CSE 343: Machine Learning

Assignment 1: Report

Aditya Aggarwal **2022028**

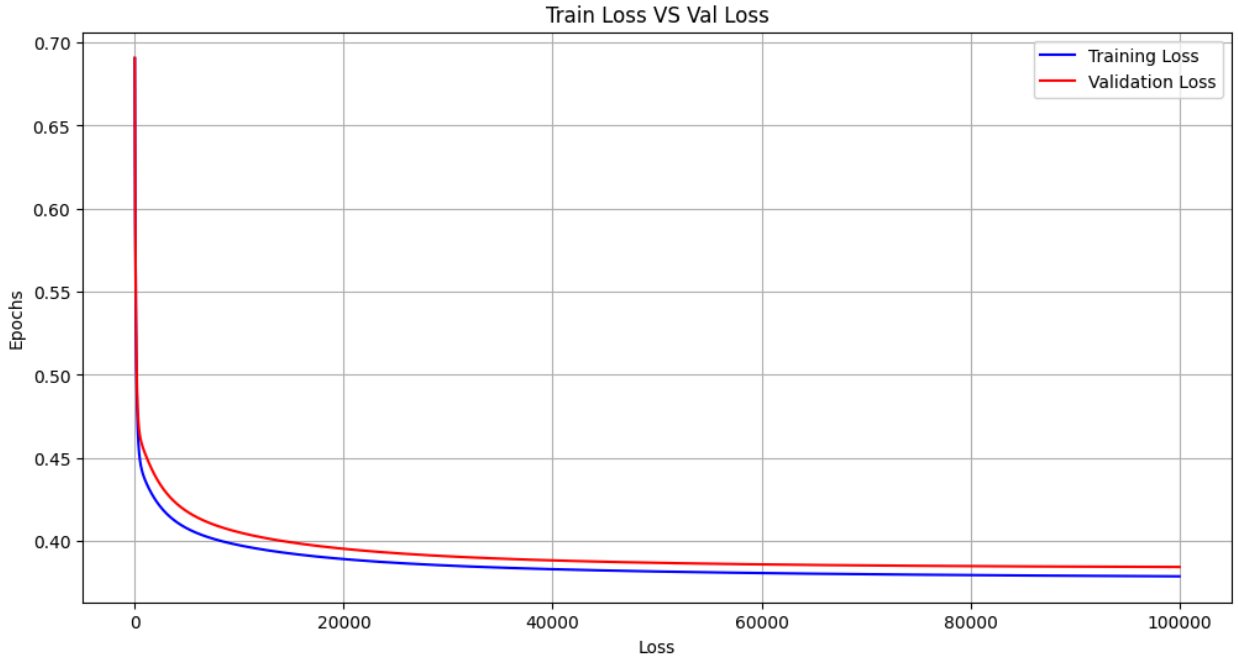
SECTION B

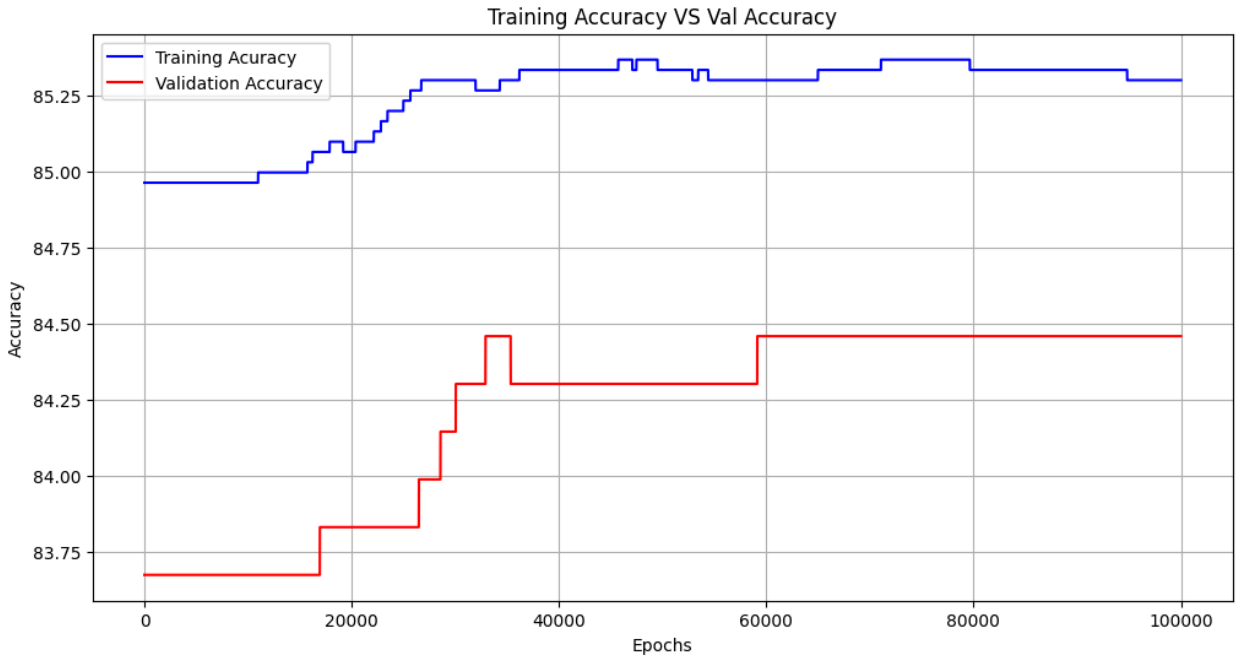
Data Preprocessing

- The dataset was shuffled with a fixed seed to avoid any unfortunate splits.
- There were originally 645 missing values across the dataset. For attributes with binary entries, the missing values were replaced by the mode of all values in that attribute. For other attributes, the missing values were replaced by the mean.
- As instructed, the dataset was split into 70:15:15 (train: test: validation).
- Min-max scaling with a feature range of $(0,1)$ was applied to a copy of this dataset for better results. (The unscaled set is also saved for comparison in part B).
- Bias was considered by adding a column of 1s to all the datasets – train, val, and test set.

Part A

To implement and train the logistic regression model on the dataset, a learning rate of 0.01, and the model was trained for 100,000 epochs. The plots of ‘Train Loss vs Val Loss’ and ‘Train Accuracy vs Val Accuracy’ are provided below.

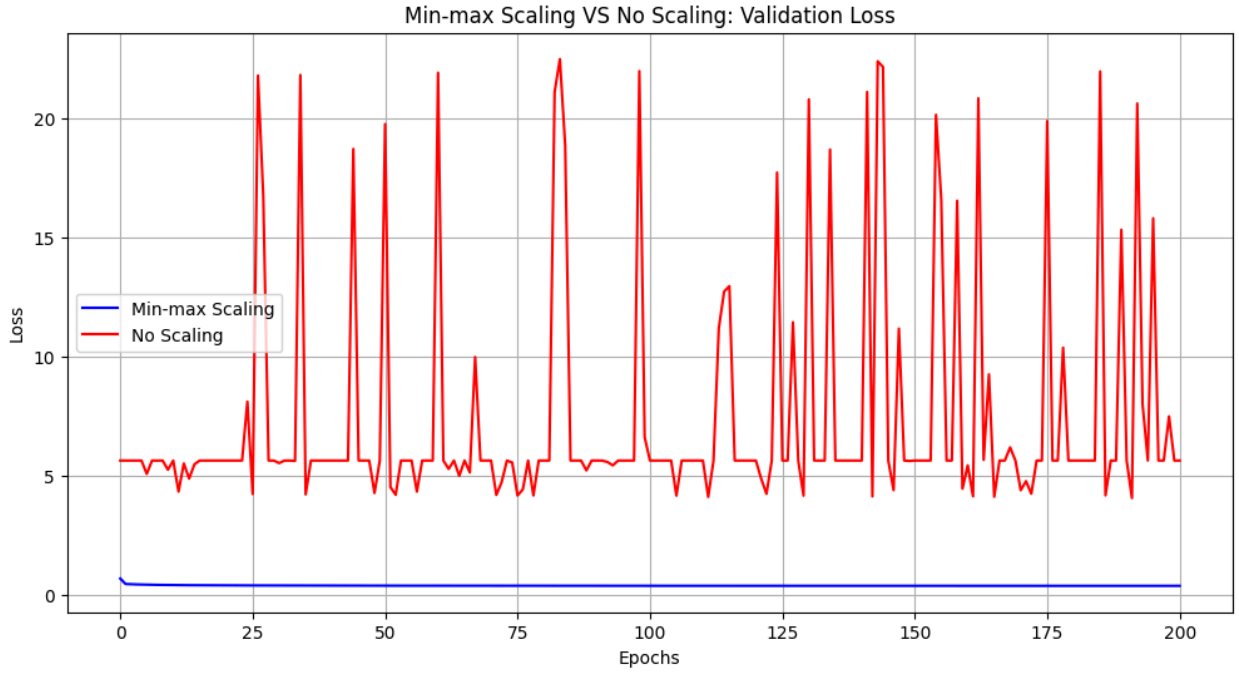




As the number of epochs increases, the train loss and the val loss decrease, the train accuracy and the val accuracy increase (slightly), and the model converges. The val loss is slightly higher than the train loss, and the val accuracy is slightly lower than the train accuracy. The minimal difference in both curves shows that the model can generalize well on unseen data.

Part B

The graph of loss against the number of epochs was plotted, once for an unscaled dataset and once for a min-max scaled dataset. This allowed us to do a comparative analysis of the effect of scaling on the dataset and the model outcome. The plots are provided below.



It can be seen that while the loss-epoch curve for the scaled dataset is smooth and monotonic, the unscaled dataset has an erratic curve. This might be because when features are scaled to a uniform range, the gradient magnitudes are more consistent across features. The algorithm struggles to optimize weights with features of vastly different scales. Thus, working with the scaled dataset is more efficient in obtaining meaningful results.

Part C

```
Precision: 85.71%
Recall: 5.77%
F1 Score: 10.81%
ROC-AUC Score: 52.79%
```

- Precision: 85.71%

  The proportion of true positive predictions among all positive predictions made by the model is fairly high. This suggests that when the model predicts a positive outcome, it will likely be a correct prediction.

