OPT2025: 17th Annual Workshop on Optimization for Machine Learning

How Does Layer Normalization Improve Deep Q-learning?

Braham Snyder BRAHAM.SNYDER@GMAIL.COM
Hadi Daneshmand DHADI@VIRGINIA.EDU
Chen-Yu Wei CHENYU.WEI@VIRGINIA.EDU
University of Virginia

Abstract

Layer normalization (LN) is among the most effective normalization schemes for deep Q-learning. However, its benefits remain not fully understood. We find *gradient interference* a promising lens through which to study these benefits. A gradient interference metric used in prior works is the inner product between semi-gradients of the temporal difference error on two random samples. We argue that, from the perspective of minimizing the loss, a more principled metric is to calculate the inner product between a semi-gradient and a full-gradient. We test this argument with offline deep Q-learning, without a target network, on four classic control tasks. Counterintuitively, we find empirically that first-order gradient interference metrics *positively* correlate with the training loss. We then find that a second-order gradient interference metric avoids this counterintuitive result, i.e. gives *negative* correlation. Theoretically, we provide supporting arguments from the linear regression setting.

1. Introduction

Deep Q-learning, including deep Q-networks (DQN) [32], is an important reinforcement learning (RL) method with wide applications including robotics [12], autonomous driving [37], and healthcare [47]. A key characteristic of DQN is that it is an off-policy algorithm that directly learns the optimal value function, which potentially improves sample efficiency and reduces the risks associated with on-policy exploration.

However, the benefit of off-policy learning often comes with optimization instabilities. DQN is prone to error propagation across iterations which could lead to performance drop or gradient explosion [23]. Many techniques have been developed to help stabilize DQN, including target networks [33, 35], replay buffers [22, 32], prioritized replay [38], and double Q-learning [41, 42].

One particularly simple yet effective stabilization technique is Layer Normalization (LN). Empirically, its benefits can rival — or even surpass — those of the RL-specific techniques above. For example, several works have empirically shown that LN can somewhat replace the benefits of a target network. Interestingly, related methods successful in supervised learning, such as batch normalization and weight normalization, do not typically provide similar benefits in RL (*cf.* [4]). This raises the question of why LN, in particular, has such a strong effect on DQN's performance without additional tricks. This discrepancy between LN and other normalization methods also extends to LLMs, where only LN has proven effective [39].

Many works have reported the empirical benefits of LN in RL, but few try to provide theoretical explanations. An exception is [10], which offers theoretical analysis and empirical backing for deep Q-learning with LN and ℓ_2 regularization. However, their theory uses ℓ_2 regularization, leaving

open the understanding of LN *by itself*. Motivated by this aspect, we revisit LN in DQN. We defer a precise comparison between our analysis and theirs to future work.

In this work, we look at three potential routes for explaining LN's benefit for deep Q-learning: gradient interference, isometry, and implicit learning rate decay. We find gradient interference, measurements of how gradients of different training samples interact, the most promising. We examine (i) Q-value gradient interference (QGI_+), (ii) semi-gradient interference (SGI), (iii) a (novel) mixed-gradient interference (MGI), and (iv) a (novel) second-order gradient interference (GI_2). As we explain more later, for all of these interference metrics except QGI_+, the standard view is that higher interference is better (only *negative* interference is bad). In other words, the standard view is that the more aligned the training gradients are between different samples, the better.

We test deep Q-learning without a target network, using SGD and Adam, with and without LN. Across four offline control tasks, we find that SGI and MGI *positively* correlate with training loss (and *negatively* correlate with returns) — whereas GI_2 *negatively* correlates with training loss (and *positively* correlates with returns). That is, we find the second-order gradient interference matches the standard view that higher is better, but we counterintuitively find that lower is better for first-order interference. Further, we find LN tends to improve all interference metrics from this correlational viewpoint. Not only that, we find that LN even tends to instantly improve GI_2 if added at any point throughout any training. We support our empirical results with a simple linear-regression argument showing that feature normalization improves an SGD progress–stability tradeoff.

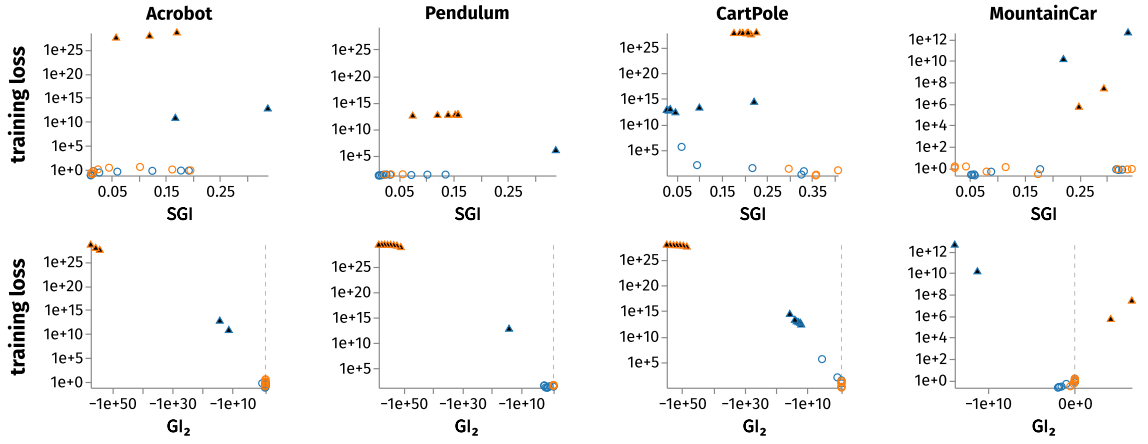

Figure 1: Deep Q-learning with LN ○ or no normalization ○, each at many learning rates. A triangle ▲ indicates at least one seed had its network weights clamped to avoid NaNs (Appendix L). **First row:** On four classic control tasks, we counterintuitively find that first-order gradient interference (SGI, defined later) *positively* correlates with the training loss, despite the standard interpretation of larger gradient interference as better. Moreover, we find LN tends to *decrease* first-order gradient interference, even though it *improves* the training loss. **Second row:** We find that including a second-order gradient interference term gives a metric (GI_2) with a more intuitive (i.e., negative) correlation with the loss. We likewise find that LN tends to increase this metric as it improves the loss and return. MountainCar largely does not follow these trends. **Both rows:** 30 seeds per data point. Learning rates as in Fig. 2. Trained with SGD. The same plots with return are shown in Appendix M.

2. Problem Setting

Q-networks. We study offline RL in continuous state spaces with discrete actions, where the state is denoted by $s \in \mathbb{R}^k$ and the action is denoted by $a \in \{1, \dots, m\}$. The reward function is $r(s, a) \in \mathbb{R}$. To approximate the Q-function, a standard single-hidden-layer neural network parameterizes the Q-function as $Q_\theta(s, a) = \langle v_a, (Ws)_+ \rangle$, where θ denotes the collection of parameters, $W \in \mathbb{R}^{d \times k}$, $v_a \in \mathbb{R}^d$, and $(x)_+ \in \mathbb{R}^d$ is the ReLU activation function. The parameters θ are updated via

$$\theta_{t+1} = \theta_t - \eta \widehat{\nabla} \ell_{\theta_t}(s_t, a_t, s'_t) \tag{1}$$

where $\eta > 0$ is the learning rate, $\gamma \in [0, 1)$ is the discount factor, (s_t, a_t, s'_t) is drawn from the offline distribution μ, and $\widehat{\nabla} \ell_{\theta_t}(s, a, s') \triangleq \big(Q_{\theta_t}(s, a) - r(s, a) - \gamma \max_{a'} Q_{\theta_t}(s', a')\big) \nabla Q_{\theta_t}(s, a)$ is the semi-gradient on the squared TD error of (s, a, s'). This method often suffers from instability, which is commonly mitigated by introducing a *target network* [33].

Layer Normalization. Throughout this paper, we focus on offline RL, but we speculate that our results apply to online RL with a replay buffer. For experimental and algorithmic simplicity, we use no target network at any point in this paper. We test four discrete-action classic control tasks [5], using an offline dataset of 10,000 uniformly random actions. All returns we report are averaged over all gradient steps over the course of training in order to measure both stability and speed. All our results are averaged over 30 random seeds unless otherwise noted. We primarily study deep Q-learning with layer normalization,

$$\mathrm{LN}(x) = \frac{x - \bar{x}}{\sqrt{\frac{1}{d} \sum_{i=1}^d (x_i - \bar{x})^2}}, \quad \bar{x} = \frac{1}{d} \sum_{i=1}^d x_i, \quad Q_\theta(s, a) = \langle v_a, (\mathrm{LN}(Ws))_+ \rangle. \tag{2}$$

While standard LN includes additional shift and scale parameters (an elementwise affine transform), in our experiments these had little effect on returns, so all of our results omit them. LN may also be placed after the ReLU rather than before, but we find the placement before about as effective or better. Our experiments further suggest, like prior work, that LN might roughly match or outperform other normalizations, such as batch normalization (BN) [14] and weight normalization (WN) [36]. Table 1 shows these results.

3. How Does Layer Normalization Improve Deep Q-learning?

Normalization layers in supervised learning have been explained from many perspectives, including feature orthogonalization [7, 30] and automatic learning rate tuning [1]. In this study, we ultimately aim for RL-specific explanations. We make progress on three: gradient interference (Section 3.1), isometry (Appendix H), and implicit learning rate decay (Appendix I).

We find gradient interference (Section 3.1) the most compelling view for explaining how LN improves deep Q-learning. Gradient interference metrics[1] measure how training gradients between different training samples interact.

