Report ML Offline-3

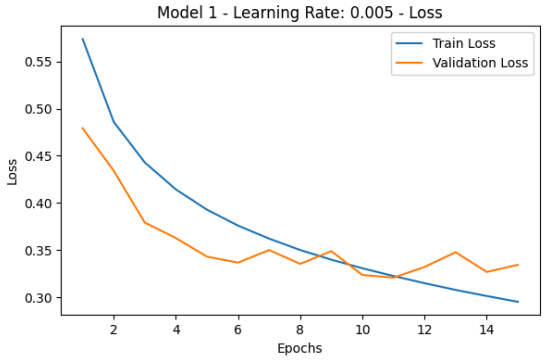
Name : Protoy Barai Student ID : 1905068

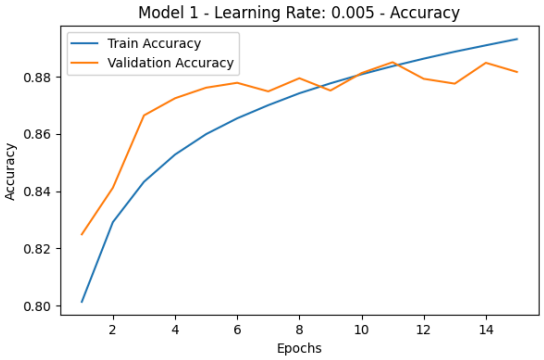
1. First, install necessary dependencies like torchvision, numpy, pandas and scroll below to the ‘Training’ section. Here is the code to train 3 different models with different optimizer learning rates. It also saves the best model as best_model.pkl file
2. Now scroll below to the ‘Saving the weights’ section to save only the weights of the best model because the full model size is too big
3. After that, load the weights to the best model and test it in the ‘Testing the best model’ section.

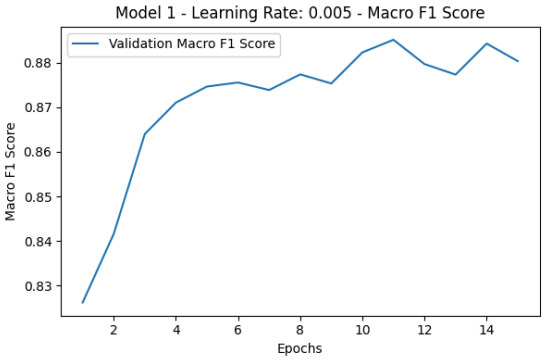
Model - 1

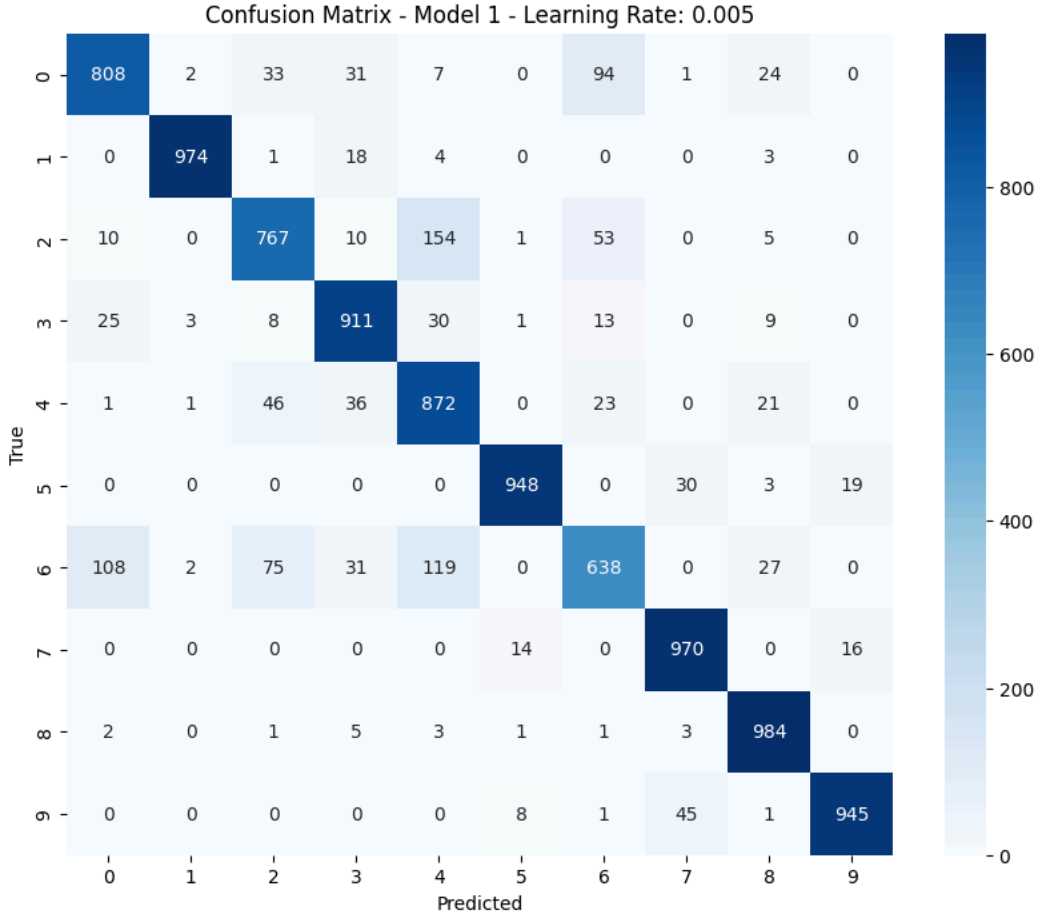
```
def build_model_1(learning_rate):
    layers = [
        DenseLayer(784, 256),
        BatchNormalization(256),
        ReLU(),
        DenseLayer(256, 128),
        BatchNormalization(128),
        ReLU(),
        Dropout(0.2),
        DenseLayer(128, 64),
        BatchNormalization(64),
        ReLU(),
        DenseLayer(64, 10),
        SoftmaxLayer()
    ]
    model = NeuralNetwork(layers, learning_rate)
    return model
```

Model-1 with learning rate-0.005

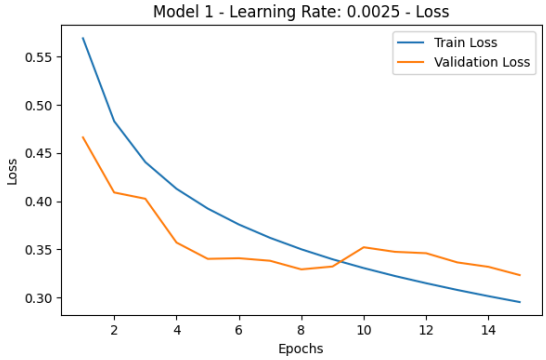

Model 1 - Learning Rate: 0.005 - Loss
Train Loss
Validation Loss
Loss
Epochs

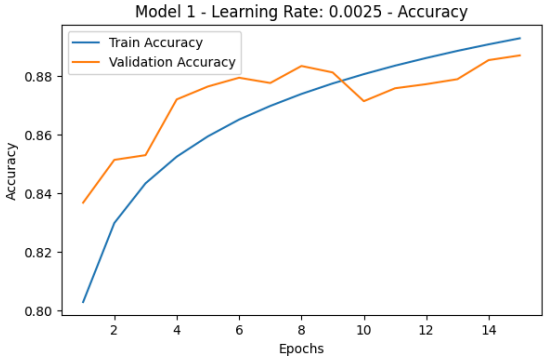

Model 1 - Learning Rate: 0.005 - Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs

