MNIST1D Final Report

1 Initial Goals

The main objectives of this study are as follows:

1. Reproduce the paper to a considerable extent, aiming to obtain and verify the results of most experiments discussed in the paper.
2. Understand the methodologies, algorithms, and assumptions used to gain insights.
3. Learn the key concepts discussed in the paper.
4. Identify the limitations in the use of values for different hyperparameters, particularly to be explored in the ablation study.
5. Conduct an ablation study focusing on the most impactful topics by varying (removing/adding) parameters or values, such as learning rate, hidden layer sizes, number of hidden layers, regularization techniques, and others as required by the topic.
6. Study the effects of components in models, specific hyperparameter values, and the robustness of the model.
7. Perform an error analysis (if any) that arises from changes in values.

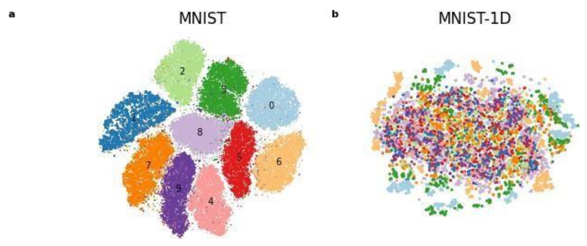


Figure 1: Distribution of data for both Original MNIST and MNIST1D, for MNIST it looks quite structered, but for MNIS1D it is an example or good scatter plot.

2 Results Throughout Study

2.1 Deep Double Descent

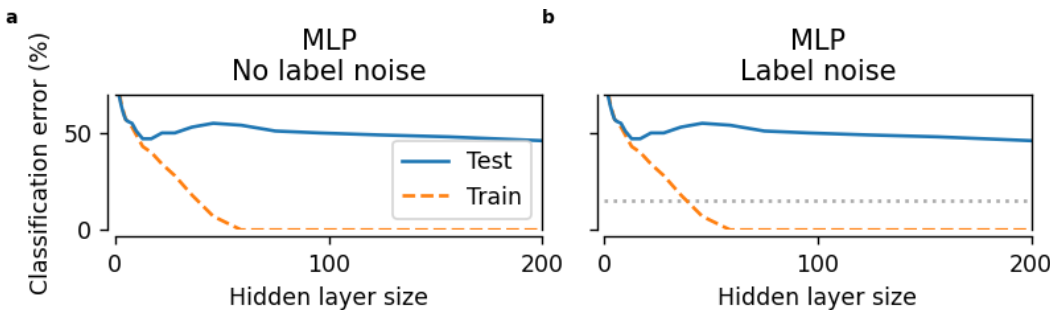


Figure 2: Image from Reproducibility demonstrating Deep Double Descent.

2.1.1 Effect of Hidden Layers (Ablation)

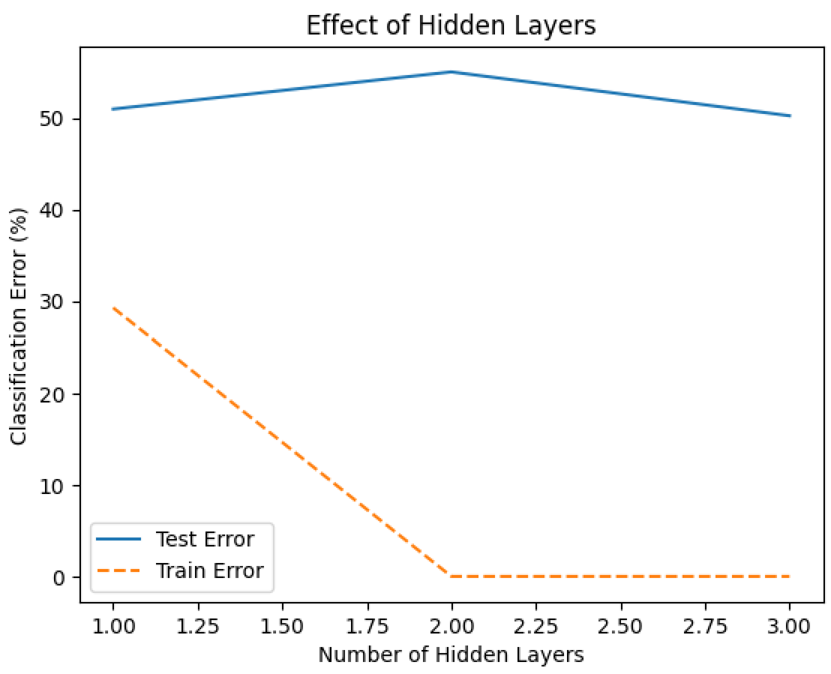


Figure 3: Impact of the number of hidden layers on the model performance.

Description: The graph above shows the effect of the number of hidden layers on train and test error:

1. **Train Error:** As the number of hidden layers increases, the training error decreases steadily, eventually dropping to zero when the number of hidden layers reaches 2 or more.
2. **Test Error:** The curve resembles the Deep Double Descent phenomenon. The test error increases initially, peaking at 2 hidden layers, and then decreases as the number of hidden layers continues to grow.

In this experiment, only the number of hidden layers was varied, while all other parameters were kept constant. The graph highlights that 2 hidden layers mark a critical point of change in performance.

2.1.2 Effect of Label Noise (Ablation)

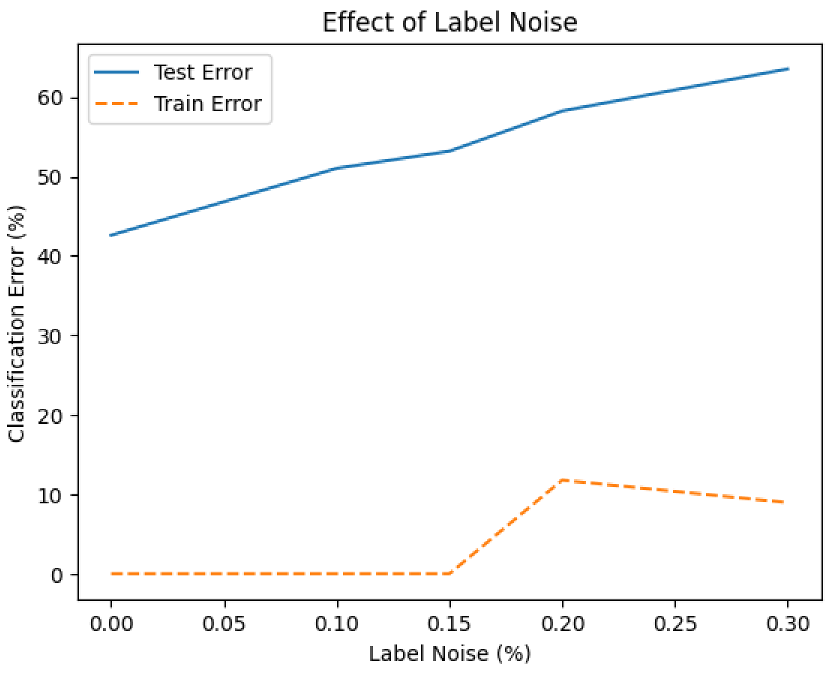


Figure 4: Impact of label noise on model performance. The graph demonstrates how label noise can affect both training and testing errors.

Description: The above graph shows the effect of label noise on both training and testing errors while keeping other parameters constant.

1. **Test Error:** Isn't it attention-grabbing that the test error increases as label noise increases?
2. **Train Error:** Interestingly, the training error remains zero until the label noise reaches 0.15. It then increases up to 0.2 and starts decreasing afterward.

This behavior suggests that the model is attempting to fit the noisy labels during training. This often implies that the model is memorizing the training data, including its noise—a fascinating observation!

2.1.3 Effect of Hidden Layer Sizes (Ablation)

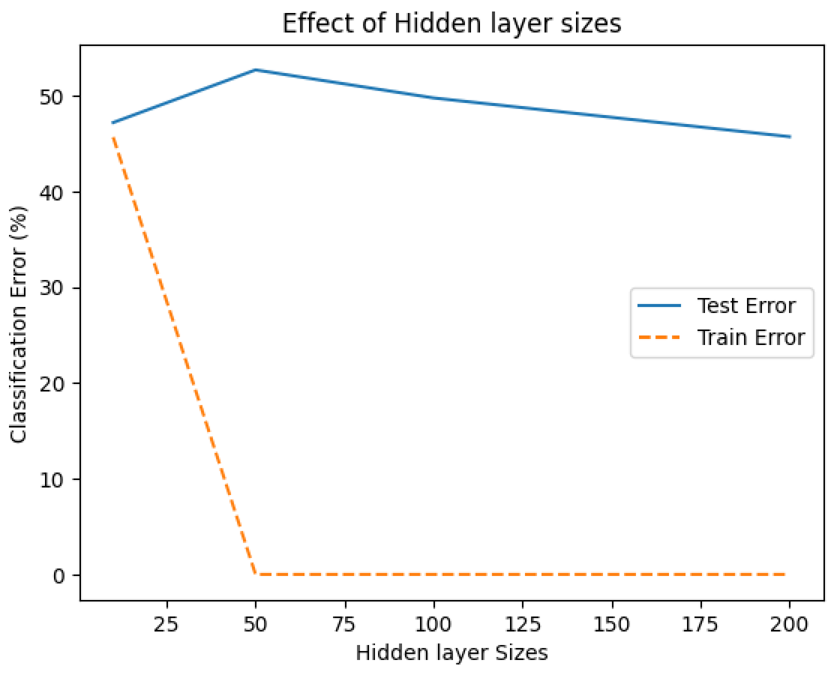


Figure 5: Impact of hidden layer sizes on model performance. It has a similar shape same as of the graph obtained for varying number of hidden layers