3.1. First-Order Gradient Interference

To distinguish good interference and bad interference, we propose to quantify "how much the squared TD error decreases on another state-action if trained on one state-action." Let $\ell_\theta(s, a, s') =$

1. Throughout, we use "metric" in the colloquial sense, not in the sense of a distance function.

$(Q_\theta(s,a) - r(s,a) - \gamma \max_{a'} Q_\theta(s',a'))^2$. After training on (s_t, a_t, s'_t), the decrease of the squared TD error on (s,a,s') is

$$\begin{aligned} \ell_{\theta_{t+1}}(s,a,s') - \ell_{\theta_t}(s,a,s') &\approx \left\langle \theta_{t+1} - \theta_t, \nabla \ell_{\theta_t}(s,a,s') \right\rangle && \text{(first-order approximation)} \\ &= -\eta \langle \widehat{\nabla} \ell_{\theta_t}(s_t,a_t,s'_t), \nabla \ell_{\theta_t}(s,a,s') \rangle, && (3) \end{aligned}$$

where the last equality is by Eq. (1). The last expression can be turned into an interference metric that uses an inner product between a semi-gradient and a full-gradient. This is different from the interference metric used in prior work such as Lyle et al. [24], which uses an inner product between two semi-gradients, $\langle \widehat{\nabla} \ell_{\theta_t}(s_t,a_t,s'_t), \widehat{\nabla} \ell_{\theta_t}(s,a,s') \rangle$. We argue that the mixed "semi–full" version is more principled from the perspective of minimizing the loss (*cf.* Fujimoto et al. [9]), as the "semi" part reflects how the parameters are updated, and the "full" part, loosely put, reflects the loss we want to minimize. We call the semi-gradients-only metric ***semi-gradient interference*** (SGI), and the mixed-gradient metric ***mixed-gradient interference*** (MGI). With the cosine similarity $\mathsf{cos}(x,y) \triangleq \frac{\langle x,y \rangle}{\|x\|\|y\|}$,

$$\mathsf{SGI} = \mathbb{E}_{(s,a,s'),(s_t,a_t,s'_t)\sim\mu}\left[\mathsf{cos}\left(\widehat{\nabla} \ell_{\theta_t}(s_t,a_t,s'_t),\ \widehat{\nabla} \ell_{\theta_t}(s,a,s')\right)\right], \quad (4)$$

$$\mathsf{MGI} = \mathbb{E}_{(s,a,s'),(s_t,a_t,s'_t)\sim\mu}\left[\mathsf{cos}\left(\widehat{\nabla} \ell_{\theta_t}(s_t,a_t,s'_t),\ \nabla \ell_{\theta_t}(s,a,s')\right)\right]. \quad (5)$$

Lyle et al. [24] mention a standard interpretation of elements of SGI (before taking the expectation): negative components indicate that "the network cannot reduce its loss on one subset without increasing its loss on another," a sign of low plasticity or low trainability that is generally bad for performance. Conversely, positive elements can indicate better generalization, as reducing the loss on a state-action helps reducing that of another.

SGI and MGI *positively* correlate with training loss in our experiments. We show SGI in Fig. 1, and in row three of Fig. 2. We show MGI (whose results are qualitatively similar) in Appendix K. This is somewhat surprising: on one hand, this contrasts with the standard interference argument mentioned in [24], where positive gradient interference indicates generalization, a beneficial property. On the other hand, this aligns with the empirical findings of [24, 26] that gradient interference magnitudes (i.e. the absolute values) can negatively correlate with task performance for non-stationary settings.

This also aligns with the combined observations that LN is known to (i) empirically improve returns, and (ii) theoretically and empirically increase isometry (Appendix H). Higher isometry suggests to some extent that training is closer to tabular RL, with lessened interference and generalization.

An additional, speculative explanation for the correlation between gradient interference and loss is overshooting, where training updates move in the correct direction, but too far. This overshooting may cause instability [6, 15, 17, 27–29]. In any case, this finding that first-order in-distribution generalization *positively* correlates with the loss remains somewhat counterintuitive. In the following sections, we ultimately aim to more precisely explain this finding, and to come up with a more intuitive metric.

3.2. Second-Order Gradient Interference

If we perform a similar approximation as Eq. (3) on the loss decrement, but up to the *second order*, then we get two additional terms: a term containing the Hessian of the TD error, and a term containing the square of the first-order gradient interference (from Eq. (3)). See Appendix D for details. For

simplicity, we defer to future work the study of the (more complicated) Hessian term. We refer to the simplified (no Hessian term) second-order approximation of the gradient interference as GI_2:

$$\mathsf{GI}_2 = \eta\, \mathbb{E}_{(s,a,s'),(s_t,a_t,s_t')\sim\mu}\Big[\big\langle \nabla\!\!\!\nabla \ell_{\theta_t}(s_t,a_t,s_t'),\, \nabla \ell_{\theta_t}(s,a,s')\big\rangle\Big] - \frac{\eta^2}{4}\, \mathbb{E}_{(s,a,s'),(s_t,a_t,s_t')\sim\mu}\left[\frac{1}{\ell_{\theta_t}(s,a,s')}\, \big\langle \nabla\!\!\!\nabla \ell_{\theta_t}(s_t,a_t,s_t'),\, \nabla \ell_{\theta_t}(s,a,s')\big\rangle^2\right]. \tag{6}$$

The second-order term could dominate in our experiments, which might partly explain our counterintuitive first-order interference results. The second-order term might be particularly important when the learning rate is too large compared to the smoothness of the loss, causing the overshooting we previously discuss. This may align with claims that LN smooths the loss landscape [20].

GI_2 ***negatively*** **correlates with training loss in our experiments (Fig. 1; fourth row of Fig. 2).** This aligns with our hypothesis, though leaves unanswered the question of which difference from MGI is most important for negating the empirical correlation with the loss: (i) the second-order term; or (ii) the use of the dot product alone, rather than cosine similarity. That is, compared to MGI, the metric GI_2 not only (i) includes a $\langle \nabla\!\!\!\nabla \ell_{\theta_t}(s_t,a_t,s_t'), \nabla \ell_{\theta_t}(s,a,s')\rangle^2$ term; but also (ii) lacks the $(\|\nabla\!\!\!\nabla \ell_{\theta_t}(s_t,a_t,s_t')\|\|\nabla \ell_{\theta_t}(s,a,s')\|)^{-1}$ factor in its first-order term. However, we find empirically that the first-order dot-product term alone, which we call DGI (dot-product gradient interference), still correlates positively with the loss (which we show in Appendix K). This suggests the second-order term is indeed more important than the first-order term in the setting we study.

LN **increases** GI_2**.** We further find that, on three out of four enivornments, LN tends to increase GI_2 compared to no normalization (Fig. 1 and Fig. 2). Not only that, we find that adding LN immediately improves GI_2 even before updating the neural network weights (Fig. 3).

3.3. Provable Benefits of Normalization

As the empirical and theoretical analyses in Section 3.1 and Appendix E suggest, the squared interference dominates the stability of deep Q-learning, and it is related to the second-order approximation of the loss decrement. However, we have not understood *why* normalization stabilizes or accelerates training. In fact, even for the simpler case of supervised learning, we are not aware of a compelling theoretical explanation in the literature, though we suspect that our analysis is not likely to be novel.

In this subsection we provide some theoretical evidence on the effect of feature normalization in stochastic gradient descent (SGD) for *linear regression*, a special case of linear Q-learning, the simplest form of Q-learning with function approximation. The problem setting is formally defined in Definition 1. In particular, we show that feature normalization improves an SGD progress–stability tradeoff. This is offered only as supporting context for our second-order interference view, not as a claim of novelty.

Definition 1 (Stochastic gradient descent for linear regression) *Let $(x,y) \in \mathbb{R}^d \times \mathbb{R}$ be drawn from a fixed distribution μ and let the goal be to minimize $L(\theta) = \mathbb{E}_{(x,y)\sim\mu}[(x^\top\theta - y)^2]$. With SGD, at each iteration t, a sample $(x_t, y_t) \sim \mu$ is drawn and the parameter is updated as $\theta_{t+1} = \theta_t - \eta x_t(x_t^\top\theta_t - y_t)$.*

Assumption 1 (realizability) *There exists a $\theta^\star \in \mathbb{R}^d$ such that $y = x^\top\theta^\star$.*

We make this *realizability* assumption for simplicity. Although Assumption 1 assumes noiseless feedback, this is also only to simplify our exposition. It is straightforward to extend it to the case with noise scale $\mathbb{E}[(y - x^\top \theta^\star)^2|x] = \sigma^2(x)$ for some function $\sigma(x)$ increasing in $\|x\|$.

We consider two standard quantities to measure the progress of training. One is $R(\theta) = \|\theta - \theta^\star\|^2$, the squared distance to the optimal solution. The other is $L(\theta)$, the expected loss (Definition 1).

Under Assumption 1, the SGD update with learning rate η yields an expected distance decrement

$$\mathbb{E}[R(\theta_{t+1})|\theta_t] = R(\theta_t) - 2\eta A_t + \eta^2 B_t,$$

for some $A_t, B_t > 0$ that depends on data distribution μ and θ_t. They are the first-order and second-order terms in optimization, similar to Eq. (9). We can prove the following:

Theorem 2 *Under fixed $R(\theta_t)$, A_t^2/B_t is maximized when $\|x\|$ is a constant in the data distribution.*

Assuming $y = x^\top \theta$ means rescaling x necessarily changes y. As a result, Theorem 2 should be read as a stylized comparison across data distributions that isolates scale-sensitivity of the loss, not as an end-to-end model of layer normalization. There are two different ways to interpret Theorem 2:

- With the optimal choice of the learning rate η, the distance decrement $R(\theta_t) - \mathbb{E}[R(\theta_{t+1})|\theta_t]$ is maximized when $\|x\|$ are all equal. This is because $\max_\eta\{2\eta A_t - \eta^2 B_t\} = A_t^2/B_t$.
- Under a fixed first-order term ηA_t, the second-order term $\eta^2 B_t$ is minimized when $\|x\|$ are all equal. This is because the second-order term $\eta^2 B_t$ is equal to $F^2 B_t/A_t^2$ where $F = \eta A_t$ is the first-order term. Similarly, under a fixed second-order term, the first-order term is maximized when $\|x\|$ are all equal.

The first interpretation indicates that feature normalization (i.e., making $\|x\|$ all equal) can achieve the most decrement in distance to $\theta^\star$, while the second interpretation indicates that feature normalization always achieves the best possible trade-off between the two terms, under any learning rate.