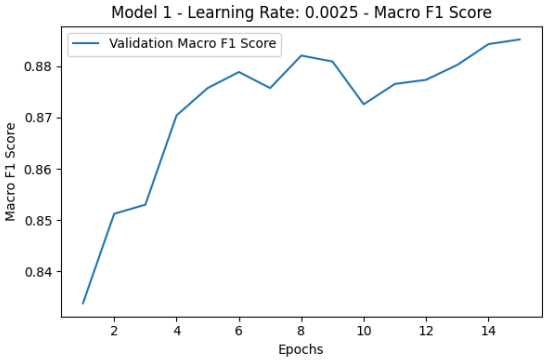

Model 1 - Learning Rate: 0.005 - Macro F1 Score
Validation Macro F1 Score
Macro F1 Score
Epochs

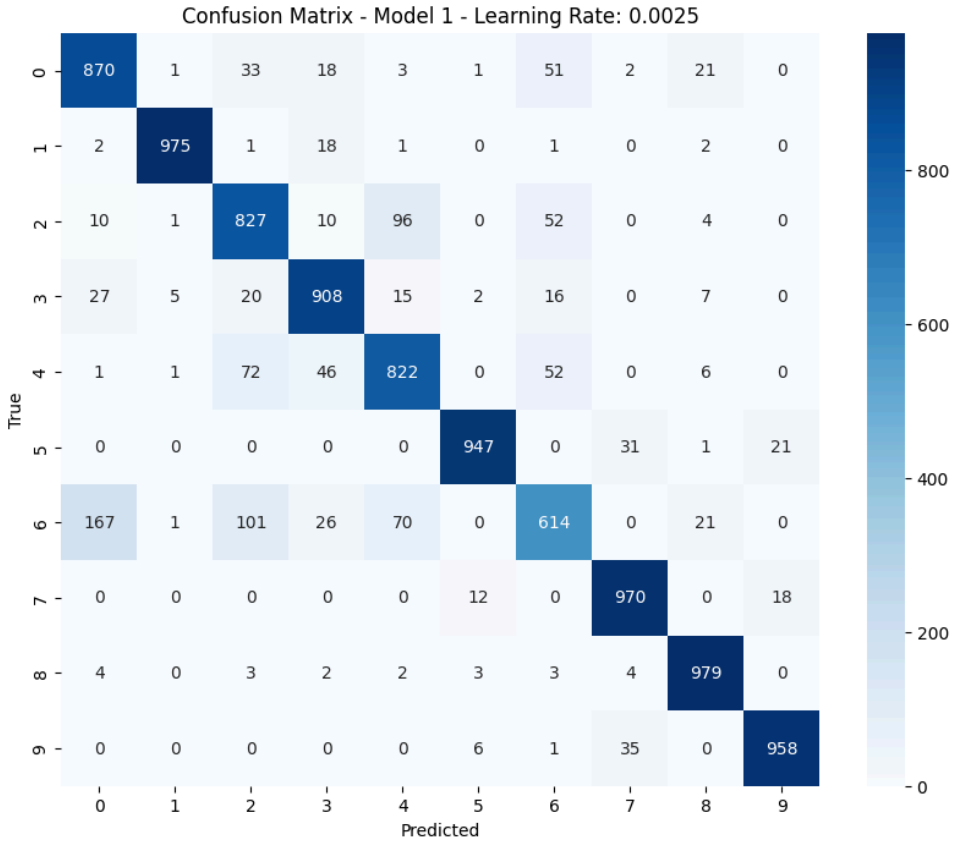

Confusion Matrix - Model 1 - Learning Rate: 0.005
True
Predicted
808 2 33 31 7 0 94 1 24 0
0 974 1 18 4 0 0 0 3 0
10 0 767 10 154 1 53 0 5 0
25 3 8 911 30 1 13 0 9 0
1 1 46 36 872 0 23 0 21 0
0 0 0 0 0 948 0 30 3 19
108 2 75 31 119 0 638 0 27 0
0 0 0 0 0 14 0 970 0 16
2 0 1 5 3 1 1 3 984 0
0 0 0 0 0 8 1 45 1 945

