CS 599: Foundations of Deep Learning

Measuring Catastrophic Forgetting in MLPs

Aiden Seay

October 26, 2025

CS599-DeepLearning-Lab2 Repository

1. Overview

The purpose of this lab is to demonstrate how traditional training approaches cannot handle incrementally learning new tasks without catastrophically forgetting previously learned training data. In this lab, we will demonstrate catastrophic forgetting as our model will be trained on 10 tasks. Each task will have a permuted MNIST dataset to train from. Each task after the first will use the previous weights, which will allow us to measure how the performance degrades as new permuted data is introduced. In this lab report, we will go through testing different loss functions, dropout rates, network depth, and optimizers to see which parameters are best to minimize the catastrophic forgetting concept. The next section will go into more detail on how this experiment will be run.

2. Experimental Setup

Dataset and Tasks

The Fashion MNIST dataset consists of grayscale images of fashion items such as shirts, shoes, bags, and coats. Each image is 28×28 pixels in size and is flattened into a 784-dimensional vector (since $28 \times 28 = 784$), which serves as the input to the neural network.

The dataset is divided into three subsets:

- **Training Set:** 54,000 samples used for learning model parameters.
- **Validation Set:** 6,000 samples used for tuning hyperparameters and monitoring overfitting.

- **Test Set:** 10,000 samples used for final performance evaluation.

In this lab, each task uses a permuted version of the Fashion-MNIST dataset, where the pixel order is randomized using a unique permutation vector. This setup allows us to study catastrophic forgetting by training on multiple distinct tasks derived from the same dataset distribution. The first task will go through 50 epochs, and the rest will all be trained from 20 epochs. An example of a permuted version of the dataset is below:

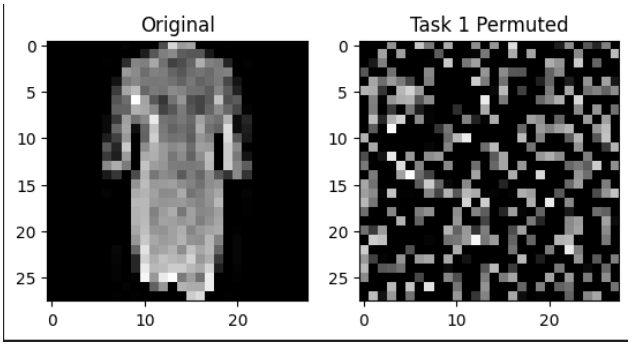


Figure 1: Permutted Image Example Compared to Original

Model Configuration

The model that will be tested will have the following baseline parameters. When testing a parameter, the rest will be in the following conditions:

Table 1: Baseline Model Configuration

Parameter	**Value**
Model Depth	2
Dropout Rate	0.3
Loss Function	NLL (Negative Log-Likelihood)
Optimizer	Adam
Learning Rate	0.001
Batch Size	100
L1 Regularization	1×10^{-5}
L2 Regularization	1×10^{-4}
Seed	12345

Note: We changed the batch size from 32 to 100 to help increase the speed for testing.

Evaluation Metrics

The four different ways we will evaluate our model are ACC, BWT, TBWT, and CBWT. Below, we will explain in detail what each metric means and its equation:

ACC - Average Accuracy

This metric takes the average accuracy across all tasks (1-10):

$$ACC = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}$$

BWT - Backward Transfer

This metric measures the difference between the model's performance on previously learned tasks before and after going through and learning the new task with the different permuted datasets.

$$BWT = \frac{1}{T-1}\sum_{i=1}^{T-1}(R_{T,i} - R_{i,i})$$

TBWT - True Backward Transfer

This metric measures how much the final performance on previous tasks is different from if all tasks were trained independently.

$$TBWT = \frac{1}{T-1}\sum_{i=1}^{T-1}(R_{T,i} - G_{i,i})$$

CBWT - Cumulative Backward Transfer

This metric measures the forgetting of a task throughout all 10 tasks. It shows the performance drop during the execution of each task and doesn't just show forgetting at the very end, after all 10 tasks are completed. This will show you where specifically the forgetting starts.

$$CBWT(t) = \frac{1}{T-t}\sum_{i=t+1}^{T}(R_{i,t} - R_{t,t})$$

3. Effect of Loss Function

For this section of the lab, we tested the effect of different loss functions had on the accuracy and catastrophic forgetting. Below, you can see that the best average accuracy is *33.25%* and the largest BWT is *-0.5965*. Although it is the best out of the four loss functions, catastrophic forgetting still has a massive impact on the grand scheme of things.

