Report on Function Approximation with Neural Network and Backpropagation

1905113 - Anamul Hoque Emtiaj

1 Setup and Execution Instructions

1.1 Dependencies Installation

To run the project, make sure to have Python installed and install the following dependencies:

```
pip install numpy pandas matplotlib tqdm torchvision scikit-learn
```

This will install essential packages required for handling datasets like FashionMNIST.

1.2 Training the Model

To train the model, follow these steps:

- Place the `1905113.ipynb` file in your project folder.
- Specify the model architecture using the following configuration:

```
config = {
    'learning_rate': 0.0005,
    'num_epochs': 50,
    'batch_size': 32,
    'hidden_layers': [
        {'size': 512, 'bn': True, 'dropout': 0.3, 'relu': True},
        {'size': 256, 'bn': False, 'dropout': 0.4, 'relu': True},
        {'size': 128, 'bn': True, 'dropout': 0.5, 'relu': True}
    ]
}
```

- Build the model by calling the function `build_model(config)`, which will return the model.
- Train the model using the following command:

```
model.fit(train_data, train_labels, val_data, val_labels)
```

1.3 Testing with Saved Model

To test the model with pre-trained parameters, proceed with the following steps:

- Place the `model_1905113.pickle` file in the project folder.
- Open the `1905113.ipynb` file, comment out the 'Validation', 'Train and Save Best model' and 'Independent Testing - Direct Trained Model'sections.
- Load the previously saved model using the following command:

```
loaded_model = load_model('model_1905113.pkl')
```

- Make predictions on the test data with the following commands:

```
    test_predictions = loaded_model.predict(X_test)
    test_loss, test_acc, test_macro_f1 = loaded_model.evaluate(X_test, y_test_one_hot)
    print(f"Test Loss: {test_loss:.4f}, Test Accuracy: {test_acc:.4f}, Test Macro F1: {test_macro_f1:.4f}"

    conf_matrix = confusion_matrix(y_test, test_predictions)
    print("Confusion Matrix:")
    print(conf_matrix)
```

2 Validation Performance

2.1 With Three Different Architectures

For this section, we experimented with three different neural network architectures. The configuration of each model and their performance results are described below.

2.1.1 Model 1

Configuration:

```
config_model_1 = {
    'learning_rate': 0.001,
    'num_epochs': 50,
    'batch_size': 32,
    'hidden_layers': [
        {'size': 512, 'bn': True, 'dropout': 0.3, 'relu': True},
        {'size': 256, 'bn': False, 'dropout': 0.4, 'relu': True},
        {'size': 128, 'bn': True, 'dropout': 0.5, 'relu': True}
    ]
}
```

Performance:

- Train Accuracy: 0.9366
- Validation Accuracy: 0.9028
- Train Loss: 0.3159
- Validation Loss: 0.40
- Validation Macro F1 Score: 0.9039

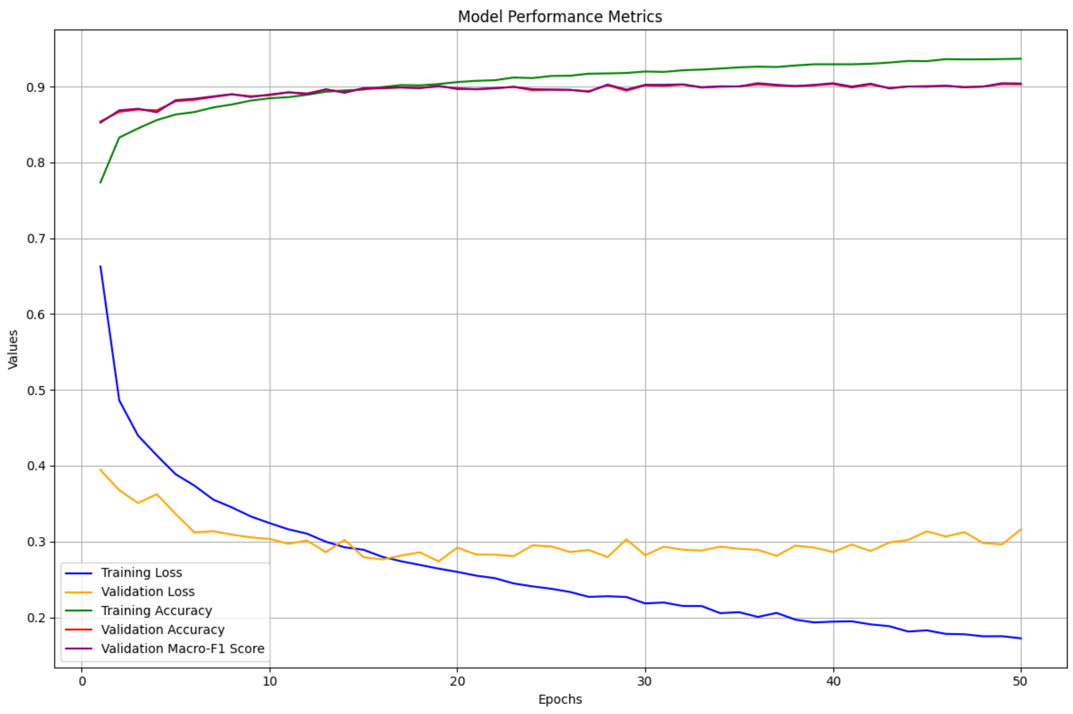

Figure 1: Training and Validation History for Model 1

Confusion Matrix:

