Understanding Plasticity in Neural Networks

Clare Lyle [1] **Zeyu Zheng** [1] **Evgenii Nikishin** [1]
Bernardo Avila Pires [1] **Razvan Pascanu** [1] **Will Dabney** [1]

Abstract

Plasticity, the ability of a neural network to quickly change its predictions in response to new information, is essential for the adaptability and robustness of deep reinforcement learning systems. Deep neural networks are known to lose plasticity over the course of training even in relatively simple learning problems, but the mechanisms driving this phenomenon are still poorly understood. This paper conducts a systematic empirical analysis into plasticity loss, with the goal of understanding the phenomenon mechanistically in order to guide the future development of targeted solutions. We find that loss of plasticity is deeply connected to changes in the curvature of the loss landscape, but that it often occurs in the absence of saturated units. Based on this insight, we identify a number of parameterization and optimization design choices which enable networks to better preserve plasticity over the course of training. We validate the utility of these findings on larger-scale RL benchmarks in the Arcade Learning Environment.

1. Introduction

It is a widely observed phenomenon that after training on a non-stationary objective, neural networks exhibit a reduced ability to solve new tasks (Lyle et al., 2021; Nikishin et al., 2022; Dohare et al., 2021). This loss of plasticity occurs most robustly when the relationship between inputs and prediction targets changes over time, and the network must learn to 'overwrite' its prior predictions (Lyle et al., 2021). While such scenarios are relatively rare in supervised learning, they are baked into the way that deep reinforcement learning (RL) agents are trained. Understanding how plasticity is lost, and whether this loss can be mitigated, is crucial if we wish to develop deep RL agents which can rise to the challenge of complex and constantly-changing environments. Existing methods to promote trainability act on a wide variety of potential mechanisms by which plasticity might be lost, including resetting of layers (Nikishin et al., 2022) and activation units (Dohare et al., 2021), and regularization of the features (Kumar et al., 2020; Lyle et al., 2021). While all of these works observe performance improvements, it is unlikely that they are all obtaining these improvements by the same mechanism. As a result, it is difficult to know how to improve on these interventions to further preserve plasticity.

This paper seeks to identify the mechanisms by which plasticity loss occurs. We begin with an analysis of two interpretable case studies, illustrating the mechanisms by which both adaptive optimizers and naive gradient descent can drive the loss of plasticity. Prior works have conjectured, implicitly or explicitly, that a variety of network properties might cause plasticity loss: we present a falsification framework inspired by the study of causally robust predictors of generalization (Dziugaite et al., 2020), and leverage this framework to show that loss of plasticity cannot be uniquely attributed to any of these properties. While difficult to characterize explicitly, we provide evidence that the curvature of the loss landscape induced by new tasks on trained parameters is a crucial factor determining a network's plasticity, particularly in value-based reinforcement learning algorithms.

We conclude by completing a broad empirical analysis of methods which aim to improve the ability of a network to navigate the loss landscape throughout training. We find that architectural choices which have been conjectured to smooth out the loss landscape, such as categorical output representations and normalization layers, provide the greatest improvements to plasticity, while methods which perturb the parameters or provide other forms of regularization tend to see less benefit. To test the generality of these findings, we apply the best-performing intervention, layer normalization, to a standard DQN architecture and obtain significant improvements in performance on the Arcade Learning Environment benchmark. We conclude that controlling the loss landscape sharpness and optimizer stability present highly promising avenues to improve the robustness and usability

[*]Equal contribution [1]Google DeepMind. Correspondence to: Clare Lyle <clarelyle@deepmind.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

of deep RL methods.

2. Background

It has long been observed that training a network first on one task and then a second will result in reduced performance on the first task (French, 1999). This phenomenon, known as catastrophic forgetting, has been widely studied by many works (Kirkpatrick et al., 2017; Lee et al., 2017; Kemker et al., 2018). This paper concerns itself with a different phenomenon: in certain situations, training a neural network on a series of distinct tasks can result in worse performance on later tasks than what would be obtained by training a randomly initialized network of the same architecture.

2.1. Preliminaries

Temporal difference learning. Plasticity loss naturally arises under non-stationarity; we will focus our analysis on temporal difference (TD) learning with neural networks, a setting known to induce significant non-stationarity. We assume the standard reinforcement learning problem of an agent interacting with an environment $\mathcal{M}$, with observation space $\mathcal{S}$, action space $\mathcal{A}$, reward R and discount factor γ, with the objective of maximizing cumulative discounted reward (Sutton & Barto, 2018). Networks trained via temporal difference learning receive as input sampled *transitions* from an agent's interaction with the environment, of the form $\tau_t = (s_{t-1}, a_t, r_t, s_t)$, where $s_{t-1}, s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$, and $r_t = R(s_t)$. We let θ' denote the *target parameters*; in practice, θ' is usually an outdated copy of θ from a previous iteration, but other choices include setting it to be equal to the current parameters, or using a moving average of past values. The network $f : \Theta \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is trained to minimize the temporal difference error

$$\ell(\theta, \tau_t) = \left\| f(\theta, s_{t-1}, a_t) - \square(r_t + \gamma f(\theta', s_t, a')) \right\|^2 \quad (1)$$

where $\square$ denotes a stop-gradient, $\gamma < 1$ is the discount factor, and a' is chosen based on the variant of TD learning used. Crucially, the regression target $r_t + \gamma f(\theta', s_t, a')$ depends on the parameters θ' and changes as learning progresses. This nonstationarity occurs even if the policy and input distribution are fixed, meaning that we can study the role of nonstationarity independent of the agent's exploration strategy. We will use the shorthand $\ell(\theta)$, with θ' implicit, for the expectation of this loss over some input distribution.

Loss landscape analysis. We will be particularly interested in the study of the structure of the loss landscape traversed by an optimization algorithm. We will leverage two principal quantities in this analysis: the Hessian of the network with respect to some loss function, and the gradient covariance. The Hessian of a network f at parameters θ with respect to some loss $\ell(\theta)$ is the matrix defined as

$$H_\ell(\theta) = \nabla^2_\theta \ell(\theta) \in \mathbb{R}^{d \times d} \quad (2)$$

where $d = |\theta|$ is the number of parameters. Of particular relevance to optimization is the eigenspectrum of the Hessian $\Lambda(H_\ell(\theta)) = (\lambda_1 \geq \cdots \geq \lambda_d)$. The maximal eigenvalue, λ_1, can be interpreted as measuring the sharpness of the loss landscape (Dinh et al., 2017), and the condition number λ_1/λ_d has significant implications for convergence of gradient descent optimization in deep neural networks (Gilmer et al., 2022).

We will also take interest in the covariance structure of the gradients of different data points in the input distribution, a property relevant to both optimization and generalization (Fort et al., 2019; Lyle et al., 2022). We will estimate this covariance structure by sampling k training points $\mathbf{x}_1, \ldots, \mathbf{x}_k$, and computing the matrix $C_k \in \mathbb{R}^{k \times k}$ defined entrywise as

$$C_k[i, j] = \frac{\langle \nabla_\theta \ell(\theta, \mathbf{x}_i), \nabla_\theta \ell(\theta, \mathbf{x}_j) \rangle}{\|\nabla_\theta \ell(\theta, \mathbf{x}_i)\| \|\nabla_\theta \ell(\theta, \mathbf{x}_j)\|} \ . \quad (3)$$

If the off-diagonal entries of C_k contain many negative values, this indicates interference between inputs, wherein the network cannot reduce its loss on one subset without increasing its loss on another. If the matrix C_k exhibits low rank (which, given a suitable ordering σ of the data points $\mathbf{x}_{\sigma(1)}, \ldots, \mathbf{x}_{\sigma(k)}$ will yield a block structure) then the gradients are approximately colinear, which can indicate either generalization when their dot product is positive, or interference when their dot product is negative.

2.2. Defining plasticity

Plasticity, broadly construed, refers to a neural network's ability to learn new things. The study of plasticity has concerned neuroscience for several decades (Mermillod et al., 2013; Abbott & Nelson, 2000), but has only recently emerged as a topic of interest in deep learning (Berariu et al., 2021; Ash & Adams, 2020; Delfosse et al., 2021). Classical notions of complexity from the computational learning theory literature (Vapnik, 1968; Bartlett & Mendelson, 2002) evaluate whether a hypothesis class contains functions that capture arbitrary patterns, but are agnostic to the ability of a particular search algorithm, such as gradient descent, to find them, making them unsuitable proxies for plasticity. A billion-parameter neural network architecture might have the *capacity* to represent a rich class of functions, but if all of its activation units are saturated then it cannot be trained by gradient descent to realize this capacity.

Studies of plasticity in both supervised and reinforcement learning have observed reduced generalization performance as a result of overfitting to limited data early in training (Ash & Adams, 2020; Berariu et al., 2021; Igl et al., 2021). Many

works have further identified an impaired ability to even reduce the learning objective on the training distribution (Dohare et al., 2021; Lyle et al., 2021; Nikishin et al., 2022). This work will leverage the formulation of Lyle et al. (2021), who define plasticity as the ability of a network to update its predictions in response to a wide array of possible learning signals on the input distribution it has been trained on. This formulation is applicable to learning problems which do not admit a straightforward train-test split, as is the case in many deep RL environments.

Concretely, we consider an optimization algorithm $\mathcal{O} : (\theta, \ell) \mapsto \theta^*$ which takes initial parameters $\theta \in \Theta$ and some objective function $\ell : \Theta \to \mathbb{R}$, and outputs a new set of parameters θ^*. The parameters θ^* need not be an optimum: $\mathcal{O}$ could, for example, run gradient descent for five steps. In order to measure the flexibility with which a network can update its predictions under this optimization algorithm, we consider a distribution over a set of loss functions $\mathcal{L}$ each defined by some learning objective. For example, consider a distribution over regression losses

$$\ell_{f,\mathbf{X}}(\theta) = \mathbb{E}_{x\sim\mathbf{X}}[(f(\theta, \mathbf{x}) - g_\omega(\mathbf{x}))^2] \tag{4}$$

where g_ω is induced by a random initialization ω of a neural network. In order to match the intuition that more adaptable networks should have greater plasticity, we set a baseline value b to be the loss obtained by some baseline function (e.g. if ℓ is a regression loss on some set of targets, we set b to be the variance of the targets), and then define plasticity to be the difference between the baseline and the expectation of the final loss obtained by this optimization process after starting from an initial parameter value θ_t and optimizing a sampled loss function ℓ subtracted from the baseline b.

$$\mathcal{P}(\theta_t) = b - \mathbb{E}_{\ell\sim\mathcal{L}}[\ell(\theta_t^*)] \text{ where } \theta_t^* = \mathcal{O}(\theta_t, \ell) \tag{5}$$

We then define the loss of plasticity over the course of a trajectory $(\theta_t)_{t=0}^N$ as the difference $\mathcal{P}(\theta_t) - \mathcal{P}(\theta_0)$. We note that this definition of plasticity loss is independent of the value of the baseline b, i.e. the difficulty of the probe task for the network, allowing us to measure the relative change in performance of checkpoints taken from a training trajectory.