For the other measurement $L(\theta)$, we can also prove a similar property, under an additional assumption that $\|x\|$ (scale) and $\frac{x}{\|x\|}$ (direction) are independent under the data distribution. Similarly, the following loss decrement equality holds for SGD:

$$\mathbb{E}[L(\theta_{t+1})|\theta_t] = L(\theta_t) - 2\eta C_t + \eta^2 D_t,$$

for some $C_t, D_t > 0$ that depends on μ and θ_t, and we have

Theorem 3 *Assume that $\|x\|$ and $\frac{x}{\|x\|}$ are independent under μ. Then under fixed $L(\theta_t)$, C_t^2/D_t is maximized when $\|x\|$ is a constant in the data distribution.*

We have the similar two ways to interpret Theorem 3 as in Theorem 2.

Extending Theorem 2 and Theorem 3 to temporal-difference learning requires several additional assumptions, which would make the analysis diverge further from practice. We therefore leave it as future work to seek a better perspective for understanding normalization in TD from the viewpoint of optimization theory. While linear regression is not the same as TD, it more closely resembles deep Q-learning with a target network, where the regression target changes slowly or remains fixed except for periodic updates. Our theory suggests that with a target network, normalization accelerates convergence in the regression problem defined by the target. Consequently, it could tolerate faster target network updates than the version without a target network. In the extreme, updating the target network at every step reduces to temporal-difference learning.

References

[1] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical Analysis of Auto Rate-Tuning by Batch Normalization. *arXiv*, December 2018. doi: 10.48550/arXiv.1812.03981.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[4] Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[6] William Dabney and Andrew Barto. Adaptive step-size for online temporal difference learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 872–878, 2012.

[7] Hadi Daneshmand, Amir Joudaki, and Francis Bach. Batch normalization orthogonalizes representations in deep random networks. *Advances in Neural Information Processing Systems*, 34:4896–4906, 2021.

[8] Mohamed Elsayed, Gautham Vasan, and A Rupam Mahmood. Streaming deep reinforcement learning finally works. *arXiv preprint arXiv:2410.14606*, 2024.

[9] Scott Fujimoto, David Meger, Doina Precup, Ofir Nachum, and Shixiang Shane Gu. Why should i trust you, bellman? the bellman error is a poor replacement for value error. *arXiv preprint arXiv:2201.12417*, 2022.

[10] Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[11] Florin Gogianu, Tudor Berariu, Mihaela C Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, pages 3734–3744. PMLR, 2021.

[12] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[13] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *International Conference on Learning Representations*, 2022.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[15] Khurram Javed, Arsalan Sharifnassab, and Richard S. Sutton. SwiftTD: A fast and robust algorithm for temporal difference learning. *Reinforcement Learning Journal*, 2:840–863, 2025.

[16] Amir Joudaki, Hadi Daneshmand, and Francis Bach. On the impact of activation and normalization in obtaining isometric embeddings at initialization. *Advances in Neural Information Processing Systems*, 36:39855–39875, 2023.

[17] Alexandra K Kearney. *Letting the Agent Take the Wheel: Principles for Constructive and Predictive Knowledge*. PhD thesis, University of Alberta, 2023.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Thomas Hofmann, Ming Zhou, and Klaus Neymeyr. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 806–815. PMLR, 2019.

[20] Hojoon Lee, Hanseul Cho, Hyunseung Kim, Daehoon Gwak, Joonkee Kim, Jaegul Choo, Se-Young Yun, and Chulhee Yun. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 36: 62270–62295, 2023.

[21] Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. *International Conference on Learning Representations*, 2025.

[22] Long-Ji Lin. *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.

[23] Ziyan Luo, Tianwei Ni, Pierre-Luc Bacon, Doina Precup, and Xujie Si. Understanding behavioral metric learning: A large-scale study on distracting reinforcement learning environments. In *Reinforcement Learning Conference*, 2024.

[24] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, pages 23190–23211. PMLR, 2023.

[25] Clare Lyle, Zeyu Zheng, Khimya Khetarpal, James Martens, Hado P van Hasselt, Razvan Pascanu, and Will Dabney. Normalization and effective learning rates in reinforcement learning. *Advances in Neural Information Processing Systems*, 37:106440–106473, 2024.

[26] Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the Causes of Plasticity Loss in Neural Networks.

In *Conference on Lifelong Learning Agents*, pages 750–783. PMLR, February 2025. URL https://proceedings.mlr.press/v274/lyle25a.html.

[27] Ashique Mahmood. Automatic step-size adaptation in incremental supervised learning. Master's thesis, University of Alberta, 2010.

[28] Ashique Rupam Mahmood, Richard S Sutton, Thomas Degris, and Patrick M Pilarski. Tuning-free step-size adaptation. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2121–2124. IEEE, 2012.

[29] Matthew McLeod, Chunlok Lo, Matthew Schlegel, Andrew Jacobsen, Raksha Kumaraswamy, Martha White, and Adam White. Continual auxiliary task learning. *Advances in neural information processing systems*, 34:12549–12562, 2021.

[30] Alexandru Meterez, Amir Joudaki, Francesco Orabona, Alexander Immer, Gunnar Ratsch, and Hadi Daneshmand. Towards training without depth limits: Batch normalization without gradient explosion. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xhCZD9hiiA.

[31] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[33] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236.

[34] Daniel Palenicek, Florian Vogt, Joe Watson, and Jan Peters. Scaling off-policy reinforcement learning with batch and weight normalization. *arXiv preprint arXiv:2502.07523*, 2025.

[35] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

[36] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[37] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*, 2017.

[38] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[39] Sheng Shen, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. Powernorm: Rethinking batch normalization in transformers. In *International conference on machine learning*, pages 8741–8751. PMLR, 2020.

[40] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[41] Hado Van Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[42] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[43] Christopher John Cornish Hellaby Watkins et al. Learning from delayed rewards. 1989.

[44] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[45] Chenjun Xiao, Bo Dai, Jincheng Mei, Oscar A Ramirez, Ramki Gummadi, Chris Harris, and Dale Schuurmans. Understanding and leveraging overparameterization in recursive value estimation. In *International Conference on Learning Representations*, 2021.

[46] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S Schoenholz. A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*, 2019.

[47] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Comput. Surv.*, 55(1), November 2021. ISSN 0360-0300. doi: 10.1145/3477600. URL https://doi.org/10.1145/3477600.

Appendix A. Limitations and Future Directions

One path for future theoretical work is to extend the linear regression theory to linear temporal difference learning, with the goal of proving the connection between gradient interference metrics and convergence speed. Another path is to connect our theory with [10]. Maybe the most important empirical directions are to experiment with a wider variety of environments, including image-based observations and continuous action spaces. Finally, both theoretically and empirically, an important direction is to consider the effects of model scaling.

Appendix B. Contrasting Returns of Various Normalizations

Table 1: Mean return of 20 seeds, each averaging over all gradient steps of 9 runs, one run for each of 9 learning rates. The highest return per environment is highlighted, as well as the return for environments whose 95% CI overlaps the CI of the highest mean. "None" means no normalization is used. Trained with SGD. On Pendulum, a random network outperforms all algorithms other than LN. Details in Appendix L.

	Random	None	LN	BN	WN
Acrobot	-452.6	-384.0	-267.2	-477.6	-409.6
Pendulum	-1281.9	-1323.2	-792.8	-1433.7	-1376.5
CartPole	19.5	163.5	289.0	19.0	270.0
MountainCar	-194.6	-163.8	-122.2	-169.0	-166.6

Appendix C. Gradient Interference Tabular Motivation and Per-Learning Rate Plot

Tabular Q-learning provably converges to $Q^\star$ under data coverage and properly tuned learning rates. In tabular Q-learning [43], only the Q-value of the sampled state-action is *directly* updated. In deep Q-learning, every training sample's gradient contributes not only to the direct update at that sample, but also *indirect* updates to other state-actions. Such indirect updates are how models *generalize* to unseen state-actions. While generalization is a common goal in machine learning, it can also introduce adverse side effects. For example, in classic examples of Q-learning with function approximation (Baird's and w-to-$2w$ examples [40]), the model diverges due to the undesired correlation between the Q-value of indirectly updated Q-values and those of directly updated Q-values. This becomes an issue in Q-learning but not in supervised learning as Q-learning is not based on gradient descent. We refer to such harmful generalization as "interference", which, as those classic works show, plays a more central role in RL than in supervised learning. We remark that although a lot of RL research studies out-of-distribution generalization (e.g., to new state distributions induced by new policies), our focus here is on in-distribution interference, which primarily affects the stability of the learning dynamics rather than generalization across distributions. To eliminate the confounding effect of out-of-distribution states, we generate training data in all our experiments from a broad state distribution. We focus on this in-distribution generalization. Below, we compare in more detail the updates of tabular Q-learning (which lacks generalization and interference, and provably converges under mild conditions) and deep Q-learning (with generalization and interference, and provably diverges in some simple cases). Recall the tabular Q-learning algorithm [43]:

$$Q_{t+1}(s,a) = Q_t(s,a) - \eta \Delta_t(s_t, a_t, s'_t) \mathbb{I}[(s,a) = (s_t, a_t)], \tag{7}$$

where $\Delta_t(s,a,s') = Q_t(s,a) - r(s,a) - \gamma \max_{a'} Q_t(s',a')$ is the (unsquared) TD error of (s,a,s'), and $\mathbb{I}[\cdot]$ is the indicator function. In deep Q-learning, we have by first-order approximation and Eq. (1):

$$\begin{aligned} Q_{\theta_{t+1}}(s,a) &\approx Q_{\theta_t}(s,a) + \langle \theta_{t+1} - \theta_t, \nabla Q_{\theta_t}(s,a) \rangle \\ &= Q_{\theta_t}(s,a) - \eta \left\langle \nabla \ell_{\theta_t}(s_t, a_t, s'_t), \nabla Q_{\theta_t}(s,a) \right\rangle \\ &= Q_{\theta_t}(s,a) - \eta \Delta_{\theta_t}(s_t, a_t, s'_t) \left\langle \nabla Q_{\theta_t}(s_t, a_t), \nabla Q_{\theta_t}(s,a) \right\rangle . \end{aligned} \tag{8}$$

Comparing tabular Q-learning (Eq. (7)) and deep Q-learning (Eq. (8)), we observe that for the directly updated state-action (s_t, a_t), the changes of its Q-value in the two algorithms are related

by $Q_{\theta_{t+1}}(s_t, a_t) - Q_{\theta_t}(s_t, a_t) = (Q_{t+1}(s_t, a_t) - Q_t(s_t, a_t)) \, \|\nabla Q_{\theta_t}(s_t, a_t)\|^2$. We also observe for other state-actions $(s, a) \neq (s_t, a_t)$, tabular Q-learning keeps $Q(s, a)$ unchanged, while deep Q-learning modifies $Q_{\theta_{t+1}}(s, a) = Q_{\theta_t}(s, a) - \eta \Delta_t(s_t, a_t, s'_t) \, \langle \nabla Q_{\theta_t}(s_t, a_t), \nabla Q_{\theta_t}(s, a) \rangle$ via indirect updates.

