CSE-472

Report on Function Approximation with Neural Network and Backpropagation

Prepared by: Tanveer Rahman (1905025)

Introduction

In this assignment, I had to implement a Feed-Forward Neural Network (FNN) from scratch and then apply the FNN to classify the 'FashionMNIST' dataset. For the complete implementation of the FNN, I have created several classes. They are :

```
BatchNorm : Batch Normalization

Adam : Optimizer

RelU : Activation function

Dropout : Implementation of Dropout

Loss : Softmax with Cross-Entropy Loss

DenseLayer : Fully connected Layer

Flatten : Flatten the Dataset to 2D array

FNN : Feed forward Neural Network
```

Run the Code

1. **From the provided pickle file:**

   In the provided notebook, I have a section named - **'Load From Pickle'** . There is also another section named **'Train FNN**'.

   Uncomment 'Load From Pickle' and **comment 'Train FNN'** and then run all the sections. Then we can find the Test output of the model in the '**Test FNN'** section.

2. **Without the pickle file:**

   If we want to run and observe the training of the FNN, then **Comment 'Load From Pickle**' and Un**comment 'Train FNN'** and then run all the sections.

   Finally **Uncomment 'Store in Pickle'** section, and then store the weights of the model in the pickle file.

Observations

I have prepared Three models, where the number of hidden layers are varied.

- One hidden layer
- Two hidden layers
- three hidden layers

Model 1 : One hidden layer

For **iteration = 10** and **batch size = 128 and learning rate = 0.0005**, the model performs the best.

Iteration	Train Loss	Train Accuracy	Val Loss	Val Accuracy	Val F1
1	0.6001	0.7861	0.4901	0.8229	0.8225
2	0.4648	0.8330	0.4579	0.8385	0.8394
3	0.4273	0.8474	0.4378	0.8433	0.8429
4	0.4110	0.8513	0.4227	0.8506	0.8513
5	0.3981	0.8557	0.4194	0.8490	0.8489
6	0.3868	0.8608	0.4129	0.8546	0.8543
7	0.3796	0.8615	0.4004	0.8572	0.8580
8	0.3733	0.8651	0.3995	0.8565	0.8576
9	0.3647	0.8664	0.4001	0.8576	0.8587
10	0.3568	0.8706	0.3981	0.8558	0.8565

Model 2 : Two hidden layers

For **iteration = 10** and **batch size = 128**, when **learning rate = 0.0005,** the model performs the best.

Iteration	Train Loss	Train Accuracy	Val Loss	Val Accuracy	Val F1
1	0.6255	0.7762	0.4892	0.8277	0.8266
2	0.4709	0.8304	0.4423	0.8441	0.8424
3	0.4338	0.8437	0.4247	0.8495	0.8488
4	0.4144	0.8508	0.4164	0.8554	0.8532
5	0.3971	0.8561	0.3990	0.8569	0.8563
6	0.3863	0.8586	0.3988	0.8620	0.8610
7	0.3749	0.8630	0.3900	0.8608	0.8601
8	0.3666	0.8679	0.3872	0.8665	0.8663
9	0.3611	0.8700	0.3740	0.8688	0.8690
10	0.3561	0.8704	0.3785	0.8662	0.8654

Model 3 : Three hidden layers

For **iteration = 10** and **batch size = 128**, when **learning rate = 0.001,** the model performs the best.

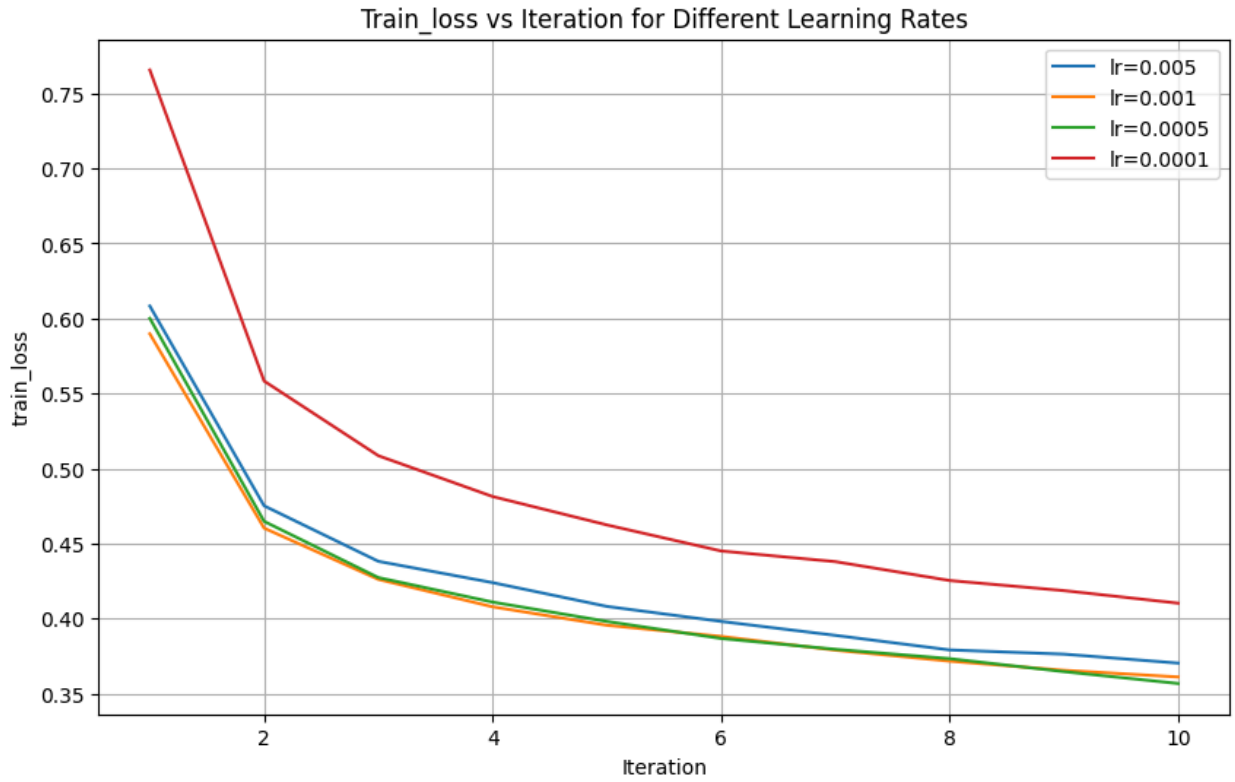
Iteration	Train Loss	Train Accuracy	Val Loss	Val Accuracy	Val F1
1	0.9885	0.6501	0.7223	0.7488	0.7462
2	0.6786	0.7681	0.6292	0.7945	0.7916
3	0.6117	0.7984	0.5948	0.8023	0.8012
4	0.5754	0.8140	0.5614	0.8174	0.8157
5	0.5439	0.8236	0.5406	0.8303	0.8278
6	0.5371	0.8269	0.5312	0.8321	0.8321
7	0.5206	0.8330	0.5154	0.8381	0.8365
8	0.5052	0.8384	0.5128	0.8387	0.8390
9	0.5022	0.8391	0.5113	0.8405	0.8387
10	0.4948	0.8429	0.4896	0.8406	0.8393