Table 2: Effect of Loss Function on ACC and BWT

Loss Function	ACC	BWT
NLL	**0.3325**	**-0.5965**
L1	0.2190	-0.7307
L2	0.2310	-0.7179
L1 + L2	0.2190	-0.7278

The lower the accuracy, the worse the model performs. The larger the negative number for BWT, the more catastrophic the forgetting is. Below is a graphical interpretation where you can see the differences in accuracy and backward transfer:

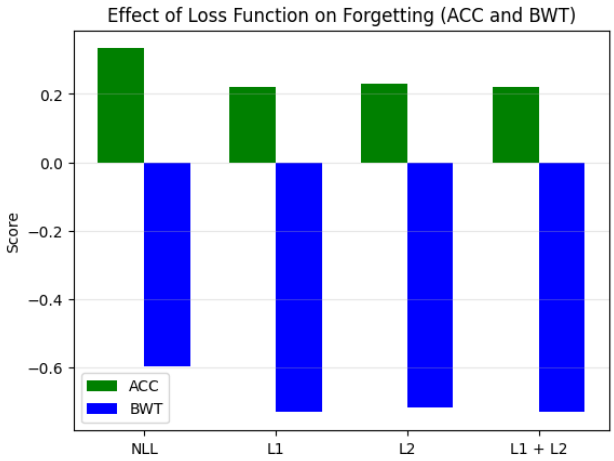


Figure 2: Effect of Loss Function on ACC and BWT

Now, let's take a look at the change in accuracy through each task. Take a look at the plot below (Figure 3). You can see there are massive drops in the first couple of tasks, with gradual improvements near the end, but there is still significant evidence of forgetting.

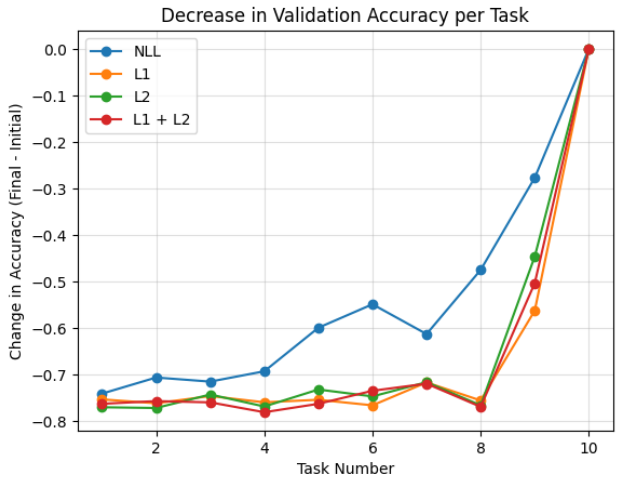
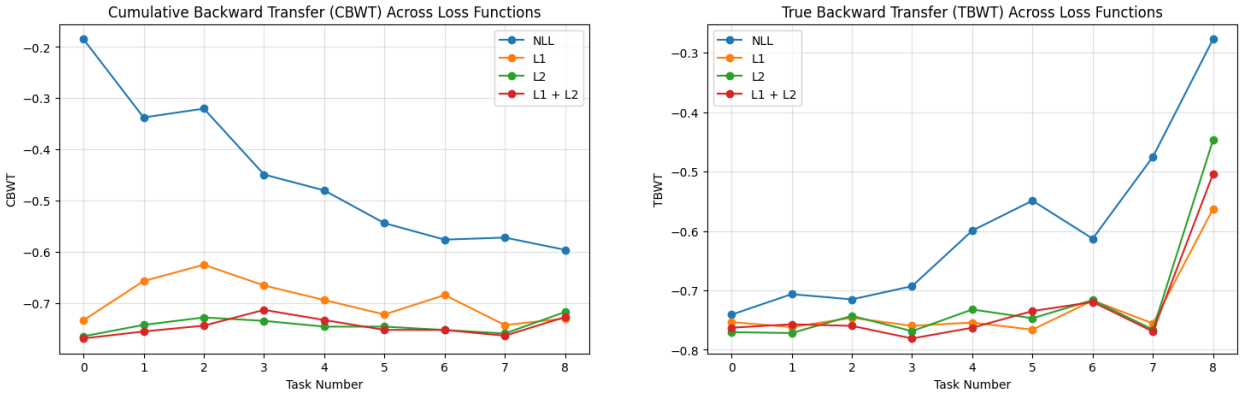


Figure 3: Effect of Loss Function on Validation Accuracy

NLL has fewer decreases in changes of accuracy when compared to its counterparts. Now, let's take a look at TBWT and CBWT.



