On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks

Nicholas H. Barbara [1] **Ruigang Wang** [1] **Ian R. Manchester** [1]

Abstract

This paper presents a study of robust policy networks in deep reinforcement learning. We investigate the benefits of policy parameterizations that naturally satisfy constraints on their Lipschitz bound, analyzing their empirical performance and robustness on two representative problems: pendulum swing-up and Atari Pong. We illustrate that policy networks with small Lipschitz bounds are significantly more robust to disturbances, random noise, and targeted adversarial attacks than unconstrained policies composed of vanilla multilayer perceptrons or convolutional neural networks. Moreover, we find that choosing a policy parameterization with a non-conservative Lipschitz bound and an expressive, nonlinear layer architecture gives the user much finer control over the performance-robustness trade-off than existing state-of-the-art methods based on spectral normalization.

1. Introduction

Deep reinforcement learning (Deep RL) has been the driving force behind many recent successes in learning-based control, including in discrete game-like problems (Mnih et al., 2015), robotic manipulation (Kalashnikov et al., 2018), and locomotion (Rudin et al., 2021). However, the applicability of Deep RL to performance- and safety-critical systems is currently limited by questions of its robustness (Huang et al., 2017). It is well known that neural networks can be highly sensitive to small input perturbations, making policy networks learned via Deep RL potentially unrobust to disturbances, noise, and targeted adversarial attacks (Szegedy et al., 2013; Chakraborty et al., 2018). Despite sharing similar sensitivity issues to neural classifiers, for which many robust neural networks have recently been developed (Bernd & Lampert, 2022; Trockman & Kolter, 2021; Wang & Manchester, 2023), the use of robust policy architectures in Deep RL has not been widely studied.

Most common approaches to improving policy robustness in Deep RL are based on adversarial training, where adversarial attacks on a policy's inputs are optimized during training to encourage the network to perform well under perturbations (Pattanaik et al., 2017). While this works well in applications where the structure of the perturbations is always similar to those seen during training, adversarial training only certifies a lower bound on a policy's robustness. It is therefore possible to find new, out-of-sample attacks that cause the policy to fail (Russo & Proutiere, 2021). An alternative strategy is to learn control policies with a certified *upper* bound on their sensitivity to perturbations using methods like randomized smoothing (Kumar et al., 2022; Wu et al., 2022) and loss-function regularization (Oikarinen et al., 2021; Nie et al., 2024). These methods bound the sensitivity of a learned policy during the training process. It is interesting to ask whether, via careful choice of a policy's architecture and parameterisation, we can directly bound the sensitivity of a policy independently of how it is trained.

One promising approach is to use neural network policies that are certifiably robust to perturbations *by construction*. This can be achieved by constraining the Lipschitz bound of the network. A neural network $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to have an ℓ_2 Lipschitz bound of γ if

$$\|f(x_1) - f(x_2)\|_2 \leq \gamma \|x_1 - x_2\|_2, \quad \forall x_1, x_2 \in \mathbb{R}^n. \quad (1)$$

The true ℓ_2 Lipschitz constant $\mathrm{Lip}(f)$ is the smallest γ satisfying (1). Lipschitz-bounded networks are then "smoother" and less sensitive to input perturbations since small variations to their inputs will induce only small variations in their outputs. Despite the wealth of recent work on constructing Lipschitz-bounded deep networks (Bernd & Lampert, 2022; Trockman & Kolter, 2021; Wang & Manchester, 2023), to the best of our knowledge, the only method tested so far in Deep RL is spectral normalization (Bjorck et al., 2021; Takase et al., 2022) which is known to give conservative bounds on the true Lipschitz constant of the network. This paper therefore investigates the following questions:

1. Can Lipschitz-bounded networks improve the empirically-observed robustness of policies in deep RL?

[1]Australian Centre for Robotics, School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Sydney, NSW 2006, Australia.. Correspondence to: Nicholas H. Barbara <nicholas.barbara@sydney.edu.au>.

Workshop on Foundations of Reinforcement Learning and Control at the 41st International Conference on Machine Learning, Vienna, Austria.

2. If so, does the policy network architecture matter? That is, do more sophisticated policy parameterizations with less-conservative Lipschitz bounds give finer control over the performance-robustness trade-off?

We provide an initial empirical study of Lipschitz-bounded policy networks in Deep RL on two representative problems: pendulum swing-up, a classical benchmark problem in control and RL; and Atari Pong, a simple proxy for vision-based autonomous decision-making tasks. To the best of our knowledge, our comparison of Lipschitz-bounded policy architectures in Deep RL is the first of its kind.