Table 1: Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	519	2	13	9	3	0	50	0	1	0
1	1	597	1	8	1	0	0	0	0	0
2	10	0	524	2	49	0	26	0	0	0
3	31	3	6	513	23	0	8	0	3	0
4	1	0	36	12	548	0	27	0	3	0
5	0	0	0	0	0	600	0	16	1	4
6	70	1	47	9	33	0	455	0	4	0
7	0	0	0	0	0	7	0	536	0	7
8	2	0	3	1	3	0	12	1	568	0
9	0	0	0	0	0	8	0	25	0	557

2.1.2 Model 2

Configuration:

```
config_model_2 = {
    'learning_rate': 0.001,
    'num_epochs': 50,
    'batch_size': 64,
    'hidden_layers': [
        {'size': 512, 'bn': False, 'dropout': 0.3, 'relu': True},
        {'size': 256, 'bn': True, 'dropout': 0.4, 'relu': True}
    ]
}
```

Performance:

- Train Accuracy: 0.9336

- Validation Accuracy: 0.9025
- Train Loss: 0.1760
- Validation Loss: 0.2910
- Validation Macro F1 Score: 0.9033

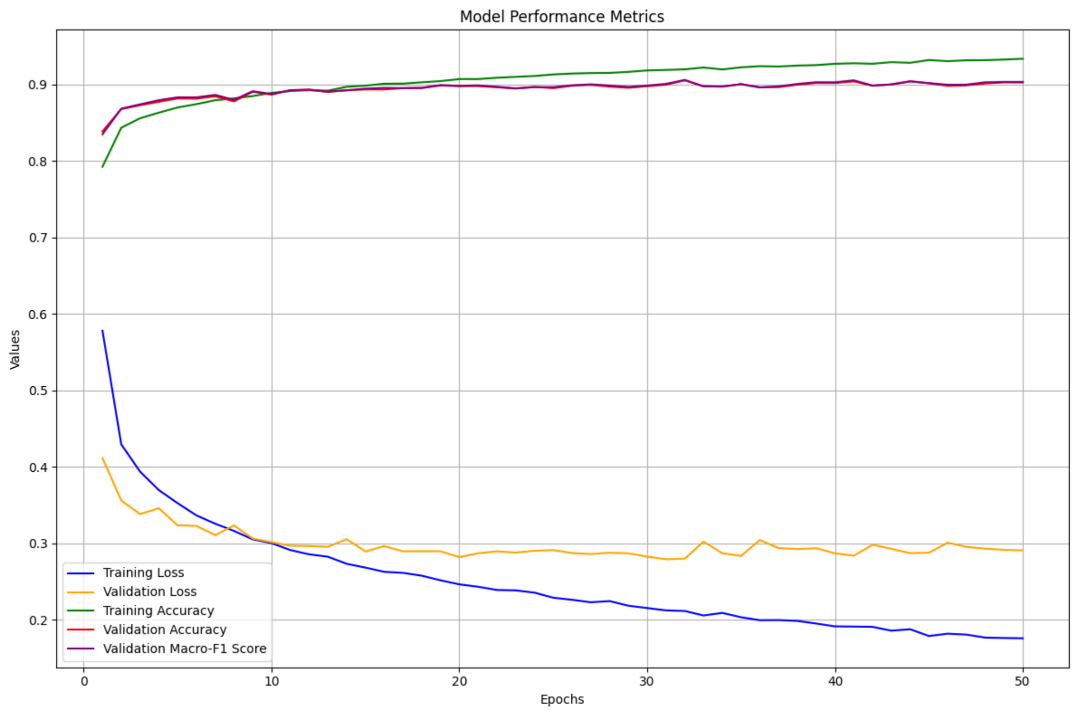

Figure 2: Training and Validation History for Model 2

Confusion Matrix:

Table 2: Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	525	0	5	16	2	0	47	0	2	0
1	1	595	0	12	0	0	0	0	0	0
2	16	0	500	4	57	0	34	0	0	0
3	13	2	2	555	13	0	2	0	0	0
4	2	0	37	27	521	0	38	0	2	0
5	0	0	0	0	0	602	0	13	0	6
6	75	1	36	17	31	0	454	0	5	0
7	0	0	0	0	0	6	0	529	0	15
8	4	0	3	2	2	1	6	1	571	0
9	0	0	0	0	0	12	0	15	0	563

2.1.3 Model 3

Configuration:

```
config_model_3 = {
    'learning_rate': 0.001,
    'num_epochs': 50,
    'batch_size': 128,
    'hidden_layers': [
        {'size': 1024, 'bn': True, 'dropout': 0.3, 'relu': True},
        {'size': 512, 'bn': True, 'dropout': 0.5, 'relu': True}
    ]
}
```

