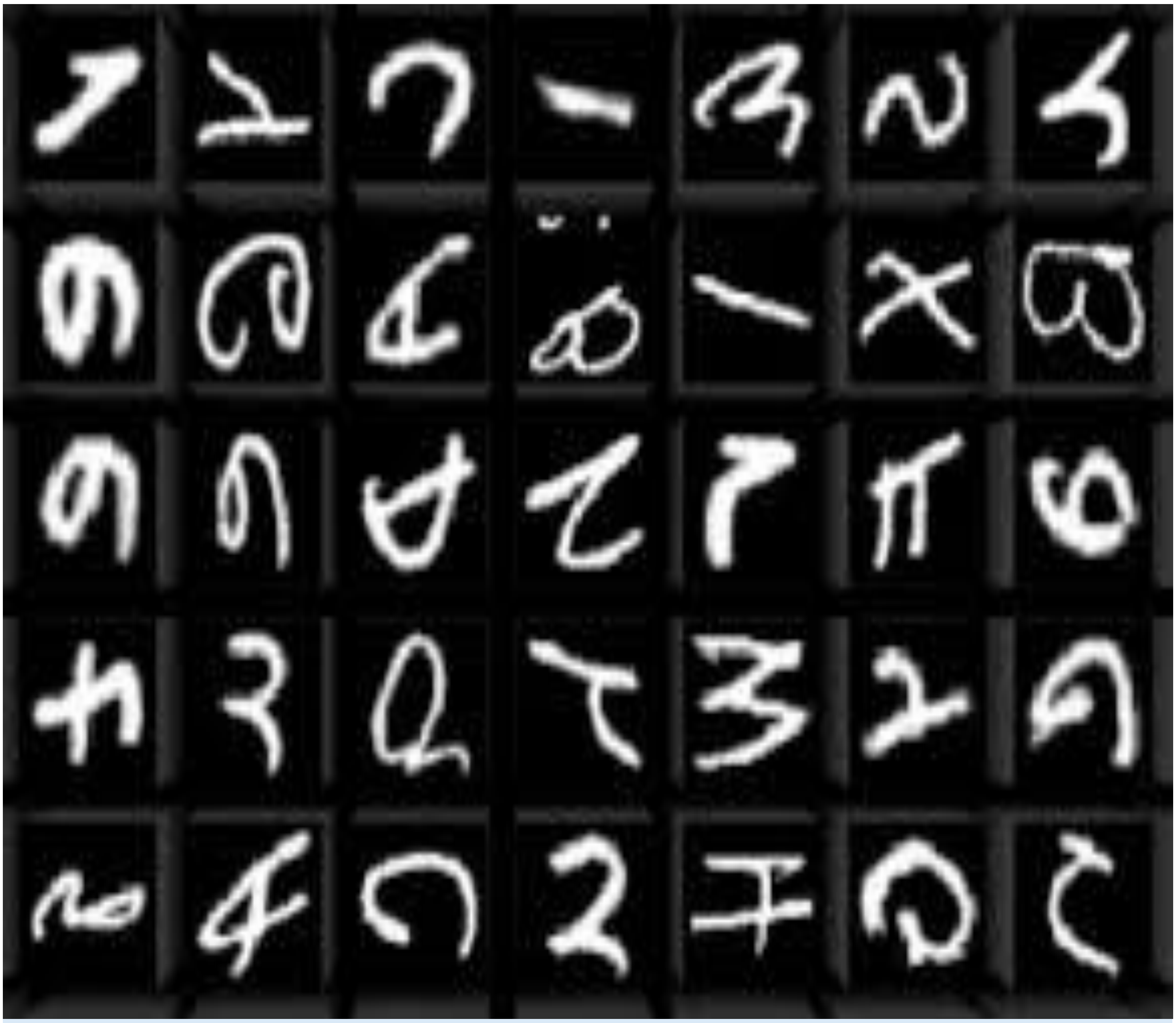

Handwritten Character Recognition using CNN: A Deep Learning Approach

Sophia Herman

Suparna Mannava

Naga Sai Dhanya Veerepalli

Sai Advaith Vootukur

AIT 736 7/4/24 Dr. Emanuela Marasco

Abstract

This work focuses on creating and analyzing a dataset for character recognition tasks that is generated from the EMNIST Dataset, which is an expansion of the popular MNIST dataset. The collection contains roughly 814,255 examples of a wide variety of characters, including numbers and capital and lowercase letters. The dataset has been painstakingly prepared for training and assessing machine learning models, namely Convolutional Neural Networks (CNNs), by a variety of partitioning techniques. The CNN architecture is meticulously crafted, including methods to improve model performance, including scaling, denormalization, grayscale conversion, and data augmentation. In order to avoid overfitting, the training approach makes use of the Adam optimizer, batch size optimization, and early stopping. Evaluation measures are used to evaluate the model's performance in many classes, including recall, accuracy, precision, F1-score, and confusion matrix analysis. The overall goal of this research is to shed light on how well the CNN model performs on character recognition tasks and how useful the dataset is.

