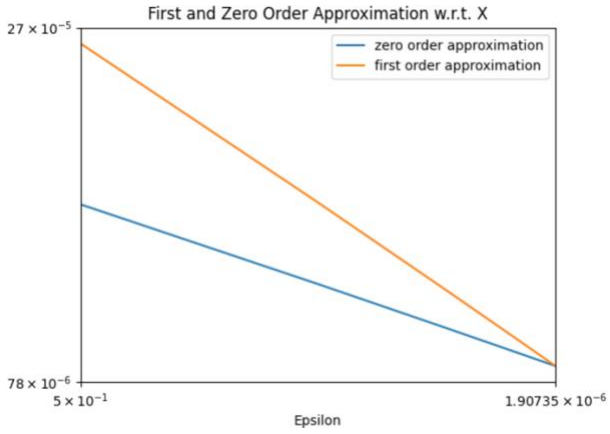
Deep Learning Assignment 1

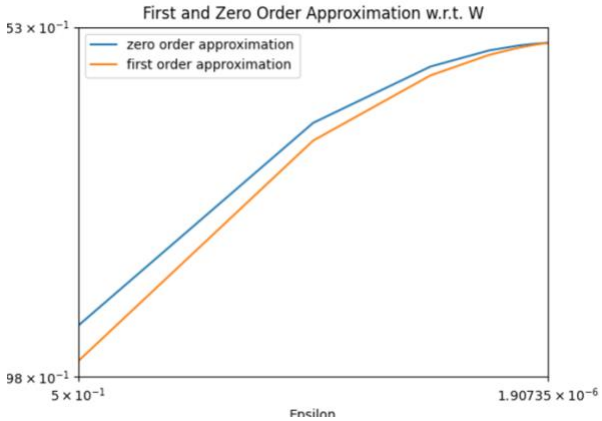
- Omer Asher 206559411
- Ohad Amzaleg 313313892

***** The late submission was approved with Omri

Part 1:

1.



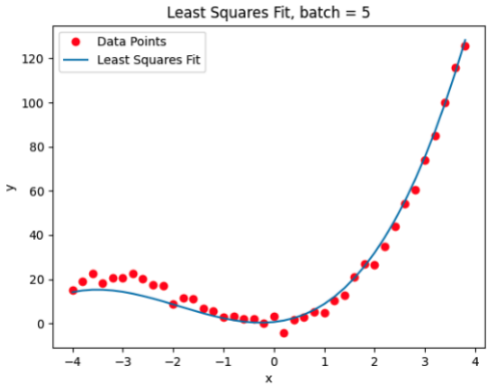


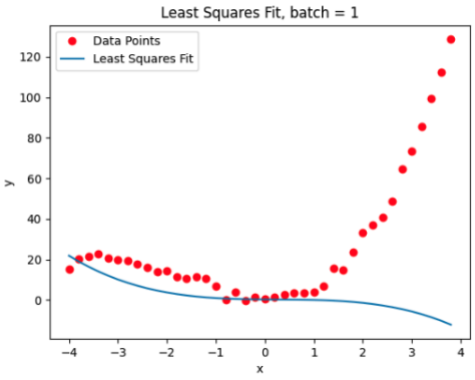
2. In the context of this assignment, the method createLeastSquaresData has been employed to generate synthetic data that simulates a noisy version of the function $x^3 + 5x^2 + 1$.

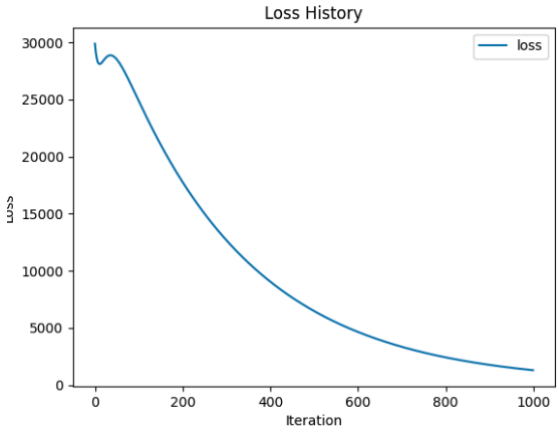
in the process of implementing the SGD optimizer, the choice of batch size plays a crucial role in determining the quality of the fit.

In our experiments, when the batch size was set to 1, the optimization process was more sensitive to individual data points. This resulted with unstable and inaccurate fit.

With a batch size of 5, the optimizer benefited from considering a small but reasonable number of data points simultaneously. This allowed for a more stable and reliable optimization process, and we obtained a good fit.





