A Modular Pipeline for Handwritten Text Recognition and Prediction

Tammer Haddad, Joseph George

Northeastern University

December 10, 2025

AI Note

This report has been formalized by Claude AI using the prompt: "I made a draft for this paper, could you make it more formal and paper-appropriate?"
The raw draft report prior to formalization is available at ***https://github.com/jgeorge123george/CS4100_final_project/blob/main/Report/Latex/RAW_REPORT.pdf.***

Abstract

This project explores machine learning approaches to handwritten text recognition and prediction. We developed a modular system for digitizing handwritten text using a pipeline that integrates Tesseract OCR for character segmentation, a Convolutional Neural Network for character classification, and a Markov model for next-character prediction. Training data was sourced from the EMNIST and MNIST datasets for the CNN, and the Common Crawl corpus for the Markov model. Our CNN achieves 85.90% accuracy across 62 character classes on the EMNIST ByClass dataset and 99.21% on MNIST digits.

1 Introduction

Despite the prevalence of digital devices, handwritten notes remain a popular method for recording information in academic and professional settings. However, handwritten notes present significant challenges for organization and retrieval. Locating specific content within physical notes is time-consuming, and sharing or editing such notes requires manual transcription. Digitization addresses these limitations by enabling searchability, editability, and seamless distribution.

Traditional Optical Character Recognition (OCR) systems relied on rule-based algorithms and template matching to detect characters. With advances in deep learning, convolutional neural networks have substantially improved recognition accuracy, particularly for handwritten text where character forms vary considerably between writers. We build upon these developments by constructing an integrated pipeline that not only transcribes handwritten notes but also predicts subsequent characters, potentially improving transcription efficiency through intelligent autocompletion.

2 Related Work

Traditional OCR systems employed template matching, comparing character images against stored glyph prototypes from known fonts [5]. While effective on clean, constrained inputs, these rule-based approaches struggled with novel fonts and handwriting due to their reliance on extensive character template libraries.

LeCun et al. introduced LeNet-5, a convolutional neural network that learned to recognize handwritten digits end-to-end without manual feature engineering [4]. LeNet established the fundamental CNN architecture still in use today—convolution, pooling, and fully connected layers—achieving sub-1% error on MNIST. This architecture was deployed commercially for processing handwritten checks and inspired subsequent architectures including AlexNet and ResNet.

The MNIST dataset, comprising 70,000 handwritten digit images, became the standard benchmark for character recognition. Cohen et al. extended this with EMNIST in 2017, incorporating uppercase and lowercase letters to create a 62-class classification task while maintaining the 28×28 grayscale format [1]. Baseline results demonstrated 51.80% accuracy with linear classifiers and 77.57% with neural networks. Ciresan et al. achieved 88.12% using committees of deep CNNs [2], while recent architectures such as SpinalNet have exceeded 91% on balanced splits [3].

N-gram language models predict subsequent words or characters based on the preceding $n - 1$

elements, employing the Markov assumption that only recent context is relevant. This mathematical framework traces back to Markov's 1913 work on letter sequences, with Shannon applying it to English text in 1948. These models remain efficient for spelling correction and text prediction, particularly when trained on large corpora.

Modern end-to-end handwriting recognition systems combine CNNs with recurrent networks (LSTMs) and employ Connectionist Temporal Classification (CTC) loss for training [6]. CTC enables learning from unaligned data without character-level segmentation. Tesseract 4.0 adopted this LSTM-based approach, achieving substantial accuracy improvements over its legacy engine. State-of-the-art systems on benchmarks such as IAM now achieve character error rates below 5%.

Our approach maintains a modular pipeline: Tesseract for segmentation, a custom CNN for EMNIST classification, and a separate Markov model for prediction. This modularity enables independent optimization of each component and provides transparency into intermediate processing stages.

3 Preliminaries

Optical Character Recognition (OCR) refers to the process of converting images of text into machine-readable character sequences. In our pipeline, OCR serves specifically to segment input images into individual character images, which are subsequently processed by the CNN classifier.

Convolutional Neural Networks (CNNs) are deep learning architectures particularly suited for image processing. CNNs employ layers of learnable filters to detect hierarchical patterns—from simple edges in early layers to complex features such as character shapes in deeper layers. Figure 1 illustrates this progression, showing how initial layers detect edges while subsequent layers capture increasingly abstract features culminating in character-level predictions.

