Jordan University of Science and Technology

Faculty of Computer and Information Technology

CS722 – Natural Language Processing

Assignment #1: Sentiment Analysis

Submitted to:

Dr. Rasha Obeidat

Submitted by:

Batool Ramadan Hamdallh – ID: 150869
Muna Ahmad Al-Nemrawi – ID: 176887

November 23, 2024

Link to the full code's repository: https://github.com/monanemrawi/SentimentAnalysis.git
Note: Run the Jupyter file for clear instructions and plots.

Question 1 - The behavior of Gradient descent can be strongly influenced by the learning rate (η). Experiment with different learning rates (10000, 1000, 100, 10, 1, .01, .001, .0001, .00001) report your observation on the convergence behavior of the gradient descent. Use a fixed number of iterations (1000) as your stopping condition. Please mention your choice clearly in your report. Note that if you observe an overflow, then the learning rate you are using is too big. Please mention the learning rates that cause an overflow. Report the training and dev accuracy for each learning rate value in a table, which one are you going to use for the final model? Notice that the smaller the learning rate you use, the more the number of iterations you need. Try to use 10000 iterations for the learning rate 0.0001, do you observe any Accuracy improvement?

Answer - The results show that the convergence behavior of Gradient Descent is heavily influenced by the learning rate. Very high rates such as (10,000 and 1,000) cause divergence or numerical overflow, while fairly high rates such as 100 fails to converge effectively. At a learning rate of 10, the model achieves moderate convergence but falls short of optimal performance. A learning rate of 1 achieves stable performance but suboptimal results. The best performance is observed at a learning rate of 0.01, yielding the highest validation accuracy (0.7625). Smaller learning rates (0.001, 0.0001, and 0.00001) show poor accuracy due to insufficient updates within the fixed iteration limit (1,000 epochs). Expanding iterations to 10,000 for a rate of 0.0001 does not result in significant improvements, and further iterations may be required. Overall, a learning rate of 0.01 is identified as optimal for the current setup, and even with the test data.

Learning Rate	Train Accuracy	Validation Accuracy
10000	Overflow	Overflow
1000	Overflow	Overflow
100	Overflow	Overflow
10	0.616114	0.6175
1	0.731212	0.71
0.01	**0.769804**	**0.7625**
0.001	0.739336	0.6
0.0001	0.584292	0.58

Question 2 - Plot the training accuracy, the validation accuracy of your model as a function of the number of gradient descent iterations for each learning rate of the three learning rates: 100, 0.1, 0.001 .00001. The y-axis is the accuracy, the x-axis is the number of iterations. What have you observed? Explain your observations.