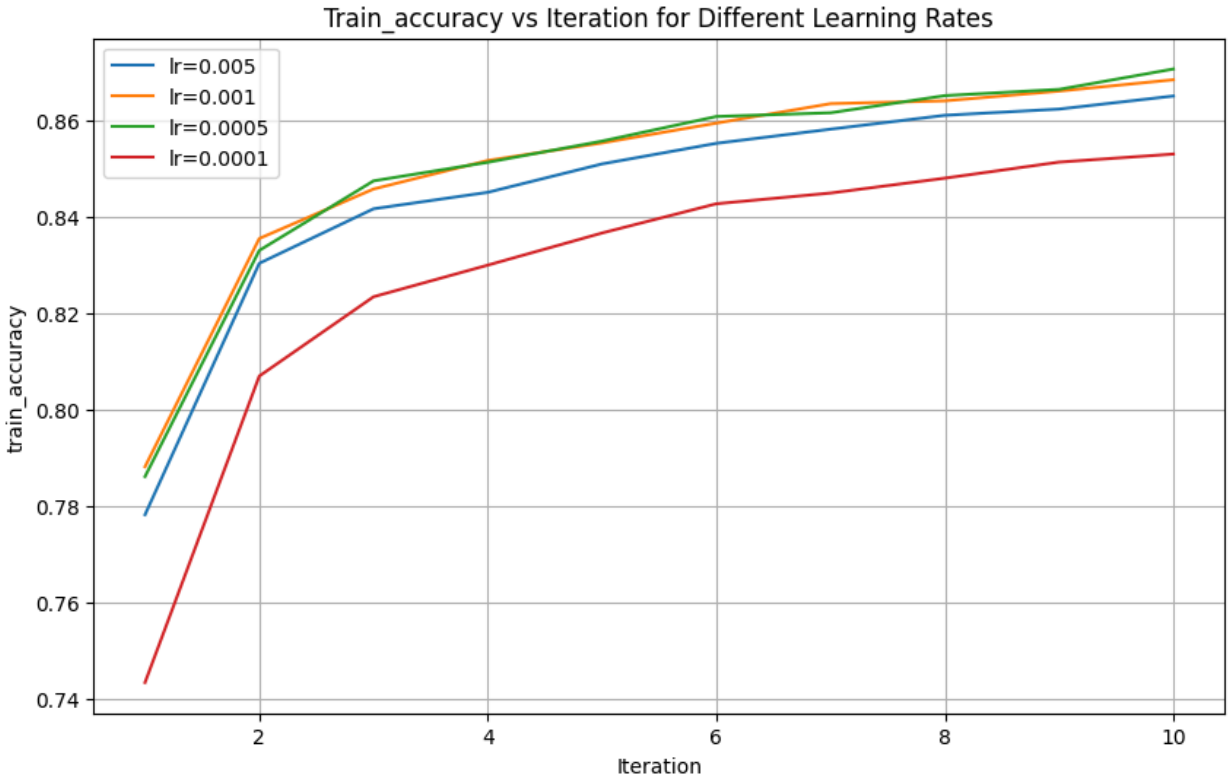
Graphs

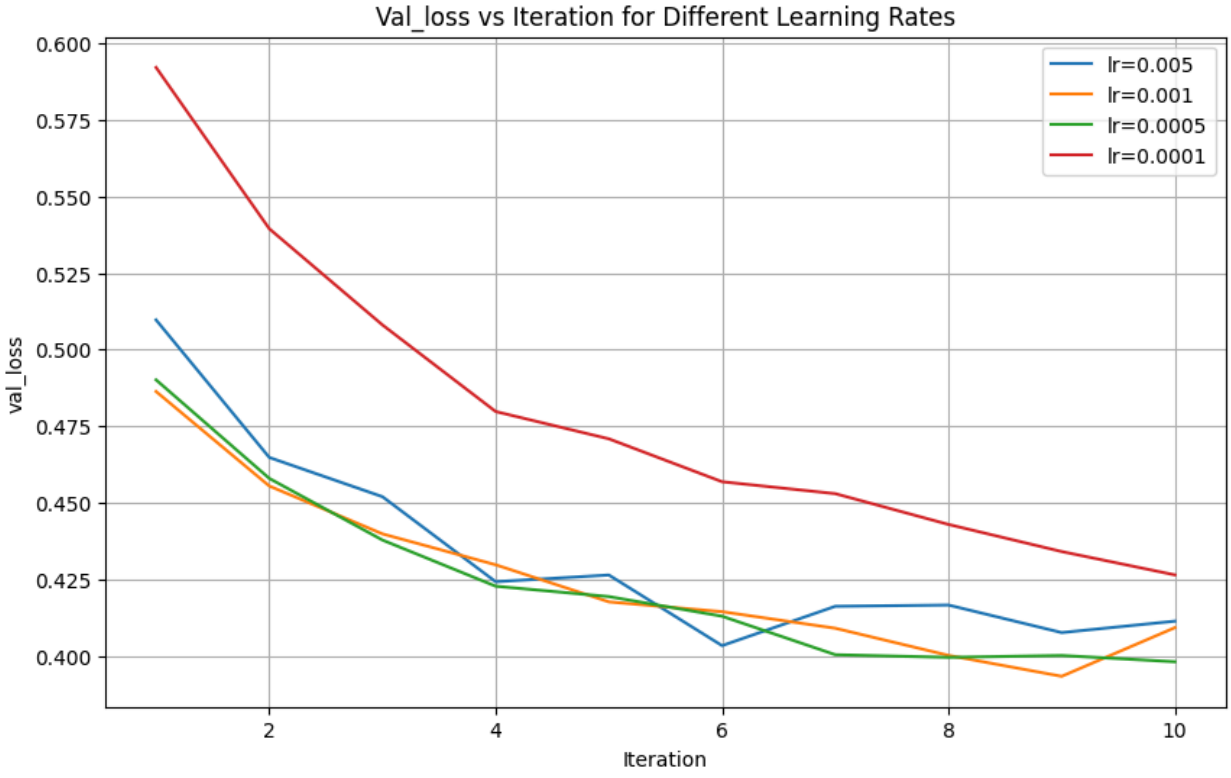
The graphs are constructed considering learning rates [0.005, 0.001], for each of the three models.