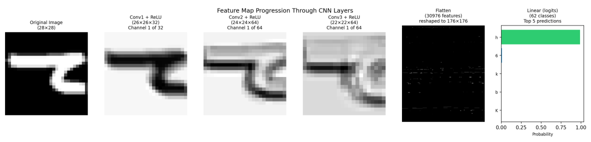
Feature Map Progression Through CNN Layers
Original Image (28×28)
Conv1 + ReLU (26×26×32) Channel 1 of 32
Conv2 + ReLU (24×24×64) Channel 1 of 64
Conv3 + ReLU (22×22×64) Channel 1 of 64
Flatten (30976 features) reshaped to 176×176
Linear (logits) (62 classes) Top 5 predictions
h
6
k
b
K
0.00
0.25
0.50
0.75
1.00
Probability

Figure 1: CNN feature progression across layers, demonstrating hierarchical feature extraction from edges to character-level representations.

Markov Models are statistical models that predict the next element in a sequence based solely on the current state or a fixed window of recent states. In our application, the Markov model predicts subsequent characters based on preceding character sequences, exploiting statistical regularities in natural language (e.g., given the sequence "good morni," the model assigns high probability to "n" and "g").

Our complete pipeline operates as follows: handwritten text images are first segmented into individual character images using OCR. Each character image is then classified by the CNN, producing predicted character sequences. These sequences are subsequently processed by the Markov model to predict likely continuations based on learned language patterns.

4 Data

4.1 Image Datasets

The MNIST dataset comprises 70,000 handwritten digit images (60,000 training, 10,000 testing) of digits 0–9. Each image is a 28×28 grayscale representation of a single digit.

The EMNIST (Extended MNIST) dataset augments MNIST with uppercase and lowercase letters, yielding 62 character classes. The ByClass split contains approximately 280,000 samples (240,000 training, 40,000 testing), maintaining the 28×28 grayscale format. Both datasets were obtained through the torchvision library.

4.2 Text Corpus

For Markov model training, we utilized the Common Crawl dataset, a large-scale web corpus providing diverse text samples. Data was accessed through the publicly available AWS S3 repository. Evaluation was performed on the C4 validation set, a held-out portion of Common Crawl designated for testing.

4.3 Preprocessing

The MNIST and EMNIST datasets required no additional preprocessing due to their standardized format. For Common Crawl, we discarded documents shorter than 100 characters, stripped leading and trailing whitespace, and concatenated documents with newline separators. Class imbalance was not a significant concern given the established balance of the image datasets.

5 Methodology

Our pipeline consists of four sequential stages: OCR segmentation, image preprocessing, CNN classification, and Markov prediction.

Character Segmentation. Tesseract OCR processes input images of handwritten text, segmenting them into individual character images.

This segmentation is necessary because our CNN is trained on single-character inputs.

Image Preprocessing. Segmented characters undergo several transformations: conversion to grayscale, color inversion, border padding to preserve aspect ratio, and resizing to 28×28 pixels. For EMNIST compatibility, images additionally require a 90-degree rotation and horizontal flip to match the dataset's native orientation.

CNN Classification. Preprocessed images are fed to our CNN, which outputs probability distributions over character classes. The architecture follows standard conventions: convolutional layers for feature extraction, pooling for spatial reduction, and fully connected layers for classification.

Markov Prediction. Character predictions from the CNN form sequences that are processed by the Markov model. The model, trained on Common Crawl text, predicts probable subsequent characters based on learned n-gram statistics.

The pipeline is implemented as a single Python script that orchestrates the CNN and Markov components. While these models currently operate independently, future work may integrate Markov predictions as auxiliary input to the CNN to improve classification accuracy through contextual information.

6 Experiments

6.1 Training Configuration

Both CNN models were trained using the Adam optimizer with learning rate 10^{-3}, batch size 32, and cross-entropy loss. The MNIST model was trained for 50 epochs, while the EMNIST model was trained for 20 epochs. Model selection was based on peak validation accuracy: epoch 40 for MNIST and epoch 3 for EMNIST. The EMNIST model exhibited signs of overfitting after epoch 3–4, with validation accuracy declining despite continued reduction in training loss (Figures 2 and 3).