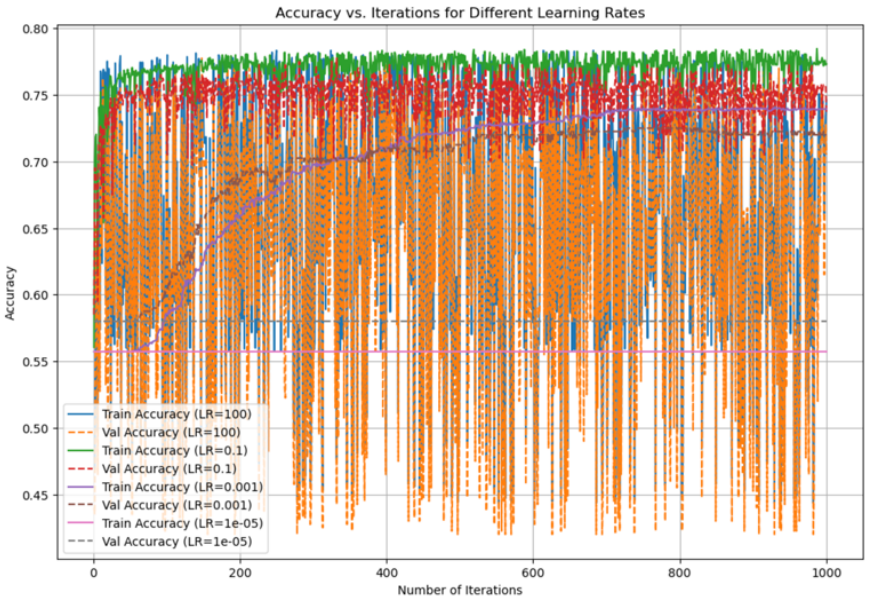


Figure 1- Results with Learning Rate 100

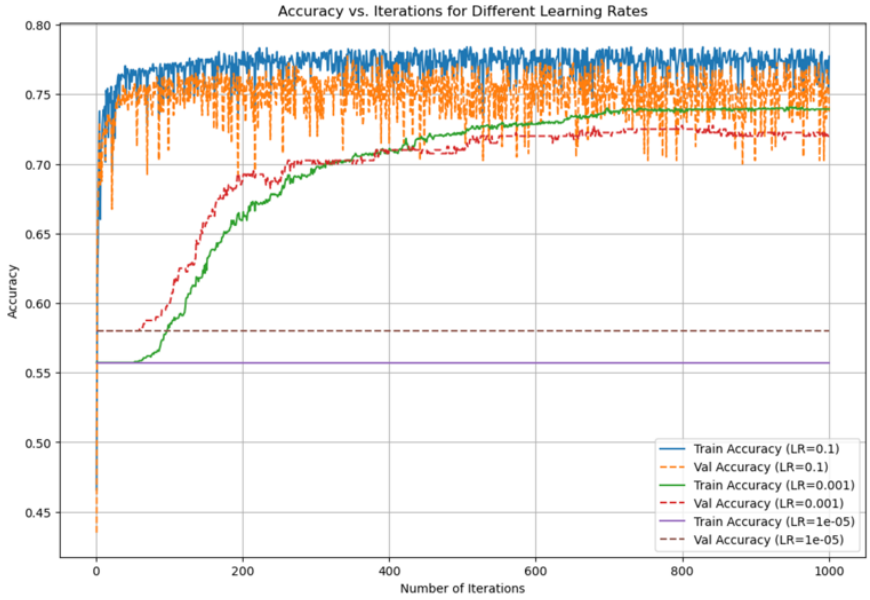


Figure 2 - Results without Learning Rate 100

Answer - The figures indicate that with a learning rate of 100, the gradient steps are too large, preventing proper convergence to the optimal solution. A learning rate of 0.1 strikes a better balance between training and validation accuracy, making it the most suitable choice among the tested rates. For smaller learning rates such as 0.001, 0.0001, and 0.00001, the accuracy decreases as the learning rate decreases, especially at 0.00001, indicating that the model does not converge efficiently within the specified number of iterations.

Question 3 - Repeat 2 by plotting the average training loss and the dev loss instead of the accuracy. The training and the dev losses are computed by dividing the loss sums by the number of samples.

