Assignment No# 1

(Deep Learning)

21L-5617

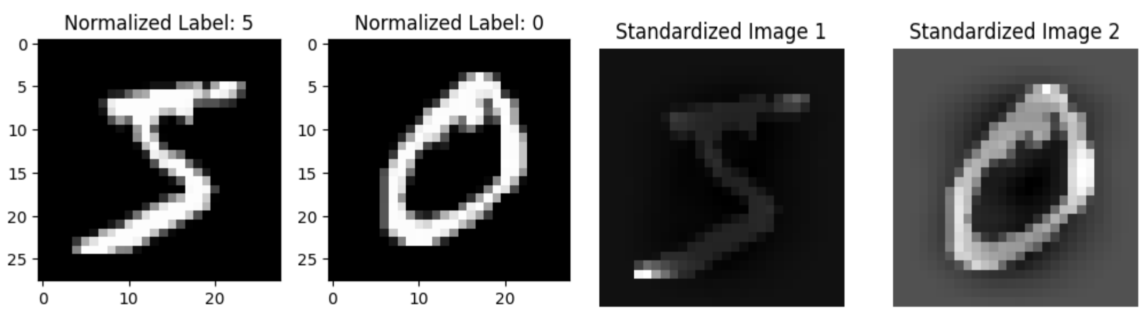
Muhammad Ahmad

BS (DS) 7A

Question 1: Logistic Regression

preprocessing

First, load the data and visualize it by displaying two images from the dataset. Then, apply preprocessing steps, including normalization, standard scaling, and PCA, and observe the changes after each step.



Part (a)

In Part A, implement a general Logistic Regression model that works with both TensorFlow and Scikit-learn. Run the model on both the preprocessed images and the original (non-preprocessed) images to compare performance

Metric	Without Preprocessing	With Preprocessing
Accuracy	0.9204	0.9265
Precision	0.9203	0.9264
Recall	0.9204	0.9265
F1 Score	0.9202	0.9264

This table clearly shows the improvement in model performance after preprocessing the data

Part (b)

The model was trained with **SGD**, achieving moderate results. **RMSprop** with 10 epochs showed improvement, but using 100 epochs led to a slight performance drop. **Adam** performed similarly to RMSprop with 10 epochs and slightly improved with 100 epochs. Finally, **Nesterov Accelerated Gradient** with 100 epochs gave the best performance.

Optimizer	**Epochs**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
SGD	**-**	0.9239	0.9237	0.9239	0.9237
RMSprop	10	0.9263	0.9263	0.9263	0.9261
RMSprop	100	0.9250	0.9250	0.9250	0.9249
Adam	10	0.9250	0.9249	0.9250	0.9248
Adam	100	0.9256	0.9254	0.9256	0.9254
Nesterov Accelerated Gradient	100	0.9256	0.9269	0.9270	0.9268

This table provides a clear comparison of the different optimizers and how the model's performance varies across different epochs and optimization methods.

Part (c)

I used 133 rounds for hyperparameter tuning in Part C, focusing on optimizing the learning rate, number of epochs, and batch size. After tuning, the best hyperparameters were found to be:

- **Learning Rate**: 0.0001976
- **Number of Epochs**: 100
- **Batch Size**: 16

Using these values, the model achieved significantly improved performance with the following results:

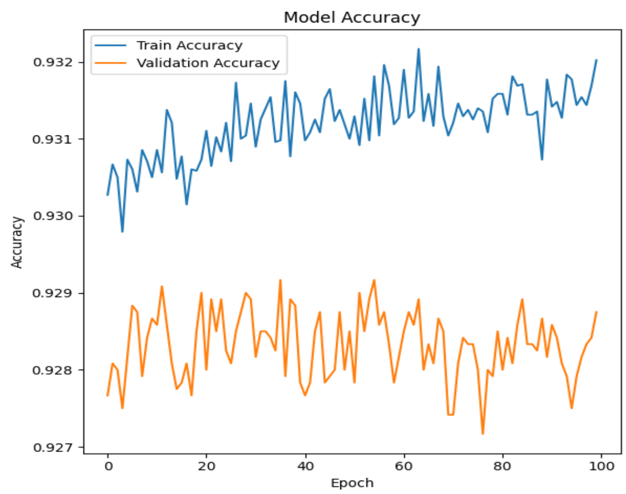
- **Model Accuracy**: 0.9639
- **Model Precision**: 0.9639
- **Model Recall**: 0.9639
- **Model F1 Score**: 0.9639

