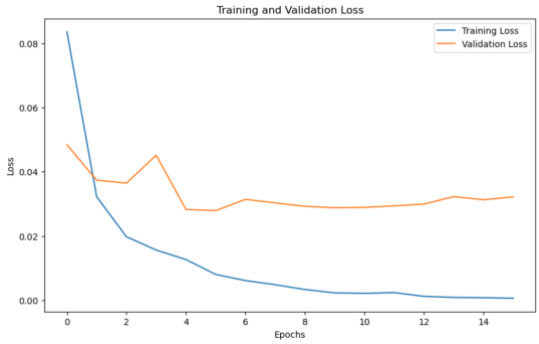
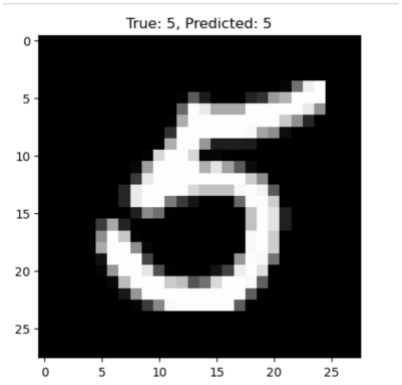
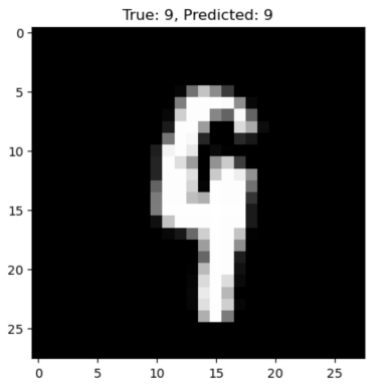
Problem 2:

A. Build a NN for binary classification with **early stopping** criteria based on validation loss. Evaluate your model on the test data. Construct a confusion matrix. Present learning curve and include some examples of your prediction

```
Confusion Matrix:
[[ 889    3]
 [   6 1003]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       892
           1       1.00      0.99      1.00      1009

    accuracy                           1.00      1901
   macro avg       1.00      1.00      1.00      1901
weighted avg       1.00      1.00      1.00      1901
```





