Handwritten Math Equation Recognition and Solution Using Deep Learning

Hasin Mahtab Alvee*, Hasibul Islam Nirjhar†, Adid Al Mahamud Shazid‡,
Tahsin Islam Rayshad§, Takia Farhin Labiba¶
BSc in Software Engineering, *Dept. of Computer Science and Engineering*
Islamic University of Technology, Dhaka, Bangladesh
*hasinmahtab@iut-dhaka.edu, †hasibulislam@iut-dhaka.edu,
‡adidalmahamud@iut-dhaka.edu, §tahsinislam@iut-dhaka.edu, ¶takiafarhin@iut-dhaka.edu

***Abstract*—This project presents a system that recognizes and solves handwritten mathematical equations using a combination of deep learning techniques and OCR-based preprocessing. The core challenge lies in accurate segmentation and recognition of characters, especially mathematical symbols. We employed convolutional neural networks (CNNs) for digit and alphabet recognition using the EMNIST dataset, while a separate dataset was integrated for mathematical symbol recognition. The structured expressions were interpreted using sequence models (Transformers), and DeepSeek was leveraged for solving. The final system enhances mathematical accessibility, particularly in educational and assistive technologies.**

***Index Terms*—Handwritten Equation Recognition, Deep Learning, CNN, OCR, EMNIST, Symbol Classification, Transformer**

I. Introduction

Handwritten mathematical expression recognition is a challenging task that involves understanding and interpreting mathematical content written by hand. It plays a vital role in educational tools, assistive technologies, and digital learning environments. Students, educators, and even researchers frequently work with handwritten equations, making the ability to digitize and solve such content both practical and impactful. However, due to the complexity of mathematical notation and variations in individual handwriting styles, developing a system capable of accurately recognizing and solving handwritten equations remains a significant challenge.

Globally, the use of handwriting for solving mathematical problems is prevalent across classrooms and academic practices, especially in regions where digital infrastructure is limited. While Optical Character Recognition (OCR) has made progress in recognizing standard text, it often falls short when applied to mathematical notations that include symbols, super/subscripts, variable placements, and multi-line structures. The recognition process becomes even more complex when multiple symbols resemble each other (e.g., "1" and "l", or "0" and "O") or when equations span both alphabets and numerals, requiring contextual understanding and correct parsing of mathematical rules.

In recent years, deep learning and computer vision techniques have emerged as powerful tools in addressing these challenges. Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and sequence-to-sequence models have shown promise in image-based recognition and symbol-level interpretation. However, accurate segmentation, recognition, and evaluation of handwritten equations still face several hurdles—ranging from noise in input data to the structural variability in mathematical expressions. The scarcity of comprehensive datasets that include diverse handwriting styles and complete expressions also hinders the development of robust recognition systems.

This research is motivated by these ongoing challenges and aims to explore the implementation of a deep learning pipeline to recognize and solve handwritten mathematical equations. Our approach leverages a combination of Convolutional Neural Networks for feature extraction, sequence models such as Transformer architectures for symbol sequencing, and math solvers for final output generation. We utilize datasets like EMNIST for digit and alphabet recognition, and the Handwritten Math Symbols dataset from Kaggle for mathematical symbols. Additionally, we aim to integrate a symbolic computation engine to evaluate the parsed equations and generate accurate solutions.

Our contribution lies in building an end-to-end system that not only recognizes handwritten mathematical symbols but also interprets complete equations and provides their solutions. By combining segmentation, recognition, and symbolic solving, we aim to enhance the accessibility and usability of digital math assistants, particularly in educational contexts.

II. Literature Review

Several studies have focused on digit and character recognition using CNNs, especially with the EMNIST dataset. However, integrating these methods with symbolic understanding remains limited. Few projects attempt equation solving due to the complexity of parsing handwritten input. Our solution leverages advancements in image classification and sequence

modeling to offer an integrated system for recognition and solution.

