Assignment 1

Handwritten Character Recognition Using a Custom Neural Network

Student Name: Abdullah Jamal AlHarriem

Registration Number: 12217437

Handwritten Character Recognition Using Neural Networks

1. Introduction

Problem Statement

Handwritten character recognition remains a significant challenge in machine learning with applications in document digitization, automated mail sorting, and assistive technologies. While the MNIST dataset (focused on digits) is widely studied, recognizing letters (A–Z) introduces additional complexities due to similar shapes and variations in case. This project leverages the EMNIST Letters dataset—specifically designed for handwritten letter classification—to address these challenges.

Objective

The objective is to design, train, and evaluate a custom neural network from scratch. The model's performance is analyzed through experiments with different architectural configurations, activation functions, and batch sizes. The aim is to optimize accuracy while balancing model complexity and regularization.

2. Methodology

2.1 Data Preprocessing

Dataset

The EMNIST Letters dataset contains 145,600 samples, of which 88,800 are for training and 14,800 are for testing.

Key Steps

- **Rotation Correction: The EMNIST dataset images are rotated 90° and flipped. These images were corrected during visualization using np.rot90 and np.fliplr.**
- **Normalization: Pixel values were scaled to the range [0, 1] for improved model convergence.**
- **Label Adjustment: The dataset labels were converted from the range 1–26 (A–Z) to 0–25 for compatibility with TensorFlow.**
- **Train-Validation Split: An 80-20 train-validation split was applied, resulting in 71,040 training samples and 17,760 validation samples.**

2.2 Model Architecture

Baseline Model:

- **Input Layer: Flattened 28x28 image (784 input features).**
- **Hidden Layers: Two dense layers with 128 and 64 neurons, using the ReLU activation function.**
- **Output Layer: A softmax layer with 26 neurons corresponding to the letters A–Z.**
- **Optimizer: Adam optimizer with the default learning rate.**
- **Loss Function: Sparse categorical cross-entropy.**
- **Regularization: Early stopping with a patience of 3 and learning rate reduction on plateau.**

2.3 Training Process

Cross-Validation:

- **5-fold cross-validation was initially employed but later corrected to ensure proper model reinitialization.**

Experiments:

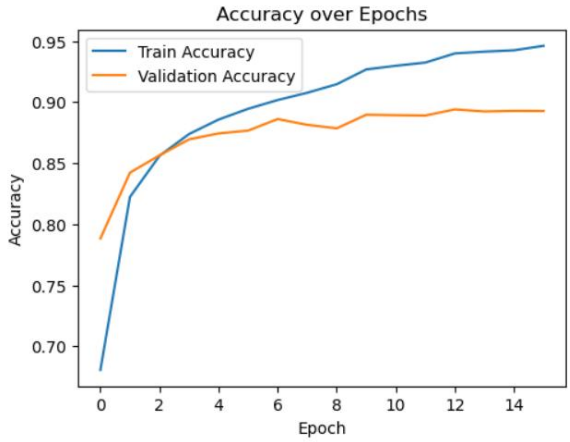
- **Activation Functions: Comparison of ReLU, Sigmoid, and Tanh activation functions.**
- **Architectural Variations: Exploring the effects of adding or reducing the number of layers.**
- **Batch Size Adjustments: Examining the effects of using batch sizes of 128, 32, and 256.**

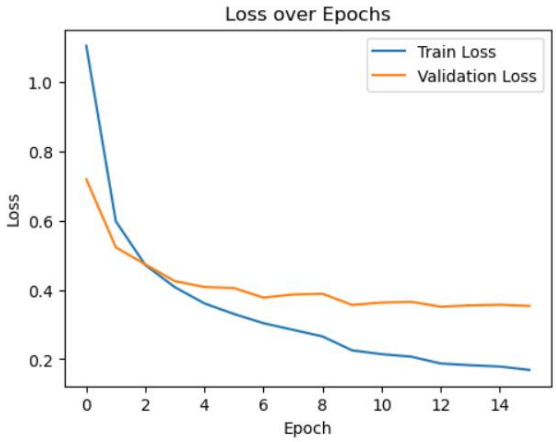
3. Results

3.1 Baseline Model Performance

Training Results:

