Observational Overfitting in Reinforcement Learning

Xingyou Song[*], **Yiding Jiang**[†], **Behnam Neyshabur**
Google Research
{xingyousong,ydjiang,neyshabur}@google.com

Yilun Du[*]
MIT
yilundu@mit.edu

Abstract

A major component of overfitting in model-free reinforcement learning (RL) involves the case where the agent may mistakenly correlate reward with certain spurious features from the observations generated by the Markov Decision Process (MDP). We provide a general framework for analyzing this scenario, which we use to design multiple synthetic benchmarks from only modifying the observation space of an MDP. When an agent overfits to different observation spaces even if the underlying MDP dynamics is unchanged, we term this *observational overfitting*. Our experiments expose intriguing properties especially with regards to *implicit regularization*, and also corroborate results from previous works in RL generalization and supervised learning (SL).

1 Introduction

Generalization for RL has recently grown to be an important topic for agents to perform well in unseen environments. Complication arises when the dynamics of the environments tangle with the observation, which is often a high-dimensional projection of the true latent state. In this work, we focus on model-free RL and use the common *zero-shot supervised framework* [1, 2, 3, 4], which treats RL generalization analogous to a classical supervised learning (SL) problem – i.e. assume there exists a distribution of MDP's $\{\mathcal{M}_\theta : \theta \in \Theta\}$ which can be generated by an environment parameter θ, train jointly on a finite "training set" $\{\mathcal{M}_\theta : \theta \sim \widehat{\Theta}_{n,train}\}$ sampled from this distribution, and check generalization gap between rewards $J_{\widehat{\Theta}}(\pi) - J_{\Theta}(\pi)$ against the entire distribution with the fixed trained policy π.

There are multiple confounding factors at play in this regime, which can be hard to separate. An agent can overfit to the MDP dynamics [5, 6], or an RNN-based policy can overfit to maze-like tasks in exploration [2], or even exploit determinism [7, 8]. Furthermore, various hyperparameters and regularizations [9, 10, 11, 12] can also affect generalization.

We isolate one broad factor affecting generalization that is most correlated with themes in SL, specifically *observational overfitting*, where an agent overfits due to properties of the observation which are irrelevant to the latent dynamics of the MDP family. To study this factor, we fix a single underlying MDP's dynamics and generate a distribution of MDP's by only modifying the observational outputs.

Our contributions in this paper are the following:

1. We discuss realistic instances where observational overfitting may occur and its difference from other confounding factors, and design a parametric theoretical framework to induce observational overfitting that can be applied to *any* underlying MDP.
2. We study observational overfitting with linear quadratic regulators (LQR) in a synthetic environment and neural networks such as MLPs and Convolutions in classic Gym environments. A primary novel result we demonstrate for all cases is that *implicit regularization*

[*]Work partially performed as an OpenAI Fellow.

[†]Work performed during the Google AI Residency Program. http://g.co/airesidency

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

occurs in this setting in RL. We further test the implicit regularization hypothesis on the benchmark CoinRun from using multi-layer perceptrons (MLP), even when the underlying MDP dynamics are changing per level.

2 Motivation and Setup

Figure 1: Example of observational overfitting in Gym Retro [3] for Sonic. Saliency maps highlight (in red) the top-left timer and background objects because they are correlated with progress. The agent could memorize optimal actions for training levels if its observation was *only from the timer*, and "blacking-out" the timer consistently improved generalization performance (see Appendix A.2.3).

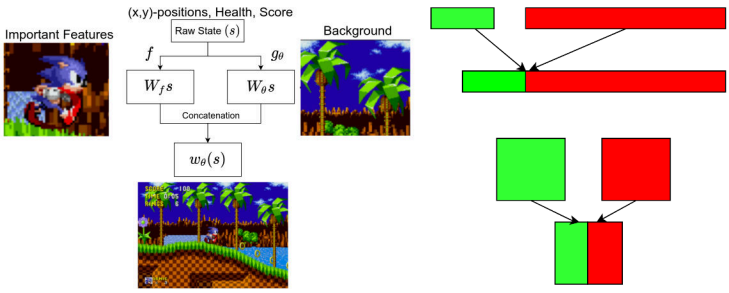
Figure 1 highlights the issues surrounding MDP's with rich, textured observations - specifically, the agent can use any features that are correlated with progress, even those which may not generalize across levels. Several artificial benchmarks [13, 14] have been proposed to study these behaviors, where an agent must deal with a changing background - however, the key difference in insight is that we explicitly require the "background" to be **correlated with the progress** rather than loosely correlated (e.g. through determinism between the background and the game avatar) or not at all, making this similar to the causal inference literature, [15, 16, 17] where a classifer is prone to produce *spurious correlations* and must find invariant features to improve generalization.

We can model the effects of Figure 1 more generally, not specific to sidescroller games. We assume that there is an underlying *state* s (e.g. xy-locations of objects in a game), whose features may be very well structured, but that this state has been projected to a high dimensional observation space by w_θ. To abstract the notion of generalizable and non-generalizable features, we construct a simple and natural candidate class of functions, where $w_\theta(s) = h(f(s), g_\theta(s))$.

In this setup, $f(\cdot)$ is a function invariant for the entire MDP population Θ, while $g_\theta(\cdot)$ is a function dependent on the sampled parameter θ. h is a "combination" function which combines the two outputs of f and g to produce a final observation. While f projects this latent data into salient and important, *invariant* features such as the avatar, monsters, and items, g_θ projects the latent data to unimportant features that do not contribute to extra generalizable information, and can cause overfitting, such as the changing background or textures. A visual representation is shown in Figure 2. Normally in a MDP such as a game, the concatenation operation may be dependent on time (e.g. textures move around in the frame). In the scope of this work, we simplify the concatenation effect and assume $h(\cdot)$ is a static concatenation, but still are able to demonstrate insightful properties. We note that this inductive bias on h allows *explicit regularization* to trivially solve this problem, by penalizing a policy's first layer that is used to "view" $g_\theta(s)$ (Appendix A3), hence we only focus on implicit regularizations.

Previously, many works interpret the decision-making of an agent through saliency and other network visualizations [18, 19] on real-world tasks such as Atari. Our work is motivated by learning theoretic frameworks to capture this phenomena, as there is vast literature on understanding the generalization properties of SL classifiers [20, 21, 22, 23]. For an RL policy with high-dimensional observations, we hypothesize its generalization performance can come from more theoretically principled reasons, as opposed to purely good inductive biases on real-world images.

This setting also leads to more interpretable generalization bounds. The Rademacher Complexity term in RL [24]) is defined $Rad_m(R_\Pi) = \mathbb{E}_{(\theta_1,\ldots,\theta_m)\sim\Theta^m}\left[\mathbb{E}_{\sigma\in\{-1,+1\}}\left[\sup_{\pi\in\Pi}\frac{1}{m}\sum_{i=1}^{m}\sigma_i R_{\theta_i}(\pi)\right]\right]$, which captures how invariant policies in the set Π with respect to θ. For most RL benchmarks, this is



(a) (b)

Figure 2: (a) Visual Analogy of the Observation Function. (b) Our combinations for 1-D and 2-D images for synthetic tasks.

not interpretable with changing dynamics, as it is hard to imagine what behaviors or network weights a policy would possess in order to produce the *exact same* total rewards, *regardless of changing dynamics*. However, in the observation case, the Rademacher Complexity is directly based on how much the policy "looks at" g_θ which is the only environment change. If Π^* is the set of policies π^* which are not be affected by changes in g_θ, then $\nabla_\theta \pi^*(w_\theta(s)) = 0 \; \forall s$ and thus $R_\theta(\pi^*) = R_{const} \; \forall \theta$; hence $Rad_m(R_{\Pi^*}) = \mathbb{E}_{\sigma \in \{-1,+1\}} \left[\sup_{\pi^* \in \Pi^*} \frac{1}{m} \sum_{i=1}^{m} \sigma_i R_{const} \right] = 0$.

3 Experiments

This setting is naturally attractive to analzying architectural differences, as it is more closely related in spirit to image classifiers and SL. One particular line of work to explain the effects of certain architectural modifications in SL such as overparametrization and residual connections is *implicit regularization* [25, 23]. As we will see in this experimental section, this can play a large role in the RL generalization regime. In order to fairly measure generalization, we use fixed hyperparameters in the RL algorithm, and only vary based on architecture for all experiments.