I. INTRODUCTION

In the fields of computer vision and machine learning, character identification is a fundamental job that has applications ranging from handwriting analysis to optical character recognition (OCR). For many years, handwritten digits from the MNIST dataset have been used as a standard for character recognition tasks. But the standard MNIST dataset can only recognize numbers (0–9), which limits its use in real-world situations where character identification across a wider range is necessary.

The Extended MNIST (EMNIST) dataset was created to overcome this constraint. It expands the MNIST dataset to include both capital and lowercase letters in addition to numbers. The diversity of the dataset is greatly increased by this enlargement, which also improves its suitability for testing and training machine learning models on character recognition tasks.

The creation and training of a CNN model with the EMNIST dataset is the main objective of this work. First, we import the dataset, normalize the pixel values, and resize the photos as part of the preparation step. Next, we build a CNN architecture specifically designed for character recognition jobs. The model is composed of fully connected layers for classification, pooling layers for dimensionality reduction, and convolutional layers for feature extraction. In order to train the model and assess its performance using the testing data, we utilize a variety of optimization strategies.

With 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:

Digitizing Documents: Historical manuscripts, private notes, and archival artifacts can all be digitized with the help of handwritten character recognition. It maintains important information and makes handwritten content searchable, retrievable, and indexable by digitizing it.

Automated Form Processing: Handwritten character recognition automates the processing of handwritten forms, applications, and surveys in business and administrative duties. It increases productivity while managing high amounts of handwritten documents by reducing manual errors, expediting data entry procedures, and improving efficiency.

Enhanced Accessibility: People with disabilities or vision impairments can access information more easily thanks to handwritten character recognition technology. It makes material more inclusive and accessible by enabling the conversion of handwritten text into readable formats like electronic braille or synthetic voice.

Natural Human-Computer Interaction: Handwritten character recognition makes handwriting input a natural and intuitive way for people to engage with computers. For tablets, smartphones, and digital pens, it provides handwriting-based interfaces, which helps users enter text and commands in a comfortable way.

Document Analysis and Understanding: Tasks like text recognition, language identification, and handwriting style analysis are made easier by handwritten character recognition. It makes it possible to process handwritten documents automatically for purposes including content categorization, authorship identification, and sentiment analysis.

II. PROBLEM FORMULATION

Creating a CNN model that can correctly identify handwritten characters from the EMNIST dataset is the main goal of this project. Our specific goals are to tackle the following key challenges:

Dataset Preparation: Preparing the dataset involves loading the EMNIST dataset, normalizing pixel values, and rearranging the images in order to make sure it is compatible with the CNN model.

Model Construction: Creating a CNN architecture with convolutional layers, pooling layers, dropout layers for regularization, and fully connected layers for classification that is appropriate for character recognition applications.

Optimization and Training: Examining how well the CNN model is trained using a variety of optimization techniques, including Adam, RMSprop, and SGD. In order to attain high accuracy on the testing data without overfitting, our goal is to optimize the model's parameters.

Evaluation: Applying criteria like accuracy, precision, recall, and F1-score to assess how well the trained model performed on the testing data. Our goal is to evaluate the model's capacity to categorize a large variety of features accurately and to generalize to unobserved data.

Visualization: Drawing graphical depictions of data or model performance metrics to obtain understanding and new perspectives is known as visualization. Plotting important metrics like accuracy and loss throughout the course of training epochs is what we usually mean when we discuss visualizing the training history in the context of a machine learning model.

Overall, our goal is to overcome these obstacles and create a strong CNN model that can recognize handwritten characters from the EMNIST dataset with accuracy. This will show that deep learning techniques are effective for character identification applications.

