CSE 472: Assignment on Feed-Forward Neural Networks

1905060: Sardar Md. Saffat Zabin

November 18, 2024

1 Instructions for Running the Code

1.1 Notebook Structure

The notebook is structured as follows:

1. Dense Layer
2. Activation Function: ReLU
3. Normalization: Batch Normalization
4. Regularization: Dropout
5. Optimizer: Adam
6. Regression Function: Softmax
7. Categorical Cross Entropy Loss
8. Neural Network
9. Load Dataset
10. One-hot encoding
11. Neural Network Initialization
12. Training
13. Plot accuracy and loss curves
14. Save the model
15. Testing

Sections 1 through 8 contains the implementations of the various classes required for this assignment. Section 10 contains code that loads the dataset. Section 11 contains the definitions of the architectures that were used for training. Section 12 contains the training code. Section 13 contains the testing code.

1.2 Running the Code

For training the neural network(s), the code cells from Sections 1 through 12 should be run. This would save the best model as a Pickle file.

For testing the neural network(s), the code cells from Sections 1 through 11 and 13 should be run. This would load the model from the Pickle file and test it on the test dataset.

2 Architectures

2.1 Architecture 1

2.2 Architecture 1

```
DenseLayer(28 * 28, 32),
ReLU(),
Dropout(0.3),
BatchNormalization(32, 0.9, 1e-5),
DenseLayer(32, 64),
ReLU(),
Dropout(0.3),
BatchNormalization(64, 0.9, 1e-5),
DenseLayer(64, 64),
ReLU(),
       Dropout(0.3),
BatchNormalization(64, 0.9, 1e-5),
DenseLayer(64, 32),
       ReLU(),
       Dropout(0.3),
BatchNormalization(32, 0.9, 1e-5),
       DenseLayer(32, 10)
Softmax()
```

2.3 Architecture 2

```
DenseLayer(28 * 28, 256),
ReLU(),
Dropout(0.1),
BatchNormalization(256, 0.9, 1e-5),
DenseLayer(256, 128),
ReLU(),
Dropout(0.1),
BatchNormalization(128, 0.9, 1e-5),
DenseLayer(128, 64),
ReLU(),
BatchNormalization(64, 0.9, 1e-5),
DenseLayer(64, 32),
ReLU(),
BatchNormalization(32, 0.9, 1e-5),
DenseLayer(32, 10),
```

```
Softmax()
```

2.4 Architecture 3

```
DenseLayer(28 * 28, 32),
ReLU(),
Dropout(0.1),
BatchNormalization(32, 0.9, 1e-5),
DenseLayer(32, 64),
ReLU(),
Dropout(0.1),
BatchNormalization(64, 0.9, 1e-5),
DenseLayer(64, 64),
ReLU(),
BatchNormalization(64, 0.9, 1e-5),
DenseLayer(64, 64),
ReLU(),
BatchNormalization(64, 0.9, 1e-5),
DenseLayer(64, 48),
ReLU(),
BatchNormalization(64, 0.9, 1e-5),
DenseLayer(48, 32),
ReLU(),
BatchNormalization(64, 0.9, 1e-5),
DenseLayer(32, 10),
Softmax()
```

3 Training

The neural networks were trained on the Fashion MNIST dataset. The dataset was split into a training set and a testing set, with 60000 and 10000 samples respectively. The training set was further split into a training set and a validation set, with 15% of the training set being used for validation.

The neural networks were trained using the Adam optimizer with learning rates of 0.005, 0.004, 0.002 and 0.001. The neural networks were trained for 26 epochs.