3.1 Overparametrized LQR

We start with a principled example in the deterministic classic control setting, by using the linear quadratic regulator (LQR) as a basis for the underlying MDP. We use full gradient descent through the loss, ignoring irrelevant aspects of reinforcement learning (exploration, entropy, γ, noise, stochastic gradients, etc.). We project the state to a high dimensional observation $o_t = \begin{bmatrix} f(s_t) \\ g_\theta(s_t) \end{bmatrix} = \begin{bmatrix} W_c \\ W_\theta \end{bmatrix} s_t$, where an agent parametrized by a high dimensional policy K performs action $a_t = K o_t$. W_c, W_θ are semi-orthogonal matrices (to prevent information loss), and W_c remains fixed while W_θ is randomly sampled per level parameter θ. If s is of shape d_{state}, then W_c projects to a shape of d_{signal}, W_θ projects to a much larger shape d_{noise}, and thus K is of shape $(d_{state}, d_{signal} + d_{noise})$.

If $P_\star$ is the unique minimizer of the original cost function, then the unique minimizer of the population cost is $K_\star = \begin{bmatrix} W_c P_\star^\mathsf{T} \\ 0 \end{bmatrix}^\mathsf{T}$. However, if we have a single level, then there exist multiple solutions, for instance $\begin{bmatrix} \alpha W_c P_\star^\mathsf{T} \\ (1-\alpha) W_\theta P_\star^\mathsf{T} \end{bmatrix}^\mathsf{T} \; \forall \alpha$. This extra bottom component $W_\theta P_\star^\mathsf{T}$ causes overfitting. In Appendix A.4.2, we show that in the 1-step LQR case (which can be extended to convex losses whose gradients are linear in the input), gradient descent cannot remove this component, and thus overfitting necessarily occurs.

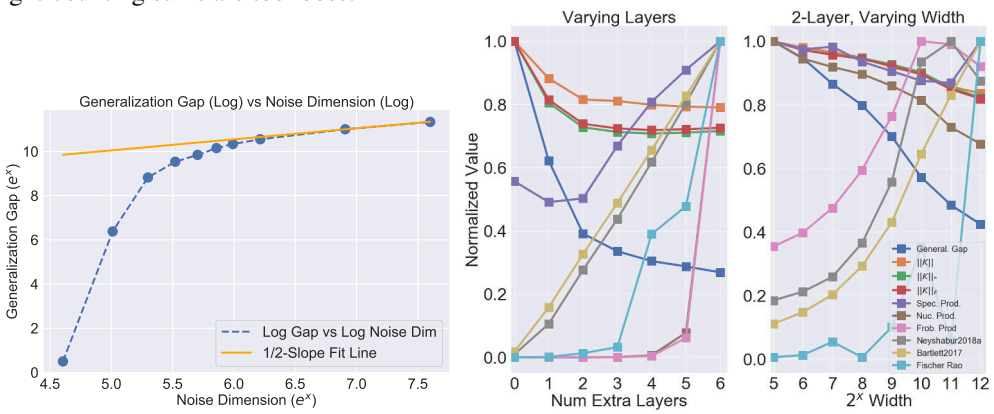
Furthermore, we find that increasing d_{noise} increases the generalization gap in the LQR setting. This is empirically verified in Figure 3 using an actual non-convex LQR loss, and the results suggest that the gap scales by $\mathcal{O}(\sqrt{d_{noise}})$. In our experiment, we set $(d_{signal}, d_{noise}) = (100, 1000)$, and also added more (100×100) linear layers $K = K_0 K_1, ..., K_j$ and increased widths for a 2-layer case (Figure 3), and observe that both settings reduce the generalization gap, and also reduce the norms (spectral, nuclear, Frobenius) of the final end-to-end policy K, without changing its expressiveness. This suggests that gradient descent under overparametrization implicitly biases the policy towards

a "simpler" model in the LQR case. However, from examining the distribution of singular values on K (Appendix A1), we find that more layers does not bias the policy towards a low rank solution, unlike [26] which shows this occurs for matrix completion and convex losses. Intriguingly, we find that many of the known bounds in SL for both width and depth do not answer this problem, although we still have a similar smoothness bound $C(K; W_\theta) - C(K'; W_\theta) \leq \mathcal{O}(\|K - K'\|)$ shown in A.4, which can be used to derive similar expressions as the SL bounds. To the best of our knowledge, our problem is not equivalent to any of the well-studied problems in overparametrization.

For instance, we used bounds of the form $\Pi \cdot \Sigma$, where Π is a "macro" product term $\Pi = \prod_{i=0}^{j} \|K_i\| \geq \left\| \prod_{i=0}^{j} K_i \right\|$ derivable from the fact that $\|AB\| \leq \|A\| \, \|B\|$ and Σ is a *weight-counting* term which deals with the overparametrized case, such as $\Sigma = \sum_{i=0}^{j} \frac{\|K_i\|_F^2}{\|K_i\|^2}$ [27] or $\Sigma = \left(\sum_{i=0}^{j} \left(\frac{\|K_i\|_1}{\|K_i\|} \right)^{2/3} \right)^3$ [28]. However, the Σ terms increase too rapidly for both the depth and width cases, as we show in Figure 3.

Terms such as Frobenius product [29] and Fischer-Rao [30] are effective for the SL depth case, but seem to be ineffective in the LQR depth case. For width, the only product which appears effective is the nuclear norm product, whereas spectral and Frobenius products [26] are ineffective.

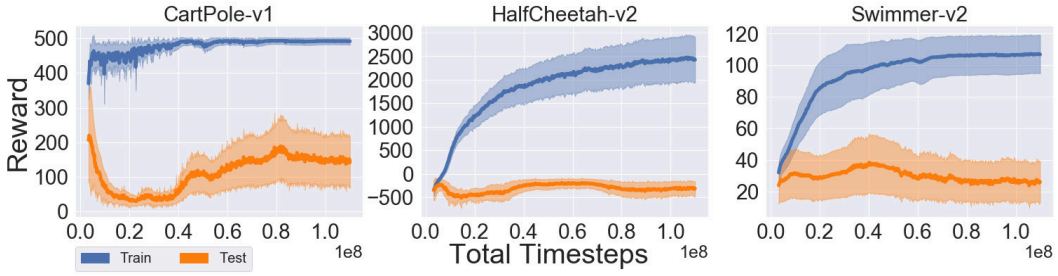
Figure 3: (Left) We show that the generalization gap vs noise dimension appears to tightly follow $\mathcal{O}(\sqrt{d_{noise}})$ as the noise dimension increases. (Right) LQR Generalization Gap vs Number of Intermediate Layers. We plotted the different Σ terms without exponents, as powers of these terms are monotonic transforms. We see that the naive spectral bound diverges at 2 layers, and the weight-counting sums are too loose.



3.2 Projected Gym Environments

We perform the above setup with non-linear environments as well, such as Gym and Mujoco environments. We show that even in the space of basic MLP policies, there are significant architectural effects.

Figure 4: Observational overfitting occurring for Gym. Full set of environments shown in A2.



Switching between ReLU and Tanh activations produces significantly different results during over-parametrization. For instance, overparametrization on Tanh layers improves generalization on CartPole-v1, and width increase with ReLU helps on Swimmer-v2. Tanh is noted to consistently

Figure 5: Effects of Depth.

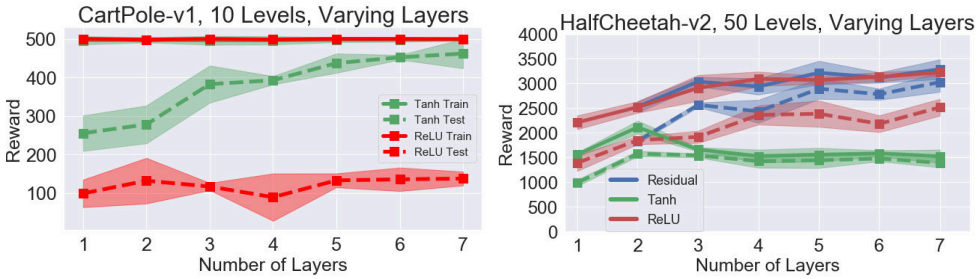
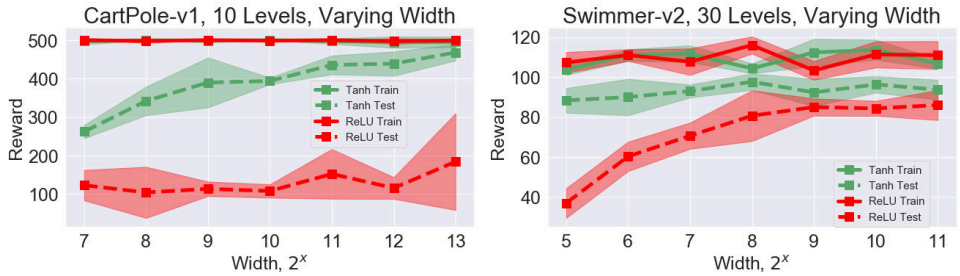


Figure 6: Effects of Width.



improve generalization performance. However, stacking Tanh layers comes at a cost of also producing vanishing gradients which can produce subpar training performance, for e.g. HalfCheetah. To allow larger depths, we use ReLU *residual* layers, which also improves generalization and stabilizes training.