3. Methodology and Motivating Questions

The following sections will present a series of experiments which tease apart different causal pathways by which plasticity loss occurs and evaluate the predictive power of a range of hypotheses concerning the root causes thereof. We now outline the experimental methodology and research questions underpinning this investigation.

3.1. Measuring plasticity

In order to determine whether a candidate intervention preserves plasticity, we must first set out a consistent standard by which we will measure plasticity. Given a suitably generic class of target functions inducing the losses ℓ, (4) characterizes the adaptability of the network to arbitrary new learning signals from the unknown set of possible future tasks. We therefore construct a distribution over regression targets which corresponds to this uniform prior over possible future update directions. A different distribution over future target functions might give different numerical results; however, we believe that a uniform distribution captures a more universal notion of plasticity.

In our empirical evaluations, we will set $\mathbf{X}$ to be the set of transitions gathered by an RL agent and stored in some replay buffer, and f to be a neural network architecture. Given some offset $a \in \mathbb{R}$, we will apply the transformation $g(x) = a + \sin(10^5 f(\mathbf{x}; \omega_0))$, with ω_0 sampled from the same distribution as θ_0, to construct a challenging prediction objective which measures the ability of the network to perturb its predictions in random directions sampled effectively uniformly over the input space. Because the mean prediction output by a deep RL network tends to evolve away from zero over time as the policy improves and the reward propagates through the value function, we will set a to be equal to the network's mean prediction in order not to bias the objective in favour of random initializations, which have mean much closer to zero. The optimizer $\mathcal{O}$ will be identical to that used by the network on its primary learning objective, and we found that running this optimizer for a budget of two thousand steps minimized iteration time while also providing enough opportunity for networks starting from random initializations to solve the task.

3.2. Environments

We construct a simple MDP analogue of image classification, i.e. the underlying transition dynamics are defined over a set of ten states and ten actions, and the reward and transition dynamics depend on whether or not the action taken by the agent is equal to the index of its corresponding state. We construct three variants of a block MDP whose state space is the discrete set $\{0, \dots, 9\}$ and whose observation space is given by either the CIFAR-10 or MNIST dataset.

True-label: each state s of the MDP produces an observation from that class in the underlying classification dataset. Given action a, the reward is the indicator function $\delta_{a=s}$. The MDP then randomly transitions to a new state.

Random-label: follows the same dynamics as the previous environment, but each image is assigned a random label in $\{0 \dots 9\}$, and the observation from an MDP state i is sampled from images with (randomized) label i.

Sparse-reward: exhibits the same observation mapping as *true-label*. The reward is equal to $\delta_{a=s=9}$. The MDP transitions to a random state if $a \neq s$ and otherwise to $s + 1$.

We design these environments to satisfy two principal desiderata: first, that they present visually interesting prediction challenges with varying degrees of reward smoothness and density, and second that they allow us to isolate non-stationarity due to policy and target network updates independent of a change in the state visitation distribution. In the true-label and random-label variants, the transition dynamics do not depend on the agent's action, whereas in the sparse environment the policy influences the state visitation distribution. The different reward functions allow us to compare tasks which are aligned with the network's inductive bias (in the true-label task) and those which are not (the random-label task).

3.3. Outline of experimental results

The experiments presented in Sections 4 and 5 aim to answer a fundamental question: **what happens when neural networks lose plasticity?** Section 4 constructs two experimental settings which illuminate phenomena driving two very different forms of plasticity loss. The first constructs a non-stationary learning problem that induces extreme forms of instability in adaptive optimizers. The second illustrates a bias in the dynamics of gradient descent which leads to a progressive sharpening of the loss landscape of not just the current task, but also new tasks.

Section 5 asks **what properties *cause* plasticity loss?** Disentangling cause from correlation is a notoriously difficult problem throughout science and economics. We put a number of quantities which have been conjectured to drive plasticity loss to the test, evaluating quantities such as weight norm, feature rank, and the number of dead units in the network on the tasks outlined in Section 3.2. We follow up the largely negative results of these experiments with a qualitative analysis of learning curves on the probe tasks described in Section 3.1 that emphasizes the critical role of the loss landscape in plasticity.

Section 6.2 addresses the question: **how can we mitigate plasticity loss?** It evaluates the effectiveness of a range of interventions on the network architecture and on the optimization protocol, focusing on methods known to increase the smoothness of the loss landscape, applying the same evaluation protocol as described in this section in order to measure plasticity across our classification MDP testbed.

4. Two Simple Studies on Plasticity

We begin with some interpretable examples of learning problems where plasticity loss occurs. These examples illustrate how the design of optimizers can interact with nonstationarity to destabilize training, and explore how the dynamics of gradient-based optimizers might affect more subtle properties of the loss landscape.

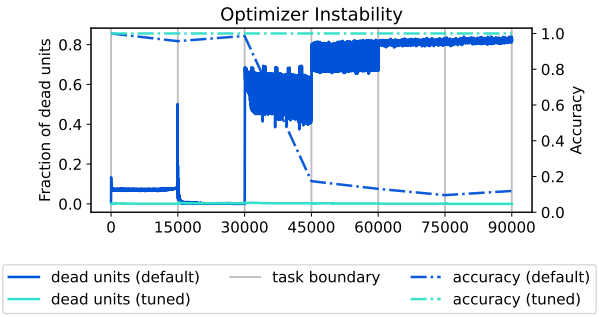

Figure 1. Abrupt task changes can drive instability in optimizers which depend on second-order moment estimates for adaptive learning rate scaling. Setting these estimators to be more robust to small gradient norms and to update moment estimates more quickly mitigates this issue.

4.1. Optimizer instability and non-stationarity

The robustness of existing optimizers across a wide range of datasets and network architectures has played a key role in the widespread adoption of deep learning methods. For example, the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 10^{-3} will often yield reasonable initial results on a range of network architectures from which the practitioner can iterate. However, when the assumptions on stationarity underlying the design of this optimizer no longer hold, the optimization process can experience catastrophic divergence, killing off most of the network's ReLU units. We can see an example of this in a simple non-stationary task in Figure 1. A two-hidden-layer fully-connected neural network is trained to memorize random labels of MNIST images (full details provided in Appendix A.1). After a fixed training budget, the labels are re-randomized, and the network continues training from its current parameters. This process quickly leads a default Adam optimizer to diverge, saturating most of its ReLU units and resulting in trivial performance on the task that a freshly

The mechanism of this phenomenon emerges when we consider the update rule for Adam, which tracks a second-order estimate $\hat{v}_t$ along with a first-order moment estimate $\hat{m}_t$ of the gradient via an exponential moving average

$$u_t = \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \bar{\epsilon}} + \epsilon}. \tag{6}$$

Gradients tend to have norm proportional to the training loss. When the loss changes suddenly, as is the case when the perfectly-memorized MNIST labels are re-randomized (or when the target network is updated in an RL agent), $\hat{m}_t$ and $\hat{v}_t$ will no longer be accurate estimates of their moment distributions. Under the default hyperparameters for deep supervised learning, $\hat{m}_t$ is updated more aggressively than $\hat{v}_t$, and so the updates immediately after a task change will

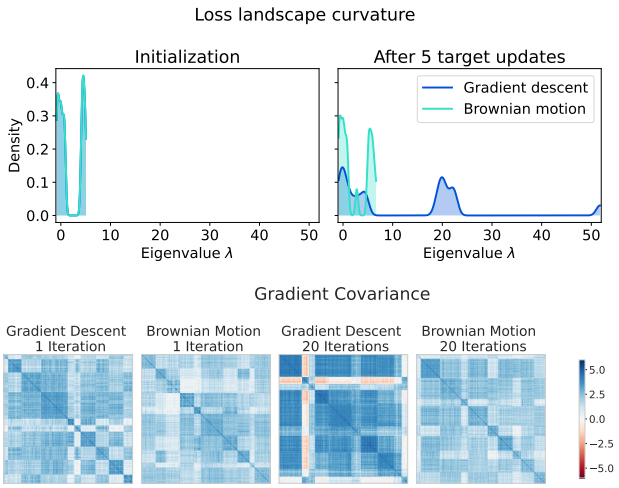

Figure 2. Evolution of the gradient and Hessian under gradient-based optimization compared to random perturbation of the parameters. Top: the density of the spectrum of the Hessian over different values of λ exhibits a larger outlier peak after gradient descent. Bottom: gradient descent induces more gradient interference between inputs and greater curvature of the loss landscape.

scale as a large number divided by a much smaller number, contributing to the instability we observe in Figure 1. In this instance, the solution is simple: we simply increase ϵ and set a more aggressive decay rate for the second-moment estimate, and the network avoids catastrophic instability. Intriguingly, a large value of ϵ is frequently used in deep RL algorithms such as DQN (Mnih et al., 2015) relative to the default provided by optimization libraries, suggesting that the community has implicitly converged towards optimizer hyperparameters which promote stability under nonstationarity. initialized network could solve perfectly.

4.2. Loss landscape evolution under non-stationarity

Even when optimization is sufficiently stable to avoid saturated units, prior work still observes reductions in network plasticity (Lyle et al., 2021). The causes of this phenomenon are more difficult to tease apart; neural network initializations have been tuned over several decades to maximize trainability, and many properties of the network change during optimization which could be driving the loss of plasticity. A natural question we can ask in RL is whether the optimization dynamics followed by a network bias the parameters to become less trainable, or whether the loss of plasticity is a natural consequence of any perturbation away from a carefully chosen initialization distribution.

We frame this question as a controlled experiment, in which we compare the evolution of two coupled updating procedures: one follows gradient-based optimization on a non-stationary objective (full details in Appendix A.1); the second follows a random walk, where we add a Gaussian perturbation to the parameters with norm equal to the size of the gradient-based optimizer update. Both trajectories start from the same set of randomly initialized parameters and apply updates of equal norm; the only difference is the direction each step takes. We evaluate how the structure of the local loss landscape with respect to a probe task evolves in both networks by comparing the Hessian eigenvalue distribution, and by comparing the covariance structure C_k of gradients on sampled inputs, with $k = 512$ equal to the batch size used for training. We compute the Hessian matrix for a regression loss towards a perturbation $\epsilon \sim \mathcal{N}(0, 1)$ of the network's current output, i.e. $\ell(\theta) = [f_\theta(\mathbf{X}) - \square f_\theta(\mathbf{X}) + \epsilon]^2$ where $\square$ indicates a stop-gradient, to obtain a proxy for how easily the network can update its predictions in arbitrary directions; we do not evaluate the Hessian or gradient structure of the primary learning objective as these will trivially differ between the trajectories.

We observe that the spectral norm of the Hessian of both processes increases over time; however, the outliers of the spectrum grow significantly faster in the network trained with gradient descent. Additionally, the network trained with gradient descent begins to exhibit negative interference between gradients, a phenomenon not observed in the Brownian motion. In other words, the inductive bias induced by gradient descent can push the parameters towards regions of the parameter space where the local loss landscape is less friendly to optimization towards arbitrary new objectives than what would be obtained by blindly perturbing randomly initialized parameters.

