Successive Weight Regularization for Plasticity in Deep Reinforcement Learning

Verona Teo*
UC Berkeley
veronateo@berkeley.edu

Arvind Rajaraman*
UC Berkeley
arvind.rajaraman@berkeley.edu

Seyone Chithrananda*
UC Berkeley
seyonec@berkeley.edu

1 Abstract

Continual learning has remained one of the essential and promising lines of research within deep reinforcement learning (RL). However, this remains a challenge for modern day methods, especially on non-stationary regression tasks. Neural networks have been shown to exhibit *plasticity loss* [2], such that it becomes difficult to adapt to new information over time. Plasticity loss tends to occur often in continual or non-stationary problems, when the relationship between inputs and prediction targets changes over time. Within reinforcement learning in particular, observations are significantly correlated over time and are influenced by the agent's policy, which is continuously changing [6]. Further, temporal difference (TD) learning also induces non-stationarity, such that the regression target, which depends on the target parameters, changes over time.

Although plasticity loss has gained attention in the deep RL literature, the causal mechanisms involved in plasticity loss, specifically in RL settings, are currently not well-understood. Some hypotheses for why neural networks experience plasticity loss include proxies such as inactive/dead activation units, e.g. ReLU, rank collapse of feature or weight matrices, and divergence due to large weight magnitudes over longer training iterations. However, recent literature such as show that many of these methods for explaining plasticity loss do not actually identify robust causal relationships [8]. Rather, they argue that plasticity loss arises due to other reasons, such as changes in the networks' loss landscape, which can be probed by either examining the Hessian of the network with respect to some loss function or the gradient covariance.

This paper explores the problem of plasticity loss in deep RL. In particular, we investigate the loss of plasticity in deep Q-networks and analyze whether and how the weights, rank, and activation neurons change over time in several experiments, as we are interested in empirically analyzing these metrics as explanations for plasticity loss. Hence, we examine different regularization methods (L_2/regenerative/singular penalties) [5][6], and propose a novel technique to control plasticity by varying the regularization strength across layers (successive regularization vs. constant/global regularization). The motivation behind our method is based on the idea that rather than fully resetting the last the layer from previous plasticity intervention methods for instance, we can apply a more gradual regularization, in which we increase the regularization in later layers of the network.

We also measure the effect of changing the frequency of introducing these penalties on the loss (every update vs. only during target update) on three different environments, LunarLander, CartPole, and MountainCar. We find that the weight magnitudes increase throughout the training of the network, and the activation footprint becomes sparser, contributing to the diminishing gradients. Further, our analysis shows that iteratively increasing the weight of any arbitrary regularization technique across layers consistently improves the models' evaluation performance and robustness on known markers of plasticity.

CS 285 Final Project Report.

2 Introduction

For a while now, it has been observed that humans and other animals lose the ability to quickly learn as they age. The phenomenon of neuroplasticity has been extensively studied by neuroscientists, and it occurs due to natural degradation of synaptic connections between neurons. Because neural networks are artificial, one would expect that neural networks can quickly adapt to new information, but surprisingly, a similar loss of ability to learn can be observed as training progresses. The mechanisms behind plasticity loss in neural networks are not well understood. A prior work cites primacy bias [11], which refers to a tendency to overfit initial experiences that damages the rest of the learning process, as a reason for decreasing plasticity over time.

Formally, plasticity loss in neural networks refers to the decreased ability of neural networks to change their predictions over time. This phenomenon is particularly a problem in value-based reinforcement learning (RL) agents, where a value critic network learns the values of states observed by the actor. This regression problem is non-stationary in nature since the value critic must keep up with the actors's changing policy, which results in the targets changing over time. Thus, plasticity partially contributes to the instability commonly seen in RL training [3, 11, 7].

This paper takes a step towards inducing plasticity in model training for deep RL. Weight decay and regularization, such as L_2 regularization, has shown promise in maintaining plasticity, since it pushes the weights away from extrema in the activation function, which make it difficult to update predictions quickly. While L_2 regularization reduces parameter magnitudes which helps control plasticity loss, regularizing toward the origin across all layers is likely to collapse the ranks of the weight matrices as well as lead to so-called mutually frozen weights [15].