Model - 1 with learning rate 0.0025

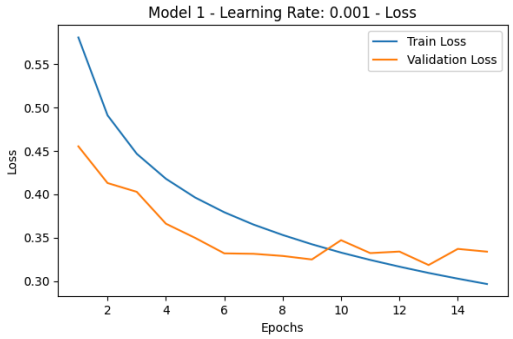


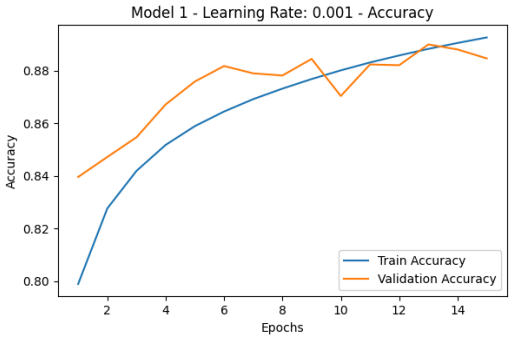


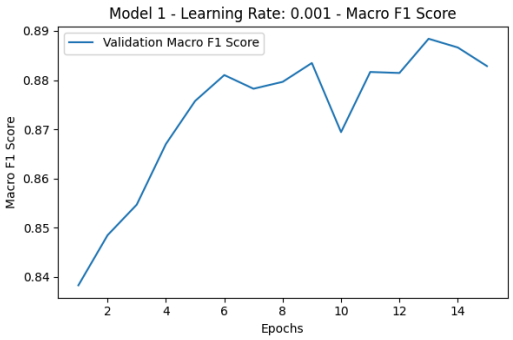


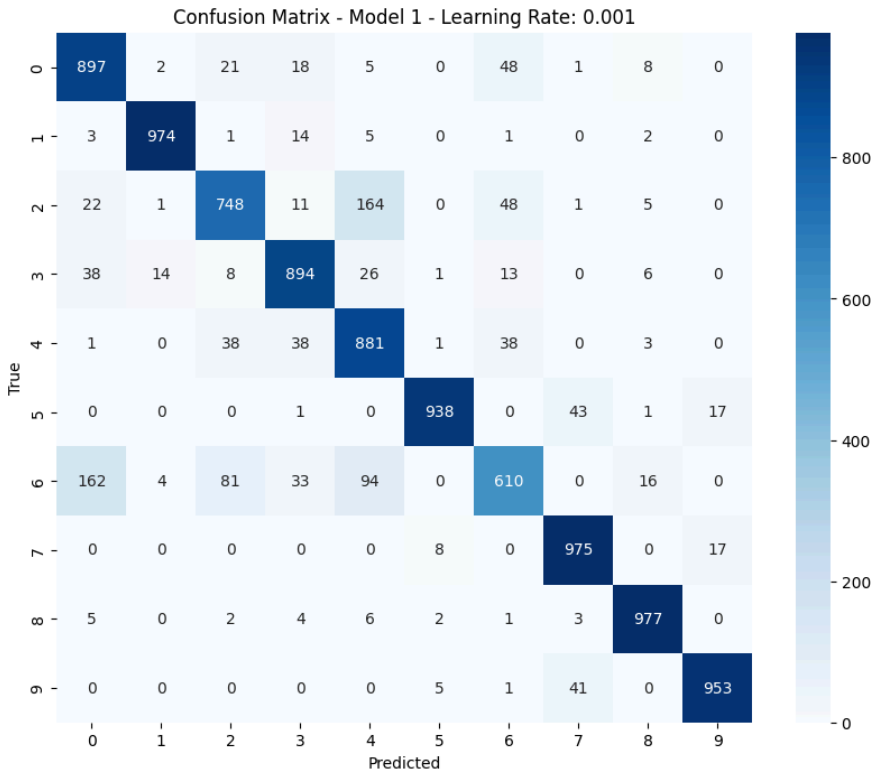


Model-1 with learning rate 0.001

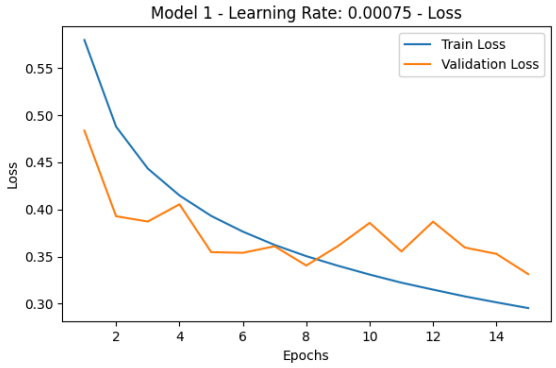


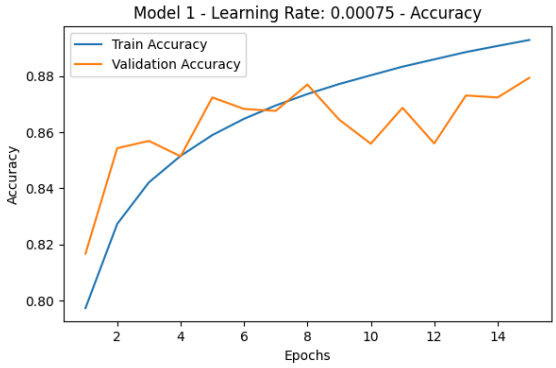


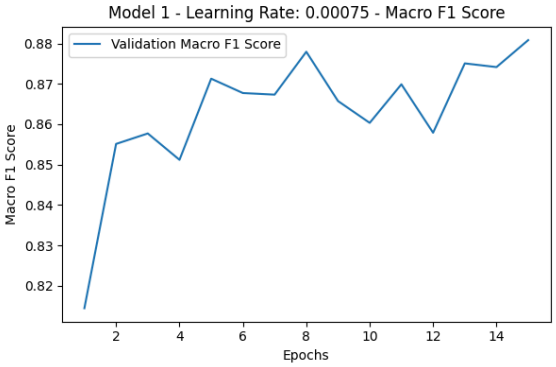


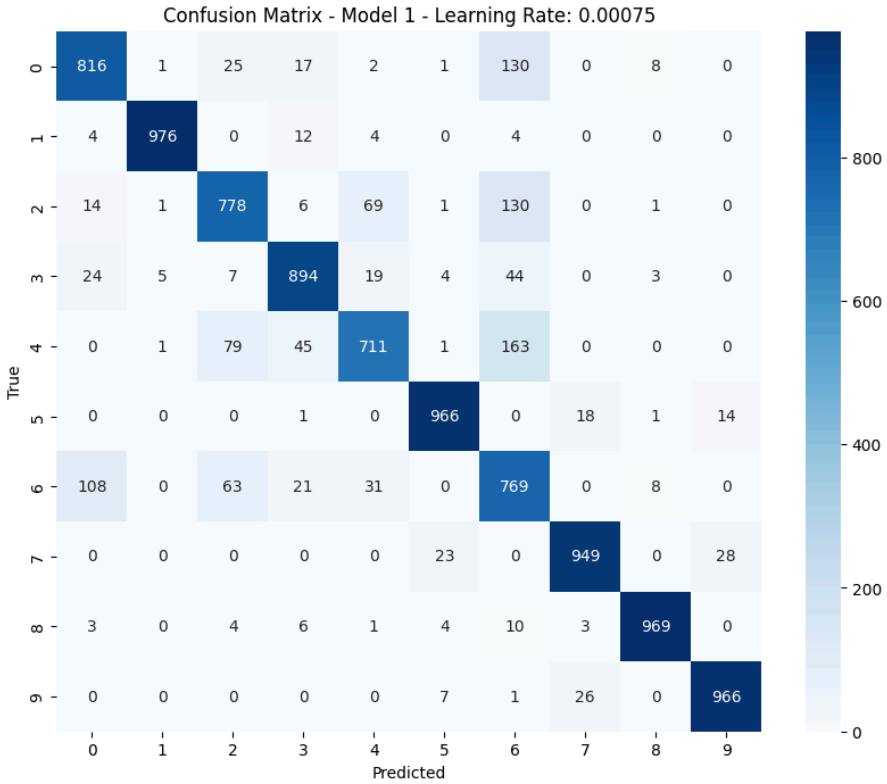


Model-1 with learning rate 0.00075





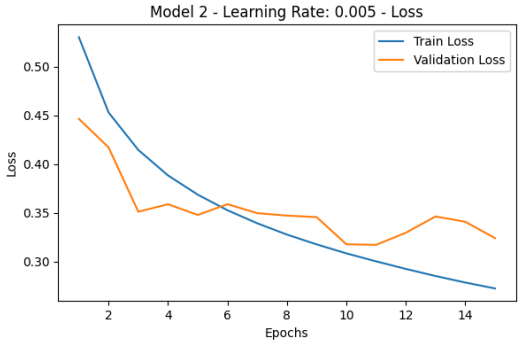


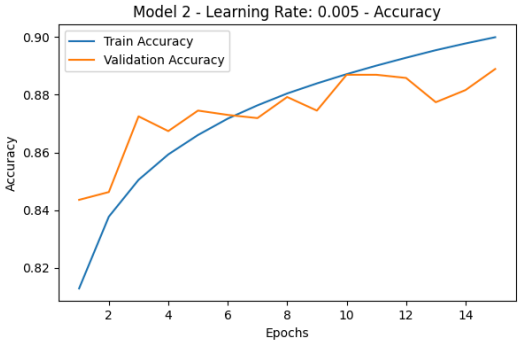


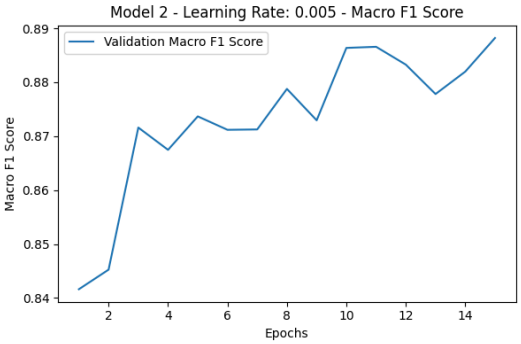
Model - 2

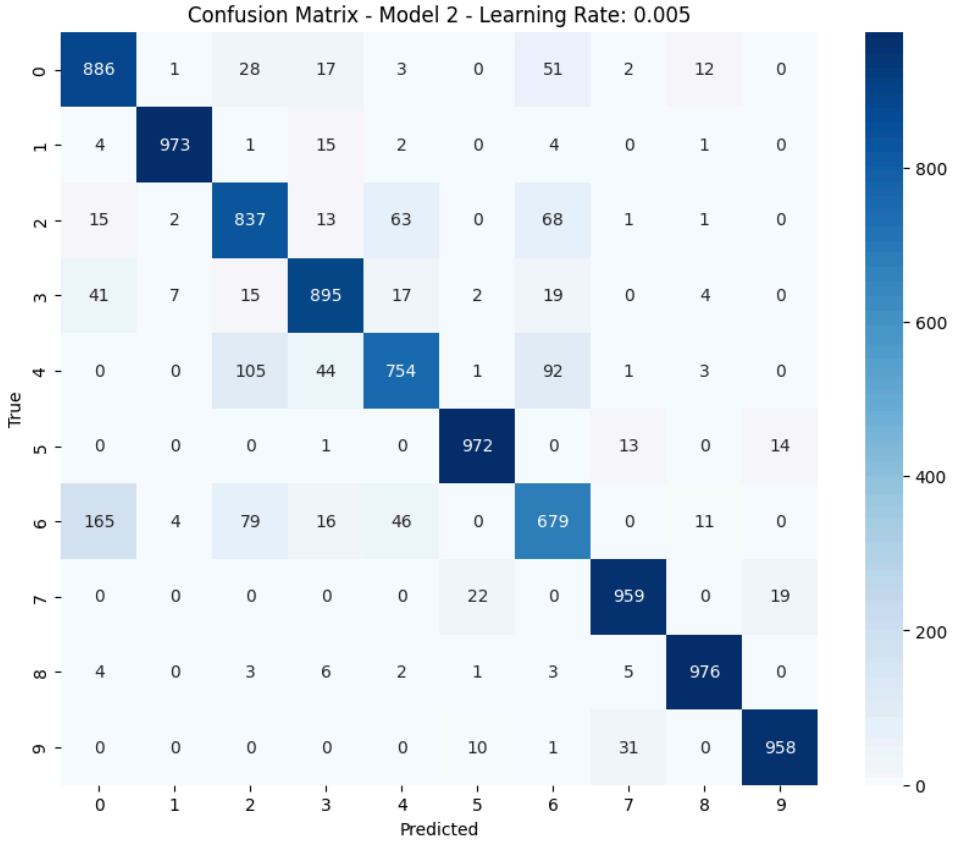
```
def build_model_2(learning_rate):
    layers = [
        DenseLayer(784, 512),
        BatchNormalization(512),