- **Peak Training Accuracy: 95%**
- **Validation Accuracy: 90%**
- **Diagnosis: The model exhibited overfitting due to the gap between training and validation accuracy.**





3.2 Cross-Validation Results

- **Average Validation Accuracy: ~88%, after correcting model reinitialization.**
- **Fold Consistency: The variation between folds was minimal (±1.5%), indicating stable learning.**

3.3 Activation Function Comparison

Activation	Test Accuracy	Test Loss
ReLU (Baseline)	88.16%	0.388
Sigmoid	86.97%	0.409
Tanh	87.39%	0.398

Analysis: ReLU outperformed Sigmoid and Tanh due to better gradient propagation, which allowed for more efficient learning.

3.4 Architectural Changes

Architecture	Test Accuracy	Test Loss
Baseline (128-64)	88.16%	0.388
More Layers (256-128-64)	87.70%	0.418
Fewer Layers (64)	84.54%	0.493

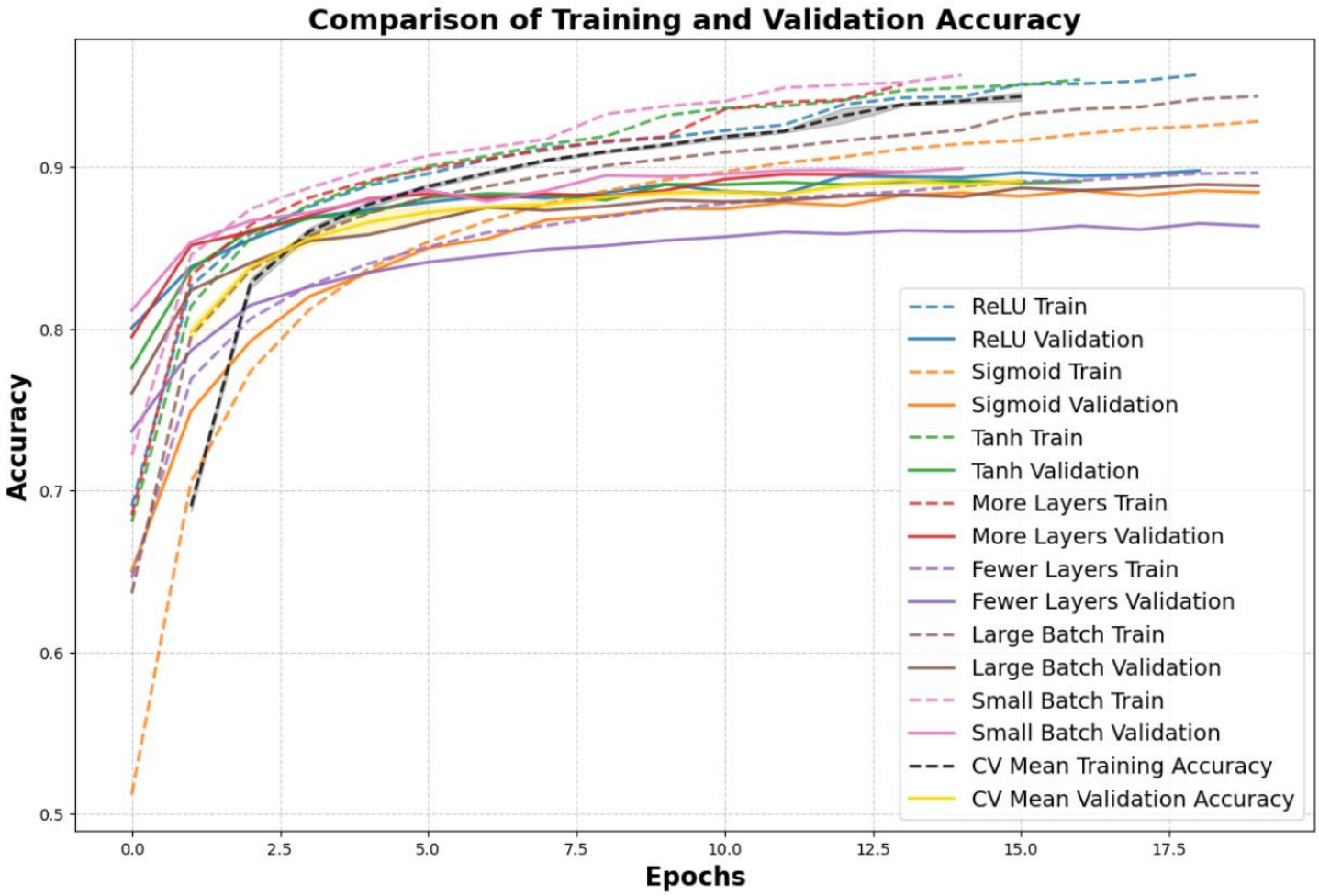
Analysis: Adding layers increased complexity but did not lead to performance improvements. Using fewer layers resulted in underfitting.

