Investigating the Role of Activation Functions in Loss of Plasticity of Deep Q-Networks in Continual Reinforcement Learning

Sajjad Alizadeh [*1] **Mobina Mosannafat** [*1] **Elaheh Toulabinejad** [*1]

Abstract

Reinforcement Learning has emerged as a prominent subfield of machine learning. The process of learning in the real world involves adapting to new data continually. Deep neural networks, by efficiently handling complex patterns and representations, empower the capability of reinforcement learning algorithms to more accurately approximate continuous processes, enhancing their ability to effectively solve continual problems in various domains. Plasticity in continual learning systems refers to the network's capacity to learn from and adapt to changes in incoming data. However, neural networks often suffer from loss of plasticity, especially in non-stationary environments, hindering their adaptability over time. This study aims to focus on mitigating plasticity loss in deep Q-Networks by exploring the role of activation functions. Specifically, we investigate two distinct activation functions: Concatenate ReLU and Leaky ReLU in the Dancing Catch environment. The findings of this study suggest that using CReLU and Leaky ReLU mitigates the loss of plasticity.

1. Introduction

Reinforcement Learning (RL) is a branch of machine learning. The idea of RL is to use an agent to interact with the environment to accomplish a goal. The incorporation of deep learning techniques into RL has led to the emergence of deep RL. RL can leverage deep neural networks (DNNs) to address continual problems. Traditional methods like Monte Carlo, Sarsa, and Q-learning (Sutton & Barto, 2018) are well-suited for episodic tasks. However, these methods might face limitations when dealing with continual problems that evolve dynamically over time. Real-world scenarios often involve continual and ever-changing environments, which leads to encountering numerous states. Also, These methods are designed for problems with discrete action spaces. Hence it is challenging for these traditional methods to adapt effectively to continual problems. On the other hand, The integration of DNNs into RL techniques addresses these challenges by allowing the modeling and approximation of complex problems. DNNs offer the capability to handle continual and high-dimensional state spaces, enabling RL algorithms to generalize better across states, learn complex mappings, and adapt to the changing nature of real-world problems. This integration enhances the capacity of RL methods to tackle complex and realistic problems that exist in continual environments, thereby expanding their applicability to a wider range of real-world scenarios. In the real world, we continuously learn from new data. This process is called continual learning (CL). In CL systems, plasticity refers to learning from new data and adapting to changes in that data (Dohare et al., 2023).

However, using neural networks to deal with these problems could be challenging because of the loss of plasticity. Loss of plasticity in neural networks occurs when they struggle to adapt to changing data patterns over time, ultimately hindering their learning capacity in non-stationary environments. Researchers have attempted to investigate and reduce plasticity loss, even though we still don't fully understand why it happens. Previous studies have grouped methods for dealing with plasticity loss into four main categories: resetting, regularization, architectural, and optimizer solutions (Kumar et al., 2023).

The primary objective of this research is to address the plasticity loss in deep Q-Networks (DQNs). Our focus is on understanding the impact of activation functions in mitigating this issue. In particular, we examine the effects of Concatenate ReLU (CReLU) and Leaky ReLU activation functions on loss of plasticity.

he subsequent sections of the paper follow this organization: In Section 2, we will cover the background and literature review. Section 3 presents our methodology. In Section 4, we discuss and evaluate our results. Section 5 concludes the paper by reviewing the main points. Finally, in Section 6, we outline the authors' contribution.

[*]Equal contribution [1]Department of Computing Science, University of Alberta, Edmonton, Canada. Correspondence to: Sajjad Alizadeh <salizad1@ualberta.ca>, Mobina Mosannafat <mosannaf@ualberta.ca>, Elaheh Toulabinejad <toulabin@ualberta.ca>.