5. Explaining Plasticity Loss

While in some instances it is straightforward to deduce the cause of plasticity loss, most learning problems induce complex learning dynamics that make it difficult to determine root causes. This section will show that a number of plausible explanations of plasticity loss, including the rank of the network's features, the number of saturated units, the norm of its parameters, and the rank of the weight matrices, do not identify robust causal relationships. We provide some evidence supporting the hypothesis that plasticity loss arises due to changes in the network's loss landscape, and conclude with a discussion of the potential trade-offs that must be faced between preserving a trainable gradient structure and accurately predicting a value function.

5.1. Experimental setting

We train a set of DQN agents on each environment-observation space combination in the classification MDP set described in Section 3.2, and evaluate the ability of each

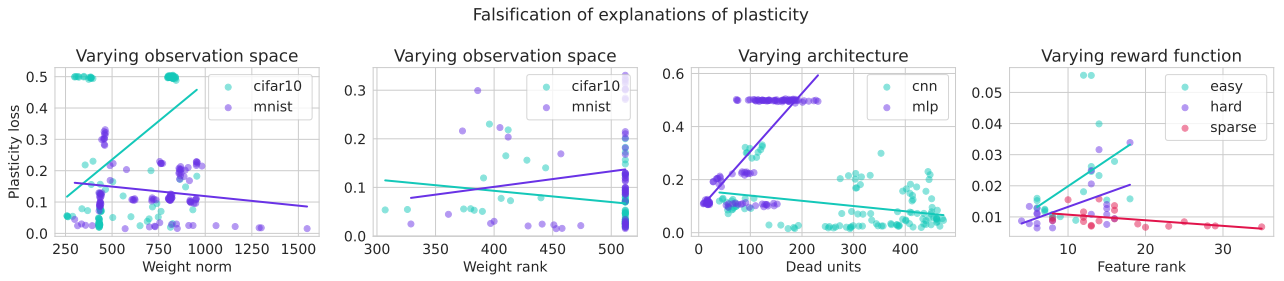

Figure 3. Results of our experimental falsification design: for any variable we consider, it is possible to construct a set of learning problems in which the variable exhibits either a positive or a negative correlation with plasticity. For example, weight norm and weight rank exhibit differing correlation signs depending on the observation space, while feature rank and sparsity depend on the reward structure of the environment.

network to fit a randomly generated set of target functions as described in Section 2.2 after a fixed number of training steps. In the experiments shown here, we run the DQN agents with a target network update period of 1,000 steps; as mentioned previously, the changing bootstrap target is the principal source of non-stationarity in the true-label and random-label tasks. Every 5000 steps, we pause training, and from a copy of the current parameters θ_t we train the network on a set of new regression problems to probe its plasticity. We log the loss at the end of 2,000 steps of optimization, sampling 10 different random functions, then resume training of the RL task from the saved parameters θ_t. We consider two network architectures: a fully-connected network (MLP) and a convolutional network architecture (CNN). Full details of the environments are included in Appendix A.2.

5.2. Falsification of prior hypotheses

Prior work has proposed a number of plausible explanations of why neural networks may exhibit reduced ability to fit new targets over time. Increased weight norm (Nikishin et al., 2022), low rank of the features or weights (Kumar et al., 2020; Gulcehre et al., 2022), and inactive features (Lyle et al., 2021; Dohare et al., 2021) have all been discussed as plausible mechanisms by which plasticity loss may occur. However, the explanatory power of these hypotheses has not been rigorously tested. While a correlation between a particular variable and plasticity loss can be useful for diagnosis, only a causal relationship indicates that intervening on that variable will necessarily increase plasticity.

This section will seek to answer whether the above candidate explanations capture causal pathways. Our analysis is based on a simple premise: that for a quantity to exhibit explanatory power over plasticity loss, it should exhibit a consistent correlation across different experimental interventions (Bühlmann, 2020). If, for example, parameter norm is positively correlated with plasticity in one observation space and negatively correlated in another, then it can be ruled out as a causal factor in plasticity loss. To construct this experiment, we train 128 DQN agents under a range of tasks, observation spaces, optimizers, and seeds. Over the course of training, we log several statistics of the parameters and activations, along with the plasticity of the parameters at each logging iteration.

In Figure 3, we show scatterplots illustrating the relationship between plasticity and each statistic, where each point in the scatterplot corresponds to a single training run. We see that for each of four quantities, there exists a learning problem where the quantity positively correlates with plasticity, and one in which it exhibits a negative correlation. In many learning problems the correlation between plasticity loss and the quantity of interest is nonexistent. In all cases we note that the correlation with plasticity is already quite weak; even so, the ability to reverse the sign of this correlation is a further mark against the utility of these simple statistics as causal explanations of plasticity. For example, we see a positive correlation between weight norm and plasticity loss in environments which use CIFAR-10 observations, but a slight negative correlation in environments which sample observations from MNIST. Analogous reversals happen for the other variable considered.

5.3. Loss landscape evolution during training

If the simple statistics we have considered thus far lack explanatory power, how should we characterize plasticity loss? One open question is whether the reduced ability to fit arbitrary new targets arises because the optimization process gets caught in local optima, or whether it arises due to overall slow or inconsistent optimization progress. To answer this question, we turn our attention towards the learning curves generated by networks as they train on the probe tasks. We study these learning curves primarily because they convey precisely the ease or difficulty of navigating the loss landscape. In particular, the learning curve tells

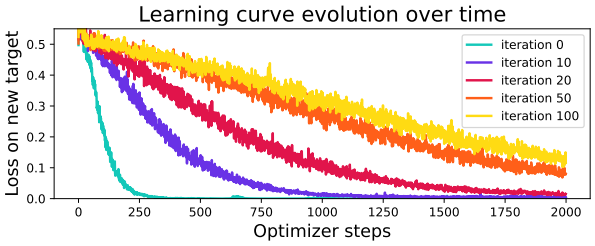

Figure 4. Plasticity loss corresponds to slower training progress, rather than higher plateaus, in the networks studied in this paper. We plot learning curves on a new target fitting task starting from network checkpoints at different points in training. This figure illustrates a CNN trained on the true-label MDP described in Section 6 with a CIFAR-10 observation space.

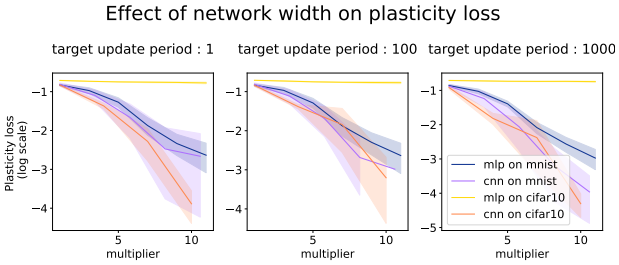

Figure 5. We observe a consistent decline in plasticity loss across different target update frequencies as a result of scaling in several architecture-dataset combinations; however, even when scaling the architecture to the point where it no longer fits on a single GPU, we are still unable to completely eliminate plasticity loss on these simple classification-inspired problems.

us whether optimization is getting trapped in bad minima (in which case the trajectory would hit an early plateau at a large loss value), or whether the network has greater difficulty reducing the loss enough to find a minimum in the first place (corresponding to a flatter slope).

We show in Figure 4 the learning curves obtained by an optimization trajectory from parameters θ_t on the probe task from different timesteps t of training on the RL task. We see that parameters from early training checkpoints quickly attain low losses, but that the slopes of these learning curves become more shallow as training progresses on the main task. Of particular note is the increasing variance of the curves: in the full-batch case, this non-monotonicity is associated with increasing loss landscape sharpness (Cohen et al., 2021). In the mini-batch optimization setting, we observed both increasing interference between minibatches as well as non-monotonicity in the loss even on the minibatch on which the gradient was computed. In short, we see that it is increasing difficulty of navigating the loss landscape that drives plasticity loss in this problem.

6. Solutions

Thus far, we have demonstrated that neural networks can lose plasticity even in a task as simple as classifying MNIST digits, assuming that a suitable non-stationarity is introduced into the optimization dynamics. We now turn our attention to means of reducing or reversing this loss of plasticity. Section 6.1 will evaluate the degree to which scaling alone can eliminate plasticity loss. Section 6.2 will evaluate the effects of a variety of interventions on plasticity. We test the applicability of these findings to larger scale tasks in Section 6.3.

6.1. The role of scaling on plasticity

Before considering sophisticated methods to address plasticity loss, we must first answer the question of whether this is simply a disease of small networks. In the context of the impressive successes of large models and the resultant scaling law phenomena (Kaplan et al., 2020), it is entirely plausible that plasticity loss, like many other challenges, vanishes in the limit of infinite computation. We find that while plasticity loss is easiest to induce in extreme forms in small networks, scaling a CNN to the limit of a single GPU's memory is insufficient to eliminate plasticity loss even in the simple classification tasks described in the previous section. We visualize the relationship between network width and plasticity loss in Figure 5.

These observations suggest that plasticity loss is unlikely to be the limiting factor for sufficiently large networks on sufficiently simple tasks, particularly when the network is sufficiently overparameterized as to smoothly interpolate its training data (Bubeck & Sellke, 2023). However, for tasks which do not align with the inductive bias of the network (as in the MLPs trained on CIFAR-10), or for which the network is not sufficiently expressive (as is the case for the small networks of any architecture), we see a reduction in the ability to fit new targets over time. Because we typically cannot guarantee a priori that a learning problem will fall in the first category, we turn our attention to other design choices which might further insure networks against plasticity loss.

6.2. Interventions in toy problems

In this section we evaluate the effect of a variety of interventions on plasticity loss. We repeat the protocol used in Section 5.1, training for 100 iterations of 1000 steps. We consider four architectures: a multi-layer perceptron (MLP), a convolutional neural network (CNN) without skip connections, a ResNet-18 (He et al., 2016), and a small transformer

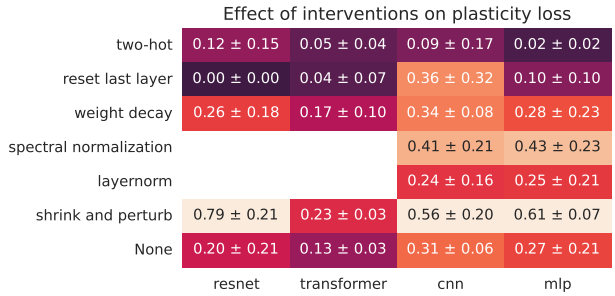

Figure 6. Effect of architectural and optimization interventions on plasticity loss. Colour indicates change in loss on challenge targets between initial and final epoch of training on RL task. Darker shading indicates less plasticity loss.

based on the Vision Transformer (ViT) architecture (Dosovitskiy et al., 2020).

We consider the following interventions: **resetting** the last layer of the network at each target network update, a simplified variant of the scheme proposed by Nikishin et al. (2022); adding **layer normalization** (Ba et al., 2016) after each convolutional and fully-connected layer of the CNN and the MLP; performing **Shrink and Perturb** (Ash & Adams, 2020): multiplying the network weights by a small scalar and adding a perturbation equal to the weights of a randomly initialized network each time the target network is updated; leveraging a **two-hot** encoding, which presents a distributional formulation of scalar regression wherein the network outputs a categorical probability distribution over fixed support and minimizes a cross-entropy loss with respect to an encoding of a regression target which distributes mass across two adjacent bins of the support; **spectral normalization** of the initial linear layer of the CNN and the MLP (Gogianu et al., 2021); and **weight decay,** setting the ℓ_2 penalty coefficient to 10^{-5}.

