ML Final Project

Chiang, Pei-Lin
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
jiangdumpling@gmail.com

Lee, Hua-Wei
Department of Interdisciplinary Program of Science
National Tsing Hua University
Hsinchu, Taiwan
lihuaway20040116@gmail.com

Diau, I-Ping
Department of Interdisciplinary Program of Science
National Tsing Hua University
Hsinchu, Taiwan
iping000@gmail.com

Lin, Wei-Sheng
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
a20040108jason@gmail.com

Chen, Ting-Jyun
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
alexchen33d@gmail.com

***Abstract*— Most of the current calculator software relies on button or keyboard inputs, which can be inefficient for handwritten data. This project focuses on developing a system to recognize handwritten numbers and mathematical symbols. Users can write directly on a digital writing pad to create an image containing handwritten elements, such as numbers and symbols. The system processes the input image, extracts individual components, and recognizes each number or symbol.[1]**

Keywords: handwritten recognition, mathematical symbols, datasets, image processing

I. Introduction

The recognition of handwritten input has seen significant advancements, particularly in the domain of individual numbers. Modern machine learning algorithms, supported by datasets such as MNIST, have made it possible to achieve highly accurate recognition of single digits. Beyond numbers, handwritten symbols, such as mathematical operators, can also be effectively recognized by training on specialized datasets tailored for these purposes. However, the complexity increases when dealing with an image that contains multiple handwritten elements, including both numbers and symbols of varying sizes, styles, and spatial arrangements. For example, consider an input image with the expression 3 + 46 ÷ 2 - 5.9 =. This single image needs to be processed to extract smaller components, such as 3, +, 4, 6, ÷, 2, -, 5, ., 9, and =. Each of these components must be segmented correctly before recognition. Unlike systems that compute the final result of such expressions, the focus of this project is solely on accurately identifying the individual handwritten numbers and mathematical symbols within the input image.

Developing such a system requires addressing several interconnected challenges. First, segmenting the input image into distinct components is not trivial, as handwritten elements can vary in size, overlap, or be closely spaced. Second, the system must be capable of recognizing a wide variety of handwritten styles, which can differ significantly from person to person and introduce variability in the shapes of numbers and symbols. Finally, the system needs to maintain high accuracy even when presented with diverse handwriting styles, image resolutions, or noise that may affect the clarity of the input.

In summary, this project aims to design and implement a robust system for recognizing handwritten numbers and symbols in a way that is both efficient and adaptable. By addressing the challenges of segmentation, recognition, and variability in handwriting, the system can serve as a foundation for applications that require accurate interpretation of handwritten mathematical expressions.

II. Background & Method

In this section, we provide an overview of the research background, detail the methodology employed along with the rationale behind its selection, and describe the architecture of our proposed model.

A. Background

Over the past decade, significant advancements in sequence recognition and computer vision models powered by deep neural networks (DNNs) have transformed the field of handwritten document processing. The proliferation of touch and pen-enabled devices, such as smartphones, tablets, and interactive panels, has expanded the scope of applications, enabling the recognition of handwritten content like mathematical expressions, diagrams, and sketches. These advancements have driven the development of more sophisticated and robust handwriting recognition systems.

Handwritten mathematical expressions (HMEs) pose unique challenges due to their two-dimensional structure and large symbol set, comprising over 1,500 symbols with many visual similarities [2][3]. Handwriting input is often preferred for mathematical content because of its intuitive nature, but achieving high recognition accuracy remains difficult. Errors in recognition can negatively affect user experience, underscoring the need for robust algorithms that can handle the structural complexity of HMEs, as well as user-friendly interfaces for efficient error correction.

Modern handwriting recognition techniques leverage both convolutional neural networks (CNNs) and recurrent neural networks (RNNs). CNNs excel at extracting spatial features from static images, making them well-suited for recognizing individual characters or symbols. However, they struggle to capture the sequential and contextual dependencies inherent in handwritten mathematical expressions. To address this, RNNs, particularly in the form of long short-term memory (LSTM) and gated recurrent unit (GRU) architectures, are employed to model temporal dependencies and sequential logic. This combination enables systems to not only identify individual components but also understand their relationships, improving overall recognition performance. Despite these advances, challenges such as handwriting variability, noise, and the need for large labeled datasets persist.

