Fadhla Mohamed Mutua

SM3201434

1. Introduction

This experiment aims to explore and compare the performance of two neural network architectures (Fully Connected Neural Network - FCNN and Convolutional Neural Network - CNN) on the KMNIST dataset. The focus is on tuning hyperparameters to improve model performance.

We will train each model using different optimizers (Adam, SGD, RMSprop) and evaluate their performance using accuracy and loss metrics. The best-performing model will be selected based on validation accuracy, and its confusion matrix will be analyzed.

2. Data Eploration

We use the KMNIST dataset, containing 28x28 pixel grayscale images of handwritten non-alpha-numeric characters. The dataset is divided into a training set (60,000 images) and a test set (10,000 images).

2.1 Data Preprocessing

The images are normalized using the ToTensor() transformation, converting them into PyTorch tensors. The dataset is then split into training and validation subsets using random_split(), with a fixed seed of 0 for reproducibility. To optimize training, the data is loaded in batches of 64, with shuffling enabled for each epoch to prevent memorization and ensure generalization.

2.2 Model Architectures

`Note: For both architectures, the number of training epochs was set to 10 to accommodate the slower convergence of the SGD optimizer and FCNN architecture (two layers)`

2.2.1

The `FCNN` consists of three fully connected layers with `ReLU` activation functions and dropout regularization. The model architecture is as follows: Input layer (784), Hidden layer 1 (128 neurons), Hidden layer 2 (64 neurons), Output layer (10 neurons for 10 classes).

2.2.2

The `CNN` model uses two convolutional layers followed by max-pooling layers. After flattening, two fully connected layers are used. The architecture is: `Conv2D` (1, 32, 3x3), `MaxPool2D` (2x2), `Conv2D` (32, 64, 3x3), `MaxPool2D` (2x2), Fully connected layers (128 neurons).

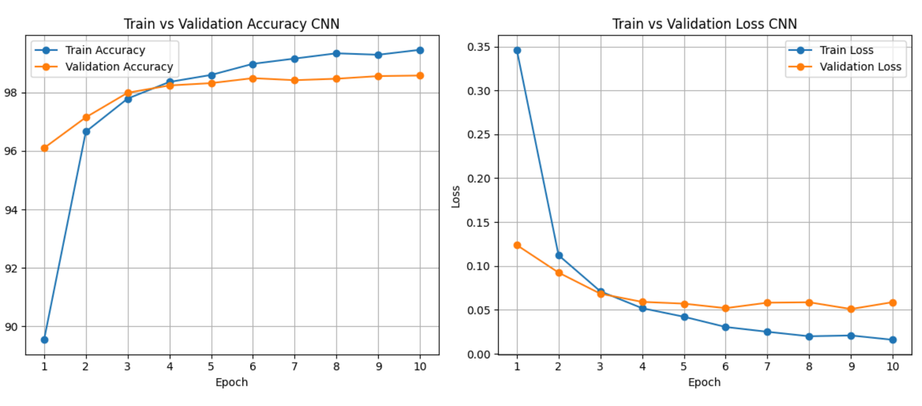
2.3 Training

Three optimizers are tested: `Adam`, `SGD`, and `RMSprop`. The learning rate for `Adam`, acquired through grid search is set to 0.001, for `SGD` it's 0.1, and for `RMSprop` it's 0.001. `Batch size` is set to 64, and `dropout` 0.2 . `Weight decay` is set to 10^{-5} for all but `SDG` which is set to 0.

Each model is trained for 10 epochs. The training subset is used for model training, while the validation set is used to evaluate performance during training.

4 Results for Adam model

4.1 CNN Accuracy and Loss Graph

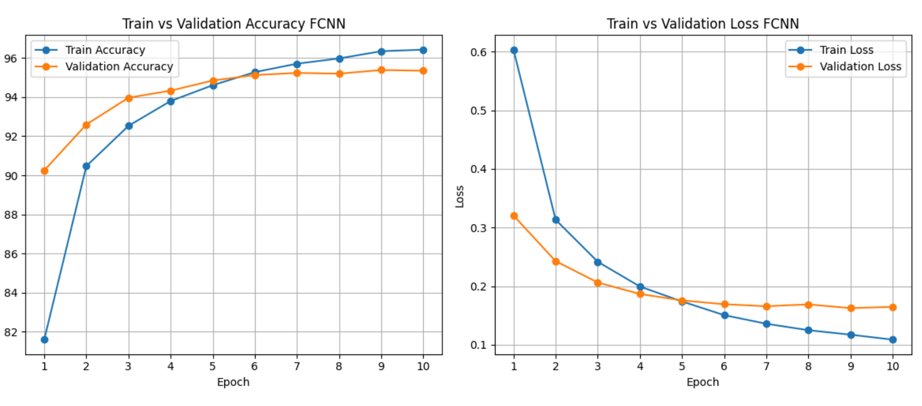


Looking at the graph, we observe that during the early epochs (1-2), the model's accuracy is relatively low, and the loss is high. This is expected as the model is just beginning to generalize, and its learning process is still in its early stages. At this point, the train accuracy is noticeably lower than the validation accuracy, and the train loss is higher than the validation loss. This suggests underfitting, as the model has not yet learned to properly fit the data.