2. Background and Prior Work

In this section, we first introduce the concepts of deep reinforcement learning and adversarial attacks. We then provide an overview of recent developments in robust neural networks with certifiable Lipschitz bounds.

2.1. Deep Reinforcement Learning

We introduce RL from a control-theoretic perspective (Fig. 1). Consider a discrete-time nonlinear dynamical system

$$x_{t+1} = f(x_t, u_t, w_t) \tag{2}$$

with state vector $x_t \in \mathbb{R}^n$, controlled inputs $u_t \in \mathbb{R}^m$, and disturbances $w_t \in \mathbb{R}^p$. The task is to learn feedback control policies of the form $u_t = \kappa(x_t; \theta)$ parameterized by $\theta \in \mathbb{R}^q$ which (locally and approximately) solve

$$\begin{aligned} &\max_{\theta} \; R := \mathbb{E}\left[\sum_{t=0}^{\infty} \rho^t r_t(x_t, u_t)\right] \\ &\text{s.t.} \;\; x_{t+1} = f(x_t, u_t, w_t), \;\; u_t = \kappa(x_t; \theta), \end{aligned} \tag{3}$$

where r_t is the instantaneous reward, $\rho \in (0, 1)$ is a "discount factor," R is the total expected reward, and the expectation $\mathbb{E}[\cdot]$ is taken over some known distributions of disturbances w_t and initial conditions x_0. In Deep RL, the controller κ is a deep neural network (DNN) parameterized by θ. In this paper we take Proximal Policy Optimization (PPO) (Schulman et al., 2017) as our training method due to its simplicity, speed, and performance.

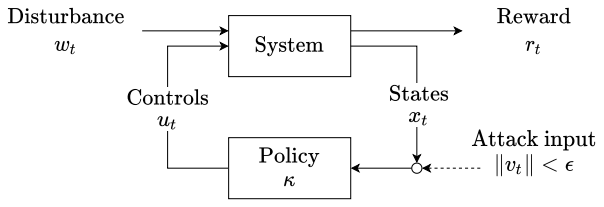


Figure 1: Reinforcement learning and adversarial attacks.

2.2. Adversarial Attacks for RL

Given a learned policy $\kappa(\cdot\,; \theta)$, an adversarial attack is an input sequence $v = (v_0, v_1, \ldots)$ with some restricted "attack size" $\epsilon > 0$ that is designed to reduce the expected cumulative reward of the policy as much as possible. While attacks can be designed to change the closed-loop behavior in many ways, the most common structure is an additive perturbation to the policy input (i.e., perturbations to the state measurements, Fig. 1) (Huang et al., 2017; Pattanaik et al., 2017). The problem can be formulated as

$$\begin{aligned} \min_{v} \; R \quad &\text{s.t.} \;\; x_{t+1} = f(x_t, \kappa(x_t + v_t; \theta), w_t), \\ &\|v_t\| \le \epsilon, \; \forall t, \end{aligned} \tag{4}$$

where $\|\cdot\|$ can be any p-norm with $1 \le p \le \infty$. If the reward function r_t and dynamic model f are both differentiable and known to the attacker, (4) can be solved by gradient descent methods. If part of the dynamical system is unknown or not differentiable, solving (4) is difficult or impossible and it is common instead to solve the simplified problem:

$$\max_{v_t} \quad \|\kappa(x_t + v_t; \theta) - \kappa(x_t; \theta)\| \quad \text{s.t.} \quad \|v_t\| \le \epsilon. \tag{5}$$

That is, find an admissible attack v_t which leads to large policy output perturbation at each time step, regardless of its effect on the final reward (Huang et al., 2017). Problem (5) can be solved by attack methods like Projected Gradient Descent (PGD) (Madry et al., 2018).

2.3. Lipschitz-Bounded Deep Networks

The adversarial attack problem in (5) is exactly the calculation of the (local) Lipschitz constant of the policy network κ. We can therefore control the effect of adversarial attacks everywhere in state space by bounding the global Lipschitz constant $\mathrm{Lip}(\kappa) \le \gamma$. Policy networks with smaller γ are then "smoother" and are likely to be more robust to small attacks than those with a large γ.

