Weight Clipping for Deep Continual and Reinforcement Learning

Mohamed Elsayed
University of Alberta
mohamedelsayed@ualberta.ca

Qingfeng Lan
University of Alberta
qlan3@ualberta.ca

Clare Lyle
Google DeepMind
clarelyle@google.com

A. Rupam Mahmood
University of Alberta
CIFAR AI Chair
armahmood@ualberta.ca

Abstract

Many failures in deep continual and reinforcement learning are associated with increasing magnitudes of the weights, making them hard to change and potentially causing overfitting. While many methods address these learning failures, they often change the optimizer or the architecture, a complexity that hinders widespread adoption in various systems. In this paper, we focus on learning failures that are associated with increasing weight norm and we propose a simple technique that can be easily added on top of existing learning systems: clipping neural network weights to limit them to a specific range. We study the effectiveness of weight clipping in a series of supervised and reinforcement learning experiments. Our empirical results highlight the benefits of weight clipping for generalization, addressing loss of plasticity and policy collapse, and facilitating learning with a large replay ratio. [1]

1 Introduction

Deep learning and reinforcement learning methods face many challenges when learning online or continually. These challenges include loss of plasticity (Lyle et al. 2023, Dohare et al. 2023a), failure to achieve further improvement (e.g., Sokar et al. 2023, Lyle et al. 2023, 2021), gradual performance decreases (e.g., Dohare et al. 2023a, Abbas et al. 2023, Elsayed & Mahmood 2024), and even loss of generalization (Ash & Adams 2019). One common feature among many instances of learning failures is their association with increasing weight magnitudes. An unbounded weight growth can make it increasingly harder for the learners to adjust the weight further (Lyle et al. 2024), causing loss of plasticity (Dohare et al. 2023a) or policy collapse (Dohare et al. 2023b). Moreover, large weight magnitudes can be harmful to the optimization dynamics (see Lyle et al. 2023 and Wortsman et al. 2023) and are often associated with overfitting (Zhang et al. 2021), leading to performance decrease and potentially explaining the learning difficulties.

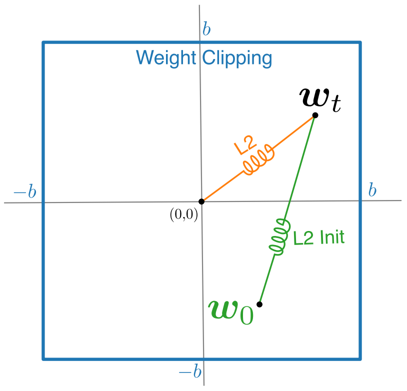


Figure 1: Weight clipping confines the weights in restricted space while *L2 Init* pulls the current weight $\boldsymbol{w}_t$ to the weight at initialization $\boldsymbol{w}_0$ and *L2* pulls the current weight $\boldsymbol{w}_t$ to the zero vector.

While many methods exist to address these learning difficulties, these methods are predominantly complex or require significant changes to the learning systems. For example, several methods require a change of the optimization method (Dohare et al. 2023a, Elsayed & Mahmood 2024, Sokar et al. 2023), use an auxiliary objective (Lan et al. 2023), change the architecture (e.g., Lyle et al. 2023, Nikishin et al. 2023, Lan & Mahmood 2023), or even require a new reinforcement learning estimator that maintains exploration (Garg et al. 2022). There are simpler techniques of weight regularization that directly promote small

[1] Code is available at `https://github.com/mohmdelsayed/weight-clipping`

weight magnitudes, such as weight decay (*L2*) (Krogh & Hertz 1991, Ash & Adams 2019, Lyle et al. 2024) and more recently, regularization toward initial weights (*L2 Init*) (Kumar et al. 2023a). In Fig. 1, we illustrate the difference between L2 and L2 Init, showing that L2 results in biasing the weights towards the zero weights while L2 Init results in biasing the weights towards the initial weights. The issue of biasing toward a particular point is that it applies to all weights regardless of their usefulness, resulting in overwriting and re-learning of previously useful weights, a primary cause of catastrophic forgetting (McCloskey & Cohen 1989, French 1999, Elsayed & Mahmood 2024). Lewandowski et al. (2024) proposed to use the empirical Wasserstein distance instead of L2 to alleviate the issue of L2 Init. However, the Wasserstein distance requires sorting the parameters, which is computationally expensive. It is desirable to address the issue of increasing weight magnitude in a computationally cheap manner without biasing the weight vector towards any point, allowing the learner to continually build on previously useful weights and overcome continual learning difficulties.

We use a simple remedy that can reduce the norm of the weights by clipping any large weight magnitude. This technique is distinct from gradient or update clipping (e.g., Brock et al. 2021, Badia et al. 2020), which aims to clip large updates or gradients, not weights. Constraining the weights prevents large weight magnitudes without biasing them toward a certain point, as depicted in Fig. 1. In this paper, we study the role of weight clipping in 1) improving generalization, 2) addressing loss of plasticity and policy collapse, and 3) facilitating learning with a large replay ratio.

2 Problem Formulation

In this section, we describe the two problem formulations we use in this paper and what metrics we use to evaluate learners in each of them.

2.1 Streaming Supervised Learning

In streaming supervised learning, the data samples are presented to the learner as they come, one sample at each time step. Each sample is processed *once* by the agent, then the learner is evaluated based on some evaluation metric, and after that, the sample is discarded immediately (Hayes et al. 2019). This setup mirrors animal learning (Hayes & Kanan 2022) and is important for various applications such as on-device learning. The target function f_t generating these data samples is typically non-stationary, producing non-independently and identically distributed (non-i.i.d.) samples such that $\boldsymbol{y}_t = f_t(\boldsymbol{x}_t)$. For simplicity, we assume that the target function is locally stationary in time, not arbitrarily non-stationary, but instead changes frequently (e.g., due to task change). The learner is expected to process the input $\boldsymbol{x}_t \in \mathbb{R}^d$ and produce a prediction $\hat{\boldsymbol{y}}_t \in \mathbb{R}^n$, after which it is evaluated based on the metric $E(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$. The goal of the learner is to maximize the average online metric (see Kumar et al. 2023b, Elsayed & Mahmood 2024) given by $\frac{1}{T}\sum_{t=1}^{T} E(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$, where T is the total number of time steps.

2.2 Reinforcement Learning

The sequential decision process of the agent and the interaction with the environment is modeled as a Markov decision process (MDP). In this paper, we consider episodic interactions between the agent and the environment in which its episodic MDP is denoted by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, d_0, \mathcal{H})$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{R} \subset \mathbb{R}$ denotes the set of reward signals, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R})$ denotes the transition dynamics model in which $\Delta(X)$ is a distribution over the set X, d_0 is the starting state distribution, $\gamma \in [0, 1]$ is the discount factor, and $\mathcal{H}$ is the set of terminal states. The agent interacts with the environments using a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that outputs a distribution over actions conditioned on the state (Sutton & Barto 2018). Each episode of interaction begins after the environment samples a state from the starting state distribution $S_0 \sim d_0$. At each time step t, the policy receives the state S_t and produces the action $A_t \sim \pi(.|S_t)$, and then the environment samples a new state and reward signal using the transition dynamics as follows: $S_{t+1}, R_{t+1} \sim p(., .|S_t, A_t)$. The interaction continues until the agent ends up in one of the terminal states $S_T \sim \mathcal{H}$, where T is the termination time step. The amount of the discounted sum of rewards collected by the agent during the episode at time step t is known as the episodic return and is given by $G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$. The goal of the agent is to maximize the expected return $\mathbb{E}_\pi[G_t]$ produced by following the policy π.

3 Method

In this section, we introduce weight clipping and show how it can be used with existing optimization methods. We propose a clipping scheme that uses the boundaries given by a uniform distribution at initialization (e.g., He et al. 2015). Consider a neural network f parameterized by the set of weights $\mathcal{W} = \{\boldsymbol{W}_1, \boldsymbol{W}_2, \dots, \boldsymbol{W}_L\}$ and the set of biases $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_L\}$. Specifically, given that the entries of the weight matrix $\boldsymbol{W}_l, \forall l$ and the bias vector $\boldsymbol{b}_l, \forall l$ are initialized from the uniform distribution $U[-s_l, s_l], s_l \in \mathbb{R}^+, \forall l$, we propose clipping any value outside the range $[-\kappa s_l, \kappa s_l]$, where $\kappa \in \mathbb{R}^+$ is a hyper-parameter defining the size of the constraint weight space. Algorithm 1 shows how weight clipping can be integrated into optimization methods.