We propose exploding weight regularization, which successively increases the penalty for large weights in later layers of the network. Our hypothesis with this intervention is that regularizing later layer weights for each update is likely to be better than earlier layer weights, and gradually increasing the level of regularization and normalization over time might be beneficial and mitigate plasticity loss. Further, we hypothesize that regularizing only on the target update is better than regularizing at every step.

Prior work has evaluated the plasticity of neural networks through metrics like weight magnitude, [5], number of dead neurons [10], maximal eigenvalue λ_{max} [1], and condition number $\frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$ [4]. We similarly evaluate our methods and baselines using these metrics both across layers and per layer. We show that successive regularization and regularizing at every step has performance and plasticity benefits.

3 Background

We introduce the standard RL problem where we a Markov Decision Process (MDP) described by an observation space $\mathcal{O}$, action space $\mathcal{A}$, reward function $R : (\mathcal{O}, \mathcal{A}, \mathcal{O}) \to \mathbb{R}$, and discount factor γ. We assume access to an environment $\mathcal{M}$ that models the dynamics $\mathbb{P}[s' \mid s, a]$. The goal of RL is to learn an optimal policy $\pi^\star : \mathcal{O} \to \mathcal{A}$ that maximizes expected reward. To learn this policy, some RL methods choose to learn Q-values of state-action pairs.

In Q-learning, the Q-function or state-action value function of a policy π, $Q^\pi(s, a)$, measures the expected return obtained from state s after taking action a and following policy π thereafter [14]. The optimal Q-function $Q^*(s, a)$, the maximum return that can be obtained from observation s, taking action a, then following the optimal policy is given by $Q^*(s, a) = \mathbb{E}[r(s, a) + \gamma \cdot \max_{a'} Q^*(s', x')]$, where r is the reward and γ is a discount factor. Q-learning uses the Bellman optimality equation as an iterative update. However, large state and/or action spaces makes it intractable to learn q-values for each state-action pair independently. Hence, we use deep Q-learning estimate the Q-values, such as by training a neural network.

The Deep Q-networks algorithm (DQN) combines the Q-learning algorithm with deep neural networks, specifically a convolutional neural network, and also incorporates experience replay [9]. With experience replay, at each time step of data collection, the transitions are added to a replay buffer. During training, mini-batches of transitions are sampled from the replay buffer and used to compute the loss and its gradient. This allows for better data efficiency by reusing each transition in multiple updates, and is more stable. DQN approximates the action values for a given state s_t. At each time

step, the agent selects an action ϵ-greedily with respect to the action values, and adds a transition to a replay buffer. The neural network has parameters $\theta_t \in \Theta$ (at timestep t), which are optimized using stochastic gradient descent to minimize the loss $\mathcal{L}(\theta_t) = r_{t+1} + \gamma \cdot \max_{a'} q_{\theta'_t}(s_{t+1}, a') - q_{\theta_t}(s_t, a_t))^2$, where t is a time step selected randomly from the replay buffer and θ'_t denotes the parameters of the target network, a copy of the online network that is not optimized directly. This network is also known as a value critic $\hat{V}_{\theta_t} : \mathcal{O} \to \mathbb{R}^{\|\mathcal{A}\|}$. The subscript is there to denote that this critic is parameterized by the parameters θ_t. Note that the gradient of the loss is backpropagated into the parameters θ of the online network.

4 Methodology

4.1 Metrics

There are several hypotheses for why networks lose plasticity. In this paper, we use four main proxies to measure plasticity: weight magnitude, effective rank, and number of dead units.

Weight magnitude. Previous studies [7][5] have studied increasing weight magnitudes of value-critic methods as an early indicator of plasticity. One reason may be that using the Adam optimizer makes it difficult to update weights with large magnitude since updates are bounded by the step size. [3].