These methods were chosen to be representative samples of a number of approaches to mitigating plasticity loss: resetting the last layer temporarily removes a source of poor conditioning from the optimization process while likely not significantly influencing training dynamics elsewhere (Zhang et al., 2021a); layer normalization and residual connections tend to make networks more robust to optimizer choices; weight decay and spectral normalization both regularize the parameters of the network in different ways; shrink and perturb applies a perturbation to the current parameters without significantly changing the decision boundary (though we note that for regression tasks this will still influence the scale of the network outputs, and so may not be desirable).

We visualize our key takeaways in Figure 6, which compares plasticity loss after 100 iterations of training on each of the architecture-intervention combinations. Overall, selecting a network parameterization which smooths out the loss landscape is the most effective means of preserving plasticity of all approaches we have considered in this setting, and even has a greater effect on plasticity than resetting the final layer of the network in some instances. We visualize some learning curves of networks with and without layer normalization in Figure 17 in the supplementary material.

We note that while the two-hot encoding does demonstrate significant reductions in plasticity loss, it does so at the cost of stability of the learned policy in several instances we considered. Additionally, this intervention required significantly different optimizer hyperparameters from the regression parameterization, suggesting that while it can be a powerful tool to stabilize optimization, it might not be suitable as a plug-in solution to mitigate plasticity loss in an existing protocol.

6.3. Application to larger benchmarks

We now evaluate whether the benefits of layer normalization on plasticity in toy classification tasks translate to larger-scale benchmarks. We use the standard implementation of double DQN (Van Hasselt et al., 2016) provided by Quan & Ostrovski (2020), and evaluate three seeds on each of the 57 games in the Arcade Learning Environment benchmark (Bellemare et al., 2013). We use the RMSProp optimizer, ϵ-greedy exploration, and frame stacking (Mnih et al., 2015). Full implementation details can be found in Appendix A.3. The only difference between the baseline implementation and our modification is the incorporation of layer normalization after each hidden layer in the network.

We see in Figure 7 that the introduction of layer normalization robustly improves performance across the benchmark, without any additional hyper parameter tuning. While this improvement cannot be definitively attributed to a reduction in plasticity loss from the evidence provided, it points towards the regularization of the optimization landscape as a fruitful direction towards more robust RL agents. We further observe that many of the environments where layer normalization offers a significant boost to performance are those where the gradient covariance structure of the default architecture is degenerate or where the Hessian is ill-conditioned, and the LN networks which obtain performance improvements tend to have correspondingly better behaved gradient covariance. We provide a hint into this phenomenon in Figure 7, and defer the complete evaluation over all 57 games to Appendix B.3.

7. Related Work

Trainability: the problem of finding trainable neural network initializations is well-studied (Glorot & Bengio, 2010;

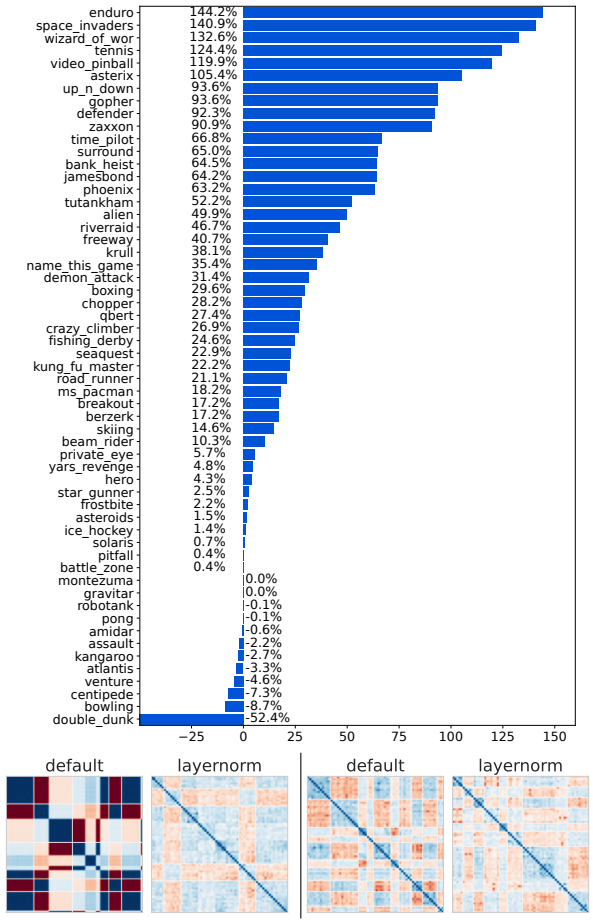

Figure 7. Layer normalization improves performance and changes the gradient covariance structure in DDQN agents. Top: Human-normalized improvement score (Wang et al., 2016) of adding layer normalization over the default double DQN agent. Bottom: Gradient covariance matrices for Freeway (left) and Kangroo (right). In environments where layer normalization significantly improves performance, it also induces weaker gradient correlation.

He et al., 2015; Sutskever et al., 2013). Without careful initialization and architecture design, vanishing and exploding gradients may arise (Yang & Schoenholz, 2017). ResNets (He et al., 2016) in particular are known to resolve many of these pathologies by biasing each layer's mapping towards the identity function, leading to better-behaved gradients (Balduzzi et al., 2017). Mean-field analysis (Yang & Schoenholz, 2017; Schoenholz et al., 2017; Yang et al., 2019), information propagation (Poole et al., 2016), and deep kernel shaping (Zhang et al., 2021b; Martens et al., 2021) have all been applied to study trainability in neural networks, and to characterize the role of residual connections. A wealth of prior work additionally studies the role of loss landscape smoothness in generalization and performance (Li et al., 2018; Santurkar et al., 2018; Ghorbani et al., 2019; Park & Kim, 2022). Other works highlight the chaotic behaviour of early training periods (Jastrzebski et al., 2020), in particular the 'edge of stability' phenomenon (Cohen et al., 2021) and the 'catapault mechanism' (Lewkowycz et al., 2020), and relate closely to the observations grounding 'linear mode connectivity' (Frankle et al., 2020) to explain generalization and trainability in deep neural networks; however, these approaches all focus on supervised learning with a stationary objective.

Continual learning encompasses a broad range of problem settings (Berariu et al., 2021; Hadsell et al., 2020; Rolnick et al., 2019) encompassing both input distribution shift and covariate shift. The connection between plasticity and non-stationarity was first observed in the case of the former (Ash & Adams, 2020); however this paper has focused on the latter as a driver of performance plateaus in RL. Most related to our study is the identification of the *loss* of plasticity as a potentially limiting factor in deep reinforcement learning (Lyle et al., 2021; Dohare et al., 2021). This study can be motivated by the rich literature studying the effect of resetting and distillation on performance (Fedus et al., 2020; Nikishin et al., 2022; Igl et al., 2021; Schmitt et al., 2018).

8. Conclusions

The findings of this paper highlight a stark contrast between recent observations on pretraining in large models, which find that suitable learning objectives can accelerate adaptation and improve generalization on later tasks, and the loss of plasticity in non-stationary prediction problems, where *un*suitable objectives can hurt a network's ability to adapt to new learning signals. However, as reinforcement learning algorithms scale up to more complex tasks, the divide between these regimes shrinks. While it is possible that in many settings, plasticity loss is not a limiting factor in network performance and so need not be a concern for many of the relatively small environments used to benchmark algorithms today, we conjecture that as the complexity of the tasks to which we apply RL grows, so will the importance of preserving plasticity.

The findings of this paper point towards stabilizing the loss landscape as a crucial step towards promoting plasticity. This approach is likely to have many ancillary benefits, presenting an exciting direction for future investigation. A smoother loss landscape is both easier to optimize and tends to exhibit better generalization, and it is an exciting direction for future work to better disentangle the complementary roles of memorization and generalization in plasticity.

Acknowledgements

We thank Mark Rowland, Tom Schaul, Georg Ostrovski, Hado van Hasselt, Diana Borsa, and Samuel L Smith for valuable feedback and discussions during the development of this work.

References

Abbott, L. F. and Nelson, S. B. Synaptic plasticity: taming the beast. *Nature neuroscience*, 3(11):1178–1183, 2000.

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

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Balduzzi, D., Frean, M., Leary, L., Lewis, J., Ma, K. W.-D., and McWilliams, B. The shattered gradients problem: If resnets are the answer, then what is the question? In *International Conference on Machine Learning*, pp. 342–350. PMLR, 2017.

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

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

Berariu, T., Czarnecki, W., De, S., Bornschein, J., Smith, S., Pascanu, R., and Clopath, C. A study on the plasticity of neural networks. *arXiv preprint arXiv:2106.00042*, 2021.

Bubeck, S. and Sellke, M. A universal law of robustness via isoperimetry. *Journal of the ACM*, 70(2):1–18, 2023.

Bühlmann, P. Invariance, causality and robustness. *Statistical Science*, 35(3):404–426, 2020.

Cohen, J. M., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

Delfosse, Q., Schramowski, P., Mundt, M., Molina, A., and Kersting, K. Adaptive rational activations to boost deep reinforcement learning. *arXiv preprint arXiv:2102.09407*, 2021.

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.

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

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Dziugaite, G. K., Drouin, A., Neal, B., Rajkumar, N., Caballero, E., Wang, L., Mitliagkas, I., and Roy, D. M. In search of robust measures of generalization. *Advances in Neural Information Processing Systems*, 33:11723–11733, 2020.

Fedus, W., Ghosh, D., Martin, J. D., Bellemare, M. G., Bengio, Y., and Larochelle, H. On catastrophic interference in atari 2600 games. *arXiv preprint arXiv:2002.12499*, 2020.

Fort, S., Nowak, P. K., and Narayanan, S. Stiffness: A new perspective on generalization in neural networks. *CoRR*, abs/1901.09491, 2019. URL `http://arxiv.org/abs/1901.09491`.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Gilmer, J., Ghorbani, B., Garg, A., Kudugunta, S. R., Neyshabur, B., Cardoze, D., Dahl, G. E., Nado, Z., and Firat, O. A loss curvature perspective on training instability in deep learning. In *ICLR*, 2022. URL `https://arxiv.org/abs/2110.04369`.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Gogianu, F., Berariu, T., Rosca, M. C., Clopath, C., Busoniu, L., and Pascanu, R. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, pp. 3734–3744. PMLR, 2021.

Gulcehre, C., Srinivasan, S., Sygnowski, J., Ostrovski, G., Farajtabar, M., Hoffman, M., Pascanu, R., and Doucet, A. An empirical study of implicit regularization in deep offline rl. *Transactions of Machine Learning Research*, 2022.

Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, 2020. ISSN 1364-6613.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Qun8fv4qSby.

Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho*, K., and Geras*, K. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1g87C4KwB.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kemker, R., McClure, M., Abitino, A., Hayes, T., and Kanan, C. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T. Overcoming catastrophic forgetting by incremental moment matching. *Advances in neural information processing systems*, 30, 2017.

