George Mason University
April 7, 2024
AIT 736

Handwritten Character Recognition using CNN: A Deep Learning Approach

Sophia Herman sherman5@gmu.edu
Suparna Mannava smannav@gmu.edu
Naga Sai Dhanya Veerepalli nveerepa@gmu.edu
Sai Advaith Vootukur svootuku@gmu.edu

Abstract

We will examine the complex workings of Convolutional Neural Networks (CNNs) and how important they are for reading handwritten characters . Also investigate the subtle differences of model designs, optimization techniques, and preprocessing approaches with the goal of expanding the field of handwritten content analysis.

Introduction

Through the use of handwritten character recognition technology, computers are now able to comprehend and interpret handwritten text, translating it into a digital format that can be processed and examined. It is essential to many applications in many sectors and businesses.

Here is a quick summary of its significance:

- ❏ Document Digitization
- ❏ Automation of Form Filling and Improved Accessibility
- ❏ Natural Human-Computer Communication
- ❏ Document Interpretation and Analysis

1
8
5
17
18
23
23
11
15
3
10
8
16
8
12
10
10
22
10
24
13
3
19
10
16
13
9
18
23
17
4
17
18
13
10
11

Problem Formulation

The primary objective of this project is to create a **CNN Model** that can correctly identify handwritten characters from the **EMNIST** dataset.

We specifically seek to address the following key challenges:

- Dataset preparation
- Model construction
- Optimization and training
- Evaluation
- Visualization

Challenges

- Data Standardization
- Data Augmentation
- Class Imbalance
- Model Complexity
- Hyperparameter Tuning
- Evaluation Metrics

Pre-development Research

Objective: Explore convolutional neural network (CNN) designs, optimization techniques, and data preprocessing methods to advance handwritten character recognition

Significance: Emphasizes the critical role of CNNs in accurately interpreting handwritten characters, pivotal for digital text analysis and automation

Research Approach:

- Comparative analysis of model architectures to determine optimal configurations for character recognition
- Evaluation of various preprocessing techniques to enhance model training efficiency and accuracy
- Investigation into advanced optimization methods for robust model performance

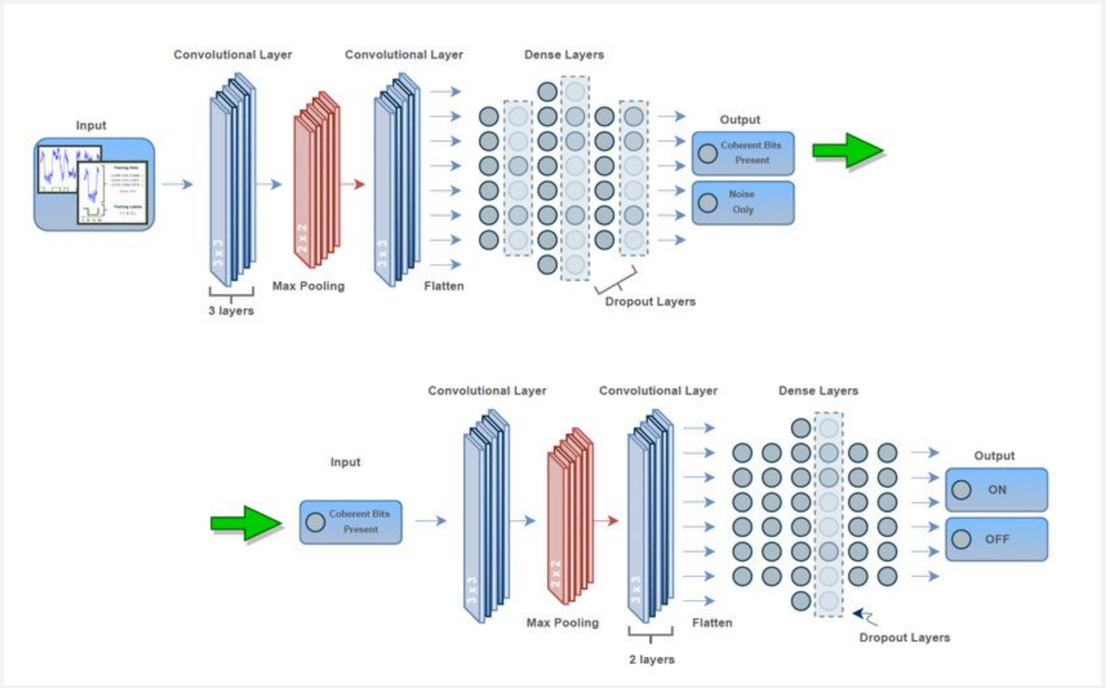
Member 3 | Sai Advaith Vootukur

The Proposed Model

Core Structure: Sequential CNN model designed for high-accuracy character recognition