(a) Cumulative Backward Transfer (CBWT)

(b) True Backward Transfer (TBWT)

Figure 4: Comparison of CBWT and TBWT for Loss Function Effect

These plots show that although NLL has the best accuracy and BWT, it is actually prone to more forgetting. As all the points are trending down in CBWT, the more the model is forgetting. CBWT shows, by each task, how much the model forgets. When looking at TBWT, all the curves go up, meaning newer tasks experience less forgetting. This shows that while NLL provides better accuracy, L2 may offer the best stability by decreasing the amount of forgetting.

4. Effect of Dropout

For this section of the lab, we tested the effect of different dropout rates had on the accuracy and catastrophic forgetting. Below you can see that the best average accuracy is *33.54%* and the largest BWT is *-0.5524*. Again, 0.5 dropout has the best results out of the rest of the dropout rates, but it still has catastrophic forgetting.

Table 3: Effect of Dropout Rate on ACC and BWT

Dropout Rate	**ACC**	**BWT**
No Dropout (0.0)	0.3006	-0.6379
Medium Dropout (0.3)	0.3183	-0.6090
High Dropout (0.5)	**0.3354**	**-0.5524**

The lower the accuracy, the worse the model performs. The larger the negative number for BWT, the more catastrophic the forgetting is. Below is a graphical interpretation where you can see the differences in accuracy and backward transfer when the dropout rate changes:

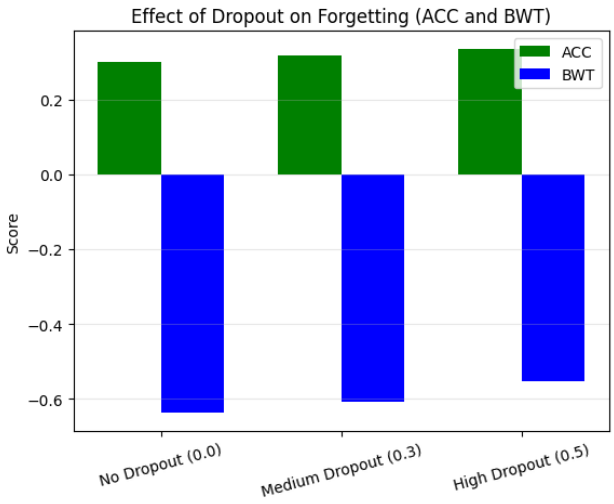


Figure 5: Effect of Loss Function on ACC and BWT

Now, let's take a look at the change in accuracy through each task. Take a look at the plot below (Figure 6). You can see below that when we introduce dropout into our model, there are fewer changes in our accuracy between each task.