- Recall: 5.77%

  The proportion of true positive cases correctly identified by the model out of all actually positive cases is much less. This suggests that the model predicts a lot of false negatives. This might hint at an imbalance in the dataset.

- F1 Score: 10.81%

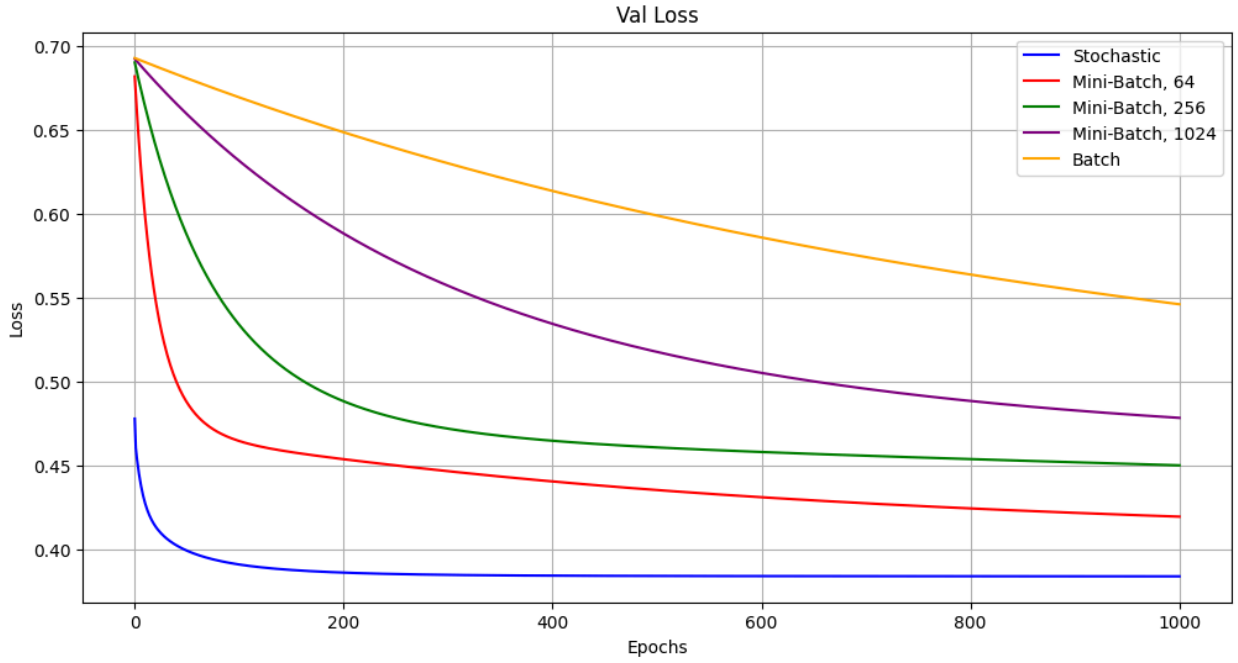
  A ten percent F1 score is also fairly low. It reflects the imbalance between precision and recall, confirming the huge gap between these two quantitative measures. It can be said with surety that the dataset is imbalanced, with most labels being negative. Hence, the model has developed a bias towards mostly negative predictions.

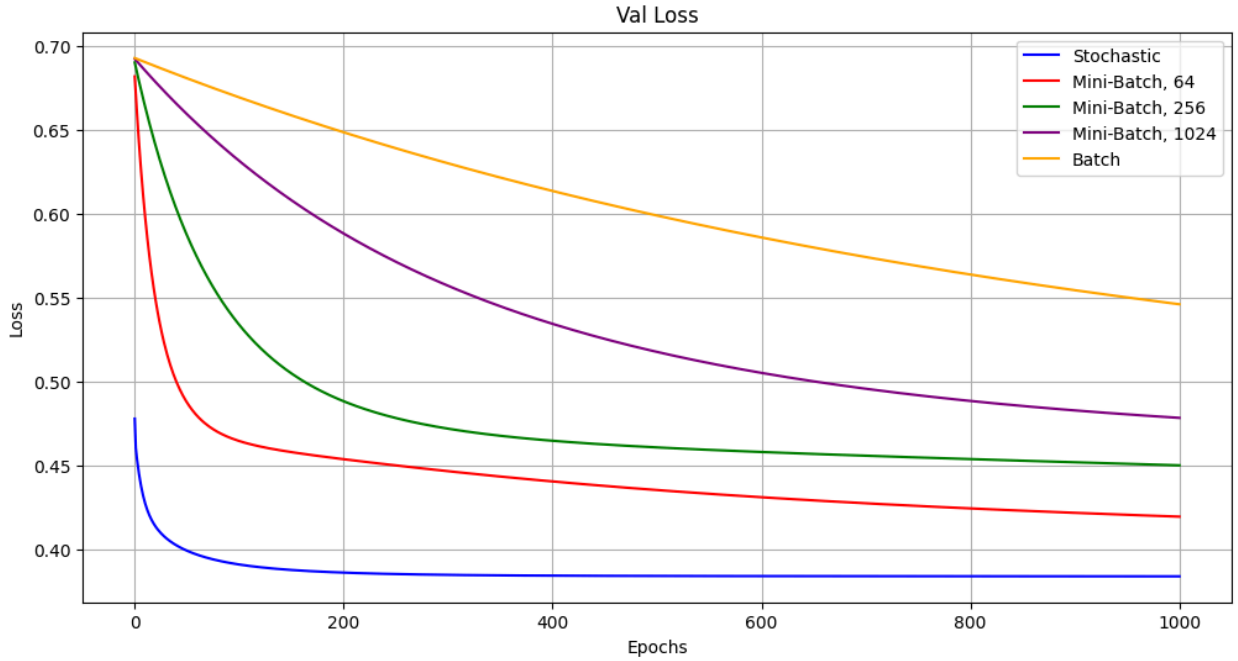
- ROC-AUC Score: 52.79%

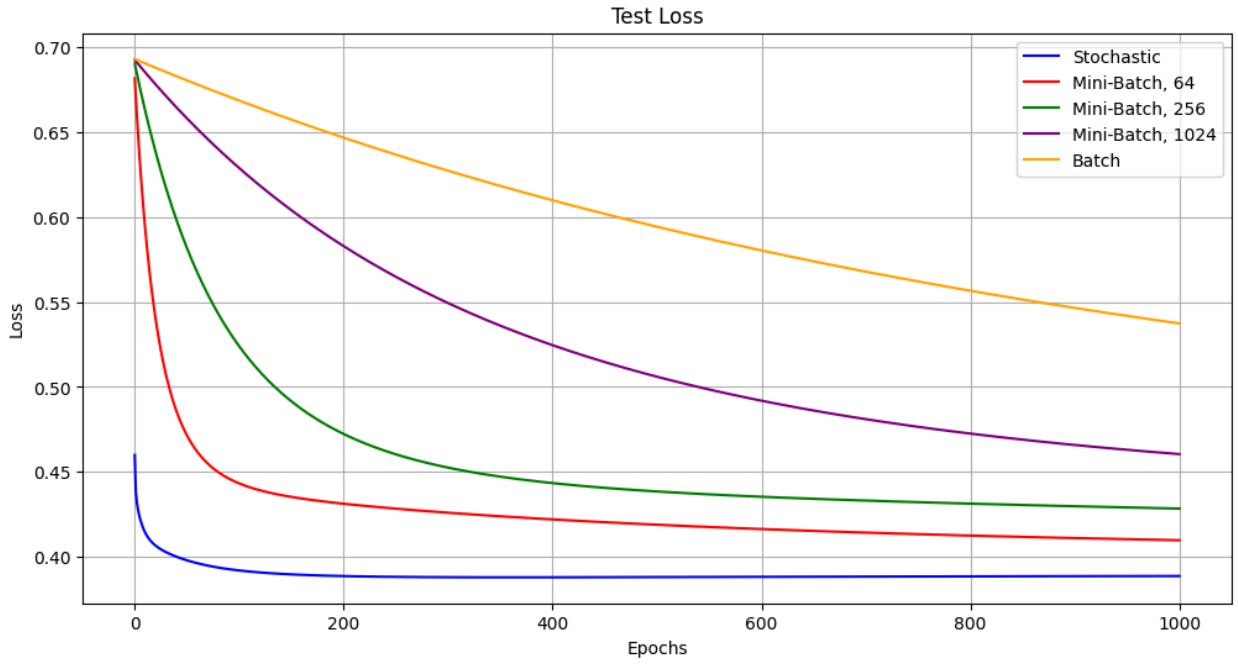
  The area under the Receiver Operating Characteristic curve measures performance across all classification thresholds. The score is close to 50%, indicating that the model is almost randomly guessing between the two labels. Its high accuracy is only due to the imbalance being propagated in the test set. On a balanced unseen dataset, the model might give poor performance.