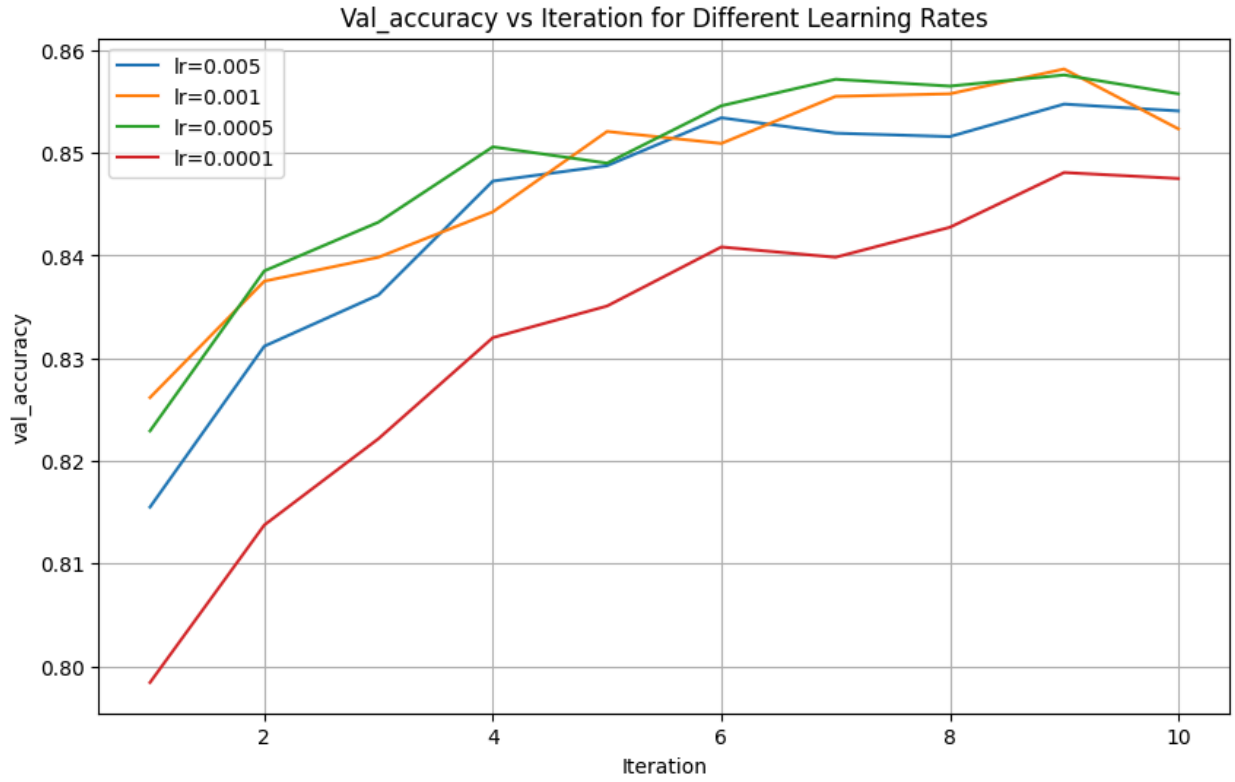
Model 1 : One hidden layer

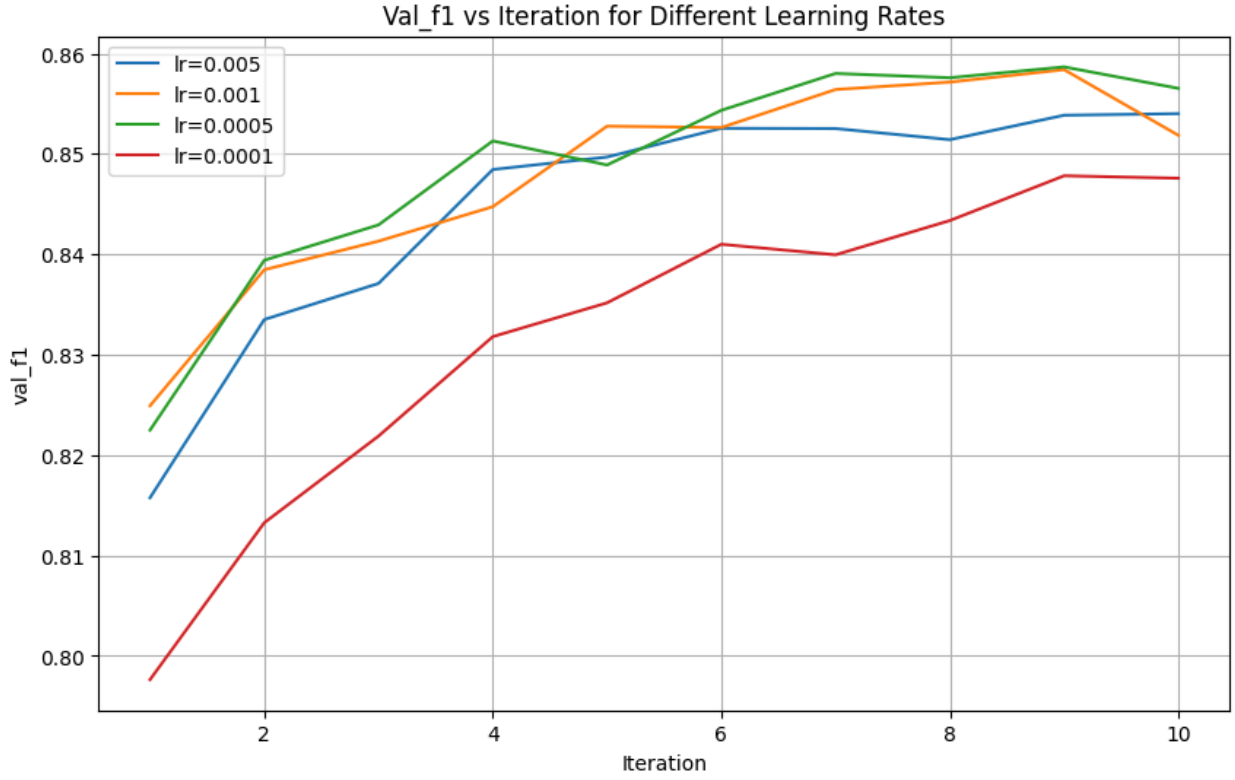
- **One hidden layer**
- **Number of iterations in training = 10**
- **Batch size = 128**




Train_accuracy vs Iteration for Different Learning Rates
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001
train_accuracy
Iteration
0.74
0.76
0.78
0.80
0.82
0.84
0.86
2
4
6
8
10