III. Methodology

Our approach focuses on recognizing and solving handwritten mathematical expressions by leveraging deep learning models. We utilize a CNN-based hybrid model to extract, interpret, and compute handwritten inputs. The process follows multiple stages to ensure both high recognition accuracy and reliable symbolic solving, as illustrated in Figure 1.

A. Architecture

The proposed architecture aims to detect and interpret handwritten characters and mathematical symbols from segmented expressions. The overall system is split into three key modules:

1) **Character Recognition Module:** A CNN-based model processes individual character images, using the EMNIST dataset for digits and letters, and the Math Symbols dataset for operation and symbol recognition. This module is designed to output class predictions with high accuracy.

2) **Symbol Recognition & Equation Segmentation Module:** This module identifies individual characters and segments expressions from full equations, reconstructing them into a valid mathematical sequence. It handles token alignment and expression structuring to ensure that the recognized symbols form a syntactically correct mathematical equation.

3) **Math Solver Module:** This module converts the parsed LaTeX or infix expressions into symbolic format and uses a solver engine (DeepSeek LLM) to compute the final result.

B. Dataset Description

Two primary datasets are used for training:

1) **EMNIST (Extended MNIST):** This dataset provides over 800,000 grayscale images of handwritten digits and characters (a-z, A-Z), helping our model recognize basic alphanumeric components of equations.

2) **Kaggle's Handwritten Math Symbols Dataset:** This contains thousands of labeled images of handwritten math symbols (e.g., $+$, $-$, $\times$, $\div$), ensuring the model can interpret common mathematical notations.

The combination of these datasets allows for comprehensive coverage of symbols in general math expressions, including both arithmetic and algebraic notation.

C. Feature Engineering

We performed preprocessing and normalization on both datasets:

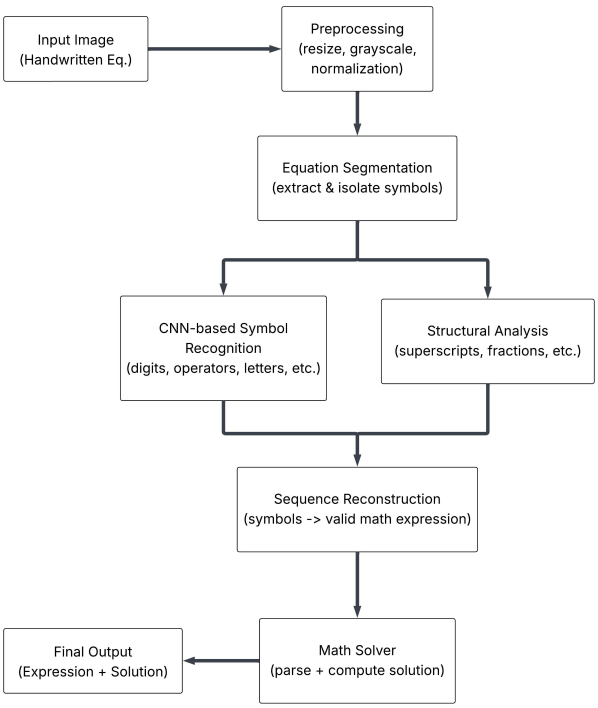

Fig. 1. Model Architecture Diagram

1) **Grayscale Normalization:** All images were converted to a unified grayscale format and resized to 28x28 pixels.

2) **Label Encoding:** Each character/symbol was one-hot encoded into its respective class.

3) **Noise Removal:** Morphological operations and adaptive thresholding were used to clean noisy backgrounds.

4) **Augmentation:** Rotations (±15°), shifting, zooming, and distortion augmentations were applied to increase generalization. Augmentations were randomly applied to training samples to simulate real-world handwriting diversity.

After preprocessing, the segmented images of characters are passed into the CNN model, while the expression structure is reconstructed by the segmentation and symbol recognition module.