Performance:

- Train Accuracy: 0.9672
- Validation Accuracy: 0.9015
- Train Loss: 0.0856
- Validation Loss: 0.3663
- Validation Macro F1 Score: 0.9021

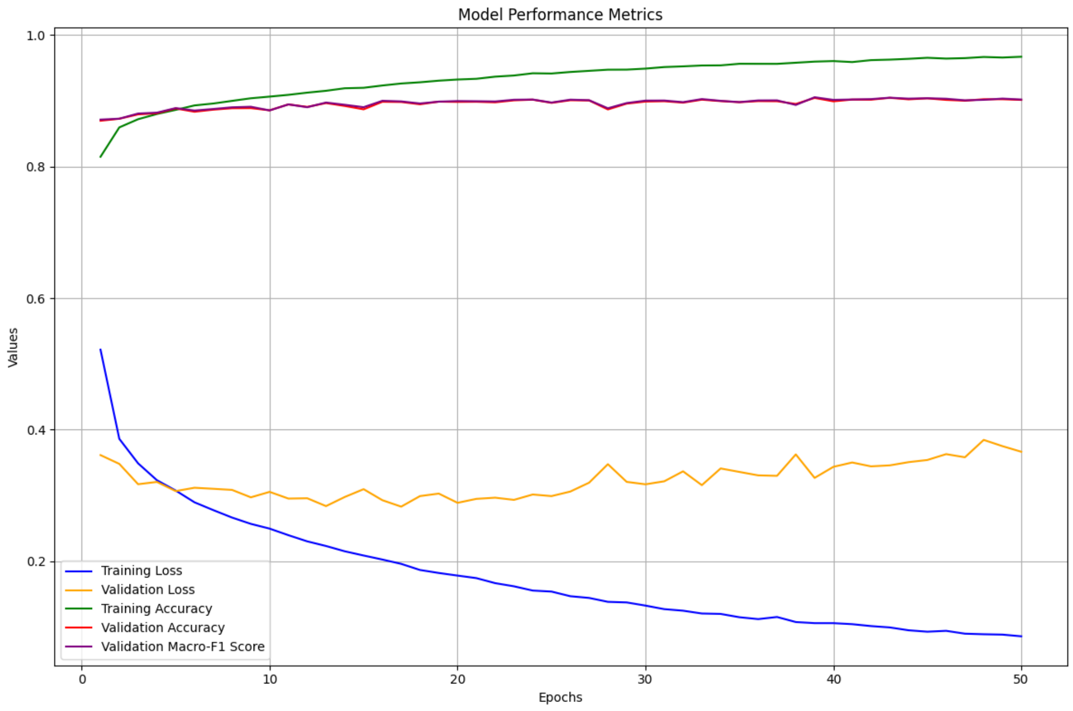

Figure 3: Training and Validation History for Model 3

Confusion Matrix:

Table 3: Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	553	0	11	3	2	0	25	0	3	0
1	1	595	1	11	0	0	0	0	0	0
2	13	0	535	2	32	0	29	0	0	0
3	29	0	5	528	14	0	9	1	1	0
4	3	0	67	18	498	0	36	0	5	0
5	0	0	0	0	0	595	0	14	1	11
6	94	1	58	16	12	0	429	0	9	0
7	0	0	0	0	0	6	0	536	0	8
8	2	1	2	1	2	0	4	2	576	0
9	0	0	0	0	0	6	0	20	0	564