Key Components:

- Initial Shape Reshaping Layer: Convert input images into 28x28 dimensions, single channel
- Convolution2D Layers: Two layers with 32 filters each, employing ReLU activation and max-norm regularization
- MaxPooling2D Layer: Reduces dimensionality, simplifying feature maps
- Flatten Layer: Prepares the data for classification by transforming it into a 1D array
- Dense Layers: A 512-neuron layer for deep feature processing and a 62-neuron layer with Softmax activation for final classification into 62 character classes



Member 3 | Sai Advaith Vootukur

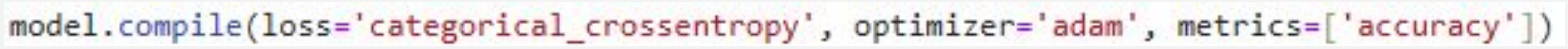
Model Method

Activation Functions:

- ReLU (Rectified Linear Unit): Utilized in hidden layers for introducing non-linearity, enhancing the model’s capability to learn complex patterns.
- Softmax: Applied in the output layer, converting logits to probabilities for each class, facilitating multi-class classification.

Loss Function:

- Cross-Entropy Loss: Employed for its effectiveness in multi-class classification, focusing on minimizing discrepancies between predicted probabilities and actual labels.



```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

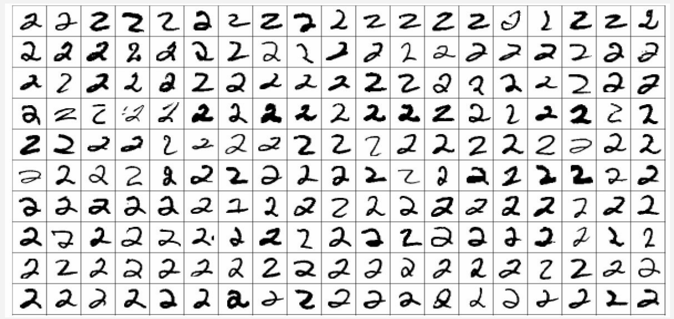
Training Strategy

Optimizer: Adam, selected for its efficiency in gradient computation, expediting the convergence process

Batch Size: Set at 128 images, optimizing the balance between computational efficiency and model performance

Epochs: Model trained for 50 epochs, with early stopping triggered by validation loss to prevent overfitting, ensuring model generalizes well to unseen data

Validation Strategy: Utilization of a validation set to fine-tune hyperparameters and a separate test set for unbiased evaluation of the final model performance



Data Description

- The **EMNIST (Extended Modified National institute of Standards and technology)** dataset is a set of handwritten character digits and converted to a 28x28 pixel image format.