Previous work [2] did not find such a pattern, suggesting that this effect may exist primarily for observational overfitting cases. While there has been numerous works on simplifying policies [31, 32] or compactifying networks [33, 34], we instead find that there are generalization benefits to overparametrization even in the nonlinear control case.

3.3 Deconvolutional Projections

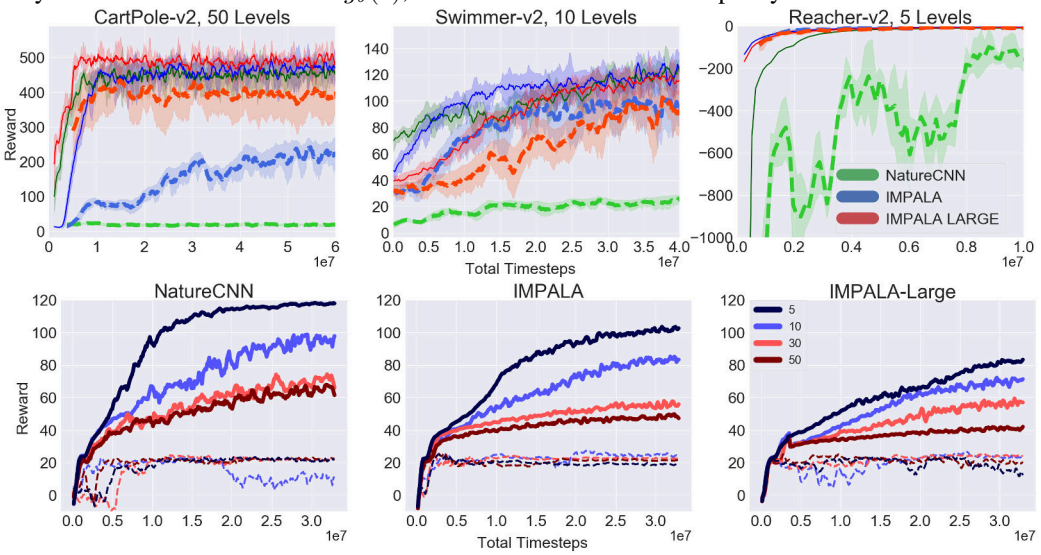
We also setup a similar construction for convolutions, by projecting the state latent to a fixed length, reshaping it into a square, and using our deconvolution projections to produce two 84×84 images. We combine the two outputs by using one half of the "image" from f, and one half from g_θ, as shown back in Figure 2.

As a ground truth reference, we the canonical networks proven to generalize well in the dataset CoinRun, which are from worst to best, NatureCNN [35], IMPALA [36], and IMPALA-LARGE (IMPALA with more residual blocks and higher convolution depths), which have respective parameter numbers (600K, 622K, 823K).

Figure 7 shows that the same ranking between the three architectures exists our synthetic dataset as found in CoinRun. This suggests that the RL generalization quality of a convolutional architecture is not limited to *real world data*, as our test purely uses numeric observations. [2] found that larger networks have a higher capacity to memorize training levels in the Gridworld domain. However, our results suggest that there may other effects at play than the parameter count and capacity. We also perform a memorization test by only showing g_θ's output to the policy, which is impossible to generalize to, as a policy cannot invert every single observation function $g_{\theta_1}(\cdot), g_{\theta_2}(\cdot), \ldots$ simultaneously. Using the underlying MDP as a Swimmer-v2 environment, we see that NatureCNN, IMPALA, IMPALA-LARGE have reduced memorization performances. [3] IMPALA-LARGE, which has more depth parameters and more residual layers (and thus technically has more capacity), memorizes *less* than IMPALA due its inherent inductive bias. We hypothesize that these extra residual blocks may be implicitly regularizing the network. This is corroborated by the fact that residual layers are also explained as an *implicit regularization* technique [25] for SL.

[3] Another memorization test can be found in Appendix A.1.2.

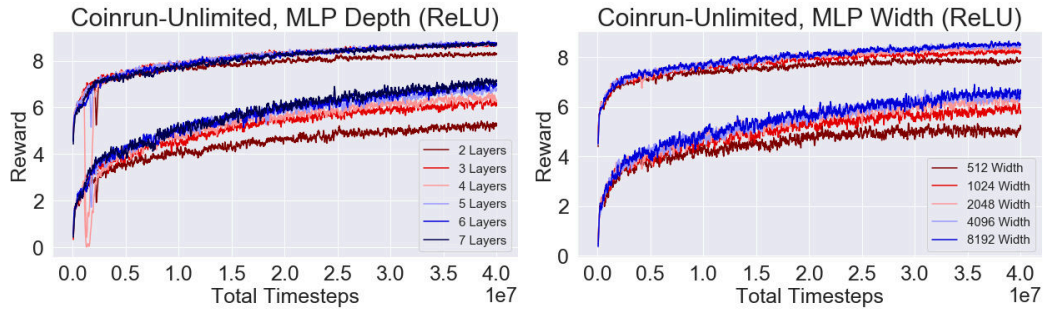
Figure 7: (Top) Performance of architectures in the synthetic Gym-Deconv dataset. (Bottom) We only show the observation from $g_\theta(s)$, which tests memorization capacity.



4 Overparametrization in CoinRun

We test our hypothesis from the above to the CoinRun benchmark, using unlimited levels for training. For MLP networks, we downsized CoinRun from native 64×64 to 32×32, and flattened the $32 \times 32 \times 3$ image for input to an MLP [4]. Two significant differences from previous cases are that 1. inherent dynamics are changing per level in CoinRun, and 2. the relevant and irrelevant CoinRun features change locations across the 1-D input vector. Regardless, we show that overparametrization can still improve generalization in this more realistic RL benchmark, much akin to [23] which showed that overparametrization for MLP's improved generalization on $32 \times 32 \times 3$ CIFAR-10.

Figure 8: Overparametrization improves generalization for CoinRun.



One may wonder how to predict the generalization gap only from the training phase. A particular set of metrics, popular in the SL community are *margin distributions* [37, 28], as they deal with the case for softmax outputs which do not explicitly penalize the weight norm of a network, by normalizing the "confidence" margin of the logit outputs. While using margins on state-action pairs (from an on-policy replay buffer) is not technically rigorous, one may be curious to see if they have predictive power, especially as MLP's are relatively simple to norm-bound. We plotted these margin distributions in Appendix A.2.2, but found that the weight norm bounds used in SL are simply too dominant for this RL case. This, with the bound results found earlier for the LQR case, suggests that current norm bounds are simply too loose for the RL case even though we have shown overparametrization helps generalization in RL, and hopefully this motivates more of the study of such theory.

[4] We also present results for ImageNet networks in Appendix A.2.1.

References

[1] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *CoRR*, abs/1806.07937, 2018.

[2] Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *CoRR*, abs/1804.06893, 2018.

[3] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in RL. *CoRR*, abs/1804.03720, 2018.

[4] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Procedural level generation improves generality of deep reinforcement learning. *CoRR*, abs/1806.10729, 2018.

[5] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2817–2826, 2017.

[6] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham M. Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6553–6564, 2017.

[7] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *J. Artif. Intell. Res.*, 61:523–562, 2018.

[8] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012.

[9] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. *CoRR*, abs/1811.11214, 2018.

[10] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *CoRR*, abs/1812.02341, 2018.

[11] Kavosh Asadi, Dipendra Misra, and Michael L. Littman. Lipschitz continuity in model-based reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 264–273, 2018.