2.1.4 Comparison of Architectures

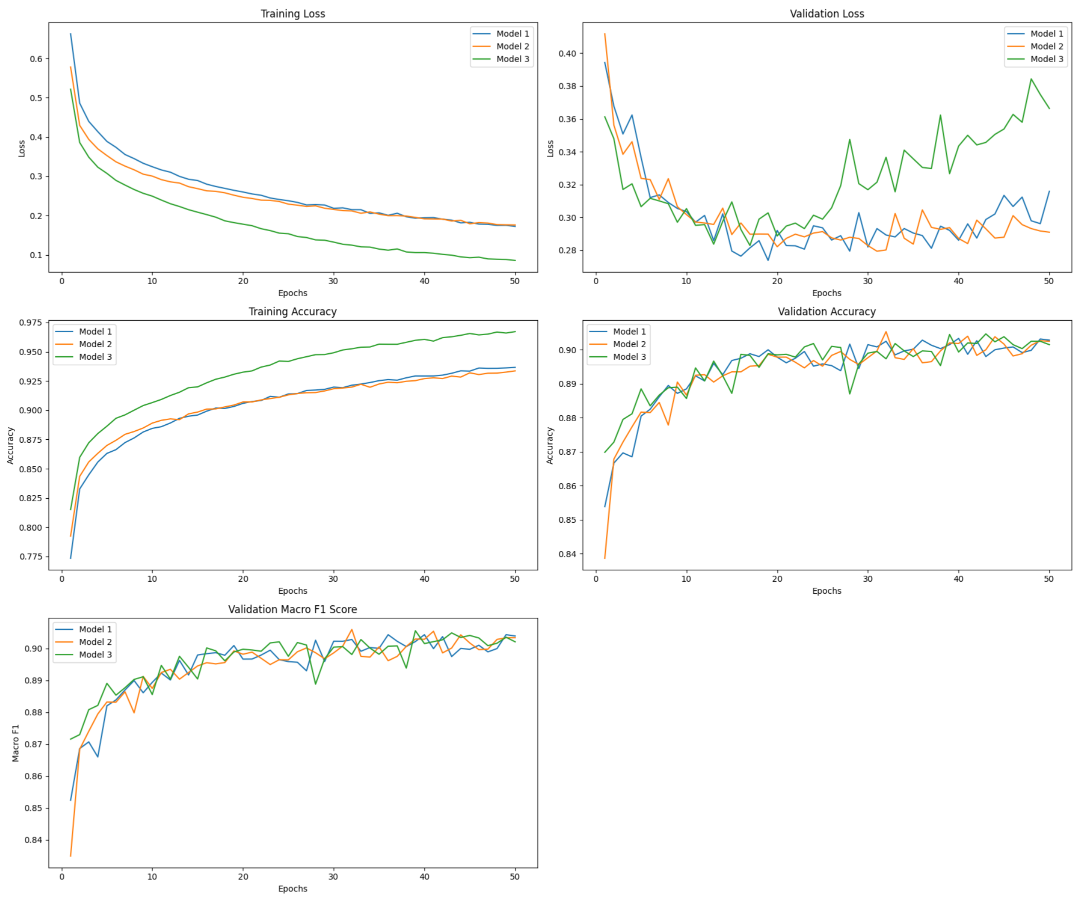

Figure 4: Comparison of Training and Validation History for All Three Architectures

2.2 With Four Different Learning Rates

We experimented with four different learning rates. Below are the results.

2.2.1 Model 1 (Learning Rate = 0.005)

Configuration:

```
config_model_1 = {
    'learning_rate': 0.005,
    'num_epochs': 50,
    'batch_size': 32,
    'hidden_layers': [
        {'size': 512, 'bn': True, 'dropout': 0.3, 'relu': True},
        {'size': 256, 'bn': False, 'dropout': 0.4, 'relu': True},
        {'size': 128, 'bn': True, 'dropout': 0.5, 'relu': True}
    ]
}
```

Performance:

- Train Accuracy: 0.9281
- Validation Accuracy: 0.8992
- Train Loss: 0.2005

- Validation Loss: 0.2995
- Validation Macro F1 Score: 0.8992

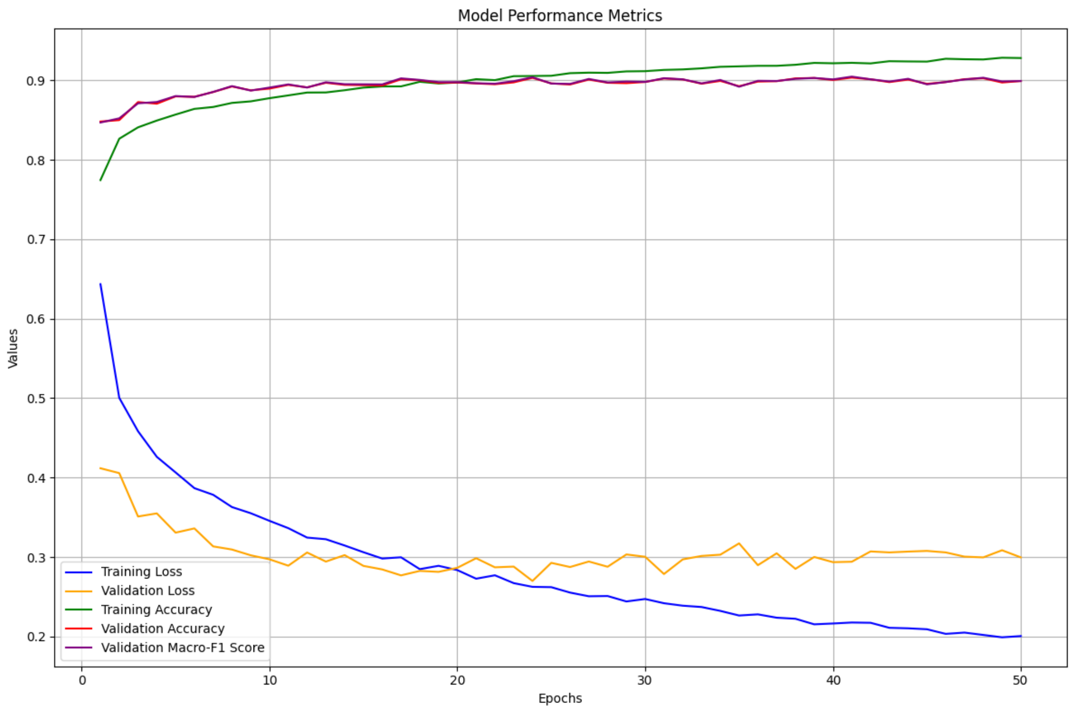

Figure 5: Training and Validation History for Model 1 (Learning Rate = 0.005)