4 Results

4.1 Training Loss

4.1.1 Architecture 1

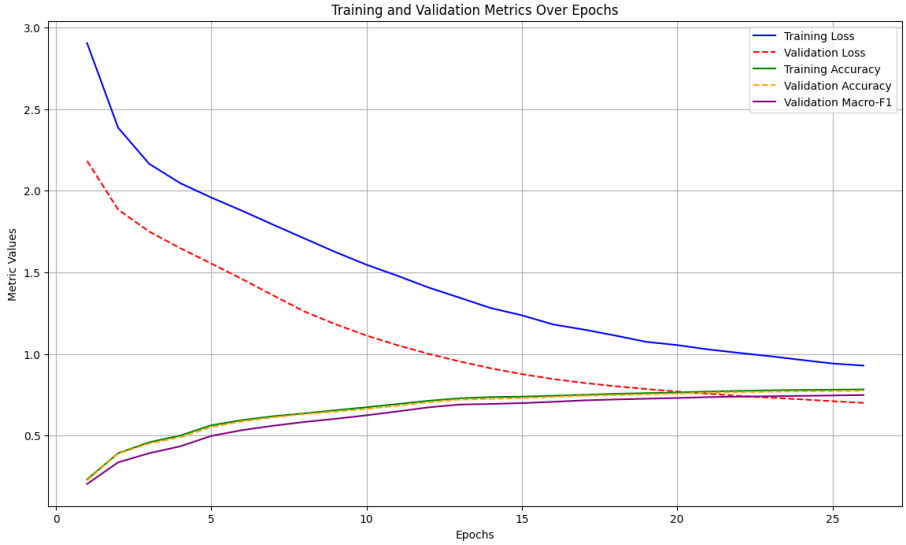

Figure 1: Training Losses and Accuracy

4.1.2 Architecture 2

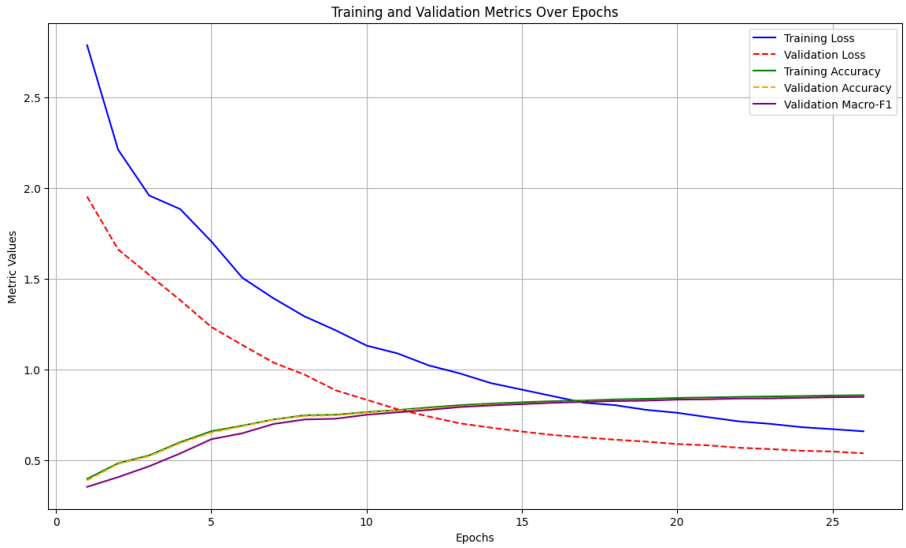

Figure 2: Training Losses and Accuracy

4.1.3 Architecture 3

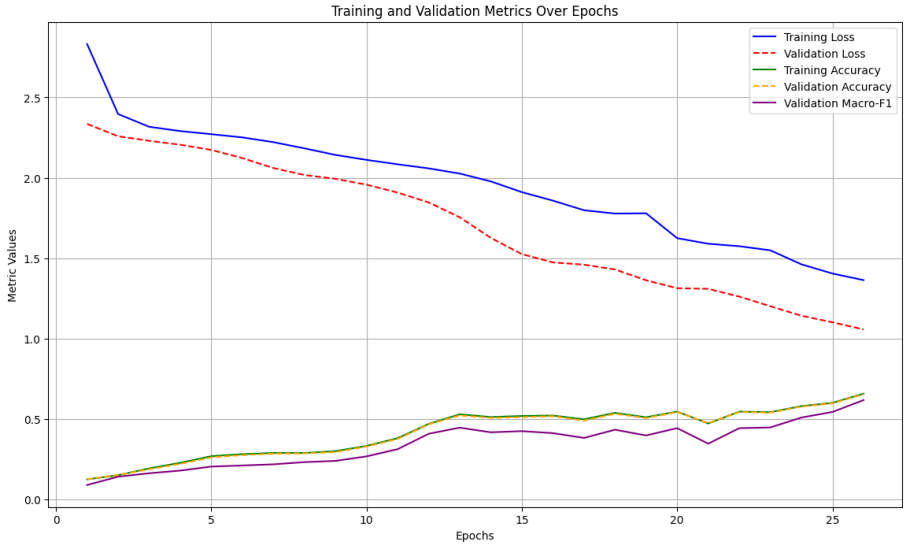

Figure 3: Training Losses and Accuracy

The best model was selected based on the validation Macro F1 score. The model was then tested on the test dataset. The best model was Architecture 2 with a learning rate of 0.005. The Macro F1 score of this model on the validation set was 0.8629.

Metric	**Score on Test Dataset**
Accuracy	0.8509
Loss	0.7158
Macro F1 Score	0.8510