- This dataset is the **MNIST(Modified National institute of Standards and technology)** which is one of the six parts of the EMNIST dataset, which specifically contains the handwritten characters. This MNIST is subset of the larger data from the **NIST** data.
- This dataset contains 70000 characters and 10 unbalanced classes.
- The training set contains **60,000** examples and the test set contains **10,000** examples. It is a subset of larger set of data from NIST and been normalized and center fixed-size image.
- The first file of the training set contains the **28*28 matrix** of each image pixel values and the other training file contains the labels of the image. Same goes with the test set.

Data Preprocessing

- ➔ The image files contain the pixel values of images ranging from 0 to 255. We normalized it to a range between 0 and 1 so that it can be easy while we apply the **CNN** model to it.
- ➔ Then we convert the 1D array of pixel data to a **2D array** which can be easily processed by CNN models
- ➔ The we convert the labels to categories as CNN is a classification algorithm.
- ➔ We can think of the labels to be the target variables to identify what character the image corresponds to.

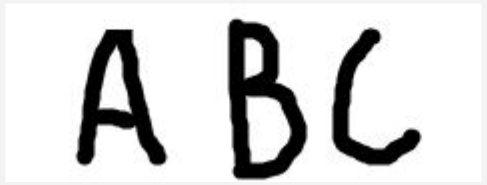
Training v.s. Testing Datasets

- As said, training set contains 60000 characters and test set contains 10000 characters
- For this set of data **Convolution Neural Networks** is applied
- The activation function used at input and hidden layers is the **Rectified Linear Unit (ReLU)**.
- For the output layer, we used the **Softmax** activation for the output layer.
- The applied model is thus saved in model.weights.h5 file
- For testing the model we used the example.png image and applied the trained model to this example.
- Before applying the model to the example a fixed model and kernel is applied to the space in which the characters should appear

Results

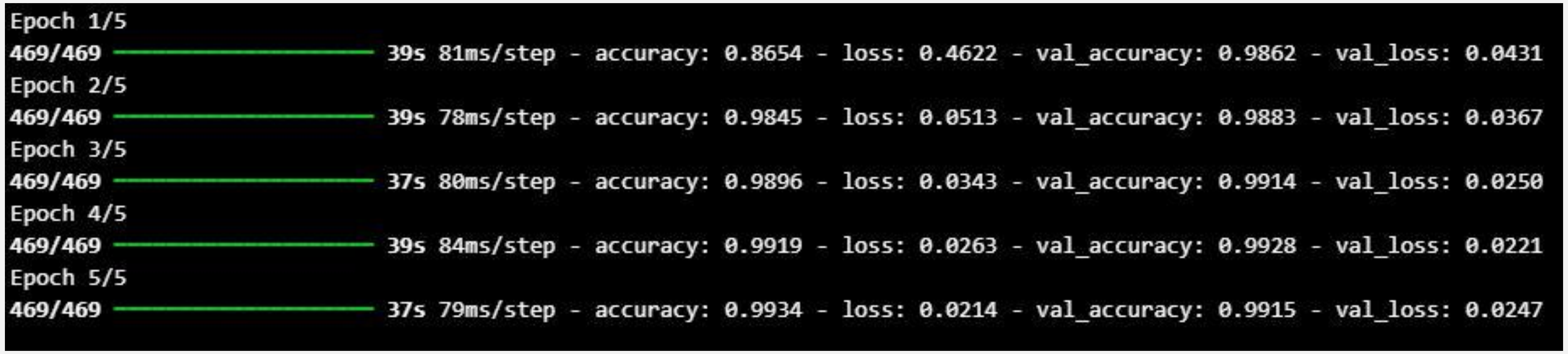
Test Input:

Handwritten letters "ABC".



Processing:

- Consistent increase in accuracy and decrease in loss over 5 epochs.
- Uses of validation data to check for overfitting and ensure generalizability.

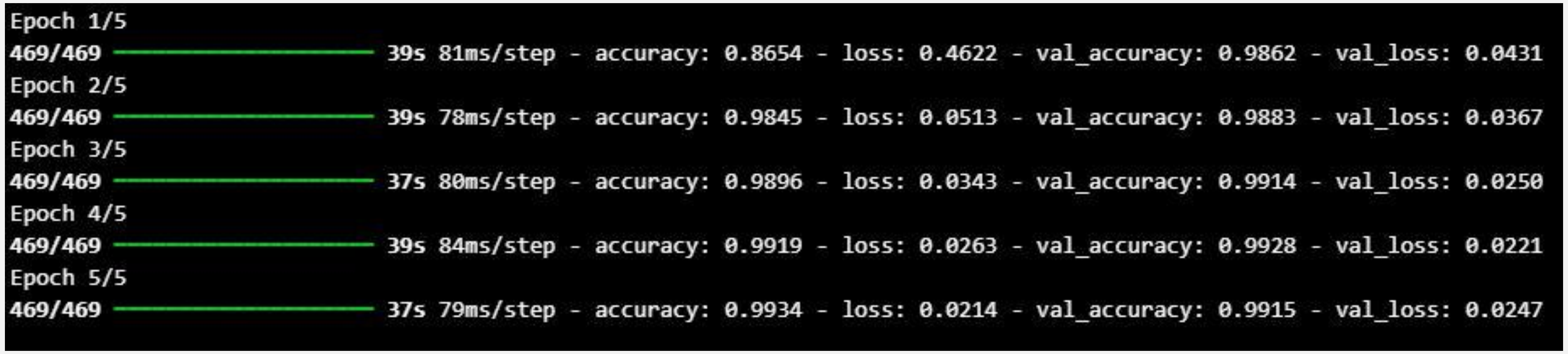