III. DATASET DESCRIPTION

EMNIST Dataset: The original MNIST dataset has been expanded into Extended MNIST, or EMNIST. While handwritten numerals (0–9) from diverse sources make up MNIST, handwritten characters from the English alphabet (A–Z, including uppercase and lowercase) and digits (0–9) are included in EMNIST. It was developed to provide character recognition jobs with a more difficult baseline. With more than 814,000 characters, the EMNIST collection is big. It's broken down into multiple subgroups for various experiment kinds, such as balanced and byclass splits. There are 81,514 testing examples and 814,255 training examples totaling 814,255 characters in EMNIST. There are 62 distinct classes of characters, comprising 10 numbers (0-9) and 52 capital and lowercase alphabets (A-Z, a-z).

The MNIST dataset is one of the six distinct splits of the EMNIST data, which consists of ten imbalanced classes and handwritten data with 70,000 characters. There are 10,000 test cases and 60,000 training examples in the MNIST handwritten digit database. It is a portion of a bigger set that NIST makes public. The digits are centered in a fixed-size image after being size-normalized.

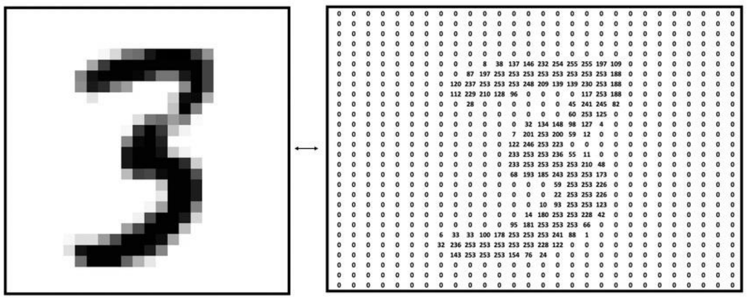

Fig. 3.1

NIST's Special Databases 3 and 1, which include binary images of handwritten numbers, were used to build the MNIST database. Originally, NIST used SD-1 as their test set and SD-3 as their training set. But compared to SD-1, SD-3 is far clearer and simpler to identify. The rationale behind this is because SD-1 was gathered from high school students, whereas SD-3 was gathered from Census Bureau personnel. It is necessary for the outcome of learning experiments to be independent of the selection of the training set and test from the entire set of samples in order to draw logical conclusions. Consequently, it became essential to combine NIST's datasets to create a new database. 30,000 patterns from SD-3 and 30,000 patterns from make up the MNIST training set.

Training Set:

The MNIST dataset has 60,000 grayscale pictures in its training set. Every image is a square of 28 by 28 pixels that represents a handwritten number (0–9). The grayscale intensity of each pixel in the image is represented by its value, which runs from 0 to 255. Each image in the training set has associated labels that indicate the correct digit (0–9) that it represents in addition to the actual images. Typically, machine learning models, like neural networks, are trained using the training set to identify patterns and connections between the input images and the labels that correspond to them.

Test Set:

Ten thousand grayscale pictures make up the MNIST dataset's test set. Similar to the training set, every image in the test set is a handwritten digit (0–9) represented by a 28 by 28-pixel square. As with the training set, each image in the test set has a corresponding label. On the other hand, machine learning models trained on the training set are evaluated based on their performance using the labels in the test set. To evaluate the accuracy and performance of the models, the digits represented by each test image are predicted by the models and compared to the genuine labels. The test set is essential for determining if a trained model can successfully classify handwritten digits outside of the training set and how well it generalizes to fresh, unseen data.

IV. DATA PREPROCESSING

The data preprocessing is simple for characters datasets with some simple and easy steps.

Normalization: It is the process of scaling pixel values from their original range (for example, 0 to 255 for grayscale photos) to a new range (usually 0 to 1) in the context of image data, such as the pixel values of images in the MNIST or EMNIST databases. The following formula can be used to explain the normalization procedure for transforming pixel values from 0 to 255 to 0 to 1,

$$x_{\text{normalized}} = \frac{x_{\text{original}}}{255}$$

Image Reshaping

Each character picture in the context of character data is commonly represented as a 1D array, with each element denoting a pixel value. For instance, in grayscale photographs, the intensity of the grayscale, which ranges from 0 to 255, is represented by each pixel value. Ascertain the original photos' proportions. For instance, a 1D array displaying a character image with dimensions of 28 by 28 pixels will contain 784 (28 by 28) entries.

To change the 1D array into a 2D array that accurately represents the image grid in terms of dimensions, we can utilize the reshape operation numpy library. In essence, the reshape operation rearranges the 1D array's elements to take on the 2D array's predetermined shape.