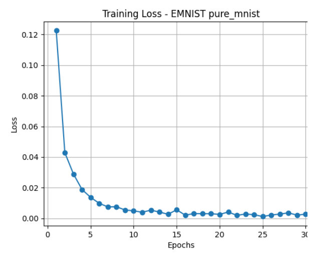

Figure 2: MNIST training loss.

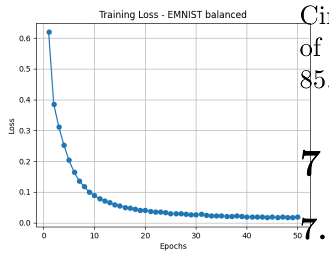

Figure 3: EMNIST training loss.

6.2 Computational Environment

Experiments were conducted across multiple environments: local machines equipped with NVIDIA GPUs and remote clusters utilizing V100 GPUs. The codebase was implemented in Python 3.8 using PyTorch and torchvision.

6.3 Evaluation Metrics

Primary evaluation employed classification accuracy on held-out test sets. Confusion matrices (Figure 4) were analyzed to identify systematic misclassification patterns.

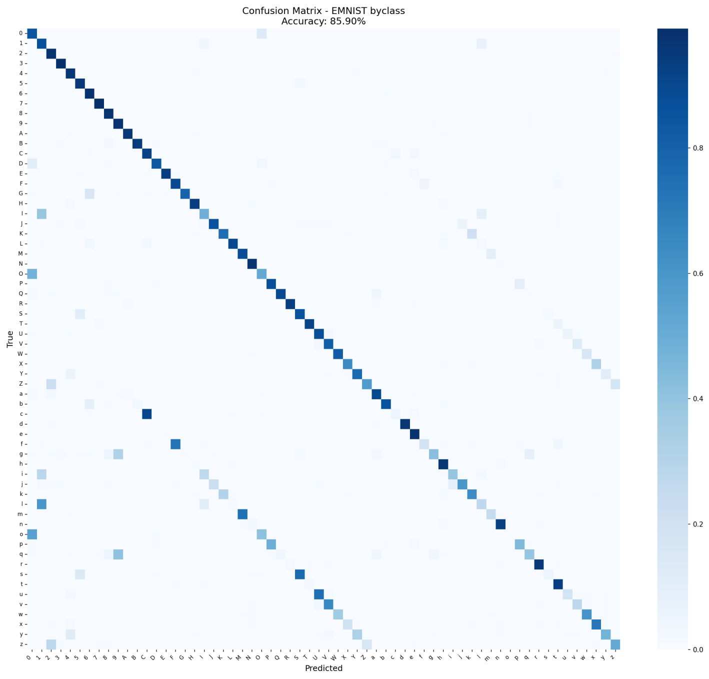

Figure 4: Confusion matrix for EMNIST ByClass classification.

6.4 Baselines

We compare our results against published baselines. The original EMNIST paper reports 51.80% accuracy with linear classifiers and 77.57% with a neural network containing 10,000 hidden units [1]. Ciresan et al. achieved 88.12% using a committee of seven deep CNNs [2]. Our single CNN achieves 85.90% on EMNIST ByClass.

7 Results

7.1 CNN Performance

On the EMNIST ByClass test set (116,323 samples), our model achieved 85.90% accuracy (99,921 correct classifications). On the MNIST test set (10,000 samples), accuracy reached 99.21% (9,921 correct). Figures 5 and 6 illustrate accuracy progression across training epochs.

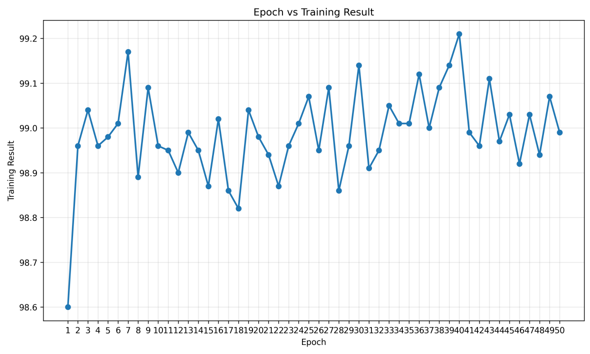

Figure 5: MNIST model accuracy over training epochs.