[12] Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1181–1189, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[13] Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *CoRR*, abs/1811.06032, 2018.

[14] Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2063–2072, 2019.

[15] Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *CoRR*, abs/1907.02893, 2019.

[16] Christina Heinze-Deml and Nicolai Meinshausen. Conditional variance penalties and domain shift robustness. *CoRR*, abs/1710.11469, 2019.

[17] Christina Heinze-Deml, Jonas Peters, and Nicolai Meinshausen. Invariant causal prediction for nonlinear models. *CoRR*, abs/1706.08576, 2019.

[18] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1787–1796, 2018.

[19] Felipe Petroski Such, Vashisht Madhavan, Rosanne Liu, Rui Wang, Pablo Samuel Castro, Yulun Li, Ludwig Schubert, Marc G. Bellemare, Jeff Clune, and Joel Lehman. An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents. *CoRR*, abs/1812.07069, 2018.

[20] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5949–5958, 2017.

[21] Van H. Vu. Spectral norm of random matrices. *Combinatorica*, 27(6):721–736, 2007.

[22] Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *CoRR*, abs/1802.08760, 2018.

[23] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *CoRR*, abs/1805.12076, 2018.

[24] Huan Wang, Stephan Zheng, Caiming Xiong, and Richard Socher. On the generalization gap in reparameterizable reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6648–6658, 2019.

[25] Behnam Neyshabur. Implicit regularization in deep learning. *CoRR*, abs/1709.01953, 2017.

[26] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 244–253, 2018.

[27] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[28] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6240–6249. Curran Associates, Inc., 2017.

[29] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018.*, pages 297–299, 2018.

[30] Tengyuan Liang, Tomaso A. Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pages 888–896, 2019.

[31] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham M. Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6553–6564, 2017.

[32] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *CoRR*, abs/1803.07055, 2018.

[33] Krzysztof Choromanski, Mark Rowland, Vikas Sindhwani, Richard E. Turner, and Adrian Weller. Structured evolution with compact architectures for scalable policy optimization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 969–977, 2018.

[34] Adam Gaier and David Ha. Weight agnostic neural networks. *CoRR*, abs/1906.04358, 2019.

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[36] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1406–1415, 2018.

[37] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. *CoRR*, abs/1810.00113, 2018.

[38] Adam Santoro, Ryan Faulkner, David Raposo, Jack W. Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy P. Lillicrap. Relational recurrent neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 7310–7321, 2018.

[39] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

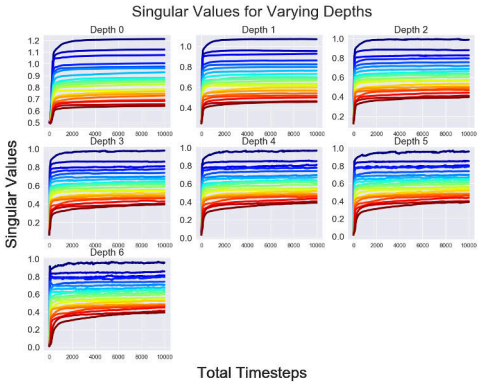
[40] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1466–1475, 2018.

[41] Roman Vershynin. Lectures in geometric functional analysis.

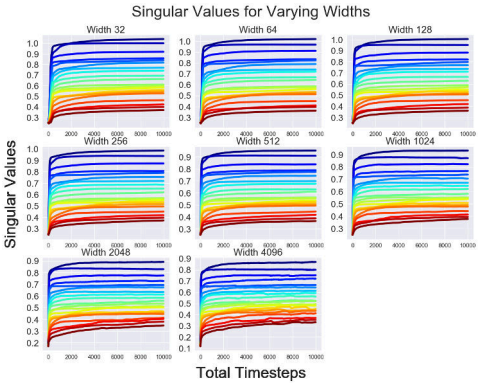
A.1 Full Plots for LQR, fg-Gym-MLP, fg-Gym-Deconv

A.1.1 LQR

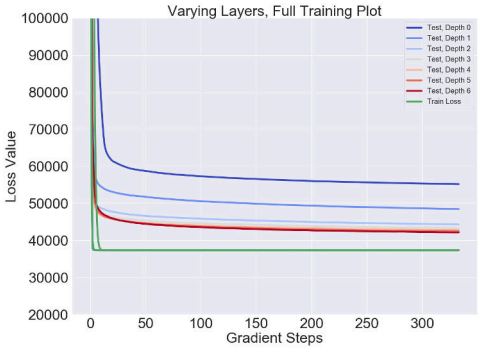
Figure A1: (a,b): Singular Values for varying depths and widths. (c,d): Train and Test Loss for varying widths and depths. (e): Train and Test Loss for varying Noise Dimensions.



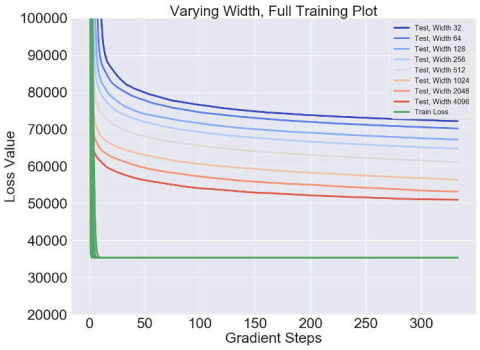
(a)



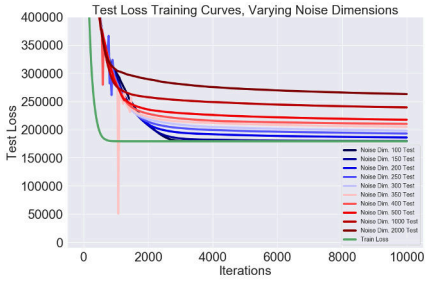
(b)



(c)



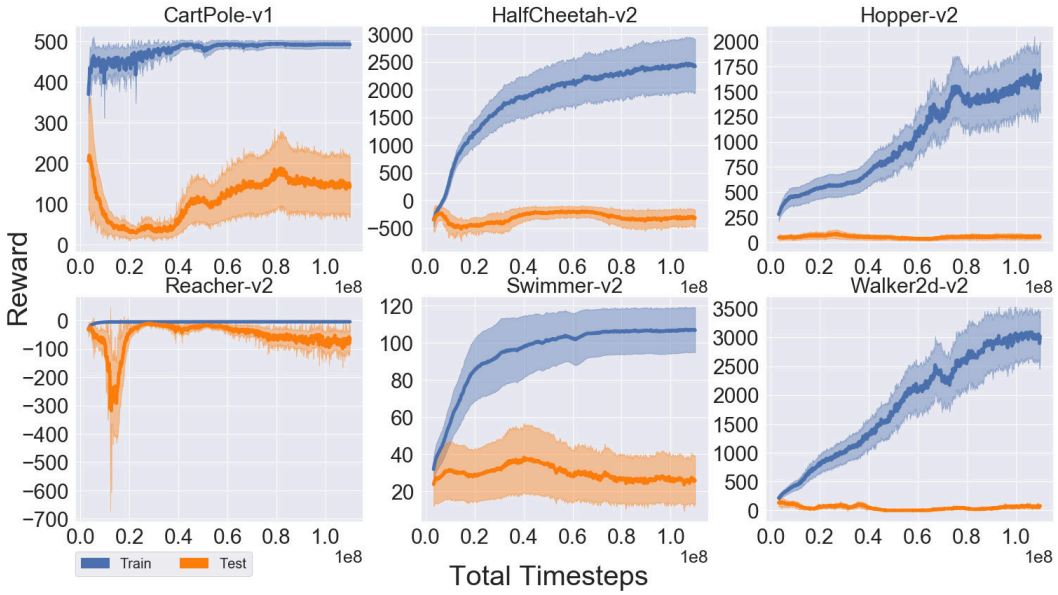
(d)



(e)

A.1.2 fg-Gym-MLP, Deconv

Figure A2: Each Mujoco task is given 10 training levels (randomly sampling g_θ parameters). We used a 2-layer Tanh policy, with 128 hidden units each. Dimensions of outputs of (f, g) were $(30, 100)$ respectively.



(a)

Figure A3: We further verify that *explicit regularization* (norm based penalties) also reduces generalization gaps, although explicit regularization may be explained by the bias of the synthetic tasks - The first layer's matrix may be regularized to only "view" the output of f, which significantly improves generalization.