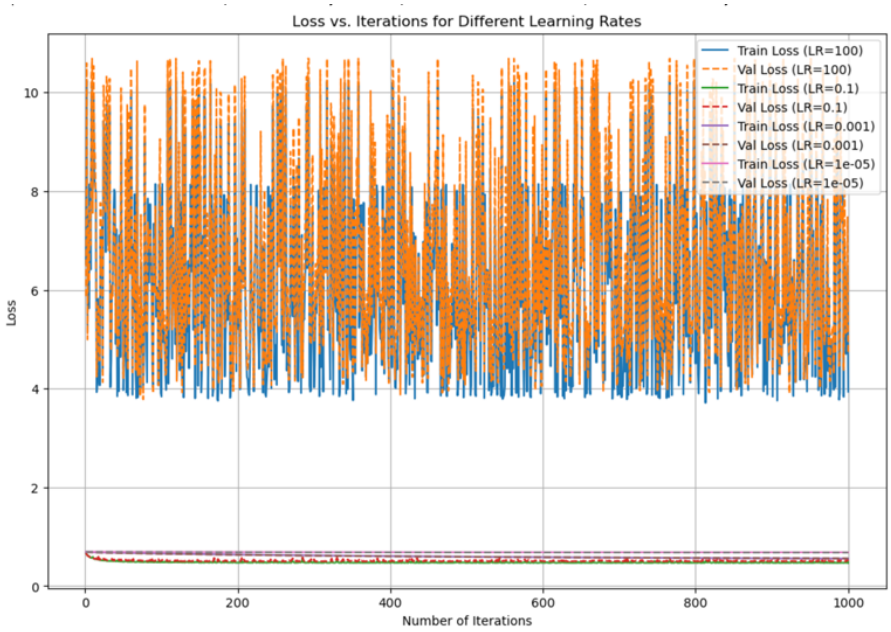


Figure 3- Results with Learning Rate 100

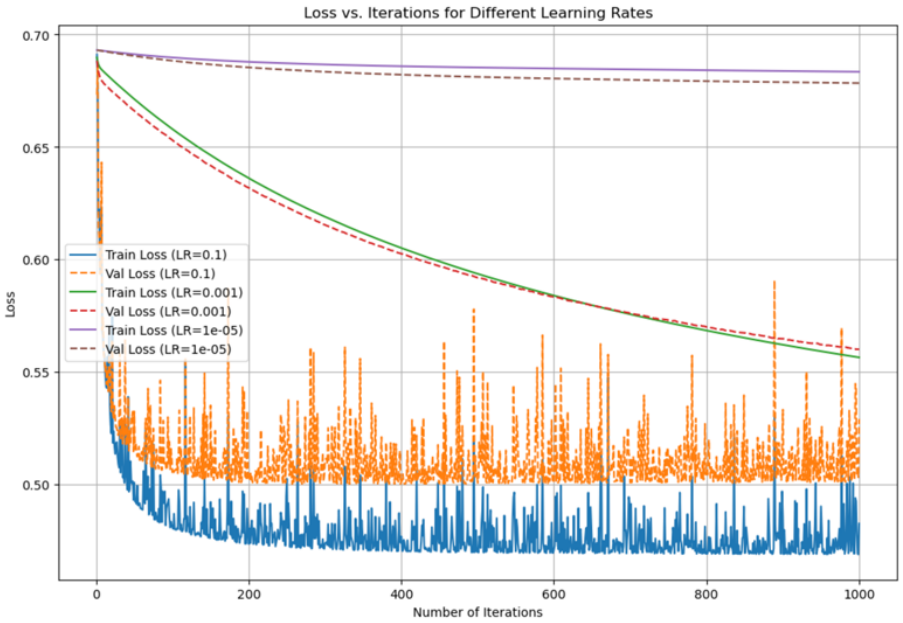
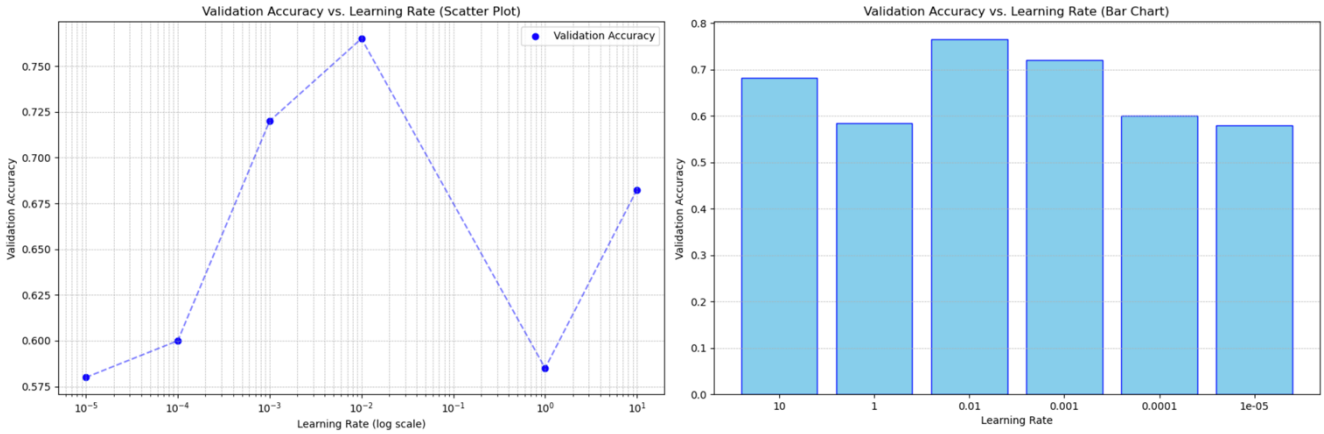