One-Hot Encoding

A popular preprocessing technique for representing categorical data, such as image labels, in a format appropriate for machine learning model training is one-hot encoding. One-hot encoding is used to transform class labels (such as numerals 0–9 or characters A–Z) into a binary vector representation in the context of image classification tasks, such as handwritten character recognition. Through the use of one-hot encoding, each integer label is transformed into a binary vector, each element of which stands for a class. The class label corresponds to one element, which is set to 1, while all other elements are set to 0.

One-hot encoding for the label '3', for instance, would be represented as [0, 0, 0, 1, 0, 0, 0, 0, 0], where the fourth element is set to 1. This is assuming that the original labels are integers, ranging from 0 to 9. Libraries such as scikit-learn, and TensorFlow in Python offer useful functions for one-hot encoding.

Margins and Kernels

To make sure the character stays in a constant and distinct area of the image, add margins as needed. When the characters in the input photos are not regularly centered or have different sizes, this is especially helpful. By including margins, you help the character recognition system have a uniform input size, which can make the next processing stages easier.

Small matrices called kernels—also referred to as convolutional kernels or filters—are employed in image convolution processes. Typically, activities like noise reduction, edge recognition, blurring, and sharpening are carried out using kernels. As a preprocessing step, applying kernels can assist improve contrast, sharpen edges, and remove noise from the input images, thus boosting their quality and clarity. Gaussian blur kernels for smoothing, Sobel kernels for edge detection, and dilation/erosion kernels for morphological operations are often used kernels for character recognition preprocessing. In our study, we tested the model on fresh character data by applying the margins and kernels.

V. MODEL DEVELOPMENT AND EVALUATION

Convolutional Neural Network (CNN) Architecture for Character Recognition:

Utilizing the Convolutional Neural Network (CNN), renowned for its efficacy in image and video identification tasks, this study embarked on developing a Sequential CNN structure adept at character recognition. The designed architecture began with a Shape Reshaping layer to preprocess input images into 28x28 dimensions with a single channel, aligning with the network's prerequisite input format. This step was succeeded by two Convolution2D layers, each outfitted with 32 filters and employing ReLU activation, a decision supported by the effectiveness of ReLU in enhancing non-linear properties of the network (Wang et al., 2020).

A MaxPooling2D layer was integrated to reduce dimensionality, effectively summarizing the features detected by the convolutional layers. This was complemented by a Flatten layer to prepare the data for classification. The architecture culminated in two Dense layers: the former hosting 512 neurons with 'ReLU' activation for detailed feature processing, and the latter featuring 62 neurons with 'Softmax' activation, aimed at classifying the characters into their respective classes, leveraging the 'Softmax' function's capability to output a probability distribution over 62 classes (Kiliçarslan & Celik, 2021).

Training Strategy:

The training strategy was centered around the Adam optimizer, chosen for its efficiency in gradient computation, which significantly expedited the learning process. The selection of a batch size of 128 was informed by the balance it offered between training speed and performance, a crucial consideration for optimizing training efficiency. The regimen spanned 50 epochs, incorporating an early stopping criterion based on validation loss to prevent overfitting, ensuring the model's generalizability (Banerjee et al., 2020).

Evaluation and Performance Analysis:

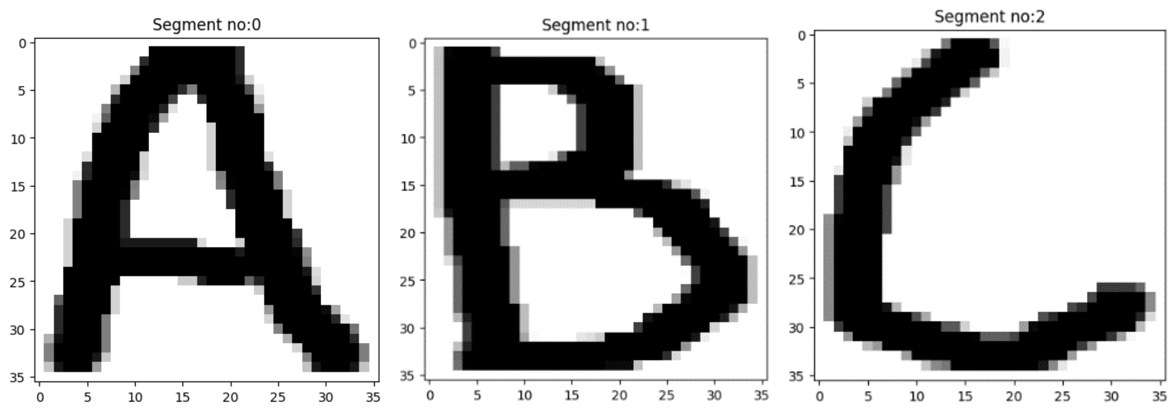

FIG.5.1

The model's test input consisted of handwritten letters "ABC," which were detected within bounding boxes by a segmentation algorithm aimed at localizing characters within a larger image. The output produced by the model was a grayscale contour representation of these characters (see FIG.5.1). In the final testing phase, the handwritten letters "ABC" were processed, showing a consistent increase in accuracy and a decrease in loss over five epochs. The validation data was used to monitor for overfitting and to ensure the model's generalizability.