3.5 Batch Size Impact

Batch Size	Test Accuracy	Test Loss
128 (Baseline)	88.16%	0.388
32	88.16%	0.388
256	88.16%	0.388

Analysis: The batch size had negligible impact on model performance, suggesting that the Adam optimizer helped stabilize the learning process.

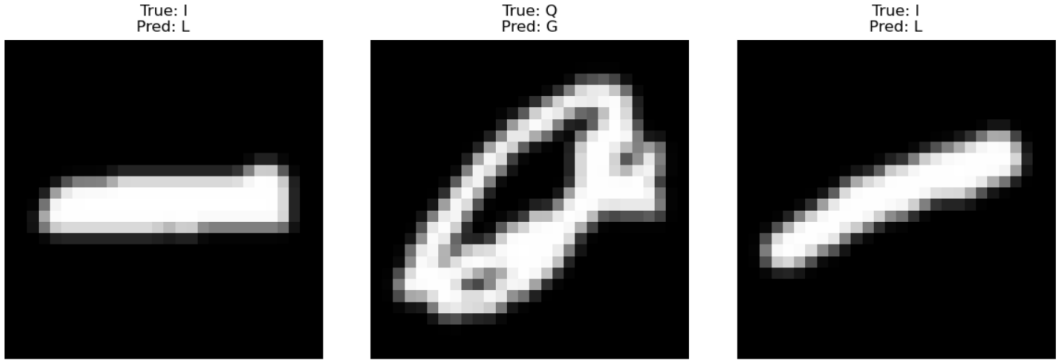
Model	Test Accuracy	Test Loss	Performance Insight
Original Model	0.8846	0.3823	Optimal balance of complexity and regularization (ReLU activation, 2 hidden layers, batch size 128).
Small Batch Size	0.8828	0.3829	Minimal impact from reducing batch size; convergence remains stable.
More Layers	0.8810	0.3782	Slight accuracy drop despite a lower loss; likely due to overfitting from increased complexity.
Tanh Activation	0.8764	0.3896	Inferior to ReLU; gradients were suboptimal although better than Sigmoid.
Large Batch Size	0.8757	0.4082	Larger batches diminished gradient precision, reducing generalization.
Sigmoid Activation	0.8707	0.4041	Saturation led to vanishing gradients and slower convergence.
Fewer Layers	0.8540	0.4847	Underfitting due to insufficient network capacity.