Effective rank. Often, with the loss of plasticity is the drop in the effective rank of the representation. We refer the output of the penultimate layer of the deep Q-network as the learned representation (feature matrix Φ), similar to [5]. Similar to the rank of a matrix, which represents the number of linearly independent dimensions, the effective rank of Φ takes into consideration how each dimension influences the transformation induced by a matrix. A high effective rank signals that most of the dimensions of the matrix contribute equally to the transformation induced by the matrix. On the other hand, a low effective rank implies that the information in most of the dimensions is close to being redundant. The effective rank of the feature matrix Φ is defined as the following, for singular values $\sigma = \sigma_1, ..., \sigma_q$, $p_k = \sigma_k / ||\sigma||_1$ [2]:

$$\begin{aligned} \text{effective\_rank}(\Phi) &= \exp(H(p_1, ..., p_q)) \\ &= -\sum_{k=1}^{q} p_k \, log(p_k) \end{aligned}$$

Dead units. If a ReLU is not active, i.e. zero output value, then it does not contribute to the change in its incoming weights due to the chain rule. Hence, this is an indication of plasticity loss because weights are not changing, irrespective of a large global loss, so the network has lost its ability to learn new things. To measure the number of dead units in a network with ReLU activation, we count the number of units with a value of zero for all state-action pairs in a minibatch. The number of dead units is then averaged by the number of units in the layer.

4.2 Regularization

Prior work suggests that resetting the last layer is a powerful way to mitigate plasticity loss [11]. This is because earlier layers in the value critic network learn low-level features that are applicable for any target, whereas later layers learn a mapping from these features to the specific targets at hand.

However, resetting the last layer's weights completely is not ideal, as the targets are unlikely to have changed drastically. Weight regularization can thus be interpreted as a "softer" version of weight resetting, since the weight magnitudes are nudged towards zero. The degree that the weights are nudged is controllable by a hyperparameter λ, which is more desirable than resetting all of the last layer's weights. Thus, we proceed with using weight regularization as our method of inducing plasticity.

Through our experiments, we explore two different types of weight regularization. The first is weight magnitude regularization, which simply penalizes the L_2-norm of the weights, while the latter penalty aims to correct rank collapse in learned feature matrices of the value critic. Under the weight magnitude scheme, the value critic's loss function for the weights θ_t at timestep t would be:

$$\mathcal{L}(\theta_t) = \|\hat{V}_{\theta_t}(s) - V(s)\|_2^2 + \lambda\|\theta_t\|_2^2$$

Another weight magnitude regularization scheme proposed by prior work that we test is regenerative regularization [6]. Given neural network parameters θ, regenerative regularization (denoted as L2 Init in the original work) augments a standard training loss function $\mathcal{L}_{train(\theta)}$ with a regularization term. Specifically, L2 Init performs L_2 regularization toward initial parameter values θ_0 at every time step for which a gradient update occurs. The augmented loss function is denoted as the following:

$$\mathcal{L}(\theta_t) = \|\hat{V}_{\theta_t}(s) - V(s)\|_2^2 + \lambda||\theta_t - \theta_0||_2^2$$

In addition to penalties involving model weights, we also explore the use of a secondary penalty aiming to mitigate implicit under-parameterization by controlling rank collapse from prior work [5]. The penalty aims to reduce the tendency of value functions trained using iterated regression onto target values to decrease in expressivity over more and more gradient updates. Loss of expressivity is characterized as a drop in the effective rank of learned value network features, leading to a later drop in performance. To encourage higher effective rank of the learned features, since effective rank is non-differentiable, the authors propose a surrogate involving the singular values of the feature matrix over a minibatch (computed using SVD). By taking the difference between the squared values of the maximum and minimum singular values, the value critic will learn to both minimize the largest singular value and maximize the smallest.

$$\mathcal{L}(\theta_t) = \|\hat{V}_{\theta_t}(s) - V(s)\|_2^2 + \lambda(\sigma_{max}^2(\Phi) - \sigma_{min}^2(\Phi))$$

4.3 Global vs. Successive Regularization

We explore the idea of a successive, layer-dependent regularization strategy using the three loss penalties above. Motivated by prior work exploring the impact of layer resetting and normalization on plasticity [7], we multiply the regularization penalty λ of the loss penalty to be layer dependent. Earlier layers, which often learn low-level features applicable for any target, are nudged far less by the penalty. On the other hand, latter layers of the value-critic responsible for mapping from these features to specific target are more heavily penalized for high weight magnitudes, high divergence from the initialized weights, or a high divergence in feature singular values, respectively.