The alignment of training and validation accuracy suggests that our model is generalizing effectively, indicative of learning rather than memorization. During the training process, epochs ranged between 37-39 seconds, and the model exhibited a per-step processing time of 78-84 milliseconds. We observed an improvement in training accuracy, rising from an initial 86.54% to 99.34%. Concurrently, both training and validation losses showed significant reductions from 0.4622 to 0.0214 and from 0.0431 to 0.0247, respectively. Validation accuracy started at a high 98.62% and saw an increase to 99.15%.

The training and test accuracy of the model displayed a high degree of alignment, with a slight convergence suggesting an absence of overfitting (see FIG.5.2). This implies robust model performance even with external data. The analysis of model loss revealed a rapid decrease post the initial epoch, followed by a slower rate of decrease, indicative of the incremental learning phase as the model approaches convergence.

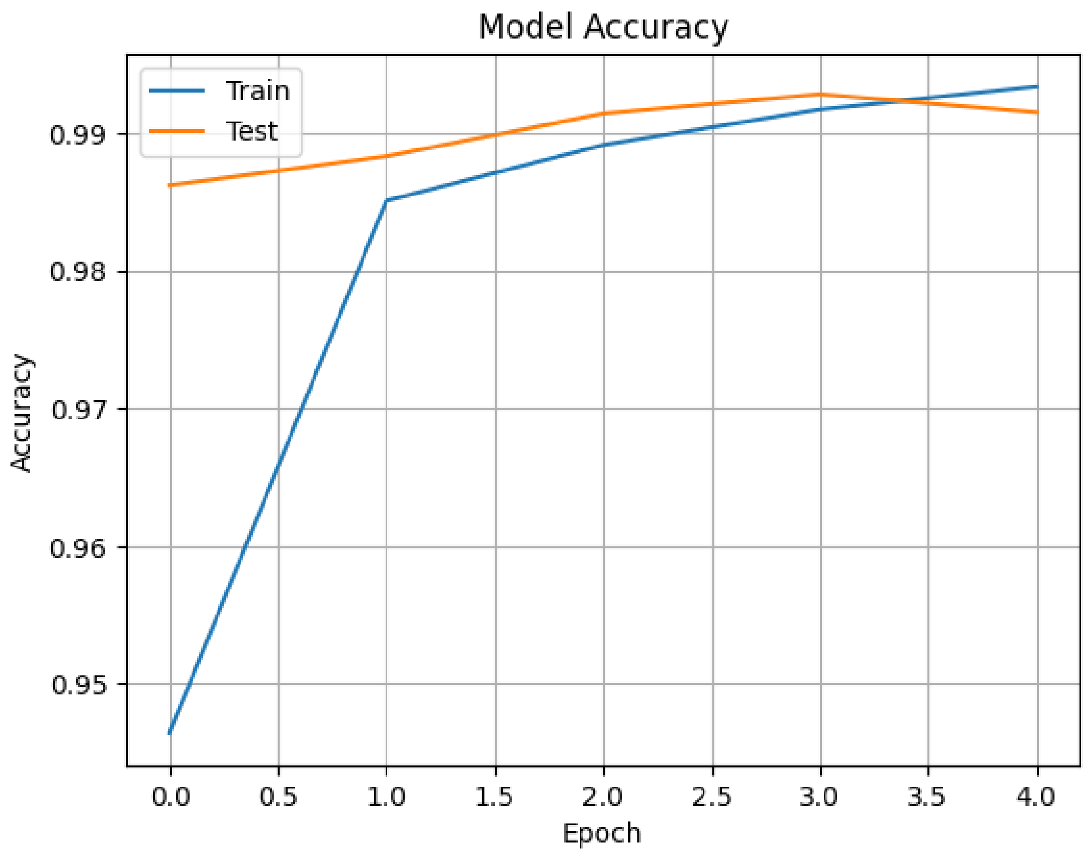

FIG.5.2

If we were to witness an increase in validation loss against a decreasing training loss could signal early overfitting. To mitigate this, regularization techniques such as L1 and L2 norms, dropout methods, or data augmentation could be implemented. The high accuracy of the model underscores its potential for practical applications in automating tasks and improving text recognition accessibility. Future research should expand the model's application to various character sets, languages, and more complex conditions like diverse handwriting styles and noisy backgrounds.