Algorithm 1 Weight-constrained Stochastic Optimization

```
Given a stream of data $\mathcal{D}$, a neural network $f$ with weights $\{\boldsymbol{W}_1, ..., \boldsymbol{W}_L\}$ and biases $\{\boldsymbol{b}_1, \dots, \boldsymbol{b}_L\}$
Set step size $\alpha$ and initialization bounds $\{s_l\}_{l=1}^{L}$ (e.g., $s_l = 1/\sqrt{n_l}$), where $n_l$ is fan_in
Set clipping parameter $\kappa$ (e.g., $\kappa = 2$)
Initialize weights $W_{l,i,j} \sim U[-s_l, s_l], \forall i, j, l$

for $S$ in $\mathcal{D}$ do
  for $l$ in $\{L, L-1, ..., 1\}$ do
    $\boldsymbol{W}_l \leftarrow \boldsymbol{W}_l - \alpha \Delta \boldsymbol{W}_l$        ▷ Weight Update (e.g., $\Delta \boldsymbol{W}_l = \nabla_{\boldsymbol{W}_l} \mathcal{L}(S)$ in SGD)
    $\boldsymbol{b}_l \leftarrow \boldsymbol{b}_l - \alpha \Delta \boldsymbol{b}_l$        ▷ Bias Update (e.g., $\Delta \boldsymbol{b}_l = \nabla_{\boldsymbol{b}_l} \mathcal{L}(S)$ in SGD)
    $\boldsymbol{W}_l \leftarrow \text{Clip}(\boldsymbol{W}_l, \min = -\kappa s_l, \max = \kappa s_l)$
    $\boldsymbol{b}_l \leftarrow \text{Clip}(\boldsymbol{b}_l, \min = -\kappa s_l, \max = \kappa s_l)$
```

One natural question is whether we reduce the expressivity of the network by weight clipping. Typically, the larger the neural network, the smaller the weight change is to reduce the loss (Ghorbani et al. 2019, Geiger et al. 2020). For example, in over-parameterized networks, the weight change is usually tiny to represent any function, which is referred to as lazy training (see Chizat et al. 2019). Thus, we suspect weight clipping would not reduce much of the expressivity of over-parameterized networks due to tiny weight changes and a relatively large clipping range. We leave studying the effect of weight clipping on neural network expressivity to future work.

Next, we investigate the effect of weight clipping on smoothing the neural network. Liu et al. (2022) showed that reducing the Lipschitz constant (e.g., via Lipschitz regularization) increases the smoothness of the functions represented by neural networks and, hence, improves generalization (Yoshida et al. 2017), stabilizes Wasserstein generative adversarial networks (Arjovsky et al. 2017), and protects against adversarial attacks (Li et al. 2019). We start our analysis by showing that weight clipping leads to bounding the Lipschitz constant. In Theorem 1, we show that weight clipping bounds the Lipschitz constant (see proof in Appendix A). For simplicity, we consider fully connected networks and 1-Lipschitz activation functions (e.g., ReLU, Leaky ReLU, Tanh), but the results can be extended to other networks (see Gouk et al. 2021) and other activation functions.

Theorem 1. ***Smoothness of Clipped Networks.*** *Consider a fully-connected neural network $f_{\mathcal{W}} : \mathcal{X} \to \mathcal{Y}$ parametrized by the set of augmented weight matrices (include biases) $\mathcal{W}_{Aug} = \{\boldsymbol{W}_1, \dots, \boldsymbol{W}_L\}$. If the activation function σ used is 1-Lipschitz (e.g., ReLU), then the clipped network $f_{\mathcal{W}}^{Clipped}$ is Lipschitz continuous. That is, $\exists k \geq 0$ such that $\|f_{\mathcal{W}}^{Clipped}(\boldsymbol{x}_1) - f_{\mathcal{W}}^{Clipped}(\boldsymbol{x}_2)\|_1 \leq k\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_1, \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$.*

This result suggests that weight clipping adds an upper bound to the sharpness level of the network. In contrast, while other regularization methods can also improve smoothness (e.g., Liu et al. 2022), they do not have any guarantees on the bounds of the Lipschitz constant, and, hence, they can sometimes still reach sharp solutions.

Finally, we show that weight clipping makes the function change bounded. Update boundedness is a desired property of reinforcement learning methods, especially for the on-policy ones such as PPO (Schulman et al. 2017), to prevent the policy from changing too much from its old policy, achieving more learning stability. In corollary 1, we show how weight clipping leads to update boundedness using the fact that clipped networks are Lipschitz continuous (see proof in Appendix A).

Corollary 1. ***Update Boundedness of Clipped Networks.*** *Consider a clipped fully-connected neural network $f_{\mathcal{W}}^{Clipped}$ parameterized by the set of augmented weight matrices (include biases) $\mathcal{W}_{Aug} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L\}$. If the activation function σ used is 1-Lipschitz (e.g., ReLU), then any function update $\|\Delta f_{\mathcal{W}}^{Clipped}\|_1$ is bounded.*

4 Experiments

In this section, we study the effectiveness of weight clipping in 1) improving generalization, 2) maintaining network plasticity, 3) mitigating policy collapse, and 4) facilitating learning with large replay ratios. We start by considering the warm start setup (see Ash & Adams 2019) and show that weight clipping can reduce loss of generalization. We then evaluate weight clipping using non-stationary streaming learning problems introduced by Elsayed and Mahmood (2024). Specifically, we use the Input-Permuted MNIST, Label-Permuted EMNIST, and Label-Permuted mini-ImageNet problems. Finally, we evaluate the effectiveness of weight clipping in addressing policy collapse in PPO (Schulman et al. 2017) and improving the performance of DQN (Mnih et al. 2015) and Rainbow (Hessel et al. 2018) with a large replay ratio.

The performance is evaluated based on the test accuracy in the warm-starting problem, the average online accuracy for the streaming learning problems, and the non-discounted episodic return for the reinforcement learning problems. We perform a hyper-parameter search (see Appendix B) for each method and use the best-performing configuration to plot in the following experiments. Our criterion for the best-performing configuration is the one that maximizes the area under the accuracy curve in supervised learning problems and the area under the non-discounted episodic return for the reinforcement learning problems.

4.1 Weight Clipping for Improved Generalization

Here, we study the role of weight clipping (WC) in improving generalization. We use the warm starting setup proposed by Ash and Adams (2019) with the CIFAR-10 dataset (Krizhevsky 2009). Two variations of SGD are trained from scratch using ResNet18 (He et al. 2016), one using 100% of the training and another in two stages: on 50% of the data, followed by the other 50%. In Fig. 2, when SGD is trained in two stages, we observe that its test accuracy is lower than if it was trained on the full data in one stage, which gives the same loss of generalization phenomenon demonstrated by Ash and Adams (2019). We introduce weight clipping in two ways: 1) clip once after the training on the first half of the data and 2) clip every time step. We observe that weight clipping only once removes the generalization gap and improves the test accuracy significantly. On the other hand, weight clipping every time step improves generalization in both stages of learning. These results show that large weights can cause overfitting and reduction in performance, which is alleviated by weight clipping that can improve generalization. The results are averaged over 10 independent runs.

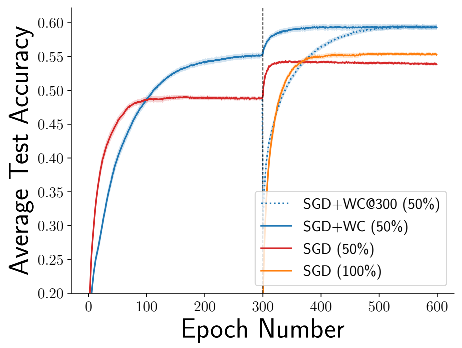
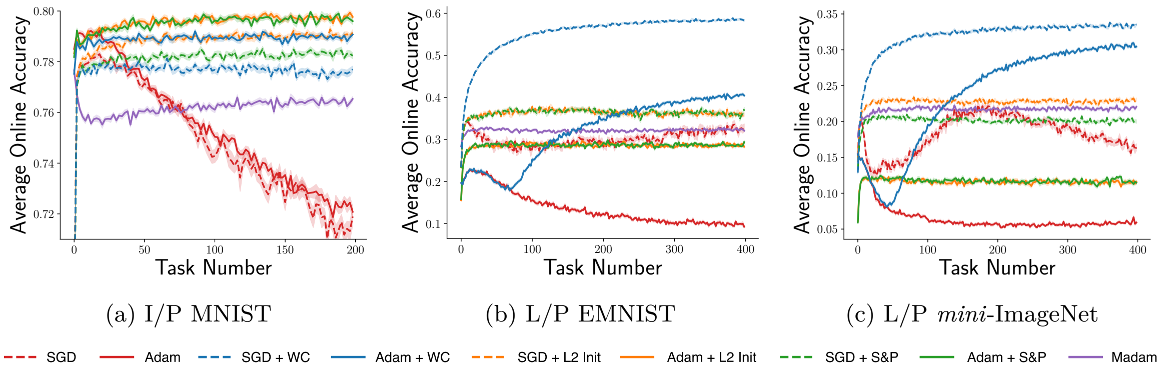


Figure 2: Performance when training on the data sequentially against when data is aggregated. The shaded region represents the standard error.



(a) I/P MNIST (b) L/P EMNIST (c) L/P *mini*-ImageNet

Figure 3: Performance of Adam and SGD with Weight Clipping on Input-permuted MNIST, Label-Permuted EMNIST, and Label-Permuted *mini*-ImageNet. All curves are averaged over 20 independent runs. The shaded area represents the standard error

4.2 Weight Clipping in Streaming Learning

Now, we study weight clipping in addressing loss of plasticity using the Input-permuted MNIST problem. The input-permuted MNIST problem is a standard problem for studying loss of plasticity in neural networks since the learned features become irrelevant to the next task after each permutation. We permute the inputs every 5000 time step. A new task starts when a permutation is performed. Next, we use the label-permuted EMNIST and label-permuted mini-ImageNet problems where loss of plasticity is intertwined with catastrophic forgetting (Elsayed & Mahmood 2024). In both label-permuted problems, we permuted the labels each 2500 time step.

