EMNIST Letter Classification Report

1.Introduction

The objective of this project is to classify handwritten letters using a neural network trained on the EMNIST dataset. The primary goals are to analyze model performance, identify misclassifications, and compare classification accuracy between difference approaches.

2. Methodology

A.Dataset:

EMNIST (Extended MNIST) - Letters split, consisting of handwritten characters.

B.Preprocessing:

- Loaded the dataset using torchvision.datasets.EMNIST.
- Normalized pixel values to the range [0,1] to ensure consistent model input.
- Reshaped images into a format suitable for ANN input.
- Converted labels into categorical format for classification.

C.Model Architecture:

Artificial Neural Network (ANN) consisting of:

- Input layer matching the image size (28x28 pixels, flattened to 784 features).
- Multiple fully connected layers with ReLU activation.
- Softmax activation in the output layer for multi-class classification.

D.Training Process:

- The model was trained for 15 epochs.
- Used early stopping to halt training when validation accuracy plateaued.
- Applied cross-validation to enhance model performance
- Applied cross-entropy loss and Adam optimizer.
- Evaluated on validation and test sets using accuracy metrics.

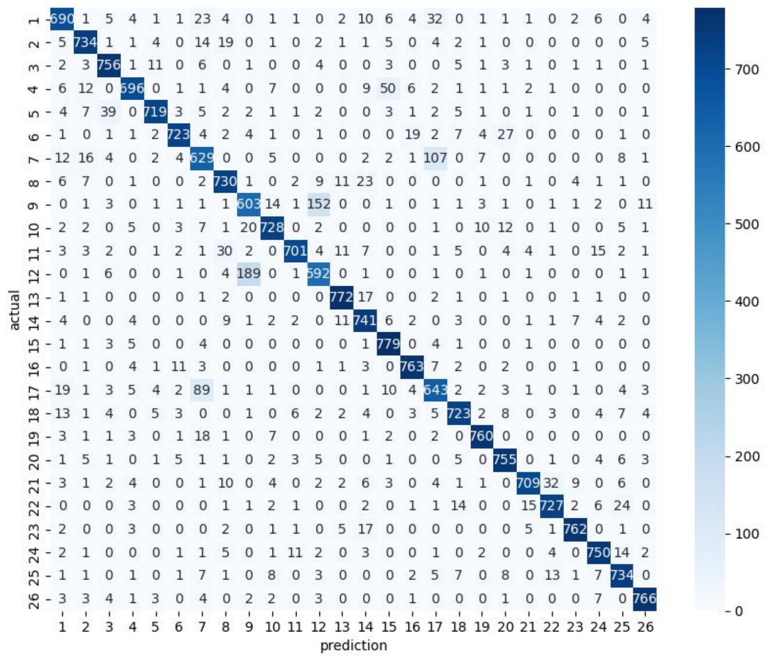
3. Results:

Performance metrics:

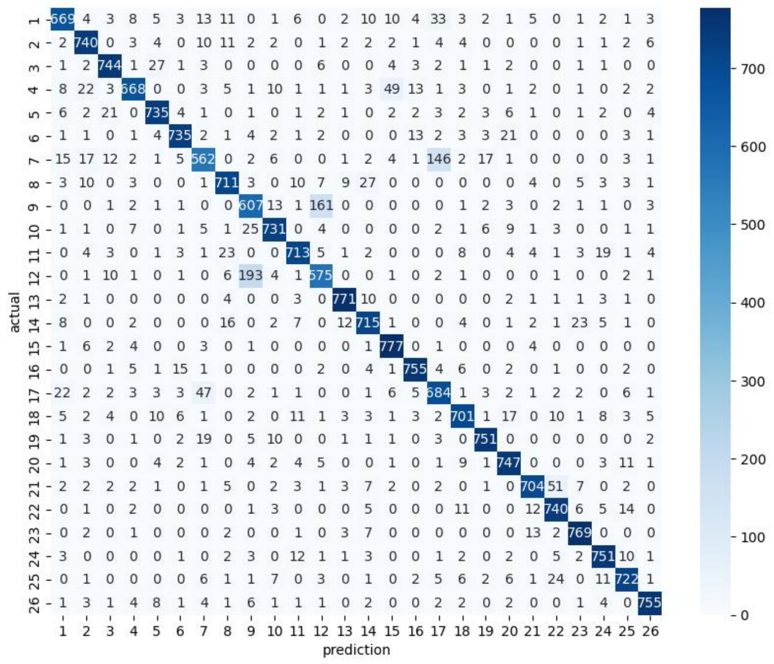
Validation and test accuracy:the values of both was almost the same around 90% accuracy