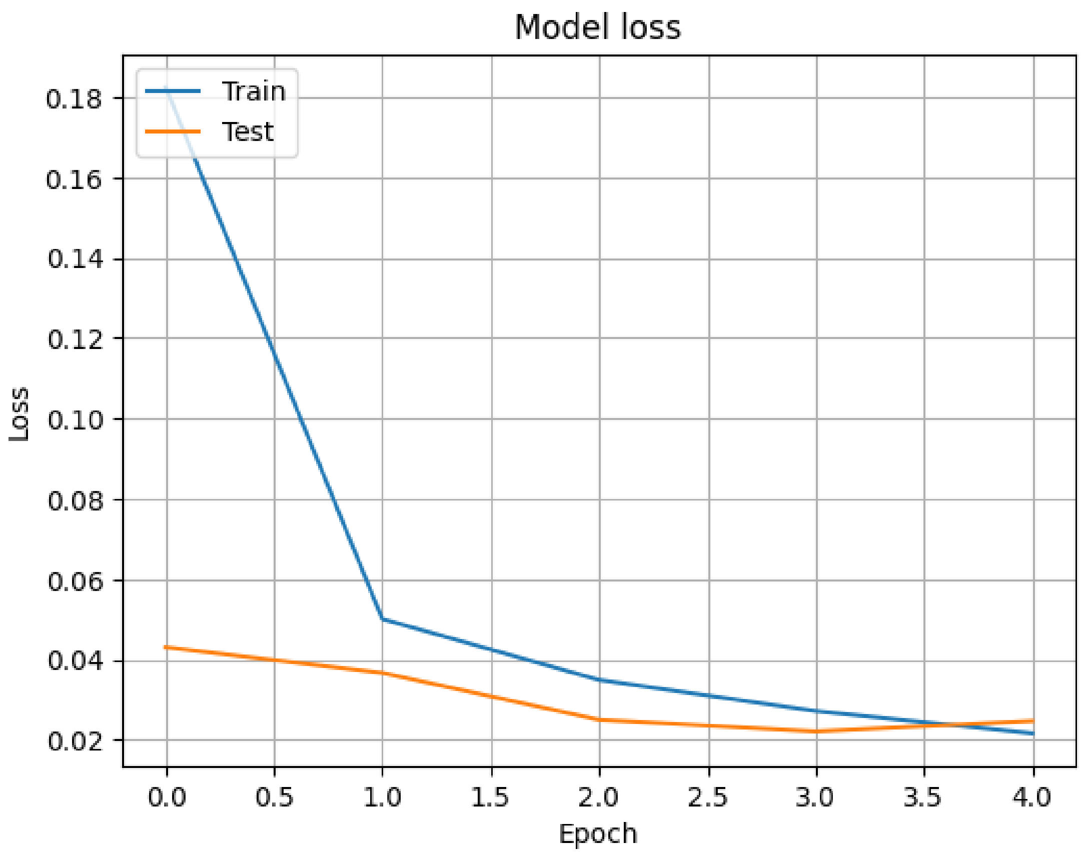

FIG.5.3

The model's excellent performance is evidenced by high accuracy and low loss (see FIG.5.3), making it well-suited for the character recognition task at hand. While Adam and its variants are preferred for their robust performance across various tasks, specific scenarios might benefit from other optimizers. Surpassing test accuracy over training accuracy occasionally could point to a regularization effect or the model's capability to better exploit certain test set patterns, emphasizing the importance of thorough model testing with recognition tasks.

The model's performance was scrutinized using various evaluation metrics, including accuracy, precision, and the confusion matrix. The F1-Score and recall were particularly emphasized, offering insights into the model's precision and sensitivity, respectively. An in-depth analysis of the confusion matrix shed light on the model's performance across different classes, revealing its strengths and pinpointing areas needing improvement. The model's performance across different optimizers was relatively uniform, with Adam and Adamax showing a slight edge over RMSprop, SGD, and Adadelta (see FIG.5.4). The latter two exhibited lower performance, potentially due to suboptimal learning rate settings or dataset characteristics that did not align with these optimizers' assumptions, suggesting the need for careful hyperparameter tuning.