5 Confusion Matrices

5.1 Architecture 1

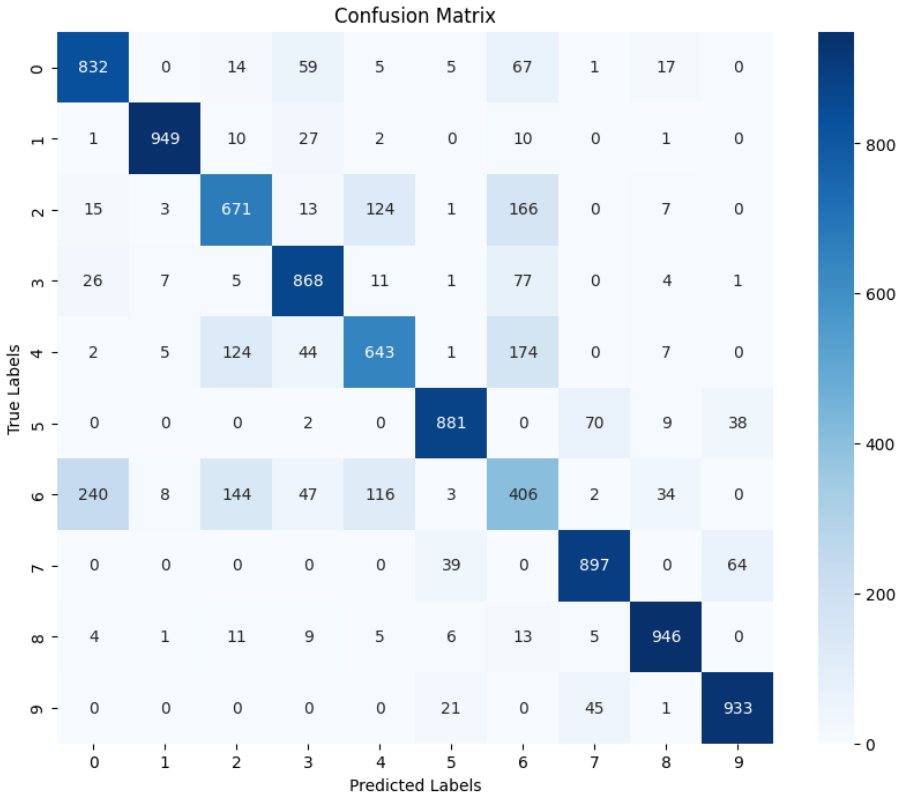

Figure 4: Confusion Matrix

5.2 Architecture 2

