belong to the same segment. Each segment is then cropped from the original image and passed to the CNN model. The results from our CNN model are reconstructed by considering the positional relationship between segments and the syntax of the $\LaTeX$.

6 Baseline Model

A random forest classifier from the scikit-learn Python library is used as the baseline model. For this model, all image data are read as grayscale. They are then converted to one-dimensional arrays and split to test, validation, and test using a 60:20:20 split (Lee, 2018). The trained model has 100 decision trees, with an average depth of 72. The test accuracy of this random forest classifier is 74.97%. Among all data classes, the "x" class has the lowest accuracy of 26%. The team concluded this because of the similarity in strokes between the x and multiplication sign.

7 Quantitative Results

7.1 CNN Model

Our final CNN model achieved a testing accuracy of 0.934 on 11,840 test images and had a final training accuracy of 1.00 and validation accuracy of 0.935. Figure 5 shows the learning curves of our model.

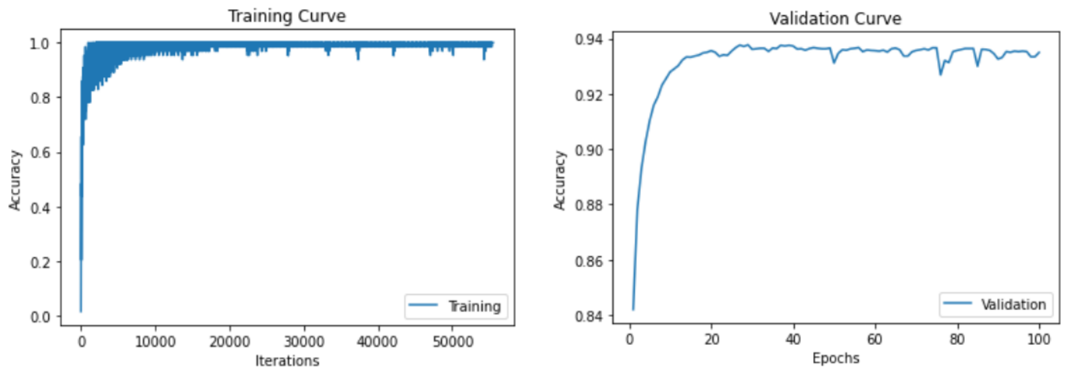


Figure 5: Learning curves of the final CNN model

8 Qualitative Results

Despite having a high accuracy overall, the CNN model struggles to classify some classes properly because of similarities in stroke between them. Class '0' has the lowest recall of 0.65 on the testing data, with 29% of the images being misclassified as the class 'o' because of similarity in shape and strokes. Similar problems were found with the 'X' and 'times' classes, 'z' and '2' classes, 'g' and '9' classes, and the '1' and ',' classes. Figure 6 shows examples of the model wrongly classifying some symbols. It can be noted that this misclassification of most of the images in the figure could reasonably be done by humans as well.

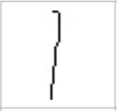
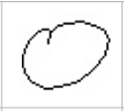
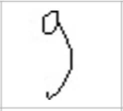
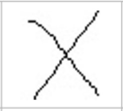
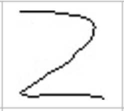
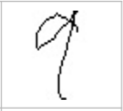
						
True Label	,	0	g	x	z	9
Predicted	1	o	9	times	2	q

Figure 6: Some examples of images misclassified by our CNN model

9 Evaluate model on new data

9.1 CNN Model

The team asked classmates to write 30 symbols for each of the 74 classes (totalling 2220 symbols). We received this data in PNG files, and all symbols were written using digital styluses and tablets. To coordinate with the model, all symbols are written with a stroke thickness of 1 pixel. Different fonts are used to ensure there are variances in the data.