2. Background

(Anthes et al., 2023) introduces the RDAC (Readout-Decomposition of Activation Change) framework, which explores the balance between stability and plasticity in CL algorithms. By analyzing how learning-induced activation changes relate to prior readouts and null space alterations, the framework sheds light on the trade-offs inherent in popular CL algorithms like Sunaptic Intelligence (which was introduced by (Zenke et al., 2017)), Elastic Weight Consolidation, Learning without Forgetting, and Gradient Episodic Memory (which proposed by (Lopez-Paz & Ranzato, 2017), that that alleviates forgetting while allowing beneficial transfer of knowledge to previous tasks), and data replay. It reveals that while some methods prioritize stability, others manage to maintain both stability and plasticity. Also, there are more research investigating CL algorithms exclusively. For example, (Kirkpatrick et al., 2017) challenges the typical limitations of neural networks in sequential task learning by proposing a method that mitigates catastrophic forgetting, forgetting previous tasks when learning new ones. It demonstrates the feasibility of training networks capable of retaining expertise in tasks not encountered for extended periods. Their approach involves selectively adjusting learning rates for crucial task-related weights, effectively preserving knowledge of prior tasks. Their method's effectiveness is validated through solving MNIST-based classification tasks and sequentially learning multiple Atari 2600 games, showcasing scalability and promising outcomes in retaining task-specific expertise. Also, there are other research on CL algorithm such as (Li & Hoiem, 2016), which uses only new task data to train the network while preserving the original capabilities which corresponds to Learning without Forgetting.

In their groundbreaking work, (Kumar et al., 2023) introduced L2 Init, a method designed to mitigate plasticity loss in neural networks confronting non-stationary data streams. By incorporating L2 regularization toward initial parameters in the loss function, L2 Init presents itself as a streamlined and promising alternative.

Also, (Kumar et al., 2023) introduces 'L2 Init,' an approach designed to address the challenge of preserving plasticity in neural networks when handling non-stationary data streams. The method incorporates L2 regularization towards initial parameters, reminiscent of standard L2 regularization but directing regularization towards the origin. By encouraging parameters to gravitate towards their initial values, 'L2 Init' aims to maintain neural network plasticity, facilitating rapid adaptation to new tasks. Empirical evaluations conducted across various nonstationary continual supervised learning scenarios demonstrate the consistent efficacy of 'L2 Init' in mitigating plasticity loss compared to previously proposed methodologies.

Moreover, (Abbas et al., 2023) looks at how well typical deep reinforcement learning methods adapt to changing environments, by playing different Atari 2600 games. It finds that these methods struggle to learn as effectively when the games keep changing. The study runs many tests for different durations and examines how the learning changes in the neural networks. They find that using a simple solution called CReLUs helps the learning process in these changing environments

Building on prior research, we decided to investigate the role of activation functions in the loss of plasticity of the continual problems. This is because specialized activation functions have the potential to effectively manage null space variations in case of input changes, thereby enhancing plasticity. Therefore, this project suggests that the adoption of distinct activation functions could mitigate the issue of plasticity loss in DQNs. In essence, the incorporation of other activation functions can empower DQNs to address an extended array of real-world, non-stationary challenges. Furthermore, mitigating plasticity loss helps to generalize the applicability of DQNs across diverse problem domains, such as traffic signal control and robotic applications.

3. Methodology

The first step to investigate the impact of the activation function of DQN on loss of plasticity in continual RL is picking an environment for experiments. While previous studies (Abbas et al., 2023; Li et al., 2021; Schaul et al., 2019) focused on the Atari environment, we have chosen to target the 'Dancing Catch' environment. We opted for the Dancing Catch environment over Atari because the agent requires more steps to learn the Atari game compared to Dancing Catch. Moreover, running multiple Atari games without altering the environment significantly prolongs the training time, potentially resulting in an exceedingly longer training duration. On the other hand, Dancing Catch is a modification of Catch that offers a simpler and more computationally efficient environment than traditional setups. In this game, a new ball appears randomly at the top of the screen in one of five columns, occurring with a 10% chance, and the agent has to catch this ball, just like in a traditional catch game. Multiple balls can coexist on the screen, but only one new ball is added per timestep. Existing balls descend by one pixel per timestep within their columns. The paddle can stay in place or move one pixel left or right per timestep. Catching a ball rewards +1, while a miss results in -1. This modification creates a dynamic, non-stationary challenge, prompting the agent to adapt its strategy continually. Despite maintaining the traditional 10-row, 5-column structure, the new setup demands real-time decision-making as the agent catches falling objects while adjusting to these intermittent changes.

To address the posed question, we conducted experiments within the Dancing Catch environment, employing a DQN agent and the Adam optimizer. The agent's performance was closely monitored and analyzed, focusing on the catch rate of the experiment. The catch rate is defined as the frequency of success divided by number of attempts, with success being attained when the agent receives a reward of +1 ($catch\ rate = \frac{1}{K}\sum_{k=1}^{K}[reward = 1]$).