To abstract away the specific regularization scheme we use (i.e. weight magnitude, regenerative, or singular value range regularization), we will assume a general regularization function $\mathcal{R} : \theta \to \mathbb{R}$. Also, we assume that the parameters can be broken into the parameters for each layer for our L-layer value critic: $\theta_t = [\theta_t^{[1]}\theta_t^{[2]} \dots \theta_t^{[L]}]^\top$. Last, we define parameter δ^l as the layer-specific regularization weight for layers $l = 1, ..., L$. Then, we can express the successive regularization loss concisely as follows:

$$\mathcal{L}(\theta) = \|\hat{V}_{\theta_t}(s) - V(s)\|_2^2 + \lambda \sum_{l=1}^{L} \delta^l \mathcal{R}(\theta_t^{[l]})$$

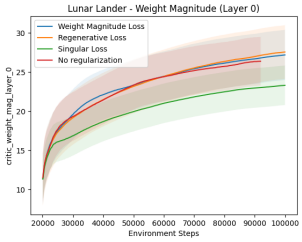
4.4 Frequency of Regularization

We additionally study the difference between introducing the successive weight decay strategy either only during the target critic update or during each update, for both the constant vs successive regularization strategies. In doing so, we aim to see whether introducing any of the three penalties only during major updates to the target critic or not greatly modifies performance on plasticity metrics. We define parameter τ that describes the interval for updating the target value critic.

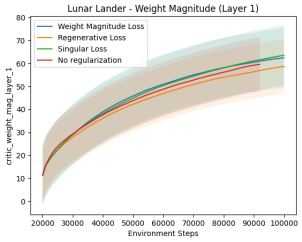
Combining all of the modifications made in this section, we have our final loss:

$$\mathcal{L}(\theta_t) = \begin{cases} \|\hat{V}_{\theta_t}(s) - V(s)\|_2^2 + \lambda \sum_{l=1}^{L} \delta^l \mathcal{R}(\theta_t^{[l]}) & \text{if } t \pmod{\tau} = 0 \\ \|\hat{V}_{\theta_t}(s) - V(s)\|_2^2 & \text{else} \end{cases}$$

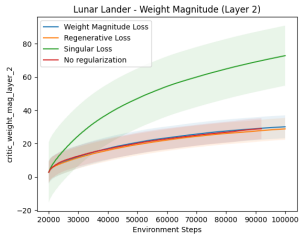
$$r(\theta_t) = \begin{cases} 0 & \text{if none} \\ \|\theta_t^{[l]}\|_2^2 & \text{if weight magnitude} \\ \|\theta_t^{[l]} - \theta_0^{[l]}\|_2^2 & \text{if regenerative regularization} \\ \sigma_{max}^2 - \sigma_{min}^2 & \text{if singular loss} \end{cases}$$



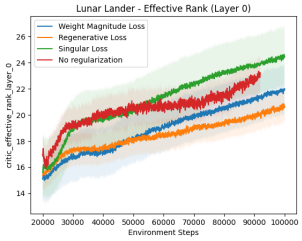
(a) Weight Mag. (l=1)



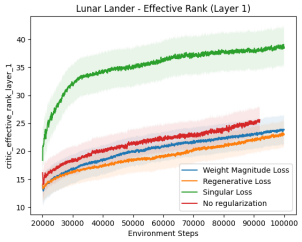
(b) Weight Mag. (l=2)



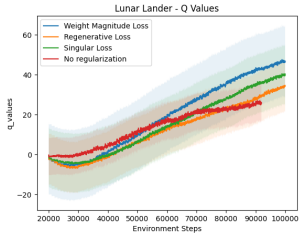
(c) Weight Mag. (l=3)



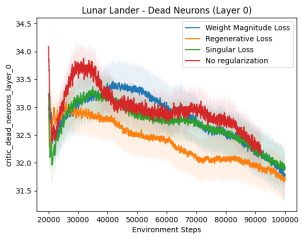
(a) Eff. Rank (l=1)



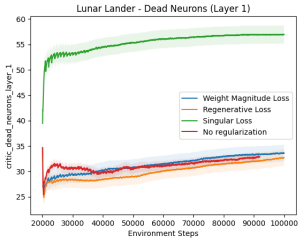
(b) Eff. Rank (l=2)



