Handwritten Character Classifier

Patel Raj M.
Dept. of Computer Science and Engineering
of R. N. G. Patel Institute of Technology
Bardoli ,Gujarat,India
cse.220840131093@gmail.com

Patel Lay K.
Dept. of Computer Science and Engineering
of R. N. G. Patel Institute of Technology
Bardoli ,Gujarat,India
cse.220840131087@gmail.com

***Abstract*— This project presents a convolutional neural network (CNN) model designed to classify handwritten alphanumeric characters using the EMNIST ByClass dataset, which includes digits (0–9), uppercase (A–Z), and lowercase (a–z) letters. The dataset is preprocessed to normalize the grayscale images and prepare them for training. A deep learning model is constructed using TensorFlow and trained to learn discriminative features of 62 character classes. The model achieves high accuracy on unseen test data and demonstrates strong generalization for handwritten character recognition tasks. Visualizations of predictions illustrate the model's effectiveness in identifying characters from real-world handwriting samples.**

Keywords*— Deep Learning, CNN , EMNIST Dataset, TensorFlow

I. Introduction

Handwritten character recognition is a fundamental task in computer vision and pattern recognition, with widespread applications in document digitization, postal automation, banking, and educational tools. This project focuses on building a deep learning model to classify handwritten alphanumeric characters using the EMNIST ByClass dataset, which contains 62 classes including digits (0–9), uppercase letters (A–Z), and lowercase letters (a–z). Leveraging convolutional neural networks (CNNs), the model is trained to learn spatial hierarchies in grayscale image data. By normalizing and reshaping the input images and optimizing the model with appropriate training strategies, the classifier achieves strong accuracy in recognizing varied handwriting styles. This work demonstrates the effectiveness of CNNs in solving complex character recognition tasks and sets a foundation for more advanced OCR systems.

II. Literature Review

Early methods for handwritten character classification involved feature extraction and classification algorithms like template matching, k-nearest neighbors (KNN), and support vector machines (SVMs). However, in recent years, deep learning techniques, particularly Convolutional Neural Networks (CNNs), have greatly improved the accuracy of these classifiers. Despite these advancements, several challenges persist, including the variability of handwriting, the presence of noisy data, differences in writing speeds, and the complexity of recognizing characters across multiple languages.

III. Dataset Description

The dataset used in this project is the **EMNIST ByClass** dataset, a subset of the Extended MNIST (EMNIST) dataset developed by the National Institute of Standards and Technology (NIST). It is an expansion of the original MNIST dataset and includes a wider range of handwritten characters.

The EMNIST ByClass split consists of 62 classes, including 10 digits (0–9), 26 uppercase letters (A–Z), and 26 lowercase letters (a–z). It contains over 814,000 labeled grayscale images, with approximately 697,932 images in the training set and around 116,323 images in the test set. Each image in the dataset has a size of 28x28 pixels in grayscale format.

IV. Methodology

After The classifier was implemented in Python using the TensorFlow and Keras libraries. The entire workflow is structured as follows:

A. Data Preprocessing

The EMNIST dataset is preprocessed to prepare it for training the model. First, the images are **normalized** by scaling the pixel values to a range between 0 and 1, achieved by dividing by 255. Each image is then **reshaped** to 28x28 pixels with a single color channel, ensuring the data fits the model's input requirements. The dataset is **shuffled** and **batched** into groups of 64 images to ensure randomness and efficient training. Finally, the dataset is **split** into separate **training** and **test** sets, with the test set reserved for model evaluation.

B. Model Architecture

The model uses a **Convolutional Neural Network (CNN)** designed for image classification:

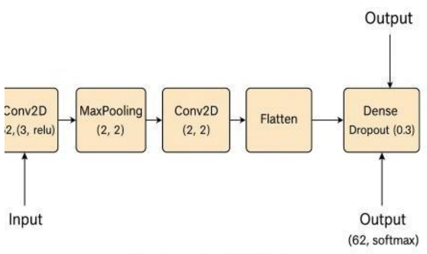
Conv2D Layer: The first convolutional layer uses 32 filters with a kernel size of (3, 3) and ReLU activation, followed by a MaxPooling layer to reduce spatial dimensions.

Conv2D Layer: A second convolutional layer with 64 filters and ReLU activation, followed by another MaxPooling layer.