```
Epoch 1/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 39s 81ms/step - accuracy: 0.8654 - loss: 0.4622 - val_accuracy: 0.9862 - val_loss: 0.0431
Epoch 2/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 39s 78ms/step - accuracy: 0.9845 - loss: 0.0513 - val_accuracy: 0.9883 - val_loss: 0.0367
Epoch 3/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 37s 80ms/step - accuracy: 0.9896 - loss: 0.0343 - val_accuracy: 0.9914 - val_loss: 0.0250
Epoch 4/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 39s 84ms/step - accuracy: 0.9919 - loss: 0.0263 - val_accuracy: 0.9928 - val_loss: 0.0221
Epoch 5/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 37s 79ms/step - accuracy: 0.9934 - loss: 0.0214 - val_accuracy: 0.9915 - val_loss: 0.0247
```

Results

Performance:

- The training and validation accuracy are closely aligned; a sign that the model is generalizing well, rather than memorizing.
    - Approximately 37-39 seconds per epoch, and 78-84 milliseconds per step.
    - Training accuracy started at 86.54% and improved to 99.34%.
    - Training and validation loss decreased from 0.4622 to 0.0214.
    - Validation accuracy started high at 98.62% and improved slightly to 99.15%.
    - Validation loss decreased from 0.0431 to 0.0247.



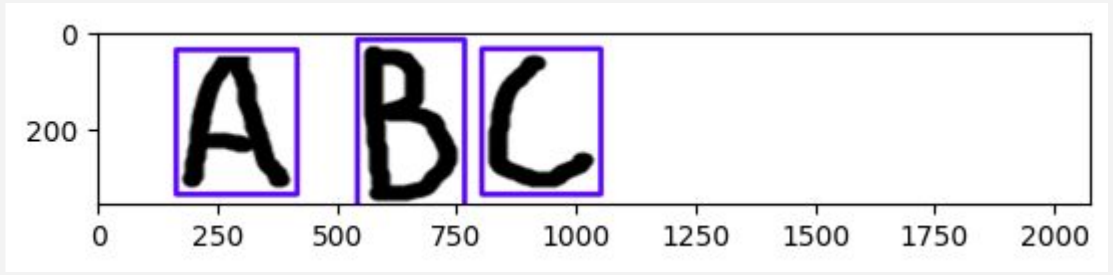
```
Epoch 1/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 39s 81ms/step - accuracy: 0.8654 - loss: 0.4622 - val_accuracy: 0.9862 - val_loss: 0.0431