All 2220 images were passed through the trained model and obtained an accuracy of 75.67%. This is slightly higher than the baseline model, thus the performance of the model is an expected improvement. According to the confusion matrix in Figure 7, class "alpha" has the lowest recall of 10%, with 60% of "alpha" being identified as "a". On the other hand, symbols commonly used in equations all have a fair recall. Most numbers have a recall of over 90%, and the frequently used Greek letters such as π and $\sum$ have a recall of nearly 100%. Also, symbols with multiple characters, such as $\sin$ and $\lim$, mostly have a recall of 100%.

9.2 Multi-Stage Classifier With CNN Model

To further test the performance of our model, the team collected 147 images of handwritten math equations and expressions to test with the multi-stage classifier. 104 of them are written by the team using various fonts and thickness of stroke, and the other 43 were randomly selected from CROHME open dataset which required independent data processing.

To produce a small dataset of full equations with ground truths, we followed these steps and generated a Python Script with helper functions (here):

1. Convert all files from `inkml` to `PNG`
2. Expand `PNG` files to squares by adding white padding to the shorter side.
3. Make the dimensions of the image 224px x 224px.
4. Randomly choose 43 images, and manually assign ground truths in and record in a `CSV` file.

The classifier achieves an accuracy of 66.02%. Among all incorrect results, 65.7% are only due to wrongly identified symbols, as seen in Figure 8. Another 20% are due to false reconstruction based on wrongly identified symbols. For example, an inaccurate syntax is used in sinusoidal function because the function's name is not identified correctly, as seen in Figure 9.

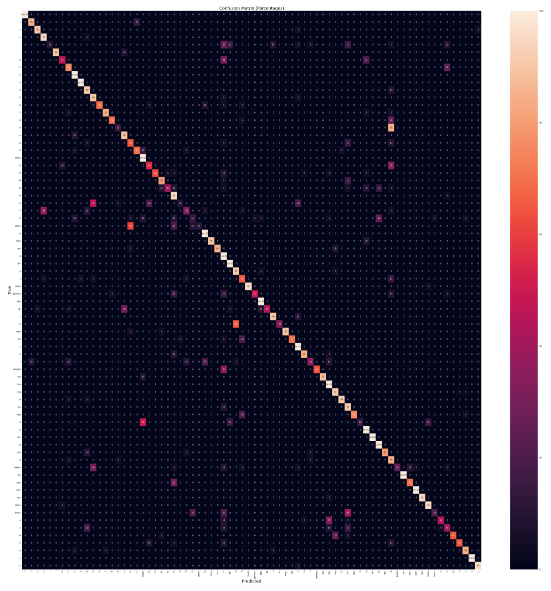
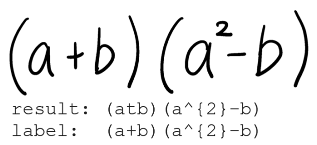
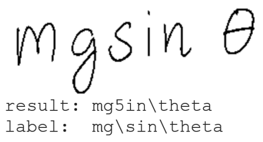


Figure 7: Confusion Matrix of CNN Model with Net New Data



```
result:  (atb)(a^{2}-b)
label:   (a+b)(a^{2}-b)
```

Figure 8: Incorrect result due to wrongly identified symbols



```
result: mg5in\theta
label:  mg\sin\theta
```

Figure 9: Incorrect result due to false reconstruction

10 Discussion

There is a significant drop in accuracy for the multi-stage classifier compared to the CNN model. The group concluded several reasons for the decrease in accuracy. Most importantly, the classifier uses a multi-stage pipeline, including segmentation, classification, and reconstruction, thus failure in any step will lead to an incorrect result. Also, the classifier would pass segments with a random thickness of strokes to the model (see Figure 10). Since the CNN model has trained with symbols with a stroke thickness of 1 pixel, it is less able to utilize the geometry information on segments from the classifier.