We compare SGD and Adam (Kingma & Ba 2014) along with their variations with two regularization methods, Shirnk & Perturb (S&P) and L2 Init, against weight clipping with SGD and Adam. In previous works (Kumar et al. 2023a, Dohare et al. 2023a), L2-Init and S&P have been shown to be effective in maintaining plasticity when combined with SGD but have not been studied when combined with Adam. In addition, we compare against the Madam optimizer (Bernstein et al. 2020), which uses weight clipping to reduce its exponential weight growth but has not been studied in non-stationary settings before. In the streaming classification problems, the learner is required to maximize the online average accuracy using a stream of data, one sample at a time. Each learner is presented with 1M samples and uses a multi-layer (300×150) network with *leaky-relu* units.

Fig. 3a shows that almost all methods except for SGD and Adam maintain their performance throughout, but by different degrees. We plot the average online accuracy against the number of tasks, where each point in the figure represents the percentage of correct predictions within the task since the sample online prediction is 1 for correct predictions and 0 for incorrect ones.

In Fig. 4, we characterize the solutions of each method using diagnostic statistics. Specifically, we show the online loss, ℓ_2-norm of weights, and ℓ_2-norm of gradients. In addition, we show the average online plasticity of each method using the sample plasticity metric (Elsayed & Mahmood 2024), which is given by $p(Z) = \max\left(1 - \frac{\mathcal{L}(\mathcal{W}^\dagger, Z)}{\max(\mathcal{L}(\mathcal{W}, Z), \epsilon)}, 0\right) \in [0,1]$, where Z is the sample, $\mathcal{W}^\dagger$ is the set of weights after performing the update and ϵ is a small number for numerical stability. We observe a gradual increase of the ℓ_2 norm of the weights of SGD and Adam compared to other methods.

Next, we use the label-permuted problems to evaluate the role of weight clipping in not biasing the weights towards some point. The label-permuted problems involve label permutation, which means the learned representation by the learner does not need to change, and the learner can continually improve upon them instead of overwriting and then re-learning. Fig. 3b and 3c show that while all methods addressing loss of plasticity can maintain their performance, only weight clipping can keep improving its performance, likely due to not biasing the weights toward a specific point. We defer the diagnostic statistics on these two problems to Appendix C.

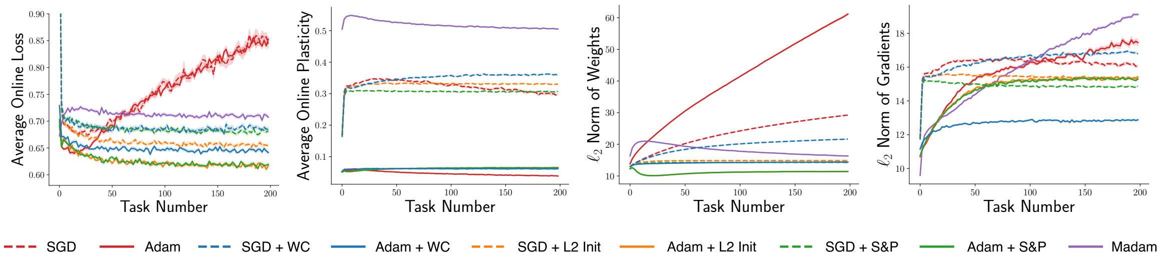


Figure 4: Diagnostic Statistics of different methods in Input-permuted MNIST. We show the online loss, the online plasticity, the ℓ_2-norm of gradients, and the ℓ_2-norm of weights.

4.3 Weight Clipping Against Policy Collapse

In this section, we study the role of weight clipping in mitigating policy collapse (Dohare et al. 2023b). Dohare et al. (2023b) demonstrated that the performance of PPO can collapse if trained for longer periods of time. We use CleanRL's implementation (Huang et al. 2022) with the default hyper-parameters as suggested by Dohare et al. (2023b). The network used is multi-layered (64×64) with *tanh* activations. Fig. 5 illustrates the phenomenon of policy collapse where the performance of PPO with Adam drops gradually with time in a number of MuJoCo (Todorov et al. 2012) environments. Weight clipping is effective in mitigating policy collapse, allowing for continual improvement. Here, we only show the performance in the MuJoCo environments with which policy collapse happens and exclude other environments where PPO with Adam experiences no policy collapse. We further investigate why policy collapse happens using the approximate KL given by $(1 - r) - \log r$ as our diagnostic metric, where r is the ratio in PPO between the current policy and the old policy. Fig. 6 shows the approximate KL throughout learning using Adam against Adam+WC. We observe that the approximate KL with Adam increases, indicating that the current policy deviates a lot from the old policy, in contrast to Adam+WC, which maintains small values, stabilizing learning in PPO. Next, we show additional diagnostic statistics in Fig. 7 to characterize the solutions found by Adam and Adam+WC. Notably, we observe that weight clipping reduces the ℓ_2 norm of the weights and reduces the percentage of saturated units. We defer diagnostic statistics on the rest of the environment in Appendix C.

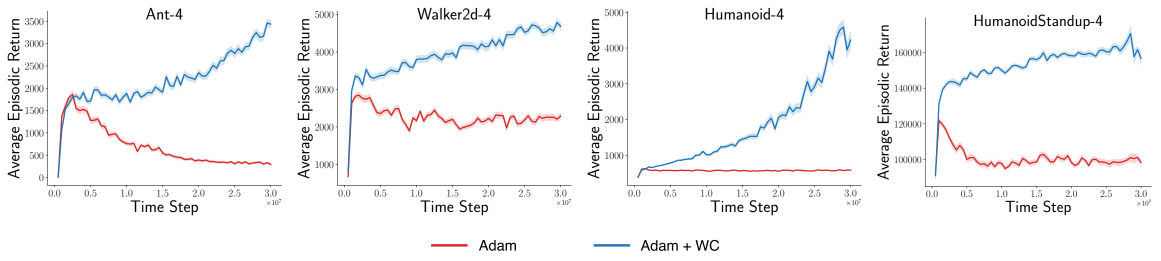


Figure 5: Policy Collapse in PPO. The performance of PPO with Adam drops when trained for longer in contrast to Adam+WC, which can keep improving its performance. All curves are averaged over 30 independent runs. The shaded area represents the standard error.

4.4 Weight Clipping with Large Replay Ratios

The replay ratio (RR) is the number of gradient updates performed per environment step. An increase of RR helps a learning agent extract as much information as possible from transitions, resulting in a higher sample efficiency. However, in practice, when the RR is too high, the learning agent would overfit to a small amount of data and lose the plasticity to learn new information due to aggressive parameter updates, thus reducing the learning performance (Nikishin et al. 2022).

In this section, we show that by incorporating weight clipping, we can prevent the agent from aggressive parameter updates and improve sample efficiency by a large amount under a high RR

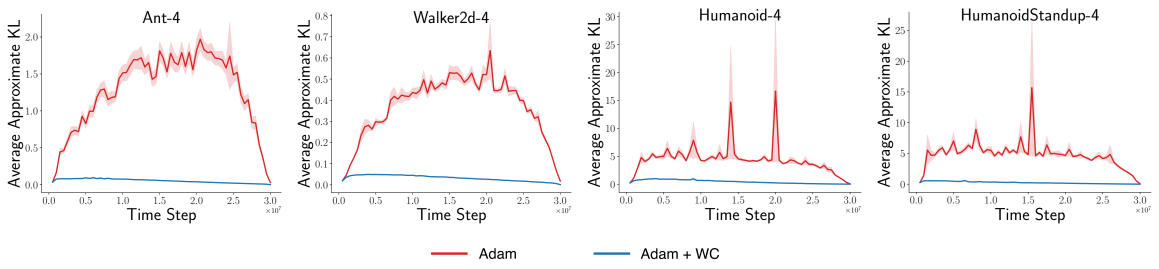


Figure 6: Approximate KL between the current and old policy. The values reach zero at the end since learning rate annealing is used.

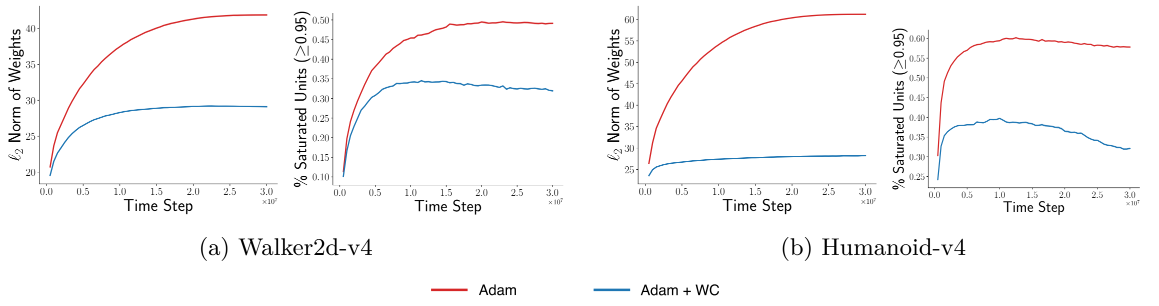


Figure 7: Diagnostic Statistics for policy collapse. The ℓ_2 norm of the weights and percentage of saturated units are shown. A tanh unit is considered saturated if $|x| \geq 0.95$, where x is its output.

setting. Specifically, we train DQN (Mnih et al. 2015) and Rainbow (Hessel et al. 2018) with $RR = 1$ in several Atari (Bellemare et al. 2013) tasks, optimized by Adam or Adam+WC. All our experiments are conducted using the Tianshou framework (Weng et al. 2022) and the default RR is 0.1. Note that due to the high computation cost of using a high RR, we train both agents for 10M frames and summarize all results across over 5 random seeds, as shown in Figure 8. Overall, without weight clipping, both DQN and Rainbow perform poorly and have low sample efficiency. Once weight clipping is applied, we observe a great improvement in sample efficiency and learning performance, demonstrating the effectiveness of our method.