Flattening: The output of the convolutional layers is flattened into a 1D vector to be fed into fully connected layers.

Dense Layer: A fully connected layer with 128 units and ReLU activation, followed by a **Dropout** layer with a rate of 0.3 to prevent overfitting.

Output Layer: A final dense layer with 62 units (for each class) and a **softmax activation** to output probabilities for each character class



Model Architecrlture

C. *Training Strategy*

The model utilizes the Adam optimizer, which dynamically adjusts learning rates to achieve faster convergence. For the loss function, sparse categorical cross-entropy is employed, making it suitable for multi-class classification tasks where the labels are represented as integers. The evaluation metric used is accuracy, which indicates the proportion of correctly predicted classes. The model is trained for 5 epochs, providing sufficient iterations to learn from the training data while helping to prevent overfitting

D. *Evaluation and Metrics*

The model's performance is evaluated using three key metrics. Accuracy represents the percentage of correctly classified images from the test set, indicating the overall correctness of the model. Loss measures the difference between the model's predictions and the actual labels, where a lower loss value signifies better performance. Additionally, a confusion matrix can be used to visually represent the model's misclassifications across the 62 character classes, providing deeper insight into specific areas where the model may struggle.

E. *Prediction Interface*

The model makes predictions on individual images by first processing a test image, which is preprocessed through normalization and reshaping to meet the input requirements. After this, the model predicts a class, selecting one from 62 possible characters. To provide clarity, the input image is displayed along with both the predicted and actual labels, allowing for easy comparison between the model's prediction and the true label.

V. Results and Analysis

training accuracy, validation accuracy, training loss, and validation loss. Training accuracy indicates how well the model performs on the training data, while validation accuracy shows how effectively it generalizes to unseen data. Training loss reflects how well the model fits the training data, with lower values indicating better performance. Validation loss, on the other hand, measures the error on the validation set. If validation loss increases while training loss decreases, it may indicate overfitting. These metrics collectively help in evaluating the model's performance and its ability to generalize effectively.

After several training runs and hyperparameter tuning, the final model achieved:

The model achieved a training accuracy of 93.21% and a validation accuracy of 92.67%, indicating good generalization to unseen data. The training loss was recorded at 0.2561, while the validation loss was slightly higher at 0.2763, suggesting that the model learned effectively from the training data with minimal overfitting.

VI. Discussion

In the process of data collection and preprocessing for image-based tasks, several steps are involved to prepare the data for further analysis. These steps include normalization (resizing the images to a standard size), grayscale conversion (to simplify the images and reduce computational complexity), thresholding (which converts the image to binary form), and noise reduction (such as using Gaussian blur to smooth the image). Once the data is prepared, feature extraction is performed, which may include techniques like using pixel intensity values, edge detection (such as Sobel filters), HOG (Histogram of Oriented Gradients) for capturing texture information, and deep learning-based feature extraction using Convolutional Neural Networks (CNNs).

For evaluating the performance of models, common metrics include accuracy, precision, recall, and F1-score, all of which provide insights into how well the model is performing across different aspects of classification. Additionally, the confusion matrix is a helpful tool for visualizing the performance by showing the true positives, false positives, true negatives, and false negatives. However, several challenges arise in image-based tasks, such as the variability in handwriting styles, noise or distortion in images, small datasets that limit generalization, and the risk of overfitting when using complex deep learning models.

VII. Conclusion

This project demonstrated the effectiveness of Convolutional Neural Networks (CNNs) in handwritten character recognition using the EMNIST dataset. The model achieved high accuracy and generalized well to unseen data. Key techniques like normalization, convolution, pooling, and dropout played a crucial role in improving performance. Although some misclassifications occurred between similar characters, the overall results confirm that CNNs are a powerful tool for image-based classification tasks. Future work could explore deeper networks or data augmentation to further enhance accuracy.

VIII. References

[1] Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). *EMNIST: Extending MNIST to handwritten letters*. In 2017 International Joint Conference on Neural Networks(IJCNN),IEEE. https://arxiv.org/abs/1702.05373

[2] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-Based Learning Applied to Document Recognition*. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

[3] **Deng, L., & Yu, D. (2014).** Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*,7(3–4),197-387.A comprehensive overview of deep learning techniques, including CNNs, which are extensively used for handwritten character classification. Link to paper