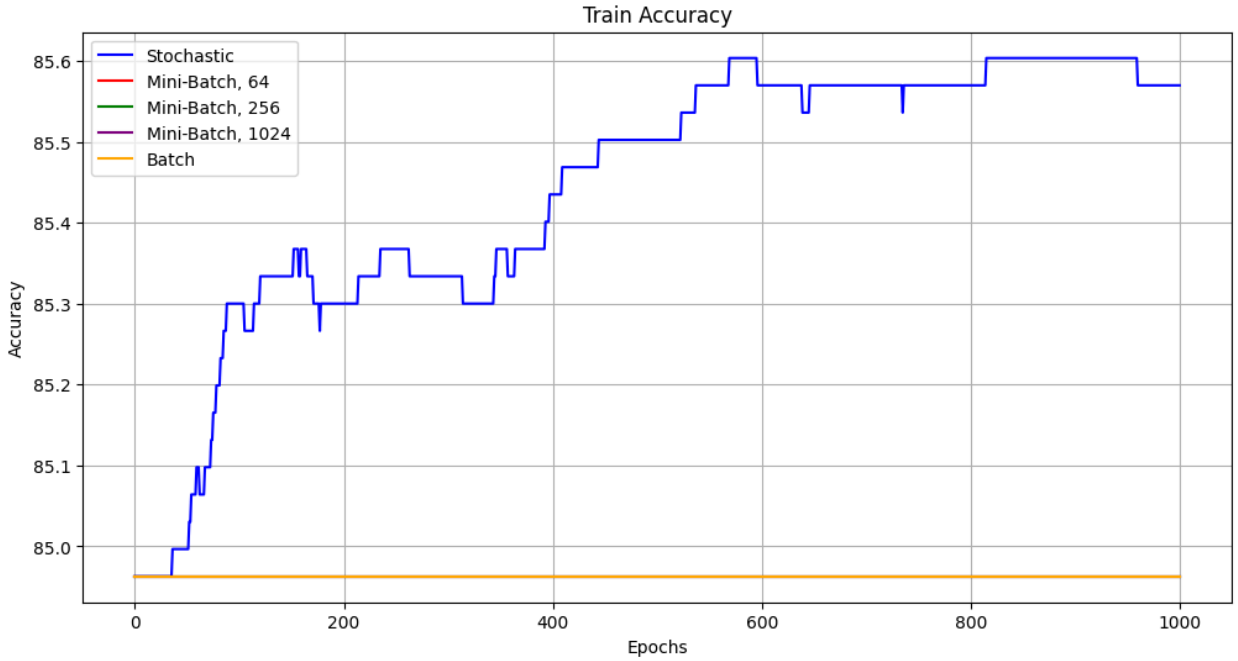
Part D

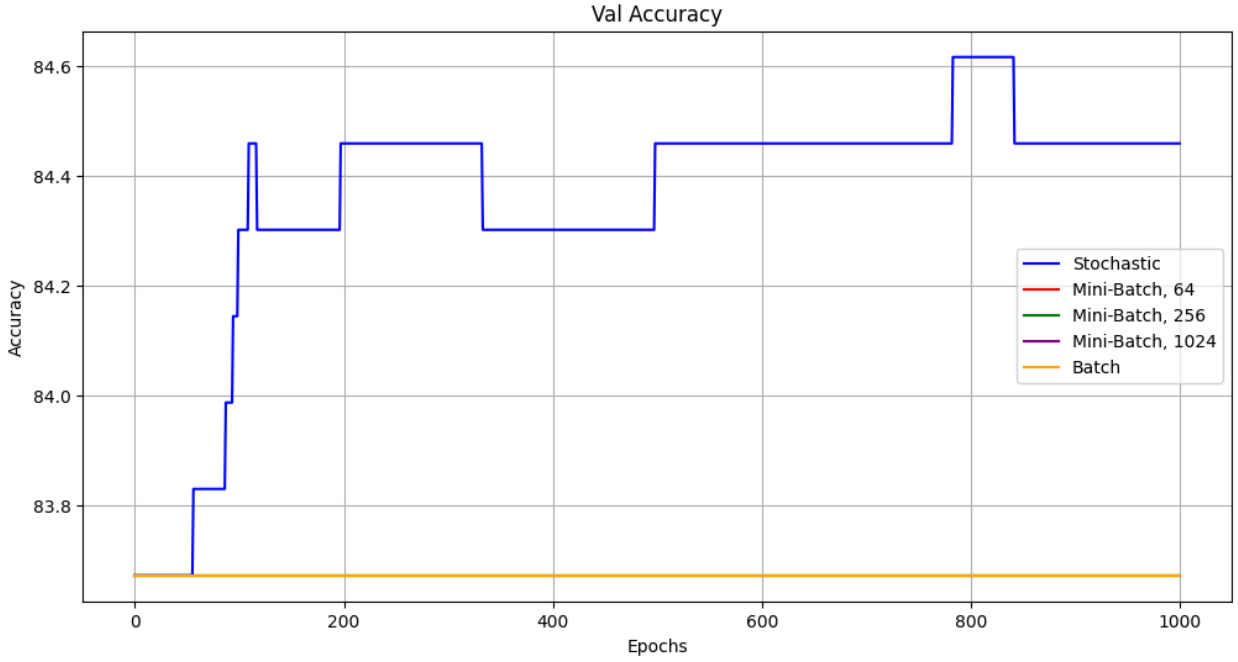
The regression model was run for various optimization algorithms. The loss and accuracy were observed for Stochastic Gradient Descent, Batch Gradient Descent, and Mini-Batch Gradient Descent with batch sizes of 64, 256, and 1028. The algorithms were run for 1000 epochs (due to runtime constraints of the SGD algorithm) with a learning rate of 1/epoch = 0.001. The results are provided below.

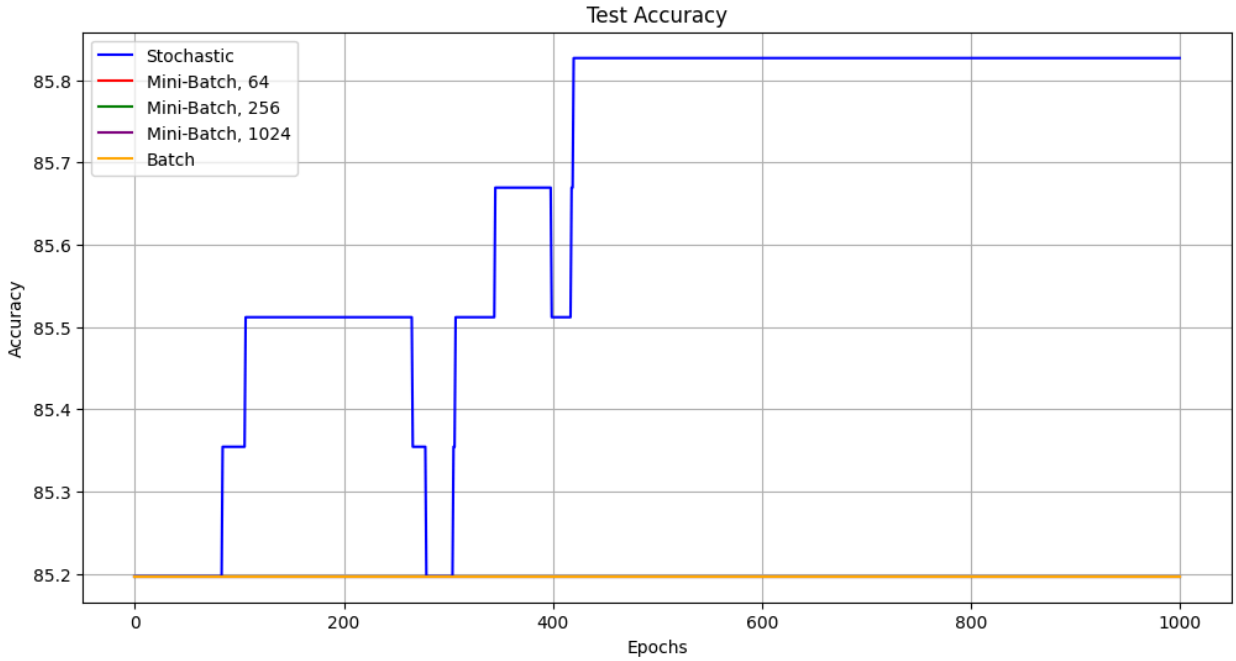



Val Loss
Stochastic
Mini-Batch, 64
Mini-Batch, 256
Mini-Batch, 1024
Batch
Loss
Epochs
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
200
400
600
800
1000


Test Loss
Stochastic
Mini-Batch, 64
Mini-Batch, 256
Mini-Batch, 1024
Batch
Loss
Epochs
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
200
400
600
800
1000


Train Accuracy
Stochastic
Mini-Batch, 64
Mini-Batch, 256
Mini-Batch, 1024
Batch
Accuracy
85.6
85.5
85.4
85.3
85.2
85.1
85.0
0
200
400
600
800
1000
Epochs


Val Accuracy
Stochastic
Mini-Batch, 64
Mini-Batch, 256
Mini-Batch, 1024
Batch
Accuracy
84.6
84.4
84.2
84.0
83.8
0
200
400
600
800
1000
Epochs



It can be observed that stochastic gradient descent provides the best results and the minimum loss on train, val, and test datasets. As batch size increases, the model's performance decreases, as indicated by its loss function. Stochastic gradient, having batch size = 1, has the best performance, while batch gradient, which has a batch size the same as the size of train data, gives the maximum loss. However, there is a tradeoff between the algorithm's running time and its performance. As the batch size decreases, the algorithms' performance increases, but it consumes more time to converge. Thus, the stochastic gradient runs the slowest, and the batch gradient runs the fastest. A fast runtime allows us to run more epochs, which might, in turn, improve the model performance, as seen in part A, where batch gradient allows us to run 100,000 epochs conveniently. Note that the accuracy plots might not give any insights into the performance of various algorithms since the accuracy is roughly the same for a small number of epochs.

Part E

Applying K-fold cross-validation on the train set, we ran the gradient descent with a learning rate of 0.01 for 10,000 epochs and obtained the following results.

```
Final Results:
Average Accuracy: 84.9916%, Standard Deviation: 0.5115
Average Precision: 0.2000, Standard Deviation: 0.4000
Average Recall: 0.0023, Standard Deviation: 0.0047
Average F1 Score: 0.0046, Standard Deviation: 0.0092
```

The average accuracy is quite high, indicating that, on average, the model performs well on the folds. However, its deviation is also pretty high, which may point to potential issues in the dataset.

The precision is poor. The main reason is that we mitigated zero-division error in the model by setting it to zero. This error was caused by the huge imbalance in the dataset, where certain folds did not contain any instances of actual positive cases. For the same reason, the variance is pretty high. While certain folds containing positive labels have excellent precision, others have precision set to zero due to null instances of positive labels.