2.1.4 Regularization: Dropout Rate (Ablation)

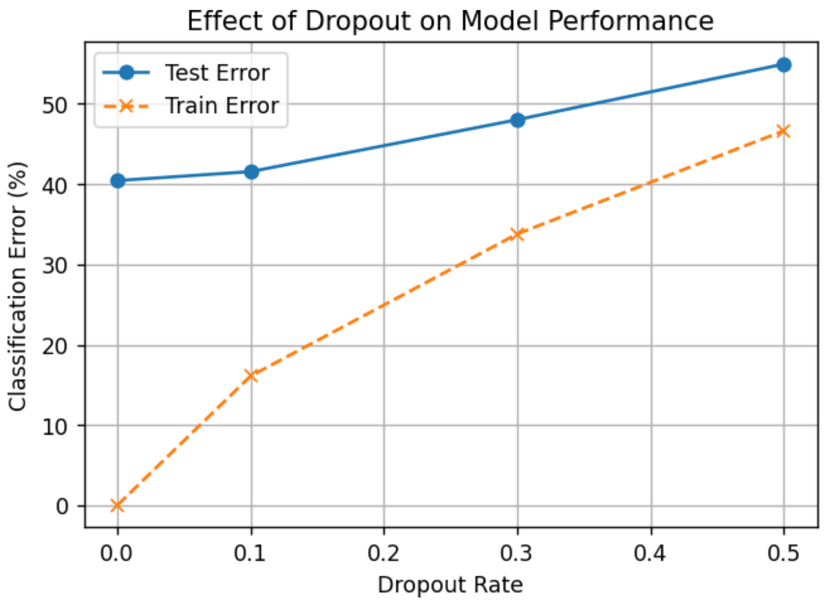


Figure 6: Impact of changing the dropout rate on model performance.

Description: Dropout regularization is a technique used in training neural networks to prevent overfitting and decreasing errors. Here both the errors are increasing as dropout rate increases, the most probable explanation for this can be underfitting. Overfitting is not creating a good impact on this network as underfitting is doing. We can interpret that no dropout is required here as there is no overfitting.

2.1.5 Combining Effects (Ablation)

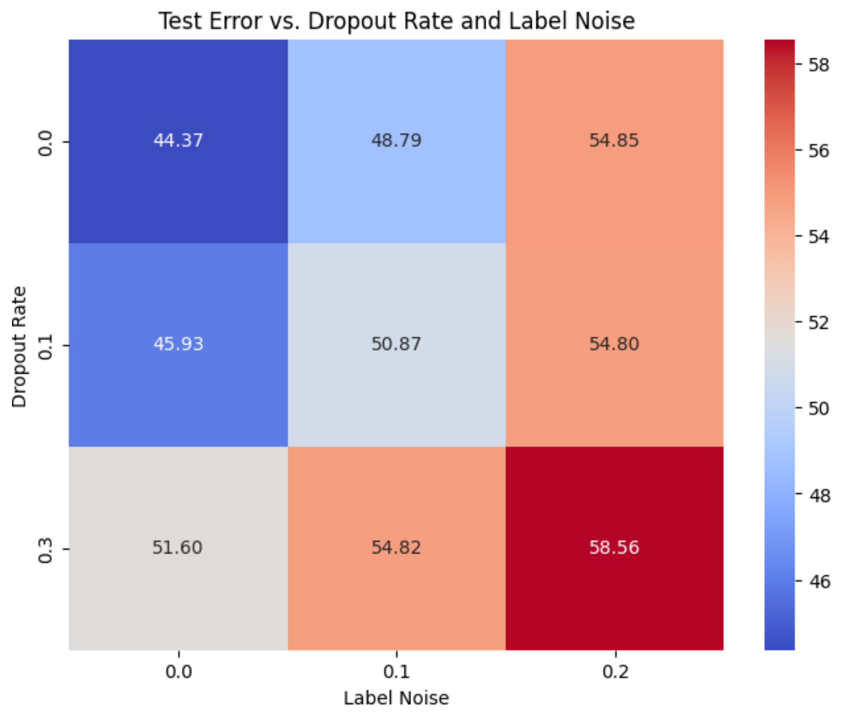


Figure 7: Heatmap for test errors against dropout rate and label noise. This visualization summarizes the combined effect of these two factors on the model's performance.

hidden_size	n layers	label noise	dropout_rate	train error	test error
10	1	0.0	0.0	tensor(45.4250)	tensor(52.3917)
10	1	0.0	0.1	tensor(53.1250)	tensor(58.4667)
10	1	0.0	0.3	tensor(59.3750)	tensor(62.8417)
10	1	0.1	0.0	tensor(47.4750)	tensor(49.5417)
10	1	0.1	0.1	tensor(55.5000)	tensor(56.6000)

Table 1: Model performance metrics for different configurations. The table presents errors for various combinations of hyperparameters.

2.2 Meta-Learning Learning Rate

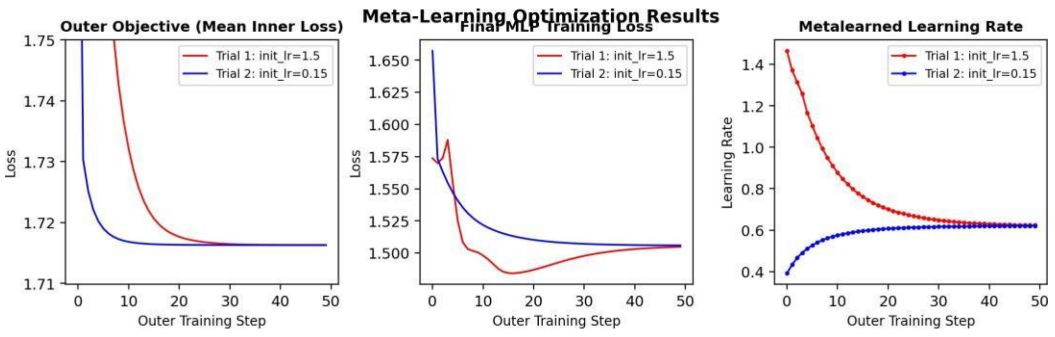


Figure 8: Image from Reproducibility.Meta-learning: learning rate by considering two learning rates with significant difference and checking whether they converge.

2.2.1 Effect of varying initial lr (Ablation)

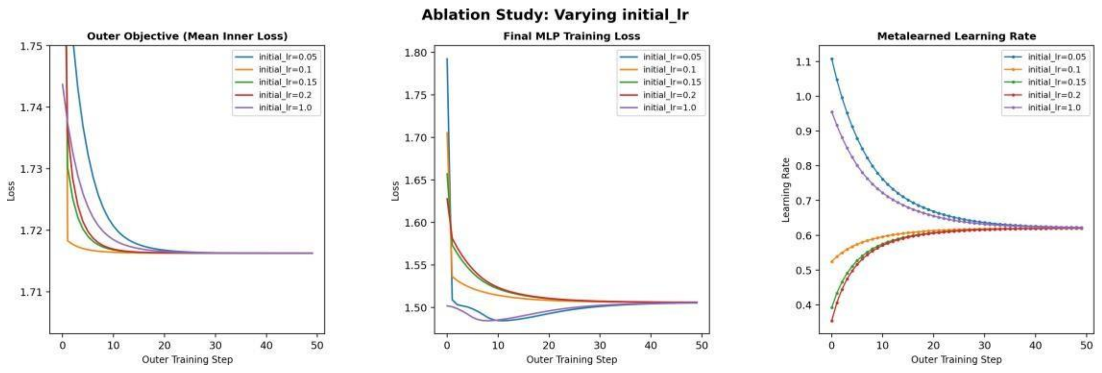


Figure 9: The graph above shows the convergence of different initial learning rates to same values for respective graph. This shows how effectively the model can minimize the values for respective graph.

2.2.2 Effect of varying hidden layer size (Ablation)

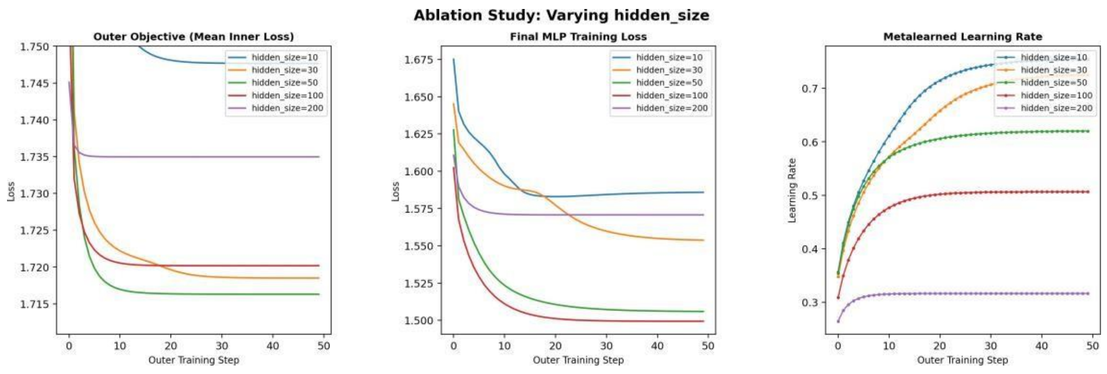


Figure 10: The graph above shows, no matter what hidden layer size you initialize, after some iterations the values saturates out.

