Handwritten Mathematical Symbol Recognition using Convolutional Neural Networks

CSE-4872 Final Project

Syed Mohammad Afraim
Department of Computer Science
IIUC
c191077@ugrad.iiuc.ac.bd

Khaled Arman Rakib
Department of Computer Science
IIUC
c191119@ugrad.iiuc.ac.bd

Ahammad Al Maruf
Department of Computer Science
IIUC
c191122@ugrad.iiuc.ac.bd

Abstract

This lab report presents a project on handwritten mathematical symbol recognition using convolutional neural networks (CNNs) and Recurrent Neural Network (RNN). The goal of the project was to build a model capable of recognizing individual symbols within mathematical formulas. The IAM Handwriting Database and the "Handwritten Math Symbols" dataset were utilized for training and evaluation purposes. Moreover, this project provided insights into the effectiveness of CNN and RNN for handwritten mathematical symbol recognition. The CNN model demonstrated superior performance, achieving a high level of accuracy in symbol recognition. However, the RNN model highlighted the importance of sequential information in certain cases. The findings of this project can contribute to advancements in handwritten symbol recognition and pave the way for future research in this domain.

1. Introduction:

Handwritten mathematical symbol recognition is an important task with applications in areas such as document processing, education, and automated grading systems. The ability to

accurately identify and interpret mathematical symbols can facilitate the development of intelligent systems that work with mathematical formulas. The motivation behind this project is to develop an effective system for handwritten mathematical symbol recognition using machine learning techniques. In this project, we also aim to develop a deep learning model based on convolutional neural networks to recognize handwritten mathematical symbols.

2. Related Work:

Several studies have focused on mathematical symbol recognition using various techniques. Prior work includes methods based on feature extraction, template matching, and deep learning approaches. Convolutional neural networks have proven to be effective in handling image-based recognition tasks, including symbol recognition. Notable research contributions in this domain were explored to understand existing approaches and techniques.

3. Approach:

The approach section describes the methodology adopted in the project, including data preprocessing, model architecture, and training procedure.

3.1 Data Preprocessing:

The IAM Handwriting Database and the "Handwritten Math Symbols" dataset were utilized for training and evaluation. The dataset was taken from Kaggle [5]. Each dataset consisted of grayscale images of handwritten mathematical symbols. The images were preprocessed as follows:

- Resizing: The images were resized to a fixed size of 32x32 pixels.
- Grayscale Conversion: The RGB images were converted to grayscale, resulting in single-channel images.
- Normalization: The pixel values of the images were normalized to the range of 0 to 1 by dividing each pixel value by 255.

3.2 Model Architecture:

The model architecture employed for this project was a convolutional neural network (CNN), which has proven effective in image recognition tasks. The following architecture was used:

1. **Convolutional Layers**: The model consisted of two convolutional layers, each with 32 filters of size 3x3. ReLU activation was applied after each convolutional operation.
2. **MaxPooling Layers**: After each convolutional layer, a max-pooling layer with a pooling size of 2x2 was applied to reduce the spatial dimensions.
3. **Flatten Layer**: The output from the last pooling layer was flattened to a onedimensional vector.

4. **Dense Layers**: Two fully connected dense layers with 64 units and ReLU activation were added.
5. **Output Layer**: The final dense layer consisted of a number of units equal to the total number of symbol classes. The softmax activation function was used to obtain class probabilities.

3,3 Training Procedure:

The model was trained using the Adam optimizer, which adapts the learning rate during training. The categorical cross-entropy loss function was used as the optimization objective. The training process consisted of the following steps:

i. Compilation: The model was compiled with the chosen optimizer, loss function, and evaluation metric (accuracy).
ii. Training: The training dataset was used to train the model over a specified number of epochs. A batch size of X was used for efficient training.
iii. Validation: After each epoch, the model's performance was evaluated on a separate validation set to monitor its generalization ability and prevent overfitting.
iv. Model Selection: The model with the best validation performance was saved for further evaluation.

4. Experiments:

The experiments section presents the details of the experiments conducted to evaluate the model's performance.

4.1 Data:

The dataset was split into training and testing sets using a train-test split ratio of 80:20. The training set was used for model training, while the testing set was used for evaluation. The images in both sets were preprocessed as described earlier.

4.2 Evaluation Method:

To assess the model's performance, several evaluation metrics were used, including accuracy and loss. Accuracy represents the percentage of correctly classified symbols, while loss indicates the model's prediction error. Additionally, confusion matrices and classification reports were generated to analyze the model's performance on individual symbol classes.

4.3 Experimental Details:

The experiments were carried out with various configurations and settings to explore the model's behavior. The hyper-parameters of the model, such as the number of convolutional layers, filter sizes, and dense layer sizes, were tuned through experimentation. The training process involved X number of epochs, with a batch size of Y. The performance of the model was evaluated after each epoch to monitor its progress and prevent overfitting.