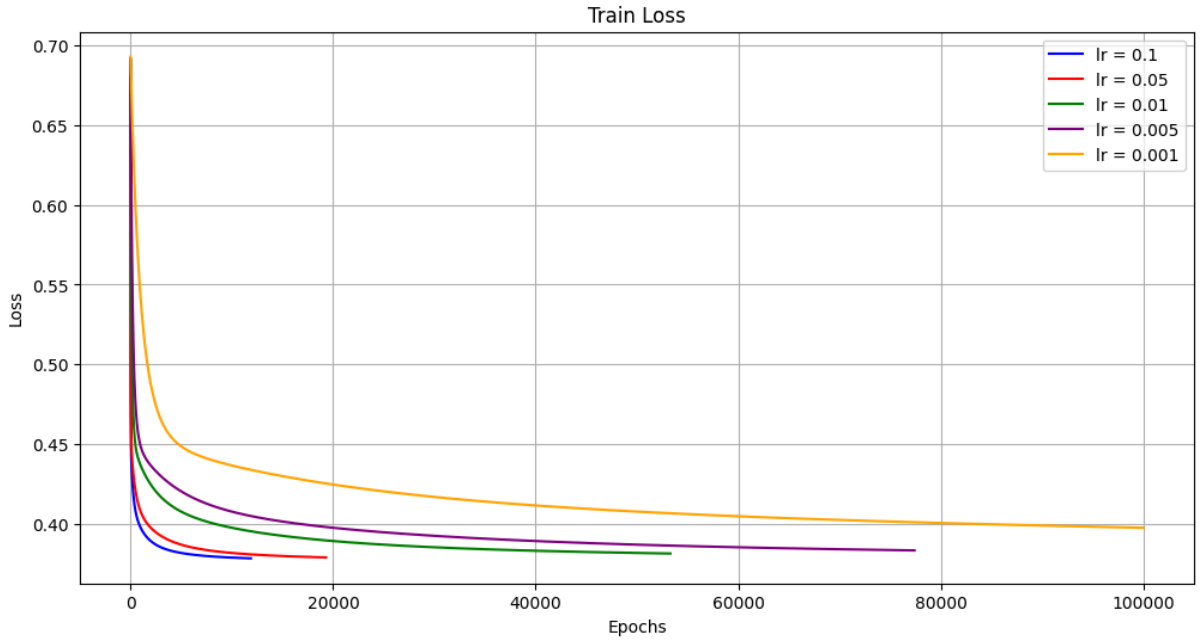
The average recall is low due to the model's inability to classify true positives. This increases the number of false negatives and decreases the model's sensitivity. The variance, however, is very low due to the problem of low recall being consistent across the folds.

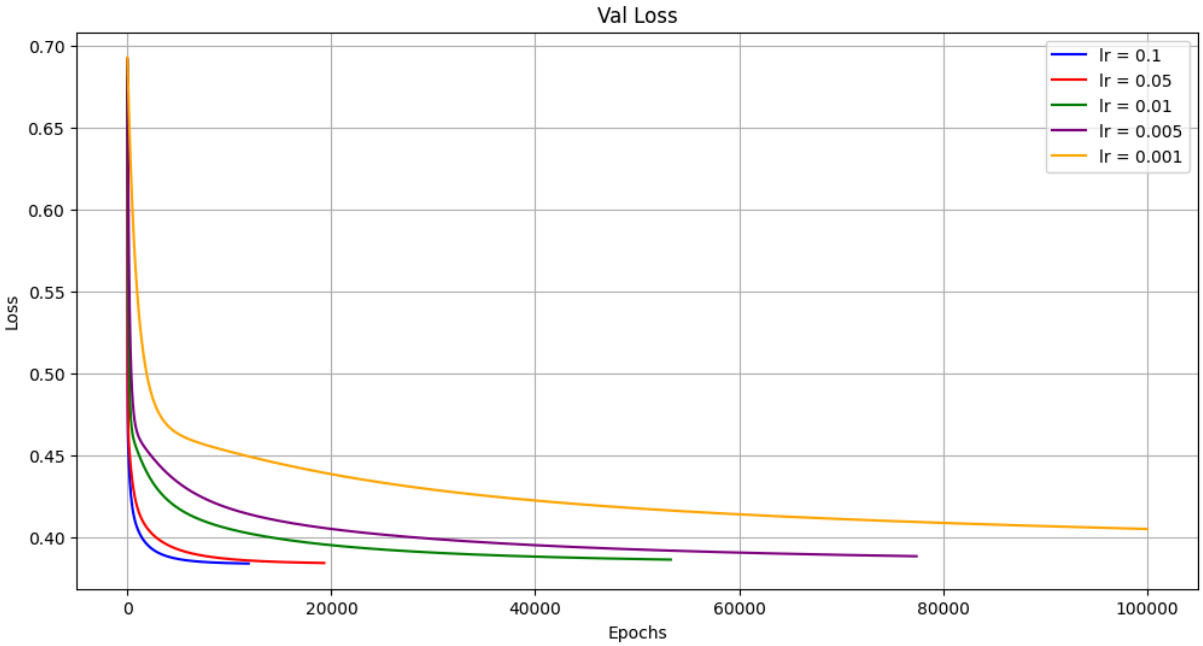
The average F1 score is very low, aligning with the low precision and recall. The low standard deviation confirms that the F1 score's poor performance is consistent across different folds, indicating an overall deficiency in the model's ability to balance precision and recall for the positive class.

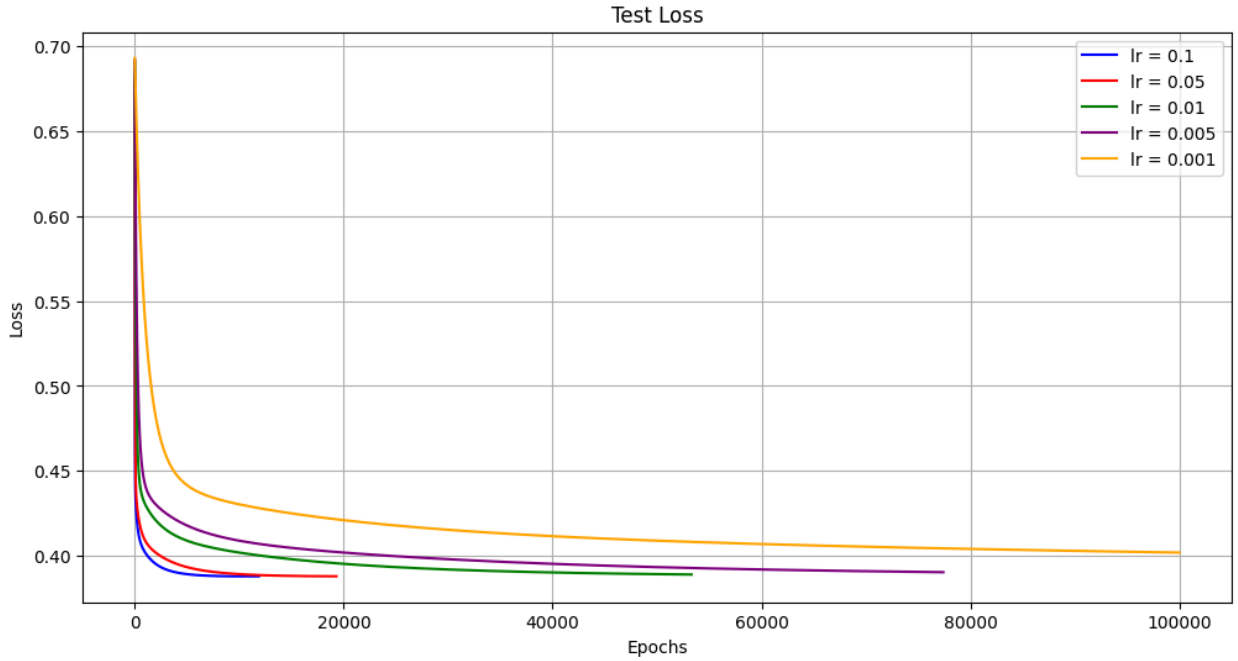
Part F

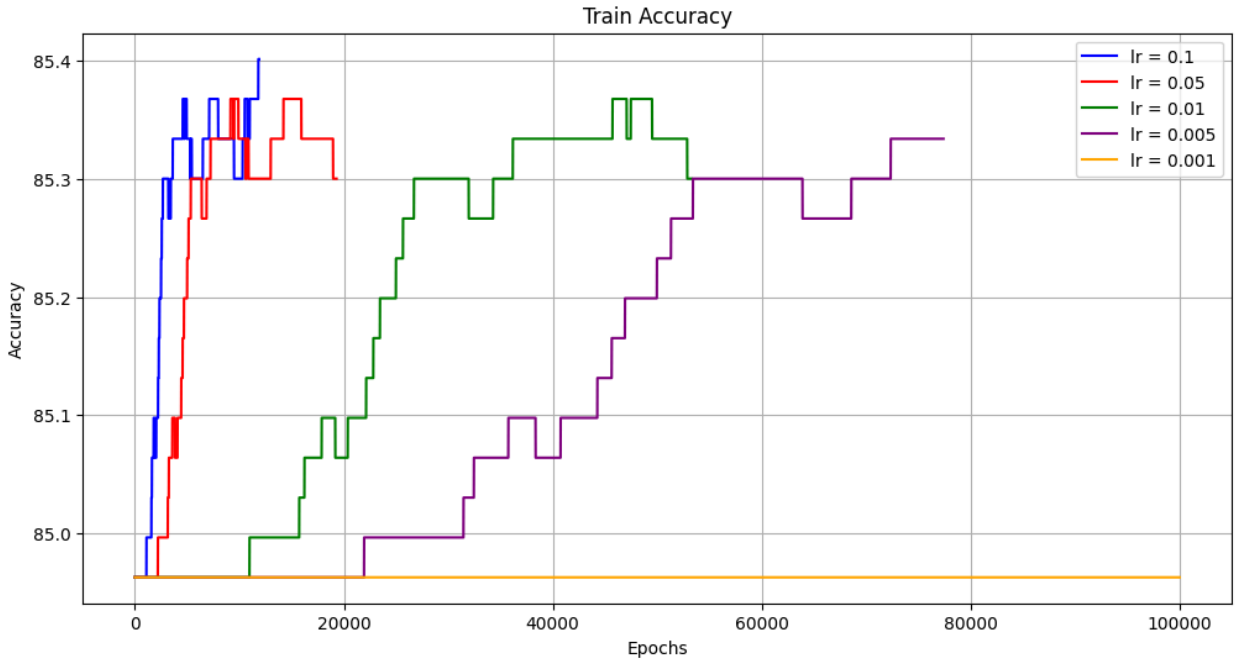
In this part, we have experimented with different regularization techniques and learning rates, applied early stopping on them, and provided an analysis of the produced plots.