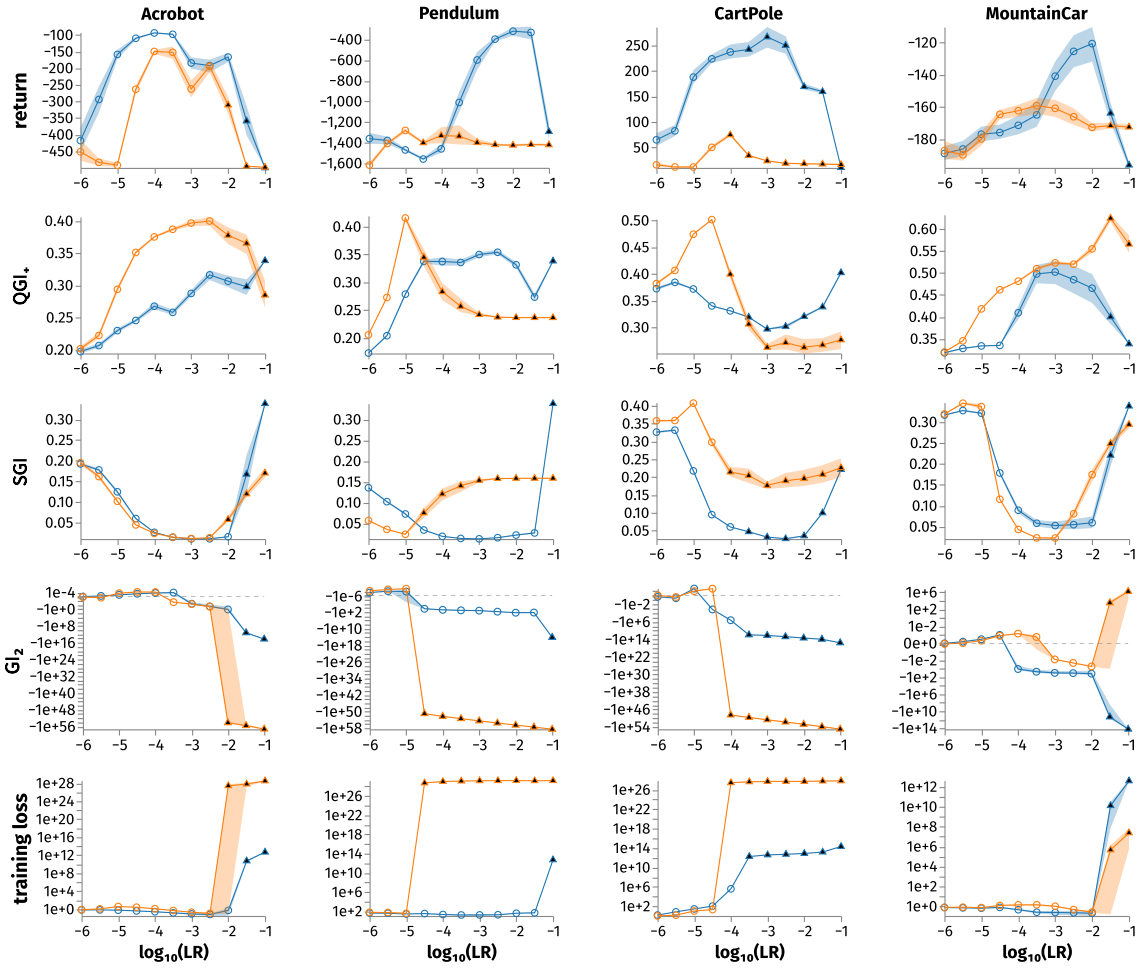

Figure 2: Every data point averages its metric (return, QGI_+, SGI, GI_2, or training loss) over 30 seeds, each seed averaging over all gradient steps of a training run. Comparing LN vs. no normalization, QGI_+ and (more so) SGI positively correlate with the loss. GI_2 negatively correlates with the loss. The QGI_+ correlations tend to break once SGD has diverged so far that its weights are clamped. Yet, for SGI and GI_2, their correlations tend to hold at any given learning rate (LR). MountainCar is an outlier for the correlation of GI_2. Shaded areas are 95% bootstrap CIs. These results use SGD. Adam results in Appendix K.

Appendix D. Full derivation of GI_2

If we perform similar approximation as Eq. (3) on the loss decrement, but up to the *second order*, then we get

$$\begin{aligned}
&\ell_{\theta_{t+1}}(s,a,s') - \ell_{\theta_t}(s,a,s') \\
&\approx \left\langle \theta_{t+1} - \theta_t, \nabla \ell_{\theta_t}(s,a,s') \right\rangle + \tfrac{1}{2}(\theta_{t+1} - \theta_t)^\top \left[\nabla^2 \ell_{\theta_t}(s,a,s')\right](\theta_{t+1} - \theta_t) \qquad (9) \\
&= \underbrace{-\eta \langle \nabla\!\!\!/ \ell_{\theta_t}(s_t,a_t,s_t'), \nabla \ell_{\theta_t}(s,a,s') \rangle}_{\text{first-order term}} + \underbrace{\tfrac{1}{2}\eta^2 \nabla\!\!\!/ \ell_{\theta_t}(s_t,a_t,s_t')^\top \left[\nabla^2 \ell_{\theta_t}(s,a,s')\right] \nabla\!\!\!/ \ell_{\theta_t}(s_t,a_t,s_t')}_{\text{second-order term } (\star)}.
\end{aligned}$$

Recall $\ell_\theta(s,a,s') = (Q_\theta(s,a) - r(s,a) - \gamma \max_{a'} Q_\theta(s',a'))^2$, and $\Delta_\theta(s,a,s') = Q_\theta(s,a) - r(s,a) - \gamma \max_{a'} Q_\theta(s',a')$. Direct calculation on the Hessian of the loss function gives:

$$\tfrac{1}{2}\nabla^2 \ell_{\theta_t}(s,a,s') = \frac{1}{4\,\ell_{\theta_t}(s,a,s')} \nabla \ell_{\theta_t}(s,a,s') \nabla \ell_{\theta_t}(s,a,s')^\top + \Delta_{\theta_t}(s,a,s') \nabla^2 h_{\theta_t}(s,a,s'),$$

where we denote $h_\theta(s,a,s') = Q_\theta(s,a) - \gamma \max_{a'} Q_\theta(s',a')$. Using this in the second-order term $(\star)$ above, we get

$$\begin{aligned}
(\star) = \frac{\eta^2}{4} \frac{\langle \nabla\!\!\!/ \ell_{\theta_t}(s_t,a_t,s_t'), \nabla \ell_{\theta_t}(s,a,s') \rangle^2}{\ell_{\theta_t}(s,a,s')} \\
+ \eta^2 \Delta_{\theta_t}(s,a,s') \nabla\!\!\!/ \ell_{\theta_t}(s_t,a_t,s_t')^\top \nabla^2 h_{\theta_t}(s,a,s') \nabla\!\!\!/ \ell_{\theta_t}(s_t,a_t,s_t').
\end{aligned}$$

Appendix E. LN-for-metrics-only ablation

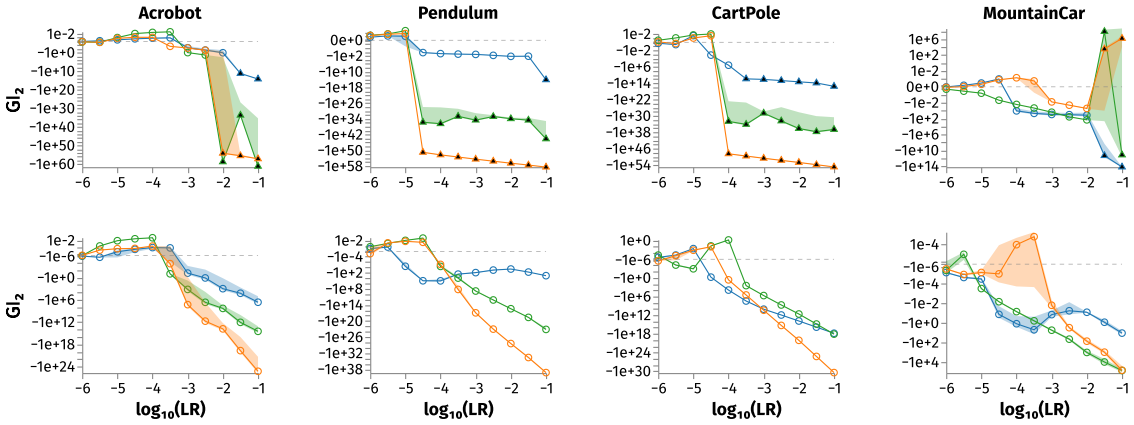

Figure 3: We use SGD (top row) and Adam (bottom row) to train without normalization again, but this time we add LN temporarily, periodically throughout training, only to measure GI_2. For comparison, we again show LN and no normalization as well. For both SGD and Adam, LN *still* tends to improve GI_2, even without allowing the network to take any training steps with LN in place. MountainCar is again an exception.