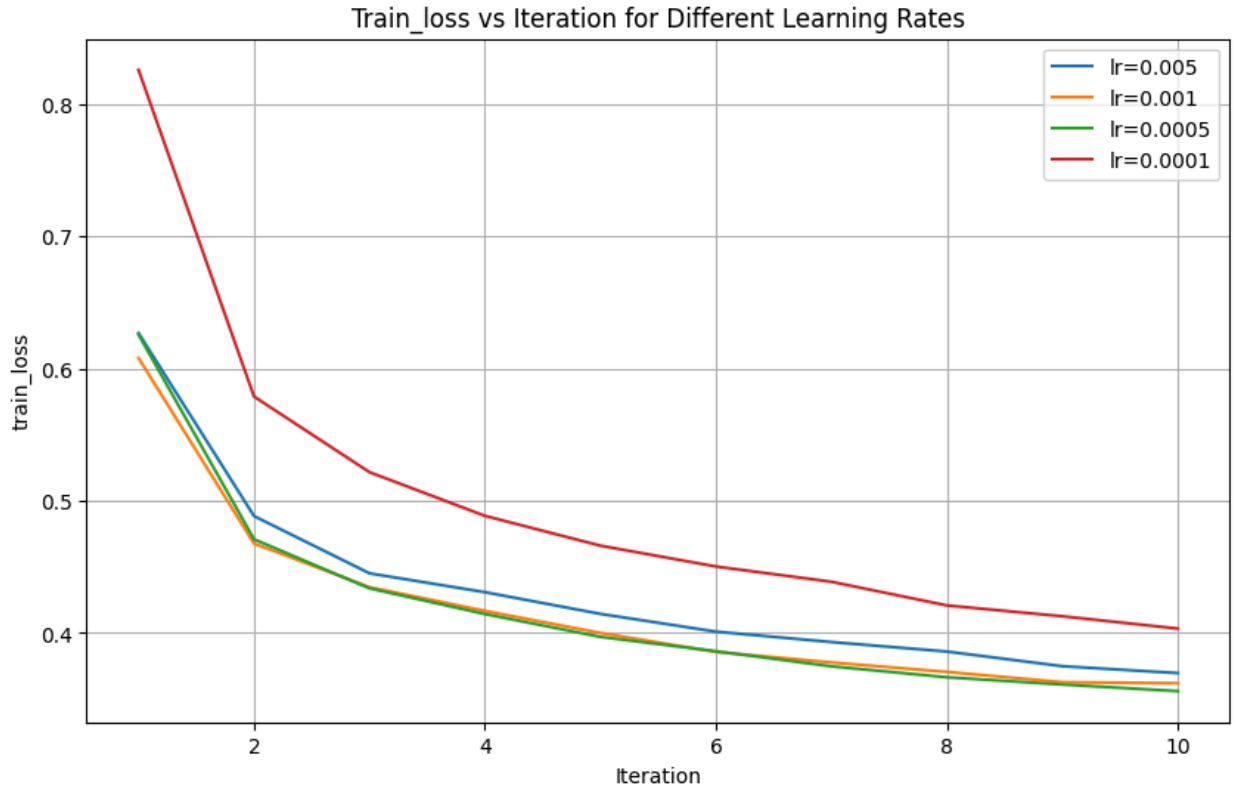

Val_loss vs Iteration for Different Learning Rates
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001
val_loss
Iteration
0.400
0.425
0.450
0.475
0.500
0.525
0.550
0.575
0.600
2
4
6
8
10

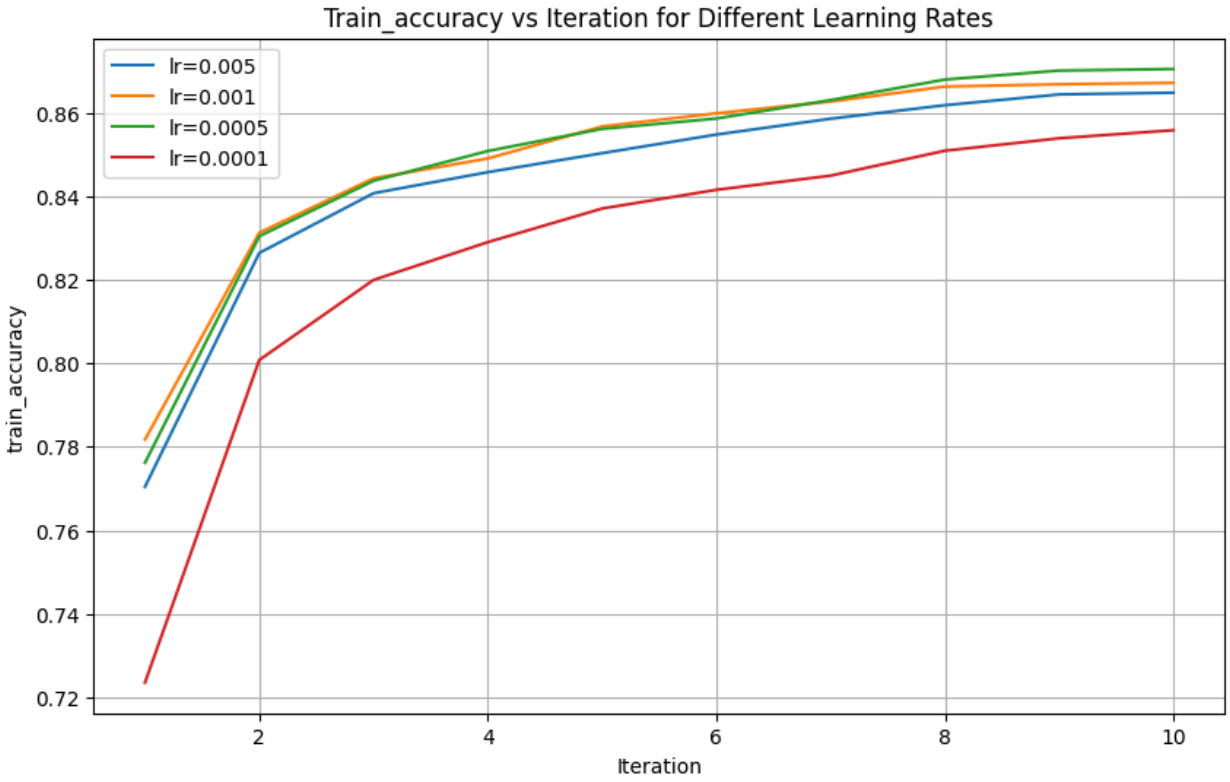

Val_accuracy vs Iteration for Different Learning Rates
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001
val_accuracy
Iteration
0.86
0.85
0.84
0.83
0.82
0.81
0.80
2
4
6
8
10