Epoch 2/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 39s 78ms/step - accuracy: 0.9845 - loss: 0.0513 - val_accuracy: 0.9883 - val_loss: 0.0367
Epoch 3/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 37s 80ms/step - accuracy: 0.9896 - loss: 0.0343 - val_accuracy: 0.9914 - val_loss: 0.0250
Epoch 4/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 39s 84ms/step - accuracy: 0.9919 - loss: 0.0263 - val_accuracy: 0.9928 - val_loss: 0.0221
Epoch 5/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 37s 79ms/step - accuracy: 0.9934 - loss: 0.0214 - val_accuracy: 0.9915 - val_loss: 0.0247
```

Results

Input:

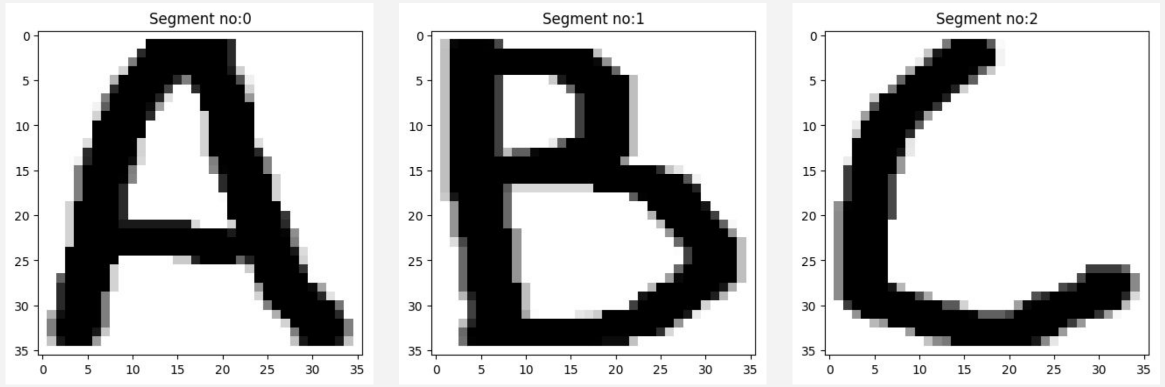
Handwritten letters "ABC".

The characters are detected within bounding boxes using a segmentation algorithm to localize characters within a larger image.



Output:

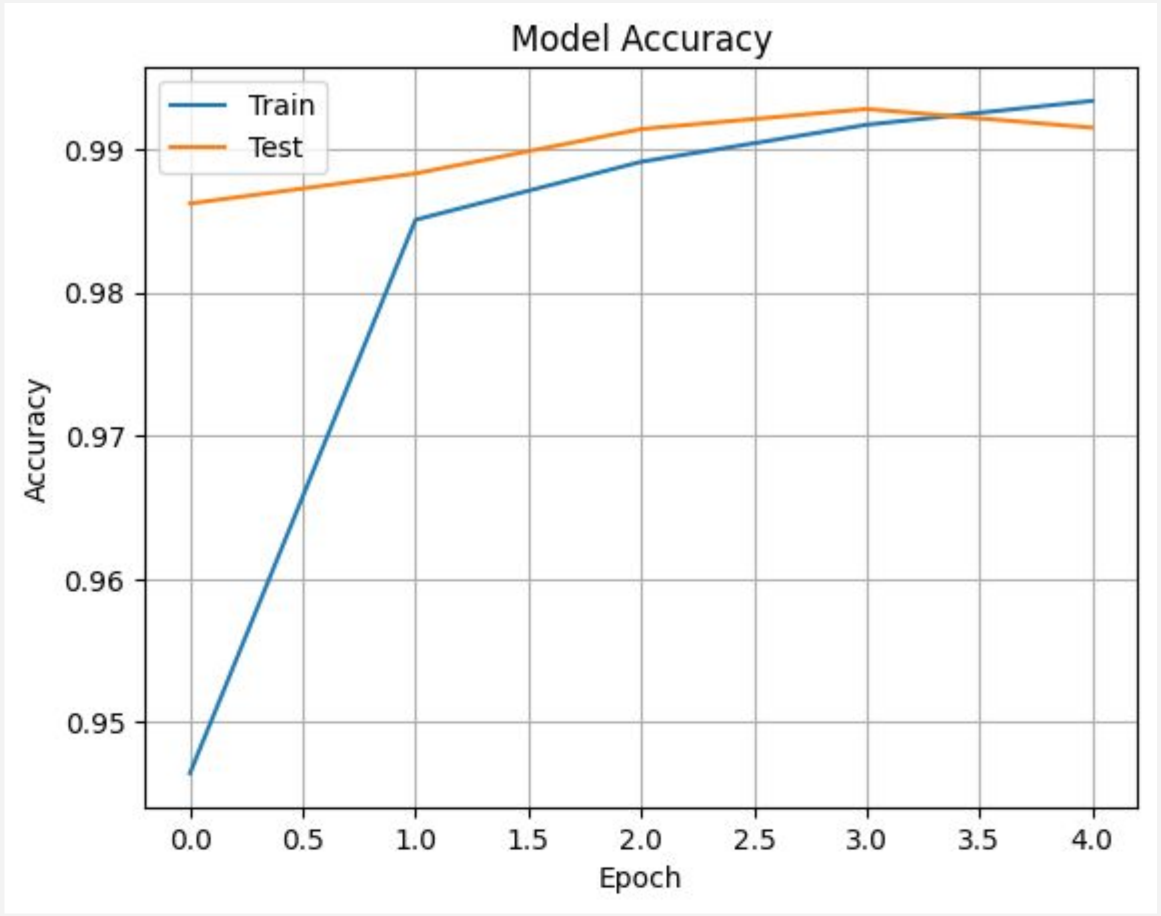
Grayscale contour of "ABC".



Analysis

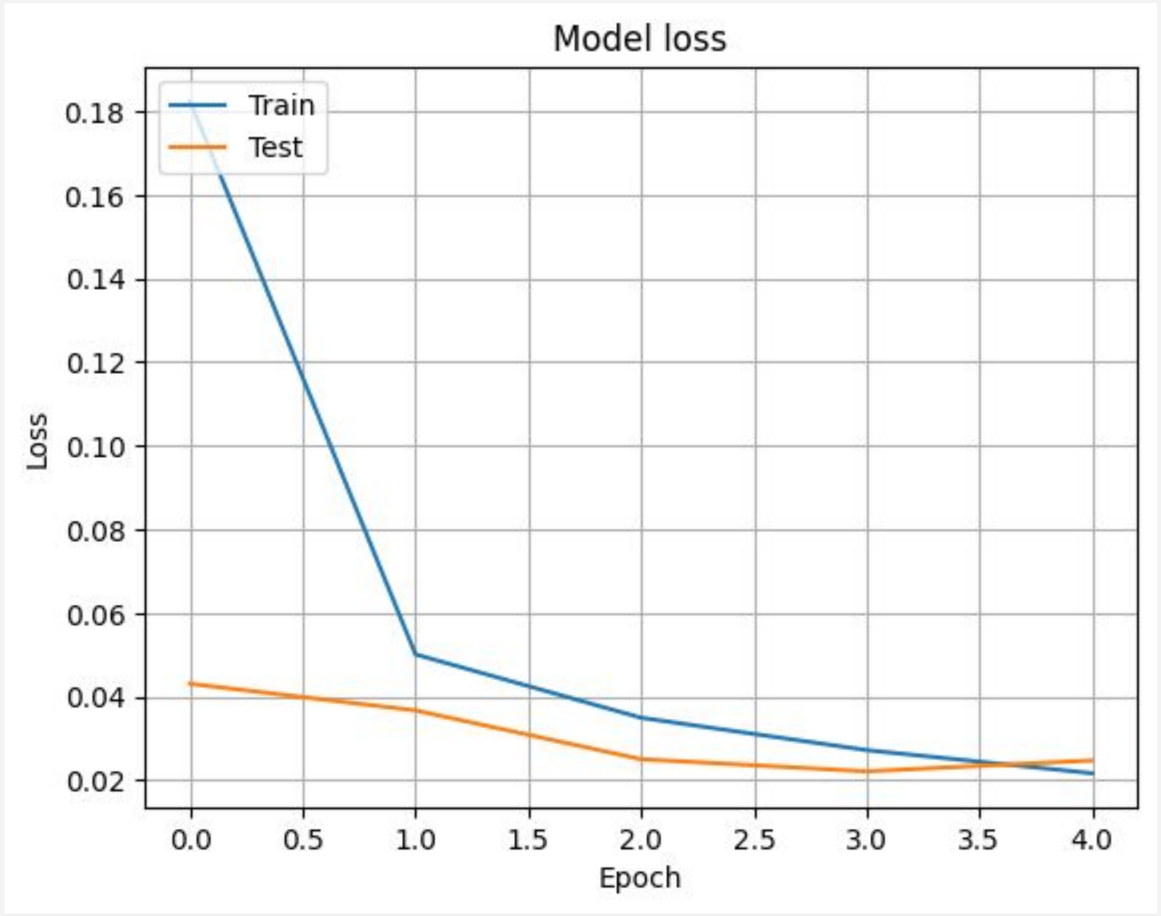
High degree of alignment between training and test accuracy.

The slight convergence suggests that the model isn't overfitting, which implies that the model will perform well on data outside of the training set.



Analysis

Loss decreases rapidly after the first epoch and continues to decrease at a diminishing rate. This is typical in training neural networks, but improvements become more incremental as the model starts to converge.

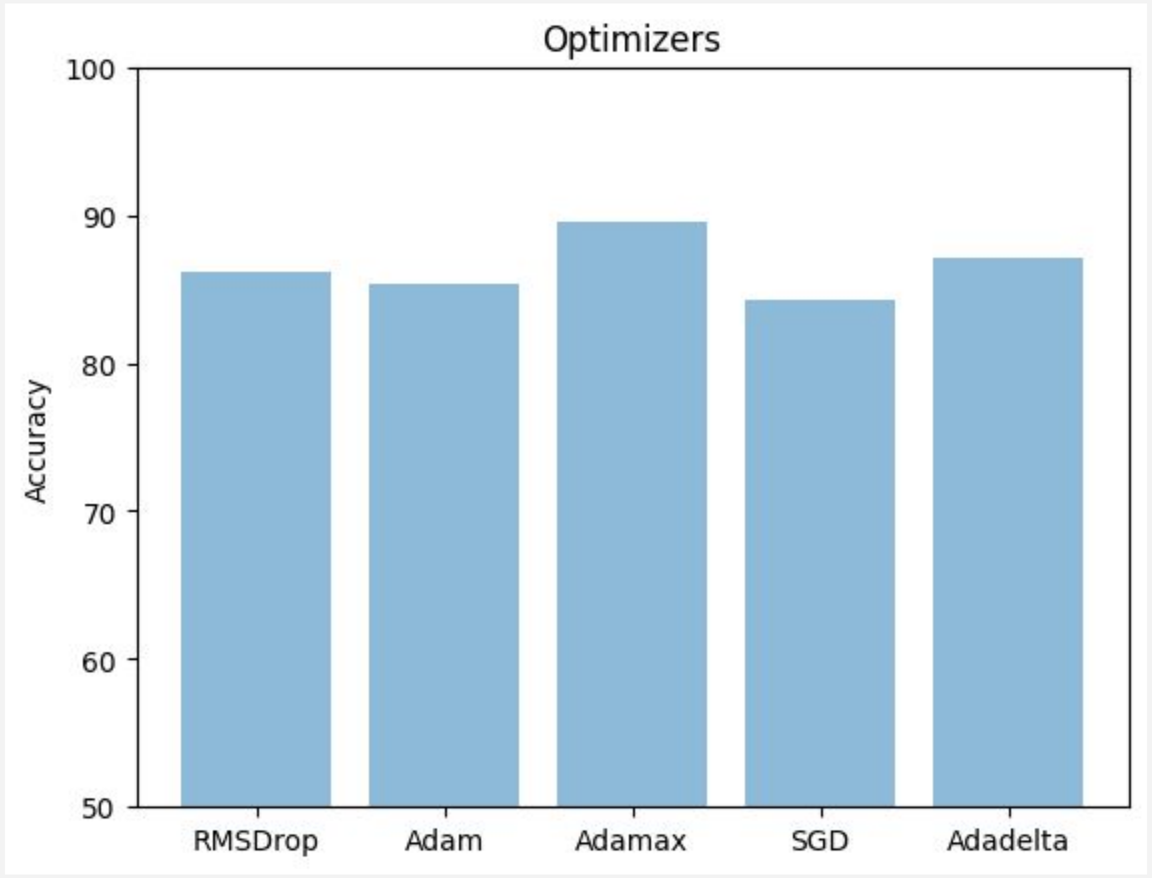


Analysis

All optimizers show relatively similar performance.

Adam and Adamax slightly outperform the others.