2.2.3 Effect of varying meta-learning rate (Ablation)

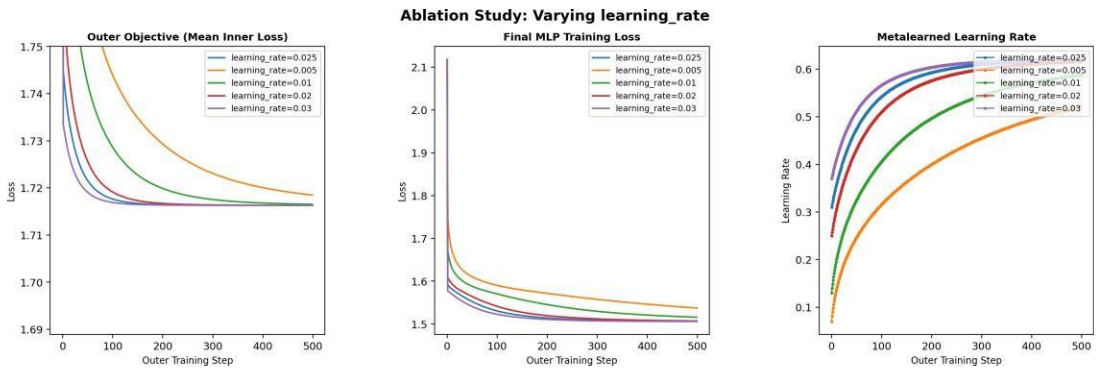


Figure 11: See the graphs above, we can observe every value is moving towards saturation, for this i have increased the number of iteration per lr to view the effect more reliably

2.2.4 Upperbound to meta-learning rate (Ablation)

Learning Rate	Step	Mean Inner Loss	Final Inner Loss	Inner LR	Time (s)
0.1	3	1.775e+00	1.526e+00	1.147e+00	1.353
0.1	25	1.750e+00	1.500e+00	9.751e-01	1.058
0.1	48	1.745e+00	1.502e+00	9.502e-01	1.032
0.3	3	4.002e+25	3.600e+25	1.571e+25	1.384
0.3	25	4.002e+25	3.600e+25	1.571e+25	1.078
0.3	48	4.002e+25	3.600e+25	1.571e+25	1.093
0.34	3	5.852e+34	1.151e+35	-4.525e+31	1.875
0.34	25	5.852e+34	1.151e+35	-4.525e+31	1.045
0.34	48	5.852e+34	1.151e+35	-4.525e+31	1.053
0.35	3	inf	inf	-1.285e+34	1.374
0.35	25	inf	inf	-1.285e+34	1.079
0.35	48	inf	inf	-1.285e+34	1.095
0.4	3	nan	nan	nan	1.958
0.4	25	nan	nan	nan	1.059
0.4	48	nan	nan	nan	1.066
0.5	3	nan	nan	nan	1.358
0.5	25	nan	nan	nan	1.073
0.5	48	nan	nan	nan	1.070

Table 2: Here, is the illustration(**small part of iteration**) of different learning rate and corresponding results. While doing the ablation study on meta-learning rates, i found that the values obtained at ouput through iterating through different learning rates are not valid(not measurable) after 0.35(excluding). According to the observation the valid range for initialization includes numbers less then or equal to 0.1.

2.3 Meta-learning Activation Function

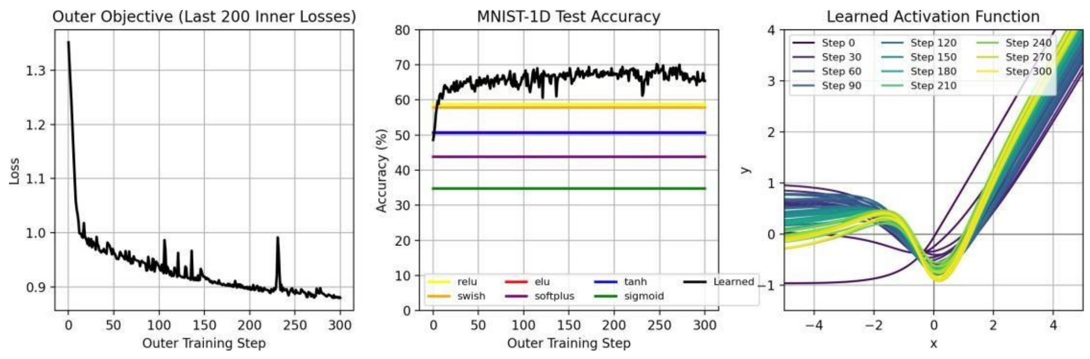


Figure 12: Image from Reproducibility.

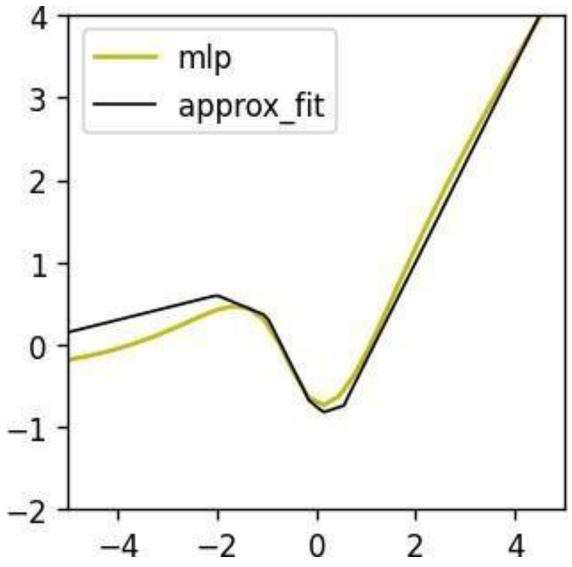


Figure 13: Image from Reproducibility. Approximation of the function,final test accuracy of the model in a baseline results of 67.4

2.3.1 Changing μ (Momentum Rate) (Ablation)

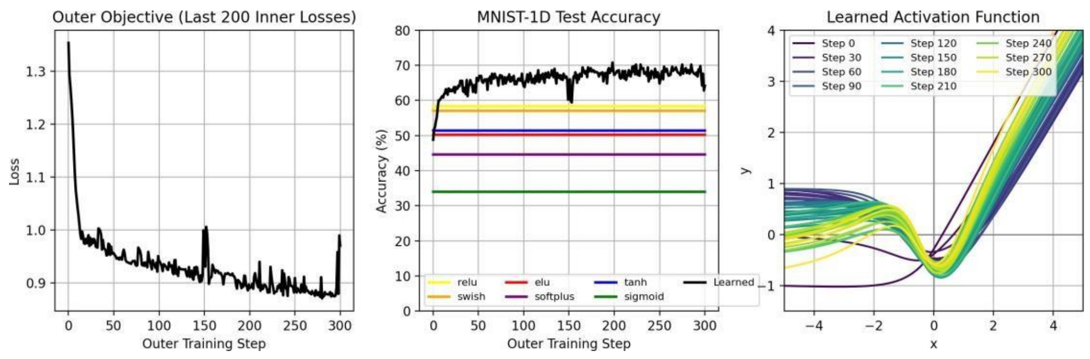


Figure 14: Effect of momentum rate $\mu = 0.3$ on the activation function.

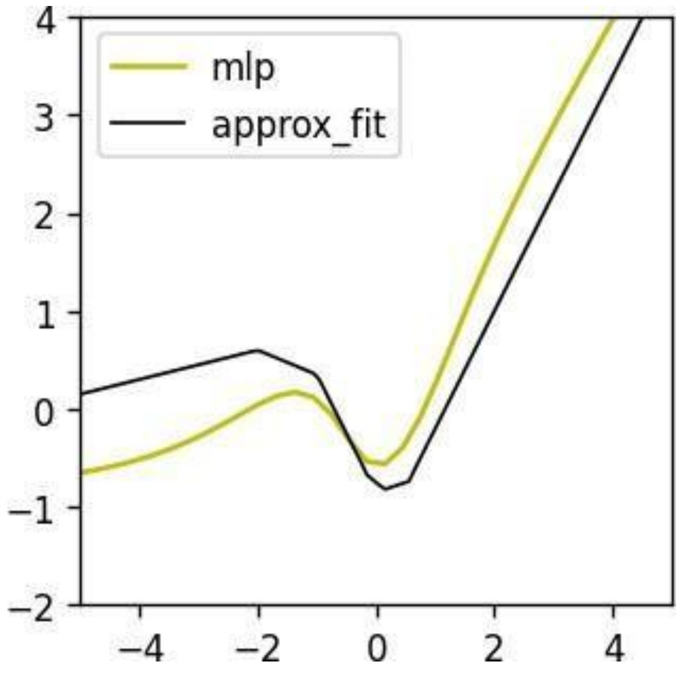


Figure 15: Approximation of the function for $\mu = 0.3$ and final test accuracy of 66.0.

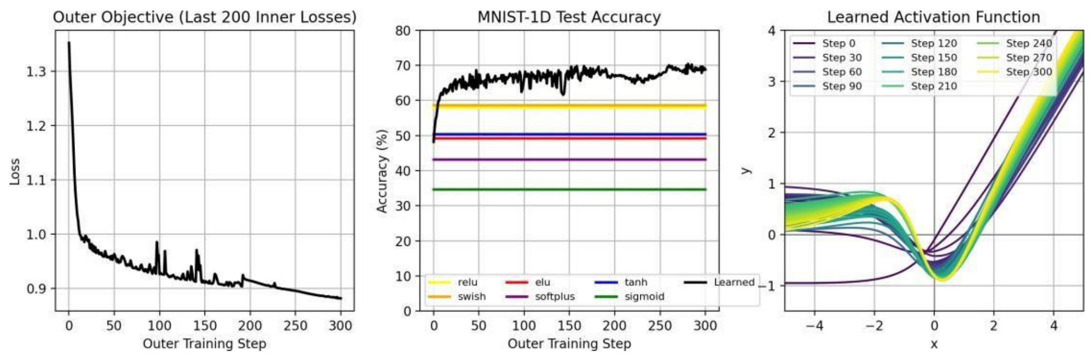


Figure 16: Effect of momentum rate $\boldsymbol{\mu} = 0.5$ on the activation function.