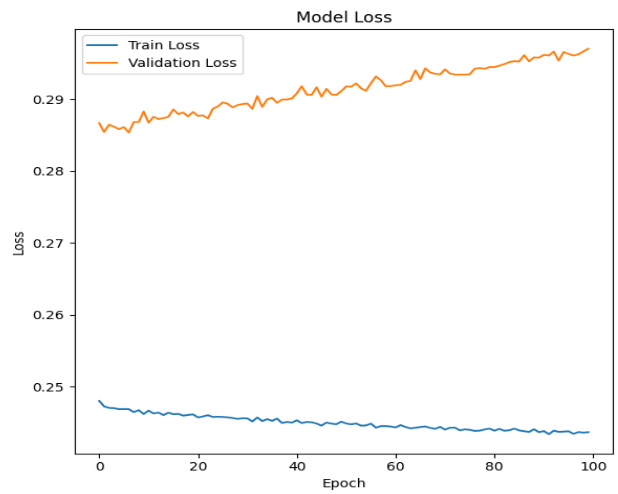
Question 2: Batch Normalization

When using the previously trained batch size without batch normalization, the model achieved the following performance:

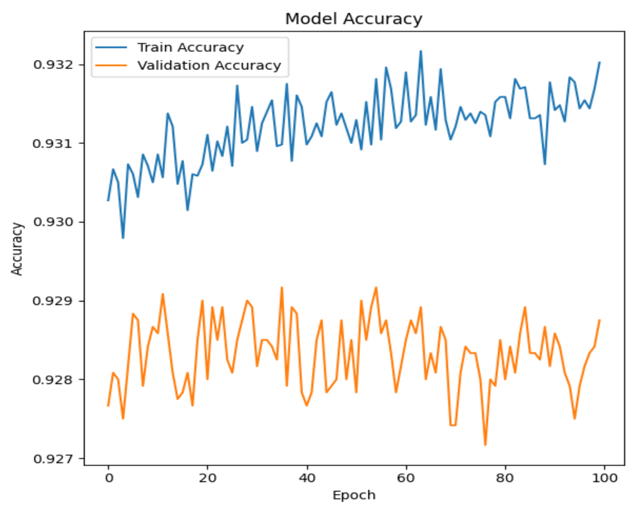
Metric	Without Batch Normalization	With Batch Normalization
Accuracy	0.9258	0.9201
Precision	0.9257	0.9199
Recall	0.9258	0.9201
F1 Score	0.9257	0.9199

Without Batch Normalization





With Batch Normalization



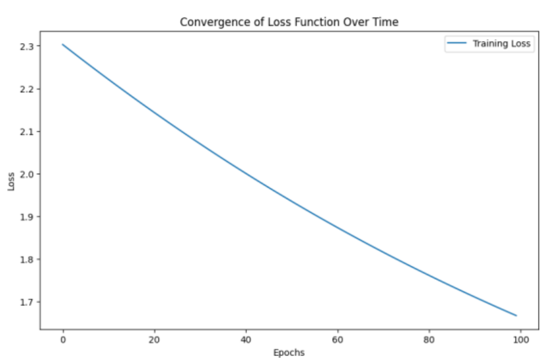
In both cases, training accuracy, validation accuracy, and training loss were the same, but batch normalization led to a more significant decrease in validation loss, indicating better generalization.