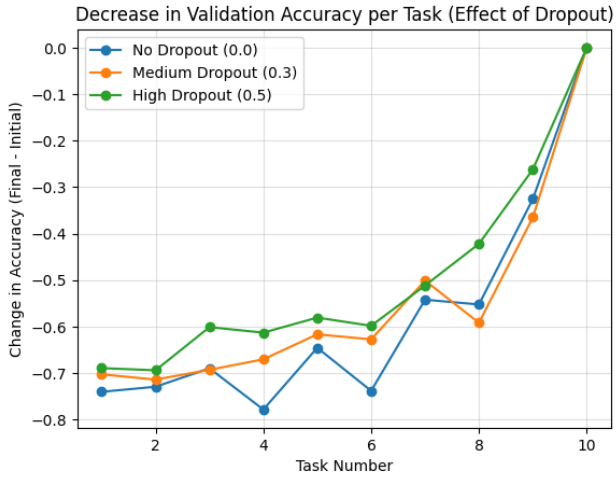
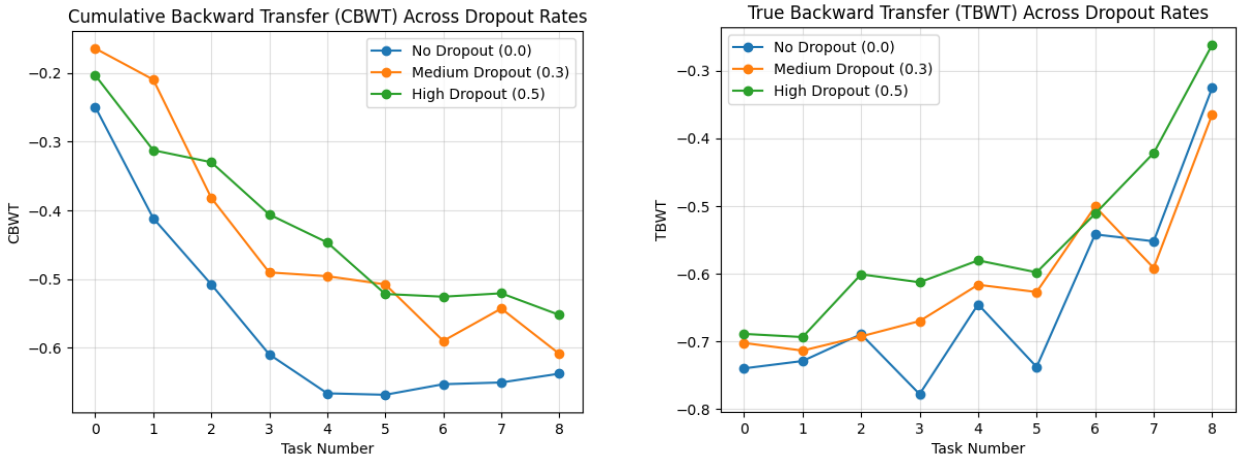


Figure 6: Effect of Dropout Rate on Validation Accuracy

The dropout rate = 0.5 has fewer decreases in changes of accuracy when compared to its counterparts. Now, let's take a look at the TBWT and CBWT metrics.



(a) Cumulative Backward Transfer (CBWT) (b) True Backward Transfer (TBWT)

Figure 7: Comparison of CBWT and TBWT for Dropout Rate

When taking a look at the CBWT graph, increasing the dropout will generally lead to less forgetting. This is also shown in TBWT. The higher the dropout rate is, the smaller the performance drops are. This shows that the 0.5 dropout rate will offer the best stability by decreasing the amount of forgetting.

5. Effect of Model Depth

For this section of the lab, we tested the effect of different network depths had on the accuracy and catastrophic forgetting. There are three different models, each with a depth of 2, 3, or 4, each consisting of 256 hidden nodes. Below you can see that the best average accuracy is *35.97%* and the largest BWT is *-0.5663*.

Table 4: Effect of Network Depth on ACC and BWT

Network Depth	ACC	BWT
2 Layers	**0.3597**	**-0.5663**
3 Layers	0.3537	-0.5670
4 Layers	0.3266	-0.5784

The lower the accuracy, the worse the model performs. The larger the negative number for BWT, the more catastrophic the forgetting is. Below is a graphical interpretation where you can see the differences in accuracy and backward transfer as the network depth changes.

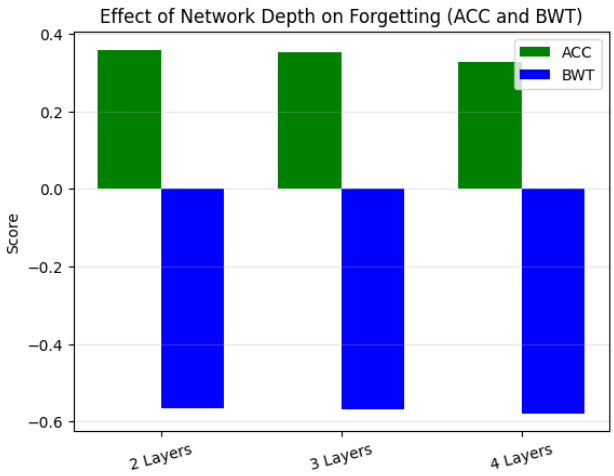


Figure 8: Effect of Network Depth on ACC and BWT

Now, let's take a look at the changes in accuracy through each task. Take a look at the plot below (Figure 9). You can see that when we increase the network depth, each layer is performing very similarly. Although it is close, the model performs at its best between two or three hidden layers.