B. Methodology

In this study, we present a hybrid approach for recognizing handwritten mathematical expressions (HMEs), aiming to address the challenges of recognizing a wide range of symbols, including digits, operators, and variables, within complex expressions. Our approach is centered around a hybrid architecture combining Convolutional Neural Networks (CNNs) as an encoder for feature extraction and Recurrent Neural Networks (RNNs) as a decoder for sequential prediction. Additionally, we leverage transfer learning, teacher forcing, and attention mechanisms to further optimize the model for recognizing complex mathematical expressions. This hybrid architecture is designed to capture both spatial and sequential dependencies, enhancing the system's ability to recognize complex mathematical expressions with greater accuracy.

1) CNN Encoder

The first part of our model utilizes a Convolutional Neural Network (CNN) to process the input images and extract relevant features. CNNs are particularly effective for tasks that involve visual data because they automatically learn hierarchical feature representations, such as edges, shapes, and textures. This ability allows the CNN to identify key components of mathematical symbols, including digits, operators, and variables, even when these symbols are handwritten and may vary in style or quality.

For the feature extraction component, we used EfficientNetB0 and EfficientNetV2M, which are state-of-the-art CNN models known for their efficiency and high accuracy in image classification tasks. EfficientNet models use a compound scaling technique that balances the depth, width, and resolution of the network, providing excellent performance with fewer parameters. By leveraging transfer learning, we employed pretrained versions of these networks. Transfer learning is a technique where a model trained on a large, general-purpose dataset (such as ImageNet) is adapted to a specific task. This approach significantly reduces the time required for training, improves model accuracy, and helps the model generalize better to our dataset of handwritten mathematical expressions.

The CNN encoder processes the input image and outputs a set of feature maps, which serve as the basis for the sequential prediction performed by the RNN decoder.

2) RNN Decoder

Handwritten mathematical expressions exhibit a sequential structure that cannot be fully captured by CNNs alone. For instance, in expressions like "5 + 3 * x", the order of the symbols and their dependencies are critical for accurate recognition. To address this, we incorporated Recurrent Neural Networks (RNNs), which are designed to model sequential data by maintaining hidden states that capture information from previous time steps. This capability allows the network to effectively process sequences and predict the next symbol in the expression based on previous context.

For this task, we employed Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN) that is particularly well-suited for capturing long-term dependencies in sequential data. LSTMs are effective in retaining information across extended sequences, making them ideal for recognizing complex mathematical expressions that span multiple tokens. In addition, we also experimented with Gated Recurrent Units (GRUs), a simplified variant of LSTMs. GRUs are computationally more efficient due to their reduced number of gates, while retaining much of the performance advantages of LSTMs. To optimize training efficiency, particularly in terms of training time and memory usage, we ultimately selected GRUs as the primary decoder for our model. This decision allowed us to balance performance with computational efficiency, making GRUs a practical choice for real-time or resource-constrained environments.

3) Attention Mechanism

One of the main challenges in recognizing handwritten mathematical expressions is the presence of complex and highly variable input sequences. In some cases, certain parts of the input image are more crucial for accurate prediction.

To address this challenge, we incorporated an attention mechanism into our model. The attention mechanism enables the model to dynamically focus on the most important parts of the input sequence at each decoding step. By learning to allocate higher attention weights to relevant regions of the input, the model improves its ability to capture long-range dependencies and handle complex structures more effectively.

In the context of mathematical expression recognition, the attention mechanism not only aids the model in focusing on critical components, such as operators and variables, but also helps mitigate the risk of the model learning irrelevant or spurious features (shortcuts). Currently, the application of attention in our project serves as a validation tool, allowing us to better understand whether the model is genuinely interpreting the expression or inadvertently relying on unintended features. This mechanism plays a crucial role in ensuring that the model learns to focus on the most relevant aspects of the input, thereby enhancing the overall accuracy of the recognized expression.

4) Teacher Forcing

Training RNN-based models for sequence prediction can be challenging, especially when dealing with long sequences, as errors tend to propagate through the network. Teacher forcing is a technique that helps mitigate this issue by providing the true output from the previous time step as input to the current time step, rather than using the model’s own prediction. This allows the model to learn the correct context more quickly and reduces the risk of accumulating errors over time.

In our system, we applied teacher forcing during the training of the GRU decoder. By using the true symbols from the ground truth sequence as input during training, the model can focus on learning the correct sequence prediction without being misled by its own mistakes. This significantly accelerates the convergence of the model, enabling faster and more stable training.

5) Model Optimization

To optimize the performance of our model, several key techniques were employed. First, to address the challenge of variable-length output sequences inherent in the recognition of handwritten mathematical expressions, we padded all sequences to a fixed maximum length. This preprocessing step ensures that each output during training maintains the same size, enabling efficient batch processing and facilitating stable training when working with Recurrent Neural Networks (RNNs).