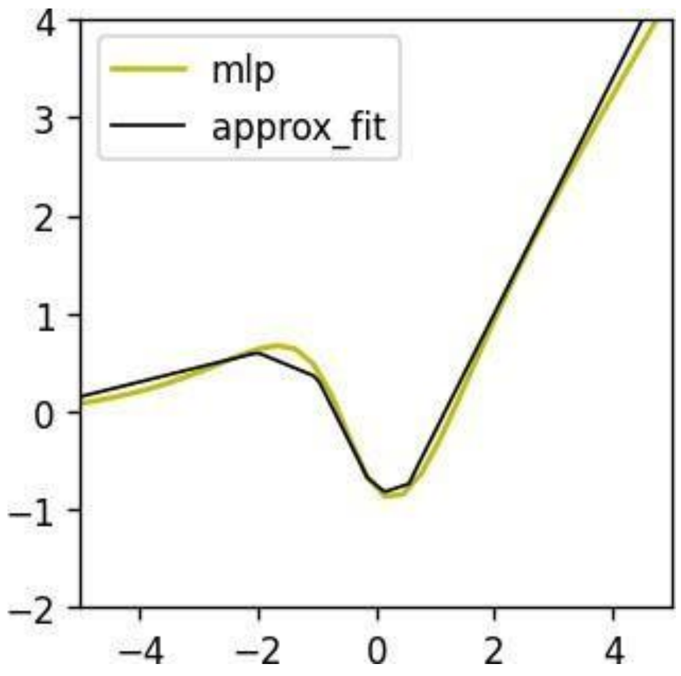


Figure 17: Approximation of the function for $\boldsymbol{\mu} = 0.5$ and final test accuracy of 68.2.

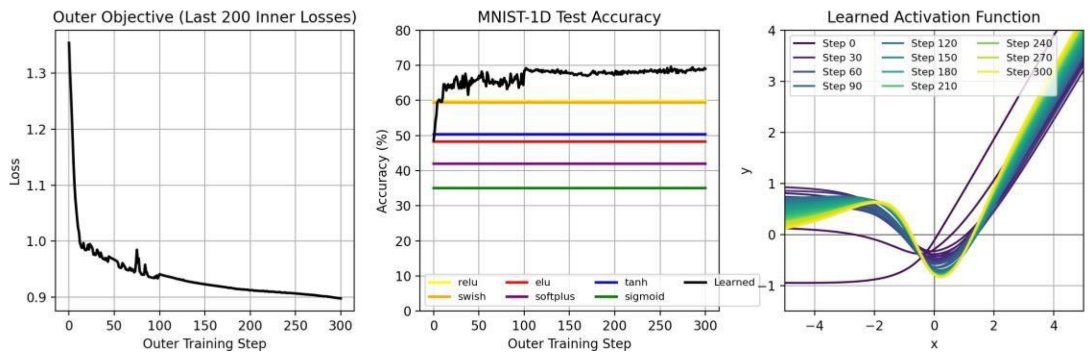


Figure 18: Effect of momentum rate $\boldsymbol{\mu} = 0.6$ on the activation function.

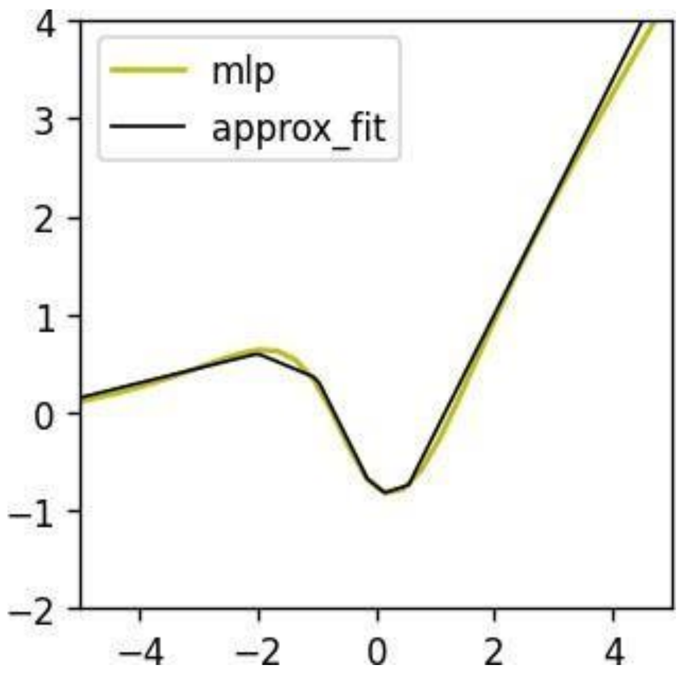


Figure 19: Approximation of the function for $\boldsymbol{\mu} = 0.6$ and final test accuracy of 67.8.

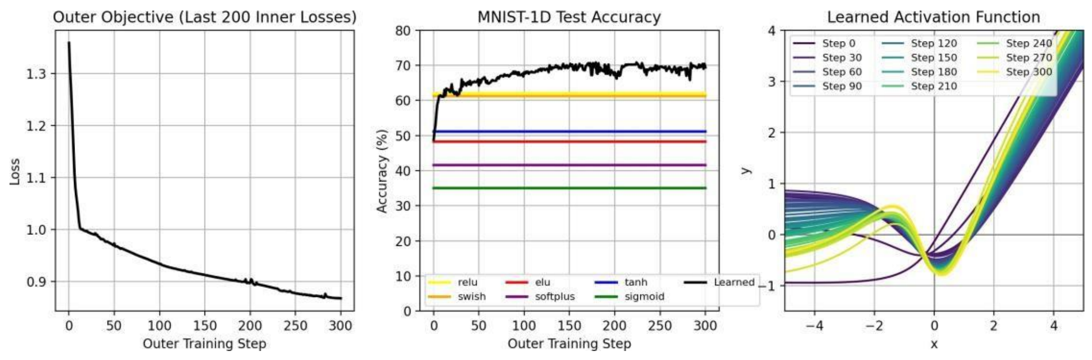


Figure 20: Effect of momentum rate $\boldsymbol{\mu} = 0.7$ on the activation function.

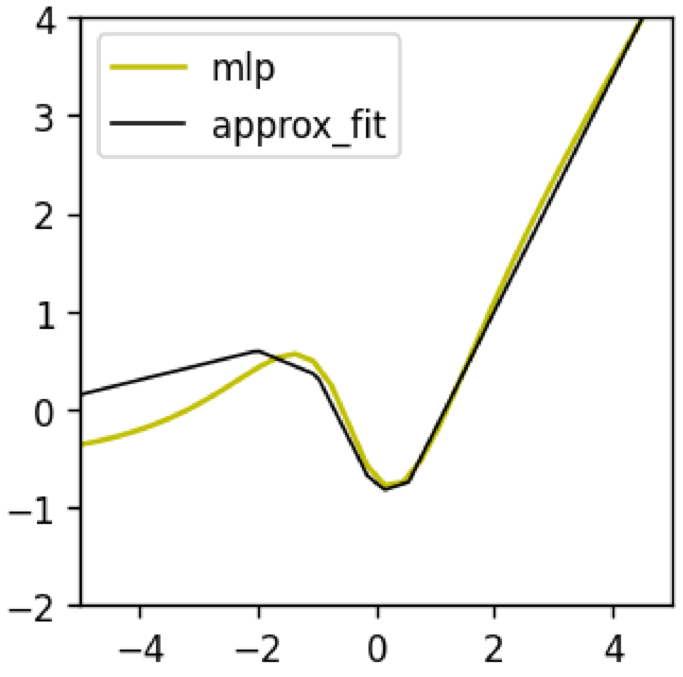


Figure 21: Approximation of the function for $\boldsymbol{\mu} = 0.7$ and final test accuracy of 66.8.

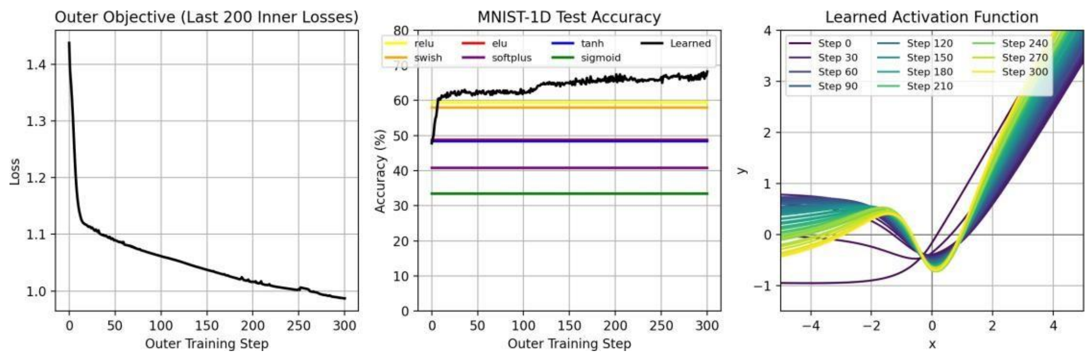


Figure 22: Effect of momentum rate $\mu = 0.9$ on the activation function.