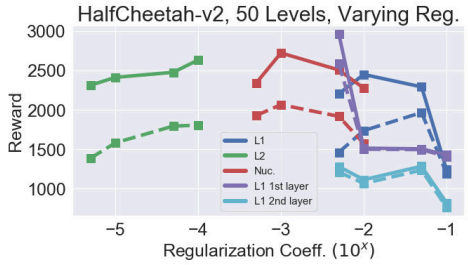
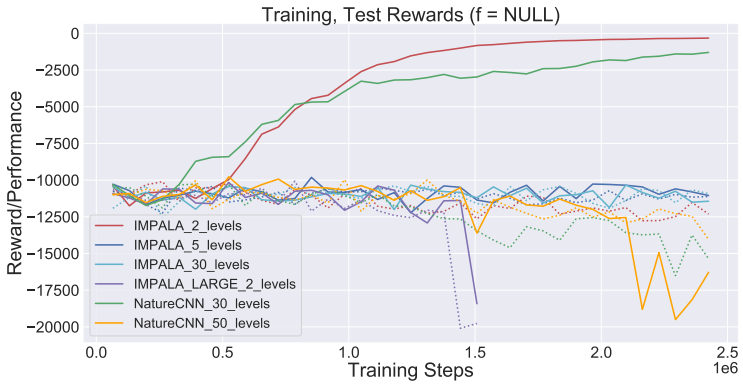


Figure A4: Another deconvolution memorization test, using an LQR as the underlying MDP. While fg-Gym-Deconv shows that memorization performance is dampened, this test shows that there can exist specific hard limits to memorization.



A.2 Extended Large RL Results

A.2.1 Large ImageNet Models for CoinRun

We experimentally verify in table (1) that large ImageNet models perform very differently in RL than during SL. We note that default network with the highest test reward was IMPALA-LARGE-BN (IMPALA-LARGE, with Batchnorm) at ≈ 5.5 test score.

In order to verify that this is inherently a *feature learning* problem rather than a *combinatorial problem* involving objects, such as in [38], we show that state-of-the-art attention mechanisms for RL such as Relational Memory Core (RMC) using pure attention on raw 32×32 pixels does not perform well here, showing that a large portion of generalization and transfer must be based on correct convolutional setups.

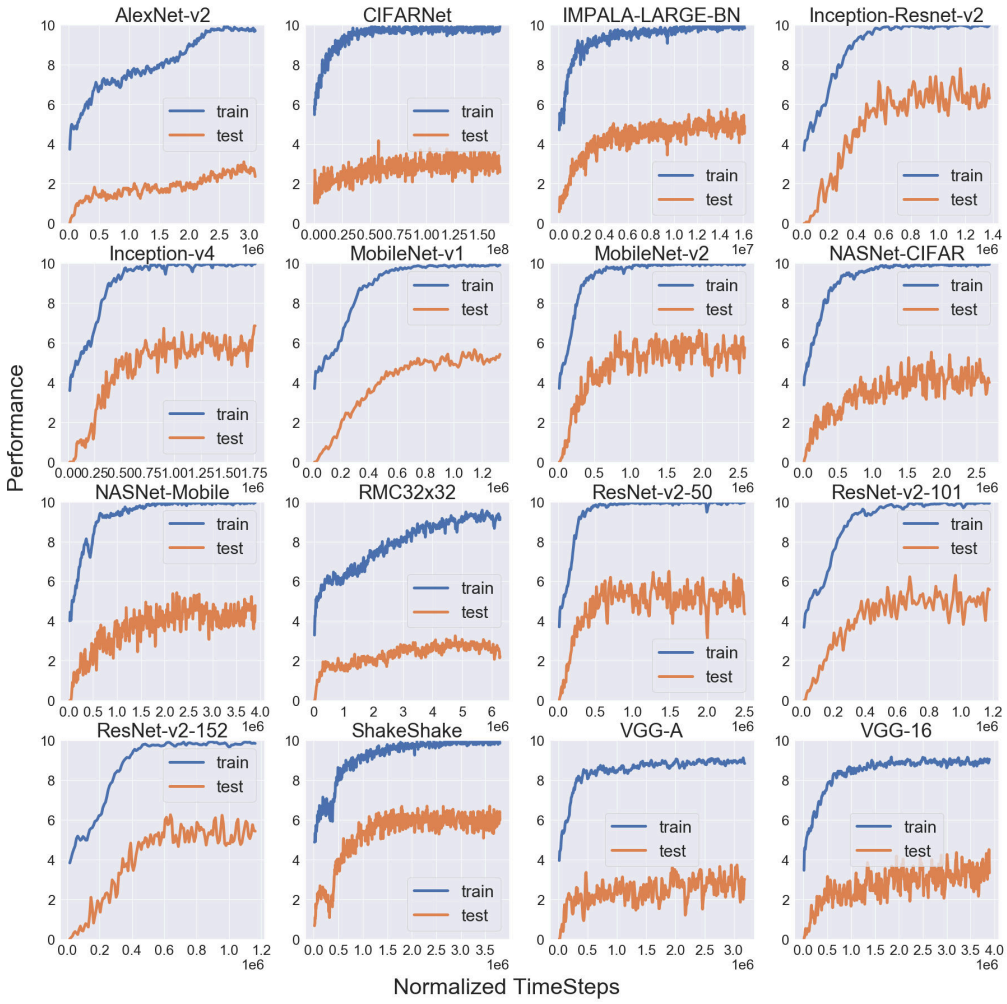
Architecture	Coinrun-100 (Train, Test)
AlexNet-v2	(10.0, 3.0)
CifarNet	(10.0, 3.0)
IMPALA-LARGE-BN	(10.0, 5.5)
Inception-ResNet-v2	(10.0, 6.5)
Inception-v4	(10.0, 6.0)
MobileNet-v1	(10.0, 5.5)
MobileNet-v2	(10.0, 5.5)
NASNet-CIFAR	(10.0, 4.0)
NASNet-Mobile	(10.0, 4.5)
ResNet-v2-50	(10.0, 5.5)
ResNet-v2-101	(10.0, 5.0)
ResNet-v2-152	(10.0, 5.5)
RMC32x32	(9.0, 2.5)
ShakeShake	(10.0, 6.0)
VGG-A	(9.0, 3.0)
VGG-16	(9.0, 3.0)

Table 1: Raw Network Performance (rounded to nearest 0.5) on CoinRun, 100 levels. Images scaled to default image sizes (32×32 or 224×224) depending on network input requirement. See Appendix A5 for training curves.

We provide the training/testing curves for the ImageNet/large convolutional models used. Note the following:

1. RMC32x32 projects the native image from CoinRun from 64×64 to 32×32, and uses all pixels as components for attention, after adding the coordinate embedding found in [38]. Optimal parameters were (mem_slots = 4, head_size = 32, num_heads = 4, num_blocks = 2, gate_style = 'memory').
2. Auxiliary Loss in ShakeShake was not used during training, only the pure network.
3. VGG-A is a similar but slightly smaller version of VGG-16.

Figure A5: Large Architecture Training/Testing Curves (Smoothed).



A.2.2 Do State-Action Margin Distributions Predict Generalization in RL?

A conceptual difference between CoinRun and our other tasks is due to the discrete action space of CoinRun. We verify in Figure A6, that indeed, simply measuring the raw norms of the policy network is a poor way to predict generalization, as it generally increases even as training begins to plateau. This is inherently because the softmax on the logit output does not penalize arbitrarily high logit values, and hence proper normalization is needed.

We are curious in measuring the *margin distribution* of action logits, as this has been used extensively to empirically predict the generalization properties of classifiers [37, 28]. For a policy, the the margin distribution will be defined as $(s, a) \rightarrow \frac{F_\pi(s)_a - \max_{i \neq y} F_\pi(x)_i}{\mathcal{R}_\pi \|S\|_2 / n}$, where $F_\pi(s)_a$ is the logit value of action a given input s, before the softmax, and S is the matrix of states in the replay buffer, and $\mathcal{R}_\pi$ is the norm-based Lipschitz bound on the policy network logits. We used the Spectral, Sharpness and

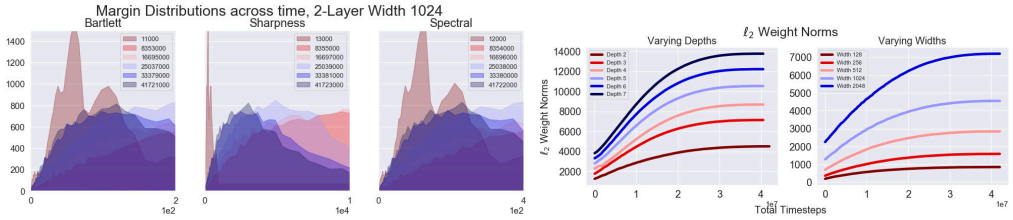
Bartlett bounds, for $\mathcal{R}_\pi$, and we replace the classical supervised learning pair $(x, y) = (s, a)$ with the state action pairs found on-policy.