In the mid-epochs (3-4), both the training and validation accuracies have increased significantly, and are nearly in alignment with each other. Additionally, the training and validation losses have dropped and are now comparable, indicating that the model is achieving a good balance between fitting the data and generalizing. This period typically reflects the model's optimal learning phase, where it captures patterns in the data without overfitting.

However, in the later epochs (5+), we notice a shift towards overfitting. Despite maintaining high accuracy and low loss on the training set, the model starts to show signs of memorizing the training data rather than generalizing well. This is indicated by the train accuracy being higher than the validation accuracy and the train loss being lower than the validation loss. As the epochs progress, particularly at the 7th, 8th, and 10th epochs, we see a rise in validation loss, signaling that the model is losing its ability to generalize and is instead overfitting to the training data.

4.2 FCNN Accuracy and Loss Graph

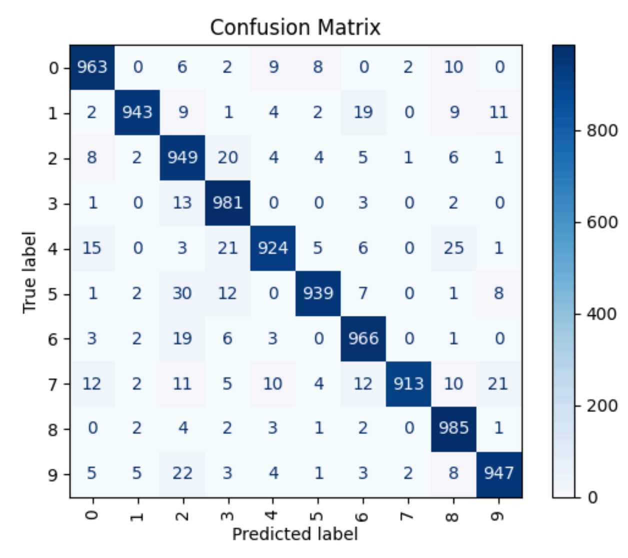


Looking at the graph, we observe that during the early epochs (1-4), the model's accuracy is relatively low, and the loss is high. This is expected as the model is just beginning to generalize, and its learning process is still in its early stages. At this point, the train accuracy is noticeably lower than the validation accuracy, and the train loss is higher than the validation loss. This suggests underfitting, as the model has not yet learned to properly fit the data.

In the mid-epochs (5-6), both the training and validation accuracies have increased significantly, and are nearly in alignment with each other. Additionally, the training and validation losses have dropped and are now comparable, indicating that the model is achieving a good balance between fitting the data and generalizing. This period typically reflects the model's optimal learning phase, where it captures patterns in the data without overfitting.

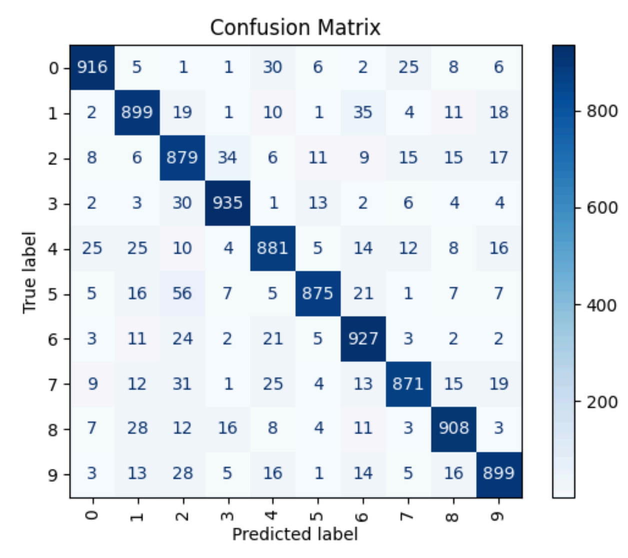
However, in the later epochs (7+), we notice a shift towards overfitting. Despite maintaining high accuracy and low loss on the training set, the model starts to show signs of memorizing the training data rather than generalizing well. This is indicated by the train accuracy being higher than the validation accuracy and the train loss being lower than the validation loss. As the epochs progress, particularly at the 8th epochs, we see a rise in validation loss, signaling that the model is losing its ability to generalize and is instead overfitting to the training data.

4.3 CNN Confusion Matrix



We note a test accuraccy of `96.07`

4.4 FCNN Confusion Matrix



We note a test accuraccy of `89.62`

5 Discussion

- The train set is used to teach the model by adjusting its parameters. The validation set evaluates the model during training, helping monitor generalization and tune hyperparameters. It also detects overfitting or underfitting. The test set is used after training to provide an unbiased measure of the model's performance on unseen data.
- Accuracy while being a common metric for classification tasks, has limitations when applied to datasets like KMNIST. Although it provides an overall performance snapshots, accuracy might not capture the model's struggles with visually similar characters in KMNIST.
- In cases where certain characters are harder to distinguish, accuracy can be misleading, as a model might perform well on simpler classes but fail on more complex ones. For a more better evaluation, additional metrics are needed to better assess how well the model handles all classes, especially the more challenging ones.
- Thus, while accuracy is useful, it shouldn't be the sole metric when evaluating KMNIST type models.