Appendix F. Related Work

A broad family of normalization techniques has played an important role in stabilizing the training of neural network, including LN [2], batch normalization [14], weight normalization [36], spectral

normalization [31], and group normalization [44]. The significant practical success of these methods has motivated a growing body of theoretical work aimed at understanding their mechanisms, providing various insights on the mechanism of different normalization modules. In this work, we are specifically interested in understanding whether and how these mechanisms carry to the RL setting.

Adaptive learning rate. [1] argue that normalization effectively adapts the learning rate during training. This property is common across normalization methods, as they impose invariance of training loss to weight scalings. Such invariance induces a monotonic growth in the weight norms during training and has been central to the development of the weight normalization method [36]. [25] study the similar phenomenon in reinforcement learning, but focus on its side effect in the multi-task setting caused by the vanishing learning rate. Our experiments, focusing on the single-task setting, show that the automatic adjustment of the learning rate is an important property in DQN which could improve performance. However, this alone cannot fully explain why other normalization layers, such as weight normalization, are less effective in DQN.

Faster regression. [19] prove batch normalization can accelerate linear regression when the input data is Gaussian. In this simplified setting, they provide quantitative convergence bounds for linear regression with batch normalization, demonstrating that gradient descent converges more rapidly with normalization. The acceleration arises from decoupling the optimization of weight magnitudes from their directions. However, their analysis relies heavily on the Gaussian assumption. In our work, we extend this proof to hold under substantially weaker assumptions on the data distribution.

Isometry. [16] introduce the notion of isometry, which quantifies the degree of orthogonality among samples or features. Their study shows that normalization layers induce an implicit bias toward orthogonalizing data representations in deep random neural networks. This contrasts with networks without normalization, where data samples become increasingly aligned as network depth grows [7]. While deep neural networks are widely used in supervised learning, DQN primarily relies on relatively shallow networks. Therefore, it remains unclear whether isometry can fully explain the mechanisms underlying DQN.

Gradient explosion with batch normalization. Despite its advantages, batch normalization can also introduce significant drawbacks that may limit its applicability. [46] prove that batch normalization leads to gradient explosion as network depth increases, a phenomenon initially thought to be unavoidable. However, [30] demonstrate that gradient explosion can in fact be mitigated through careful choices of hyperparameters and initialization.

Normalization in reinforcement learning. An important open question is whether explanations for supervised learning extend to reinforcement learning settings. Addressing this is crucial, as some normalization modules do not transfer effectively across different learning paradigms. For example, batch normalization can enhance the training of transformers, but can also destabilize it [39]. Interestingly, a similar pattern is observed in DQN, where only LN has consistently been shown to be effective. For example, [3, 8, 10, 13, 20, 21] all proposed combinations of techniques to enhance deep RL algorithms, and all include LN as an important component. Among them, [3] argue that LN improves the landscape of Q-function, preventing catastrophic value extrapolation, and [20] argues that LN makes the loss landscape smoother, avoiding the loss of plasticity.

Besides LN, attempts to use other normalization schemes have been made. [4] propose an adaptation of batch normalization to DQN (called CrossQ) that addresses the issue of distribution

shift of naive application of batch normalization to DQN. [34] further enhance CrossQ by combining it with weight normalization. On the other hand, [11] argue that spectral normalization provides automatic learning rate adaptation that benefits the training dynamics. Overall, however, other normalization schemes in DQN are rarer than LN.

Appendix G. Postponed Proofs

Standing assumptions for this section. Throughout this section, expectations are conditional on θ_t (equivalently on $e_t = \theta_t - \theta^\star$) and taken over fresh i.i.d. samples. We assume: (i) finite moments and nondegeneracy, $\mathbb{E}\|x\|^4 < \infty$ and $\mathbb{E}\|x\|^2 > 0$; (ii) the magnitude–direction model $x = az$ with $a = \|x\| \geq 0$ and $z = x/\|x\|$ when $a > 0$, with a independent of z (and z defined arbitrarily on $\{a = 0\}$); (iii) when $(x, \hat{x})$ appear together they are independent i.i.d. draws, each factorizable as az with the same independence structure.

Proof [Proof of Theorem 2] Under Assumption 1, let $e_t \triangleq \theta_t - \theta^\star$. One SGD step gives $e_{t+1} = e_t - \eta\, x_t x_t^\top e_t$, hence

$$\mathbb{E}\big[\|e_{t+1}\|^2 \mid \theta_t\big] = \|e_t\|^2 - 2\eta A_t + \eta^2 B_t,$$

with

$$A_t \triangleq e_t^\top \mathbb{E}[xx^\top] e_t, \qquad B_t \triangleq e_t^\top \mathbb{E}[xx^\top xx^\top] e_t.$$

Write $x = az$ with $a = \|x\|$ and $z = x/\|x\|$. Then

$$A_t = \mathbb{E}[a^2 v^2], \qquad B_t = \mathbb{E}[a^4 v^2], \quad \text{where } v \triangleq z^\top e_t.$$

By Cauchy–Schwarz applied to the measure with density v^2 (i.e., with $Y = a^2$ and $W = v^2$ so that $(\mathbb{E}[YW])^2/\mathbb{E}[Y^2 W] \leq \mathbb{E}[W]$),

$$\frac{A_t^2}{B_t} = \frac{\big(\mathbb{E}[a^2 v^2]\big)^2}{\mathbb{E}[a^4 v^2]} \leq \mathbb{E}[v^2] = e_t^\top \mathbb{E}[zz^\top] e_t,$$

with equality if and only if a^2 is almost surely constant on $\{v \neq 0\}$ (it may vary on $\{v = 0\}$ without effect). Therefore, for fixed $R(\theta_t) = \|e_t\|^2$, the ratio A_t^2/B_t is maximized when $\|x\|$ is constant under μ. ∎

Proof [Proof of Theorem 3] Let $\mathbb{E}_{(x,y)}$ denote $\mathbb{E}_{(x,y)\sim\mu}$ and define $\varepsilon_t(x, y) \triangleq x^\top \theta_t - y$. Expanding one SGD step yields

$$\begin{aligned} L(\theta_{t+1}) &= \mathbb{E}_{(x,y)}\Big[\big(\varepsilon_t(x,y) - \eta\, x^\top x_t\, \varepsilon_t(x_t, y_t)\big)^2\Big] \\ &= L(\theta_t) - 2\eta\, \mathbb{E}_{(x,y)}\big[\varepsilon_t(x,y)\, \varepsilon_t(x_t,y_t)\, x^\top x_t\big] + \eta^2\, \mathbb{E}_{(x,y)}\big[\varepsilon_t(x_t,y_t)^2\, (x^\top x_t)^2\big]. \end{aligned}$$

Taking expectation over $(x_t, y_t) \sim \mu$, we obtain

$$\mathbb{E}[L(\theta_{t+1}) \mid \theta_t] = L(\theta_t) - 2\eta\, C_t + \eta^2 D_t,$$

where

$$C_t \triangleq \mathbb{E}_{(x,y)}\mathbb{E}_{(\hat{x},\hat{y})}\big[\varepsilon_t(x,y)\, \varepsilon_t(\hat{x},\hat{y})\, x^\top \hat{x}\big],$$

$$D_t \triangleq \mathbb{E}_{(x,y)}\mathbb{E}_{(\hat{x},\hat{y})}\left[\varepsilon_t(\hat{x},\hat{y})^2\,(x^\top \hat{x})^2\right].$$

Thus, for any η, the loss decrement has the quadratic form above, and the ratio governing the optimal decrement is C_t^2/D_t.

Under Assumption 1, write $\varepsilon_t(x,y) = x^\top e_t$ with $e_t \triangleq \theta_t - \theta^\star$. Decompose $x = az$ and $\hat{x} = \hat{a}\,\hat{z}$ with $a = \|x\|$, $z = x/\|x\|$. Using independence between magnitude and direction and i.i.d. sampling across the two draws, define

$$U_t \coloneqq \mathbb{E}_{z,\hat{z}}\left[(z^\top e_t)(\hat{z}^\top e_t)(z^\top \hat{z})\right],$$
$$W_t \coloneqq \mathbb{E}_{z,\hat{z}}\left[(\hat{z}^\top e_t)^2(z^\top \hat{z})^2\right].$$

Then

$$C_t = \mathbb{E}\left[(a\,z^\top e_t)(\hat{a}\,\hat{z}^\top e_t)(a\hat{a}\,z^\top \hat{z})\right] = \mathbb{E}[a^2]^2\,U_t,$$
$$D_t = \mathbb{E}\left[(\hat{a}\,\hat{z}^\top e_t)^2(a\hat{a}\,z^\top \hat{z})^2\right] = \mathbb{E}[a^2]\,\mathbb{E}[a^4]\,W_t.$$

Consequently,

$$\frac{C_t^2}{D_t} = \frac{\mathbb{E}[a^2]^4}{\mathbb{E}[a^2]\,\mathbb{E}[a^4]} \cdot \frac{U_t^2}{W_t}.$$

For fixed $L(\theta_t) = \mathbb{E}[\varepsilon_t(x,y)^2] = \mathbb{E}[a^2]\,\mathbb{E}[(z^\top e_t)^2]$, the directional term $\mathbb{E}[(z^\top e_t)^2]$ is fixed, hence $\mathbb{E}[a^2]$ is pinned. Thus maximizing C_t^2/D_t over the magnitude distribution reduces to minimizing $\mathbb{E}[a^4]$, which is achieved when a^2 is almost surely constant (by Jensen). Hence C_t^2/D_t is maximized when $\|x\|$ is constant under μ. ■

Degenerate edge cases. If $e_t = 0$ or $z^\top e_t = 0$ almost surely, then $A_t = C_t = 0$ and the one-step maximization statements are vacuous. If $\mathbb{P}(a = 0) > 0$, values of z on $\{a = 0\}$ are irrelevant since all weighted terms place zero mass there.