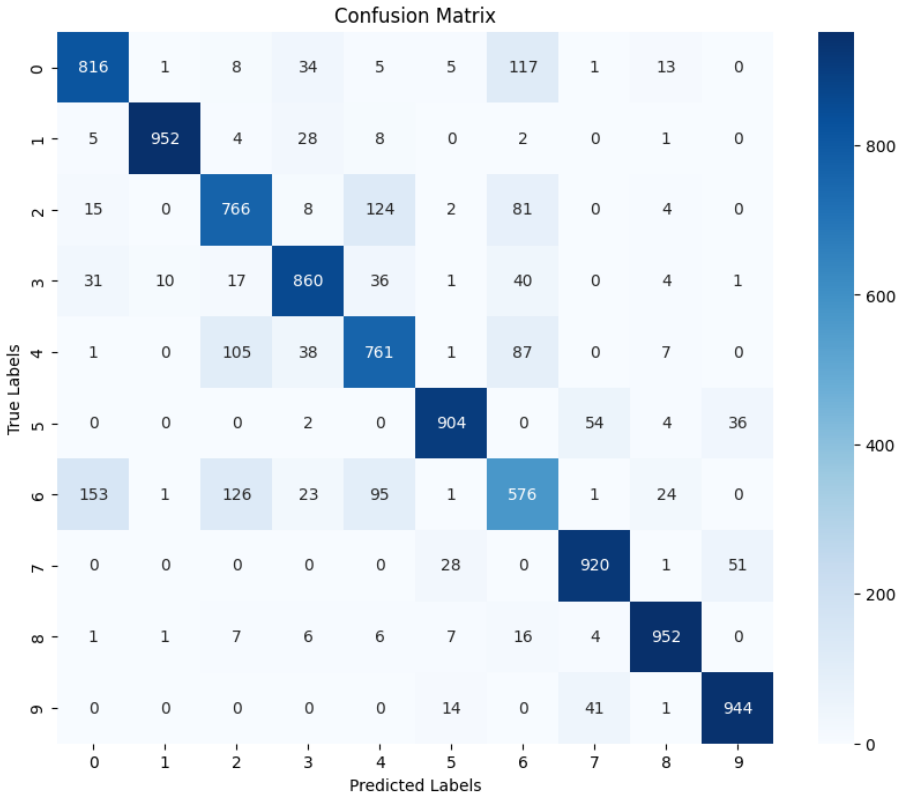

Figure 5: Confusion Matrix

5.3 Architecture 3

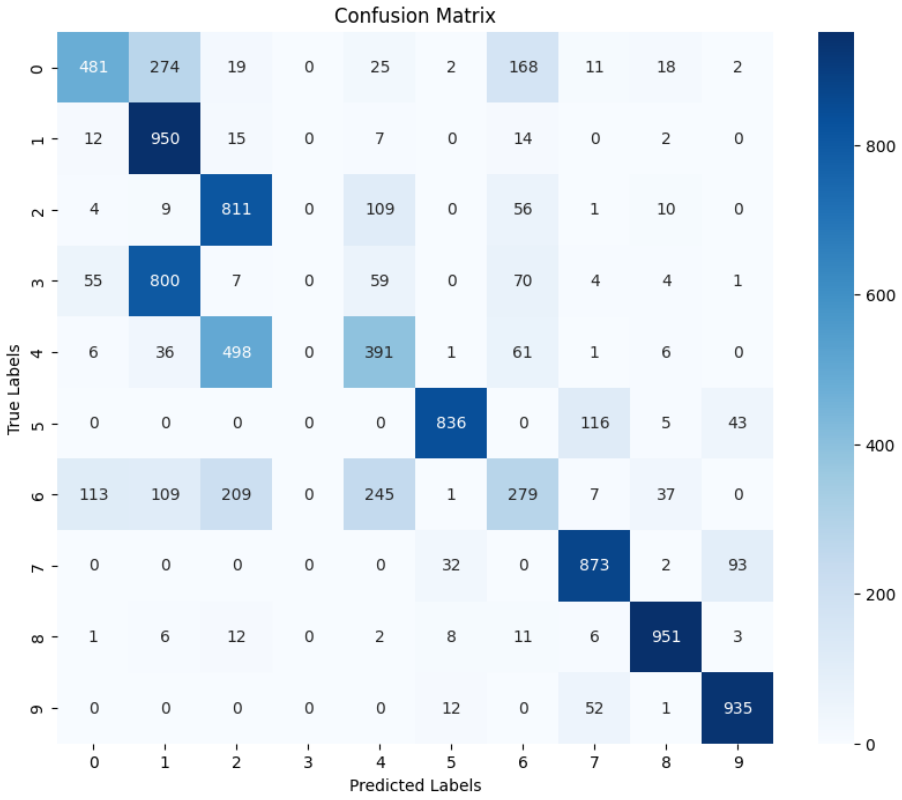

Figure 6: Confusion Matrix