To further investigate the influence of weight clipping in terms of optimization and generalization, we visualize the gradient covariance matrices of training DQN in Space Invaders, optimized by Adam and Adam+WC in Fig. 9. The heatmaps for all Atari tasks are included in Appendix C.3. Specifically, the gradient covariance matrix is known to be strongly related to optimization and generalization (Fort et al. 2019, Lyle et al. 2022; Lyle et al. 2023). Formally, we estimate it by sampling k training points and compute it entrywise as

$$C_k[i,j] = \frac{\langle \nabla_\theta \ell(\theta, \mathbf{x}_i), \nabla_\theta \ell(\theta, \mathbf{x}_j)\rangle}{\|\nabla_\theta \ell(\theta, \mathbf{x}_i)\| \|\nabla_\theta \ell(\theta, \mathbf{x}_j)\|},$$

where $\mathbf{x}_1, \cdots, \mathbf{x}_k$ are randomly sampled training points and ℓ is the loss function. In Fig. 9, we observe that when weight clipping is applied, the gradient covariance matrix has smaller off-diagonal values, indicating a smoother loss landscape and less interference.

5 Related Works

Biologically plausible NNs. Physical and biological systems usually have bounded outputs, as components or elements within these systems often have inherent limits or constraints. For example, in neurobiological processes, synaptic weights are assumed to have a maximum value (Michiels van Kessenich et al. 2016). Weight clipping can be viewed as a biologically plausible mechanism that improves plasticity in artificial neural networks.

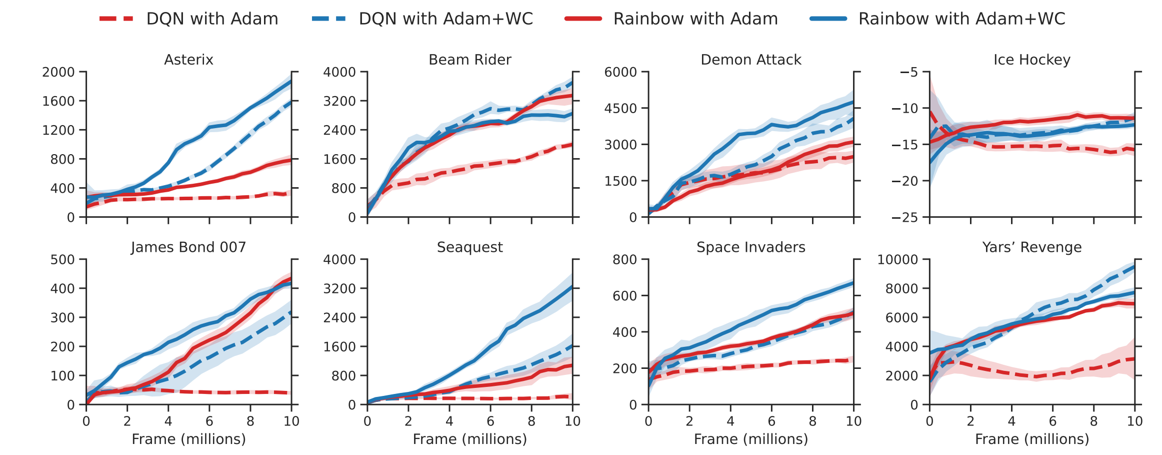


Figure 8: The performance of DQN and Rainbow trained with $RR = 1$ in Atari tasks, optimized by Adam and Adam+WC. Solid lines correspond to the mean performance over 5 random seeds, and the shaded areas correspond to 90% confidence interval.

Gradient Clipping. Gradient clipping has been used to stabilize learning which avoids instability by preventing large updates to the network. Barczewski & Ramon (2023) showed that weight clipping improves optimization performance compared to gradient clipping, likely due to the fact that weight clipping is not biased. Nevertheless, gradient clipping is often an essential component of some reinforcement learning algorithms to stabilize learning and control the maximum size of the update (e.g., Badia et al. 2020, Hafner et al. 2023, Mnih et al. 2016).

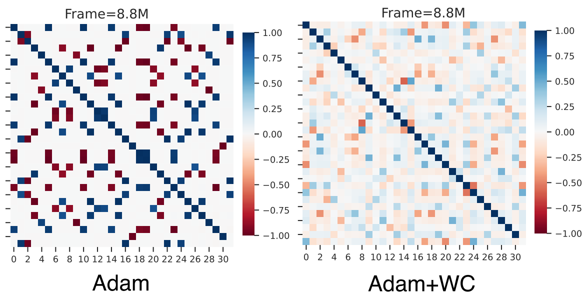


Figure 9: The gradient covariance heatmap with DQN in Space Invaders, optimized by Adam and Adam+WC, respectively. Optimizing with Adam+WC results in smaller off-diagonal values in the heatmap compared to Adam.

Weight Clipping in the Literature. Weight clipping has appeared in previous works to achieve different desiderata. For example, Bernstein et al. (2020) introduced the Madam optimizer, which uses weight clipping to limit the exponential growth of the weights when using a multiplicative weight update rule. Arjovsky et al. (2017) used weight clipping to stabilize Wasserstein generative adversarial networks. Moreover, weight clipping has been used in binary neural networks to help binarize the weights (Courbariaux et al. 2015, Alizadeh et al. 2018).

Wasserstein Regularization. Wasserstein regularization (Lewandowski et al. 2024) aims to address the loss of plasticity while allowing parameters to deviate from initialization. However, the Wasserstein metric usually requires sorting of the parameters, which can be expensive. Weight Clipping allows the weights to deviate instead of biasing towards a specific point, achieving a similar goal as Wasserstein regularization but having substantially lower computational requirements.

6 Conclusion

In this paper, we introduced weight clipping as a simple mechanism that helps with learning under non-stationarity. Weight clipping can be used besides existing methods without any major change to the optimizer or the network used. Our results show that weight clipping can help mitigate loss of plasticity in streaming learning, alleviate policy collapse, and improve performance when learning with large replay ratios. Future work can perform adaptive weight clipping that does not require any hyper-parameter tuning or develop Lipschitz regularization methods that guarantee smoothness.

Acknowledgments

We gratefully acknowledge funding from the Canada CIFAR AI Chairs program, the Reinforcement Learning and Artificial Intelligence (RLAI) laboratory, the Alberta Machine Intelligence Institute (Amii), and the Natural Sciences and Engineering Research Council (NSERC) of Canada. We would also like to thank the Digital Research Alliance of Canada for providing the computational resources needed. We thank Mark Rowland for the valuable feedback that helped improve this paper. Qingfeng Lan would also like to acknowledge Alberta Innovates for the support they provided him for his research.

References

Abbas, Z., Zhao, R., Modayil, J., White, A., & Machado, M. C. (2023). Loss of plasticity in continual deep reinforcement learning. *arXiv preprint arXiv:2303.07507.*

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. *International Conference on Machine Learning* (pp. 214-223).

Alizadeh, M., Fernández-Marqués, J., Lane, N. D., & Gal, Y. (2018). An empirical study of binary neural networks' optimisation. *International Conference on Learning Representations.*

Ash, J., & Adams, R. P. (2020). On warm-starting neural network training. *Advances in Neural Information Processing Systems*, *33*, 3884-3894.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., & Blundell, C. (2020). Agent57: Outperforming the Atari human benchmark. *International Conference on Machine Learning* (pp. 507-517).

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, *47*, 253-279.

Brock, A., De, S., Smith, S. L. & Simonyan, K. (2021). High-Performance Large-Scale Image Recognition Without Normalization. *International Conference on Machine Learning* (pp. 1059-1071).

Bernstein, J., Zhao, J., Meister, M., Liu, M. Y., Anandkumar, A., & Yue, Y. (2020). Learning compositional functions via multiplicative weight updates. *Advances in neural information processing systems*, *33*, 13319-13330.

Barczewski, A., & Ramon, J. (2023). DP-SGD with weight clipping. *arXiv preprint arXiv:2310.18001.*

Courbariaux, M., Bengio, Y., & David, J. P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. Advances in neural information processing systems, 28.

Chizat, L., Oyallon, E., & Bach, F. (2019). On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, *32.*

Dohare, S., Sutton, R. S., & Mahmood, A. R. (2021). Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*.

Dohare, S., Hernandez-Garcia, J. F., Rahman, P., Sutton, R. S., & Mahmood, A. R. (2023a). Loss of plasticity in deep continual learning. *arXiv preprint arXiv:2306.13812*.

Dohare, S., Lan, Q., & Mahmood, A. R. (2023b). Overcoming policy collapse in deep reinforcement learning. *European Workshop on Reinforcement Learning*.

Elsayed, M., & Mahmood, A. R. (2024). Addressing loss of plasticity and catastrophic forgetting in continual learning. *International Conference on Learning Representations*.