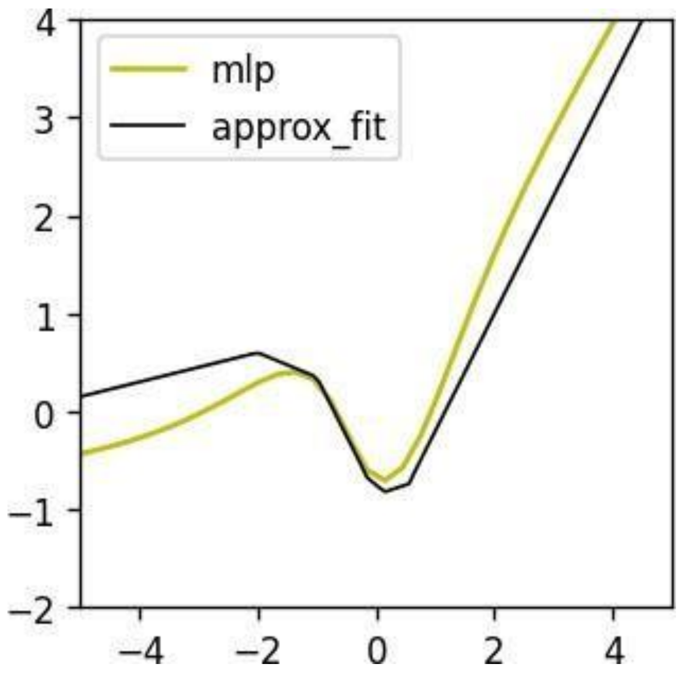


Figure 23: Approximation of the function for $\mu = 0.9$ and final test accuracy of 64.4.

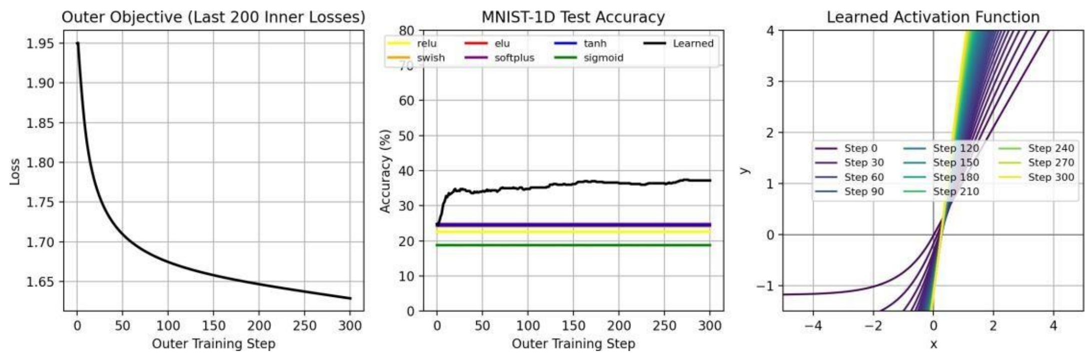


Figure 24: Effect of momentum rate $\boldsymbol{\mu} = 1.0$ on the activation function.

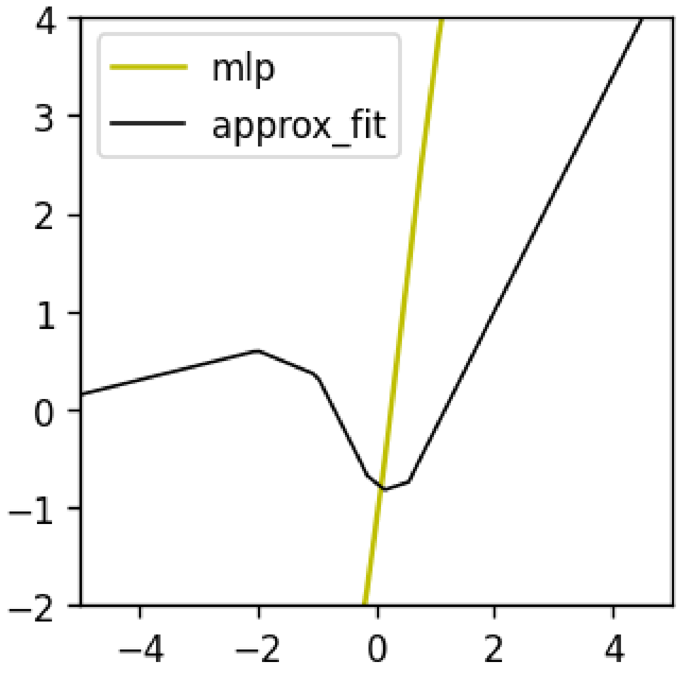


Figure 25: Approximation of the function for $\boldsymbol{\mu} = 1.0$ and final test accuracy of 30.4.

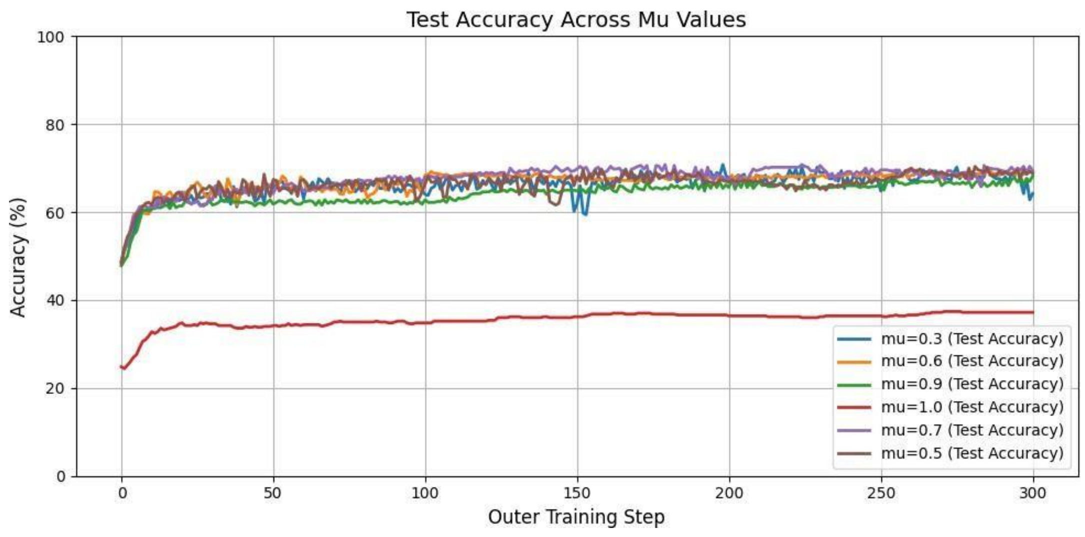


Figure 26: Combining Results

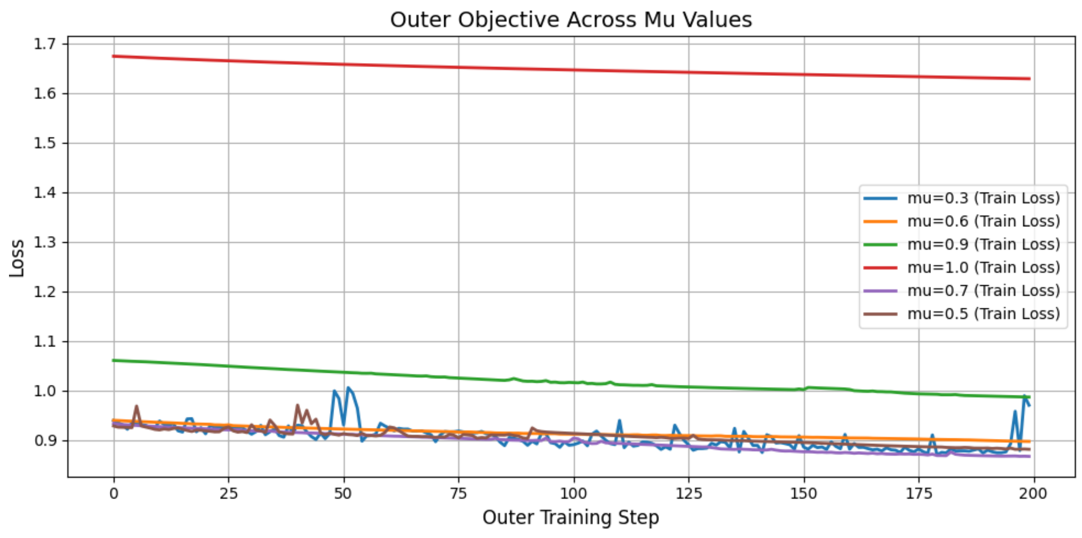


Figure 27: Combining Results

2.3.2 Changing hidden layer sizes (Ablation)

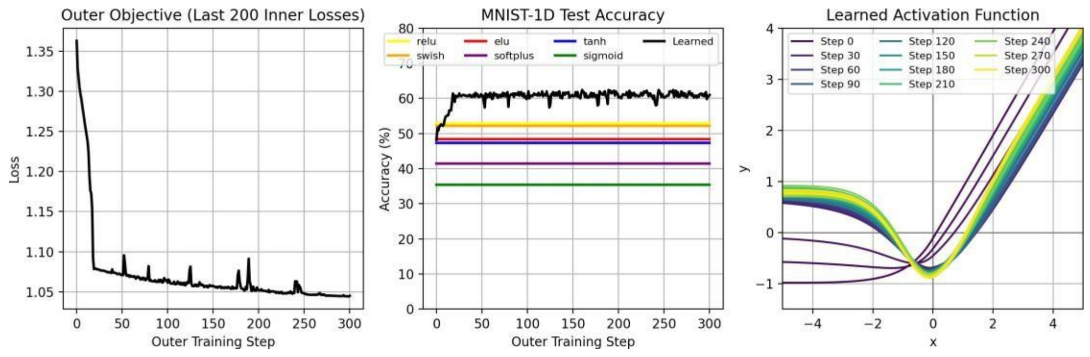


Figure 28: Effect of hs = 50 on the activation function.