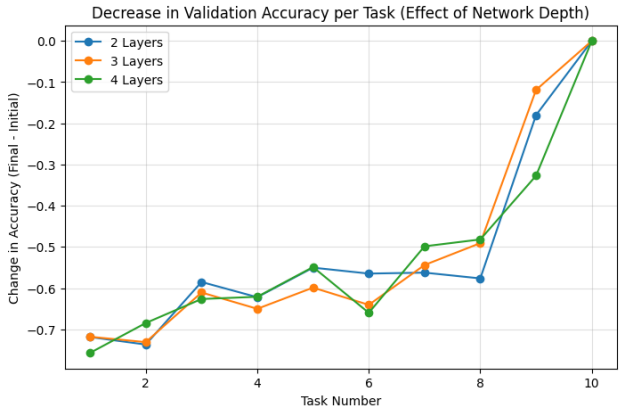
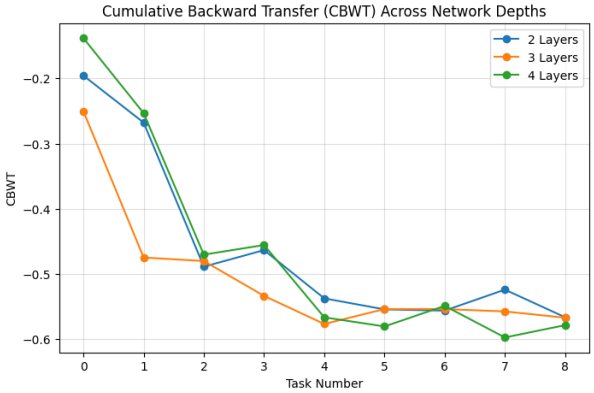
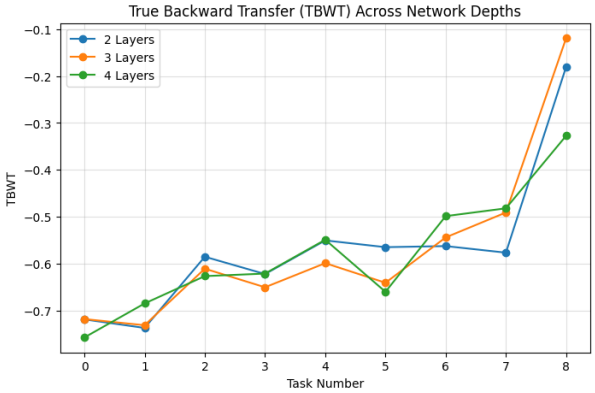


Figure 9: Effect of Network Depth on Validation Accuracy

The network depth = 2 has fewer decreases in changes of accuracy when compared to its counterparts. It is extremely close, though. Sometimes four layers perform better, and sometimes three layers perform better. That just depends on the task they are on.



(a) Cumulative Backward Transfer (CBWT)



(b) True Backward Transfer (TBWT)

Figure 10: Comparison of CBWT and TBWT for Network Depth

See above for the results of the CBWT and TBWT graphs. All network depths are performing roughly the same when compared to each other, so it is hard to make a decision solely based on them. When comparing it with accuracy, though, 2 layers is the clear winner, followed closely by 3 layers.

6. Effect of Optimizer Type

For this last section of the lab, we tested the effect of different optimizers, including ADAM, RMSProp, and Stochastic Gradient Descent. Below you can see the best average accuracy is at *70.24%* and the largest BWT is *-0.118*. You can see the full results in Table 5.

Table 5: Effect of Optimizer on ACC and BWT

Optimizer	ACC	BWT
Adam	0.3418	-0.5879
SGD	**0.7024**	**-0.1180**
RMSProp	0.3248	-0.5944

As you can see above, SGD outperforms both of the other optimizers tested. Below is the graphical interpretation, where you can see how much better SGD is compared to the other optimizers.

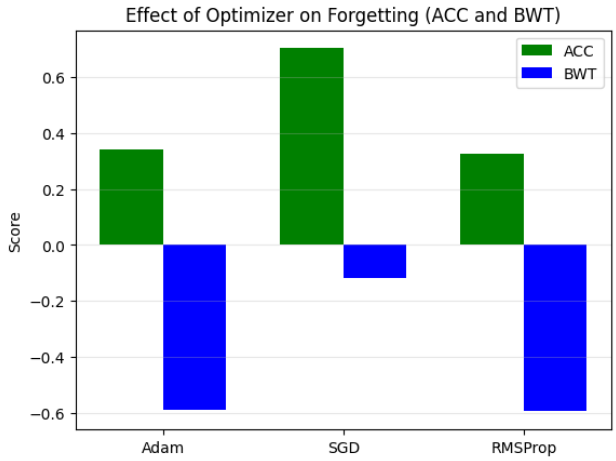


Figure 11: Effect of Optimizer Type on ACC and BWT

In Figure 12 below, SGD constantly has the lowest decrease in accuracy for every task below. Adam and RMSProp have very similar results, having much worse results than SGD.