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., and Gur-Ari, G. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Lyle, C., Rowland, M., and Dabney, W. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*, 2021.

Lyle, C., Rowland, M., Dabney, W., Kwiatkowska, M., and Gal, Y. Learning dynamics and generalization in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 14560–14581. PMLR, 2022.

Martens, J., Ballard, A., Desjardins, G., Swirszcz, G., Dalibard, V., Sohl-Dickstein, J., and Schoenholz, S. S. Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping. *arXiv preprint arXiv:2110.01765*, 2021.

Mermillod, M., Bugaiska, A., and Bonin, P. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

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

Park, N. and Kim, S. How do vision transformers work? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=D78Go4hVcxO.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/148510031349642de5ca0c544f31b2ef-Paper.pdf.

Quan, J. and Ostrovski, G. DQN Zoo: Reference implementations of DQN-based agents, 2020. URL http://github.com/deepmind/dqn_zoo.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf.

Schmitt, S., Hudson, J. J., Zidek, A., Osindero, S., Doersch, C., Czarnecki, W. M., Leibo, J. Z., Kuttler, H., Zisserman, A., Simonyan, K., et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=H1W1UN9gg.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Vapnik, V. On the uniform convergence of frequencies of occurrence of events to their probabilities. *Theory of Probability and its Applications*, 16, 1968.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Yang, G. and Schoenholz, S. Mean field residual networks: On the edge of chaos. *Advances in neural information processing systems*, 30, 2017.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. A mean field theory of batch normalization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SyMDXnCcF7.

Zhang, C., Bengio, S., and Singer, Y. Are all layers created equal? *Journal of Machine Learning Research*, 2021a.

Zhang, G., Botev, A., and Martens, J. Deep learning without shortcuts: Shaping the kernel with tailored rectifiers. In *International Conference on Learning Representations*, 2021b.

A. Experiment details

A.1. Case studies

Optimizer instability: we consider a memorization problem on the MNIST dataset (Deng, 2012), where the network is trained to classify its inputs according to randomly permuted labels of a subset of 5000 MNIST images. We use a fully-connected multi-layer perceptron (MLP) with two hidden layers of width 1024. At the end of each training iteration we log the accuracy on a sample of 4096 states, and re-randomize the labels, keeping the input set fixed. We train the network with an adam optimizer with learning rate equal to 0.001, first-order moment decay $b_1 = 0.9$, second-order moment decay $b_2 = 0.999$, $\varepsilon = 10^{-9}$, and $\bar{\varepsilon} = 0$. With the tuned optimizer, we se $b_2 = 0.9$ and $\bar{\varepsilon} = 10^{-3}$.

Brownian motion: we train the network via a Q-learning loss on batches of transitions generated by the easy classification MDP with the MNIST observation space. We use a stochastic gradient descent optimizer with learning rate 0.001. We use a batch size of 512.

We evaluate the gradient covariance matrix C_k by sampling a batch of transitions and computing the gradient of each transition individually. We then take the normalized dot product matrix and permute the rows and columns according to a k-means clustering, where we set $k = 10$ to match the number of latent states in the environment. We update the target network in the Q-learning objective once every 5000 steps.

To compute the Hessian eigenvalue density, we follow the implementation of Ghorbani et al. (2019) in order to obtain a Gaussian approximation to the eigenvalue distribution. We sample a single large batch of transitions, and use the Lanczos algorithm to obtain a set of centroids which are then convolved with a Gaussian distribution to obtain the final density.

A.2. Toy RL environments

We use the same agent structure for both the Atari and classification MDP environments. The agent collects data by interacting with the environment following an ϵ-greedy policy and stores states in a replay buffer. We then interleave interaction with the environment and optimization on sampled batches from the replay buffer. In the classification MDP, we train the network for 10,000 steps before updating the target network. We tried a variety of target network update frequencies and found that shorter update periods resulted in poor action-value estimation in the hard environment. We probe the ability of the network to fit a new set of regression targets once every 5000 optimizer steps: we draw 10 randomly sampled target functions generated by the procedure described in Section 2.2, and for each run the network's optimizer from the current parameters to minimize the loss with respect to these new targets for 2000 steps.

The architectures we consider in our plasticity evaluations are as follows:

- MLP: we use two hidden layers of varying width. For all evaluations other than the width sweep used to generate Figure 5, we use a width of 512. For the width sweep, we set a base width of 16 and then multiply by factors of 1, 2, 4, 8, 12, and 16.
- CNN: we use two convolutional layers followed by two fully-connected layers. The first convolutional layer uses 5x5 kernels, while the second uses 3x3; both have 64 channels. The fully-connected layers have widths of 256.
- ResNet: we use a standard resnet18 architecture (He et al., 2016).
- Vision Transformer: we use our own implementation based on (Dosovitskiy et al., 2020). We use a patch size of 3 to construct the convolutional embeddings, model dimension of 256 and a feedforward width of 1024. We use a single transformer block, and a dropout rate of 0.1. All components of the model are trained from scratch on the task.

A.3. Double DQN

We follow the standard training protocol on Atari, training for 200 million frames and performing optimizer updates once every 4 environment steps (Quan & Ostrovski, 2020). We add layer normalization after each hidden layer of the network. We use a replay buffer of size 100,000, and follow an ϵ-greedy policy during training with $\epsilon = 0.1$.

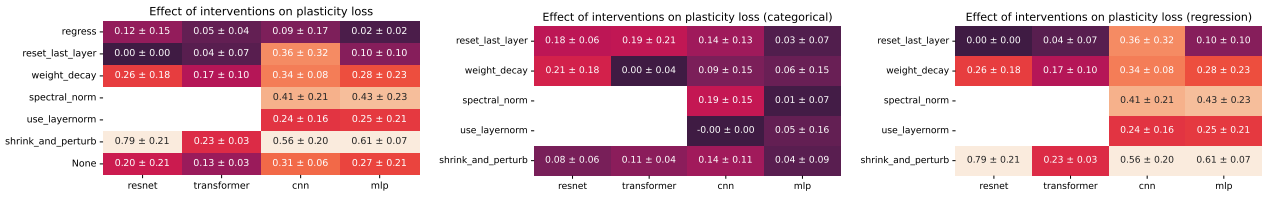

Figure 8. We repeat the analysis of Figure 6, but also include the effect of interventions on regression and categorical output encodings. We observe a significant benefit in the transformer, CNN, and MLP architectures from using the categorical encoding. This figure shows results for the CIFAR-10 dataset.

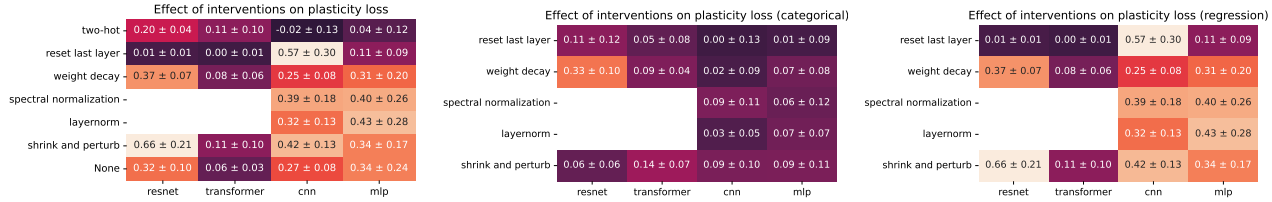

Figure 9. We repeat the analysis of Figure 6, but also include the effect of interventions on regression and categorical output encodings. We observe a significant benefit in the transformer, CNN, and MLP architectures from using the categorical encoding.

B. Additional analysis

B.1. Detailed intervention analysis

We provide a more detailed analysis of the effects of a variety of interventions on plasticity loss in different neural network architectures. Figures 8 and 9 show the *change* in the average final probe task loss after training on the toy RL environments for 1 million optimizer steps. We see that resetting the last layer, incorporating a two-hot output representation, and performing layer normalization have beneficial effects on plasticity, whereas shrink and perturb, weight decay, and resetting only the optimizer state do not improve plasticity.

In Figures 10 and 11 we show the effect of each intervention scheme on the initial and final losses. Some interventions improve trainability even from initialization, for example using a categorical output for the transformer, in addition to reducing the final probe task loss. The categorical representation appears to combine nicely with other interventions such as layer normalization, resetting, and shrink and perturb. The benefits of the categorical parameterization on shrink and perturb are expected, as regression targets are not output-scale-invariant in the same way that softmax logits are.

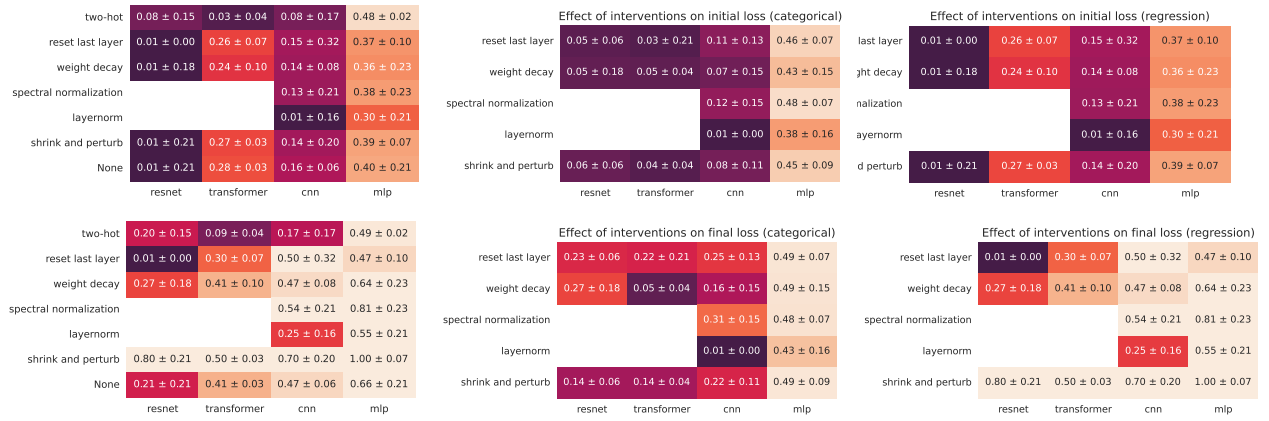

Figure 10. Initial and final loss evaluation on CIFAR-10 input distribution.