We used the following metrics $\mathcal{R}_\pi$ (after removing irrelevant constants)

1. Bartlett Bound: $\left(\prod_{i=1}^{d} \|W_i\|\right)\left(\sum_{i=1}^{d} \frac{\|W_i\|_1^{2/3}}{\|W_i\|^{2/3}}\right)^{3/2}$ [28]
2. Sharpness Bound: $\sqrt{\frac{\sum_{i=1}^{d} \left\|W_i - W_i^0\right\|_F^2 + \ln(2m/\delta)}{m}}$ [39]
3. Spectral Bound: $\sqrt{\frac{\ln(d)\prod_{i=1}^{d}\|W_i\|_2^2 \sum_{j=1}^{d} \frac{\left\|W_j - W_j^0\right\|_F^2}{\|W_j\|_2^2} + \ln(\frac{6m}{\delta})}{m}}$ [27]

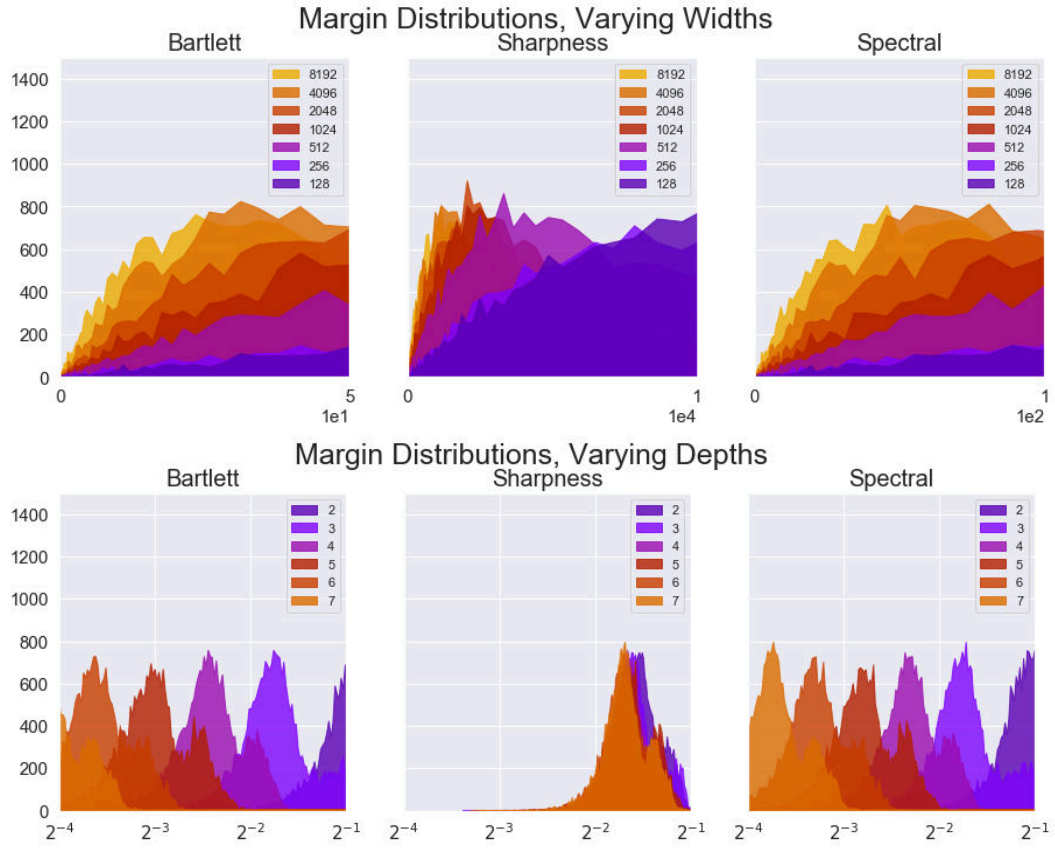
Unlike the other metrics mentioned, the margin distribution converges to a fixed distribution even long after training has plateaued. However, unlike SL, the margin distribution is conceptually not fully correlated with RL generalization on the total reward, as a policy overconfident in some state-action pairs does not imply bad testing performance. This correlation is stronger if there are Lipschitz assumptions on state-action transitions, as noted in [24]. For empirical datasets such as CoinRun, a metric-distance between transitioned states is ill-defined however. Nevertheless, the distribution over the on-policy replay buffer at each policy gradient iteration is a rough measure of overall confidence.

Figure A6: Margin Distributions at the end of training.



We note that there are two forms of modifications, *network dependent* (explicit modifications to the policy - norm regularization, dropout, etc.) and *data dependent* (modifications only to the data in the replay buffer - action stochasticity, data augmentation, etc.). Ultimately however, we find that current norm bounds R_π become too dominant in the fraction, leading to the monotonic decreases in the means of the distributions as we increase parametrization.

Figure A7: Margin Distributions at the end of training.



A.2.3 Gym-Retro (Sonic)

In the Gym-Retro benchmark (Sonic), the agent is given 47 training levels with rewards corresponding to increases in horizontal location. The policy is trained until 5k reward. At test time, 11 unseen levels are partitioned into starting positions, and the rewards are measured and averaged.

We briefly mention that the agent strongly overfits to the scoreboard (i.e. an artifact correlated with progress in the level), which may be interpreted as part of the output of $g_\theta(\cdot)$. In fact, the agent is still able to train to 5k reward from purely observing the timer as the observation. By blacking out this scoreboard with a black rectangle, we see an increase in test performance.

Settings	IMPALA	NatureCNN
Blackout	1250 ± 40	1141 ± 40
NoBlackout	1130 ± 40	1052 ± 40

Table 2: IMPALA vs NatureCNN test rewards, with and without Blackout. We refer the reader to see a saliency video of the agent, at `https://youtu.be/M-xf1YkHEn0`

A.3 Hyperparameters and Exact Setups

A.3.1 Exact infinite LQR

For infinite horizon case, see [40] for the the full solution and notations. Using the same notation (A, B, Q, R), denote $C(K) = \sum_{x_0 \sim \mathcal{D}} x_0^T P_K x_0$ as the cost and $u_t = -K s_t$ as the policy, where P_K satisifies the infinite algebraic-Ricatti equation:

$$P_K = Q + K^T R K + (A - BK)^T P_K (A - BK)$$

We may calculate the precise LQR cost by vectorizing (i.e. flattening) both sides and using the Kroncker product, which leads to a linear regression problem on P_K, which has a precise solution, implementable in TensorFlow:

$$\text{vec}(P_K) = \text{vec}(Q) + \text{vec}(K^T R K) + ((A - BK)^T \otimes (A - BK)^T)\,\text{vec}(P_K)$$
$$(I_{n^2} - ((A - BK)^T \otimes (A - BK)^T))\,\text{vec}(P_K) = \text{vec}(Q) + \text{vec}(K^T R K)$$

Parameter	Generation
A	Sampled uniformly randomly from set of orthogonal matrices on $n \times n$, scaled 0.99
B	I_n
Q	I_n
R	I_n
n	9

Table 3: Hyperparameters for LQR

A.3.2 Projection Method

The basis for producing f, g_θ outputs is due to using batch matrix multiplication operations, or "BMV", where the same network architecture uses different network weights for each batch dimension, and thus each entry in a batchsize of B will be processed by different network weights. This is to simulate the effect of g_{θ_i} - The numeric ID i of the environment is used as an index to collect a specficic set of network weights θ_i from a global memory of network weights (e.g. using `tensorflow.gather`). We did not use nonlinear activations for the BMV architectures, as they did not change the outcome of the results.