Confusion Matrix:

Table 4: Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	542	1	7	17	1	0	25	0	4	0
1	1	600	0	7	0	0	0	0	0	0
2	17	0	539	4	26	0	25	0	0	0
3	15	2	2	533	24	0	8	0	3	0
4	2	0	60	12	514	0	37	0	2	0
5	0	0	0	0	0	603	0	10	1	7
6	108	1	63	13	28	0	399	0	7	0
7	0	0	0	0	0	7	0	535	0	8
8	4	1	3	2	2	0	7	1	570	0
9	0	0	0	0	0	7	0	23	0	560

2.2.2 Model 2 (Learning Rate = 0.0001)

Configuration:

```
config_model_2 = {
    'learning_rate': 0.0001,
    'num_epochs': 50,
    'batch_size': 32,
    'hidden_layers': [
        {'size': 512, 'bn': True, 'dropout': 0.3, 'relu': True},
        {'size': 256, 'bn': False, 'dropout': 0.4, 'relu': True},
        {'size': 128, 'bn': True, 'dropout': 0.5, 'relu': True}
    ]
}
```

Performance:

- Train Accuracy: 0.9267
- Validation Accuracy: 0.9003
- Train Loss: 0.2025
- Validation Loss: 0.2774
- Validation Macro F1 Score: 0.9010

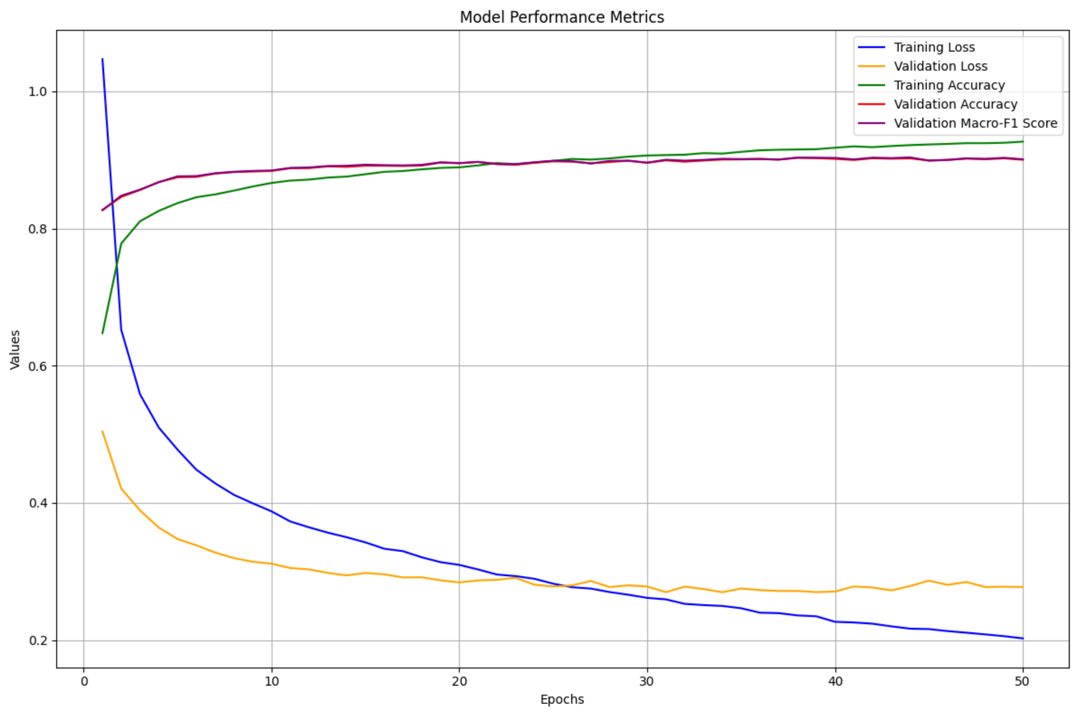

Figure 6: Training and Validation History for Model 2 (Learning Rate = 0.0001)

Confusion Matrix:

Table 5: Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	524	1	7	18	1	0	45	0	1	0
1	1	597	1	9	0	0	0	0	0	0
2	18	0	521	5	35	0	32	0	0	0
3	16	0	2	545	18	0	5	0	1	0
4	0	0	47	16	523	0	39	0	2	0
5	0	0	0	0	0	605	0	10	1	5
6	81	2	56	17	26	0	432	0	5	0
7	0	0	0	0	0	10	0	530	0	10
8	2	0	2	2	3	1	8	2	570	0
9	0	0	0	0	0	10	0	25	0	555