For model optimization, we utilized the Adam optimizer, a widely adopted method in deep learning. Adam is an adaptive learning rate optimizer that dynamically adjusts the learning rate for each parameter based on both the gradients and their variances. This characteristic allows Adam to stabilize the training process, accelerate convergence, and efficiently handle the complex and variable nature of handwritten mathematical expressions. The use of Adam was particularly beneficial in ensuring robust model performance, enabling the model to learn effectively from large and diverse datasets while improving both the speed of training and the accuracy of the final predictions.

C. Model Architecture

The mathematical expression recognition system is designed to take a handwritten mathematical expression image as input and output its corresponding LaTeX representation, rendered on the screen. The model architecture consists of three main components: a CNN encoder, an RNN decoder, and an optimization strategy using the Adam optimizer. The CNN encoder utilizes EfficientNet, a pre-trained model known for its efficiency in feature extraction, to capture high-level features from the input image. These features are further processed through fully connected layers to generate a compact representation of the expression. The RNN decoder is constructed using two stacked GRUs, which efficiently capture long-term dependencies in sequential data. An attention mechanism is integrated into the decoder to allow the model to focus on the most relevant parts of the input sequence, ensuring that important components such as operators and variables are prioritized during prediction. This attention mechanism helps improve recognition accuracy by allocating higher attention weights to crucial regions of the image.

To further enhance the training process, teacher forcing is employed during the training phase. Teacher forcing is a technique in which the true output from the previous time step is provided as input to the model during training, rather than relying solely on the model’s own previous predictions. This technique helps the model learn more effectively, especially in the early stages of training, by reducing error accumulation and speeding up convergence.

D. Model Deployment and Interactive Features

To further enhance the user experience, our graphical user interface (GUI), developed using CustomTkinter, incorporates several customizable features aimed at improving both usability and personalization. One such feature is the ability to adjust the interface theme between dark mode and light mode. This provides users with an optimal viewing experience, allowing them to select the mode that best suits their preferences or ambient lighting conditions. This customization ensures that the interface remains comfortable to use, even during extended sessions.

In addition to theme customization, the GUI allows users to adjust the pen and eraser sizes, offering flexibility for those who require finer control over their writing or need to make corrections on the canvas. This feature proves particularly useful for users working on detailed mathematical expressions, as they can easily modify their drawing tools to meet their specific needs. To further improve the user experience, we have also integrated a "clean" functionality, enabling users to clear the entire canvas with a single click. This eliminates the need for manually erasing each stroke, making the process of starting over quick and efficient.

Recognizing that different users may have unique workspace requirements, we have provided the option to adjust the window size, accompanied by zoom functionality. This allows users to personalize their workspace and gain better control over the user interface, especially when working with various screen sizes or viewing preferences.

In addition to these customizable features, we have implemented a powerful feature called "Guess You Want to Say" to further enhance the system’s usability. This feature is designed to predict potential outputs based on the input provided, particularly in cases where the model is uncertain about a specific character or sequence. The model initially processes the input image using a simple Convolutional Neural Network (CNN) approach, where the image is divided into smaller segments, and a feedforward pass is used to predict the character. However, due to factors like noise or ambiguity in the image, the predictions may sometimes be incorrect.

To address this challenge, we introduced the "High Two" design, which stores the top two potential predictions for each character. When the model encounters uncertainty, the "Guess You Want to Say" feature activates, presenting the top two predictions to the user. For example, if the model is unsure whether a character is a "1" or a left parenthesis "(", both options will be displayed. This allows the user to see the two most likely predictions and select the correct one.

This interactive feedback loop serves as a valuable tool for refining the model’s output. By providing alternative predictions, users can verify or adjust the system's guesses, making the tool more user-friendly and adaptable. In cases where the model struggles with specific characters or symbols, the "Guess You Want to Say" feature enhances the interaction experience by offering possible outputs. This added functionality not only improves the system's accuracy but also increases the flexibility of the model, allowing it to more effectively recognize handwritten mathematical expressions in a wide range of contexts.

III. RESULTS

The following section is to demonstrate the results of the model after actually been trained. We mainly evaluate our model implementations based on a metric called “C.E.R”, i.e. character error rate.

$$\boxed{C.E.R = \frac{(Total.Char.Errors)}{(Total.Characters)} * 100}$$

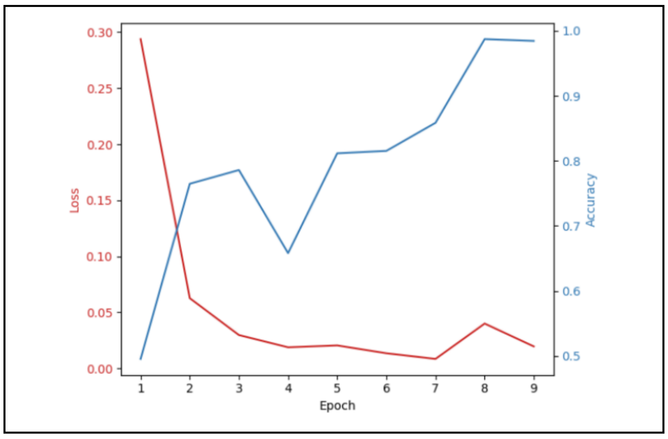
We draw error curve plot by defining y-axis to be 1 –C.E.R, the higher means the higher correctness character-wisely.