With our environment and agent in place, our attention shifts towards investigating the influence of different activation functions in DQN on plasticity loss. Our objective is to assess and compare the efficacy of ReLU, Leaky ReLU, and CReLU in addressing the challenge of plasticity loss. Following is an overview of each activation function:

- ReLU is one of the most commonly used activation functions in DNNs. It sets all negative values in the input to zero and leaves positive values unchanged. The mathematical representation is $f(x) = max(0, x)$.
- Leaky ReLU is an extension of ReLU. Instead of setting negative values to zero, Leaky ReLU allows a small gradient for negative values, preventing neurons from completely getting deactivated. Mathematically, it's defined as $f(x) = max(\alpha x, x)$, where α is a small constant (e.g., 0.01) determining the slope of the negative part of the function.
- CReLU is an activation function that operates on pairs of consecutive elements in a vector. It concatenates the ReLU activation of a vector with its negated ReLU activation. Mathematically, for each element x_i in the input vector x CReLU outputs $[max(0, x_i), max(0, -x_i)]$. CReLU is known for its ability to capture both positive and negative features simultaneously, potentially enhancing the expressive power of neural networks.

Our DQN agent includes a component which is a two-layer network, consisting of a hidden layer and an output layer. The hidden layer size is set at 64 for ReLU and Leaky ReLU, and 32 for CReLU. The activation function is situated between the hidden layer and the output layer. The hidden layer size for CReLU differs from other activation functions due to the CReLU operation, which doubles the number of output channels compared to the input size. This operation applies the ReLU function to the input and its negation separately, effectively doubling the output size. Consequently, in terms of the number of output channels, the CReLU size is twice that of the ReLU size. To maintain uniform output sizes across all activation functions (ReLU, Leaky ReLU, and CReLU), we decided to halve the input size for CReLU. The network has an input size of 32×50 and an output size of 32×3. One of the parameters of the network is w which helps The DQN agent to consistently adjust a parametric estimation of future reward known as the value function which can be defined as:

$$\hat{q}(s, a; w) \approx \mathbb{E}[G_t | S_t = s, A_t = a]$$

In the above equation, G_t represents the return or cumulative reward from time step t onward. It is defined as the sum of discounted rewards obtained in the future which can be defined using γ, the discount factor, and R_{t+k+1}, the reward obtained at time step $t + k + 1$:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

In our experiments, we tracked specific metrics, including the L_0-Norm of the activation function, L_0-Norm, L_1-Norm, and L_2-Norm of the gradient of the hidden layer, L_2-Norm of weight changes in the hidden layer, and the loss. L_0-Norm, L_1-Norm and L_2-Norm could be defined as follow:

L_0-Norm:

$$\|x\|_0 = \text{number of non-zero elements in } x$$

L_1-Norm:

$$\|x\|_1 = \sum_i |x_i|$$

L_2-Norm:

$$\|x\|_2 = \sqrt{\sum_i x_i^2}$$

To understand better why one activation function might perform better than another, we calculated and plotted these metrics. The value of the other hyper-parameters of the implementation can be found in Table 1. The explanation of these parameters is as follows:

- Discount Factor (γ): Determines the extent of emphasis placed on future rewards in an agent's decision-making process, with higher values prioritizing long-term rewards
- BATCH SIZE: Number of experiences used for updating neural network parameters during training.
- BUFFER SIZE: Size of the replay memory
- MIN REPLAY SIZE: Threshold number of experiences required in the replay memory before starting the training process.
- MAX STEPS: Number of training process steps.
- TARGET UPDATE FREQ: Frequency of updating target network from online network

- HIDDEN LAYER SIZE: Input size of the hidden layer of the DQN.
- LEARNING RATE: Step size for adjusting model parameters during optimization to reach the optimal solution
- SWAP EVERY: Number of steps for the environment to change itself
- AVERAGE WINDOW SIZE: Number of consecutive iterations or episodes over which a moving average is calculated to smooth out fluctuations in performance metrics, such as the average return or loss, providing a clearer trend during training.