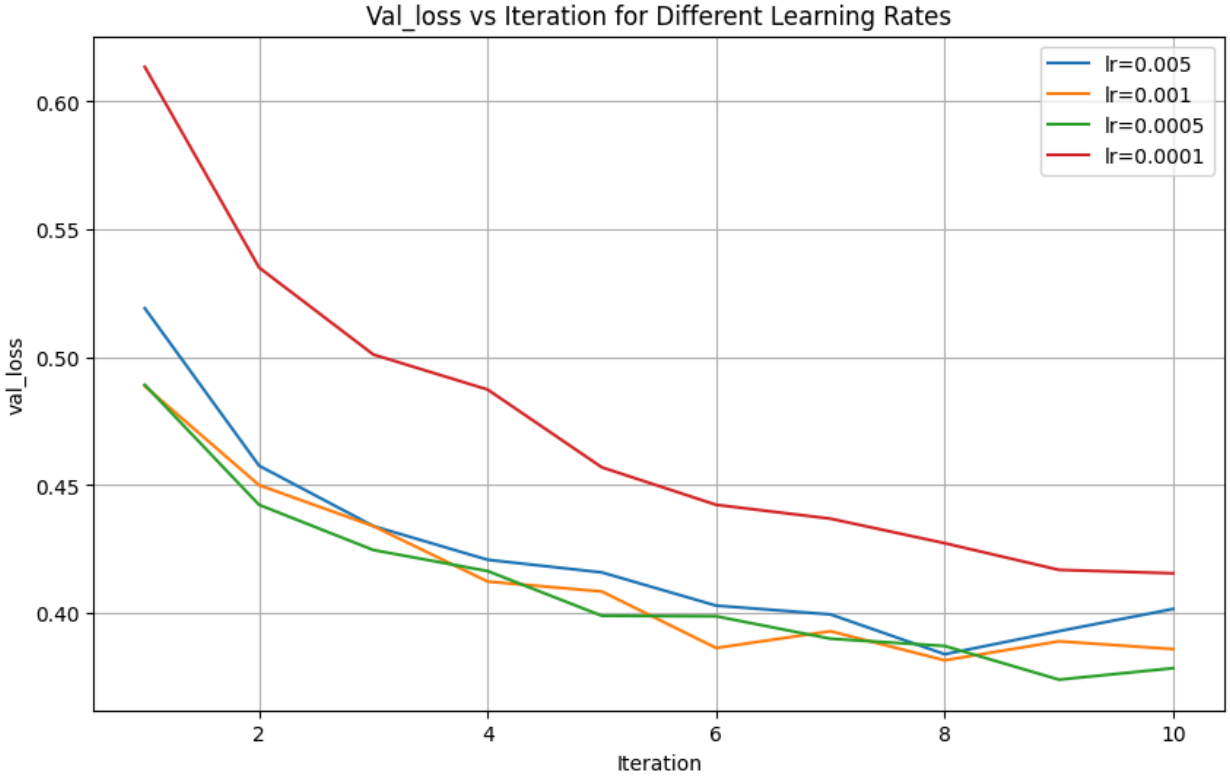

Val_f1 vs Iteration for Different Learning Rates
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001
val_f1
Iteration
0.86
0.85
0.84
0.83
0.82
0.81
0.80
2
4
6
8
10

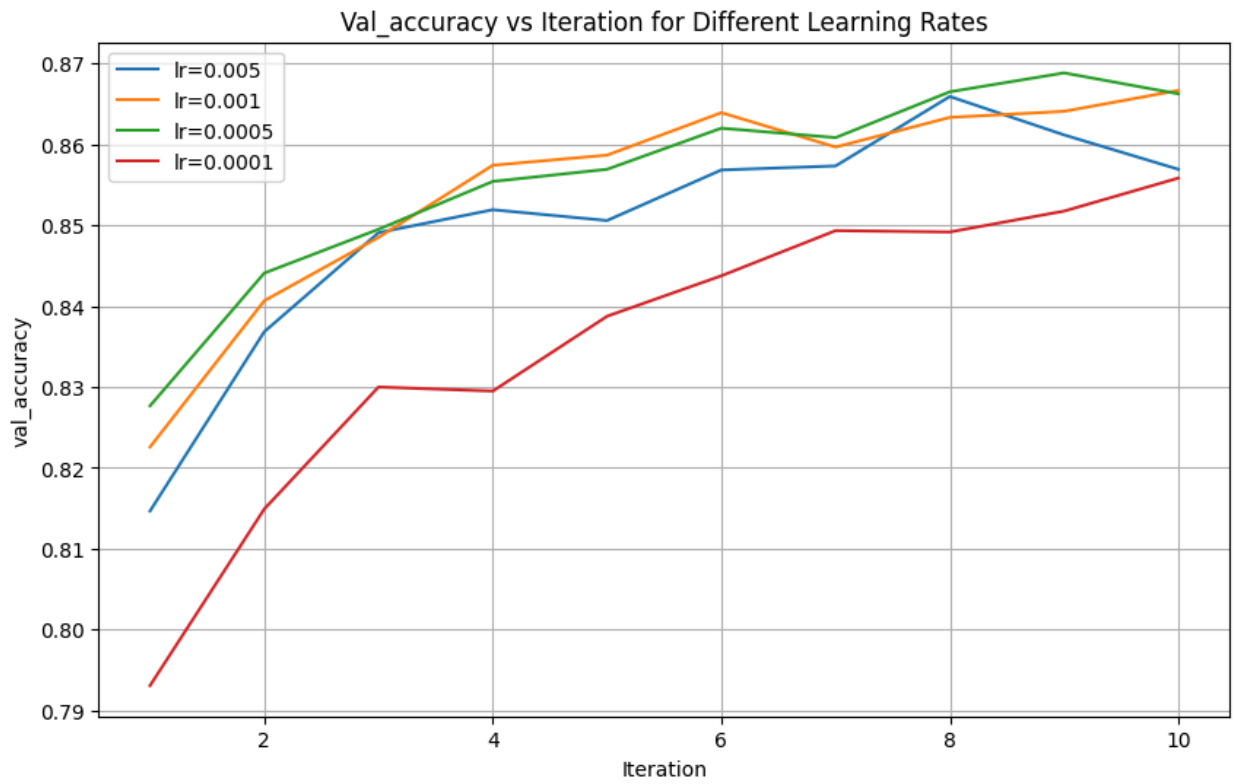
Model 2 : Two hidden layers

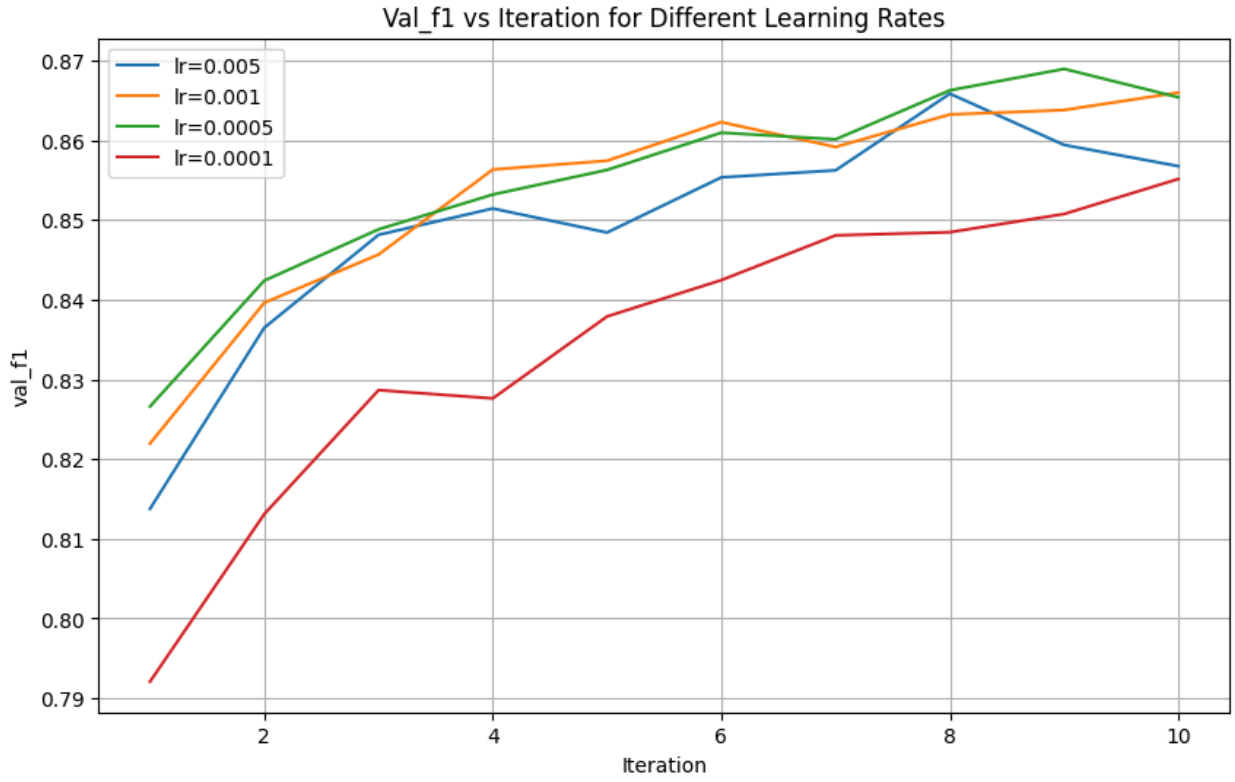
- **Two hidden layers**
- **Number of iterations in training = 10**
- **Batch size = 128**