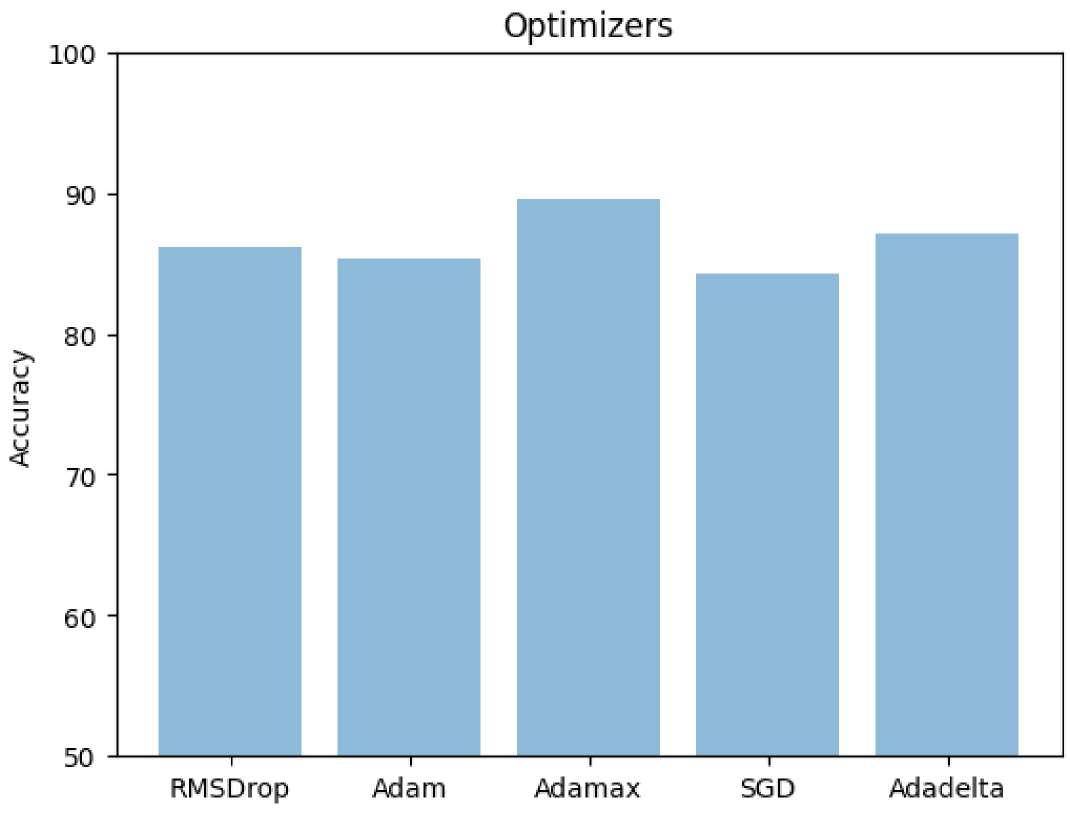

FIG.5.4

Out graphical analysis further illustrates the model's learning progression, with accuracy trends during training and validation phases providing a visual depiction of the model's adaptation and convergence over time. However, despite the model's robustness in digit recognition, challenges in accurately identifying certain letters were identified, indicating areas for future research and model enhancement.

We discovered this when we experimented with the test model and found that the predicted labels (such as '2', '5', and '7') did not match the expected alphabetical characters ('A', 'B', 'C') (see FIG.5.5 on the next page). Although character detection was successful, the prediction step did not accurately map to the correct alphabetical characters, which could be attributed to encoding faults, insufficient training dataset size, or issues in post-processing steps.