Question 3: Gradient Descent and Its Variants

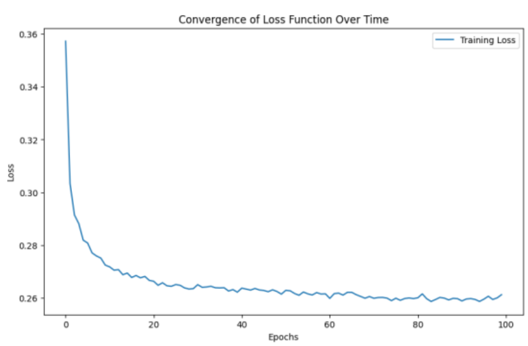
Gradient Descent Method	Accuracy	Precision	Recall	F1 Score
Simple Gradient Descent	0.9236	0.9237	0.9236	0.9235
Stochastic Gradient Descent	0.9235	0.9234	0.9235	0.9233
Mini-batch Gradient Descent	0.9226	0.9227	0.9226	0.9225

The training loss decreases exponentially for Simple Gradient Descent over 100 epochs, shows a rapid but rough decline for Stochastic Gradient Descent, and reaches a smooth convergence with Mini-batch Gradient Descent, all within approximately 10 epochs.

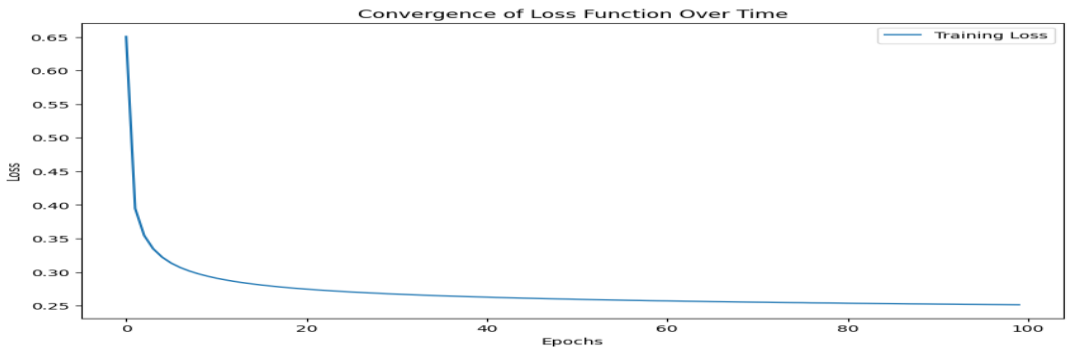
Basic Gradient Decent



SGD



Mini batch DS



Question 4: Hyperparameter Tuning

Data Preprocessing

The dataset was preprocessed using:

- **Normalization**: Scaled pixel values to the range [0, 1].
- **Standard Scaler**: Standardized the data to have a mean of 0 and a standard deviation of 1.
- **PCA**: Reduced dimensionality to enhance training efficiency.

Tuning Process

Using the **KerasTuner** library, hyperparameter tuning was performed for:

- **Learning Rate**: Tuned to find the optimal value.
- **Number of Layers**: Explored 1 to 5 layers.
- **Batch Size**: Various sizes tested.
- **Dropout Rate**: Adjusted from 0 to 0.5.

Results

The best hyperparameters found were:

- **Learning Rate**: 0.00021183086516208134
- **Number of Layers**: 2
- **Units per Layer**: [224, 224]
- **Dropout Rates**: [0.4, 0.1]
- **Batch Size**: 16

Model Training

The neural network was trained for 100 epochs, yielding:

- **Model Accuracy**: 0.9808
- **Model Precision**: 0.9808
- **Model Recall**: 0.9808
- **Model F1 Score**: 0.9808

Comparison

The tuning process significantly enhanced model performance, achieving the highest accuracy of 0.9808, which outperformed all previous experiments conducted throughout this project, including those using logistic regression and various gradient descent methods. This improvement highlights the importance of optimizing hyperparameters, as it led to more effective learning and classification compared to earlier results with models like SGD and Mini-batch Gradient Descent.