Architecture	Setup
BMV-Deconv	(filtersize = 2, stride = 1, outchannel = 8, padding = "VALID") (filtersize = 4, stride = 2, outchannel = 4, padding = "VALID") (filtersize = 8, stride = 2, outchannel = 4, padding = "VALID") (filtersize = 8, stride = 3, outchannel = 3, padding = "VALID")
BMV-Dense	f : Dense 30, g : Dense 100

A.3.3 ImageNet Models

For the networks used in the supervised learning tasks, we direct the reader to the following repository: `https://github.com/tensorflow/models/blob/master/research/slim/nets/nets_factory.py`. We also used the RMC: `deepmind/sonnet/blob/master/sonnet/python/modules/relational_memory.py`

A.3.4 PPO Parameters

For the projected gym tasks, we used for PPO2 Hyperparameters:

PPO2 Hyperparameters	Values
nsteps	2048
nenvs	16
nminibatches	64
λ	0.95
γ	0.99
noptepochs	10
entropy	0.0
learning rate	$3 \cdot 10^{-4}$
vf coeffiicent	0.5
max-grad-norm	0.5
total time steps	Varying

See [10] for the default parameters used for CoinRun. We only varied nminibatches in order to fit memory onto GPU.

A.4 Theoretical (LQR)

In this section, we use notation consistent with [40] for our base proofs. However, in order to avoid confusion with a high dimensional policy K we described in 3.1, we denote our low dimensional base policy as P and state as s_t rather than x_t.

A.4.1 Notation and Setting

Let $\|\cdot\|$ be the spectral norm of a matrix (i.e. largest singular value). Suppose $C(P)$ was the infinite horizon cost for an (A, B, Q, R)-LQR where action $a_t = -P \cdot s_t$, s_t is the state at time t, state transition is $s_{t+1} = A \cdot s_t + B \cdot a_t$, and timestep cost is $s_t^T Q s_t + a_t^T R a_t$.

$C(P)$ for an infinite horizon LQR, while known to be non-convex, still possess the property that when $\nabla C(P^*) = 0$, P^* is a global minimizer, or the problem statement is rank deficient. To ensure that our cost $C(P)$ always remains finite, we restrict our analysis when $P \in \mathcal{P}$, where $\mathcal{P} = \{P : \|P\| \leq \alpha \text{ and } \|A - BP\| \leq 1\}$ for some constant α, by choosing A, B and the initialization of P appropriately, using the hyperparameters found in A.3.1. We further define the observation modified cost as $C(K; W_\theta) = C\left(K \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}^\intercal\right)$.

A.4.1.1 Smoothness Bounds

As described in Lemma 16 of [40], we define

$$T_P(X) = \sum_{t=0}^{\infty} (A - BP)^t X [(A - BP)^T]^t \tag{1}$$

and $\|T_P\| = \sup_X \frac{T_P(X)}{\|X\|}$ over all non-zero symmetric matrices X.

Lemma 27 of [40] provides a bound on the difference $C(P') - C(P)$ for two different policies P, P' when LQR parameters A, B, Q, R are fixed. During the derivation, it states that when $\|P - P'\| \leq \min\left(\frac{\sigma_{min}(Q)\mu}{4C(P)\|B\|(\|A-BP\|+1)}, \|P\|\right)$, then:

$$\begin{aligned} C(P') - C(P) \leq 2\|T_P\|\left(2\|P\|\|R\|\|P' - P\| + \|R\|\|P' - P\|^2\right) + \\ 2\|T_P\|^2 2\|B\|\left(\|A - BP\| + 1\right)\|P - P'\|\|P\|^2\|R\| \end{aligned} \tag{2}$$

Lemma 17 also states that:

$$\|T_P\| \leq \frac{C(P)}{\mu\sigma_{min}(Q)} \tag{3}$$

where

$$\mu = \sigma_{min}(\mathbb{E}_{x_0 \sim D}[x_0 x_0^T]) \tag{4}$$

Assuming that in our problem setup, x_0, Q, R, A, B were fixed, this means many of the parameters in the bounds are constant, and thus we conclude:

$$C(P') - C(P) \leq \mathcal{O}\left(C(P)^2\left[\|P\|^2\|P - P'\|\left(\|A - BP\| + \|B\| + 1\right) + \|P\|\|P - P'\|^2\right]\right) \tag{5}$$

Since we assumed $\|A - BP\| \leq 1$ or else $T_P(X)$ is infinite, we thus collect the terms:

$$C(P') - C(P) \leq \mathcal{O}\left(C(P)^2\left[\|P\|^2\|P - P'\| + \|P\|\|P - P'\|^2\right]\right) \tag{6}$$

Since α is a bound on $\|P\|$ for $P \in \mathcal{P}$, note that

$$\|P\|^2\|P - P'\| + \|P\|\|P - P'\|^2 = \|P - P'\|\left(\|P\|^2 + \|P\| + \|P - P'\|\right) \tag{7}$$

$$\leq \|P - P'\| \left(\|P\|^2 + \|P\| \left(\|P\| + \|P'\|\right) \leq (3\alpha^2) \|P - P'\| \right. \tag{8}$$

From (6), this leads to the bound:

$$C(P') - C(P) \leq \mathcal{O}\left(C(P)^2 \|P - P'\|\right) \tag{9}$$

Note that this directly implies a similar bound in the high dimensional observation case - in particular, if $P = K \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}^\mathsf{T}$ and $P' = K \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}^\mathsf{T}$ then $\|P - P'\| \leq \|K - K'\| \left\| \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}^\mathsf{T} \right\| = \|K - K'\|$.

A.4.2 Gradient Dynamics in 1-Step LQR

We first start with a convex cost 1-step LQR toy example under this regime, which shows that linear components such as $\beta \begin{bmatrix} 0 \\ W_\theta \end{bmatrix}^\mathsf{T}$ cannot be removed from the policy by gradient descent dynamics to improve generalization. To shorten notation, let $W_c \in \mathbb{R}^{n \times n}$ and $W_\theta \in \mathbb{R}^{p \times n}$, where $p \ll n$. This is equivalent to setting $d_{signal} = d_{state} = n$ and $d_{noise} = p$, and thus the policy $K \in \mathbb{R}^{n \times (n+p)}$.

In the 1-step LQR, we allow $s_0 \sim \mathcal{N}(0, I)$, $a_0 = K \begin{bmatrix} W_c \\ W_\theta \end{bmatrix} s_0$ and $s_1 = s_0 + a_0$ with cost $\frac{1}{2} \|s_1\|^2$, then

$$C(K; W_\theta) = \mathbb{E}_{x_0} \left[\frac{1}{2} \left\| x_0 + K \begin{bmatrix} W_c \\ W_\theta \end{bmatrix} x_0 \right\|^2 \right] = \frac{1}{2} \left\| I + K \begin{bmatrix} W_c \\ W_\theta \end{bmatrix} \right\|_F^2 \tag{10}$$

and

$$\nabla C(K; W_\theta) = \left(I + K \begin{bmatrix} W_c \\ W_\theta \end{bmatrix} \right) \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}^\mathsf{T} \tag{11}$$

Define the population cost as $C(K) = \mathbb{E}_{W_\theta}[C(K; W_\theta)]$.

Proposition 1. *Suppose that $W_\theta \sim \mathrm{Unif}(O(p, n))$, where $O(p, n) = \{X \in \mathbb{R}^{p \times n} : X^\mathsf{T} X = I\}$. Then*

(i) The minimizer of $C(K)$ is unique and given by $K_\star = \begin{bmatrix} -W_c^\mathsf{T} & 0 \end{bmatrix}$.

(ii) Thus, the minimizer cost is $C(K_\star) = \frac{5n}{2}$.