Train_accuracy vs Iteration for Different Learning Rates
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001
train_accuracy
Iteration
0.72
0.74
0.76
0.78
0.80
0.82
0.84
0.86
2
4
6
8
10

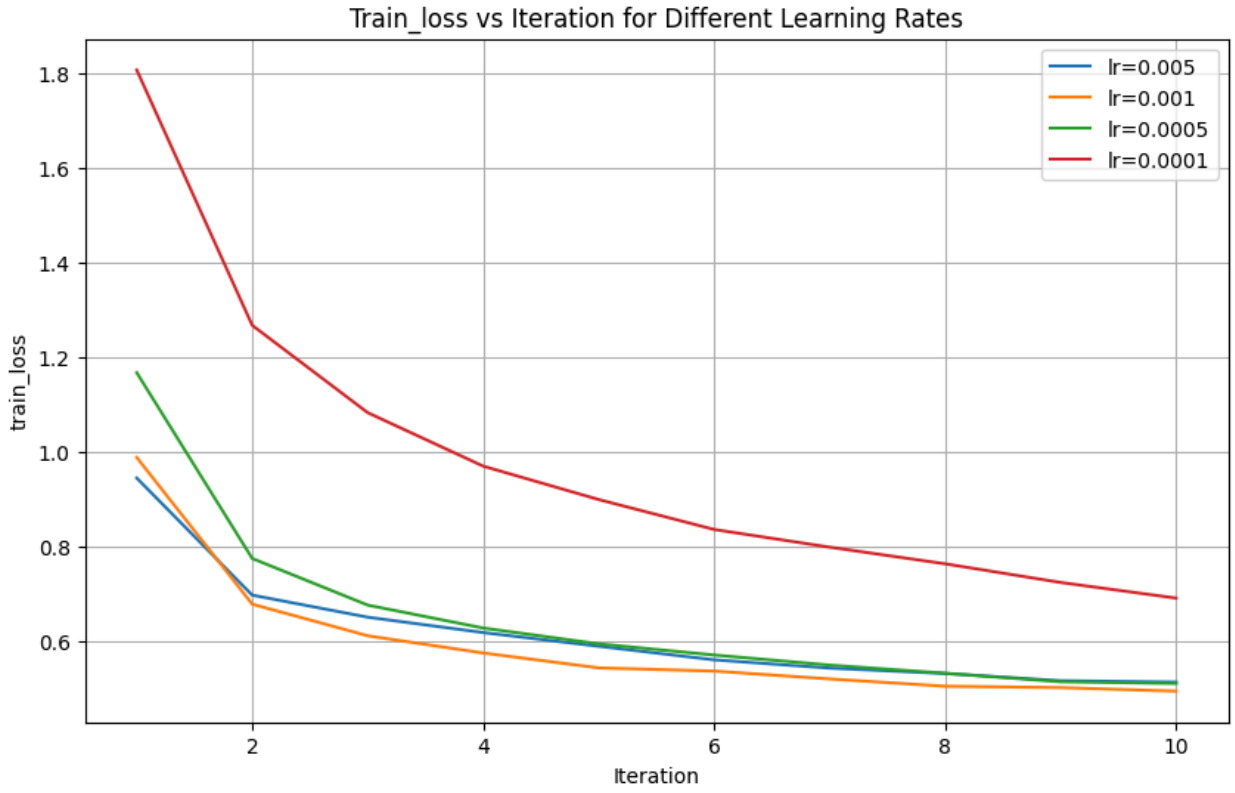

Val_loss vs Iteration for Different Learning Rates
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001
val_loss
Iteration
0.40
0.45
0.50
0.55
0.60
2
4
6
8
10

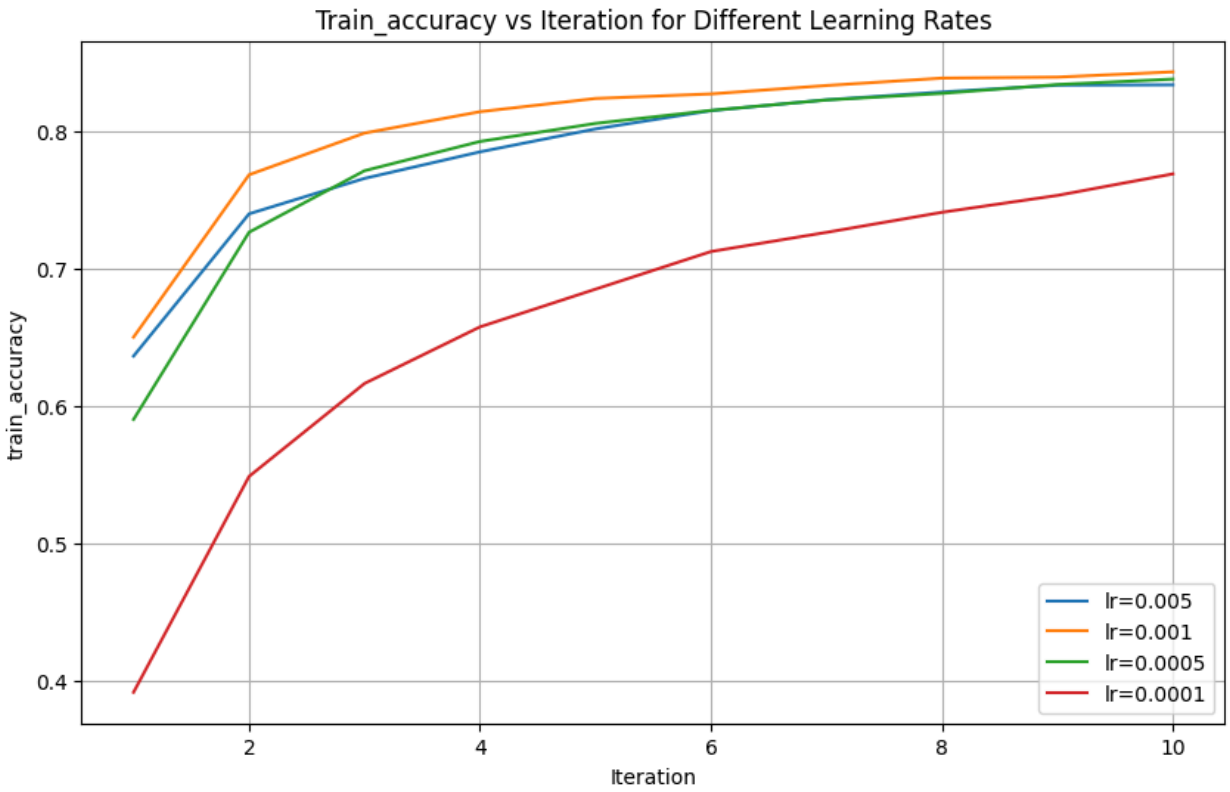

Val_accuracy vs Iteration for Different Learning Rates
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001
val_accuracy
Iteration
0.87
0.86
0.85
0.84
0.83
0.82
0.81
0.80
0.79
2
4
6
8
10