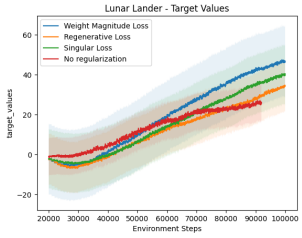
(c) Q Values (l=3)



(a) Dead Neurons (l=1)



(b) Dead Neurons (l=2)



(c) Target Values

Figure 1: Shown are the plasticity metrics and training dynamics (i.e. Q-values and target values) for the LunarLander environment. Our regularization techniques decrease the weight magnitude in later layers. We see that effective rank is increasing over time, which is evidence of higher plasticity (since rank collapse becomes less likely); our singular value formulation has the highest effective rank. Paradoxically though, singular loss increased the amount of dead neurons in the middle layer. Results are averaged over 2 seeds.

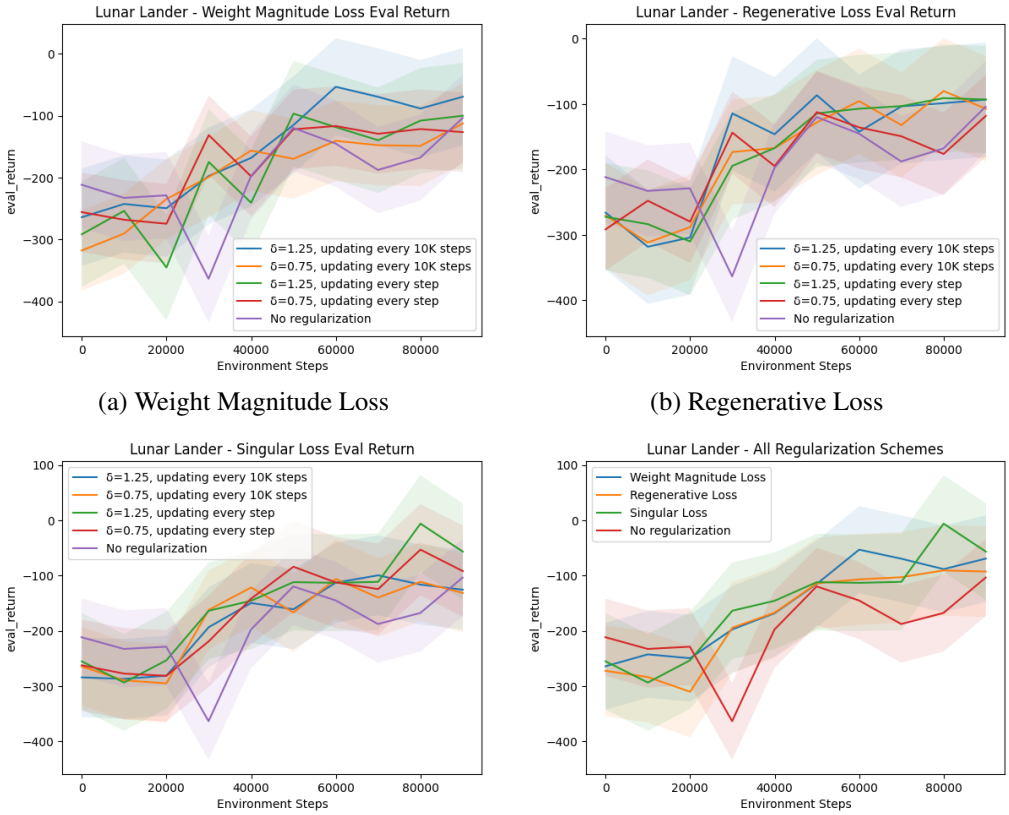
5 Experiments

The goal of our experiments is to determine the effects of our method on plasticity loss in continual learning. In particular, we evaluate the effects of varying levels of regularization across layers on the environments mentioned in Section 3.1.