Table 1. Hyper-parameters

Name	Value
γ	0.9
BATCH SIZE	32
BUFFER SIZE	1000
MIN REPLAY SIZE	32
MAX STEPS	1000000
TARGET UPDATE FREQ	128
HIDDEN LAYER SIZE (ReLU, Leaky ReLU)	64
HIDDEN LAYER SIZE (CReLU)	32
LEARNING RATE	0.01
SWAP EVERY	10000
AVERAGE WINDOW SIZE	1000

Furthermore, each DQN was trained using ReLU, CReLU, and Leaky ReLU with five distinct random seeds to generate varied random numbers. Employing different random seeds aids in evaluating the model's robustness and adaptability. This approach enables us to analyze the model's performance across diverse random starts, offering valuable insights into algorithm stability, the influence of randomness, and the diversity of outcomes due to distinct initial conditions. [1]

4. Experimental Results

4.1. Demonstrating Loss of Plasticity in DQNs with ReLU Activation Function

Initially, our objective is to assess the efficacy of the DQN agent employing the ReLU activation function in engaging with the Dancing Catch game throughout two million steps and 500 data samples (we stored required data for every 4000 steps as we had memory limitations). Throughout this period, the environment undergoes alterations every 10,000 steps to maintain nonstationarity. To reduce the possibility of coincidental outcomes, we replicated this experiment in five seeds, thereby enhancing the robustness and reliability of the results. As illustrated in Figure 1, the agent's learning ability diminishes across all seeds as timesteps progress. To comprehend the reasons for this trend, we delve into various characteristics of the network.

First, we evaluate the alterations in weights of the hidden layer, as depicted in Figure 2(a). It is evident from the figure that the magnitude of weight changes diminishes. One possible explanation for the phenomenon of diminishing weight changes in a neural network is the convergence of the model during training. As the training progresses, the model gradually approaches a minimum of the loss function. In the later stages of training, the updates to the weights tend to become smaller as the model converges towards a solution. However, it does not make sense as we mentioned that the agent's learning ability reduces after each environment changes.

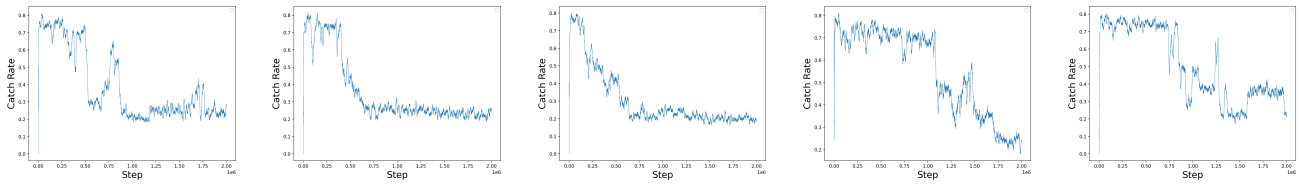
In our investigation, we conducted a thorough assessment of the loss values (Figure 2(b)) to gain insights into this behavior. Contrary to expectations, we observed that although the weights were changing less significantly, the associated loss values remained relatively large which showed that the network cannot keep learning.

Another contributing factor to the diminishing weight changes can be attributed to a reduction in the gradients of the hidden layer. This phenomenon occurs in regions of the parameter space where the gradient of the loss concerning the weights becomes notably small. As illustrated in Figure 2(c), our empirical observations indicate a decreasing trend in gradients throughout training. In such regions of reduced gradient magnitudes, the updates to the weights naturally become smaller.

In our final evaluation, we analyzed the activation function to figure out why weight changes and gradient reduced but the loss remained relatively large. As depicted in Figure 2(d), there appears to be a decrease in the normalized L_0-Norm of the activation function. Given that L_0-Norm functions as an indicator of non-zero elements in a vector—representing active units in our context—this observed reduction suggests a potential decline in the number of active units. Consequently, the diminishing count of the activation function's active units might imply a limitation in their contribution to the modification of weights within the neural network. Hence, this reduction in active units may contribute to, albeit uncertainly, the decrease in weight changes.