        ReLU(),
        DenseLayer(512, 256),
        BatchNormalization(256),
        ReLU(),
        Dropout(0.3),
        DenseLayer(256, 128),
        BatchNormalization(128),
        ReLU(),
        DenseLayer(128, 10),
        SoftmaxLayer()
    ]
    model = NeuralNetwork(layers, learning_rate)
    return model
```

Model-2 with learning rate 0.005

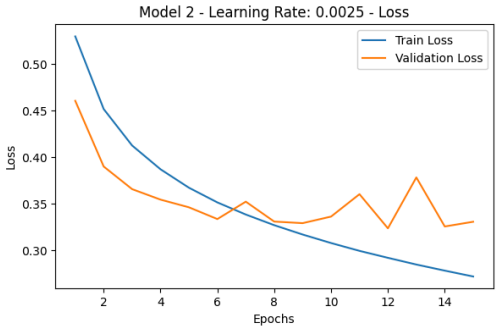

Model 2 - Learning Rate: 0.005 - Loss
Train Loss
Validation Loss
Loss
Epochs

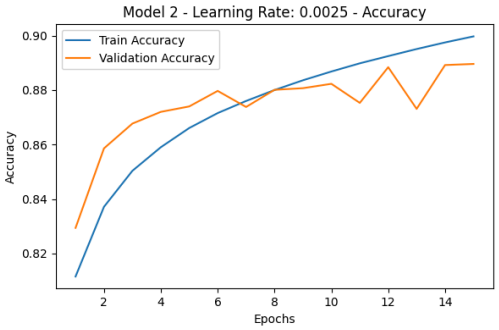

Model 2 - Learning Rate: 0.005 - Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs

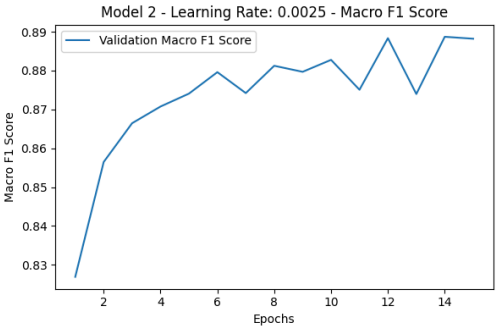

Model 2 - Learning Rate: 0.005 - Macro F1 Score
Validation Macro F1 Score
Macro F1 Score
Epochs