2.2.3 Model 3 (Learning Rate = 0.0005)

Configuration:

```
config_model_3 = {
    'learning_rate': 0.0005,
    'num_epochs': 50,
    'batch_size': 32,
    'hidden_layers': [
        {'size': 512, 'bn': True, 'dropout': 0.3, 'relu': True},
        {'size': 256, 'bn': False, 'dropout': 0.4, 'relu': True},
        {'size': 128, 'bn': True, 'dropout': 0.5, 'relu': True}
    ]
}
```

Performance:

- Train Accuracy: 0.9367
- Validation Accuracy: 0.9035
- Train Loss: 0.1724
- Validation Loss: 0.2959
- Validation Macro F1 Score: 0.9034

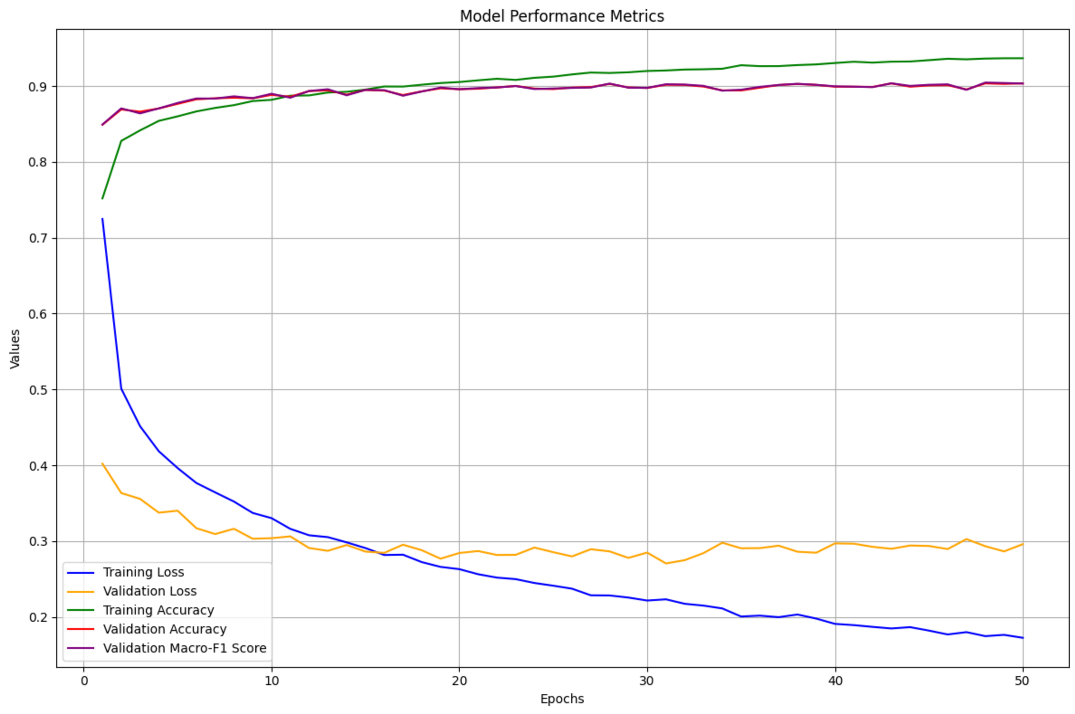

Figure 7: Training and Validation History for Model 3 (Learning Rate = 0.0005)

Confusion Matrix:

Table 6: Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	547	0	7	11	1	0	27	0	4	0
1	1	599	0	8	0	0	0	0	0	0
2	18	0	521	5	47	0	20	0	0	0
3	29	2	1	537	12	0	4	0	2	0
4	1	0	38	22	542	0	20	0	4	0
5	0	0	0	0	0	598	0	12	2	9
6	102	3	45	16	39	0	408	0	6	0
7	0	0	0	0	0	9	0	535	0	6
8	1	1	0	3	2	0	6	1	576	0
9	0	0	0	0	0	7	0	25	0	558

2.2.4 Model 4 (Learning Rate = 0.00001)

Configuration:

```
config_model_4 = {
    'learning_rate': 0.00001,
    'num_epochs': 50,
    'batch_size': 32,
    'hidden_layers': [
        {'size': 512, 'bn': True, 'dropout': 0.3, 'relu': True},
        {'size': 256, 'bn': False, 'dropout': 0.4, 'relu': True},
        {'size': 128, 'bn': True, 'dropout': 0.5, 'relu': True}
    ]
}
```

Performance:

- Train Accuracy: 0.8512
- Validation Accuracy: 0.8778
- Train Loss: 0.4281
- Validation Loss: 0.3324
- Validation Macro F1 Score: 0.8791