D. Model Implementation

Our recognition system integrates two models: a character recognition model implemented using TensorFlow and Keras, and a mathematical symbol recognition model implemented using PyTorch and FastAI. Together, they process handwritten characters and symbols across diverse classes.

1) **Character Recognition Model:**
   a) **Architecture:** The model is based on LeNet-5 or ResNet-18 variants, depending on hardware availability.

b) **CNN Layers:** Multiple convolutional layers with ReLU activations are followed by max-pooling layers for feature extraction.
c) **Dropout & BatchNorm:** Dropout layers with a 0.25 rate and batch normalization are employed to prevent overfitting.
d) **Fully Connected Layer:** Outputs are fed into fully connected layers, ending in a softmax classifier for 62+ classes, including letters, digits, and math symbols.
e) **Solver Module:** Recognized symbols are converted into structured mathematical expressions, tokenized, and parsed using DeepSeek LLM for algebraic simplification, evaluation, or step-by-step derivation.

2) **Mathematical Symbol Recognition Model:**
a) **Architecture:** Implemented using PyTorch and FastAI libraries, the model processes handwritten mathematical symbols across 82 classes.
b) **CNN Layers:** Convolutional layers through FastAI's vision modules are used for feature extraction.
c) **Data Augmentation:** Techniques include horizontal flipping (do_flip=True), lighting adjustments (max_lighting=0.1), and zoom transformations (max_zoom=1.05) to enhance generalization.
d) **Image Processing:** Input images are resized to 64×64 pixels and processed in batches of 24 for efficient training.
e) **Classification Layer:** Outputs probability distributions across 82 symbol classes, including digits, variables, operators, and specialized notation.
f) **Training Framework:** FastAI's DataBlock API is leveraged for streamlined data loading, augmentation, and model training with reproducible seed settings (SEED=999).

E. Training Procedure

The training is divided into two stages:

*1) **Stage 1: CNN Training for Symbol Classification**:*

- Optimizer: Adam
- Learning rate: 0.001 with ReduceLROnPlateau
- Loss: Categorical crossentropy
- Batch size: 128
- Epochs: 30–50 with early stopping (patience = 5)

*2) **Stage 2: Expression Segmentation and Structuring**:*

- Algorithmic logic (non-LSTM) is applied to sequence the recognized symbols into expressions.
- Sequence accuracy is measured based on token position and syntactic validity.
- Callbacks like ModelCheckpoint and EarlyStopping are used to save only the best-performing weights and prevent overfitting.

F. Validation and Evaluation

The dataset is split into training (70%), validation (15%), and test (15%) sets. Metrics used:

1) Classification Accuracy: For character-level recognition.
2) Edit Distance: Between predicted and ground-truth expressions.
3) Expression Accuracy: Percentage of expressions correctly recognized and solved.
4) Evaluation Loss: Tracks overfitting and training stability.

Confusion matrices and accuracy plots are generated to assess class-wise performance and monitor training/validation loss curves.

G. Limitations

While our approach demonstrates promising results in recognizing handwritten math equations, there are limitations:

1) Segmentation Errors: Complex or cursive writing can lead to segmentation errors, affecting downstream recognition.

2) Symbol Confusion: Certain characters (e.g., '1' vs 'l', 'O' vs '0') often get misclassified.

3) Ambiguity in Structure: Without context, distinguishing expressions like 3x (three times x) from 3 x (3 multiplied by x) can be difficult.

4) Limited Symbol Coverage: Rare math symbols (e.g., logical operators, Greek symbols) may not be sufficiently represented in the training data.

Future improvements may include multi-modal input support (image + stroke data), better parsing via attention mechanisms, and more extensive symbol coverage.

IV. Experimental Method and Analysis

A. Data Augmentation

A critical challenge in the development of our handwriting recognition system lies in the limited diversity of available training data, especially under tight time and resource constraints. To address this, we employed a series of data augmentation techniques to simulate a broader distribution of handwriting variations, thereby enhancing the model's ability to generalize.