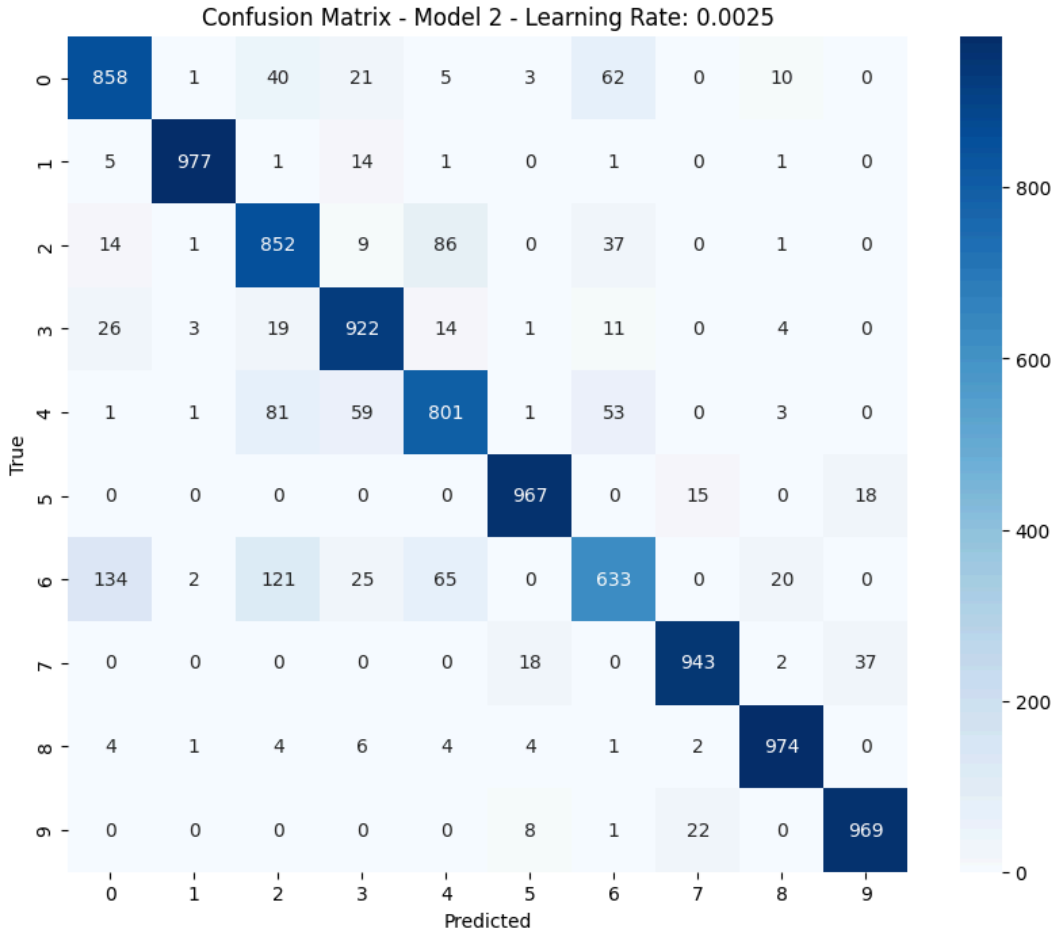

Confusion Matrix - Model 2 - Learning Rate: 0.005
True
Predicted

Model-2 with learning rate 0.0025

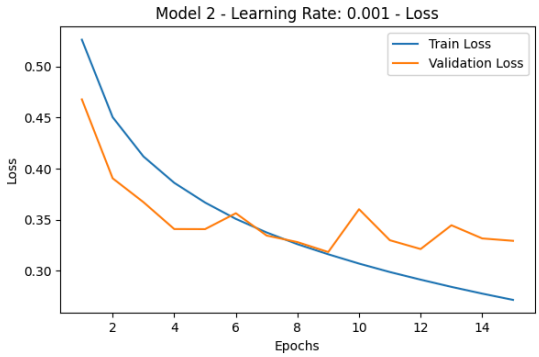


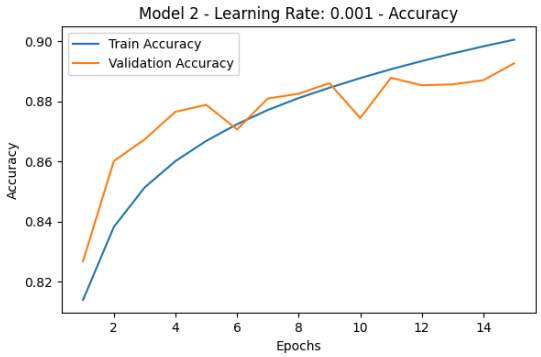


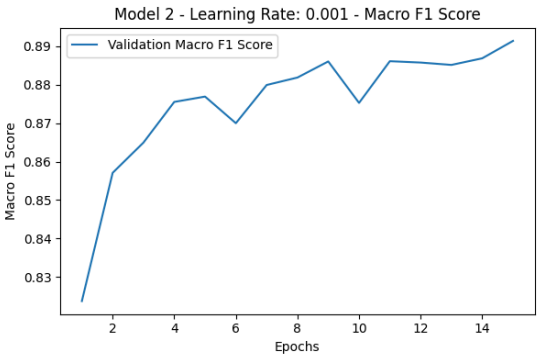


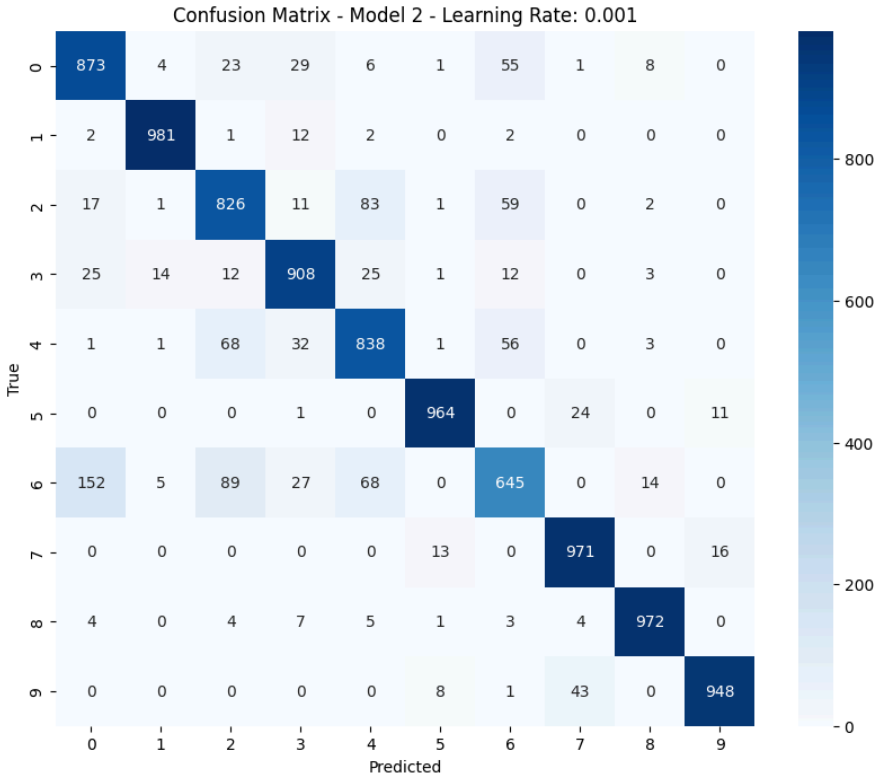


Model-2 with learning rate 0.001

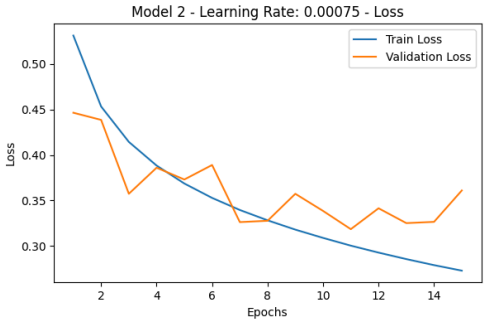


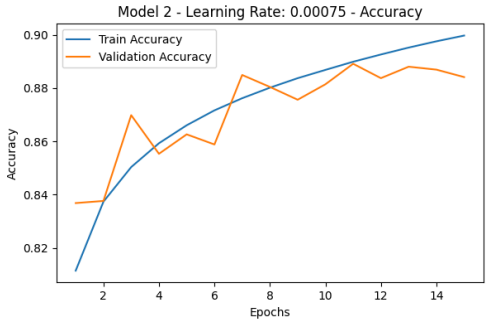


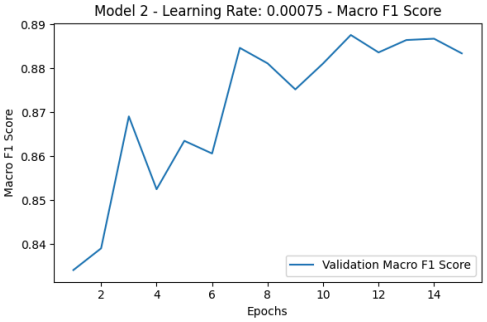


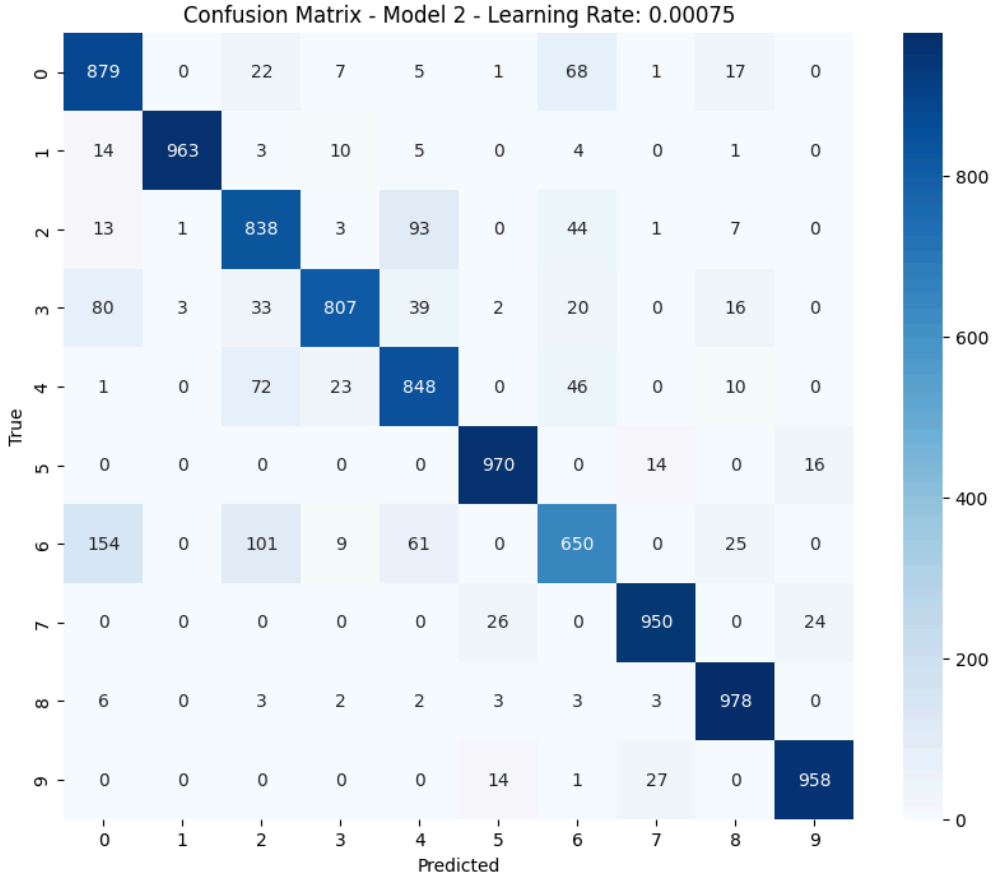


Model-2 with learning rate 0.00075





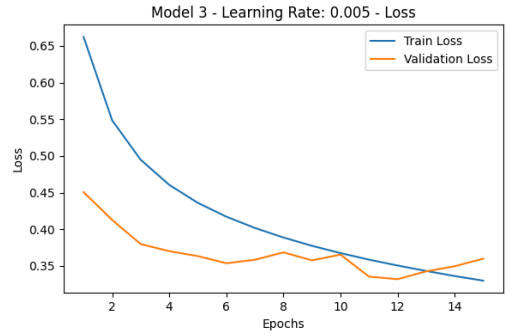