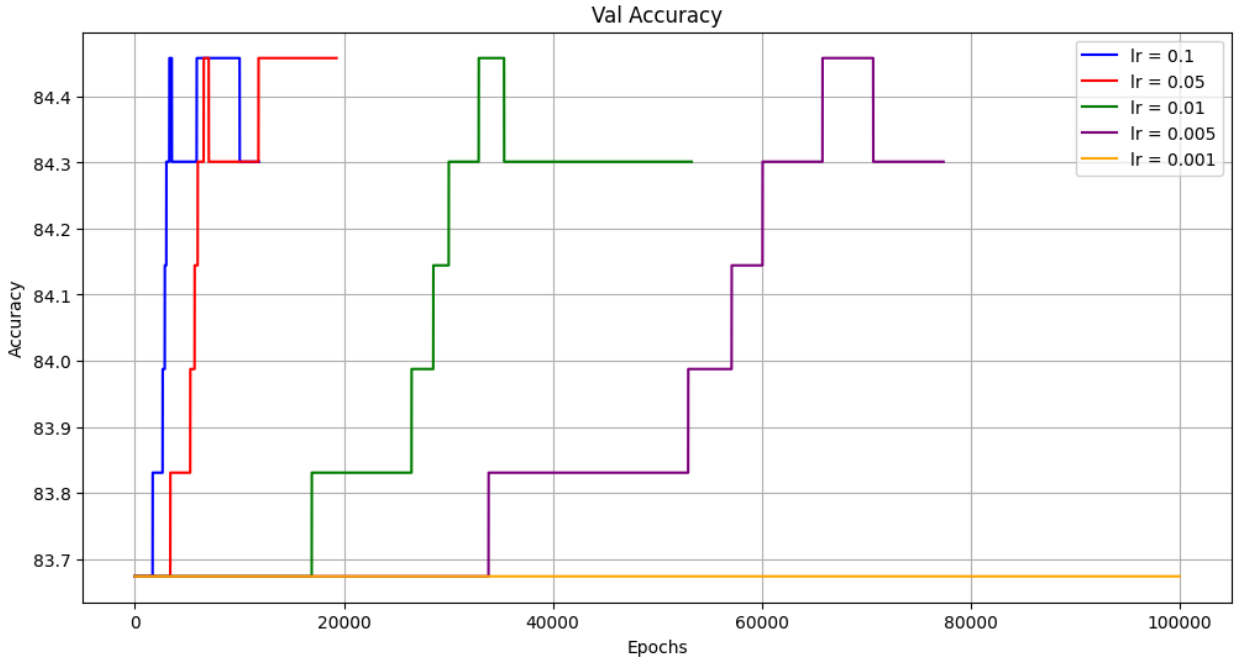
A. No regularization, 100,000 epochs, patience (early stopping) = 1000

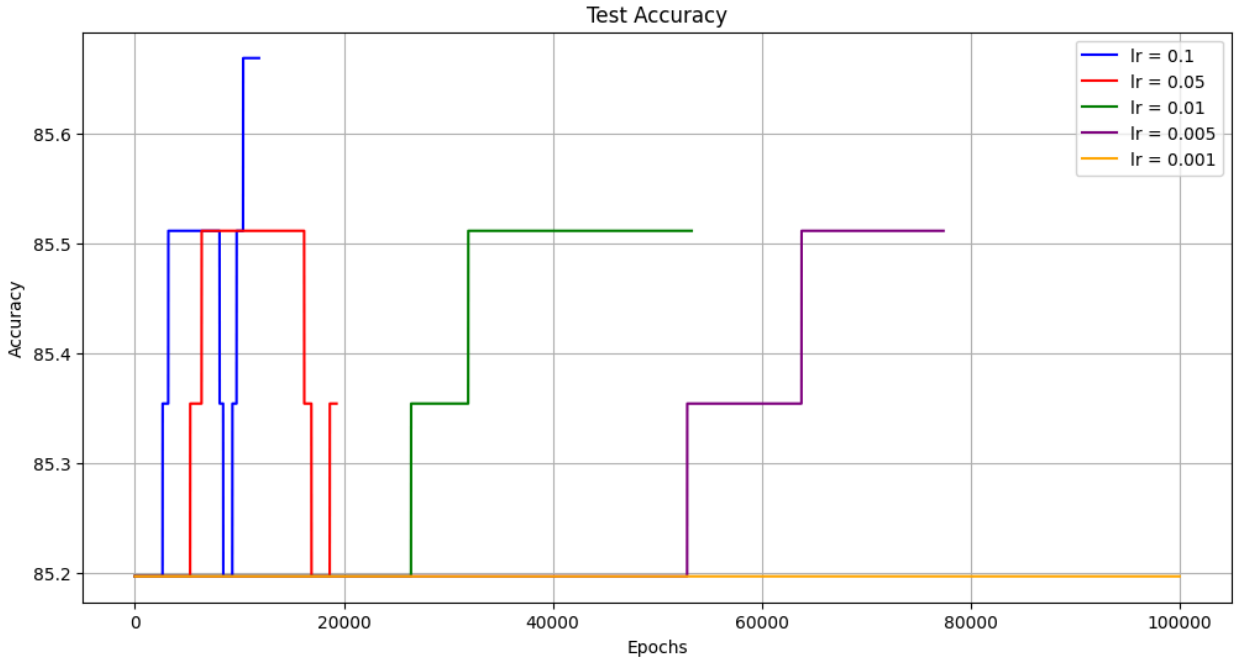





Test Loss
lr = 0.1
lr = 0.05
lr = 0.01
lr = 0.005
lr = 0.001
Loss
Epochs
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
20000
40000
60000
80000
100000


Train Accuracy
lr = 0.1
lr = 0.05
lr = 0.01
lr = 0.005
lr = 0.001
Accuracy
Epochs
85.4
85.3
85.2
85.1
85.0
0
20000
40000
60000
80000
100000