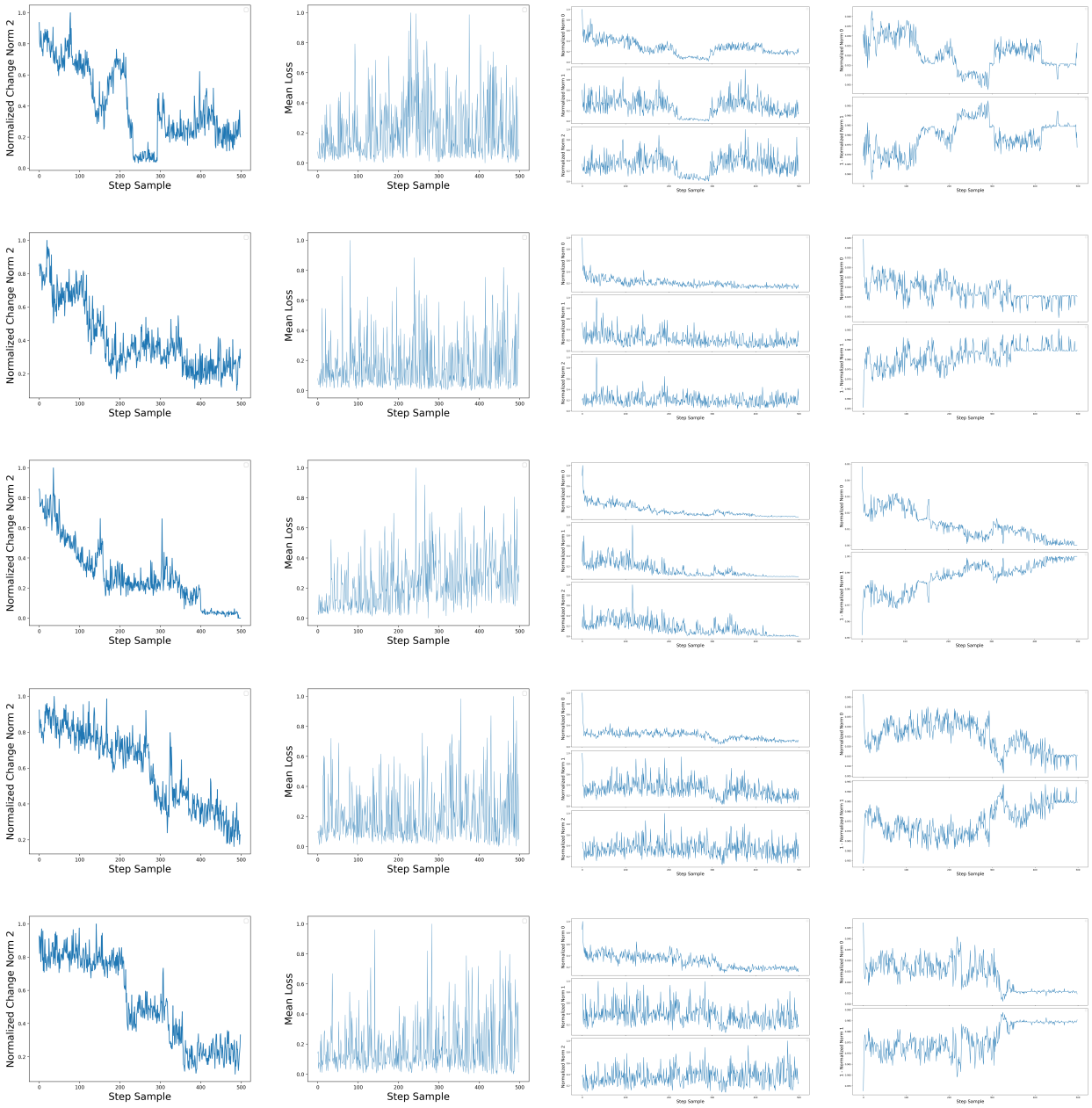
Hence, it appears that the reduction in the normalized L_0-Norm of the activation function is associated with a decrease in gradients. This decline in gradients, in turn, impedes

[1] Source code available at: https://github.com/sajializ/Loss-of-Plasticity



a. First seed b. Second seed c. Third seed d. Forth seed e. Fifth seed

Figure 1. Catch rate for five random seeds using ReLU activation function



a. Weight Change b. Loss Function c. Gradient d. Activation

Figure 2. Weight change, loss function, gradient, and activation metrics for five random seeds using ReLU activation function

weight changes, culminating in a lack of learning. As a result, we posit that the issue of plasticity loss might originate from the characteristics of the activation function. Consequently, we undertake an effort to modify the activation function, aiming to assess the validity of this hypothesis.