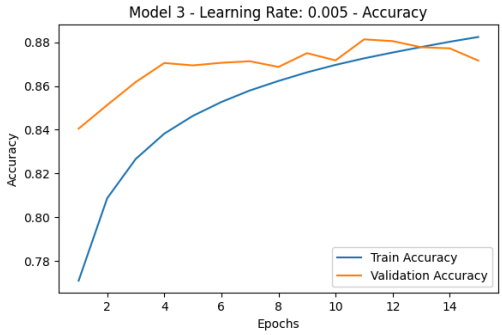


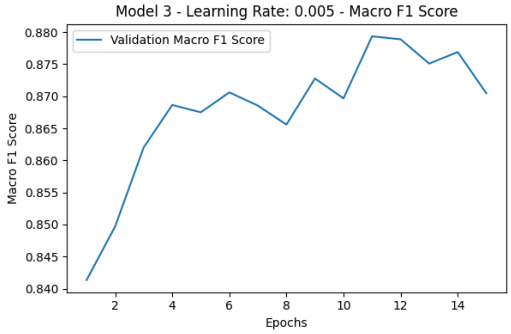
Model-3

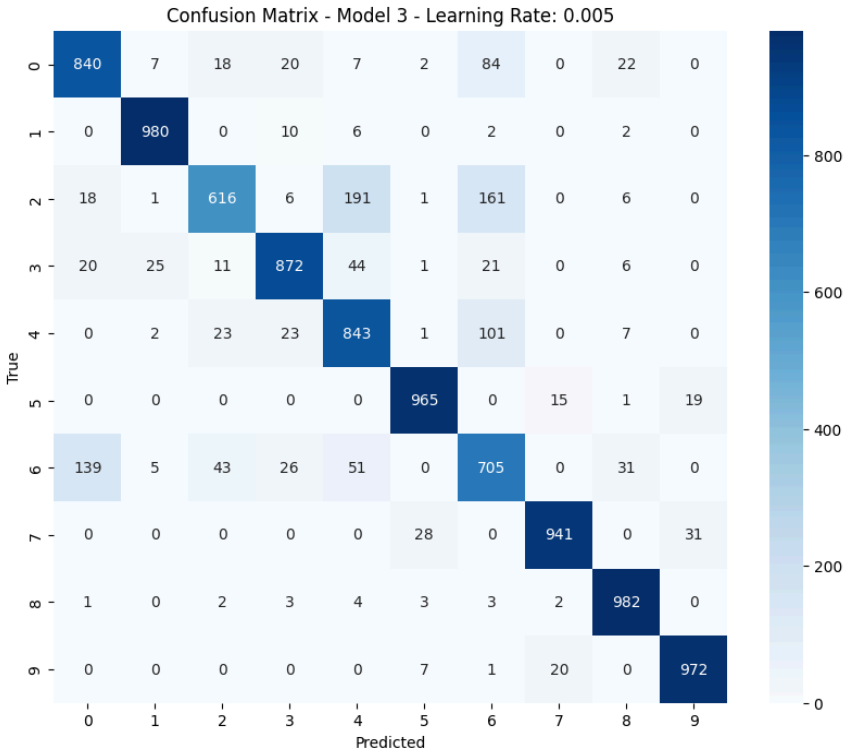
```python
def build_model_3(learning_rate):
    layers = [
        DenseLayer(784, 128),
        BatchNormalization(128),
        ReLU(),
        DenseLayer(128, 64),
        BatchNormalization(64),
        ReLU(),
        Dropout(0.1),
        DenseLayer(64, 32),
        BatchNormalization(32),
        ReLU(),
        DenseLayer(32, 10),
        SoftmaxLayer()
    ]
    model = NeuralNetwork(layers, learning_rate)
    return model
```