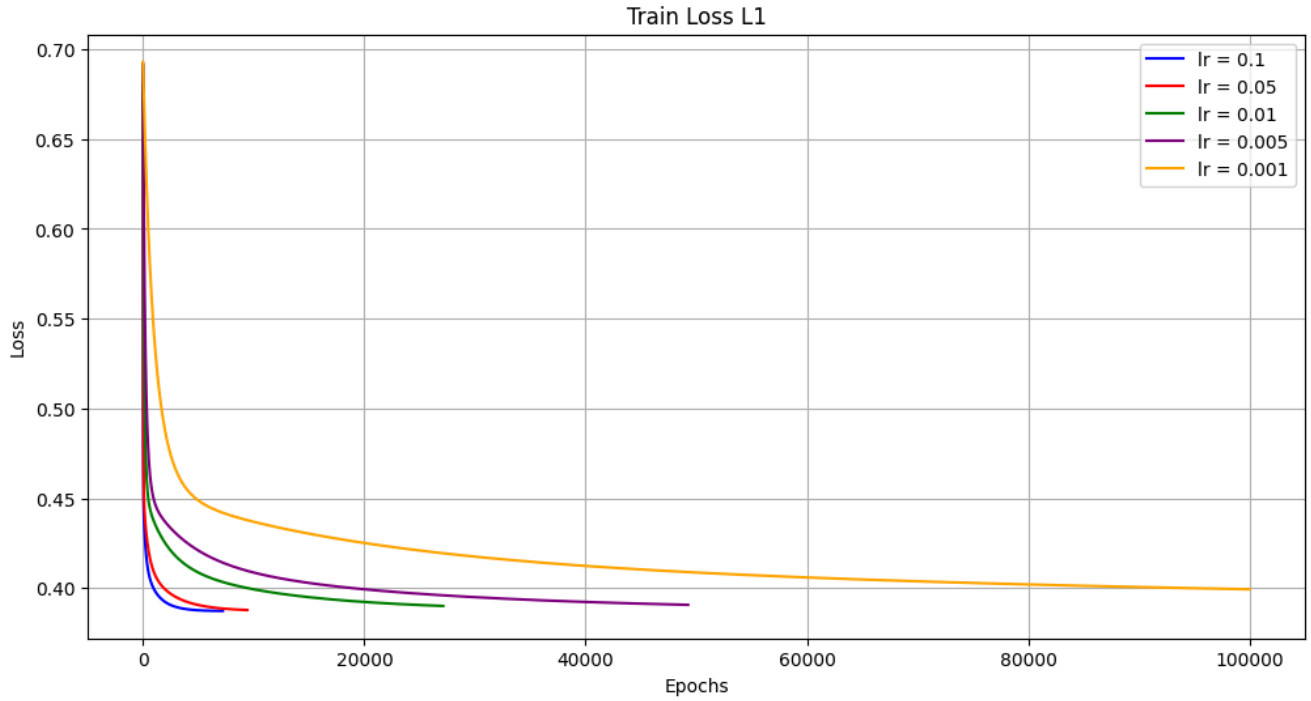


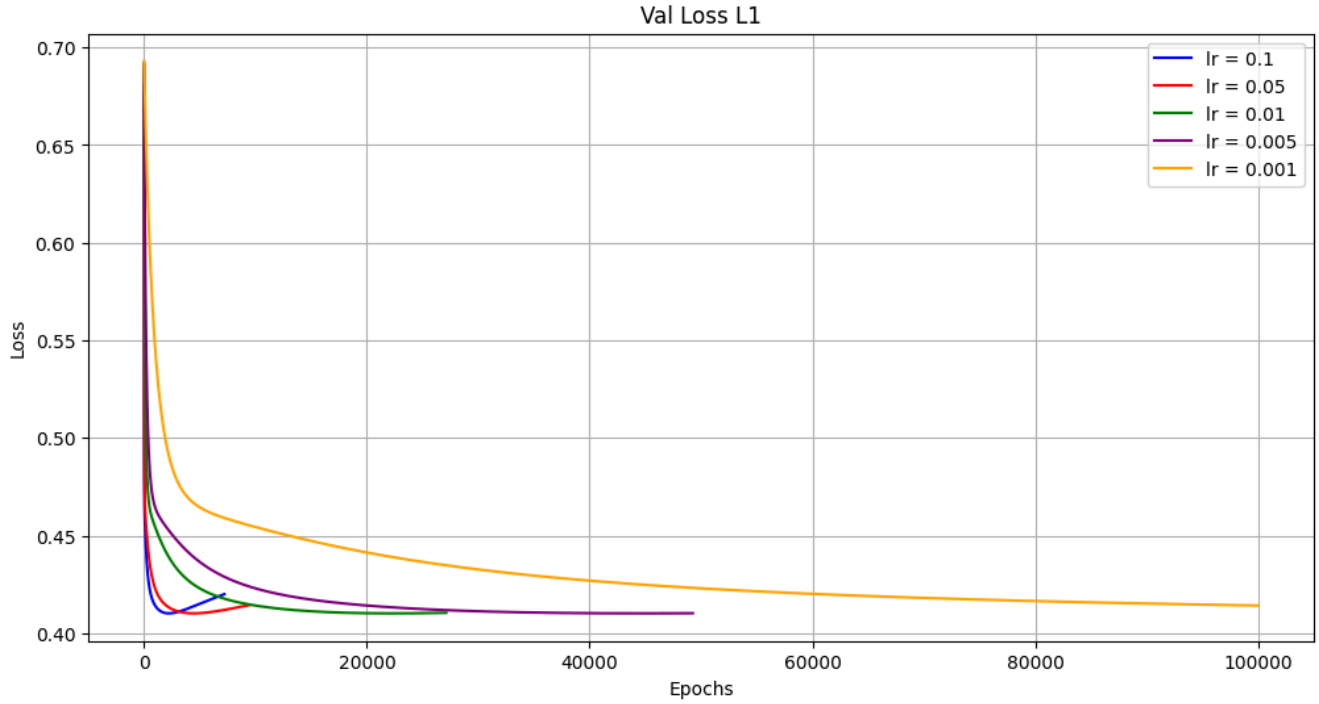


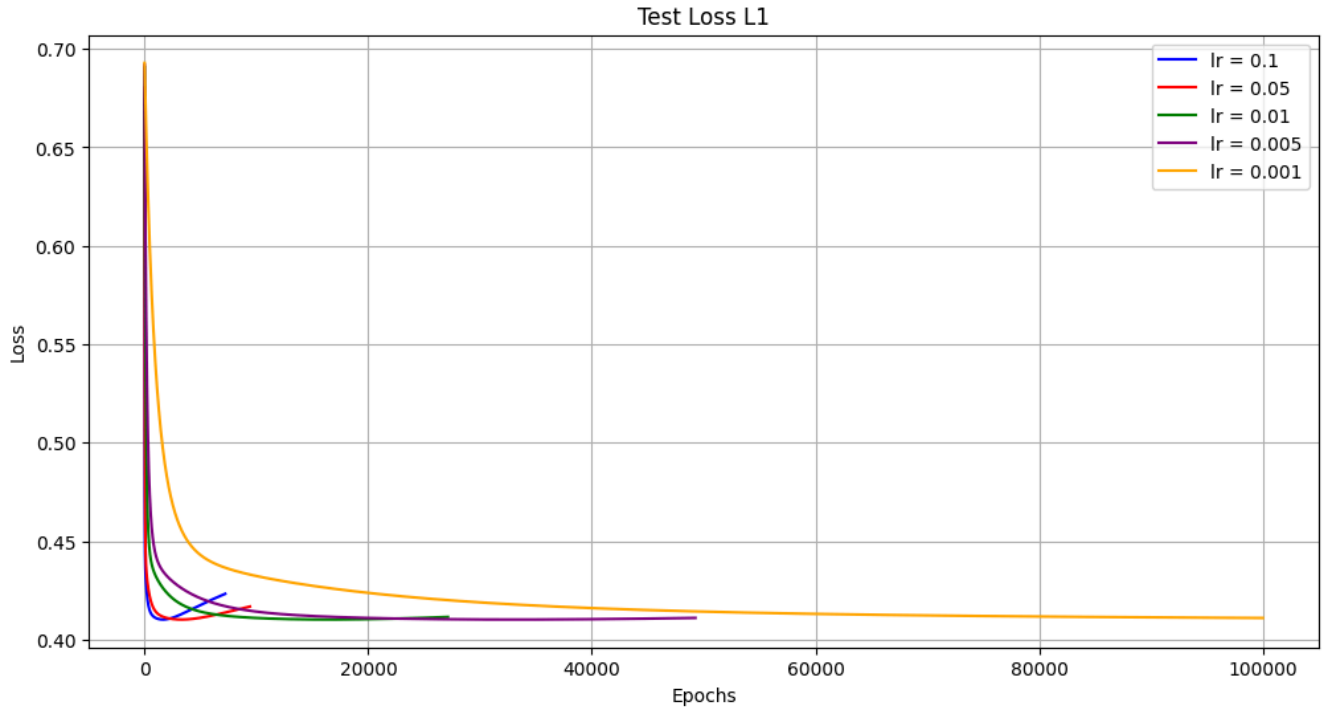
With no regularization, we have no clear indication of where to initiate early stopping. Hence, early stopping is implemented when it is observed that the val loss is not decreasing even after the patience is exhausted. This helps save computational resources, and we hope it might prevent early stopping.

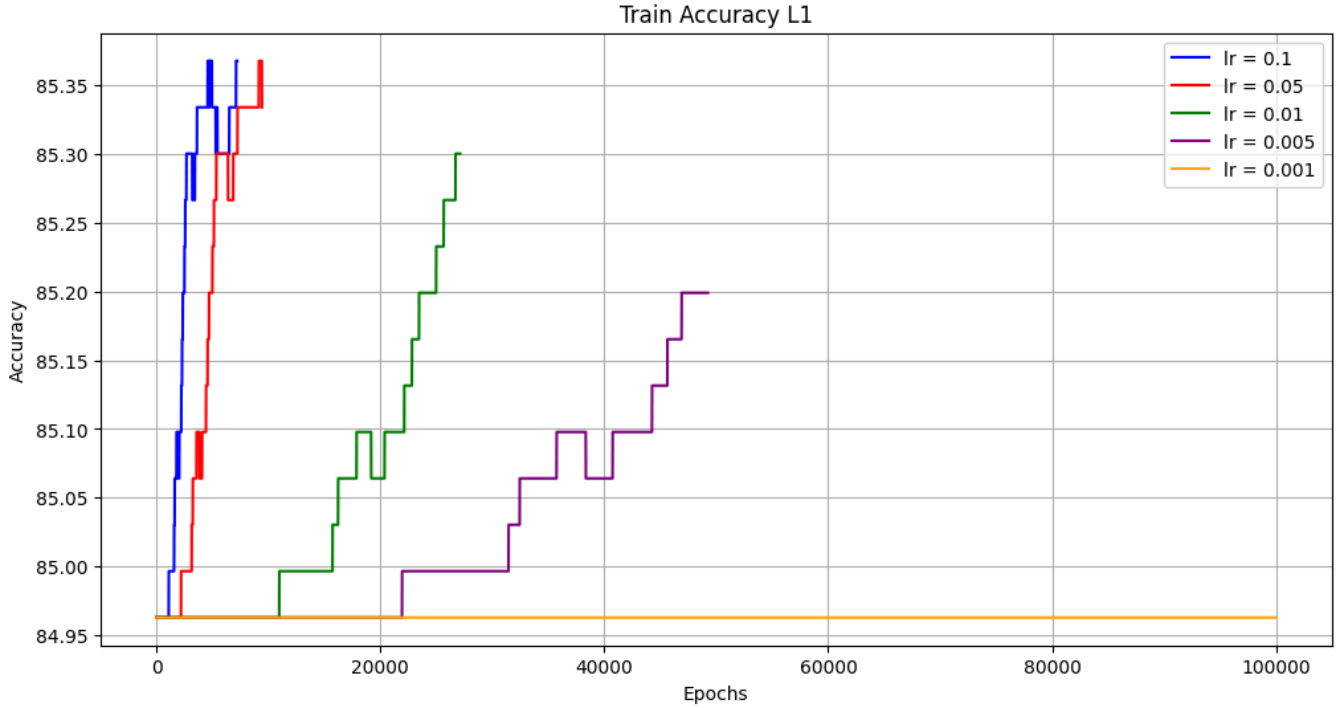
When regularization terms are added, a more concrete indication of where to implement early stopping is observed.