Figure 4 -Results without Learning Rate 100

Answer - When plotting the training and validation losses across iterations for different learning rates, we see distinct patterns related to the optimization process. Since accuracy and loss are inversely related (i.e., as accuracy increases, loss decreases and vice versa), the loss plots for training and validation will be symmetric to the accuracy plots, but in the opposite direction. With a learning rate of 100, the losses oscillate wildly, indicating divergence due to excessively large

gradient steps. At 0.1, the losses decrease steadily and stabilize, showing good convergence and proper optimization. For smaller learning rates like 0.001 and 0.00001, the losses decrease smoothly but much more slowly, with 0.00001 exhibiting minimal improvement over 1000 iterations, reflecting the inefficiency of excessively small steps. Overall, a learning rate of 0.1 achieves the best balance between convergence speed and stability, whereas too large or too small learning rates hinder effective optimization.

Question 4 - Plot the validation accuracy as a function of learning rate (the ones that did not lead to overflow)? What do you observe? Which learning rate gave you the best results. The y-axis is the accuracy, the x-axis is the learning rate (bar charts or dot charts are good choices for this question).

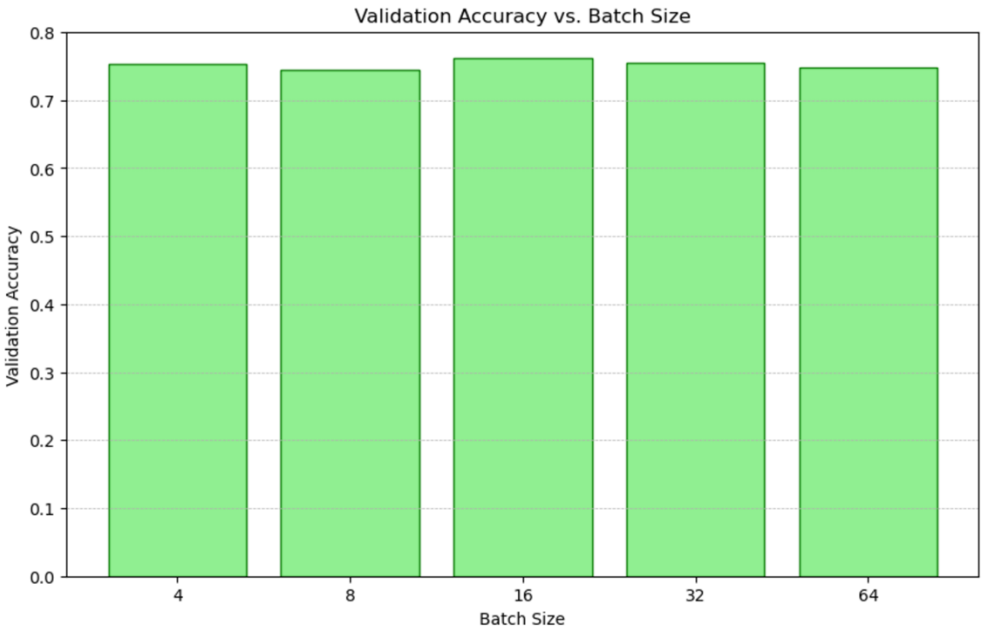