Stochastic Gradient Descent (SGD) and Adadelta indicate a lower performance. This could be due to the learning rate settings or characteristics of the dataset. These optimizers may require careful tuning of the learning rate and can benefit from momentum and learning rate schedules.



Discussion

- If the validation loss begins to increase while training loss decreases, that could be an early sign of overfitting.
  - To avoid this, we could consider implementing techniques such as regularization (e.g., L1, L2), dropout (randomly dropping out nodes during training), or data augmentation (randomly altering images to increase the dataset's diversity).
- The accuracy of this model indicates suitability for practical applications.
  - This work could be substantial in the context of automating tasks that currently require human labor, as well as improving accessibility through text recognition.
- Our future research should aim to explore the model's applicability to a broader range of characters, different languages, or in more challenging conditions (e.g., different handwriting styles, noisy backgrounds, etc.).

Discussion

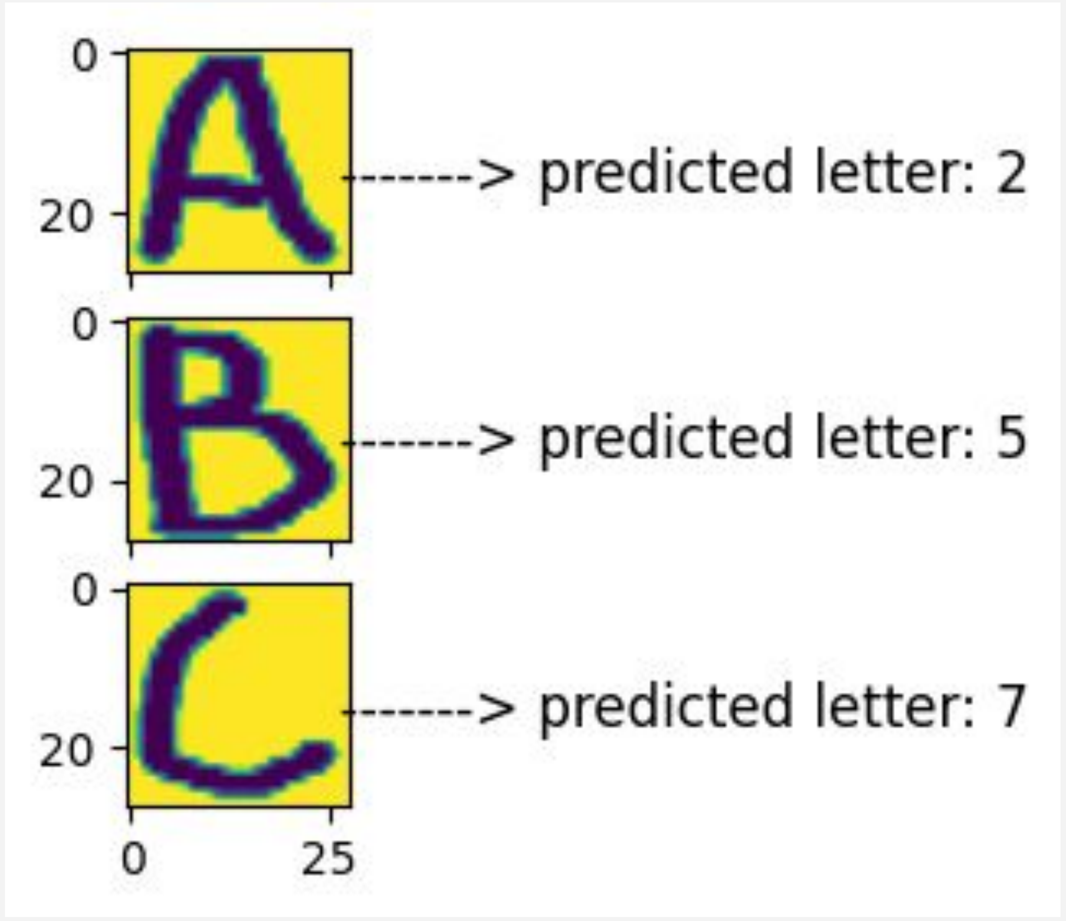
- The model is performing excellently with high accuracy and low loss, suggesting it's well-configured for the character recognition task.
  - The optimizer analysis suggests that Adam and its variations are good default choices, but depending on the exact use-case and computational resources, other optimizers might be preferred.