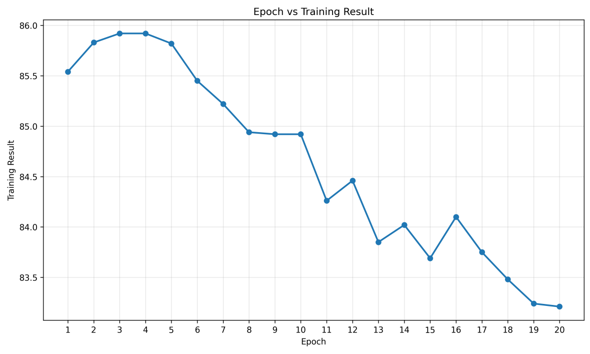

Figure 6: EMNIST ByClass model accuracy over training epochs.

Analysis of the confusion matrix reveals that the most frequent errors involve confusion between lowercase and uppercase variants of the same letter (e.g., 'O'/'o', 'C'/'c') and visually similar characters across classes (e.g., 'I'/'l'/'1'). These confusions are expected given the minimal visual distinctions, particularly regarding character size, which is normalized during preprocessing.

7.2 Markov Model Performance

The Markov model achieved the following prediction accuracies on the C4 validation set: Top-1: 59.84%, Top-3: 77.03%, Top-5: 82.43%. Performance varied by character frequency: spaces were predicted with 88.72% accuracy, the letter 'e' with 71.87%, while less frequent characters such as 'm' achieved only 40.12%.

8 Discussion

8.1 Limitations

The primary limitation of our current pipeline is error propagation: misclassification of a single character by the CNN corrupts the context for subsequent Markov predictions, potentially cascading into extended error sequences. This brittleness suggests that tighter integration between the CNN and Markov components could improve robustness.

8.2 Comparison to State-of-the-Art

Contemporary approaches employing transformer architectures and larger training corpora substantially outperform our results. SpinalNet achieves 95.88% on EMNIST Letters and 91.05% on the balanced split [3]. Our modular approach prioritizes interpretability and component-wise optimization over raw performance.

8.3 Future Work

Several extensions are planned. First, integrating Tesseract directly into the main codebase would streamline the pipeline. Second, developing a web application would enable direct image upload from users, potentially extending to video input via frame extraction. Third, and most significantly, incorporating Markov predictions as auxiliary features for CNN classification could leverage linguistic context to disambiguate visually similar characters.

8.4 Applications

The combination of CNN-based character recognition with Markov prediction has applications beyond note digitization. License plate recognition could incorporate jurisdiction-specific format priors. Historical document transcription could leverage period-appropriate language models. Any domain involving handwritten text stands to benefit from context-aware recognition systems.

9 Repository and Contributions

The complete project repository is available at `github.com/jgeorge123george/CS4100_final_project`.

9.1 Team Contributions

Tammer Haddad: Conceived the project. Extended the CNN from MNIST to full EMNIST classification. Developed the Markov model for text prediction. Integrated components into the unified pipeline. Authored the majority of this report.

Joseph George: Implemented the original MNIST CNN for digit recognition. Configured Tesseract OCR for character segmentation. Wrote the Abstract and Introduction sections; edited the report for formal presentation.

References

[1] Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. "EMNIST: Extending MNIST to Handwritten Letters." *arXiv preprint arXiv:1702.05373*, 2017.

[2] Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. "Convolutional Neural Network Committees for Handwritten Character Classification." *Proc. International Conference on Document Analysis and Recognition*, pp. 1135–1139, 2011.

[3] Kabir, H. M. D., et al. "SpinalNet: Deep Neural Network with Gradual Input." *IEEE Transactions on Artificial Intelligence*, vol. 4, no. 5, pp. 1165–1177, 2023.

[4] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] Eikvil, L. "Optical Character Recognition." Norsk Regnesentral Technical Report, 1993.

[6] Shi, B., Bai, X., and Yao, C. "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition." *IEEE TPAMI*, vol. 39, no. 11, pp. 2298–2304, 2017.