Appendix H. Isometry

Theoretical hypothesis. Let $X = \{x_1, x_2 \ldots, x_n\} \subset \mathbb{R}^d$. The Gram matrix $G(X)$ is defined as $G(X)_{ij} = \langle x_i, x_j\rangle, \forall i,j \in [n]$. Following [16], the *isometry* of X is defined as

$$\mathsf{ISO}\,(X) = \det(G(X))^{\frac{1}{|X|}} / \left(\tfrac{1}{|X|}\,\mathrm{Tr}(G(X))\right),$$

where $|X| = n$ is the cardinality of X. $\mathsf{ISO}(X)$ is at most 1 and captures the degree of isotropy: $\det(G(X))$ is equal to the squared volume of the parallelotope spanned by $x_1, \ldots, x_n$, and $\mathrm{Tr}(G(X)) = \sum_{i=1}^n \|x_i\|^2$ is sum of their squared norms. The closer this ratio is to 1, the more isotropic X is.

To extend this notion to distributions, we define the order-k isometry of a distribution μ as

$$\mathsf{ISO}_k\,(\mu) = \mathbb{E}_{X=\{x_1,\ldots,x_k\}\sim\mu^k}\left[\mathsf{ISO}(X)\right],$$

where X consists of k independent samples from μ. This quantity measures the expected isometry of size-k datasets. Note that $\det(G(X)) = 0$ whenever the vectors in X are linearly dependent, so k should not be chosen larger the dimension of the subspace spanned by the support of μ.

Let $z \sim \overline{\mu}$ be the distribution generated by first drawing $x \sim \mu$ and setting $z = x/\|x\|$. It is straightforward to show that for any k, $\mathsf{ISO}_k(\mu) \le \mathsf{ISO}_k(\overline{\mu})$, given [16]'s theorem:

Theorem 4 (Theorem 1 of [16]) *Let $X = \{x_1, \ldots, x_n\}$ and be an arbitrary dataset and let $Z = \{x_1/\|x_1\|, \ldots, x_n/\|x_n\|\}$. Then $\mathsf{ISO}(X) \leq \mathsf{ISO}(Z)$.*

Theorem 4 implies that for any size-k datasets X and its normalized version Z, $\mathsf{ISO}(X) \leq \mathsf{ISO}(Z)$. The inequality $\mathsf{ISO}_k(\mu) \leq \mathsf{ISO}_k(\overline{\mu})$ follows by taking expectation over the data distribution μ.

[16] provide empirical evidence that, in supervised learning, LN results in increased ISO of the network's intermediate features throughout training. They also empirically find that ISO positively correlates with the convergence rate. We hypothesize that LN will similarly increase ISO_k, even after training, in deep Q-learning. We likewise hypothesize that ISO_k will correlate with returns.

Empirical evidence. Beyond the fact that k should not be chosen larger than the rank of the Gram matrix, it is unclear what k should be set to. There are several potential options: setting k as the dimensionality of the observation space of the environment, or the total dimensionality of the inputs (either the observation dimensionality plus the action dimensionality, or the observation dimensionality times the action dimensionality), or a small, fixed constant such as 3. We find that all such k values (not shown) give results aligning fairly well with our hypotheses. We show ISO_p, which sets k to the observation **p**lus action dimensionality, in Fig. 4. We show results for Adam and for all other k values in Appendix J.

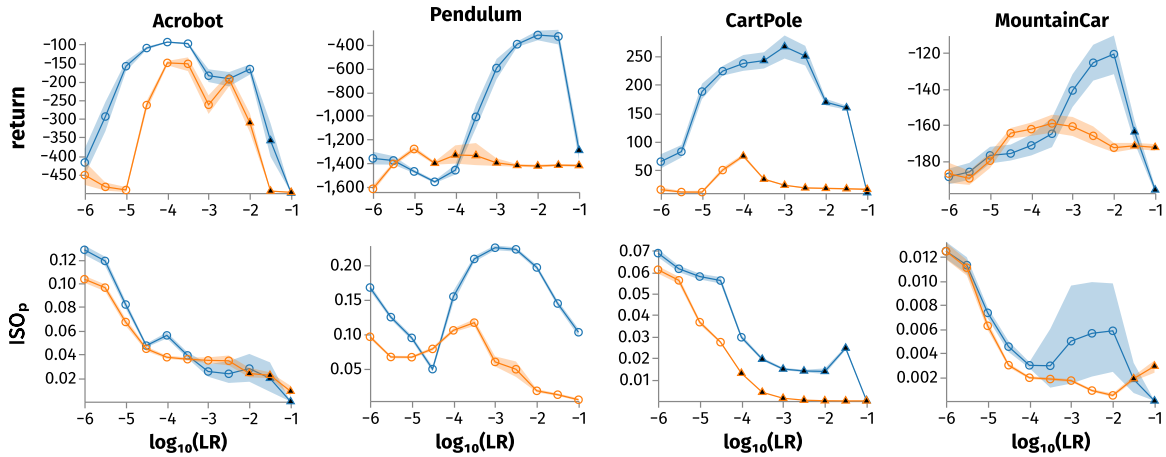

Figure 4: ISO_p correlates well with return. These correlations tend to hold at any given learning rate (LR), and also sometimes across learning rates. SGD results only. Results with Adam and other values of k (for ISO_k) are in Appendix J.

Future work. Theoretically connecting the isometry metric to the convergence rate (even in supervised learning) is perhaps the most important open question here.

Isometry, how clustered the eigenvalues of the Gram matrix of the learned representations are [16]. We derive a new version of this metric for the setting where batches are large compared to feature dimensions, and find LN still improves this new metric in our RL setting. Theoretically connecting this metric to the convergence rate (even in supervised learning) is perhaps the most important open question here.

Appendix I. Implicit Learning Rate Decay

Theoretical hypothesis. In this subsection, we focus on the *effective learning rate* for the first layer of our neural network, i.e., the W parameter in Eq. (2). Notice that the output value will not change if we replace the W in Eq. (2) with αW for any $\alpha > 0$ as the function LN is invariant to the scale of W.

Now, we review general properties of this type of functions. Assume $f(\alpha w) = f(w)$ for any $\alpha > 0$. Since $f(w)$ is invariant to the scale of w, a meaningful update on $f(w)$ must make an update in the unit vector $\hat{w} = \frac{w}{\|w\|}$ that represents the *direction*. The following theorem by [1] shows (i) that the effective learning rate in the space of $\hat{w}$ is roughly $\frac{\eta}{\|w\|^2}$, and (ii) that $\|w\|$ monotonically increases over time. Jointly, they imply that the effective learning rate decreases over time.

Lemma 5 (Theorem 2.5 of [1]) *Let $f(w)$ be invariant to the scale of w. Denote $\hat{w} \triangleq \frac{w}{\|w\|}$. For the gradient update $w^+ = w - \eta \nabla_w f(w)$, we have $\hat{w}^+ = \hat{w} - \frac{\eta}{\|w\|^2} \nabla_{\hat{w}} f(\hat{w}) + O(\eta^2)$ and $\|w^+\|^2 = \|w\|^2 + \frac{\eta^2}{\|w\|^2} \|\nabla_{\hat{w}} f(\hat{w})\|^2$.*

Such a learning rate decay argument not only applies to LN, but also to other forms of normalization such as batch normalization and weight normalization. [25] observe that such an implicit learning rate decay could be harmful to continual reinforcement learning, as the learner becomes slower and slower in learning new tasks. To address this, they periodically project W back to a ball of fixed radius.

Empirical evidence. In preliminary experiments (not shown), following [25], we tested removing LN's implicit learning rate decay by projecting the weights to back to their initial norm after every gradient step. We also tested explicitly adding LN's implicit learning rate decay (i.e., setting the learning rate based on the norm of the weights) to a network without normalization. In both cases, aligning with [25], our results suggested that implicit learning rate decay might sometimes increase returns, but did not fully account for LN's increase.

Implicit learning rate decay, the fact that LN (and other normalizations) implicitly decay the learning rate as the magnitudes of the weights increase throughout training. Aligning with Lyle et al. [25], our results suggest that implicit learning rate decay might sometimes increase returns, but does not fully account for LN's increase of returns.