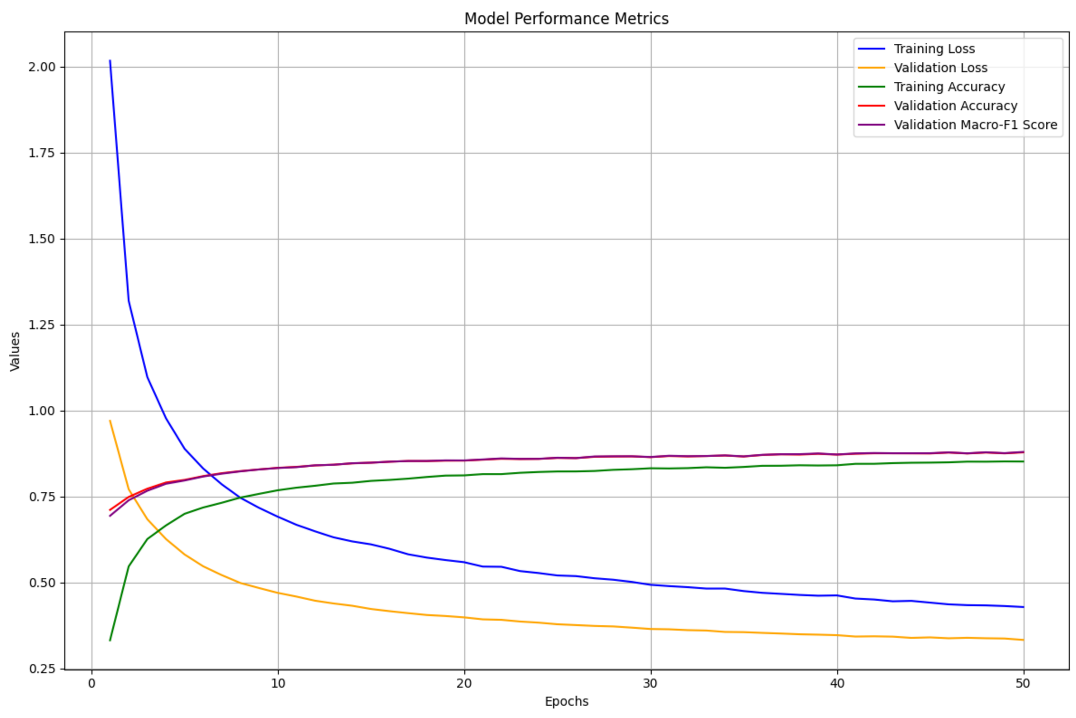

Figure 8: Training and Validation History for Model 4 (Learning Rate = 0.00001)

Confusion Matrix:

Table 7: Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	503	0	10	20	1	0	62	0	1	0
1	1	587	1	16	2	0	1	0	0	0
2	5	0	497	3	59	0	47	0	0	0
3	21	2	4	525	23	0	9	0	2	1
4	0	0	45	21	518	0	39	0	4	0
5	0	0	0	0	0	595	0	16	1	9
6	77	0	63	13	47	0	411	0	8	0
7	0	0	0	0	0	19	0	514	1	16
8	1	1	3	2	3	2	16	3	559	0
9	0	0	0	0	0	10	0	22	0	558