Confusion matrix: Used to compare predicted values and actual values and to identify the most misclassified values

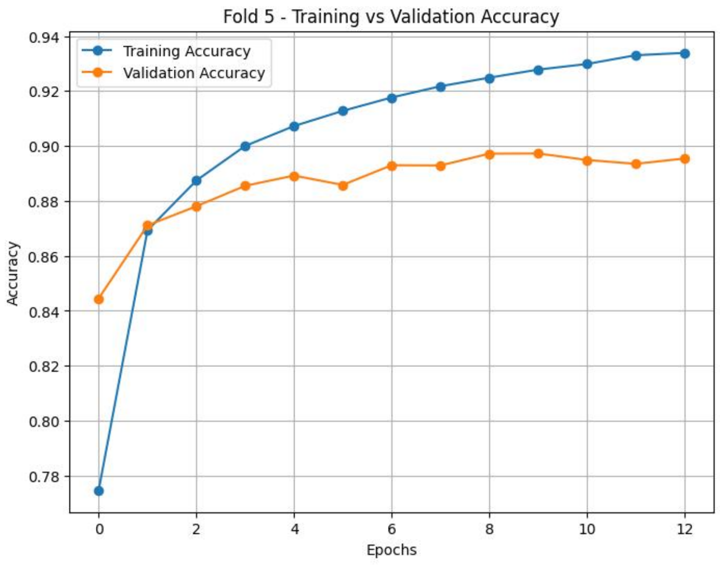
Using RELU activation function



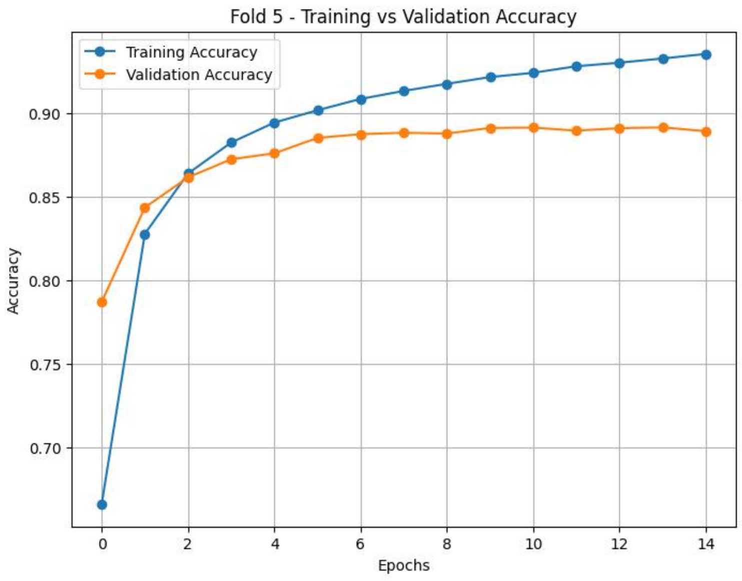
Using sigmod activation function



Visualizations and Analysis:
For relu



For sigmoid



4. Discussion:

Throughout this project, different approaches were explored to optimize model performance. Initially, both ReLU and Sigmoid activation functions were tested, yielding similar accuracy values. However, the training curve with Sigmoid activation was smoother and more stable. The number of hidden layers was adjusted from two to three and finally to one, but accuracy remained largely unchanged, suggesting that increasing complexity did not significantly impact performance.

Batch size was another important hyperparameter that affected training dynamics. Larger batch sizes resulted in faster convergence but sometimes led to less generalization, while smaller batch sizes provided more stability but increased training time. Tuning the batch size was crucial in balancing performance and efficiency.

The most challenging aspect of the project was the computational time required, especially when performing cross-validation while tuning hyperparameters. This process significantly extended training duration, making experimentation with different architectures more time-consuming. Despite these challenges, valuable insights were gained into the effects of activation functions, model complexity, and hyperparameter tuning on classification performance.

5. Conclusion

The results remained consistent across different approaches, indicating that while activation functions and the number of layers influenced training dynamics, they did not drastically alter final accuracy. Future work could explore more advanced techniques such as convolutional neural networks (CNNs) to better capture spatial patterns in letters. Additionally, fine-tuning hyperparameters such as learning rate and dropout rate, along with using data augmentation, could further enhance classification accuracy and model robustness.