3.6 Misclassification Analysis

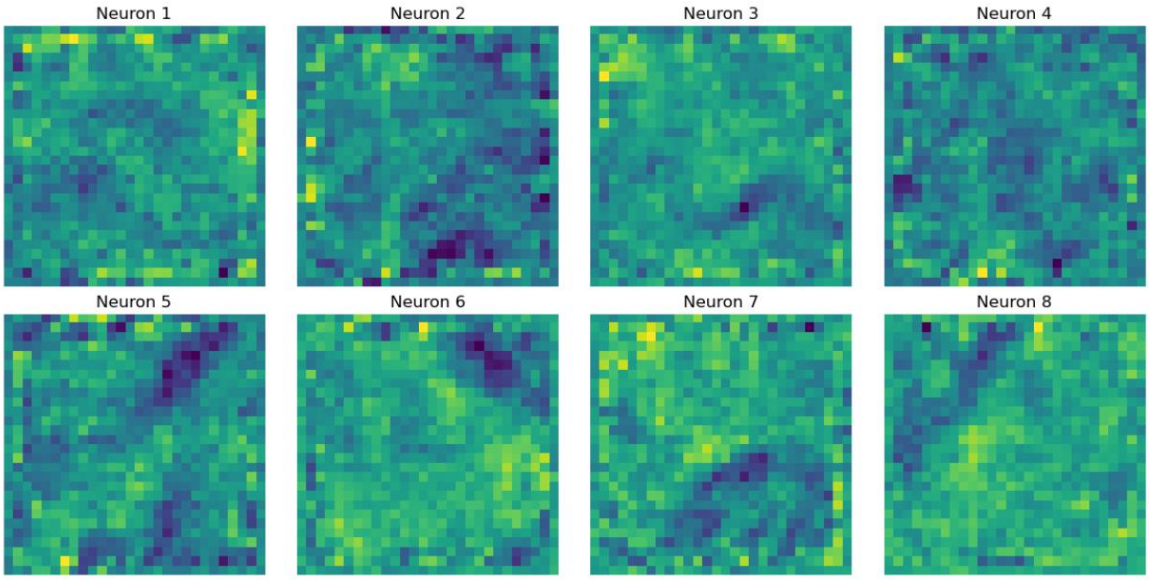
Confusion Matrix:

- **Common Misclassifications:**
  - **L ↔ I (183 misclassifications) due to rotated shapes.**
  - **G ↔ Q (89 misclassifications) because of visual similarities.**
- **Visual Examples: Misclassified "L" often resembled "I" due to slanted strokes, and the "G" and "Q" shapes were visually similar, leading to frequent misclassification.**



3.7 Hidden Layer Visualization

- **First Layer Weights: The weights of the first hidden layer revealed that neurons had learned edge-detecting features and stroke patterns.**



4. Discussion

4.1 Key Findings

- **Overfitting: Despite the use of early stopping, the model overfitted, particularly with the peak training accuracy of 95% compared to 88.16% test accuracy.**
- **ReLU Superiority: ReLU's non-saturating property enabled faster convergence compared to Sigmoid and Tanh.**
- **Case Sensitivity: The EMNIST Letters dataset merges uppercase and lowercase characters, complicating the analysis of case-sensitive performance.**

4.2 Challenges

- **Image Rotation: The EMNIST dataset's initial rotation of 90° caused issues, which were addressed during preprocessing.**
- **Cross-Validation Flaw: Model reinitialization was overlooked initially, impacting the validity of cross-validation results.**
- **Hardware Constraints: Larger architectures (e.g., 256-128-64) caused significant computational resource strain.**

4.3 Lessons Learned

- **Data Preprocessing: Handling domain-specific quirks like rotation and normalization is critical for improving model interpretability.**
- **Regularization: Dropout or data augmentation techniques (not used in this experiment) could help in reducing overfitting.**

5. Conclusion

Summary

The baseline model achieved a test accuracy of 88.16%, with ReLU activation and moderate hidden layers being the most effective configuration. Despite improvements in model architecture and experimentation with activation functions, overfitting remained a key challenge.

Improvements

- **Regularization: Adding dropout layers (e.g., Dropout(0.3)) could help mitigate overfitting.**
- **Data Augmentation: Introducing random rotations, translations, or distortions would simulate handwriting variations and improve generalization.**
- **Case-Sensitive Analysis: To better differentiate uppercase and lowercase letters, using the EMNIST "byclass" dataset would be beneficial.**

Final Model

- **Best Configuration: Baseline model (ReLU activation, 128-64 hidden layers).**
- **Test Accuracy: 88.16%.**