2.2.5 Comparison of Models with different Learning rate

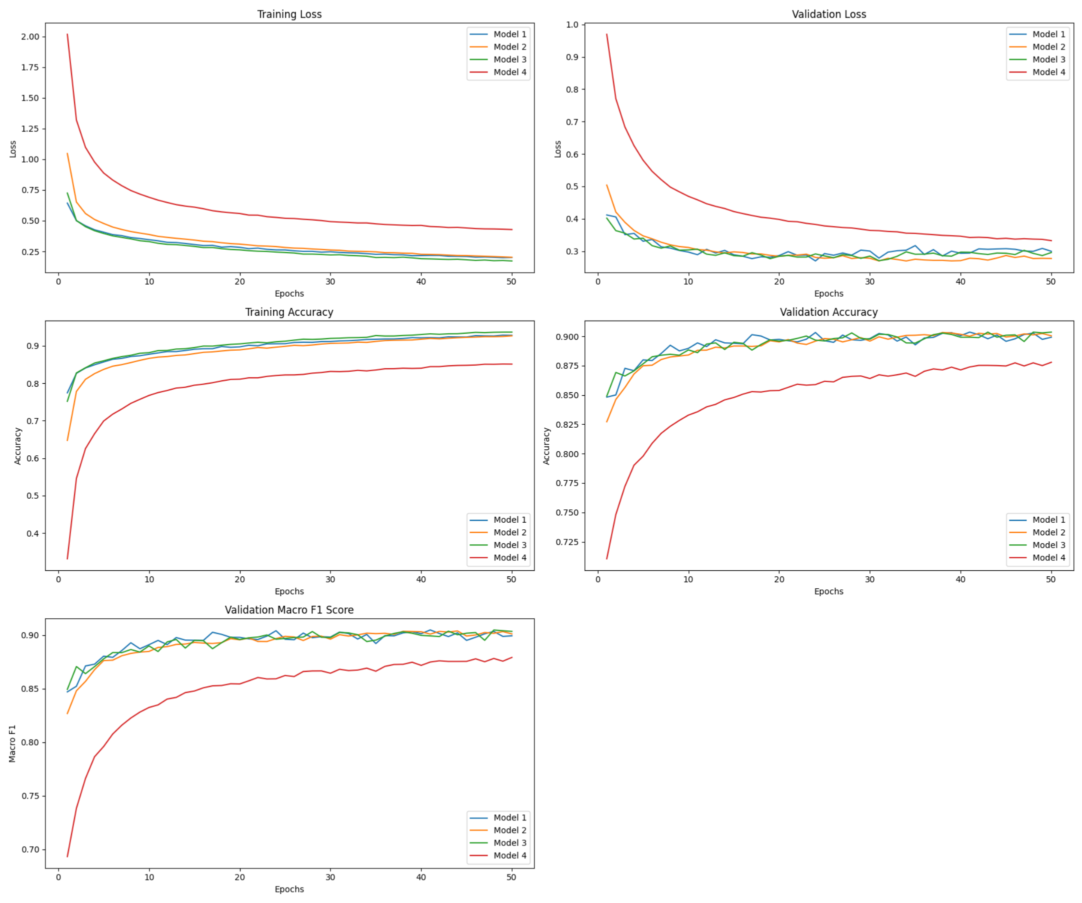

Figure 9: Comparison of Training and Validation History for All Four Models

3 Training the Final Model

The best-performing model from the validation phase was selected as the final Model.

Configuration:

```
config_model_best = {
    'learning_rate': 0.005,
    'num_epochs': 50,
    'batch_size': 32,
    'hidden_layers': [
        {'size': 512, 'bn': True, 'dropout': 0.3, 'relu': True},
        {'size': 256, 'bn': False, 'dropout': 0.4, 'relu': True},
        {'size': 128, 'bn': True, 'dropout': 0.5, 'relu': True}
    ]
}
```

Training Performance:

- Train Accuracy: 0.9283
- Train Loss: 0.1972

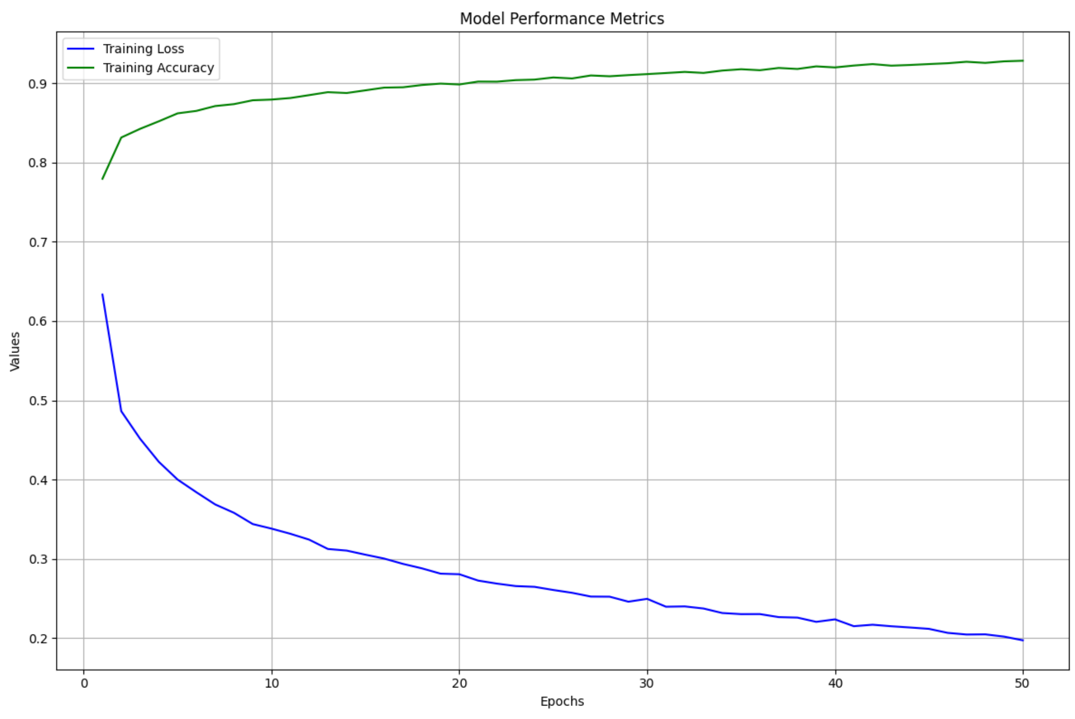

Figure 10: Training History of Best Model

4 Independent Testing Performance

Independent Testing on Final Model. **Performance:**

- Test Accuracy: 0.9066
- Test Loss: 0.3026
- Test Macro F1 Score: 0.9067

Confusion Matrix:

Table 8: Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	828	0	11	21	4	1	128	0	7	0
1	3	985	1	7	0	0	2	0	2	0
2	15	1	854	7	57	0	65	0	1	0
3	14	7	7	921	31	0	17	0	3	0
4	0	0	78	25	834	1	60	0	2	0
5	0	0	0	1	0	982	0	9	1	7
6	94	0	59	26	56	0	760	0	5	0
7	0	0	0	0	0	19	0	957	0	24
8	5	0	5	3	1	3	2	3	978	0
9	0	0	0	0	0	8	1	24	0	967