Model-3 with learning rate 0.005

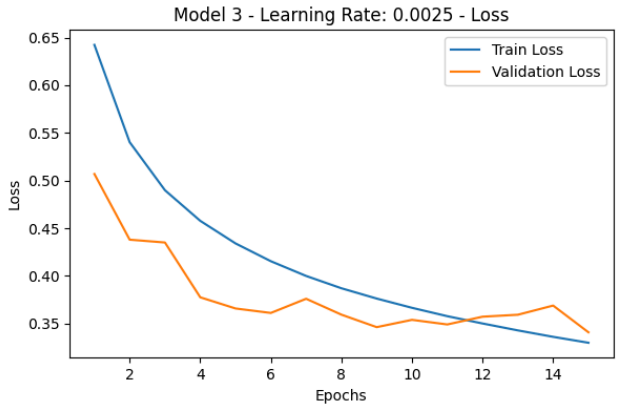


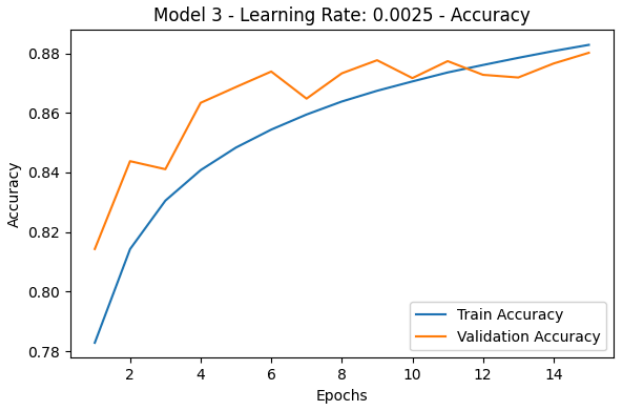


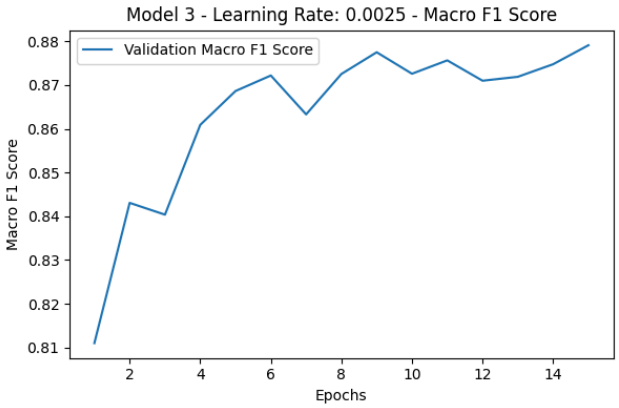


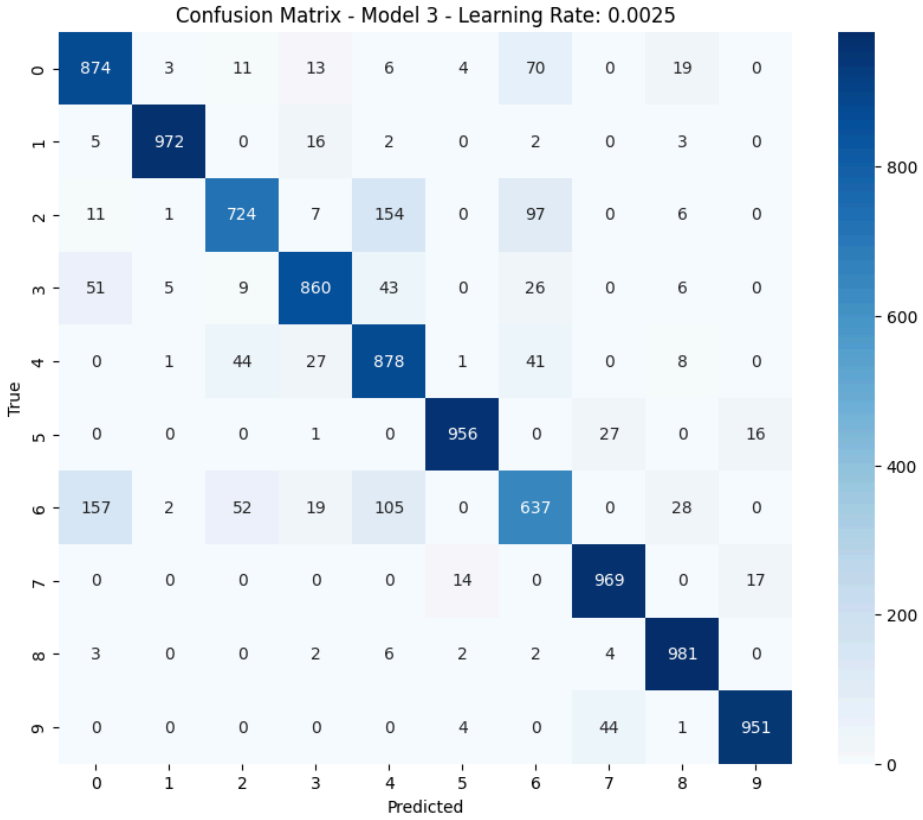


Model-3 with learning rate 0.0025

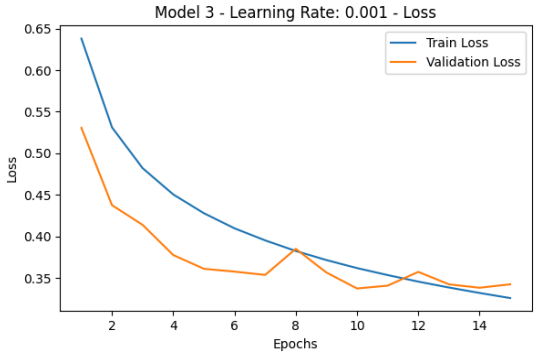

Model 3 - Learning Rate: 0.0025 - Loss
Train Loss
Validation Loss
Loss
Epochs

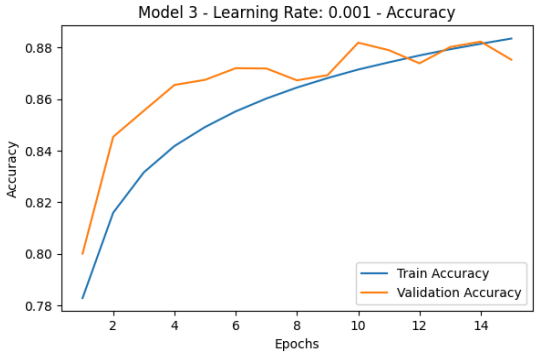

Model 3 - Learning Rate: 0.0025 - Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs

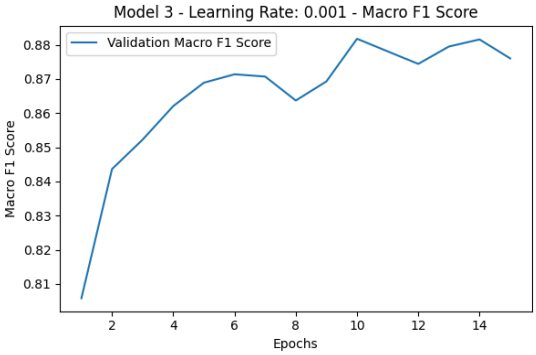

Model 3 - Learning Rate: 0.0025 - Macro F1 Score
Validation Macro F1 Score
Macro F1 Score
Epochs