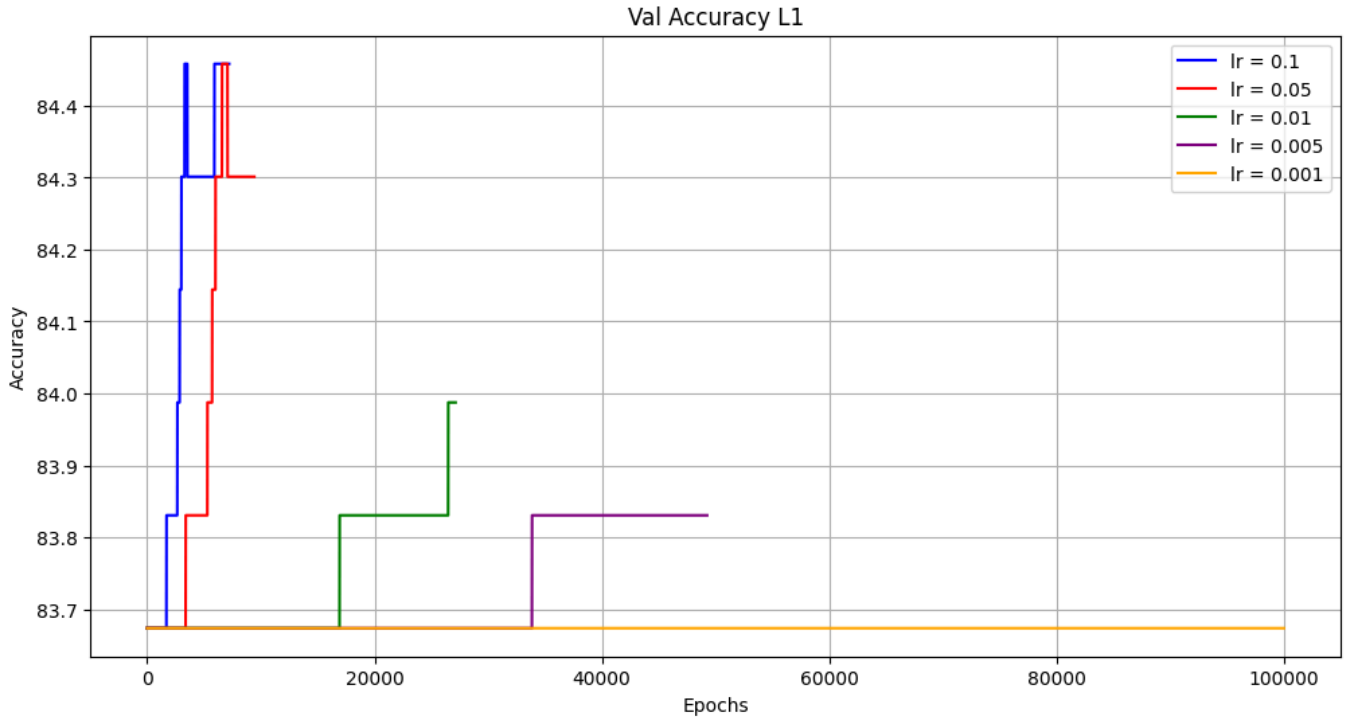
B. L1 regularization, λ = 2, 100,000 epochs, patience = 5000

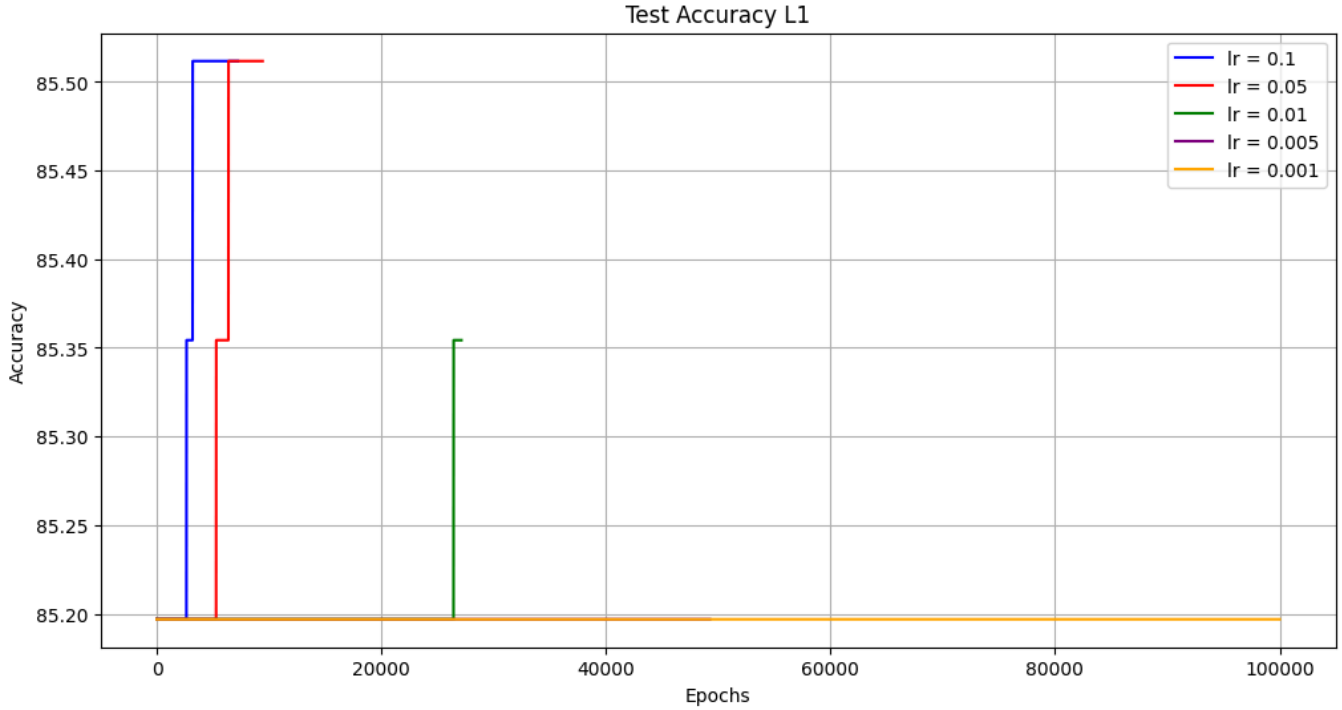




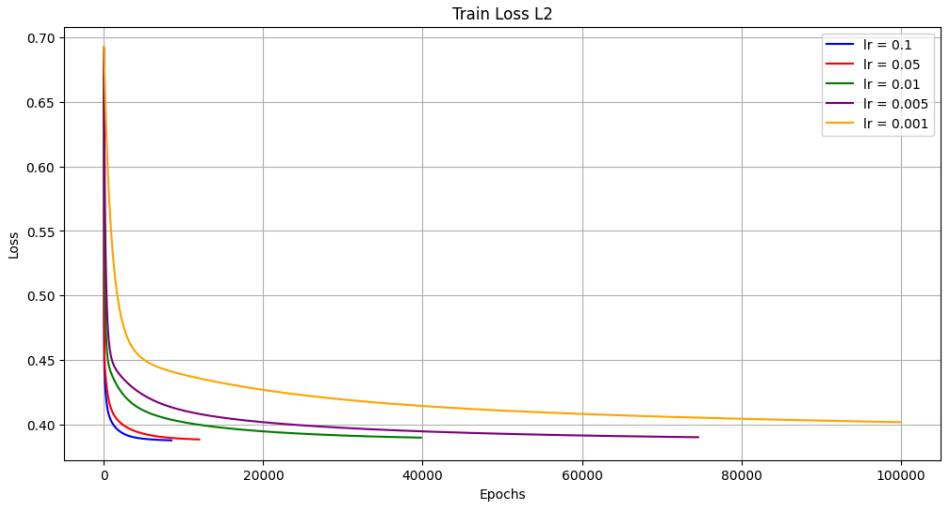

Test Loss L1
Loss
Epochs
lr = 0.1
lr = 0.05
lr = 0.01
lr = 0.005
lr = 0.001
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
20000
40000
60000
80000
100000

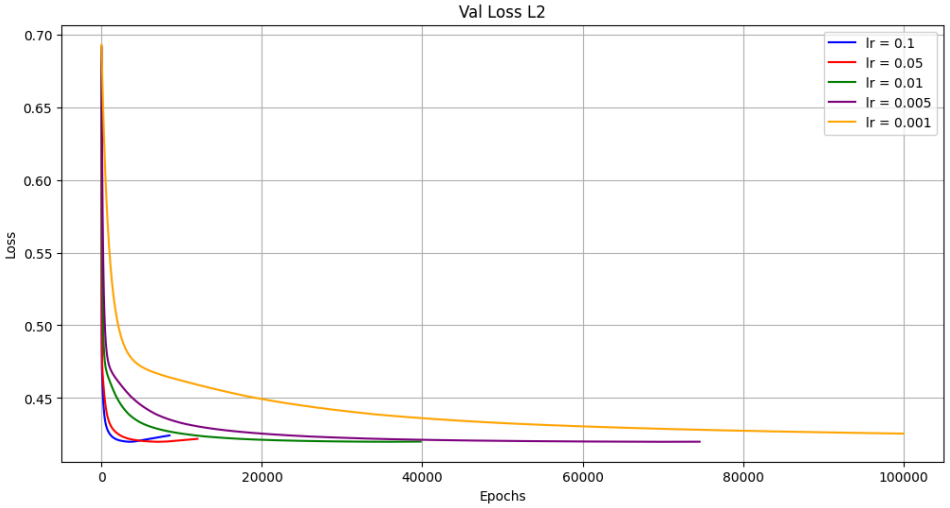

Train Accuracy L1
Accuracy
Epochs
lr = 0.1
lr = 0.05
lr = 0.01
lr = 0.005
lr = 0.001
85.35
85.30
85.25
85.20
85.15
85.10
85.05
85.00
84.95
0
20000
40000
60000
80000
100000

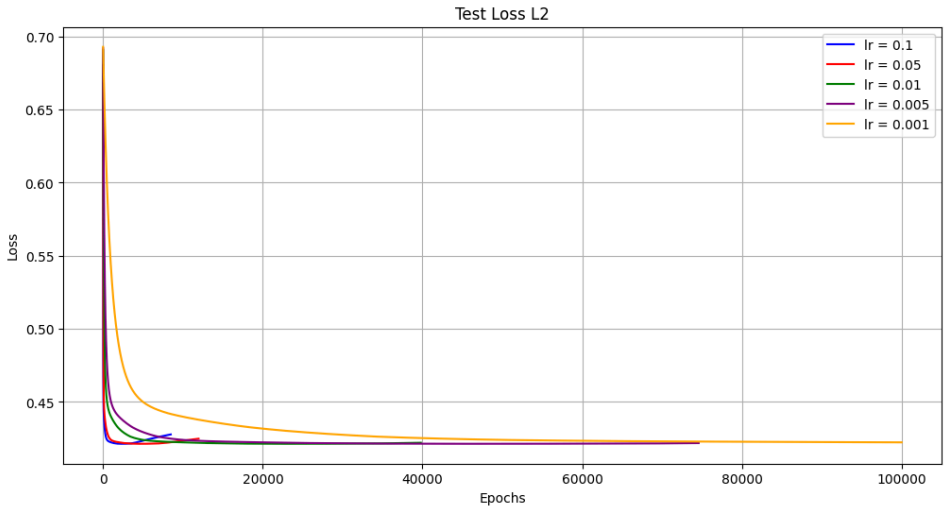

Val Accuracy L1
lr = 0.1
lr = 0.05
lr = 0.01
lr = 0.005
lr = 0.001
Accuracy
84.4
84.3
84.2
84.1
84.0
83.9
83.8
83.7
0
20000
40000
60000
80000
100000
Epochs