4.2. Mitigating Loss of Plasticity in DQNs with Leaky ReLU and CReLU Activation Functions

In this section, we assess the impact of replacing the ReLU activation function with CReLU and Leaky ReLU on the loss of plasticity. To maintain consistency in all experiments, we keep all parameters the same as in the previous trial, with the sole modification being the activation function. This adjustment ensures a fair and direct comparison, allowing us to isolate and assess the specific influence of the varied activation function on the observed outcomes. We enable the agent to interact with the environment using setups identical to those in the previous section. The only difference is that we substitute the ReLU activation function with both the CReLU and Leaky ReLU activation functions.

As illustrated in Figure 3 and Figure 5, the catch rate diagram for both the CReLU and Leaky ReLU activations shows no discernible decrease, and there is no observable reduction in the rate of learning. It seems that the loss of plasticity did not happen after replacing the activation function. Moreover, we observe that half of the activation function units in CReLU and about 99% of units in Leaky ReLU, even higher than CReLU, are still active while the remaining units are dead (See Figure 4(b)).

In the subsequent analysis, we explore the behavior of our gradients, anticipating an absence of reduction. As shown in Figure 6(a), the corresponding gradient behavior does not exhibit a decrease. Remarkably, upon examination of the plots in Figure 6(b), there is also no discernible reduction in weight changes. Moreover, when comparing the loss for CReLU and Leaky ReLU with ReLU, we observe a smaller magnitude in their loss. Moreover, comparing the loss of CReLu and Leaky ReLU also shows that Leaky ReLU's loss magnitude is lower than CReLU's, which makes sense as it has a higher number of activation function's active units.

Finally, We can observe the weight changes of networks with CReLU and LeakyReLU activation functions. As we know, the loss magnitudes are in a smaller range in comparison with ReLU, and the gradient of the hidden layer does not diminish. Therefore, as we expected, the weight changes did not shrink as networks can update them.

5. Conclusion

In order to harness the potential integration of DNNs and RL in CL problems, which represent a common type of many real-world challenges, addressing a fundamental challenge of DQNs is imperative—the issue of plasticity loss. To achieve this goal, we conducted a series of experiments utilizing a Dancing Catch environment and a DQN agent. These experiments aimed to investigate the underlying reasons behind the problem and explore potential solutions.

In our experimental approach, we observed various metrics associated with the DQN agent, including the output of the activation function, the gradient of the hidden layer, weight changes in the hidden layer, and alterations in loss under conditions of plasticity loss. The examination of the L_0-Norm of the ReLU activation fun ction reveals the deactivation of its units throughout the training period. We hypothesize that this deactivation is a contributing factor to the observed reduction in gradient norms, consequently leading to the shrinkage of weight changes. Such a failure in weight adjustments may potentially impede the learning process.

In the subsequent phase, we substituted the ReLU activation function with CReLU and Leaky ReLU, reevaluating the previously monitored metrics. The outcomes revealed that 50% of the units in the CReLU activation function and the majority of units in the Leaky ReLU activation function remain active throughout the training process. We posit that maybe this phenomenon leads to a narrower range of loss, particularly evident in the case of Leaky ReLU. Consequently, the DQN agent retains its learning capability without experiencing a significant decline. In conclusion, DQN agents employing CReLU and Leaky ReLU as their activation functions demonstrate the maintenance of plasticity throughout the learning process.

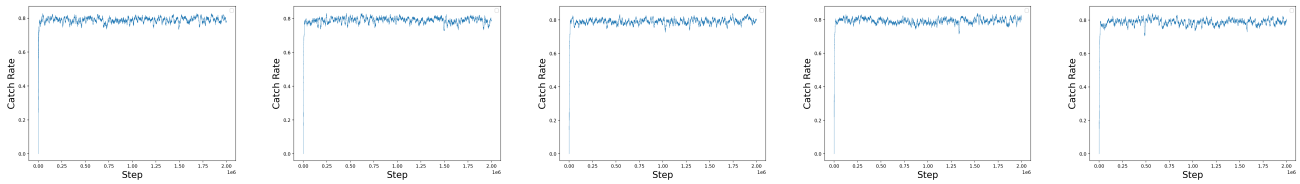
One constraint in our experiments stemmed from insufficient computational resources to store information at a high resolution per step. Consequently, we employed a subsampling method for storing and plotting results. Additionally, the limitations in computational resources and time constraints necessitated a restriction on the number of seeds we could run.