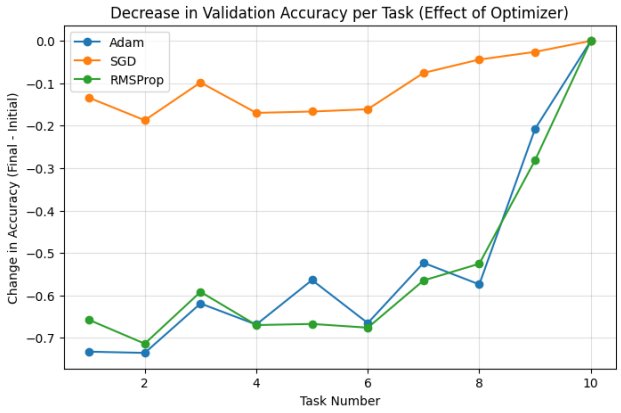
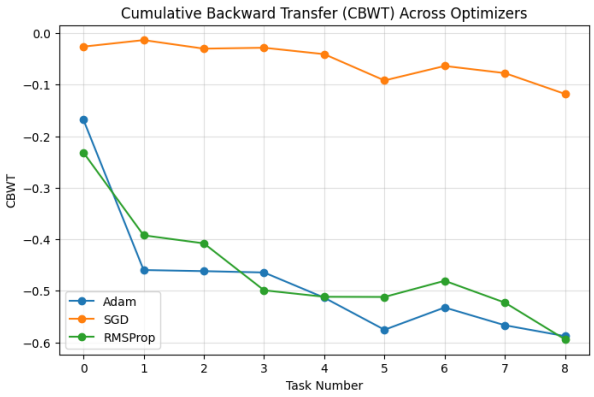
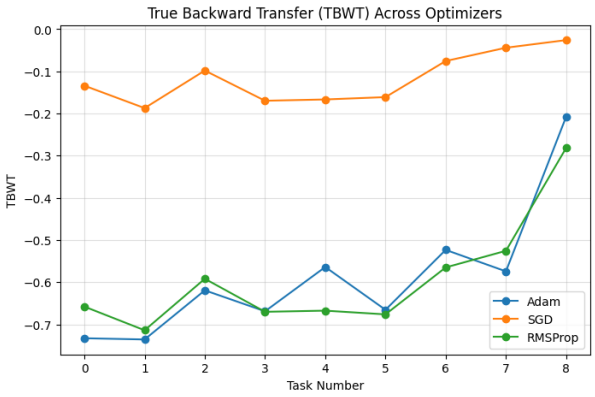


Figure 12: Effect of Optimizer Type on Validation Accuracy



(a) Cumulative Backward Transfer (CBWT)



(b) True Backward Transfer (TBWT)

Figure 13: Comparison of CBWT and TBWT for Optimizer Type

Based on the results above, you can see there was much less forgetting when looking at the CBWT graph for the SGD optimizer compared to Adam and RMSProp. Adam and RMSProp drop dramatically, while SGD has a very gradual drop. For TBWT, we see the same results where SGD outperforms the other optimizers.

7. Results Summary

Overall, below are the best traits that were tested independently that do the best to decrease catastrophic forgetting in our model. Look below at the table to see the best parameters.

Table 6: Best Model Configuration and Corresponding Performance Metrics

Parameter	**Optimal Value**	**Performance (ACC / BWT)**
Loss Function	NLL	0.3325 / -0.5965
Dropout Rate	High Dropout (0.5)	0.3354 / -0.5524
Network Depth	2 Layers	0.3597 / -0.5663
Optimizer	SGD	0.7024 / -0.1180

References

1. Lopez-Paz, D. et al. *Gradient Episodic Memory for Continual Learning.* NeurIPS (2017).

2. Ororbia, A. et al. *Lifelong Neural Predictive Coding.* CoRR (2019).