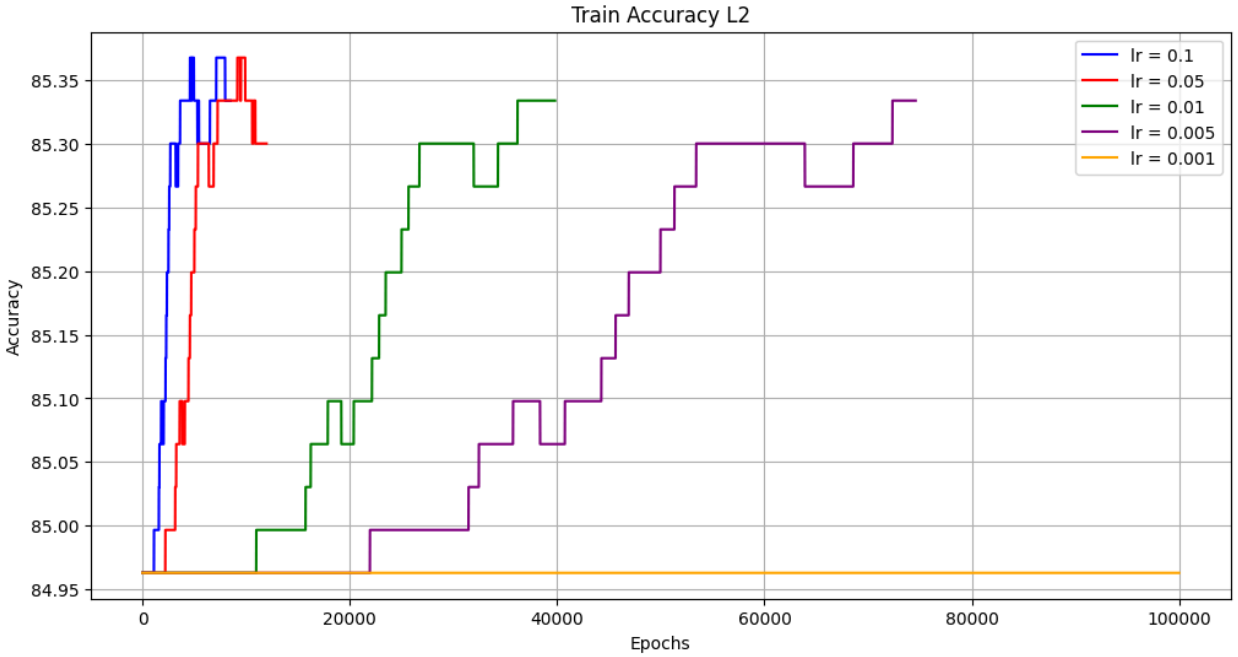

Test Accuracy L1
lr = 0.1
lr = 0.05
lr = 0.01
lr = 0.005
lr = 0.001
Accuracy
85.50
85.45
85.40
85.35
85.30
85.25
85.20
0
20000
40000
60000
80000
100000
Epochs

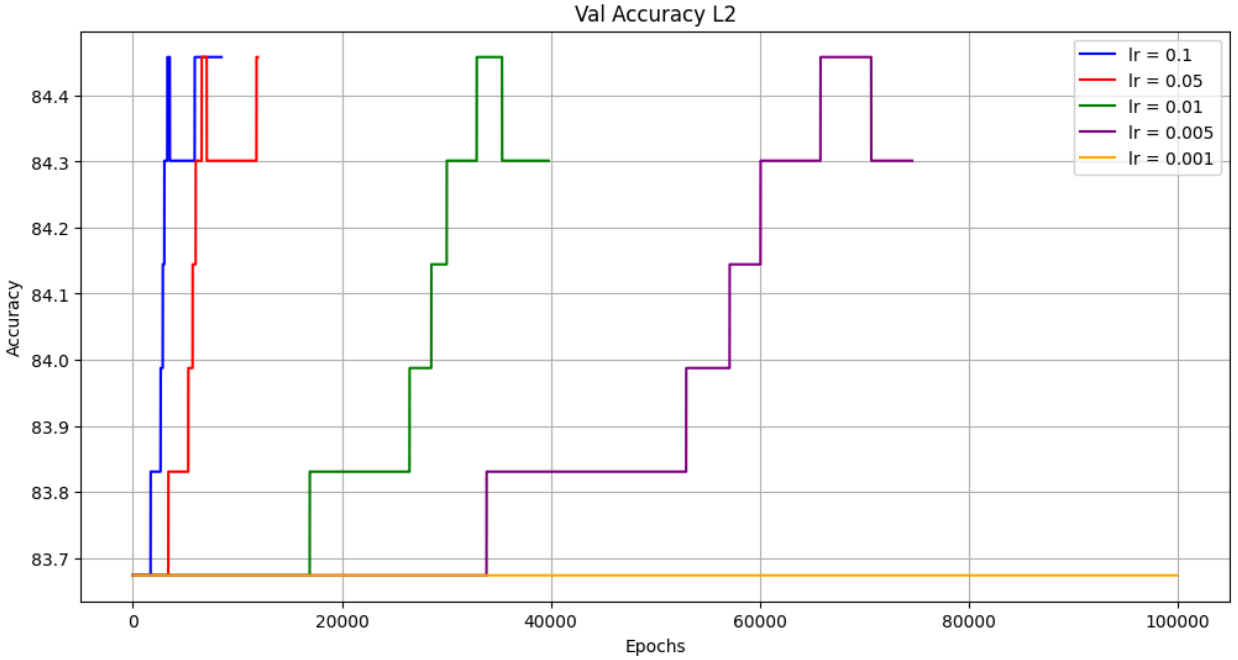
C. L2 regularization, λ = 2, 100,000 epochs, patience = 5000

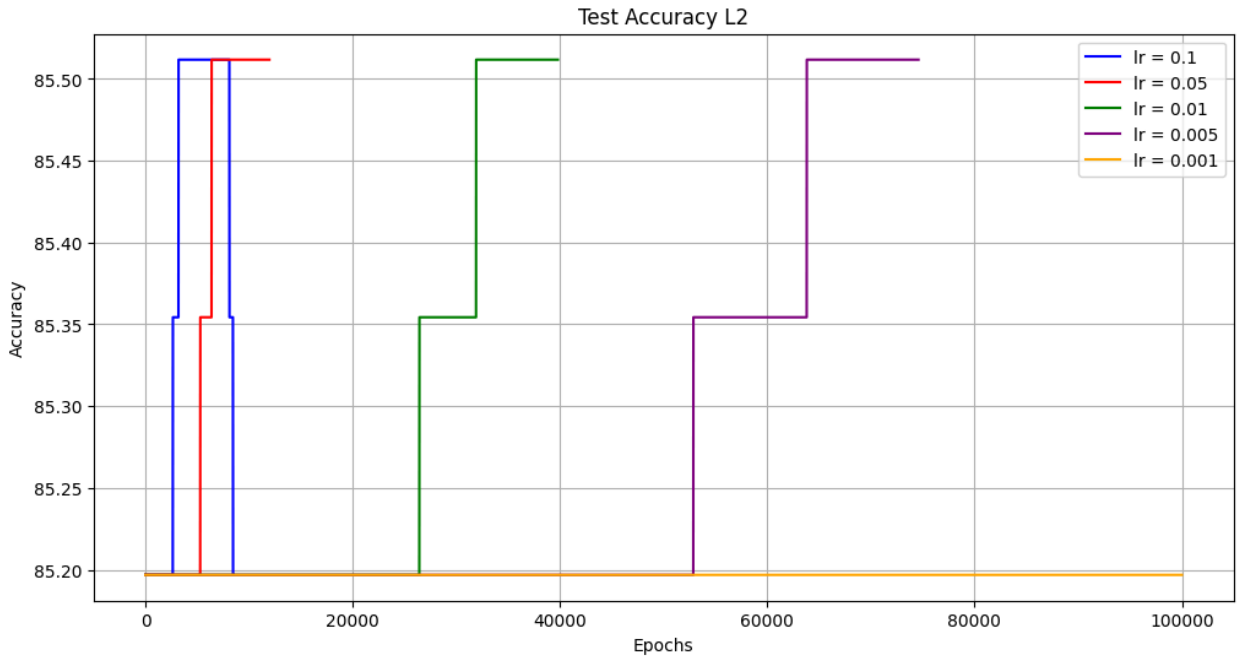







Train Accuracy L2
lr = 0.1
lr = 0.05
lr = 0.01
lr = 0.005
lr = 0.001
Accuracy
85.35
85.30
85.25
85.20
85.15
85.10
85.05
85.00
84.95
0
20000
40000
60000
80000
100000
Epochs


Val Accuracy L2
lr = 0.1
lr = 0.05
lr = 0.01
lr = 0.005
lr = 0.001
Accuracy
84.4
84.3
84.2
84.1
84.0
83.9
83.8
83.7
0
20000
40000
60000
80000
100000
Epochs



In both these regularizations, it is evident from the plots of loss against the number of epochs that the loss starts increasing again after a certain threshold. When this happens, it indicates that the model has started to overfit. The point where the slope of the loss function changes its direction is the perfect point for early stopping. Beyond that point, test error decreases. In the plots above, a large value of patience and regularizing constant was taken to showcase this pattern of rising error. It is also evident that the larger the learning rate, the faster the model learns the data; thus, the faster it crosses the optimal threshold. Note that the accuracy of the curve with lr = 0.001 is flat due to the slow learning of the model because of a low learning rate.