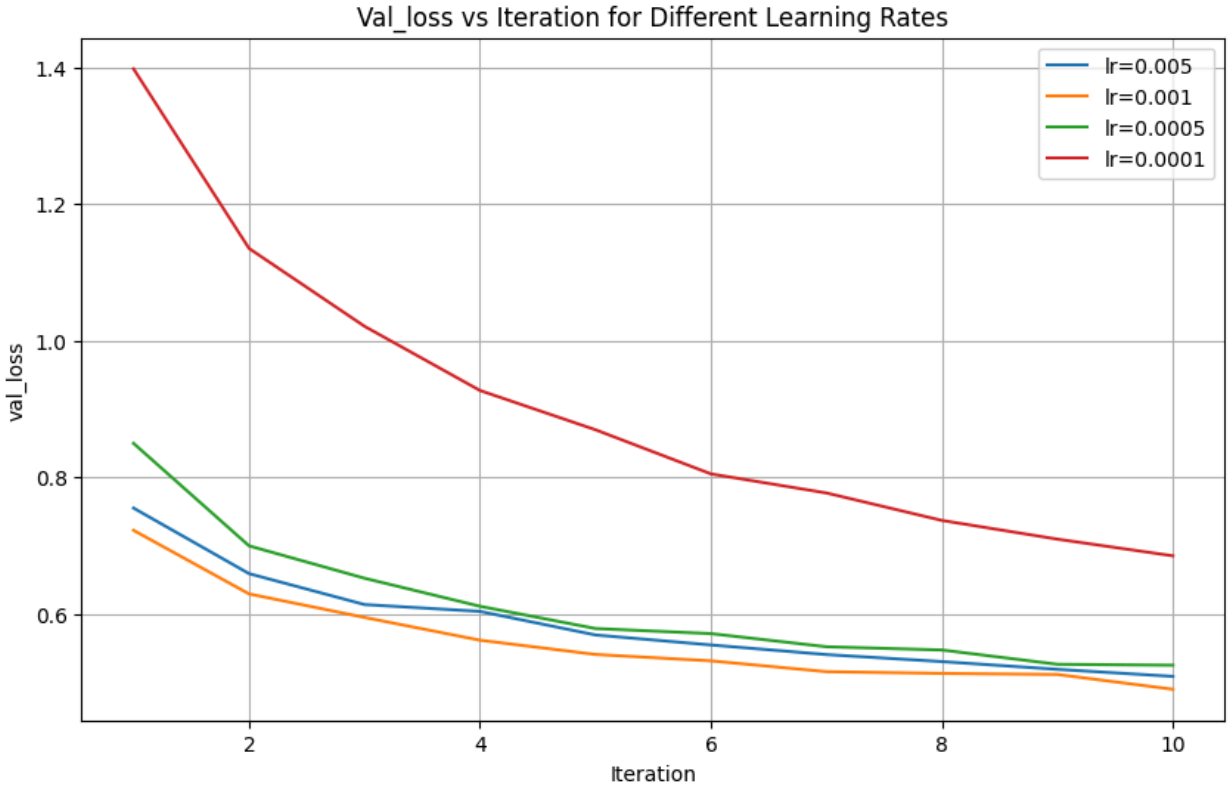

Val_f1 vs Iteration for Different Learning Rates
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001
val_f1
Iteration
0.87
0.86
0.85
0.84
0.83
0.82
0.81
0.80
0.79
2
4
6
8
10