Appendix J. All ISO_k Plots

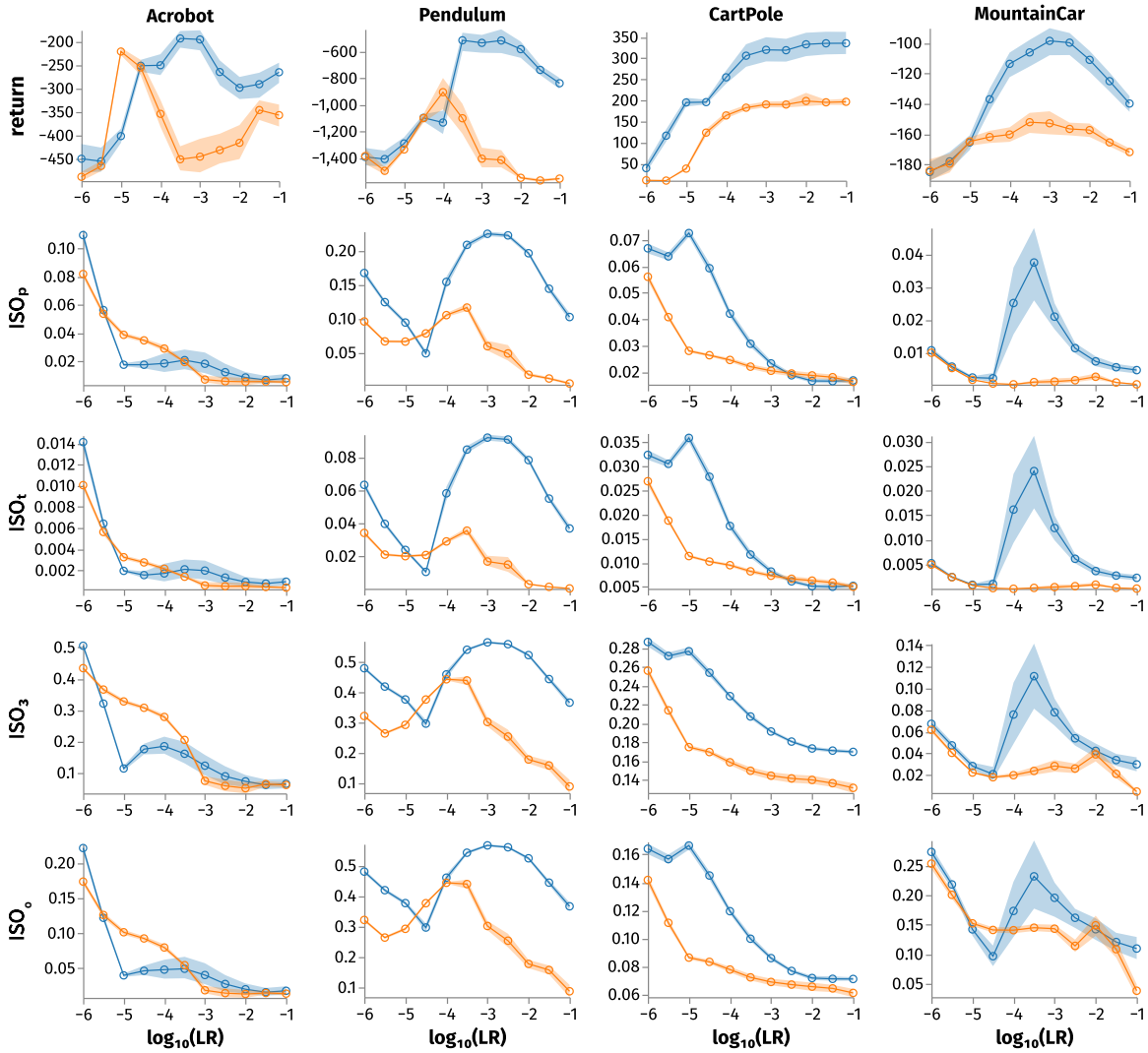

Figure 5: The same as Fig. 4, but using Adam instead of SGD, and including all of the k values mentioned in Appendix H for ISO_k. The results are mostly similar at all values of k. ISO_t sets k to the dimensionality of the observation space **t**imes the dimensionality of the action space. ISO_3 sets k to 3. ISO_o sets k to the dimensionality of the **o**bservation space alone.

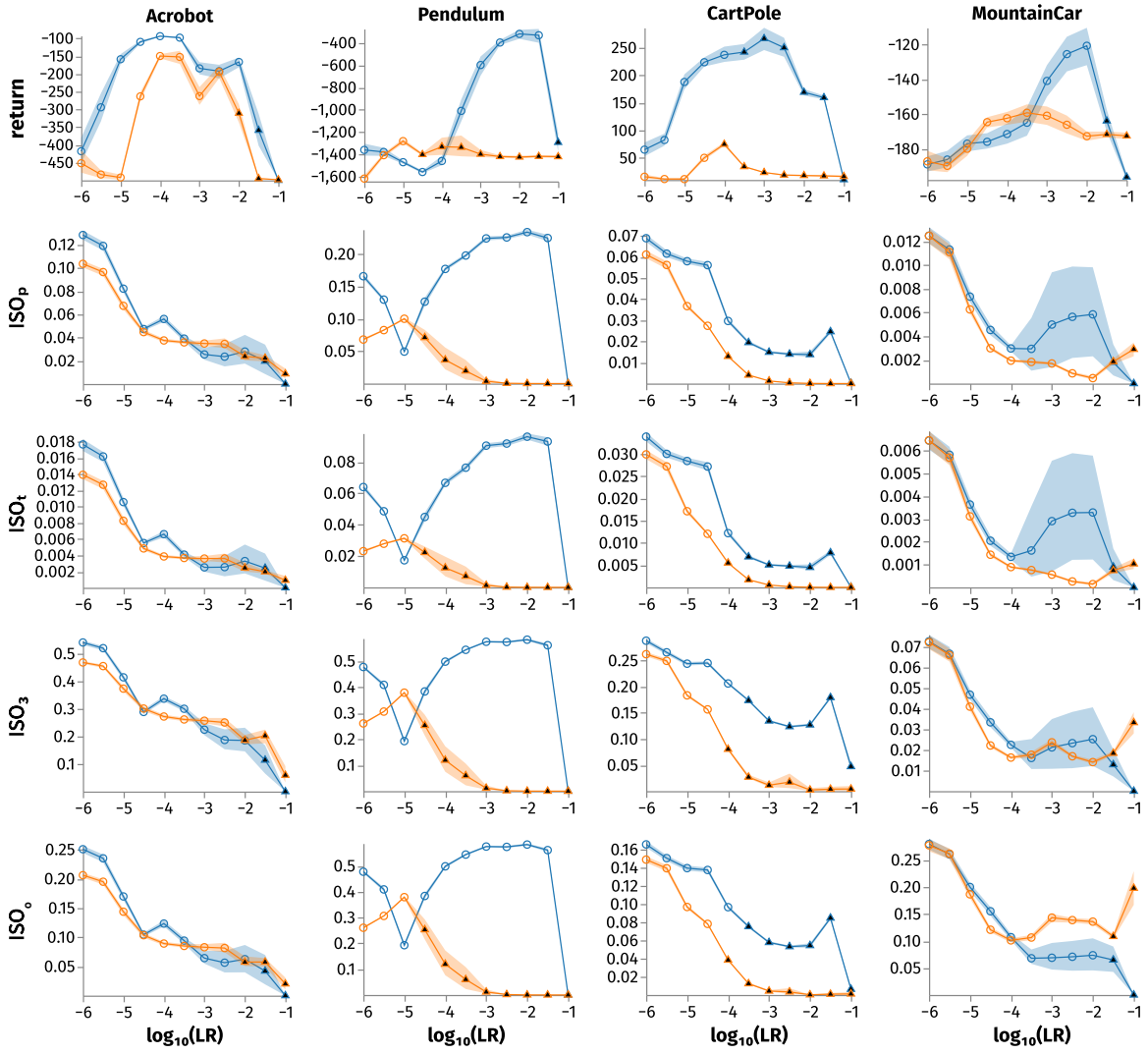

Figure 6: The same as Fig. 5, but using SGD. The various values of k again give mostly similar results.

Appendix K. All Per-Learning Rate Gradient Interference Plots

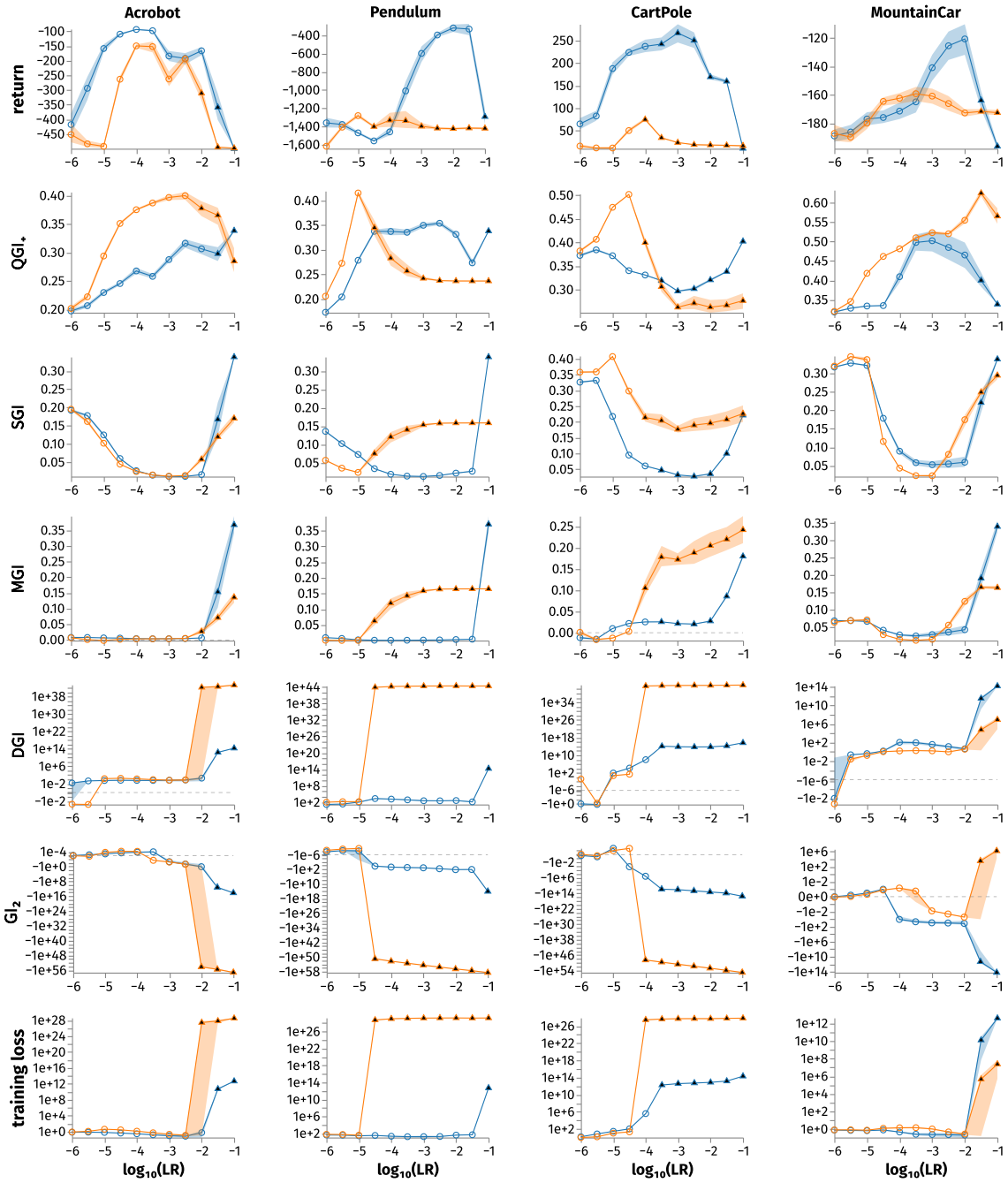

Figure 7: All of the gradient interference metrics when training with SGD. The return, training loss, QGI_+, and SGI are the same as plots as in Fig. 2. MGI is the mixed-gradient interference, i.e. the cosine similarity of each pair of semi-gradients and full-gradients of the TD error. As mentioned in Section 3.1, there are theoretical arguments for focusing on MGI instead of SGI, though in our experiments it still counterintuitively correlates with return. DGI is the dot-product gradient interference, the dot product version of MGI. That is, DGI is the dot product, rather than the cosine similarity, of each semi-gradient and full-gradient. As mentioned in the main text, DGI still positively correlates with the loss. GI_2 is again the same as the plots in Fig. 2.