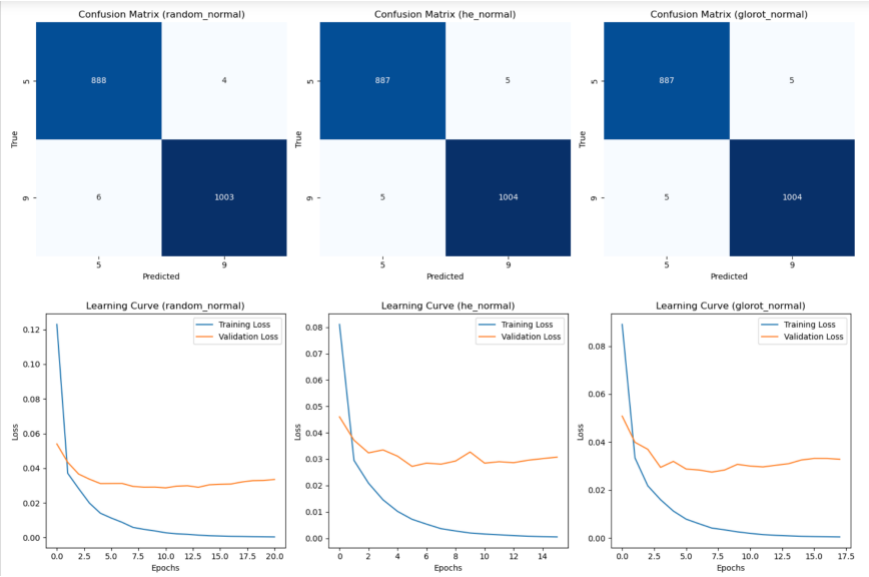



```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_5 (Flatten)         (None, 784)               0

 dense_10 (Dense)            (None, 128)               100480

 dense_11 (Dense)            (None, 1)                 129

=================================================================
Total params: 100609 (393.00 KB)
Trainable params: 100609 (393.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

B. Build three NNs for binary classification using three **different weight initializers** held other hyperparameters constant. a. Construct three different confusion matrices

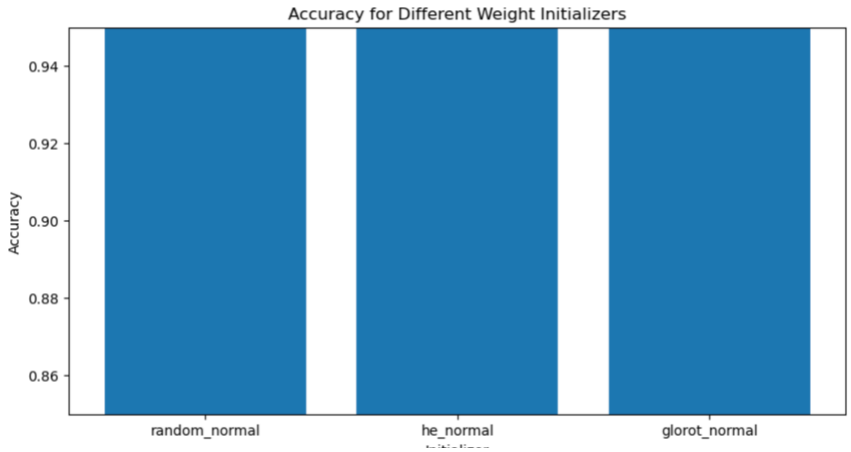
b. Show three different learning curves and explain any differences



Confusion Matrices: The confusion matrices show how well each model is performing in terms of true positives, true negatives, false positives, and false negatives. Different initializers may result in different confusion patterns.

Learning Curves: Learning curves display the training and validation loss over epochs for each model. Differences in learning curves can indicate variations in training behavior, such as convergence speed or overfitting tendencies.

c. Show accuracy using bar plots and explain – _if there any difference in results for using three different initializers

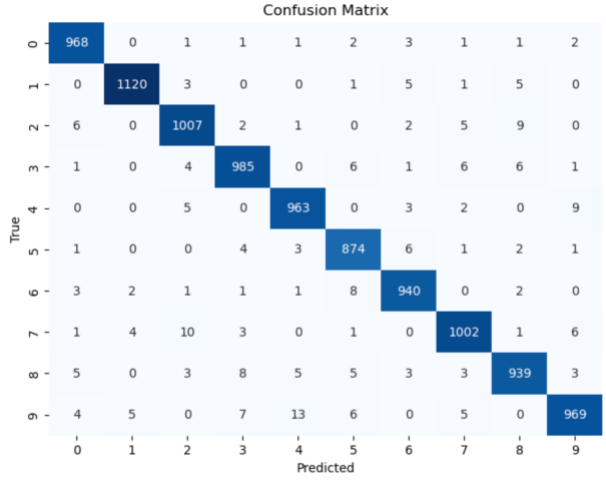


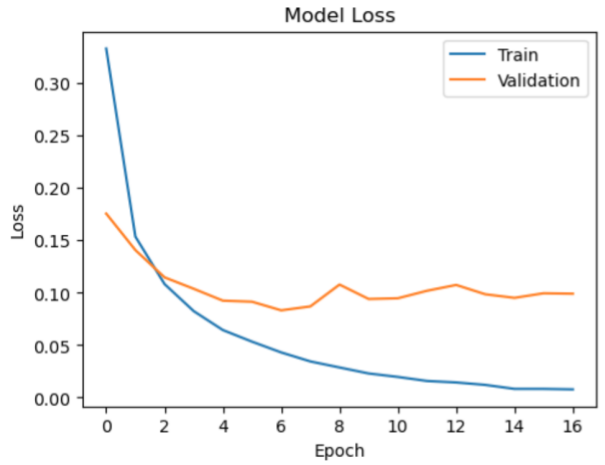
Accuracy: The bar plot of accuracy shows how well each model performs in terms of overall accuracy. It allows you to compare the models directly.
By using different weight initializers, you can observe how these initializations impact the model's performance and training behavior.

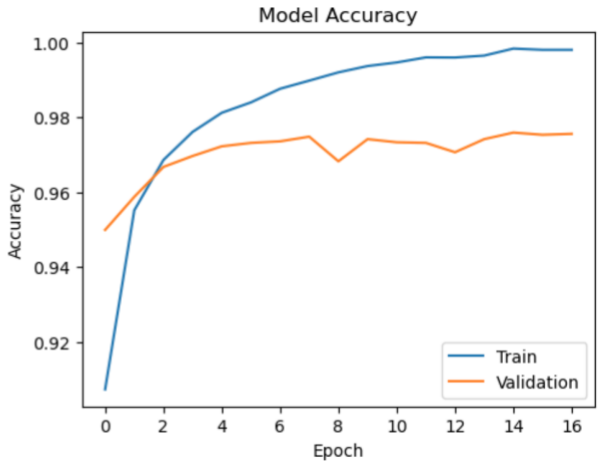
```
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_8 (Flatten)         (None, 784)               0

 dense_16 (Dense)            (None, 128)               100480

 dense_17 (Dense)            (None, 1)                 129

=================================================================
Total params: 100609 (393.00 KB)
Trainable params: 100609 (393.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Problem 3**:** Build a NN for multi-class classification considering all the classes (10 classes) in the MNIST digit dataset: consider **early stopping criteria based on the validation loss** and finally construct confusion matrix and discuss the results.







```
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_9 (Flatten)         (None, 784)               0

 dense_18 (Dense)            (None, 128)               100480

 dense_19 (Dense)            (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Hyperparameters

Activation Function (Hidden Layer): ReLU

Activation Function (Output Layer): Softmax

Weight Initializer: Default (Random Initialization)

Number of Hidden Layers: 1

Neurons in Hidden Layer: 128

Loss Function: Sparse Categorical Cross-Entropy

Optimizer: Adam

Number of Epochs: 50

Batch Size: 64

Learning Rate: Default (Adaptive in Adam)

Evaluation Metric: Accuracy