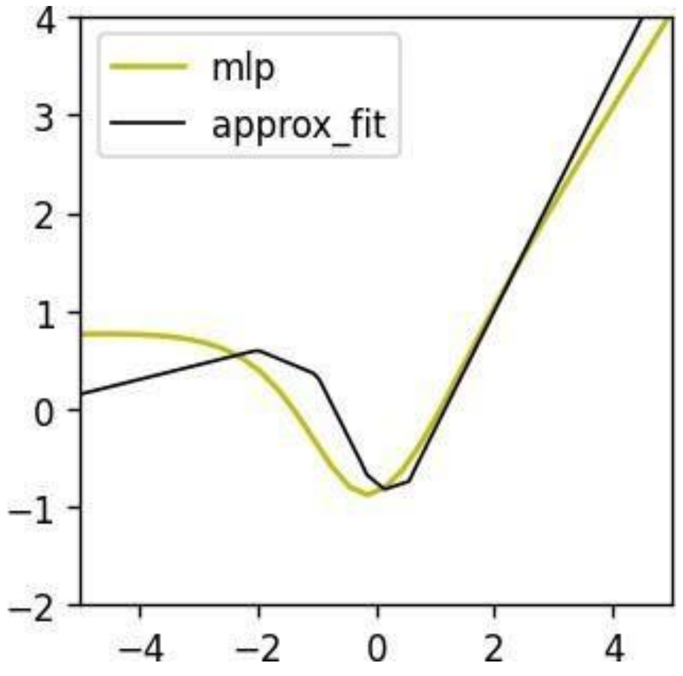


Figure 29: Approximation of the function for hs = 50 and final test accuracy of 59.4.

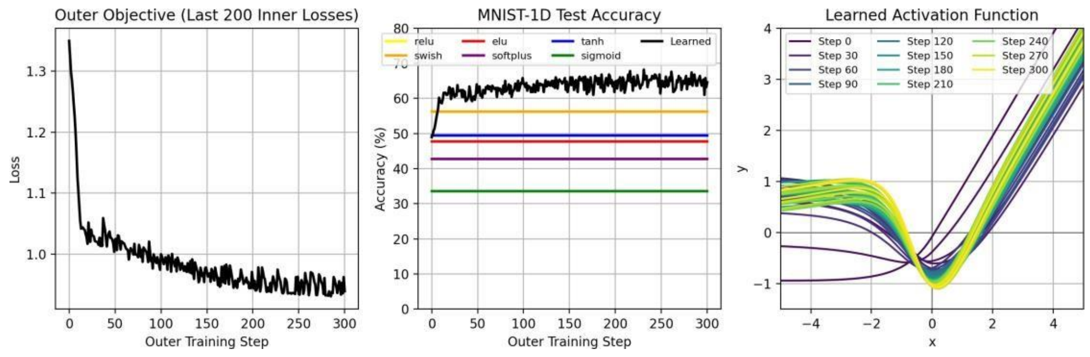


Figure 30: Effect of hs = 100 on the activation function.

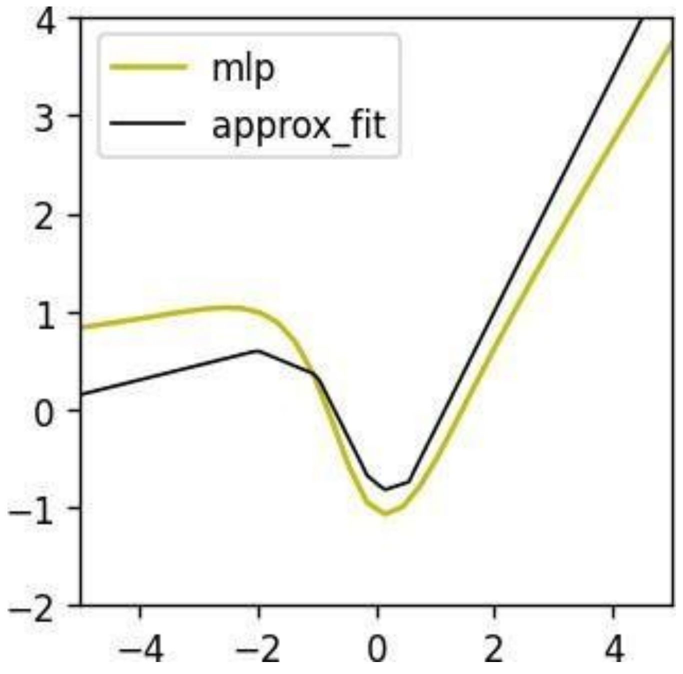


Figure 31: Approximation of the function for hs = 100 and final test accuracy of 63.4.

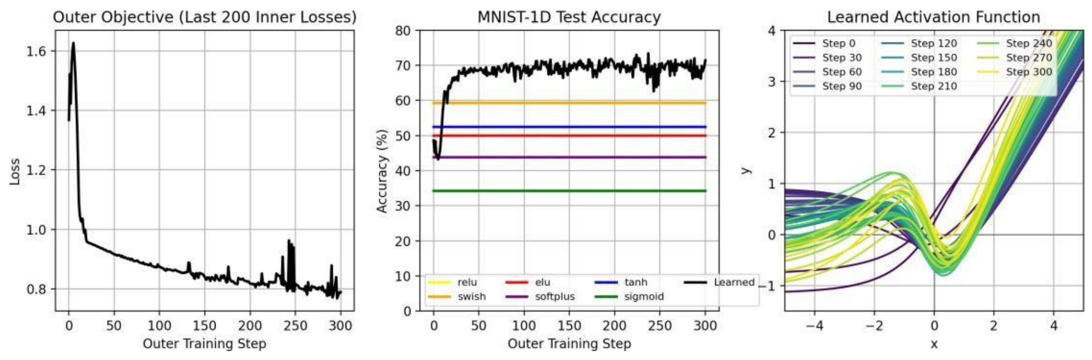


Figure 32: Effect of hs = 200 on the activation function.

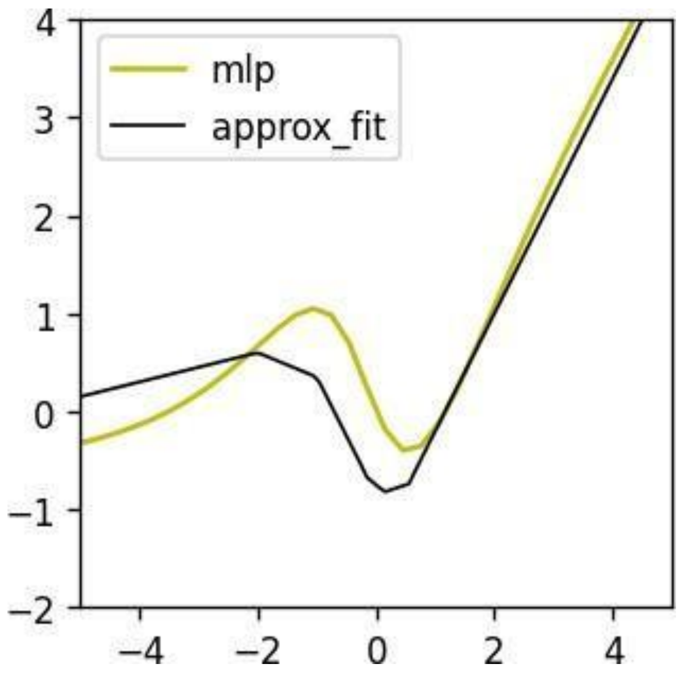


Figure 33: Approximation of the function for hs = 200 and final test accuracy of 69.0.