5.1 Training and Hyperparameters

We perform experiments in three discrete settings: Lunar Lander, Cart Pole, and Mountain Car, and train a simple DQN agent using a series of MLP layers. We train for $100,000$ steps in each of the environments. We used two hidden layers of width 64 and ReLU activations, with an output layer mapping `(64, n_actions)`. We train each agent with the Adam optimizer with a constant decaying learning rate of $\alpha = 0.001$ with a factor of 1. For Lunar Lander, the first environment, we sweep over

regularization strength, i.e. the regularization penalty, $\lambda = 0.01$ and the exponential/successive factor $\delta \in \{0.75, 1, 1.25\}$. For the singular value loss, we used $\lambda = 0.001$. For the environments Cart Pole and Mountain Car, we used the regenerative regularizer and $\lambda = 0.01$ and $\delta = 1.25$.



(a) Weight Magnitude Loss

(b) Regenerative Loss

(c) Singular Loss

(d) All Losses

Figure 2: Above is our hyperparameter search over δ and how often to add the regularization penalty. We see that $\delta = 1.25$ performs better than $\delta = 0.75$ consistently, confirming that later layers should be regularized more. In (d), we select the highest performing configuration from each of (a), (b), (c), along with the No Regularization baseline. Comparing all the regularization schemes, singular loss has the highest return at eval time. Each configuration is tested over 2 seeds.

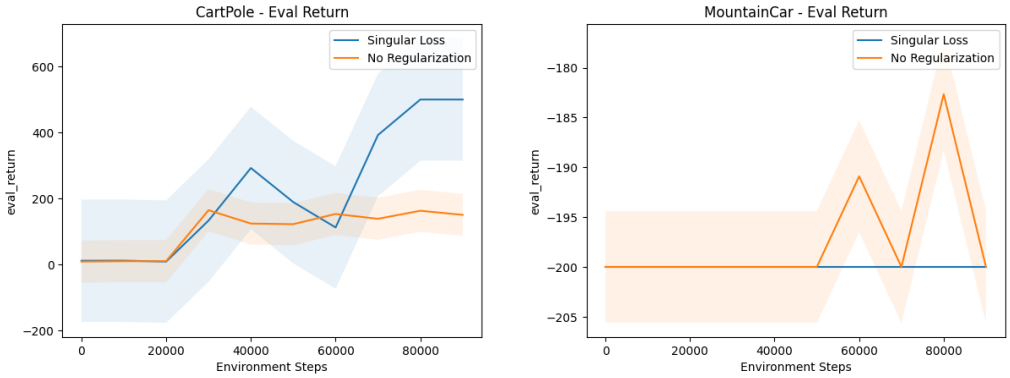
5.2 Evaluation

On each problem, we perform a hyperparameter sweep for each method and average results over 2 seeds. Each seed determines the initialization of the neural network. As a baseline method, we run a basic DQN with no regularization.

For the first environment, Lunar Lander, we determine the optimal values for the hyperparameters based on the maximum evaluation return, and apply those values to the other two environments, Cart Pole and Mountain Car. We smooth evaluation result curves across all environments using a simple moving average.

6 Conclusion

Our investigation and results demonstrate that deep RL algorithms often lose plasticity over time, i.e. they learn their abilities to learn. We evaluated plasticity loss through several metrics including weight magnitude, dead units, and effective rank. We applied a regularization schema that involves successive regularization/weight decay. We found that the singular loss performed the best, with the largest effective rank and evaluation return, despite having a high weight magnitude. Hence, singular



(a) CartPole (b) MountainCar

Figure 3: The eval return curves in the CartPole and MountainCar are shown for our singular loss regularization and the baseline. We see that our method strongly beats the baseline in the former environment, while it performs worse in the latter. The reason for our decreased performance in MountainCar is likely because we didn't train long enough, as evidenced by the decreasing Q-values and target values (see the appendix). Each configuration is tested over 2 seeds.

loss seems to mitigate plasticity loss. Further, using a successive regularizer was shown to perform well for all regularization methods and increase network plasticity. A third insight is that contrary our hypothesis on how often to regularize, regularizing on every step led to better performance than regularizing only on the target updates.