Future endeavors include replicating the experiment across a diverse set of environments and utilizing various types of problems instead of relying on a single problem. Furthermore, there is potential for improvement by delving into the influence of different hyperparameters and activation functions on the occurrence of plasticity loss and the mechanisms contributing to plasticity maintenance.

6. Contributions

Sajjad contributed to the project by aiding in code development, model implementation, result visualization, paper composition, and revision.

Mobina contributed to the project by contributing to idea

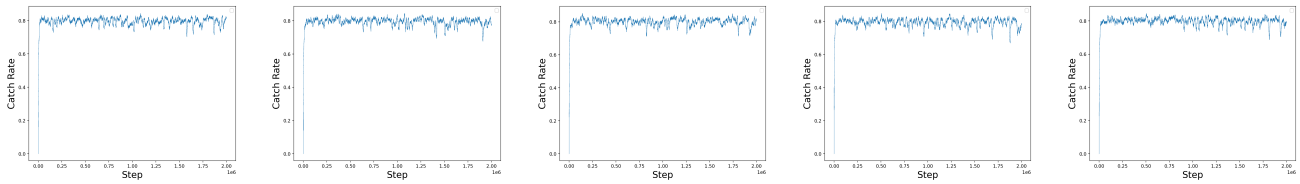


a. First seed b. Second seed c. Third seed d. Forth seed e. Fifth seed

Figure 3. Catch rate for five random seeds using CReLU activation function

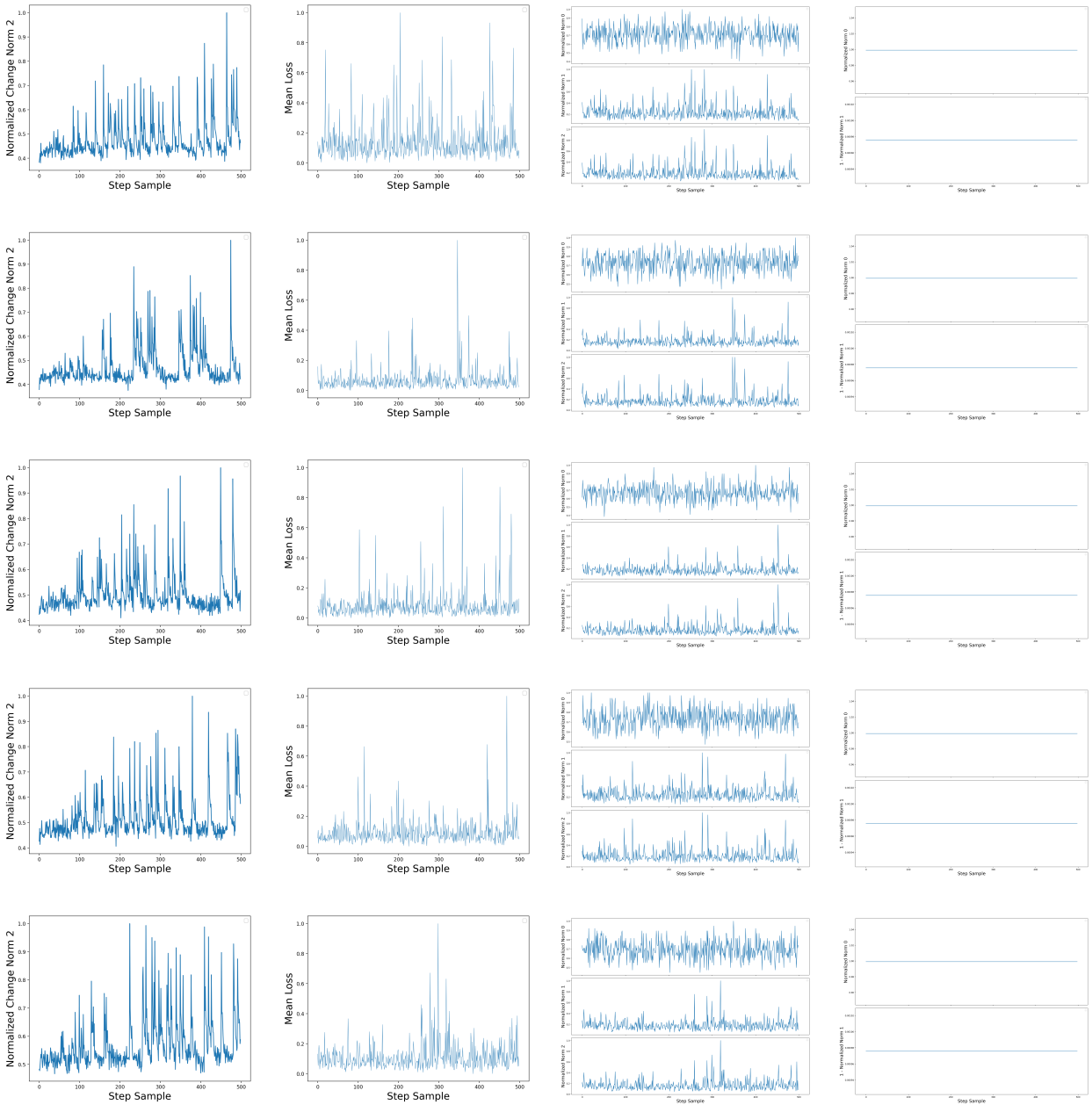
a. Weight Change b. Loss Function c. Gradient d. Activation