One thing we want to clarify is that the two models are trained and evaluated on different datasets, the results can’t be directly compared.

A. CNN Model

We began with a simple CNN implementation. This model is only trained by raw data that contain each character in one image.

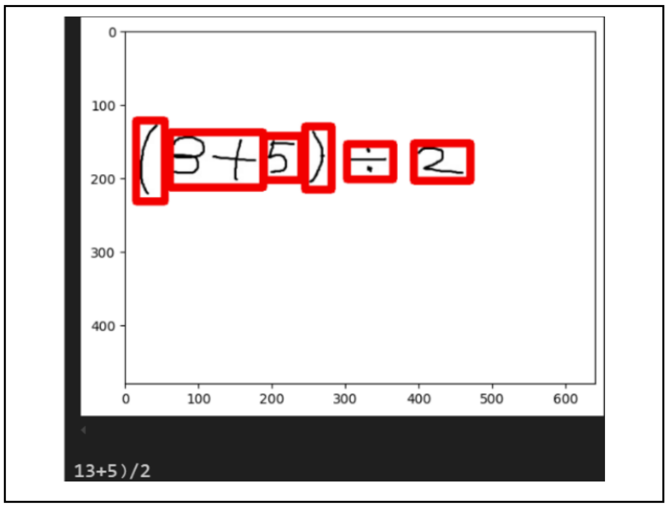
Training dataset is collected from *Handwritten math symbol and digit dataset* on Kaggle.[4]



Overall accuracy achieved 0.984 on single character testing.

Noted that, since this model is only trained by one char per image, it doesn't acquire the hidden logic of the real-world expression. So, it is not practical to apply it on real application.

For instance, the following example demonstrate its down downside.



```
13+5)/2
```

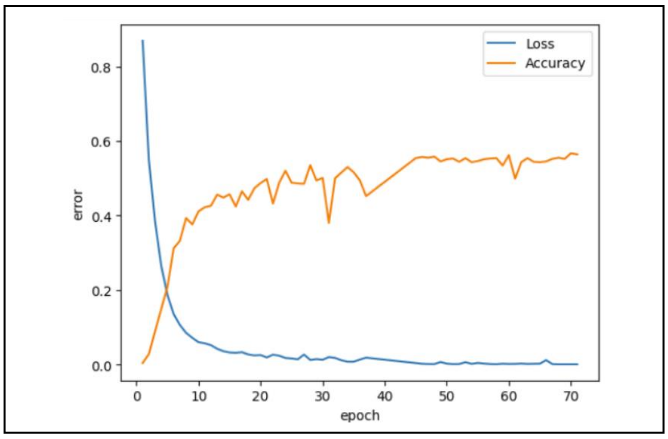
It can't suggest the first character to be a '1', based on the subsequent connection to the following right- parenthesis.

And this indicates that to fully recognize the mathematical expression in hand-written form, we have to provide semantic analysis.

B. RNN Model

This RNN applied the technique and structure that we've introduced in the former topic.

And it's trained by *CHROME16* handwritten mathematical expression dataset[5]. Which contains 8Ks of a wide variety of mathematical expressions and is capable of recognizing over 100 symbols.

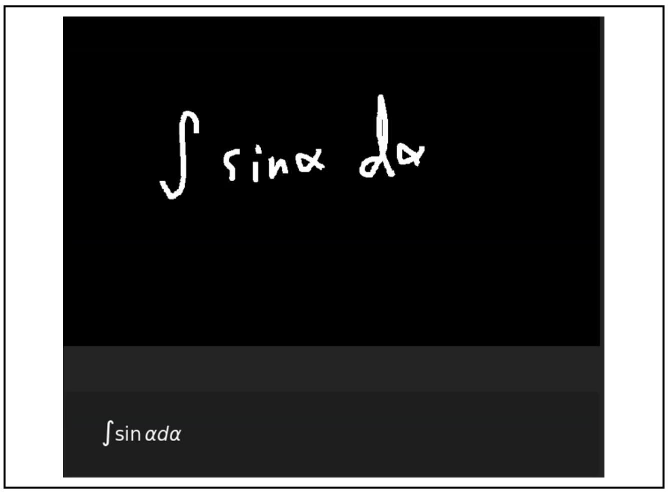


Overall accuracy converged at around 0.575.