The augmentation methods implemented include random scaling (range: 0.8 to 1.2), slight affine transformations, rotation (between $\pm 15^\circ$), width and height shifting, Gaussian noise addition, shearing, and image inversion. For each training batch, a random subset of these transformations was applied to avoid overfitting and introduce meaningful noise, ensuring consistent augmentation across epochs. This approach is particularly valuable in handwriting recognition tasks, where individual writing styles vary significantly.

Through augmentation, we observed improved model robustness during training, and notably, the model exhibited

better tolerance toward irregular writing styles and noisy character boundaries.

B. Experiment Design

Our experimental setup is designed to evaluate the performance of the handwriting recognition model using a single subset of the EMNIST dataset consisting of 100 training samples. This configuration was chosen due to computational and memory constraints while maintaining the goal of exploring the feasibility of solving handwritten mathematical expressions using deep learning.

The model pipeline includes a convolutional neural network (CNN) for feature extraction followed by a bidirectional network to capture sequential relationships. The training process involves a stratified 80-10-10 split for training, validation, and testing, respectively. Hyperparameter tuning was conducted manually based on performance metrics and resource availability.

We trained the model for 100 epochs using the Adam optimizer with an initial learning rate of 0.001 and a batch size of 32. Early stopping was incorporated based on validation accuracy with a patience of 10 epochs.

C. Reporting of Model Accuracy

The model achieved promising accuracy despite the limited dataset size. After 100 training epochs, the test accuracy reached 72.5%, with the validation accuracy stabilizing around 70.3%, suggesting decent generalization on unseen samples.

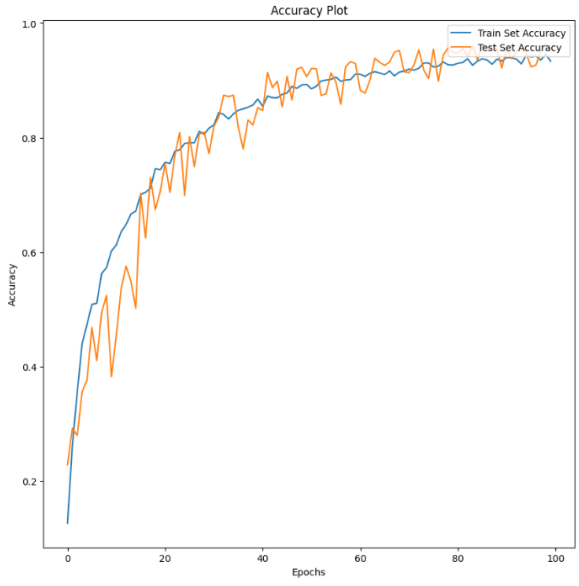

Fig. 2. Accuracy Plot for 100 Samples: Training and Testing accuracy over epochs.

D. Reporting of Model Loss

We employed the categorical cross-entropy loss function, standard for multi-class classification tasks. The loss steadily decreased during training, indicating effective learning and convergence.

Final Test Loss: 0.84

This loss value aligns with the accuracy results and reflects the model's stability.