- The test accuracy surpasses the training accuracy at certain points, which could be due to a regularization effect or the presence of certain patterns in the test set that the model is able to exploit better.
  - It’s best to test the model and ensure these results maintain their accuracy with recognition tasks.

Experiment with the Test Model

The predicted labels ('2', '5', and '7') do not correspond to the expected alphabetical characters ('A', 'B', 'C').

While the model seems to be detecting characters successfully, the prediction step is not accurately mapping these letters to the correct alphabetical characters.

This can happen due to an encoding fault, insufficient training dataset size, or a misinterpretation caused by problems hidden in the post-processing steps.



Conclusion

Overall, this study emphasizes the importance of deep learning approaches, specifically CNNs, in handwritten character identification tasks. With potential applications in document processing, optical character recognition (OCR), and digital image analysis, among other domains, the discoveries boost automated character recognition systems. In order to improve recognition accuracy and resilience, future study may concentrate on enhancing the model architecture even more, investigating sophisticated training techniques, and tackling particular issues.

References

[1] “MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges,” yann.lecun.org. http://yann.lecun.org/exdb/mnist/index.html

[2] Banerjee, K., Gupta, R. R., Vyas, K., & Mishra, B. (2020). Exploring alternatives to softmax function. *arXiv Preprint arXiv:2011.11538*.

[3] Dong, X., Li, W., Wang, X., & Wang, Y. (2019). Learning a deep convolutional network for colorization in monochrome-color dual-lens system. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 8255–8262.

[4] Hamdan, Y. B., & Sathesh, A. (2021). Construction of statistical SVM based recognition model for handwritten character recognition. *Journal of Information Technology*, *3*(02), 92–107. https://scholar.archive.org/work/vb6spp7oavhb5he34jzzuzzwwa/access/wayback/https://irojournals.com/itdw/V3/I2/03.pdf

[5] Heydarian, M., Doyle, T. E., & Samavi, R. (2022). MLCM: Multi-label confusion matrix. *IEEE Access*, *10*, 19083–19095.

[6] Kiliçarslan, S., & Celik, M. (2021). RSigELU: A nonlinear activation function for deep neural networks. *Expert Systems with Applications*, *174*, 114805.

[7] Miao, Z., Li, Y., & Wang, X. (2021). Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond. *Proceedings of the 2021 International Conference on Management of Data*, 1303–1316.

[8] Mohd, M., Qamar, F., Al-Sheikh, I., & Salah, R. (2021). Quranic optical text recognition using deep learning models. *IEEE Access*, *9*, 38318–38330.

George Mason University
April 7, 2024
AIT 736

Thank you.

Sophia Herman	sherman5@gmu.edu
Suparna Mannava	smannav@gmu.edu
Naga Sai Dhanya Veerepalli	nveerepa@gmu.edu
Sai Advaith Vootukur	svootuku@gmu.edu