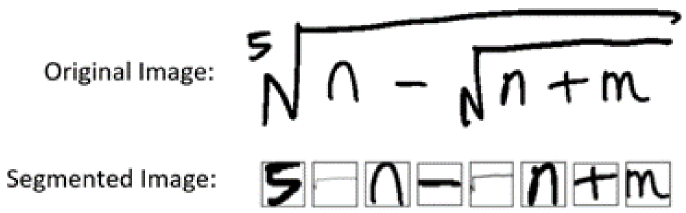


Figure 10: Image before and after segmentation

Generally speaking, the performance of our model is acceptable. It can correctly identify most symbols that appear in manifold equations, especially for ones with a distinctive shape, such as $\div$ and $\lim$. Nonetheless, it does make mistakes when classifying symbols with similar strokes, particularly a and α, and e and $\in$. This type of mistake occurs because the CNN only uses the geometry information on a single symbol. However, the correct class is generally clear if contextual equation data and nearby symbols are considered. Thus, this issue should be improved if we can implement a hybrid neural network with both CNN and RNN, since RNN allows the model to be mindful of the context in the whole equation.

11 Ethical Considerations

Two of the major ethical considerations of our model are intellectual property theft and unauthorized gain of private/confidential data. Using our model, users may be able to convert notes or documents into a digital format. This introduces the ability to digitize another person's work and sell it for profit without permission. This also introduces the ability to store handwriting data without authority. Handwriting is unique (Srihari et al., 2002), and this stored data could be used as a key identifying factor to link back to the person. Privacy during data collection could be a concern with biometric improvements in handwriting styles (Faundez-Zanuy et al., 2020).

12 Project Difficulty / Quality

This project introduces further complexity beyond a basic digit recognizer in the following ways:

- More classes. The MNIST dataset has 10 classes, and CROHME has a representative 101 symbols, of which we use 74.
- Implementation of a Multi-Step Classifier with segmentation and recombination algorithms before and after the CNN classifier to accept and convert multi-digit equations as the input.
- A higher number of classes required extensive data augmentation (horizontal/vertical flip, rotation, salt and pepper noise, white padding) decision making for each class which increased data processing complexity.
- Creation of a Web App for the demonstration facilitates the input of net new data more easily. This requires knowledge and requirements gathering for the CNN model's operating environment.

Our team's self-assessment of the project quality is as follows:

- The multi-stage classifier has an accuracy of 66% because there are 3 possible sources of error - the CNN Classifier (93% accuracy), Segmentation algorithm, and Reconstruction Algorithm in a single pipeline. As an input passed through each step, errors are propagated and have an increased chance of happening at any one step. This structure is a limitation we acknowledge and would address in future work (below).
- The individual symbol classifier has a 93% accuracy which means it was successfully able to distinguish between over 70 different symbols and is extensible to more classes.

Our team would make the following improvements in future iterations of the CNN model and web app:

- Trying an autoencoder and decoder to detect variant stroke thicknesses, different colours, and images with cropped symbols (as pictured here) (Li et al., 2021).
- Improve our Reconstruction Algorithm to support expressions that read vertically like fractions. (LArchCS, 2018)
- Reduce the need for manual ground truth labelling of our full equation dataset by trying semi-supervised learning (Ball & Srihari, 2009).
- Increase our symbol classes to include all greek letters, 101 original CROHME represented symbols, and a broader range of mathematical relations such as **munderover** or **munder** (vertically written expressions) (Mouchère, 2014).

References

Gregory R. Ball and Sargur N. Srihari. Semi-supervised learning for handwriting recognition. In *2009 10th International Conference on Document Analysis and Recognition*, pp. 26–30, 2009. doi: 10.1109/ICDAR.2009.249.

Kartik Chaudhary and Raghav Bali. Easter: Efficient and scalable text recognizer. *arXiv preprint arXiv : 2008.07839*, 2020.

Kartik Chaudhary and Raghav Bali. Easter2. 0: Improving convolutional models for handwritten text recognition. *arXiv preprint arXiv : 2205.14879*, 2022.

Patrick Doetsch, Michal Kozielski, and Hermann Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pp. 279–284, 2014. doi: 10.1109/ICFHR.2014.54.