We hope our method opens up avenues for future work on mitigating plasticity loss. There are several promising directions for future work. We hope to conduct a more extensive hyperparameter search on different values of λ and δ and train for a larger number of steps. A slight modification to our method that could be interesting to explore is using L_1 regularization (instead of L_2 regularization). Further, we aim to extend our analysis from discrete to continuous states with methods such as soft-actor critics (SAC), trying under different algorithms like Proximal Policy Optimization (PPO) [13] and Trust Region Policy Optimization (TRPO) [12], as well as investigating experiment settings with more advanced architectures. Trying other losses that penalize eigenvalues could also provide additional insights on the mechanisms behind plasticity. In addition to regularization, we hypothesize that incorporating other architectural changes such as different activations and cross-connecting layers can improve plasticity, and hence believe that combining several of these interventions can provide benefits for plasticity. We also believe that it would be useful to evaluate our method on a broader set of problems, even outside of RL.

7 Acknowledgements

We want to thank Professor Sergey Levine for his highly useful insights about reinforcement learning that guided our project. We also wanted to thank the Fall 2023 course staff for Berkeley's CS 285 (Deep Reinforcement Learning), which includes Kyle Stachowicz, Vivek Myers, Joey Hong, and Kevin Black. We thank Machine Learning at Berkeley (student organization), for providing computing resources for training.

8 Contributions

Each member contributed equally to this project. We discussed and iterated on ideas for our method, experimental set-ups, and initial hyperparameters. We pair-programmed together every session, and each contributed to writing the report. For instance, Arvind focused more on the regularization methods, generating the plots, and the figures in the report. Verona and Seyone worked to implement the different metrics like weight magnitude, effective rank, and dead units, and worked on many parts of the report, e.g. abstract, introduction, background, methodology, experiments, and conclusion. We all analyzed the results together.

References

[1] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. 2017.

[2] Shibhansh Dohare, J. Fernando Hernandez-Garcia, Parash Rahman, Richard S. Sutton, and A. Rupam Mahmood. Loss of plasticity in deep continual learning, 2023.

[3] Shibhansh Dohare, Richard S Sutton, and A Rupam Mahmood. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

[4] Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instability in deep learning. *International Conference on Learning Representations (ICLR)*, 2021.

[5] Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2020.

[6] Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining plasticity via regenerative regularization. *arXiv preprint arXiv:2308.11958*, 2023.

[7] Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2022.

[8] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks, 2023.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International conference on machine learning (ICML)*, 2010.

[11] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. 2022.

[12] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

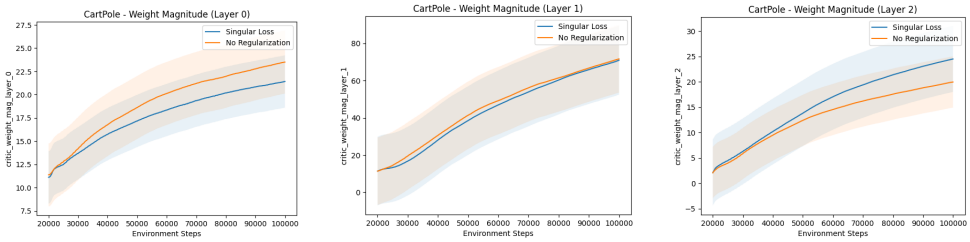
[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[14] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

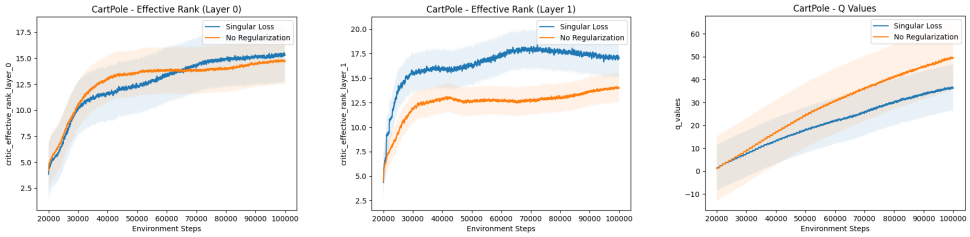
[15] Julian Zilly, Alessandro Achille, Andrea Censi, and Emilio Frazzoli. On plasticity, invariance, and mutually frozen weights in sequential task learning. *Advances in Neural Information Processing Systems*, 34:12386–12399, 2021.