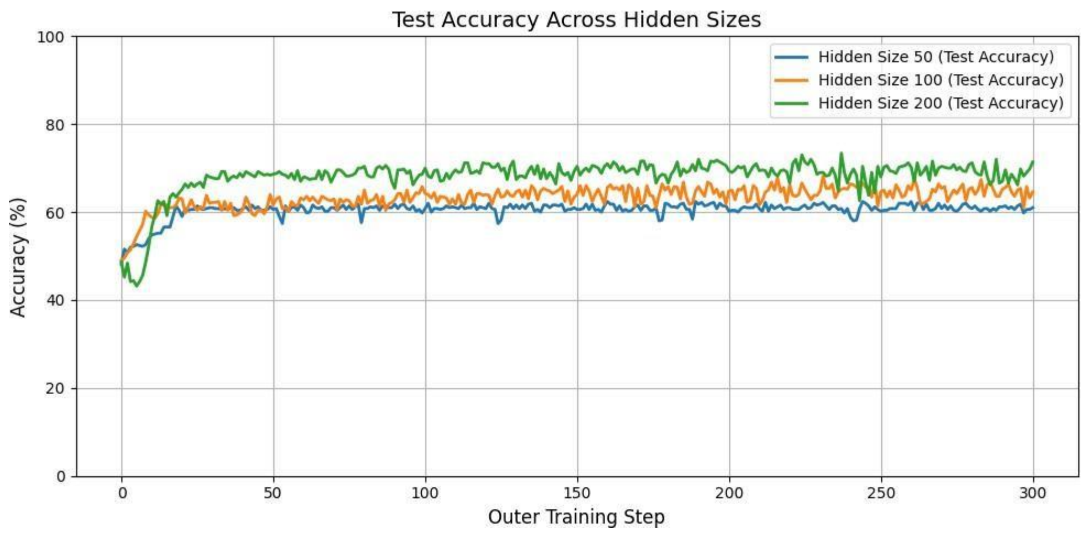


Figure 34: Combining Results

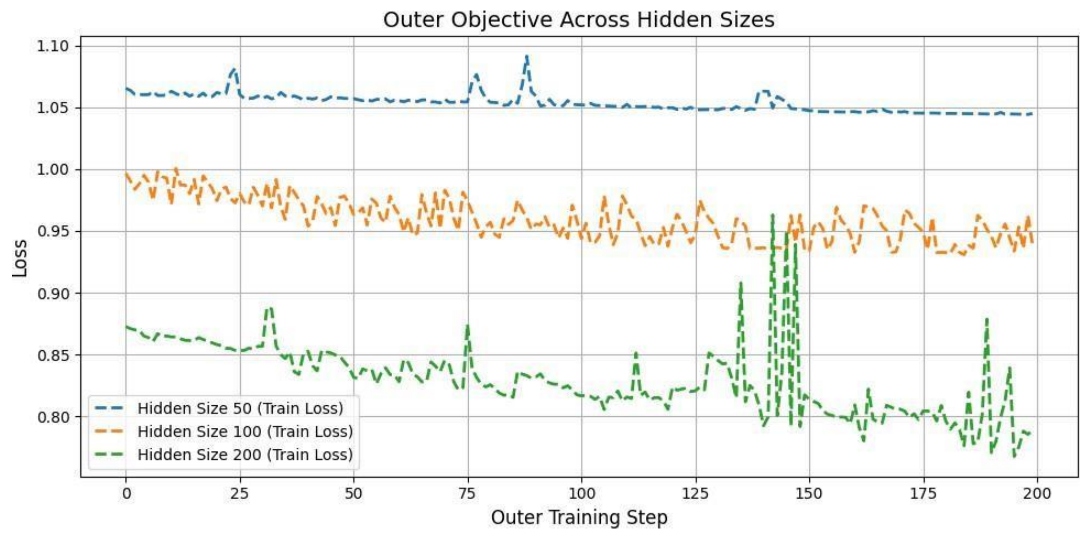


Figure 35: Combining Results

Description: As the hidden layer size is increasing both train and test accuracy is increasing. I also tried for size 400 and 500 but colab has RAM issure for these sizes or higher as these number of sizes require lot of RAM consumption even on GPU.

Try observing the graphs for learned activation function for different sizes above, the size is incresing the distance between the curves for different steps is also increasing and for 50 all curves for different steps are quite closed or regular.

I needed to perform the experimet for differnt sizes in differnent notebook because when i tried in same notebook the experiment was giving memory error so i had to perform in different notebooks, combined the the pickles files obtained and plot the results.

2.3.3 Removing noise Statement (Ablation)

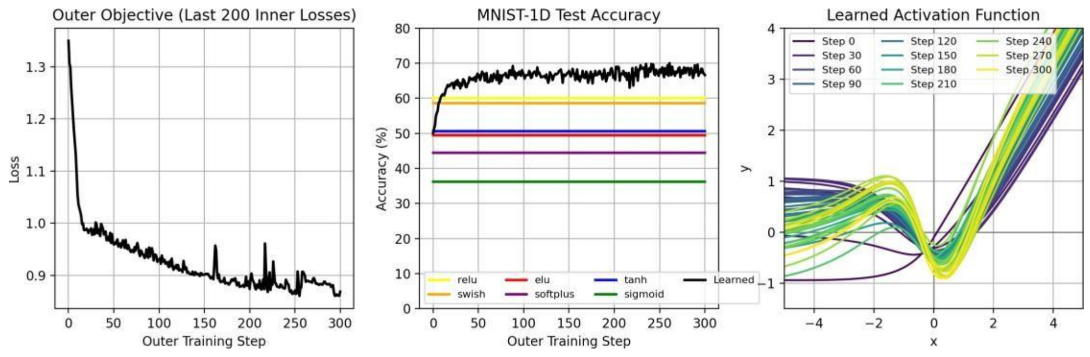


Figure 36: Without Noise

Comparison of Test Accuracy: When noise is present(figure 11), the test accuracy levels of tanh and elu overlap, indicating similar performance. However, in the absence of noise, tanh achieves marginally higher accuracy compared to elu, showing a clear distinction in their effectiveness.

Comparison of Learned Activation: In the learned activation graph, the distance between different curves for various steps is more pronounced when noise is absent compared to when noise is present. This suggests that the absence of noise allows for better differentiation and separation between activation behaviors over different steps.

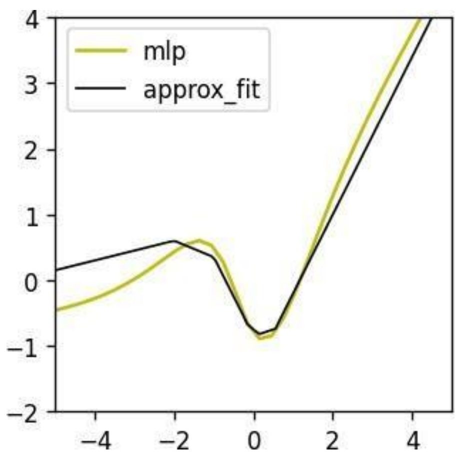


Figure 37: Without Noise, final test accuracy : 67.6

Effect of Noise on Test Accuracy: Removing the statement (h = h + 0.1*(2*torch.rand(*b1.s that introduces noise in the network does not significantly affect the test accuracy. With noise, the accuracy was 67.4%, and without noise, it was 67.6%, indicating minimal impact.

Robustness in MNIST-1D: In the case of MNIST-1D, the data is relatively simple and low-dimensional, which likely makes the model inherently robust to noise, reducing the effect of its presence or absence on the overall performance.

2.4 Lottery ticket and spatial inductive biases

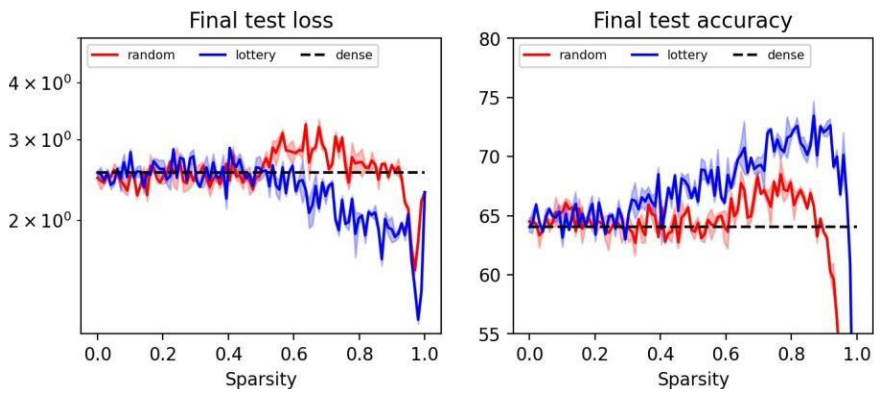


Figure 38: comparing the accuracies with varying sparsity.It can be seen that loss graph of sub network (lottery ticket) is decreasing at faster rate that the original network and accuracy is is higher for that sub network

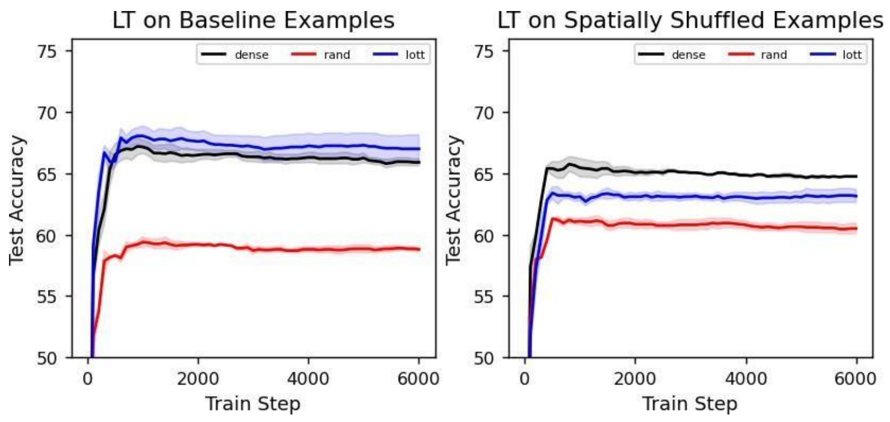


Figure 39: spatially shuffling the dataset effect the accuracy.Isn't it crazy that, on original dataset lottery ticket performed best, but when shuffled spatially dataset is used original network performed best, that means altering data spatially.

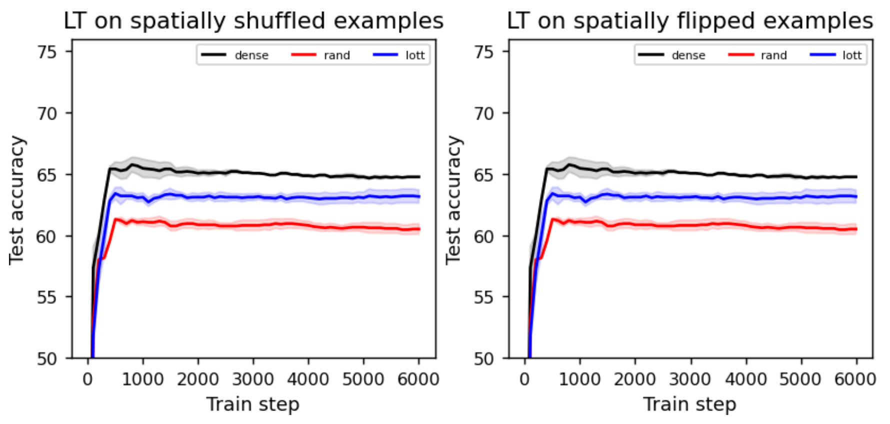


Figure 40: Spatially shuffled vs spatially flipping the dataset.It seems no difference, great!