Lyzandra D'souza and Maruska Mascarenhas. Offline handwritten mathematical expression recognition using convolutional neural network. In *2018 International Conference on Information , Communication, Engineering and Technology (ICICET)*, pp. 1–3, 2018. doi: 10.1109/ICICET.2018.8533789.

Marcos Faundez-Zanuy, Julian Fierrez, Miguel A. Ferrer andMoises Diaz, Ruben Tolosana, and Réjean Plamondon. Handwriting biometrics: Applications and future trends in e-security and e-health. *Cognitive Computation*, 12, 2020. doi: https://doi.org/10.1007/s12559-020-09755-z.

Arun Gandhi. Data augmentation — how to use deep learning when you have limited data—part 2, 2021. URL `https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/`.

Jennifer Jepkoech, David Muchangi Mugo, Benson K Kenduiywo, and Edna Chebet Too. The effect of adaptive learning rate on the accuracy of neural networks. *International Journal of Advanced Computer Science and Applications*, 12(8), 2021.

Daniel Kirsch and Jim Radford. Detexify, 2020. URL `https://detexify.kirelabs.org/classify.html`.

LArchCS. Handwritten equation recognition tensorflow, 2018. URL `https://github.com/LArchCS/Handwritten-Equation-Recognition-Tensorflow`.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Chang Hsin Lee. Improving the validation and test split, 2018. URL `https://changhsinlee.com/better-validation-test/`.

Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. Trocr: Transformer - based optical character recognition with pre - trained models. *arXiv preprint arXiv : 2109.10282*, 2021.

F.C.F. Marques, T.P. De Araujo, C.C. Nator, A.A. Saraiva, J.V.M. Sousa, A.M. Pinto, and R.T. Melo. Recognition of simple handwritten polynomials using segmentation with fractional calculus and convolutional neural networks. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 245–250, 2019. doi: 10.1109/BRACIS.2019.00051.

Inc. Mathpix. Digital science, instantly, 2022. URL `https://mathpix.com/`.

Harold Mouchère. Oct 2014. URL `http://www.iaprtc11.org/mediawiki/index.php/CROHME:_Competition_on_Recognition_of_Online_Handwritten_Mathematical_Expressions`.

Xai Nano. Handwritten math symbols dataset, 2016. URL `https://www.kaggle.com/datasets/xainano/handwrittenmathsymbols?resource=download`.

Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs.traditional computer vision. In *Science and information conference*, pp. 128–144. Springer, 2019.

Irwansyah Ramadhan, Bedy Purnama, and Said Al Faraby. Convolutional neural networks applied to handwritten mathematical symbols classification. In *2016 4th International Conference on Information and Communication Technology (ICoICT)*, pp. 1–4, 2016. doi: 10.1109/ICoICT.2016.7571941.

Sagar Shinde, RB Waghulade, and DS Bormane. A new neural network based algorithm for identifying handwritten mathematical equations. In *2017 International Conference on Trends in Electronics and Informatics(ICEI)*, pp. 204–209. IEEE, 2017.

Sargur N Srihari, Sung Hyuk Cha, Hina Arora, and Sangjik Lee. Individuality of handwriting. *Journal of forensic sciences*, 47(4):856–872, 2002.

Rafi Ibn Sultan, Md. Nahid Hasan, and Mohammad Kasedullah. Recognition of basic handwritten math symbols using convolutional neural network with data augmentation. In *2021 5th International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, pp. 1–6, 2021. doi: 10.1109/ICEEICT53905.2021.9667794.

Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. Handwriting recognition with large multidimensional long short - term memory recurrent neural networks. In *2016 15th International Conference on Frontiers in Handwriting Recognition(ICFHR)*, pp. 228 – 233, 2016. doi: 10.1109/ICFHR.2016.0052.

Mohamed Yousef, Khaled F Hussain, and Usama S Mohammed. Accurate, data - efficient, unconstrained text recognition with convolutional neural networks. *Pattern Recognition*, 108:107482, 2020.