Fort, S., Nowak, P. K., Jastrzebski, S., & Narayanan, S. (2019). Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*.

Garg, S., Tosatto, S., Pan, Y., White, M., & Mahmood, A. R. (2022). An alternate policy gradient estimator for softmax policies. *International Conference on Artificial Intelligence and Statistics*.

Gouk, H., Frank, E., Pfahringer, B., & Cree, M. J. (2021). Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning*, *110*, 393-416.

Geiger, M., Spigler, S., Jacot, A., & Wyart, M. (2020). Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2020*(11), 113301.

Ghorbani, B., Mei, S., Misiakiewicz, T., & Montanari, A. (2019). Limitations of lazy training of two-layers neural network. *Advances in Neural Information Processing Systems*, *32*.

Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision* (pp. 1026-1034).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition* (pp. 770-778).

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., ... & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. *AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).

Hayes, T. L., & Kanan, C. (2022). Online Continual Learning for Embedded Devices. *Conference on Lifelong Learning Agents* (pp. 744-766).

Hayes, T. L., Cahill, N. D., & Kanan, C. (2019). Memory efficient experience replay for streaming learning. *International Conference on Robotics and Automation* (pp. 9769-9776).

Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., & AraÃšjo, J. G. (2022). Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, *23*(274), 1-18.

Kumar, S., Marklund, H., & Van Roy, B. (2023a). Maintaining plasticity via regenerative regularization. *arXiv preprint arXiv:2308.11958.*

Kumar, S., Marklund, H., Rao, A., Zhu, Y., Jeon, H. J., Liu, Y., & Van Roy, B. (2023b). Continual learning as computationally constrained reinforcement learning. *arXiv preprint arXiv:2307.04345.*

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Krizhevsky, A. (2009) *Learning Multiple Layers of Features from Tiny Images.* Ph.D. dissertation, University of Toronto.

Krogh, A., & Hertz, J. (1991). A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*, *4*.

Lan, Q., & Mahmood, A. R. (2023). Elephant neural networks: Born to be a continual learner. *arXiv preprint arXiv:2310.01365.*

Lan, Q., Pan, Y., Luo, J., & Mahmood, A. R. (2023). Memory-efficient Reinforcement Learning with Value-based Knowledge Consolidation. *Transactions on Machine Learning Research.*

Li, Q., Haque, S., Anil, C., Lucas, J., Grosse, R. B., & Jacobsen, J. H. (2019). Preventing gradient attenuation in Lipschitz constrained convolutional networks. *Advances in Neural Information Processing Systems*, *32*.

Liu, H. T. D., Williams, F., Jacobson, A., Fidler, S., & Litany, O. (2022). Learning smooth neural functions via Lipschitz regularization. *ACM SIGGRAPH 2022 Conference Proceedings* (pp. 1-13).

Lewandowski, A., Tanaka, H., Schuurmans, D., & Machado, M. C. (2023). Curvature Explains Loss of Plasticity. *arXiv preprint arXiv:2312.00246.*

Lyle, C., Rowland, M., & Dabney, W. (2021). Understanding and Preventing Capacity Loss in Reinforcement Learning. *International Conference on Learning Representations.*

Lyle, C., Rowland, M., Dabney, W., Kwiatkowska, M., & Gal, Y. (2022). Learning dynamics and generalization in reinforcement learning. *International Conference on Machine Learning.*

Lyle, C., Zheng, Z., Nikishin, E., Pires, B. A., Pascanu, R., & Dabney, W. (2023). Understanding Plasticity in Neural Networks. *International Conference on Machine Learning.*

Lyle, C., Zheng, Z., Khetarpal, K., van Hasselt, H., Pascanu, R., Martens, J., & Dabney, W. (2024). Disentangling the causes of plasticity loss in neural networks. *arXiv preprint arXiv:2402.18762.*

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529-533.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *International Conference on Machine Learning* (pp. 1928-1937).

Michiels van Kessenich, L., De Arcangelis, L., & Herrmann, H. J. (2016). Synaptic plasticity and neuronal refractory time cause scaling behaviour of neuronal avalanches. *Scientific reports*, *6*(1), 32071.

Nikishin, E., Oh, J., Ostrovski, G., Lyle, C., Pascanu, R., Dabney, W., & Barreto, A. (2023). Deep reinforcement learning with plasticity injection. *Advances in Neural Information Processing Systems*, 36.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P. L., & Courville, A. (2022). The primacy bias in deep reinforcement learning. *International Conference on Machine Learning* (pp. 16828-16847).

Sokar, G., Agarwal, R., Castro, P.S., & Evci, U. (2023). The Dormant Neuron Phenomenon in Deep Reinforcement Learning. *International Conference on Machine Learning.*

Sutton, R. S. & Barto, A. G. (2018). Reinforcement Learning: An Introduction. *MIT Press.*

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347.*

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. International Conference on Intelligent Robots and Systems (pp. 5026-5033).

Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., ... & Zhu, J. (2022). Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, *23*(1), 12275-12280.

Wortsman, M., Liu, P. J., Xiao, L., Everett, K., Alemi, A., Adlam, B., ... & Kornblith, S. (2023). Small-scale proxies for large-scale Transformer training instabilities. *arXiv preprint arXiv:2309.14322.*

Yoshida, Y., & Miyato, T. (2017). Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941.*

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, *64*(3), 107-115.

A Proofs

A.1 Proof of Theorem 1

Proof. Let us first rewrite our feed-forward neural network function for a given input $\boldsymbol{x}$ as follows:

$$f_{\mathcal{W}} = (\phi_L \circ \sigma \ldots \sigma \circ \phi_1)(\boldsymbol{x}),$$

where ϕ_i defines the linear operation given by $\boldsymbol{W}_i$ and $\circ$ denotes the composition between two functions. The composition of k_1-Lipschitz and k_2-Lipschitz function is $k_1 k_2$-Lipschitz (see Gouk et al. 2021). Thus, the Lipschitz constant of the network is given by

$$L(f) \leq \prod_{i=1}^{L} L(\sigma_i) L(\phi_i) = \prod_{i=1}^{L} L(\phi_i),$$

where $L(\phi_i) = \sup_{\boldsymbol{a} \neq 0} \frac{\|\boldsymbol{W}_i \boldsymbol{a}\|_p}{\|\boldsymbol{a}\|_p}$ is the operator norm of $\boldsymbol{W}_i$. Since $L(f)$ depends on the operator norm of weight matrices at each layer, it can change during learning, producing less smooth functions.

Next, we show the Lipschitz constant of the clipped network. For simplicity of the proof, we consider $p = 1$. The Lipschitz constant of the clipped network $L(f^{\text{Clipped}})$ is bounded by:

$$L(f^{\text{Clipped}}) \leq \prod_{l=1}^{L} \sup_{\boldsymbol{a} \neq 0} \frac{\|\boldsymbol{W}_l \boldsymbol{a}\|_1}{\|\boldsymbol{a}\|_1} = \prod_{l=1}^{L} \max_j \left(\sum_{i=1}^{m_l} |W_{l,i,j}| \right) \leq \kappa^L \prod_{i=1}^{L} s_l m_l,$$

where $\boldsymbol{W}_l \in \mathbb{R}^{m_l \times n_l}$ and $|W_{l,i,j}| \leq \kappa s_l, \forall l, i, j$. □

A.2 Proof of Corollary 1

Proof. For simplicity, let us vectorize $\mathcal{W}$ and combine it with the input vector $\boldsymbol{x}$ in a single vector $\boldsymbol{\theta} \in \mathcal{P}$. Let us consider the two instances $\boldsymbol{\theta}^{(1)}$ and $\boldsymbol{\theta}^{(2)}$ corresponding to before and after making an update with the same input, $\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} + \Delta \boldsymbol{\theta}^{(1)}$. From Theorem 1, $\exists k \geq 0$ such that the clipped network $f_{\mathcal{W}}^{\text{Clipped}}$ is k-Lipschitz. Thus we can write the following:

$$\|f^{\text{Clipped}}(\boldsymbol{\theta}^{(2)}) - f^{\text{Clipped}}(\boldsymbol{\theta}^{(1)})\|_1 \leq k \|\boldsymbol{\theta}^{(2)} - \boldsymbol{\theta}^{(1)}\|_1, \forall \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)} \in \mathcal{P}.$$

Thus, the change in the function $f_{\mathcal{W}}^{\text{Clipped}}$ can be written as:

$$\|\Delta f_{\mathcal{W}}^{\text{Clipped}}\|_1 \leq k \|\boldsymbol{\theta}^{(2)} - \boldsymbol{\theta}^{(1)}\|_1 = k \sum_{l=1}^{L} \sum_{i=1}^{m_l} \sum_{j=1}^{n_l} |W_{l,i,j}^{(2)} - W_{l,i,j}^{(1)}| \leq 2k \sum_{l=1}^{L} m_l n_l s_l,$$

where $\boldsymbol{W}_l^{(1)} \in \mathbb{R}^{m_l \times n_l}$ is the weight matrix at layer l before making the update and $\boldsymbol{W}_l^{(2)} \in \mathbb{R}^{m_l \times n_l}$ is the weight matrix at layer l after making the update. □

B Hyperparameter Search Space

In this section, we present the hyper-parameter search space in Table 1 and the best set of hyper-parameter configurations of each method in Table 2.