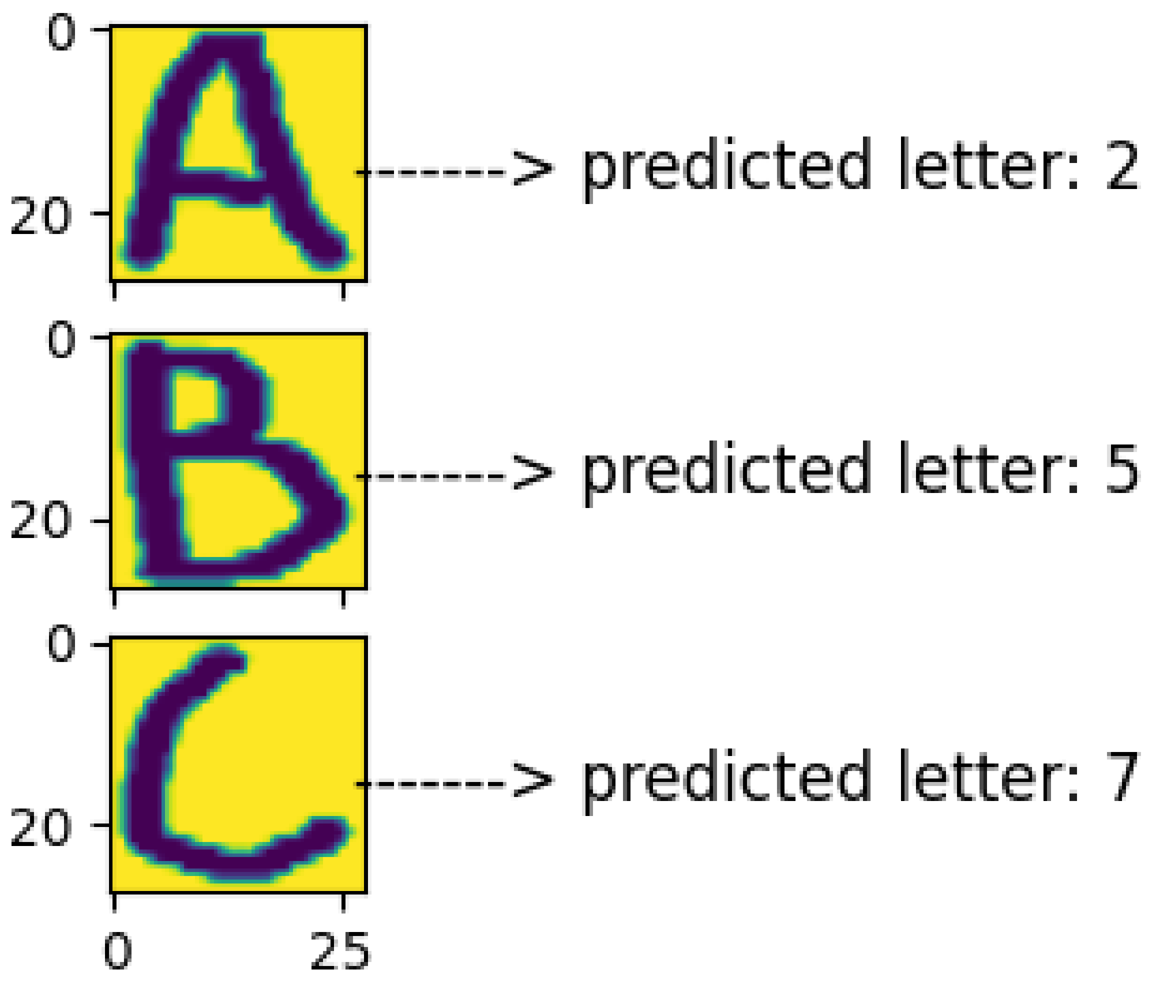

FIG.5.5

VI. CHALLENGES:

Data Standardization: When preprocessing a dataset, standardization methods including scaling, grayscale conversion, and normalization are applied. However, managing differences in writing styles and maintaining consistency throughout many preprocessing phases can be difficult, which affects the model's performance.

Data Augmentation: To improve model generalization, the dataset must be augmented to make it more resilient to noise and volatility. It can be difficult and necessitate experimentation to choose the right augmentation methods and parameters while preserving data integrity.

Class Imbalance: There are 62 classes in the dataset, and the number of samples in each class varies. It's critical to address class imbalances during model evaluation and training in order to avoid biased predictions and guarantee equitable performance rating for all classes.

Model Complexity: Careful design and optimization are needed to create a Convolutional Neural Network (CNN) architecture that can recognize a variety of handwritten characters. A major problem is to balance model complexity with available computational power and training time.

Hyperparameter tuning: To achieve high accuracy, it is essential to choose the best hyperparameters for the CNN model, such as learning rate, batch size, and number of epochs. It can take a lot of time and computing power to try until you find the ideal set of hyperparameters.

Evaluation Metrics: Considering the multi-class nature of the classification problem, selecting relevant evaluation metrics is crucial to evaluating the model's performance. It is difficult to determine which metrics—accuracy, precision, recall, or F1-score—best capture the model's performance in character recognition.

VII. CONTRIBUTIONS AND REFERENCES

This investigation into CNN-based character recognition contributes significantly to the fields of deep learning and optical character recognition (OCR), highlighting the potential of CNNs in advancing automated document processing and human-computer interaction. The methodologies and findings presented are supported and enriched by the following references.

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.