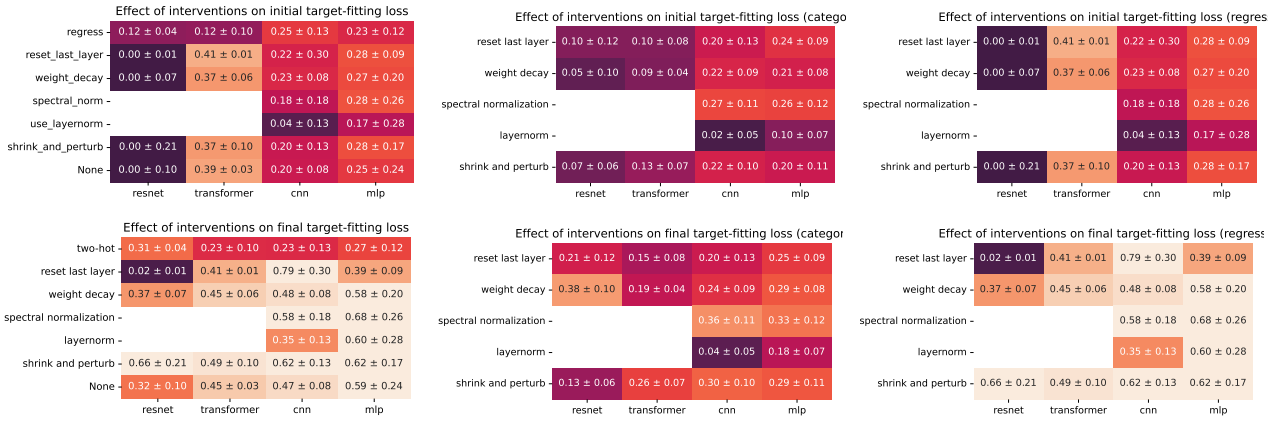

Figure 11. Initial and final loss evaluation on MNIST input distribution.

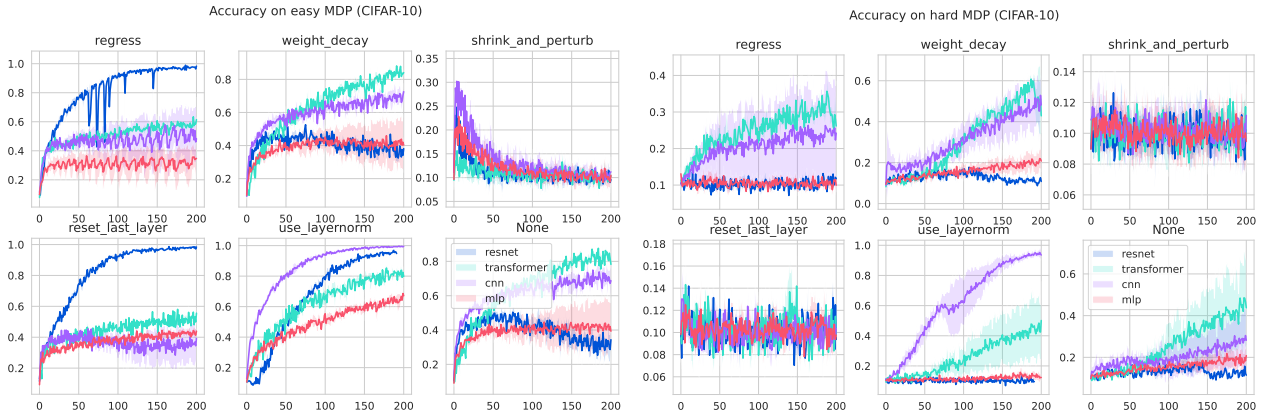

Figure 12. We visualize agent performance on the 'easy' and 'hard' tasks on MDPs using the CIFAR-10 observation space.

We additionally validate whether these interventions have the potential to interfere with learning on the primary task of interest in Figures 12 and 13. We observe that the methods which perturb the network weights often interfere with learning, particularly in the more challenging 'hard' reward structure that requires training for several tens of thousands of steps to make performance improvements. We note that we did not perform extensive hyperparameter tuning for each intervention, so it is possible that are more carefully tuned optimizer would produce better performance. Instead, these results should be taken holistically as an indicator of the robustness of an intervention to using a reasonable optimizer as was used to train the original network to which it is being applied.

Finally, we observe that in the convolutional architecture layer normalization often provides a significant stabilization effect on the sharpness of the loss landscape. We observe in the convolutional network trained via a regression loss that the added layer normalization prevents an increase in sharpness that occurs in the default architecture, as visualized in Figure 14.

B.2. Learning curves for classification MDPs

B.2.1. Training accuracy

Figure 16 and 15 provides a visualization of the learning curves of different networks during training in the classification MDP experiment. Notably, while performance on the easy and hard reward functions differs dramatically in most agents, the TD losses behave more similarly. This can be attributed to two properties: first, the TD loss changes less in an environment where the agent receives fewer rewards, so the poorer-performing policies induce easier learning problems. Second, the value functions of the two problems are very similar even though the optimal policies are very different. As a result, an 'accurate' value function with mean squared error of 0.1 can nonetheless correspond to a policy that does no better than

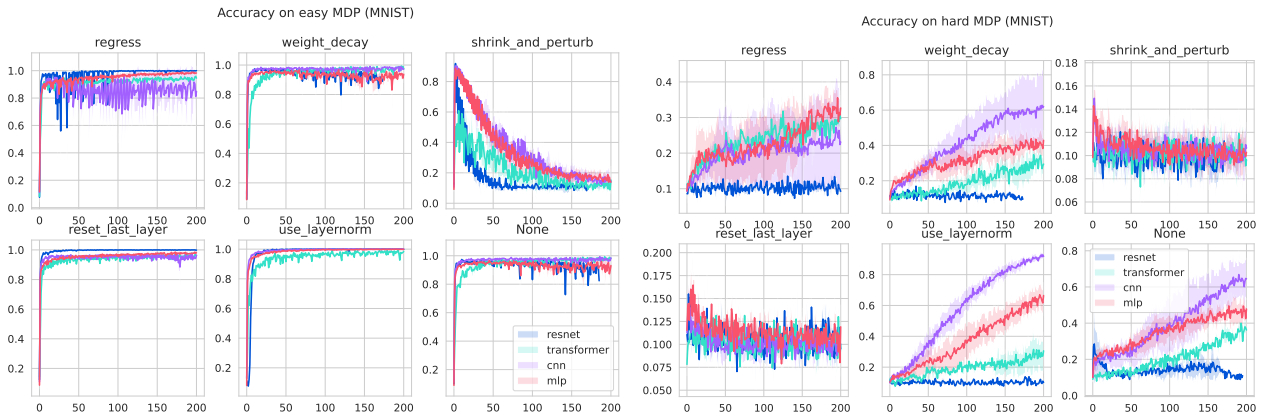

Figure 13. We visualize agent performance on the 'easy' and 'hard' tasks on MDPs using the MNIST observation space.

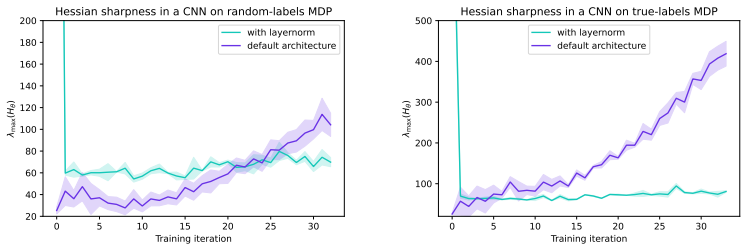

Figure 14. Layer normalization stabilizes the spectral norm of the Hessian in convolutional networks trained with regression on the classification MDP tasks. The network with larger output magnitudes (right) exhibits a greater smoothing effect from layer normalization.

random guessing. The temporal difference losses also give us an indication on whether the agent's predictions are diverging, identifying that layer normalization and the two-hot encodings enable significantly more stable learning, even though this does not always correspond to optimal policies.

B.2.2. Probe tasks

We include a visualization of the learning curves of some networks on the probe tasks to illustrate the subtlety of measuring plasticity loss in Figures 17 and 18.

B.3. Qualitative findings in DDQN

In addition to the gradient covariance analysis presented in the main body, we show more detailed learning curves and illustrate the evolution of the gradient covariance over time in Figures 20 and 19. We also visualize the spectrum of the network Hessian in Figures 21 and 22. We observe particularly intriguing trends in the gradient covariance heat maps shown in Figure 19. We note that the covariance structure of gradients varies significantly across environments, networks, and even random seeds. In many situations, gradients appear to be largely colinear, corresponding to significant interference (both positive and negative) between transitions in a minibatch.

One phenomenon we noticed in several environments was a tendency for network gradients to start off highly colinear, and then to become more independent later in training. In general, networks which stay in this colinear phase longer are also those whose learning curves struggle to 'take off'. Notably, in the game Freeway which is known to produce extremely high variance outcomes wherein agents either maximize the game score or fail to learn at all, we saw a one-to-one mapping between gradient degeneracy and learning progress. All random seeds where the agent made learning progress exhibited heavier weight on as opposed to off the diagonal, whereas the random seeds which did not ever improve preserved their initial degenerate gradient structure. A representative example can be observed in the first row of Figure 19. In general, networks with layer normalization exhibited a slight bias towards less degenerate gradients. However, it is not clear whether

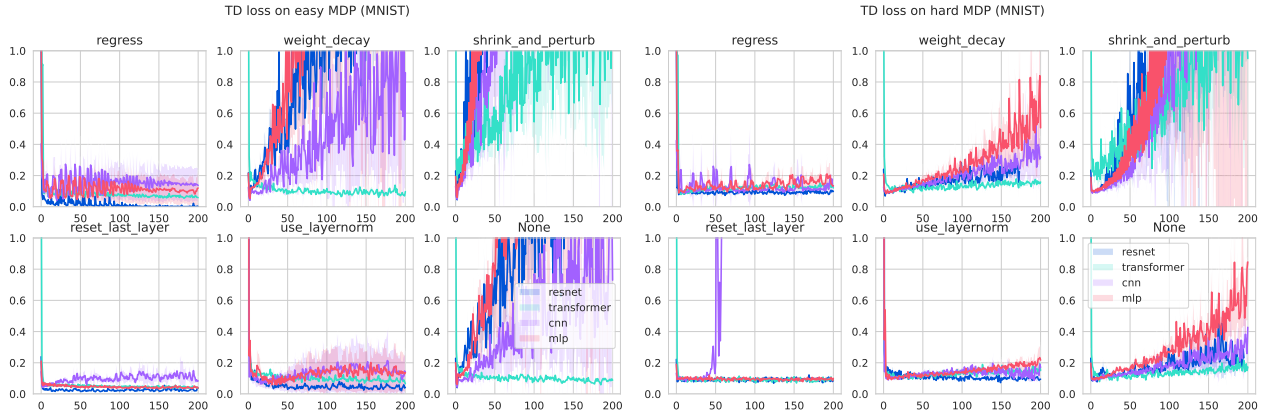

Figure 15. We visualize the TD loss obtained by agents trained on the 'easy' and 'hard' tasks on MDPs using the MNIST observation space.