9 Appendix

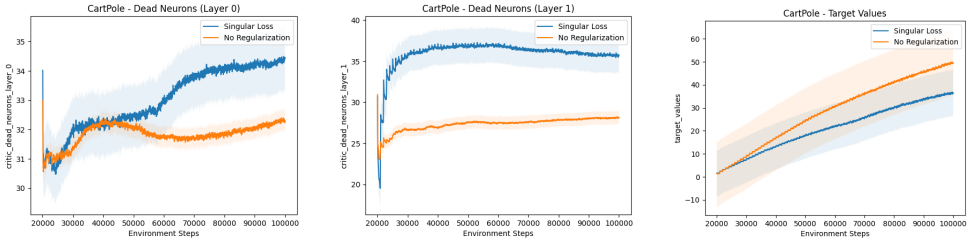
We include our plasticity metrics and training dynamics for the CartPole and MountainCar environments below.



(a) Weight Mag. (l=1) (b) Weight Mag. (l=2) (c) Weight Mag. (l=3)

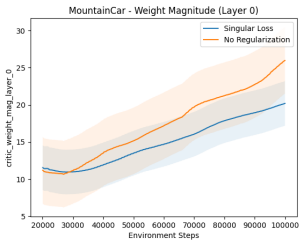


(a) Eff. Rank (l=1) (b) Eff. Rank (l=2) (c) Q Values (l=3)

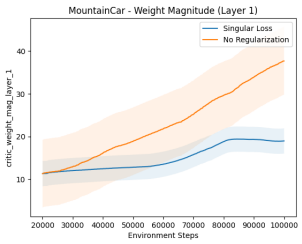


(a) Dead Neurons (l=1) (b) Dead Neurons (l=2) (c) Target Values

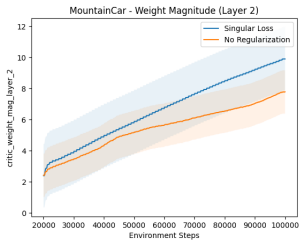
Figure 4: Shown are the plasticity metrics and training dynamics (i.e. Q-values and target values) for the CartPole environment. We see that singular regularization exhibits strong plasticity injection.



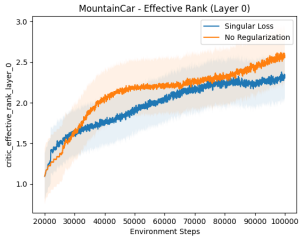
(a) Weight Mag. (l=1)



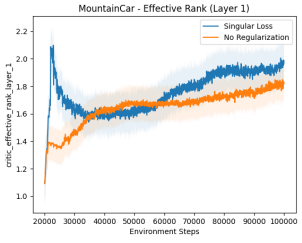
(b) Weight Mag. (l=2)



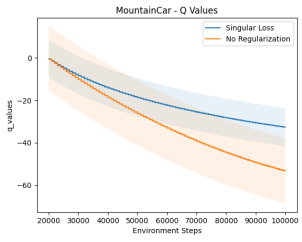
(c) Weight Mag. (l=3)



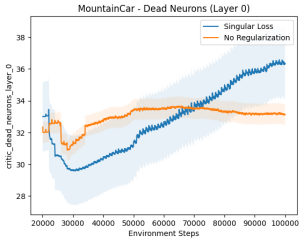
(a) Eff. Rank (l=1)



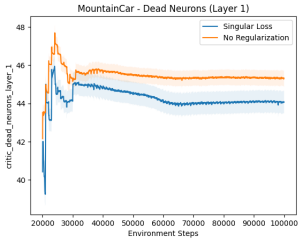
(b) Eff. Rank (l=2)



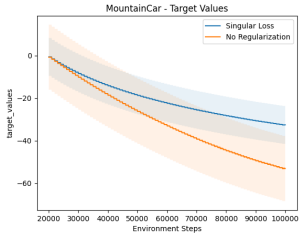
(c) Q Values (l=3)



(a) Dead Neurons (l=1)



(b) Dead Neurons (l=2)



(c) Target Values

Figure 5: Shown are the plasticity metrics and training dynamics (i.e. Q-values and target values) for the MountainCar environment. The stagnant eval return under our regularization scheme is explained by the decreasing Q-values and target values, which means we didn't train long enough. Given more time, we would have trained for longer.