Problem		**Space**	**Method**
Input-permuted MNIST **Label-permuted EMNIST** **Label-permuted *mini*-ImageNet**	step size α	$\{0.1, 0.01, 0.001, 0.0001\}$	All
	noise std. deviation σ	$\{0.0, 0.1, 0.01, 0.001, 0.0001\}$	S&P
	weight decay factor λ	$\{0.0, 0.1, 0.01, 0.001, 0.0001\}$	S&P, L2 Init
	clipping param κ	$\{1, 2, 3, 4, 5\}$	Madam, WC
	Number of Seeds N	$\{20\}$	All
Atari Environments	step size α	DQN $\{0.0001\}$, Rainbow $\{0.0000625\}$	All
	clipping parameter κ	$\{1, 5\}$	Adam+WC
	Number of Seeds N	$\{5\}$	All
MuJoCo Environemnts	step size α	$\{0.0001\}$	All
	clipping param κ	$\{1, 3, 5\}$	Adam+WC
	Number of Seeds N	$\{30\}$	All
Warm Starting	step size α	$\{0.001\}$	All
	clipping param κ	$\{1, 2, 5, 10, 20\}$	SGD+WC
	Number of Seeds N	$\{10\}$	All

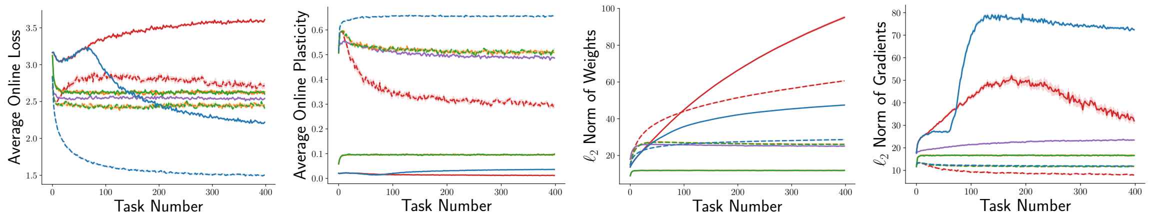
Table 1: Search Space for Streaming Learning Experiments.

Problem	**Method**	**Best Set**
Input-permuted MNIST	SGD	$\alpha = 0.001$
	Adam	$\alpha = 0.0001$
	SGD + L2 Init	$\alpha = 0.001, \lambda = 0.01$
	Adam + L2 Init	$\alpha = 0.0001, \lambda = 0.001$
	SGD + S&P	$\alpha = 0.001, \sigma = 0.1, \lambda = 0.01$
	Adam + S&P	$\alpha = 0.0001, \sigma = 0.1, \lambda = 0.001$
	SGD + WC	$\alpha = 0.001, \kappa = 2.0$
	Adam + WC	$\alpha = 0.0001, \kappa = 1.0$
	Madam	$\alpha = 0.01, \kappa = 4,$
Label-permuted EMNIST	SGD	$\alpha = 0.01$
	Adam	$\alpha = 0.0001$
	SGD + L2 Init	$\alpha = 0.01, \lambda = 0.001$
	Adam + L2 Init	$\alpha = 0.001, \lambda = 0.01$
	SGD + S&P	$\alpha = 0.01, \sigma = 0.01, \lambda = 0.001$
	Adam + S&P	$\alpha = 0.001, \sigma = 0.001, \lambda = 0.01$
	SGD + WC	$\alpha = 0.01, \kappa = 2.0$
	Adam + WC	$\alpha = 0.0001, \kappa = 3.0$
	Madam	$\alpha = 0.01, \kappa = 5,$
Label-permuted *mini*-ImageNet	SGD	$\alpha = 0.01$
	Adam	$\alpha = 0.0001$
	SGD + L2 Init	$\alpha = 0.01, \lambda = 0.01$
	Adam + L2 Init	$\alpha = 0.001, \lambda = 0.01$
	SGD + S&P	$\alpha = 0.01, \sigma = 0.01, \lambda = 0.01$
	Adam + S&P	$\alpha = 0.001, \sigma = 0.0, \lambda = 0.01$
	SGD + WC	$\alpha = 0.01, \kappa = 1.0$
	Adam + WC	$\alpha = 0.0001, \kappa = 3.0$
	Madam	$\alpha = 0.01, \kappa = 5,$
MuJoCo Environments	Adam	$\alpha = 0.0003$
	Adam+WC	$\alpha = 0.0003$ and $\kappa = 3$ for all except for Ant-v4 ($\kappa = 5$)
Atari Environments	Adam	$\alpha = 0.0001$ for DQN and 0.0000625 for Rainbow
	Adam+WC	$\alpha = 0.0001$ for DQN and 0.0000625 for Rainbow with $\kappa = 1$
Warm Starting	SGD	$\alpha = 0.001$
	SGD + WC	$\alpha = 0.001, \kappa = 10$
	SGD + WC@300	$\alpha = 0.001, \kappa = 20$

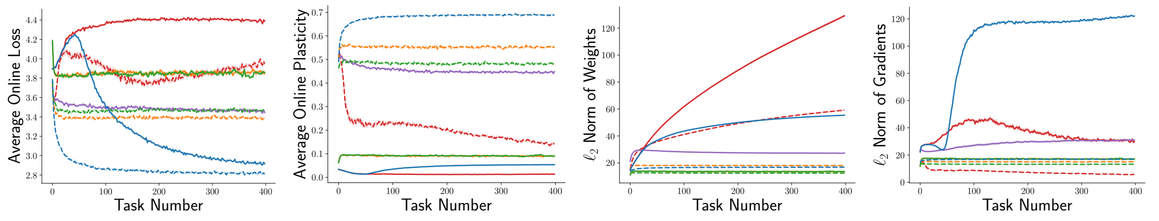
Table 2: Best hyperparameter set of each method in each problem.

C Additional Experimental Results

C.1 Diagnostic statistics for Label-permuted EMNIST and *mini*-ImageNet



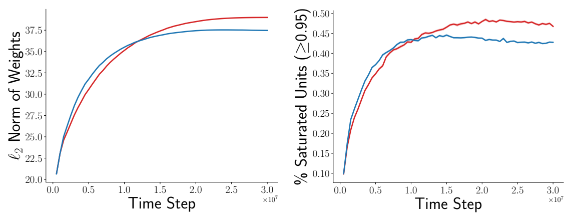
(a) Label-pertmued EMNIST.



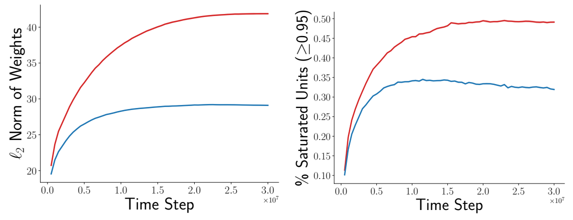
(b) Label-pertmued *mini*-ImageNet.

Figure 10: Diagnostic Statistics of different methods in Label-permuted EMNIST and Label-permuted *mini*-ImageNet. We show the online loss, the online plasticity, the ℓ_2-norm of gradients, and the ℓ_2-norm of weights.

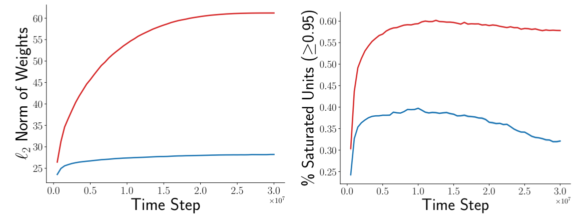
C.2 Diagnostic statistics of PPO



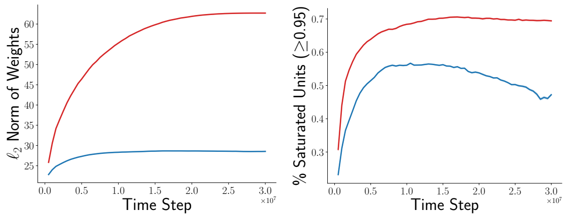
(a) Ant-v4



(b) Walker2d-v4



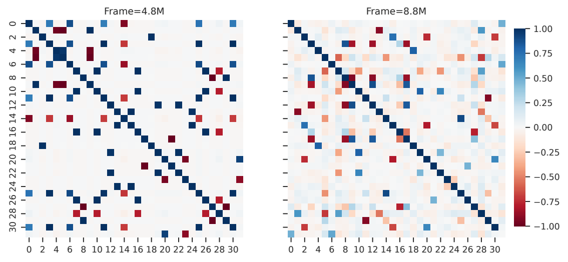
(c) Humanoid-v4



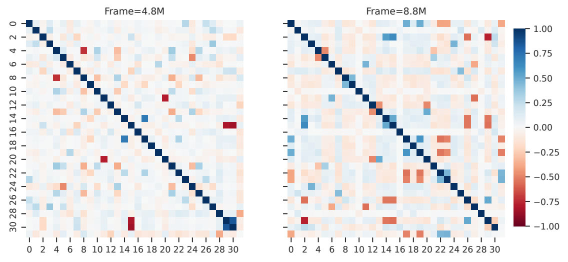
(d) HumanoidStandup-v4

Figure 11: Diagnostic Statistics for policy collapse. The ℓ_2 norm of the weights and percentage of saturated units are shown. A tanh unit is considered saturated if $|x| \leq 0.95$, where x is its output. In all environments, we use $\kappa = 3$ except for Ant-v4, in which we found that $\kappa = 5$ performs better. The larger value of κ used in Ant-v4 explains the smaller effect of weight clipping on the ℓ_2 norm of the weights and the percentage of saturated units compared to other environments, although it still has a large effect on reducing the approximate KL (see Fig. 6).