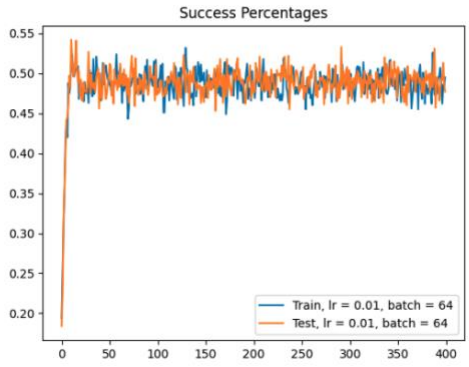


We observed that, when the batch size was 5, the loss decreased most of the time. This trend suggests that using a larger batch size could potentially further improve the optimization process and yield graphs that are more consistent.

3. In our experimentation, we explored different combinations of learning rates and batch sizes to understand their influence on the classification performance. We ran SGD for a sufficient number of iterations.

After trying various combinations, we found that a learning rate of 0.01 and a batch size of 64 yielded the best results. These hyperparameters struck a balance between convergence speed and stability, providing optimal

performance for the softmax function minimization task.

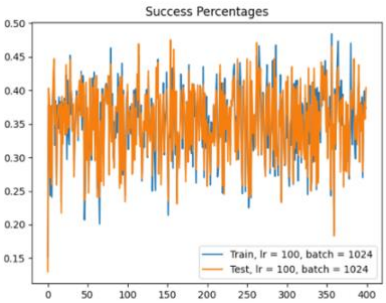


In addition, we explored extreme scenarios to understand the impact of very high and very low learning rates, as well as very large and very small batch sizes.

1. **Very High Learning Rate (100) with Very Large Batch Size (1024)**

In this extreme situation, the learning rate is set to an exceptionally high value, and the batch size is significantly large. As a result, we witness a swift and frequent oscillation in the success rate, with it quickly alternating between low and high values. This dynamic behavior reflects the impact of aggressive parameter updates and the substantial influence of a large batch size on the optimization process.

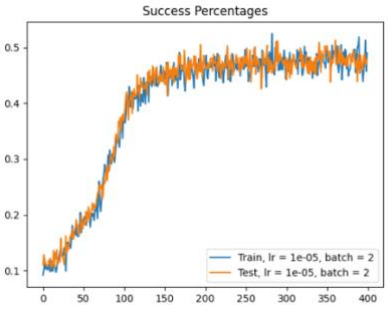
Graph:



2. Very Low Learning Rate (0. 00001) with Very Small Batch Size (2)

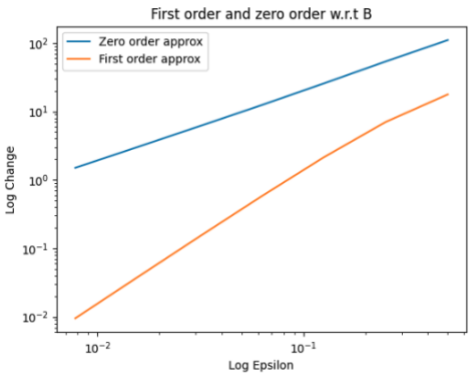
In this extreme scenario, the learning rate is very low, and the batch size is minimal. This led us to very slow convergence and eventually getting stuck in local minima. The small batch made the optimization process not stable.

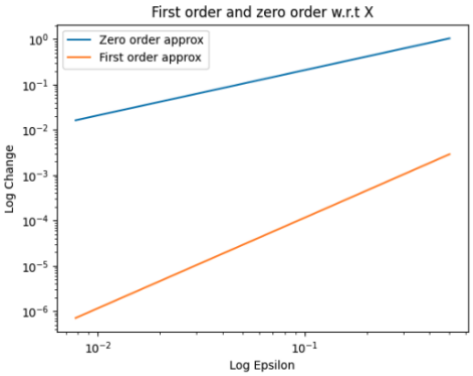
Graph:

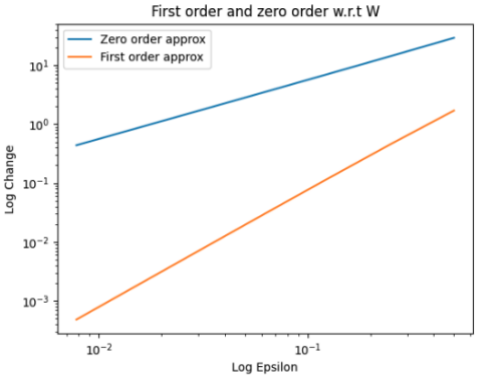


Part 2:

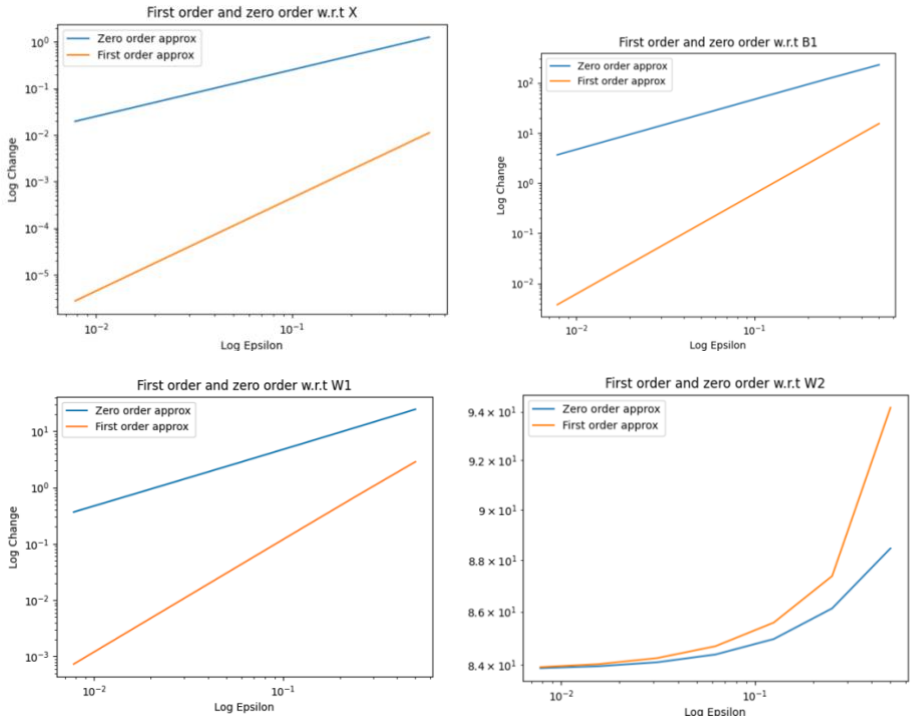
1. The code of the standard neural network attached in part2/part2.1.py
The Jacobian tests attached in Classes/Tests/jacobian_tests.py





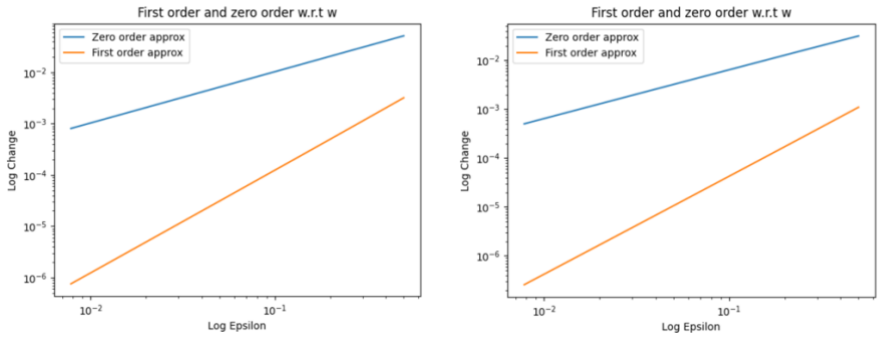


2. The code of the resNet attached in part2/part2.2.py
The Jacobian tests attached in Classes/Tests/jacobian_tests.py



3. The code of the experiment attached in Tests/gradients_test

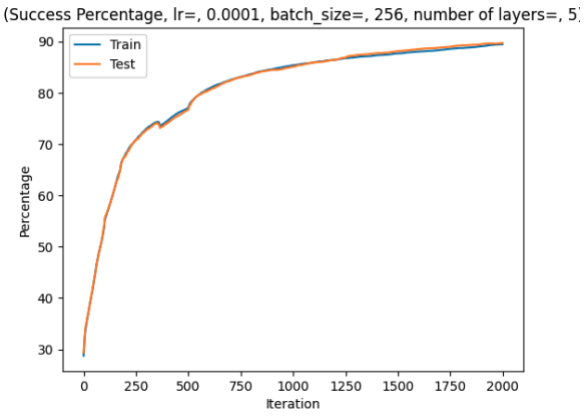
The gradient test of feed forward neural network:

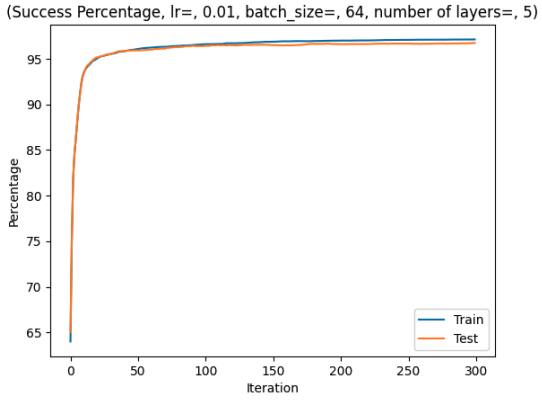


4. The code of the experiment attached in part2/part2.1.4.py

In this experiment, we designed feed forward networks with varying numbers of hidden layers and considered different learning rates and batch sizes.

When a low learning rate of 0.0001 was chosen, the training process had a slow convergence, requiring a huge number of epochs (2000) to reach an apparent stopping point. The model appeared to be stuck in a local minima, achieving a success percentage of 90% while other networks with different learning rates surpassed this performance, achieving success rates exceeding 97%. This observation highlights the sensitivity of the model's convergence to the learning rate parameter.

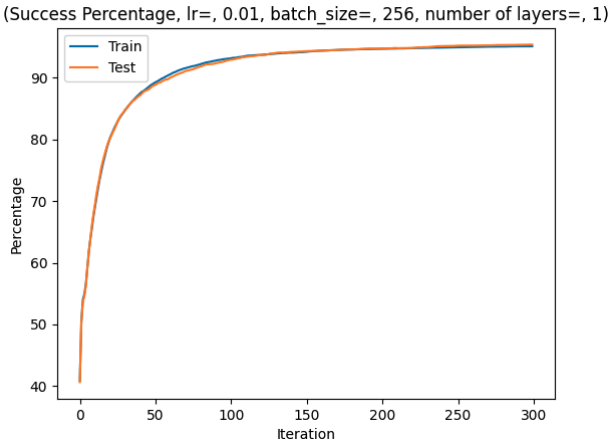




In addition, the number of layers in a neural network plays a crucial role in determining its expressive power and, consequently, its ability to learn intricate patterns in the data. Experimenting with different network depths revealed interesting insights.

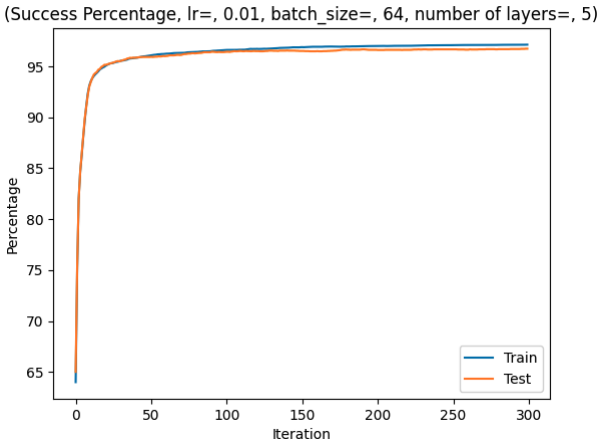
Single Hidden Layer:

Using a learning rate of 0.01 and a batch size of 256, a network with only one hidden layer achieved a test success rate of 92%. While this is a respectable performance, it indicates that a single layer may not capture the complexity of the underlying patterns efficiently.

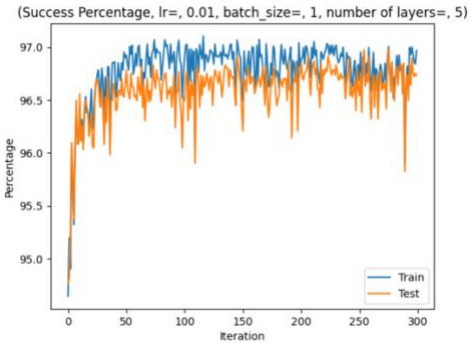


Increasing Depth:

As the number of hidden layers increased, the success percentage also improved. With 5 hidden layers, the success rate rose to 95%, demonstrating that additional layers contribute to better feature extraction and abstraction.



Moreover, Smaller batch sizes led us to more unstable updates while larger batch sizes provide more stable updates.



5. The code of the experiment attached in part2/part2.1.5.py

We requested to reduce the train set to 200 randomly sampled data points. the poor performance on the test set strongly suggests a overfitting, where the neural network appears to have memorized the the limited training samples without extracting broader patterns for robust generalization.