Figure 4. Weight change, loss function, gradient, and activation metrics for five random seeds using CReLU activation function



a. First seed b. Second seed c. Third seed d. Forth seed e. Fifth seed

Figure 5. Catch rate for five random seeds using Leaky ReLU activation function



a. Weight Change b. Loss Function c. Gradient d. Activation

Figure 6. Weight change, loss function, gradient, and activation metrics for five random seeds using Leaky ReLU activation function

development, participating in feedback discussions, coding, implementing models, visualizing results, composing the paper, and revising it.

Elaheh was responsible for reviewing related works, as well as contributing to the drafting of the proposal. She participated in proposal review and revisions. Moreover, she held discussions with professors and fellow students engaged in similar research to help make an informed choice regarding the research environment. She contributed to idea development, participating in feedback discussions, coding, implementing models, visualizing and interpreting results, crafting the paper, and revising it.

References

Abbas, Z., Zhao, R., Modayil, J., White, A., and Machado, M. C. Loss of Plasticity in Continual Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:2303.07507, March 2023. doi: 10.48550/arXiv.2303.07507.

Anthes, D., Thorat, S., König, P., and Kietzmann, T. C. Balancing Stability and Plasticity in Continual Learning: the Readout-Decomposition of Activation Change (RDAC) Framework. *arXiv e-prints*, art. arXiv:2310.04741, October 2023. doi: 10.48550/arXiv.2310.04741.

Dohare, S., Hernandez-Garcia, J. F., Rahman, P., Sutton, R. S., and Rupam Mahmood, A. Loss of Plasticity in Deep Continual Learning. *arXiv e-prints*, art. arXiv:2306.13812, June 2023. doi: 10.48550/arXiv.2306.13812.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Kumar, S., Marklund, H., and Van Roy, B. Maintaining Plasticity in Continual Learning via Regenerative Regularization. *arXiv e-prints*, art. arXiv:2308.11958, August 2023. doi: 10.48550/arXiv.2308.11958.

Li, S., Chi, H., and Xie, T. Multi-agent Combat in Non-Stationary Environments. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9534036.

Li, Z. and Hoiem, D. Learning without Forgetting. *CoRR*, abs/1606.09282, 2016. URL http://arxiv.org/abs/1606.09282.

Lopez-Paz, D. and Ranzato, M. Gradient Episodic Memory for Continual Learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6470–6479, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Schaul, T., Borsa, D., Ding, D., Szepesvari, D., Ostrovski, G., Dabney, W., and Osindero, S. Adapting Behaviour for Learning Progress. *CoRR*, abs/1912.06910, 2019. URL http://arxiv.org/abs/1912.06910.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Zenke, F., Poole, B., and Ganguli, S. Continual Learning Through Synaptic Intelligence. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/zenke17a.html.