Answer - The validation accuracy was plotted against different learning rates, excluding those that caused overflow (10000, 1000, and 100). Both the scatter plot and bar chart illustrate that as the learning rate decreases from 10 to 0.01, the validation accuracy steadily improves. However, for smaller learning rates, such as 0.001 and 0.0001, the validation accuracy drops, likely due to slower convergence and insufficient progress during training. The learning rate of 0.01 achieves the best validation accuracy (0.7625), proving to be the most effective in balancing convergence speed and stability. This conclusion aligns with the results of the first experiment, further validating the choice of 0.01 as the optimal learning rate.

Question 5 - Report the testing accuracy that corresponds to the best learning rate (you decided from the previous step).
Note: Use the best learning rate you got from the previous step to answer the following questions (I mean don't re-tune it in order to limit the number of experiments you need to perform, not because this is the best practice)

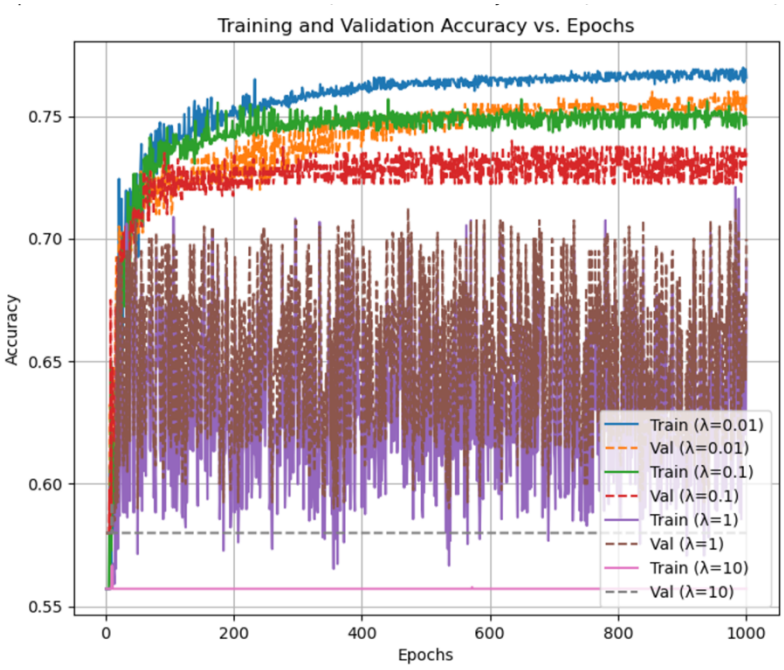
Answer - Testing Accuracy with best learning rate (0.01): 0.7825

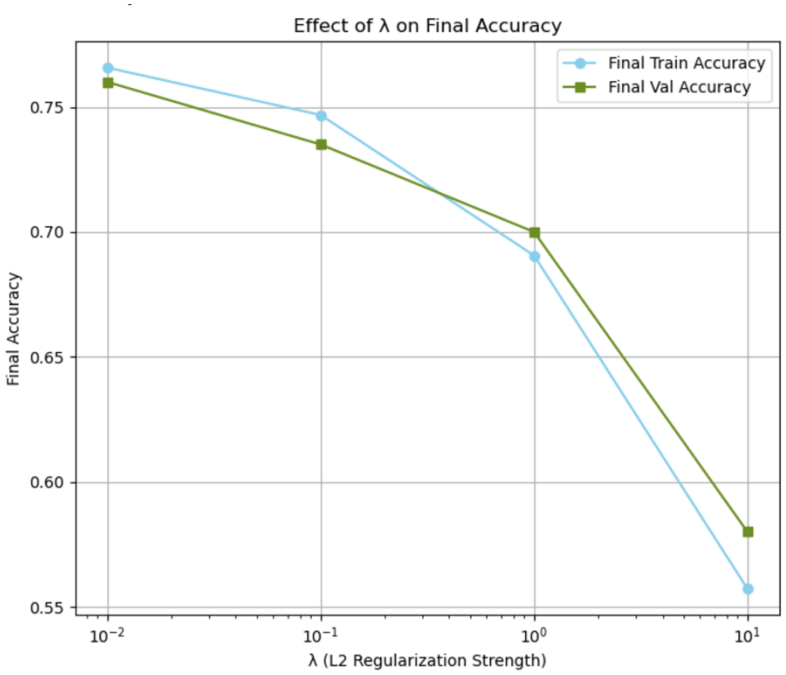
Question 6 - Try different batch sizes (4, 8, 16, 32, 64) and plot the validation Accuracy for these values in one figure, which batch size gave the best performance ()?