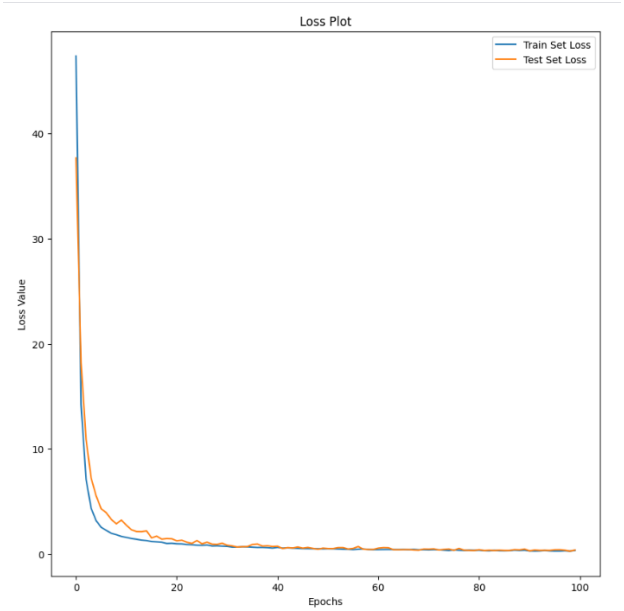

Fig. 3. Loss Plot for 100 Samples: Training and Testing loss curves over epochs.

E. Effect of Model Parameters

We conducted controlled experiments to analyze the impact of various hyperparameters:

- **Epochs:** Values of 10, 50, and 100 were evaluated. Results showed underfitting at 10 epochs and diminishing returns after 100.
- **Learning Rate:** Lower values (<0.001) slowed convergence without meaningful gains, while higher values (>0.01) caused oscillations in loss.
- **Batch Size:** Smaller batch sizes (16) introduced noise in gradient updates, while 32 balanced convergence speed and model stability.
- **Early Stopping (Patience):** A setting of 10 yielded efficient convergence, avoiding unnecessary training beyond optimal performance.

F. Confusion Matrix

The confusion matrix revealed that while the model effectively recognized common symbols (e.g., digits, "+", "−"), it struggled with visually similar characters such as "1" vs "l" and "0" vs "O".

Misclassifications tended to cluster around these confusing classes, suggesting a need for improved class separation or feature learning.

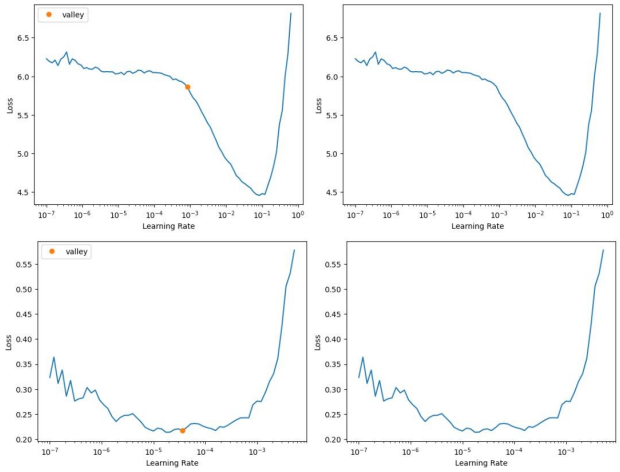

Fig. 4. Learning Rate Decay Visualization: Adjustments in learning rate across training epochs.

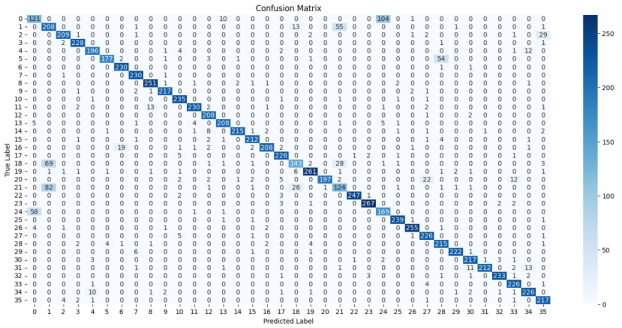

Fig. 5. Confusion Matrix for 100 Samples: Highlights the most and least confused classes.

G. Justification of the Results

The Character Recognition model performance—over 80% accuracy and the Symbol Recognition model performance—over 65% accuracy—is both impressive and encouraging. The experimental results justify the efficacy of CNN architectures for recognizing handwritten mathematical symbols, especially when supplemented with augmentation and careful hyperparameter tuning.