It is not too high, due to the limited training data, and couldn't achieve higher even with data augmentation technique. The model design is great, but unfortunately the dataset we selected was insufficient so that it caused underfitting.

But it could handle more realistic daily usage. And have the capability to recognize complex symbols and grammar such as integration, fraction and exponent.

For example:



$\int \sin \alpha d\alpha$

Next time we would try it on a natively larger dataset such as *HME100K*.

IV. Data and Code Availability

This project focuses on recognizing handwritten mathematical expressions. The code and resources are publicly available on GitHub[1].

Our implementation relies on key dependencies such as TensorFlow GPU, Keras, Numpy, and others, operating within a Python 3.9 environment. To get started, follow the instructions to download the necessary files and checkpoints. Once setup is complete, execute ./model.py to launch the application and access the intuitive GUI.

V. Discussion / Conclusion

In this study, we developed a handwritten mathematical equation recognition system aimed at accurately interpreting complex handwritten inputs. The results demonstrate that the proposed model effectively handles various styles of handwriting, achieving a recognition accuracy of 57.5% on the test dataset.

A. *Problems and Improvement*

The overall recognition accuracy of our model for handwritten mathematical equations still has room for improvement. Although we attempted to use data augmentation techniques to increase the size of our dataset, the impact was limited. We speculate that the primary issue lies in the insufficient amount of training data, which may have hindered the model's ability to learn features effectively. In the future, expanding the dataset by collecting more diverse handwriting samples could further enhance the model's performance. Additionally, incorporating more advanced data augmentation techniques may significantly improve the recognition accuracy and practical applicability of our model.

Additionally, we can collect data that specifically targets various handwriting styles to further diversify the dataset. This enhancement could enable the model to cover a broader range of use cases in the future.

B. *Conclusion Purpose*

This project aims to create a deep learning system for recognizing handwritten mathematical symbols and equations. Its key applications include education, research, office productivity, and mobile device integration, enabling efficient digitization and processing of handwritten mathematical content.

The project enables the digitization of handwritten mathematical content. For example, teachers can use this system to convert students' handwritten equations and solutions into LaTeX format. This digitization facilitates online grading or AI-assisted evaluation, streamlining the assessment process and enhancing efficiency in educational settings.

Future work involves optimizing the model with larger datasets and advanced architectures, expanding support for complex mathematical structures, enhancing real-time recognition for mobile devices, integrating with educational tools such as Wolfram Alpha API and exploring research opportunities like NLP and low-resource language adaptability. These advancements aim to broaden the project's impact across various fields.

VI. Author Contribution Statements

CHEN,TING-JYUN(13%): Study design, peer review, final representation.

DIAU,I-PING(6%): Study design, peer review.

LEE,HUA-WEI(15%): Study design, data analysis, data collection, programming.

CHIANG,PEI-LIN(30%): Study design, image processing, programming, statistical and data interpretation, final presentation.

LIN,WEI-SHENG(36%): Data collection, data analysis, figure design and writing, programming, statistical and data interpretation, final presentation.

Acknowledgment

We would like to express our sincere gratitude to Professor Po-Chih Kuo for the invaluable guidance throughout the course and to the teaching assistants for their support beyond the classroom. Additionally, we appreciate the insights and perspectives provided by the authors of the referenced literature, which greatly contributed to the development of this work.

References

[1] Project gitHub https://github.com/jason34105533/ML2024-Group19-Handwritten-Expression-Recognition.git

[2] D. Zhelezniakov, V. Zaytsev, and O. Radyvonenko, "Online handwritten mathematical expression recognition and applications: A survey," IEEE Access, vol. 9, pp. 33986–34002, Mar. 2021, doi: 10.1109/ACCESS.2021.3063413.

[3] P. Gervais, A. Fadeeva, and A. Maksai, "MATHWRITING: A dataset for handwritten mathematical expression recognition," arXiv, arXiv:2404.10690 [cs.CV], Apr. 2024. [Online]. Available: https://arxiv.org/abs/2404.10690

[4] mathAI Dataset https://github.com/Roujack/mathAI/tree/master

[5] Pytorch-Handwritten-Mathematical-Expression-Recognition Dataset https://github.com/whywhs/Pytorch-Handwritten-Mathematical-Expression-Recognition