Answer - The validation accuracy for each batch size was tracked and plotted to compare the results. Among the different batch sizes tested, a batch size of 16 achieved the highest validation accuracy of (0.7625), as proved in Question 1. This suggests that a batch size of 16 strikes the best balance between the stability of gradient updates (which is better with larger batches) and the diversity of mini-batch samples (which is higher with smaller batches). Therefore, a batch size of 16 was chosen as the optimal setting for training the model.

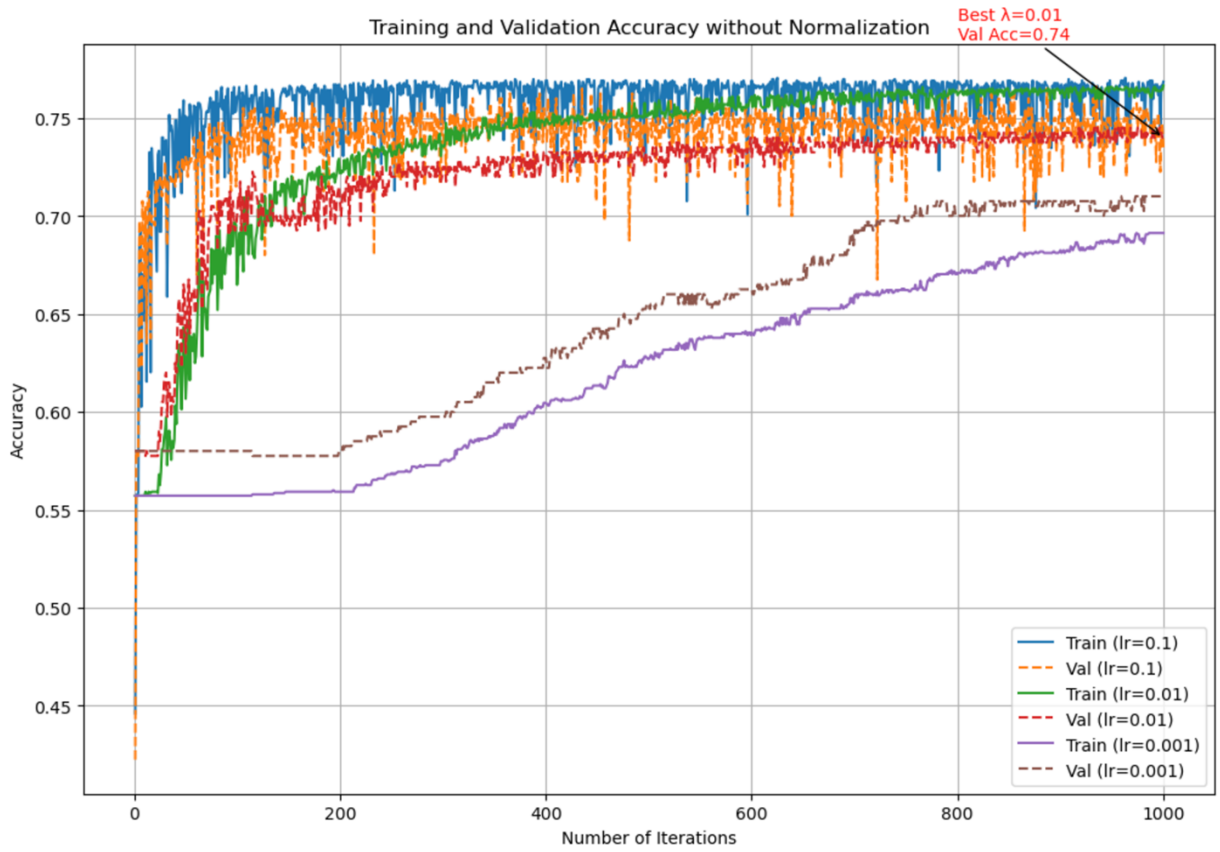
Question 7 - Add L2 regularization to the implementation of gradient decent, what did you add to your implementation? Did adding L2 regularization change to the best validation and testing accuracy you have got? What is the value of λ you have chosen and why? Do you recommend using regularization? Plot the training and the validation accuracy for different values of λ, you the best learning rate value you have obtained to do this part.