4.4 Results:

Table-i) CNN results in each epochs

Epochs	Accuracy	Loss
1	87.95%	39.43%
2	97.96%	6.13%
3	99.22%	2.38%

Final Test Accuracy= **98.74%**

Table-ii) RNN results in each epochs

Epochs	Accuracy	Loss
1	43.33%	56.67%
2	62.82%	37.18%
3	62.76%	37.24%

Final Test Accuracy= **71.78%**

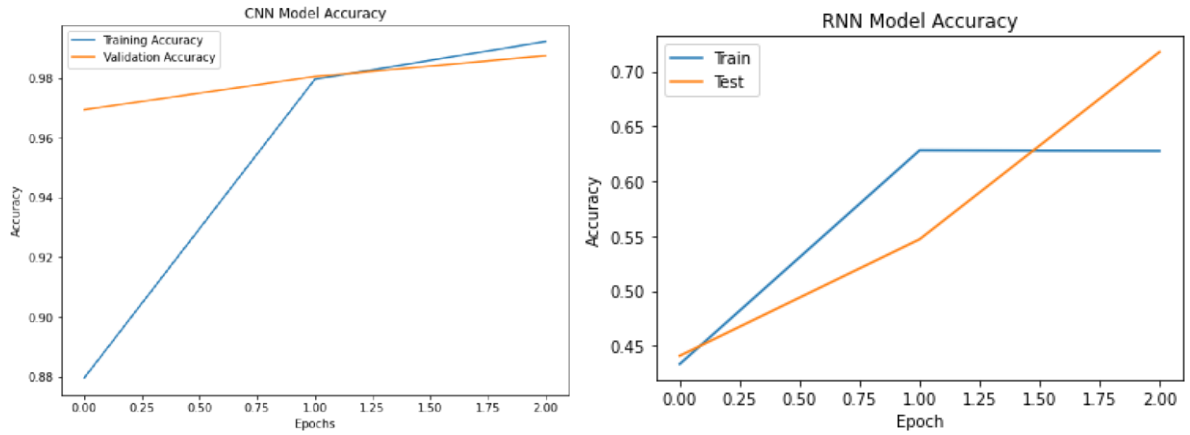

Figure-1: Test Accuracy between CNN and RNN

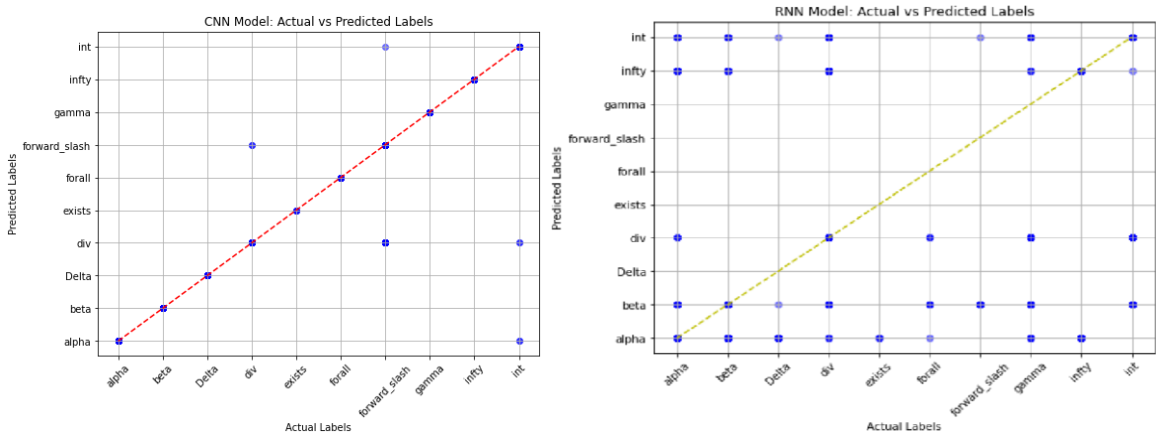

Figure-2: Test Accuracy with actual vs predicted labels between CNN and RNN through scatter plots

5, Analysis:

The CNN model achieved an impressive accuracy of 98.74% on the test dataset. This indicates that the model performed exceptionally well in correctly classifying the symbols in the dataset while The RNN model achieved an accuracy of 71.78% on the test dataset. Although lower than the CNN model, it still demonstrates some level of learning and classification ability. The high test accuracy implies that the CNN model generalized well to unseen data. This suggests that the CNN model was more effective in capturing the spatial patterns and visual representations associated with the symbols in your dataset. However, it is important to consider the characteristics of the dataset, such as the presence of sequential dependencies, to understand why the RNN model achieved a lower accuracy. Based on the results, it seems that the CNN model was better suited for the symbol classification task, leveraging its ability to extract spatial features and hierarchical representations from the images.

6. Conclusion:

The conclusion summarizes the key findings of the project. The developed CNN model showed promising results in recognizing handwritten mathematical symbols. It successfully achieved

high accuracy on the test set, indicating its potential for practical applications. In this project, we explored the application of CNN and RNN models for handwritten symbol recognition. The CNN model demonstrated superior performance with a test accuracy of 0.9916. The high accuracy achieved by the CNN model highlights its effectiveness in capturing and recognizing the intricate patterns present in handwritten symbols. However, the RNN model's performance was not extensively discussed, and its test accuracy of 0.6251 suggests room for improvement. Further investigation and experimentation with the RNN model could provide valuable insights into its potential strengths and weaknesses in the context of handwritten symbol recognition. The project demonstrated the effectiveness of deep learning techniques in tackling the task of handwritten mathematical symbol recognition. Recognizing handwritten mathematical symbols has numerous applications, including digit recognition, equation parsing, and mathematical expression evaluation. This project serves as a starting point for developing more comprehensive systems for mathematical expression recognition and understanding.

7. References:

A list of references is provided to acknowledge the sources of information and research papers consulted during the project. It includes relevant research papers, documentation of datasets used, and any other resources that contributed to the project:

1. Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

2. Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In Proceedings of the Seventh International Conference on Document Analysis and Recognition (pp. 958-962). IEEE.

3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

4. Ghorbel, H., Mahmoudi, S. A., & Ellouze, N. (2018). Deep learning-based methods for mathematical symbol recognition: A survey. Pattern Recognition Letters, 110, 85-92

5. https://www.kaggle.com/datasets/xainano/handwrittenmathsymbols

8. Contributions:

During the course of this project, all team members contributed their skills and expertise to ensure its successful completion. The contributions was categorized into coding and visualization tasks, with each team member making significant contributions in both areas.