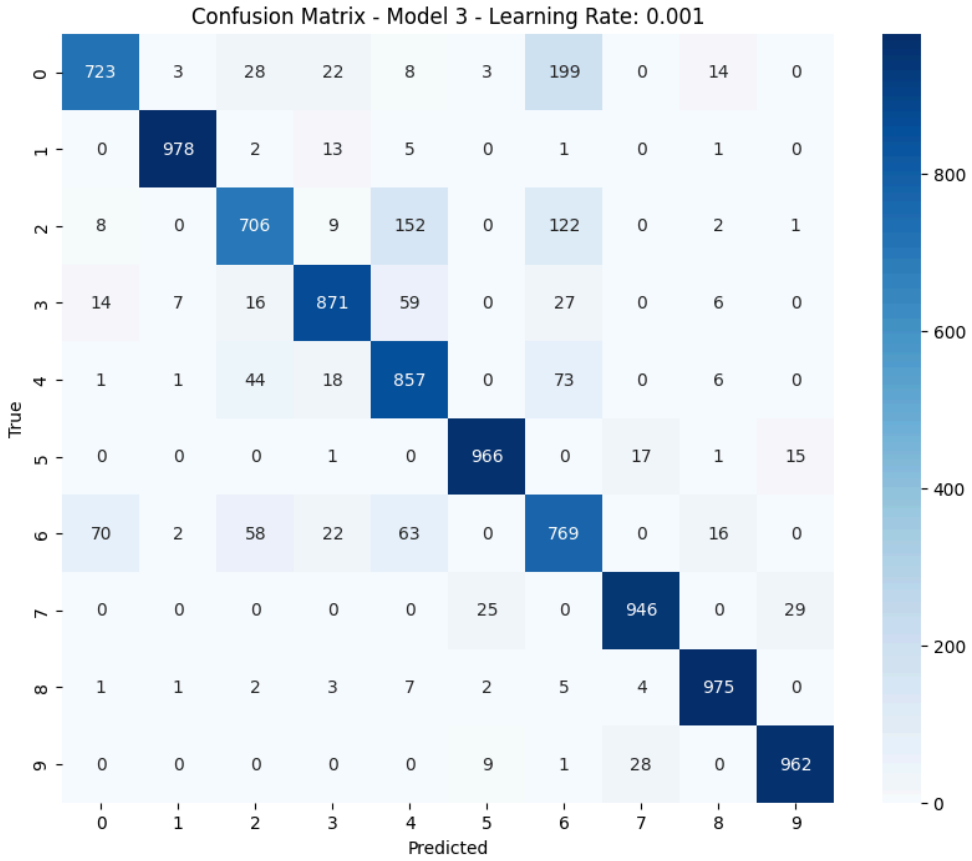

Confusion Matrix - Model 3 - Learning Rate: 0.0025
True
Predicted

Model-3 with learning rate 0.001

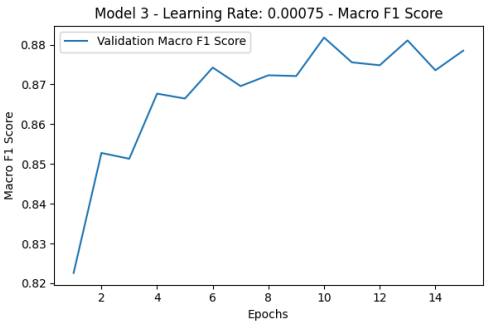


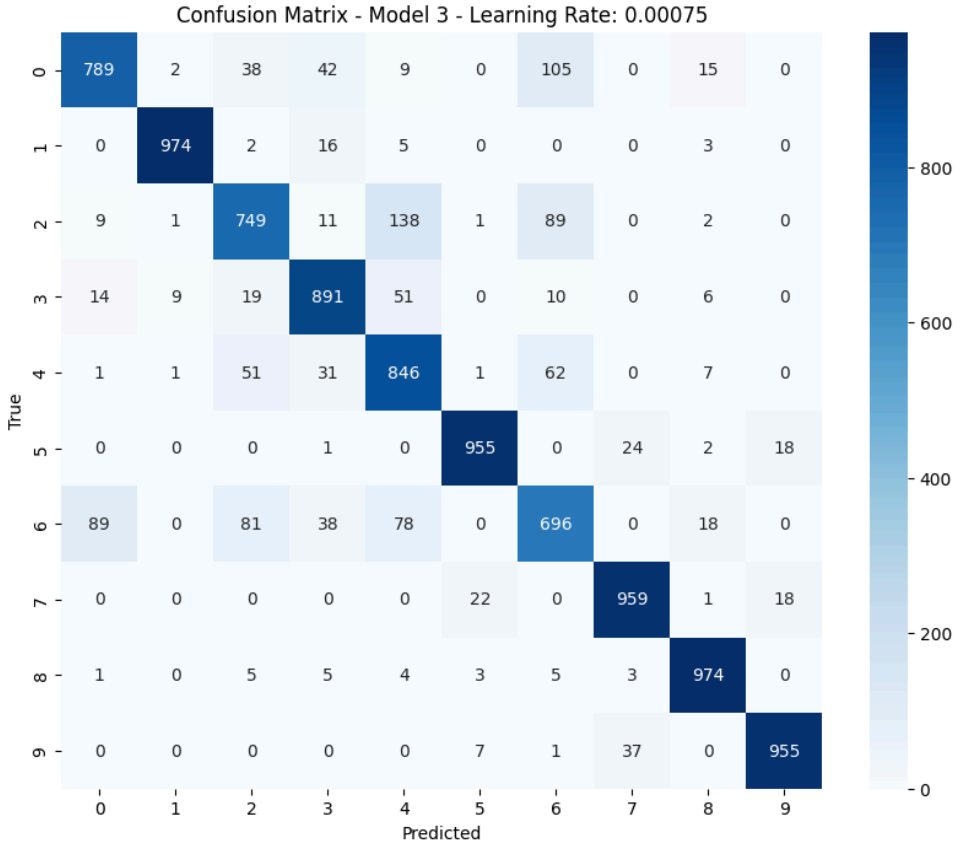






Model-3 with learning rate 0.00075





Model Performance :

Model	Learning Rate	Training Loss	Validation Loss	Train Accuracy	Validation Accuracy	Validation macro F1
Model 2	**0.00100**	**0.271397**	**0.329285**	**0.900464**	**0.8926**	**0.891347**
Model 2	0.00250	0.271912	0.330645	0.899719	0.8896	0.888197
Model 2	0.00500	0.272389	0.324055	0.899901	0.8889	0.888195
Model 1	0.00250	0.295305	0.323335	0.892844	0.8870	0.885265
Model 2	0.00075	0.272682	0.360921	0.899693	0.8841	0.883375
Model 1	0.00100	0.296605	0.333930	0.892667	0.8847	0.882844
Model 1	0.00075	0.295227	0.331307	0.892887	0.8794	0.880822
Model 1	0.00500	0.295134	0.334354	0.893186	0.8817	0.880303
Model 3	0.00250	0.329868	0.340848	0.882859	0.8802	0.879105
Model 3	0.00075	0.326605	0.346331	0.883825	0.8788	0.878500
Model 3	0.00100	0.325852	0.342397	0.883494	0.8788	0.878500
Model 3	0.00500	0.329797	0.359857	0.882390	0.8716	0.870483

Best Model :
Model 2 with learning rate 0.001