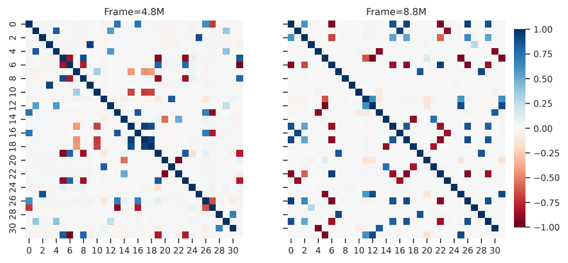
C.3 The Gradient Covariance Heatmaps of Training DQN and Rainbow in Atari Games



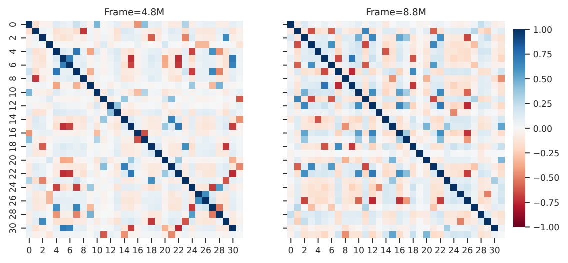
(a) DQN in Asterix with Adam



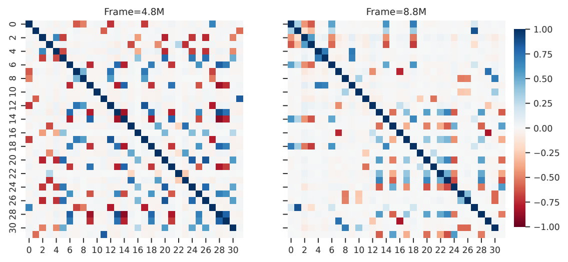
(b) DQN in Asterix with Adam+WC



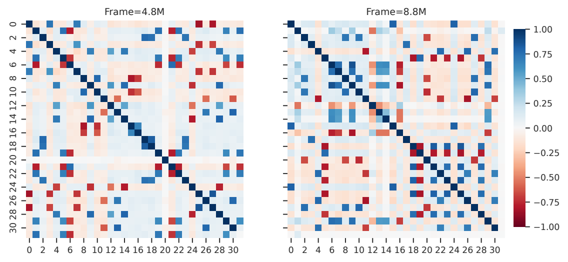
(c) DQN in Beam Rider with Adam



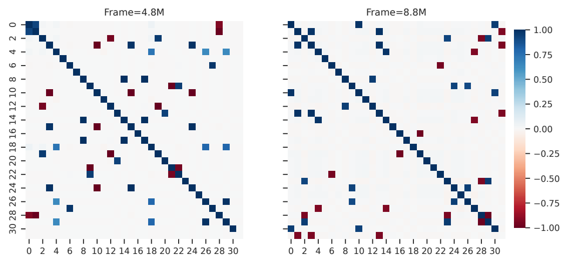
(d) DQN in Beam Rider with Adam+WC



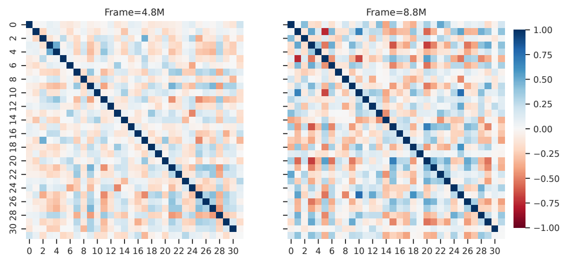
(e) DQN in Demon Attack with Adam



(f) DQN in Demon Attack with Adam+WC

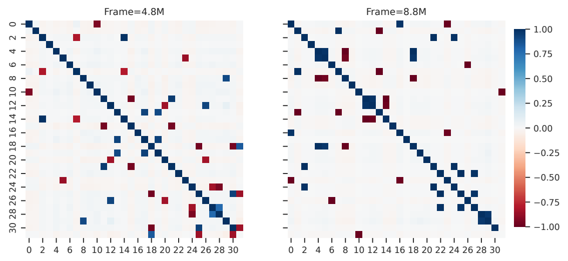


(g) DQN in Ice Hockey with Adam

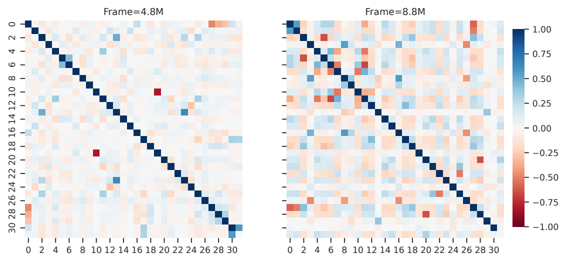


(h) DQN in Ice Hockey with Adam+WC

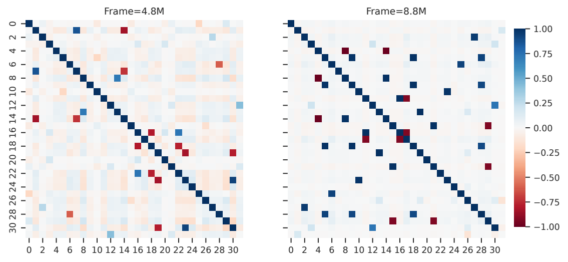
Figure 12: The gradient covariance heatmaps of training DQN in Asterix, Beam Rider, Demon Attack, and Ice Hockey, optimized by Adam and Adam+WC, respectively.



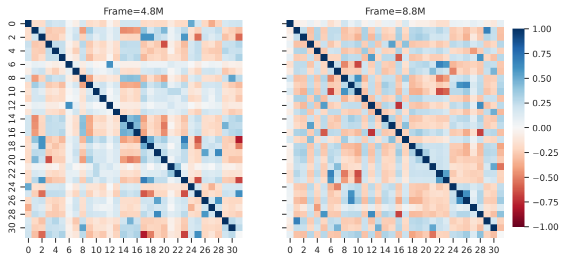
(a) DQN in James Bond with Adam



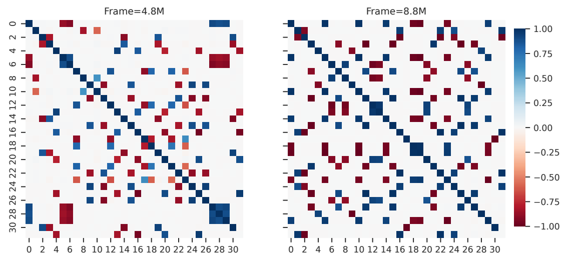
(b) DQN in James Bond with Adam+WC



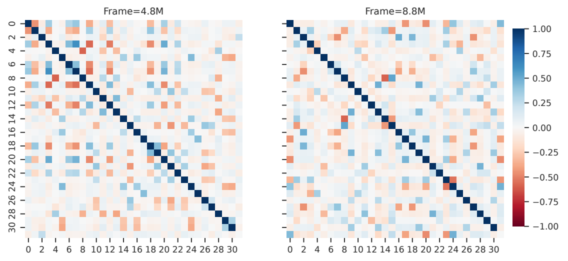
(c) DQN in Seaquest with Adam



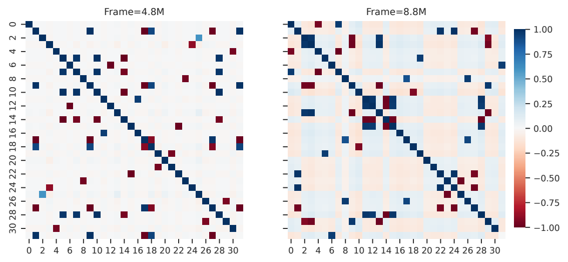
(d) DQN in Seaquest with Adam+WC



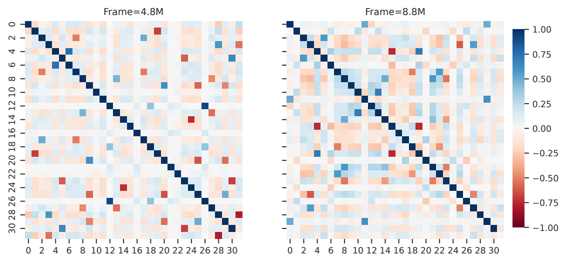
(e) DQN in Space Invaders with Adam



(f) DQN in Space Invaders with Adam+WC

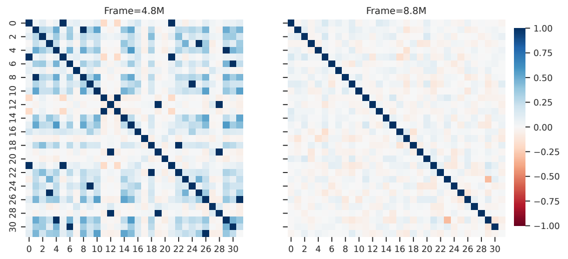


(g) DQN in Yars' Revenge with Adam

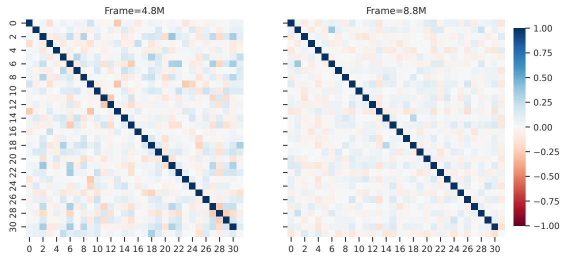


(h) DQN in Yars' Revenge with Adam+WC

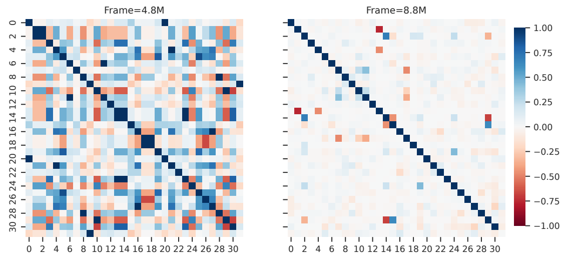
Figure 13: The gradient covariance heatmaps of training DQN in James Bond, Seaquest, Space Invaders, and Yars' Revenge, optimized by Adam and Adam+WC, respectively.