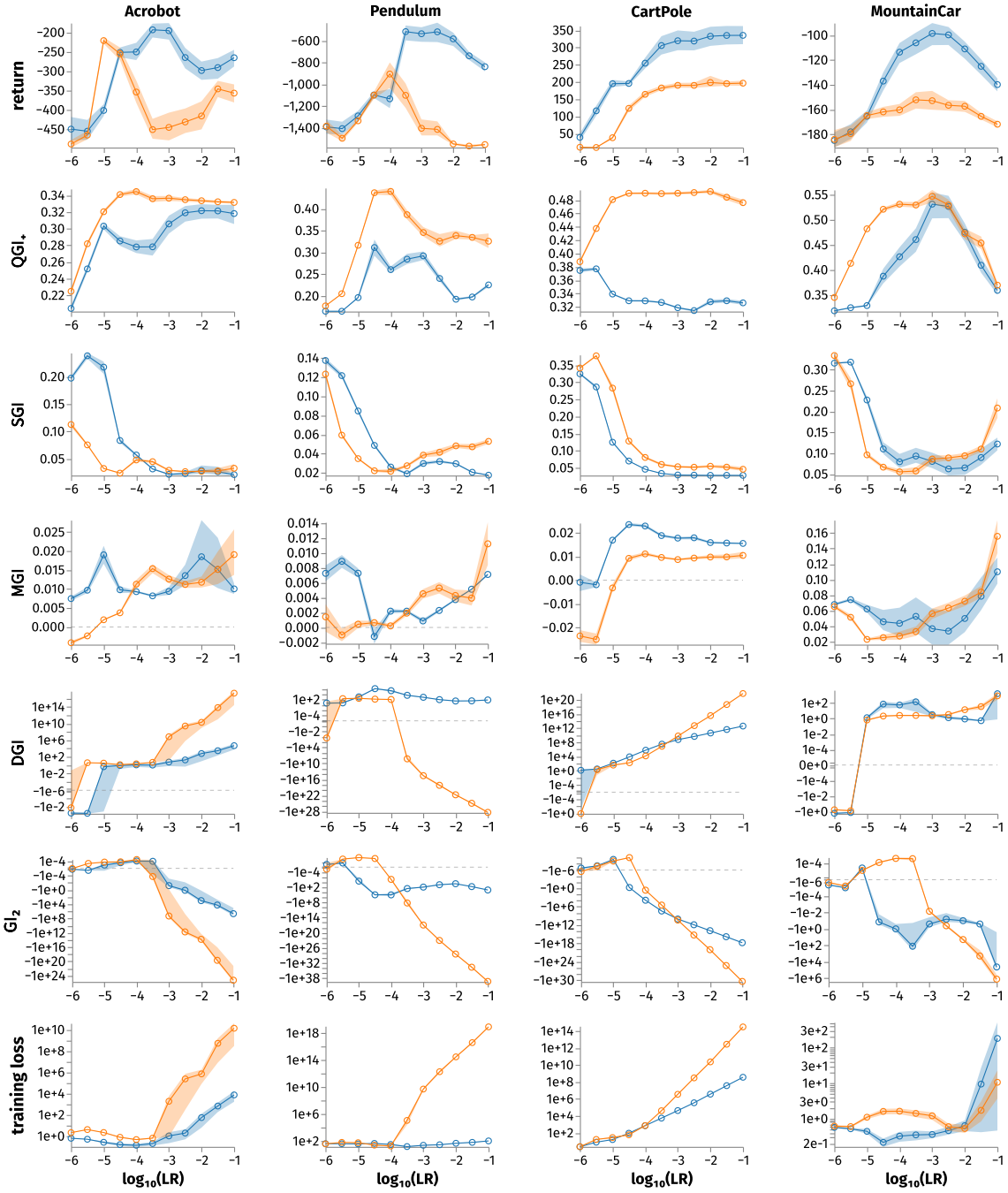

Figure 8: The same as Fig. 7, but using Adam. The results are similar to the SGD case.

Appendix L. Experimental Details

Everywhere that we use SGD in this work, we use zero momentum. For Adam [18], we use its default hyperparameters.

For Table 1, the 9 learning rates are $\{10^{-1}, 10^{-1.5}, \ldots, 10^{-5}\}$. BN is batch normalization, WN is weight normalization. The "Random" column uses the Q-values of the randomly initialized network without any normalization layers. We additionally tested CrossQ [4] — in our preliminary experiments (not shown), CrossQ achieved around the same returns as LN, though CrossQ requires an additional hyperparameter, the BN momentum. An interesting direction for future work is to study how CrossQ and other normalizations affect the metrics we study.

For all experiments, we use a batch size of 128, a hidden layer width of 128, a discount factor of 0.99, and 200,000 training steps (gradient steps). Environments are: `Pendulum-v1` with a uniformly random initial state-action distribution [45], and an action-discretization from the default continuous space $[-2, 2]$ to instead $\{-2, 0, 2\}$; `MountainCar-v0` with a uniformly random initial state-action distribution; `Acrobot-v1`; and `CartPole-v1`. For any hyperparameters not explicitly specified, we use the PyTorch default. For example, for LN, we use the PyTorch default $\varepsilon = 10^{-5}$.

To compute gradient interference metrics and isometry metrics, we again use a batch of size 128, randomly drawn for each measurement.

For all experiments, to avoid NaN metric values, we clamp the neural network weights after every gradient step to be equal to or under an absolute value of one million. Empirically, this clamping only occurs when using SGD, never when using Adam.

Appendix M. All Gradient Interference Scatter Plots

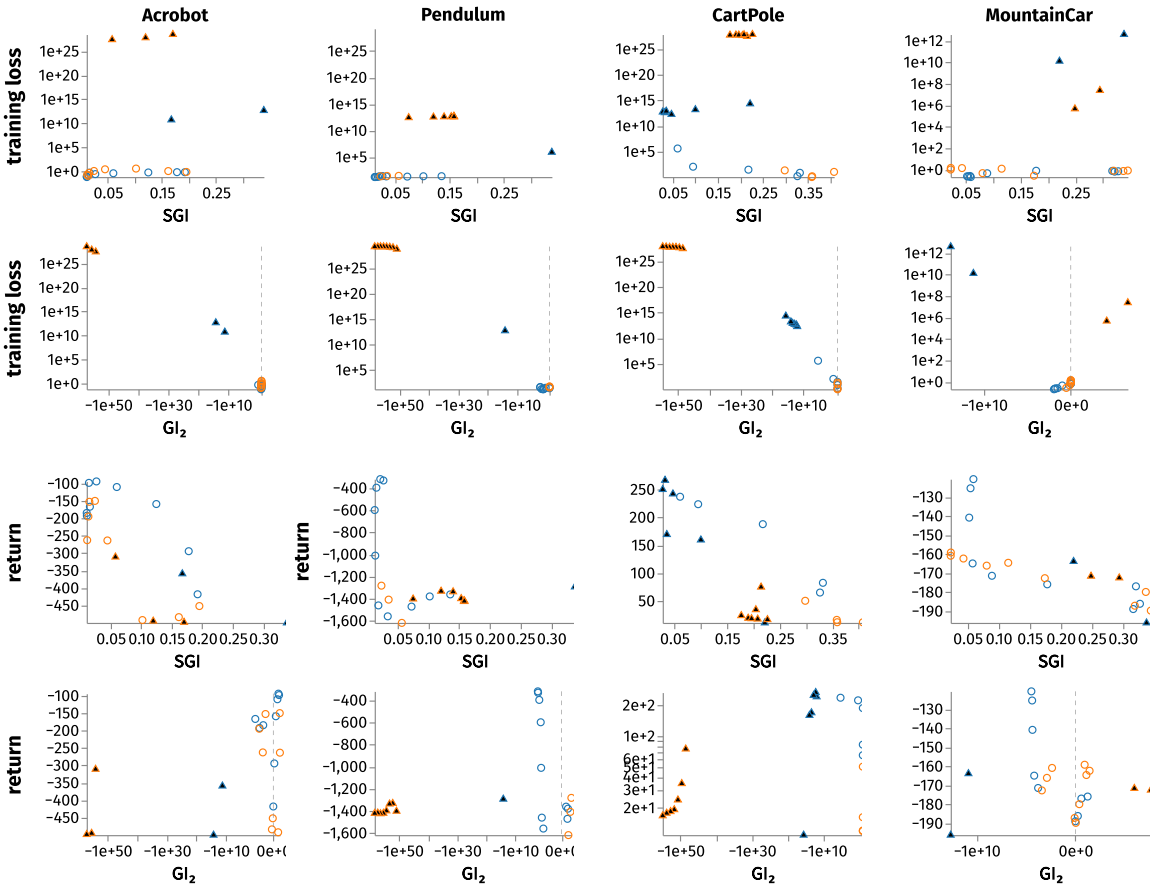

Figure 9: The same as Fig. 1, but with additional plots showing the correlation with return, not only with training loss. On CartPole, SGI correlates negatively with the return, even though it correlates negatively with the loss as well. For Adam (not shown), the correlation between SGI and loss reverses, but the correlation between SGI and return remains negative.