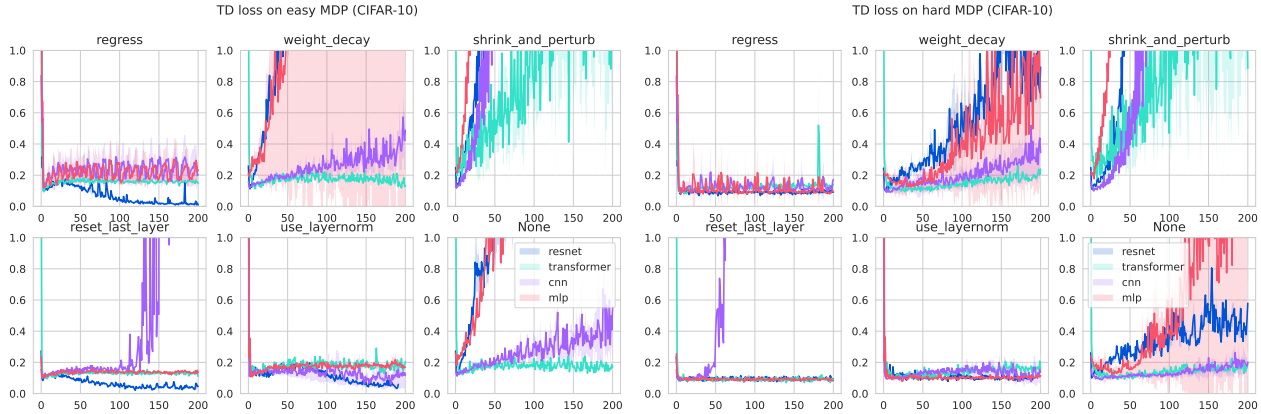

Figure 16. We visualize the TD loss obtained by agents trained on the 'easy' and 'hard' tasks on MDPs using the CIFAR-10 observation space.

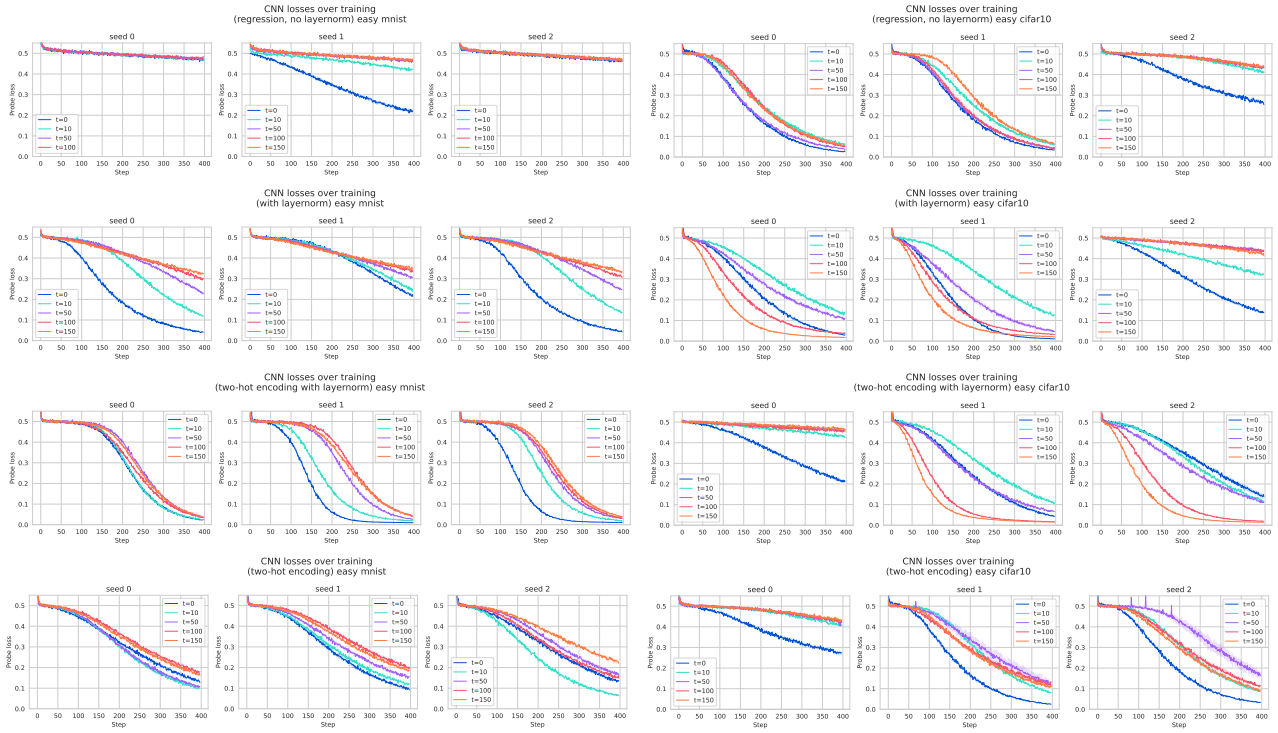

Figure 17. We see markedly different trajectories in networks trained with and without layernorm when tasked with a new optimization objective.

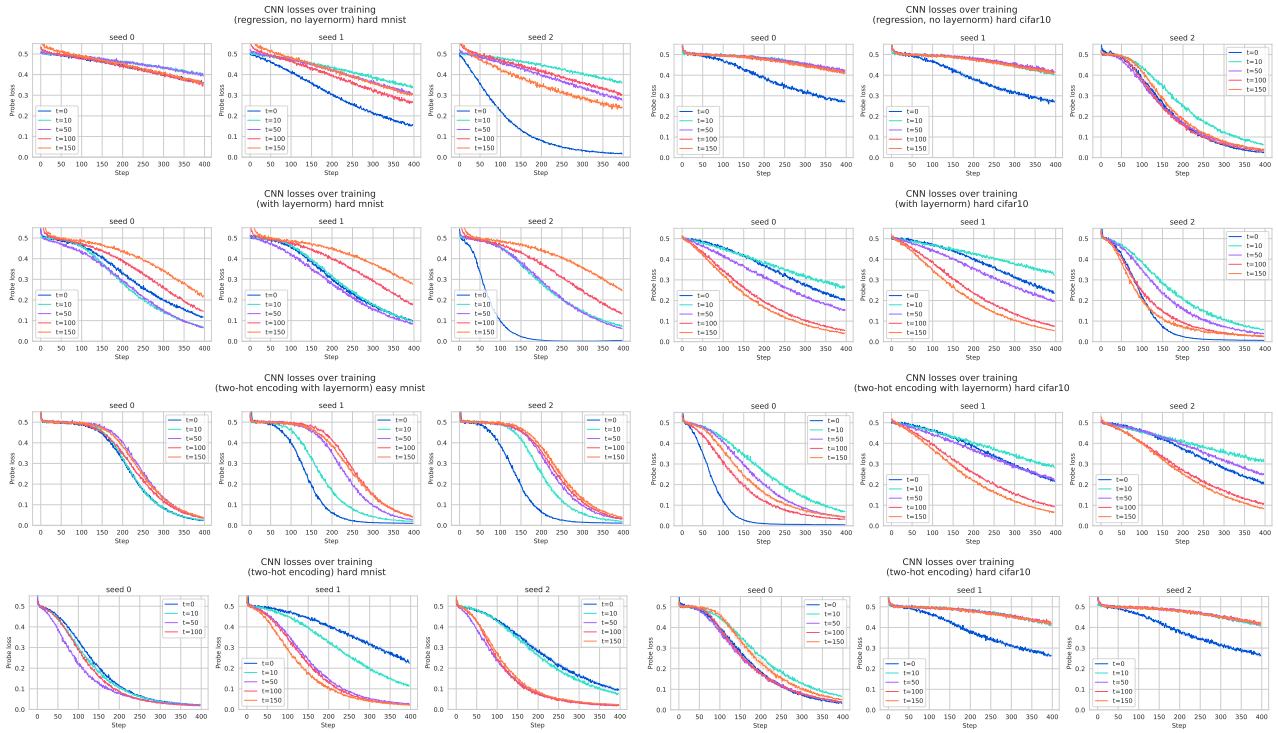

Figure 18. We see markedly different trajectories in networks trained with and without layernorm when tasked with a new optimization objective.

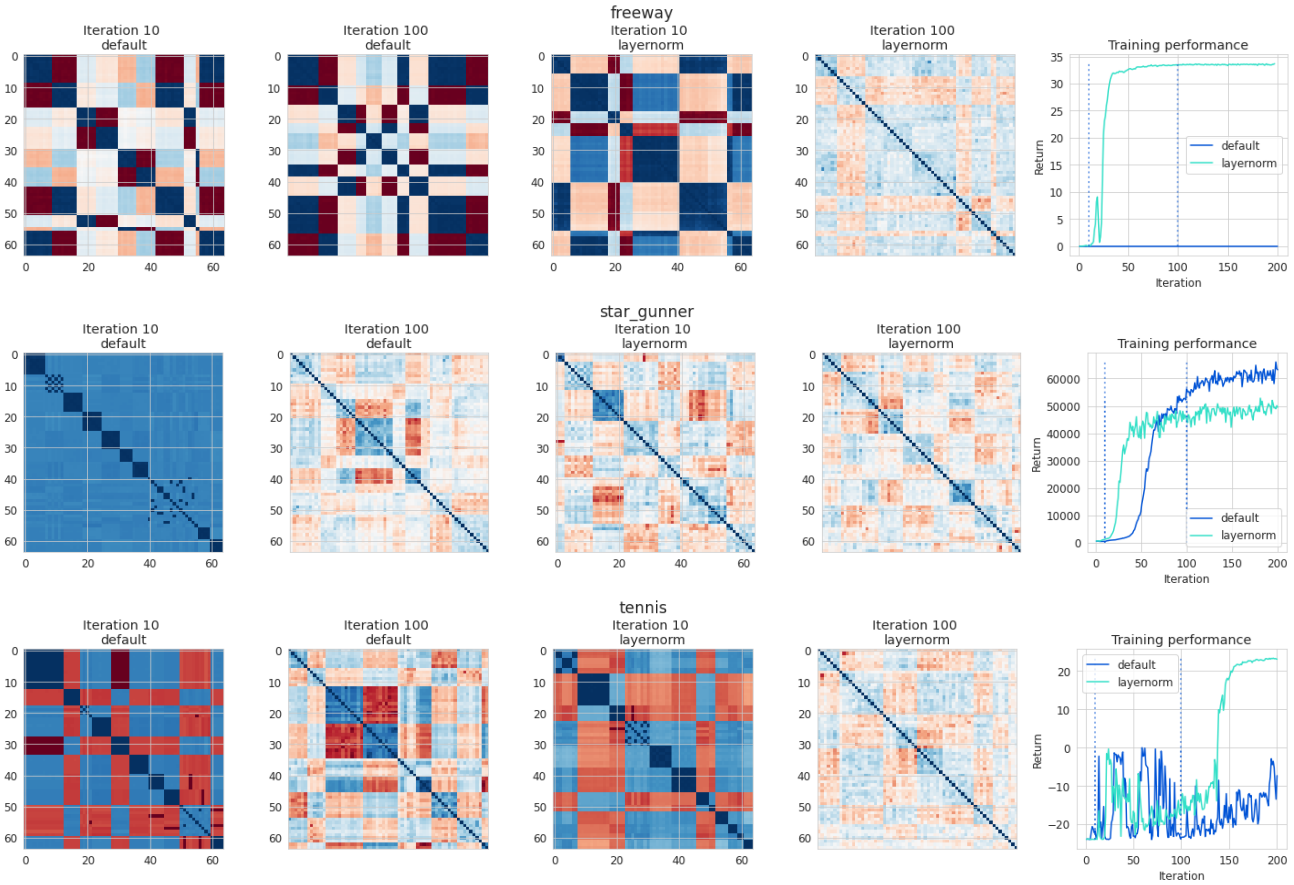

Figure 19. Gradient covariance plots vs performance for three sample games from the Arcade Learning Environment, which highlight the role of the gradient structure in learning progress. We find that when agents fail to learn, they tend to exhibit highly degenerate gradient structure, corresponding to the large off-diagonal values in the heatmaps visualized here.

gradient degeneracy is a symptom or a cause of performance plateaus. Further investigation into this phenomenon presents an exciting avenue for future work.