Answer - To incorporate L2 regularization into gradient descent, we modified the loss function to include an L2 penalty term proportional to λ and updated the gradients by adding $\frac{\lambda}{m}$ x weights for the weight updates. After adding L2 regularization, we observed an improvement in the model's generalization, as the validation accuracy stabilized and slightly improved compared to the results without regularization. The best validation accuracy was achieved with λ=0.1, which balances preventing overfitting while retaining the model's ability to learn effectively. We used the optimal learning rate (0.01) obtained earlier to isolate the effects of regularization. For larger values of λ (e.g., 1 or 10), we noticed a significant drop in both training and validation accuracy, as seen in the plots, indicating that excessive regularization can overly constrain the model, hindering learning. Regularization is recommended, especially when the model shows signs of overfitting, as it helps to constrain the model's complexity and improves performance on unseen data. The training and validation accuracies for different λ values, plotted using the optimal learning rate (0.01), further demonstrate the impact of regularization on the model's performance.

Question 8 - train a model without data normalization, does it work? Is it easy to train? Did you need more iterations? Explain your observation? Which rate of learning works the best for unnormalized data? (Do similar plot for 3 learning rate values of your choice like the plots in part Q1.)



Answer - Training a model without data normalization is more challenging because unnormalized features can cause the gradient descent process to become unstable. When features have different scales, gradient updates may be extremely large for certain features and too small for others, leading to inefficient convergence. In comparison to models trained with normalization, where the features are on a similar scale and gradients are more balanced, the unnormalized model shows greater difficulty in finding the optimal weights efficiently.

Despite these challenges, the model without normalization did achieve reasonable performance. A learning rate of 0.01 provided the best results, with a training accuracy of 0.766 and a validation accuracy of 0.74. Larger learning rates, such as 0.1, caused instability and led to oscillations in the training process, while smaller rates, such as 0.001, resulted in slower convergence and lower accuracy. This behavior reflects the importance of tuning the learning rate carefully, especially when data normalization is not used.

For unnormalized data, the **best learning rate** was found to be **0.01**, which strikes a balance between stability and convergence speed. As seen in the plots comparing different learning rates, the learning rate of 0.01 performed better than both 0.1 (which exhibited instability) and 0.001

(which converged too slowly). However, in models with normalized data, learning rates as high as 0.1 could still work effectively, highlighting how normalization helps with faster convergence.
In summary, while unnormalized data can still produce reasonable results, it requires more careful tuning of hyperparameters, particularly the learning rate. The comparison with normalized data shows that normalization significantly enhances convergence speed and stability.

Learning Rate	Train Accuracy	Validation Accuracy
0.1	0.750169	0.7375
0.01	0.766418	0.7400
0.001	0.689912	0.7125