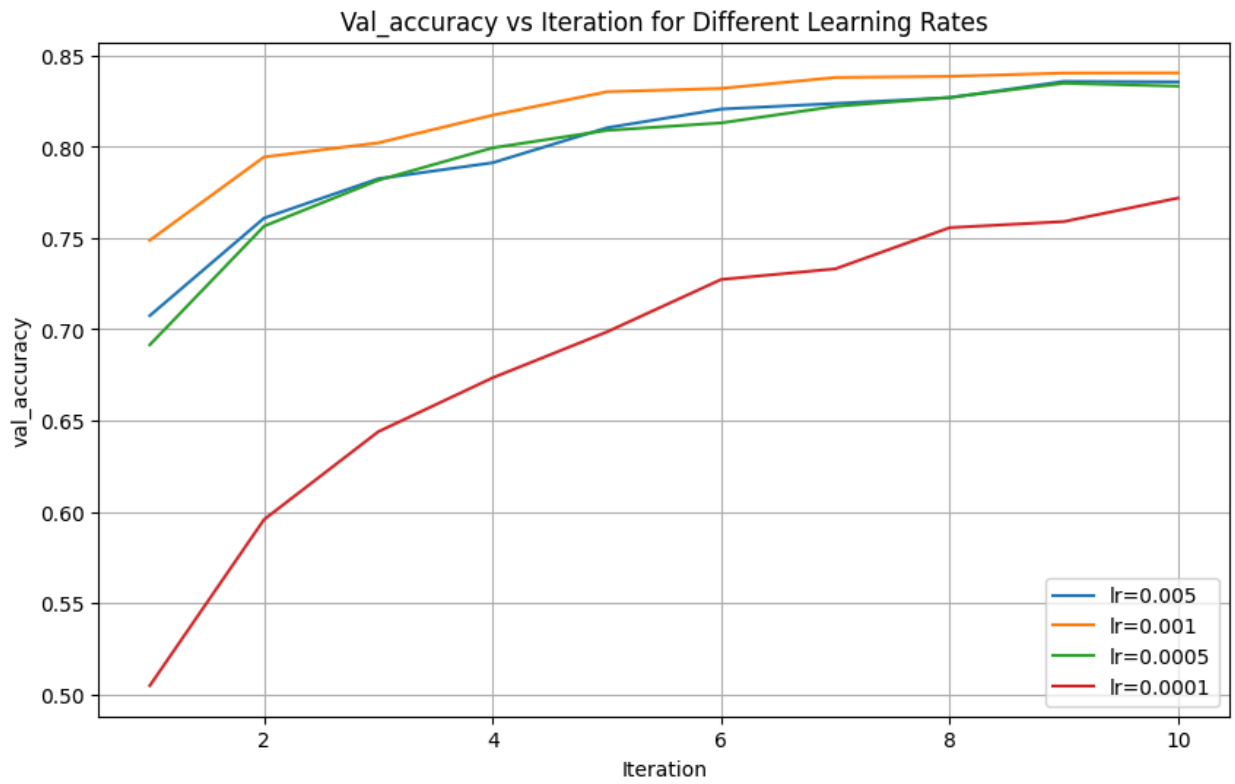
Model 3 : Three hidden layers

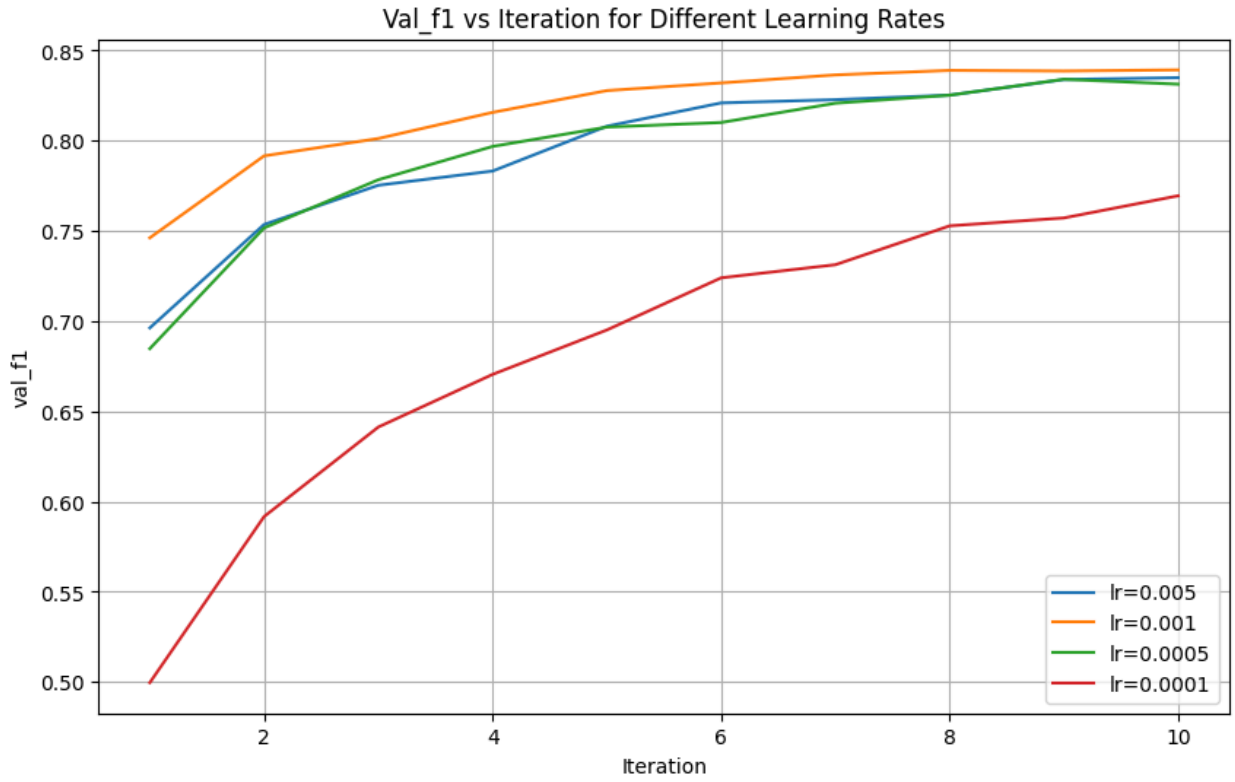
- **Two hidden layers**
- **Number of iterations in training = 10**
- **Batch size = 128**




Train_accuracy vs Iteration for Different Learning Rates
train_accuracy
Iteration
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001


Val_loss vs Iteration for Different Learning Rates
val_loss
Iteration
0.6
0.8
1.0
1.2
1.4
2
4
6
8
10
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001


Val_accuracy vs Iteration for Different Learning Rates
val_accuracy
Iteration
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
2
4
6
8
10
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001


Val_f1 vs Iteration for Different Learning Rates
val_f1
Iteration
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
2
4
6
8
10
lr=0.005
lr=0.001
lr=0.0005
lr=0.0001

Confusion Matrix

In the Confusion matrix:

- The diagonal elements represent the number of points for which test_label = true_label
- The mislabelled elements are the off-diagonal elements.

The confusion matrix for all the three models are given below:

Model 1 : One hidden Layer

```
[[ 976    2   23   47    2    0   88    0    8    1]
 [   9 1131    2   32    4    0    2    0    1    0]
 [  19    4  921   18  171    1   93    0    5    0]
 [  44    7    6 1051   42    0   27    0    6    0]
 [   4    2  102   33  986    0   78    0    7    0]
 [   1    1    1    2    1 1155    1   51    7   28]
 [ 204    2  142   40  131    0  696    0   15    0]
 [   0    0    0    0    0   40    0 1123    1   26]
 [   5    1    5   10    5    3   12   10 1167    1]
 [   0    0    1    0    0    7    0   55    1 1094]]
```

Model 2 : Two hidden Layers

```
[[1034    6   27   62    4    1  129    0   11    0]
 [   5 1157    6   28    4    0    2    1    0    1]
 [  11    0  908   11  141    0  100    0    6    0]
 [  44   14   18 1008   51    0   33    1    3    0]
 [   4    3  143   26  952    0   74    0    4    1]
 [   0    2    0    1    1 1092    1   59    7   28]
 [ 182    2  124   27  109    0  719    1   17    0]
 [   0    0    0    1    0   31    0 1099    1   42]
 [   4    3   12    8    4    5   17    3 1155    1]
 [   0    0    1    0    0   13    0   65    0 1129]]
```

Model 3 : Three hidden Layers

```
[[ 904    5   27   83   14    3  164    1   16    4]
 [   7 1127    2   32    9    1    6    0    8    2]
 [  25    5  713   12  214    2  186    1   30    0]
 [  77   37   10  994   49    2   51    4    7    2]
 [   4   11  149   44  829    4  112    0   17    2]
 [   4   14    2    4    0 1047    0   73   11   68]
 [ 244    5  239   48  165    1  487    1   25    1]
 [   0    1    0    3    2   60    0 1013    5  107]
 [   9   10   26    3    9    8   16    6 1077    6]
 [   4    4    2    1    1   34    1   55    4 1086]]
```

Conclusion

From the above discussion,

- For **model 1**, when **learning rate = .0005**, at the **8 th iteration** the validation f1-score becomes the highest **.8587**
- For **model 2**, when **learning rate = .0005**, at the **9 th iteration** the validation f1-score becomes the highest **.8690**
- For **model 3**, when **learning rate = .0005**, at the **10 th iteration** the validation f1-score becomes the highest **.8393**

So, I have selected model 2 as my best model and the learning rate is set to .0005.

Test Result for the selected model:

Test loss : 0.4180

Test accuracy : 0.8531

Test f1-score : 0.8523