We did not observe any obvious correlations between the spectrum of the Hessian and the agent's performance, but include the computed spectra for a single seed of each game in Figures 21 and 22.

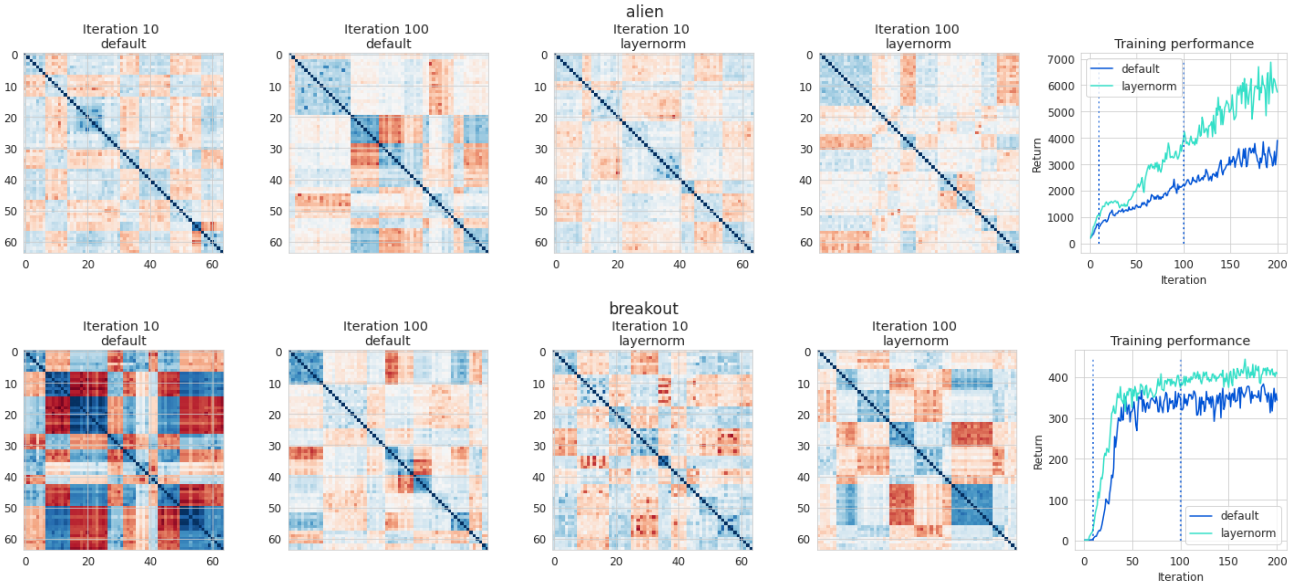

Figure 20. Gradient covariance plots vs performance for three sample games from the Arcade Learning Environment, which highlight the role of the gradient structure in learning progress. In games where agents make consistent positive progress increasing their returns, we see covariance structures with heavier weight on the diagonal.

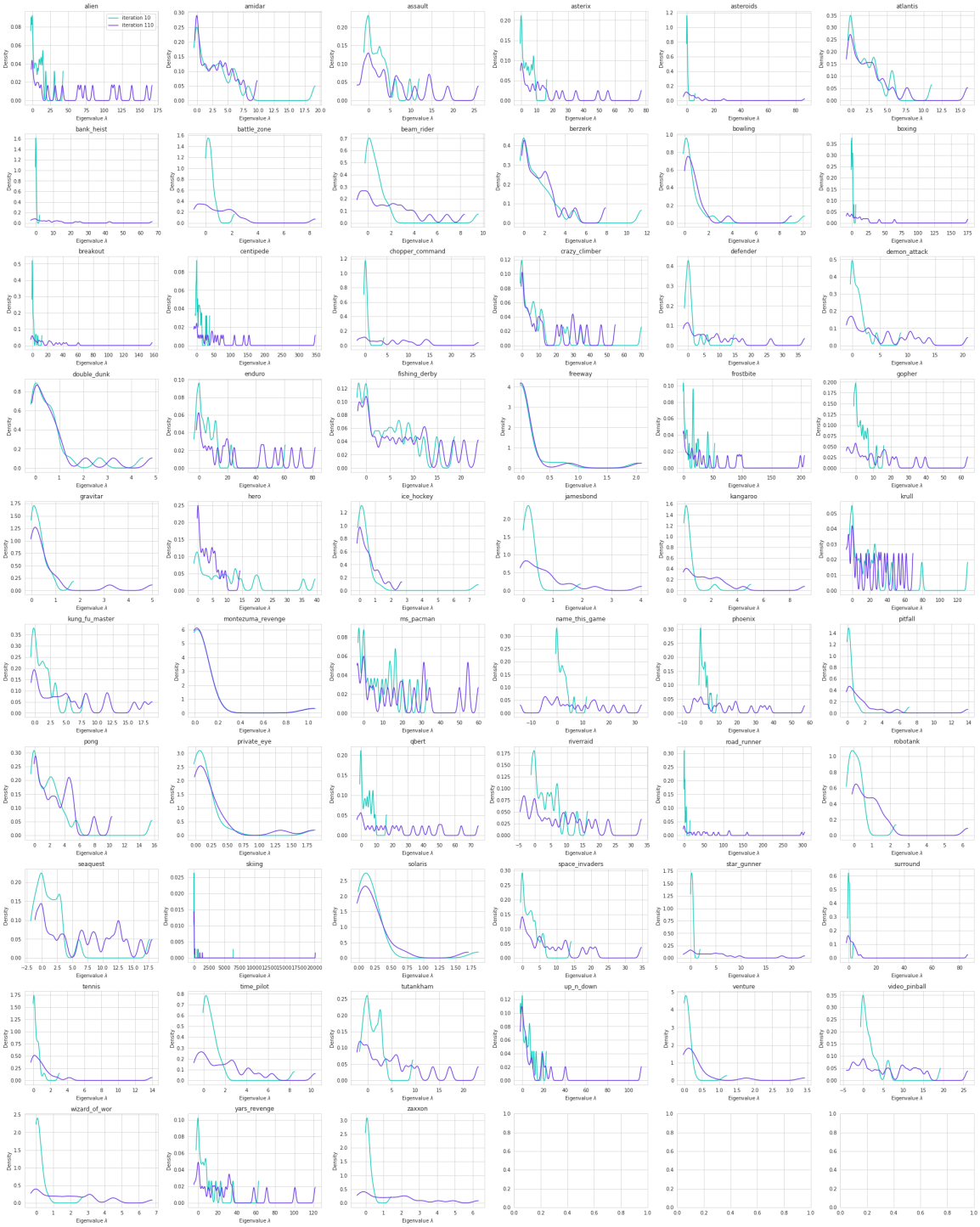

Figure 21. Visualization of Hessian approximations for a double DQN agent **without** layer normalization.

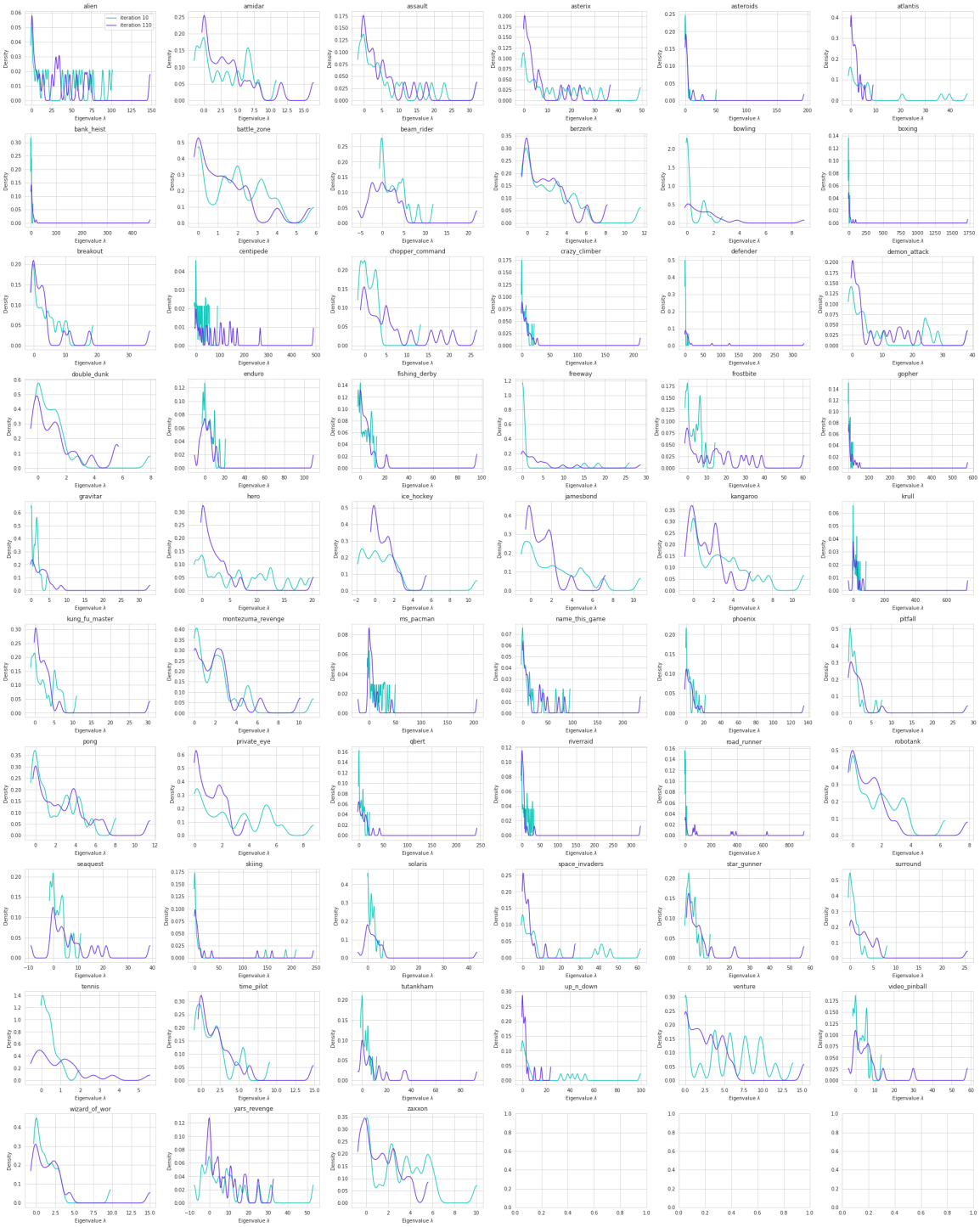

Figure 22. Visualization of Hessian approximations for a double DQN agent with layer normalization.