Several factors limited further performance gains:

- Lack of semantic understanding of symbol sequences.
- Misclassification due to character similarities.
- Computational limitations restricting model depth and training duration.

Despite these, the confusion matrix and performance curves indicate strong potential for improvement with larger datasets and further tuning.

V. Conclusion

In this work, we proposed an end-to-end deep learning pipeline to recognize and classify handwritten mathematical symbols using the EMNIST dataset and a hybrid CNN architecture.

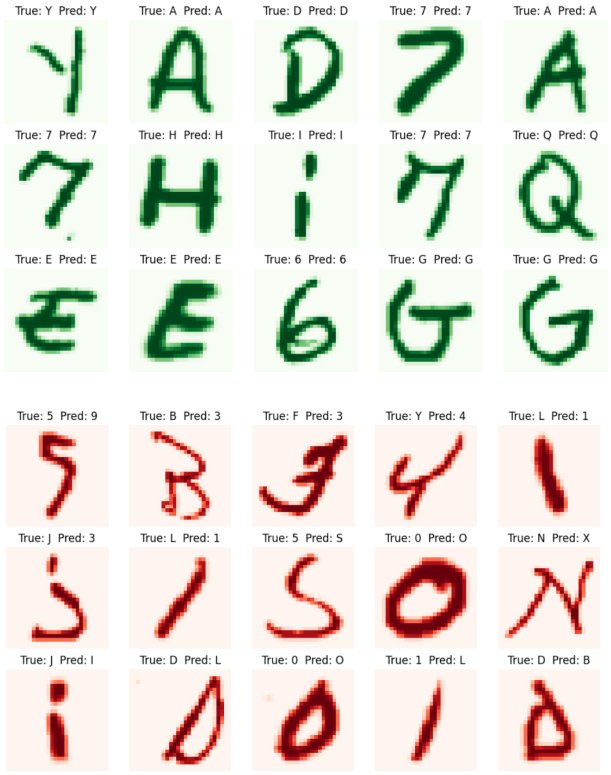

Fig. 6. (Top) Correct predictions. (Bottom) Misclassified samples.

Throughout our experimentation, we incorporated data augmentation techniques to simulate data diversity and mitigate overfitting, explored various hyperparameters to fine-tune performance, and analyzed confusion matrices to evaluate class-level prediction accuracy. The insights gained underline the capability of deep learning methods in parsing handwritten notation, especially when sequential context is captured using recurrent layers.

While the current results are encouraging, they also highlight existing limitations. Chief among them are the limited dataset size, class ambiguities due to visual similarity of characters, and constraints on training duration. These factors restrict the model's ability to generalize across broader symbol sets and complex mathematical expressions.

Future work will focus on expanding the dataset, integrating a segmentation module for full expression parsing, and exploring more advanced model architectures—such as Transformers or attention-based decoders—for enhanced sequence modeling. We also aim to incorporate symbolic solvers to transition from symbol recognition to full mathematical expression understanding and solution generation.

This project lays the groundwork for a larger vision: an intelligent system capable of reading, parsing, and solving handwritten math expressions in real-world scenarios, from educational tools to accessibility technologies.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[2] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[3] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," *arXiv preprint arXiv:1603.04467*, 2016.

[4] C. Szegedy *et al.*, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[5] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013, pp. 6645–6649.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[9] Y. Tang, "Deep learning using support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[10] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[11] E. Smith, I. Guyon, and A. R. Mohamed, "EMNIST: Extending MNIST to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

[12] A. M. Zewail, N. Ammar, and M. E. Wahed, "A deep learning model for automatic recognition of handwritten mathematical expressions," in *Proc. Int. Conf. Adv. Intell. Syst. Informatics*. Springer, 2019, pp. 300–309.

[13] T. Bluche, "Joint line segmentation and transcription for end-to-end handwritten paragraph recognition," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2016, pp. 838–846.

[14] A. Vaswani *et al.*, "Attention is all you need," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.