2.4.1 Perturbation (Ablation)

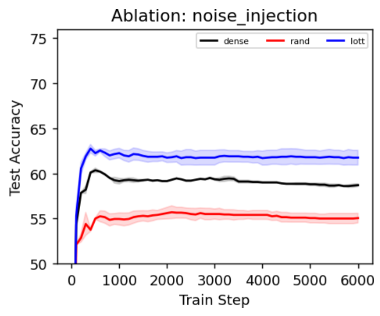


Figure 41: Injecting noise to data

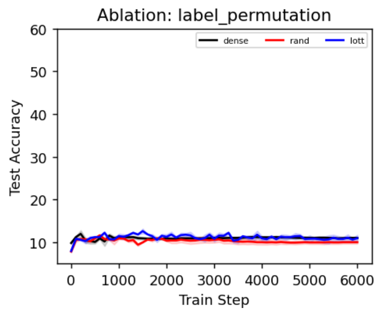


Figure 42: Randomizing the labels

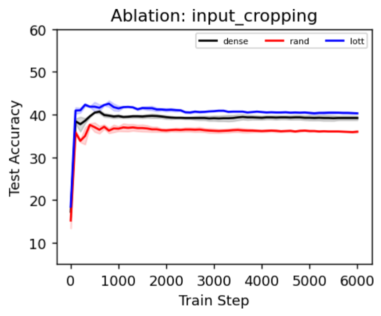


Figure 43: Input cropping

2.4.2 Exponential sparsity schedule (Ablation)

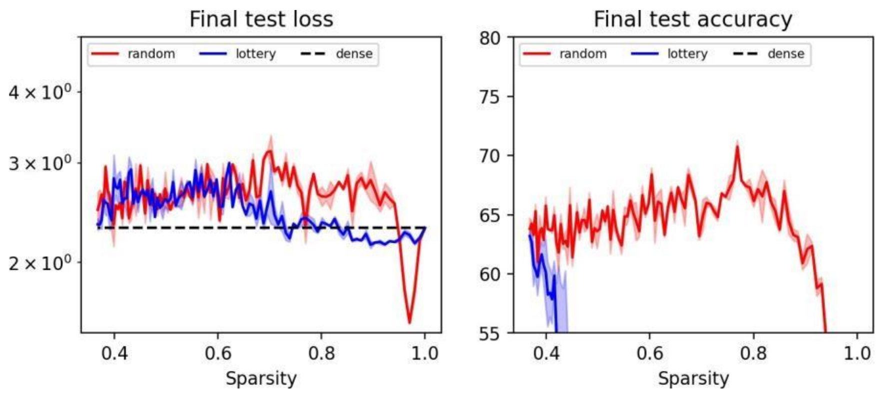


Figure 44: Exponential sparsity schedule is not advantageous over linear schedule

2.4.3 Introducing Xavier init (Ablation)

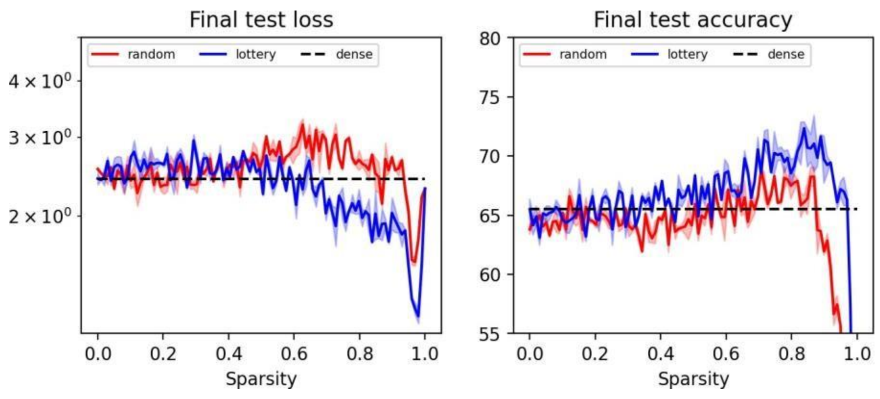


Figure 45: Doesn't make much of a difference, compared to figure 37

2.4.4 Initialization with different learning rates (Ablation)

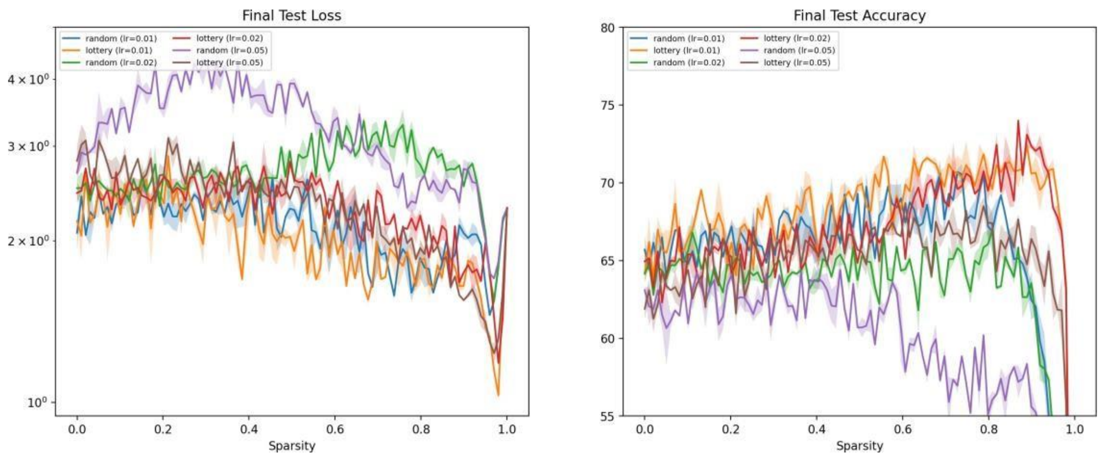
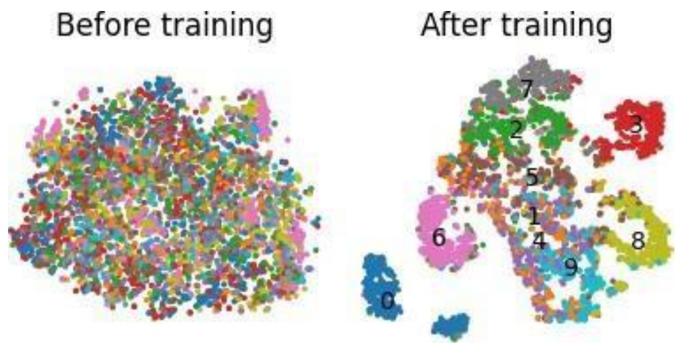


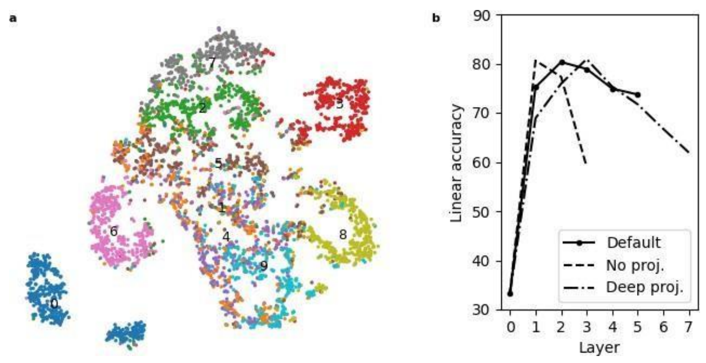
Figure 46: Accessing the accuracies(or losses) with different lrs

2.5 Self-supervised contrastive learning on MNIST-1D

2.5.1 Distribution of data before and after training



2.5.2 Projection heads Assessment



3 Details of the Final Solutions

1. Increasing the number of hidden layer sizes and hidden layers (up to hardware limits such as GPU and RAM) improves results. However, hidden layer sizes should not exceed 300 due to memory and runtime constraints.
2. No regularization is required for optimal performance.
3. The value of $\boldsymbol{\mu}$ (momentum rate), if used, should range between 0.3 and 0.7, with 0.5 being most preferable.
4. A linear sparsity schedule outperforms an exponential sparsity schedule.
5. Meta-learning hyperparameters is more effective than random selection.

4 Challenges Faced

1. Difficulty in understanding the code due to the use of newer syntaxes and libraries.
2. Challenges in comprehending the complex equations presented in the paper.
3. Lack of good resources to learn certain concepts.
4. Runtime and RAM limitations on Colab, especially during large epochs required in some experiments.
5. Disconnections after long runtimes (over 11 hours), causing incomplete results.
6. Persistent RAM usage issues on Colab during experiments.
7. Long runtimes (over 6.5 hours) on alternative platforms with better GPUs, yet errors in storing results persisted.
8. Frequent errors during code modifications, necessitating the use of the original code for certain experiments.

5 Learnings

Throughout the course of this study, I have learned the following:

1. Deep Double Descent and its implications on model performance.
2. Gradient-based meta-learning of hyperparameters.
3. Meta-learning activation functions and their impact on model convergence.
4. Lottery ticket pruning as an efficient model training strategy.
5. Self-supervised contrastive learning and its application.
6. The importance of the projection head in self-supervised learning tasks.
7. Observing that modifying a single hyperparameter can have a significant impact on model accuracy.
8. The advantage of representing complex data in 1D for model training, which reduces computational load.

6 Future Work

1. Reproducing research papers and analyzing the effects of parameters in different scenarios will deepen my understanding of machine learning and deep learning.
2. I plan to work on Generative Adversarial Networks (GANs) and aim to build a model capable of generating images based on given prompts. This challenge will require high-end machines to execute effectively.