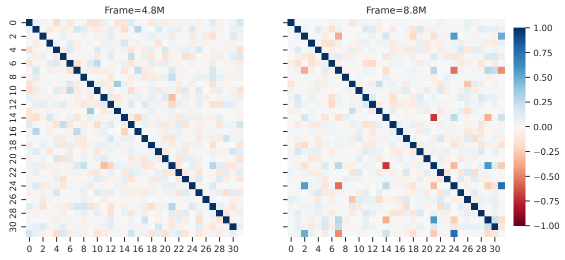
(a) Rainbow in Asterix with Adam



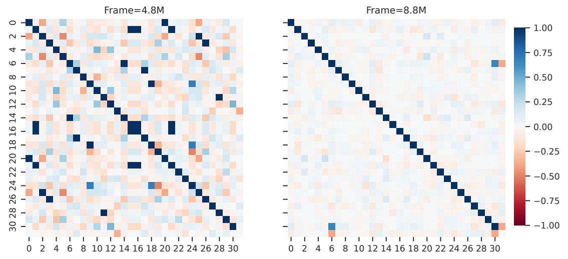
(b) Rainbow in Asterix with Adam+WC



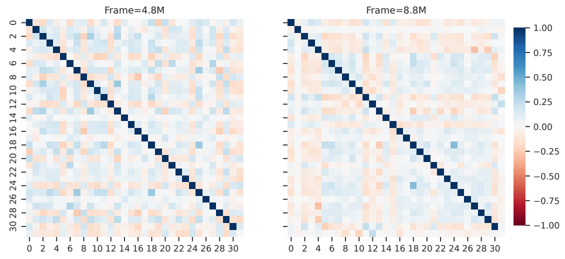
(c) Rainbow in Beam Rider with Adam



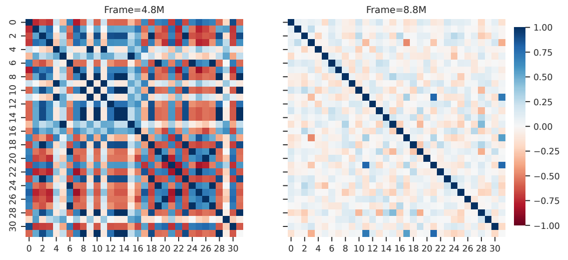
(d) Rainbow in Beam Rider with Adam+WC



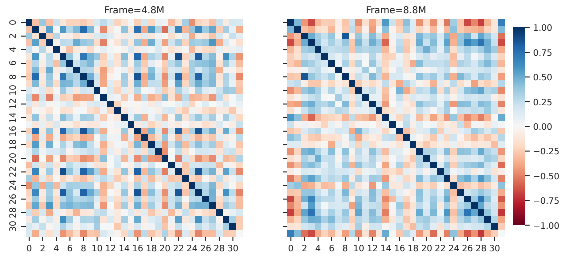
(e) Rainbow in Demon Attack with Adam



(f) Rainbow in Demon Attack with Adam+WC

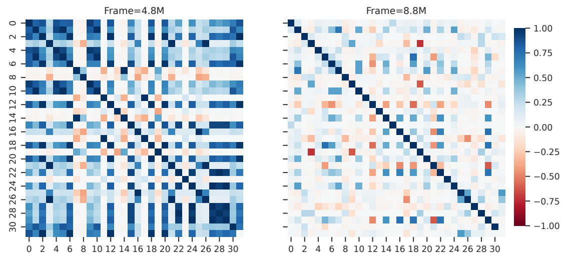


(g) Rainbow in Ice Hockey with Adam

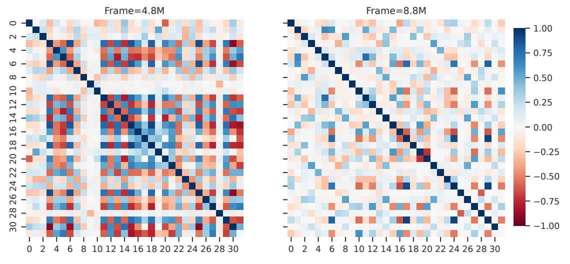


(h) Rainbow in Ice Hockey with Adam+WC

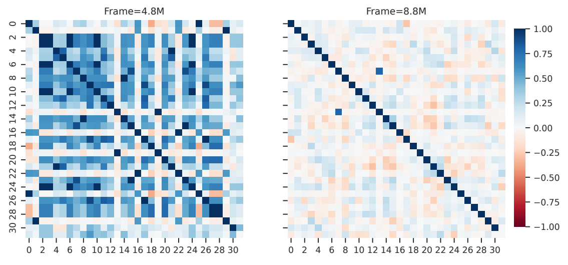
Figure 14: The gradient covariance heatmaps of training Rainbow in Asterix, Beam Rider, Demon Attack, and Ice Hockey, optimized by Adam and Adam+WC, respectively.



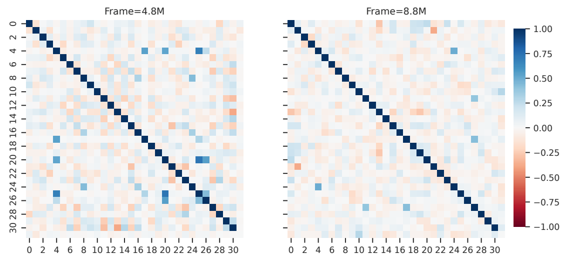
(a) Rainbow in James Bond with Adam



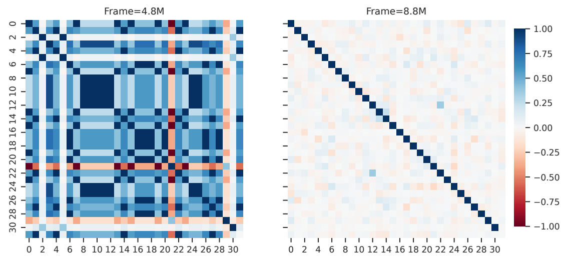
(b) Rainbow in James Bond with Adam+WC



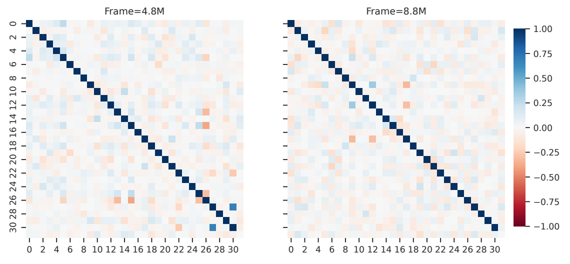
(c) Rainbow in Seaquest with Adam



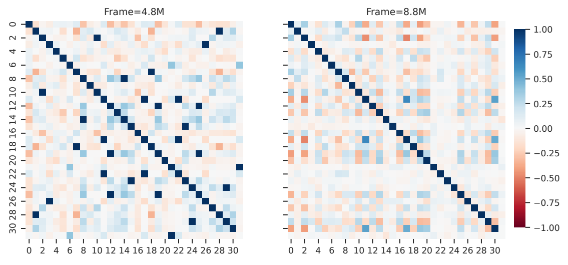
(d) Rainbow in Seaquest with Adam+WC



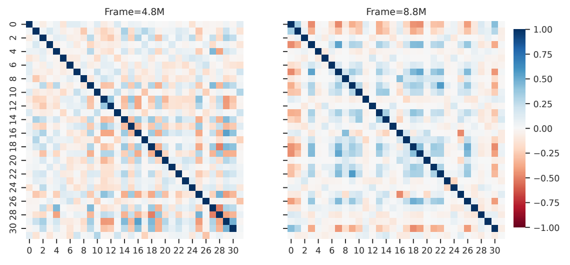
(e) Rainbow in Space Invaders with Adam



(f) Rainbow in Space Invaders with Adam+WC



(g) Rainbow in Yars' Revenge with Adam



(h) Rainbow in Yars' Revenge with Adam+WC

Figure 15: The gradient covariance heatmaps of training Rainbow in James Bond, Seaquest, Space Invaders, and Yars' Revenge, optimized by Adam and Adam+WC, respectively.