Proof. By standard properties of the Haar measure on $O(p, n)$, we have that $\mathbb{E}[W_0] = 0$ and $\mathbb{E}[W_0 W_0^\mathsf{T}] = \frac{n}{p} I$. Therefore,

$$\begin{aligned} C(K) &= \frac{n}{2} + \frac{1}{2} \mathbb{E} \left[\left\| K \begin{bmatrix} W_c \\ W_\theta \end{bmatrix} \right\|_F^2 \right] + \mathbb{E} \left[\mathrm{Tr} \left(K \begin{bmatrix} W_c \\ W_\theta \end{bmatrix} \right) \right] \\ &= \frac{n}{2} + \frac{1}{2} \mathrm{Tr} \left(K^\mathsf{T} K \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p} I \end{bmatrix} \right) + \mathrm{Tr} \left(K \begin{bmatrix} W_c \\ 0 \end{bmatrix} \right) . \end{aligned}$$

We can now differentiate $C(K)$:

$$\nabla C(K) = K \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p} I \end{bmatrix} + \begin{bmatrix} W_c^\mathsf{T} & 0 \end{bmatrix} .$$

Both claims now follow. □

Note that if $K = K' + \beta \begin{bmatrix} 0 \\ W_\theta \end{bmatrix}^\mathsf{T}$, then $\nabla C(K; W_\theta) = \nabla C(K'; W_\theta) + \beta \begin{bmatrix} 0 \\ W_\theta \end{bmatrix}^\mathsf{T}$. In particular, if we perform gradient descent dynamics $K_{t+1} = K_t - \eta \nabla C(K; W_\theta)$, then we can show

$$K_t = K_0 (I - \eta M)^t + B (I - \eta M)^{t-1} + ... + B \tag{12}$$

where $M = \begin{bmatrix} W_c \\ W_\theta \end{bmatrix} \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}^\mathsf{T}$ is the Hessian of $C(K; W_\theta)$ and $B = -\eta \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}$. Note that M is has rank $p << n + p$, and thus at a high level, $K_0(I - \eta M)^t$ does not diminish if some additive portion of K_0 lies in the nullspace of M. If the initialization is $K_0 \sim \mathcal{N}(0, I)$, then it is highly likely we can find a subspace Q of the nullspace of M for which $K_0 Q \neq 0$.

We show the precise error achieved under gradient descent in the following propositions. Let $\overline{W} := \frac{1}{\sqrt{2}} \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}$. Observe that $\overline{W} \in O(n + p, n)$. Let $\overline{W}_\perp$ denote the orthogonal complement of $\overline{W}$.

Proposition 2. *As long as $\eta < 1/2$,*

$$K_\infty := \lim_{t \to \infty} K_t = K_0 \overline{W}_\perp \overline{W}_\perp^\mathsf{T} - \frac{1}{\sqrt{2}} \overline{W}^\mathsf{T} .$$

Proof. We first show that the series:

$$B(I - \eta M)^{t-1} + B(I - \eta M)^{t-2} + ... + B$$

converges to $-\frac{1}{2} \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}^\mathsf{T}$ as $t \to \infty$. Note that we can write:

$$I - \eta M = I - 2\eta \overline{W}\overline{W}^\mathsf{T} = (1 - 2\eta) \overline{W}\overline{W}^\mathsf{T} + \overline{W}_\perp \overline{W}_\perp^\mathsf{T} .$$

Therefore:

$$(I - \eta M)^t = (1 - 2\eta)^t \overline{W}\overline{W}^\mathsf{T} + \overline{W}_\perp \overline{W}_\perp^\mathsf{T} .$$

We can also write $B = -\sqrt{2}\eta \overline{W}$. Therefore, $B(I - \eta M)^t$ is:

$$B(I - \eta M)^t = -\sqrt{2}\eta (1 - 2\eta)^t \overline{W}^\mathsf{T} = -\eta (1 - 2\eta)^t \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}^\mathsf{T} .$$

This shows that $\sum_{t=0}^{\infty} B(I - \eta M)^t = -\frac{1}{2} \begin{bmatrix} W_c \\ W_\theta \end{bmatrix}^\mathsf{T}$ since $\eta \sum_{t=0}^{\infty} (1 - 2\eta)^t = \frac{1}{2}$. □

Proposition 3. *Suppose that $\eta < 1/2$ and each entry of K_0 is drawn i.i.d. from $\mathcal{N}(0, \psi^2)$. We have that:*

$$\mathbb{E}_{K_0}[C(K_\infty)] = \left(\frac{\psi^2 n^2}{4} + \frac{n}{16} \right) (1 + n/p) .$$

Proof. We write $K_\infty = K_{\infty,o} + K_{\infty,s}$. With this decomposition and letting $R_\perp := \overline{W}_\perp$,

$$\begin{aligned} K_\infty^\mathsf{T} K_\infty &= K_{\infty,o}^\mathsf{T} K_{\infty,o} + K_{\infty,o}^\mathsf{T} K_{\infty,s} + K_{\infty,s}^\mathsf{T} K_{\infty,o} + K_{\infty,s}^\mathsf{T} K_{\infty,s} \\ &= R_\perp K_0^\mathsf{T} K_0 R_\perp - \frac{1}{\sqrt{2}} R_\perp K_0^\mathsf{T} \overline{W}^\mathsf{T} - \frac{1}{\sqrt{2}} \overline{W} K_0 R_\perp + \frac{1}{2} \overline{W}\overline{W}^\mathsf{T} . \end{aligned}$$

Therefore,

$$\begin{aligned} \mathbb{E}_{K_0}[K_\infty] &= -\frac{1}{\sqrt{2}} \overline{W}^\mathsf{T} , \\ \mathbb{E}_{K_0}[K_\infty^\mathsf{T} K_\infty] &= \psi^2 n R_\perp + \frac{1}{2} \overline{W}\overline{W}^\mathsf{T} . \end{aligned}$$

We have:

$$\begin{aligned} \mathbb{E}_{K_0}[C(K_\infty)] &= \frac{n}{2} + \frac{1}{2} \operatorname{Tr}(\mathbb{E}_{K_0}[K_\infty^\mathsf{T} K_\infty] \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p} I \end{bmatrix}) + \operatorname{Tr}(\mathbb{E}_{K_0}[K_\infty] \begin{bmatrix} W \\ 0 \end{bmatrix}) \\ &= \frac{n}{2} + \frac{\psi^2 n}{2} \operatorname{Tr}(R_\perp \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p} I \end{bmatrix}) + \frac{1}{4} \operatorname{Tr}(\overline{W}\overline{W}^\mathsf{T} \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p} I \end{bmatrix}) - \frac{1}{\sqrt{2}} \operatorname{Tr}(\overline{W}^\mathsf{T} \begin{bmatrix} W \\ 0 \end{bmatrix}) \\ &= \frac{\psi^2 n}{2} \operatorname{Tr}(R_\perp \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p} I \end{bmatrix}) + \frac{1}{4} \operatorname{Tr}(\overline{W}\overline{W}^\mathsf{T} \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p} I \end{bmatrix}) \\ &= \frac{\psi^2 n}{2} \operatorname{Tr}(R_\perp \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p} I \end{bmatrix}) + \frac{1}{16} (n + n^2/p) . \end{aligned}$$

Next we have:

$$R_\perp \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p}I \end{bmatrix} = (I - \frac{1}{2}WW^\mathsf{T}) \begin{bmatrix} I & 0 \\ 0 & \frac{n}{p}I \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{2}I & -\frac{n}{p}WW_o^\mathsf{T} \\ -\frac{1}{2}W_oW^\mathsf{T} & \frac{n}{p}I - \frac{n}{2p}W_oW_o^\mathsf{T} \end{bmatrix} .$$

Therefore:

$$\mathbb{E}_{K_0}[C(K_\infty)] = \frac{\psi^2 n}{2}(\frac{n}{2} + \frac{n^2}{2p}) + \frac{1}{16}(n + n^2/p) .$$

□

Note that this is clearly a $poly(n)$ difference from $C(K_\star)$, and thus we show a generalization gap that cannot be removed from using standard gradient descent.

This is a specific example of the general case where if the Hessian of a function $f(x)$ is degenerate everywhere (e.g. has rank $k < n$), then a initialized x_0 cannot converge under gradient descent to a minimizer that lives in the span of the Hessian, as the non-degenerate components do not change. In particular, Proposition 4.7 in [41] points to the exact magnitude of the non-degenerate component in the relevant subspace Q: $\mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\|Proj_Q(x)\|^2\right] = \frac{n-k}{n}$

The generalization gap may decrease if the number of level samples was high enough. We define the sample Hessian as: $\widehat{C}(K) = \frac{1}{m}\sum_{i=1}^m C(K; W_{\theta_i})$ is $\widehat{M} = \frac{1}{m}\sum_{i=1}^m M_i$ where $M_i = \begin{bmatrix} W_c \\ W_{\theta_i} \end{bmatrix}\begin{bmatrix} W_c \\ W_{\theta_i} \end{bmatrix}^\mathsf{T}$ $\forall i$. In particular, as m increases, the rank of $\widehat{M}$ increases, which allows gradient descent to recover the minimizer $K_\star$ better.