In deep RL, the policy network κ is often parameterized by a multi-layer perceptron (MLP) or a convolutional neural network (CNN) of the form of

$$\kappa = g_L \circ \sigma \circ g_{L-1} \circ \cdots \circ \sigma \circ g_1 \tag{6}$$

where σ is a fixed monotone and 1-Lipschitz scalar activation (e.g., ReLU, tanh, sigmoid, etc.) and g_k is a linear layer $g_k(x) = W_k x + b_k$ with W_k, b_k as the weights and biases, respectively. The Lipschitz constraint for (6) is then

$$\mathrm{Lip}(\kappa) = \sup_{J_2, \ldots, J_L} \|W_L J_L W_{L-1} \cdots J_2 W_1\|_2 \le \gamma \tag{7}$$

where J_k is a diagonal matrix with $0 \le J_{k,ii} \le 1$. Since it is NP-hard to compute the exact Lipschitz constant (Virmaux & Scaman, 2018), a practical approach is to find an upper

bound $\overline{\gamma}$ for $\text{Lip}(\kappa)$ and then impose the constraint $\overline{\gamma} = \gamma$. There are many bound estimation methods resulting in different constructions of Lipschitz networks. A metric to measure the expressive power of Lipschitz networks is *tightness*, which is defined as $\underline{\gamma}/\gamma$ (Wang & Manchester, 2023) with $\underline{\gamma}$ as the empirical lower Lipschitz bound:

$$\underline{\gamma} := \max_{x \in \mathbb{X}, \|v\|_2 \leq \epsilon} \frac{\|\kappa(x+v) - \kappa(x)\|_2}{\|v\|_2} \tag{8}$$

where $v \in \mathbb{R}^n$ and $\mathbb{X} \subset \mathbb{R}^n$ is some compact region. Note that $\text{Lip}(\kappa) \in [\underline{\gamma}, \gamma]$.

A simple Lipschitz bound estimation is the layer-wise spectral norm bound $\overline{\gamma}_s = \prod_{k=1}^{L} \|W_k\|_2$. This approach utilizes the fact that common activation functions σ are 1-Lipschitz. Thus, we can construct Lipschitz policy networks via scaling factors and 1-Lipschitz linear layers, where three representative examples are given as follows.

- **(SN)** The spectral normalization (SN) layer (Miyato et al., 2018) is a linear layer with weight $W = (1/\rho)A$ where A is a learnable weight, and ρ is its maximal singular value (i.e., $\|W\|_2 = 1$). Lipschitz networks composed of SN layers often have quite loose Lipschitz bounds such that $\underline{\gamma}/\gamma$ is very small.
- **(AOL)** The almost orthogonal Lipschitz (AOL) layer (Bernd & Lampert, 2022) is a linear layer with weight $W = AD$ where A is a learnable matrix and D is a diagonal matrix with $D_{ii} = \sqrt{\sum_j |A^\top A|_{ij}}$. The experimental results in Bernd & Lampert (2022) show that the learned weight W tends to have singular values close to 1, which helps to improve the model tightness.
- **(Cayley)** To obtain an even tighter Lipschitz bound, an orthogonal layer was proposed in Trockman & Kolter (2021) by leveraging the Cayley transform. Specifically, given a free learnable weight $P \in \mathbb{R}^{n \times n}$, one first obtains a skew symmetric matrix $A = P - P^\top$ and then constructs the weight matrix by $W = (I - A)(I + A)^{-1}$. It is easy to verify that $W^\top W = I$ and so all singular values of W are 1.

A much tighter bound estimation method was recently proposed in Fazlyab et al. (2019), which explores both the monotonicity and Lipschitz properties of the activation σ by leveraging the integral quadratic constraint (IQC) framework (Megretski & Rantzer, 1997). Direct (i.e. unconstrained) parameterizations based on IQC were proposed in Revay et al. (2020) for deep equilibrium networks, in Araujo et al. (2023) for residual networks, and in Wang & Manchester (2023) for deep MLPs and CNNs. In particular, Wang & Manchester (2023) shows that the IQC-based approach can lead to a 1-Lipschitz nonlinear layer as follows.

- **(Sandwich)** The Sandwich layer is a 1-Lipschitz nonlinear layer of the form

$$g(x) = \sqrt{2} A^\top \Psi \sigma\big(\sqrt{2}\Psi^{-1} B x + b\big) \tag{9}$$

where $Q = \begin{bmatrix} A & B \end{bmatrix}$ is a semi-orthogonal matrix ($QQ^\top = I$) parameterized by the Cayley transformation, and Ψ is a positive diagonal matrix parameterized via an exponential mapping. Based on the above 1-Lipschitz layer, we can construct the γ-Lipschitz policy network κ via

$$\kappa(x) = \sqrt{\gamma} g_L \circ g_{L-1} \circ \cdots \circ g_1(\sqrt{\gamma} x). \tag{10}$$

Note that (10) can be transformed back to the MLP form (6). In Wang & Manchester (2023), experimental results on image datasets show that (10) can achieve better performance than 1-Lipschitz linear layers such as AOL (Bernd & Lampert, 2022) and Cayley (Trockman & Kolter, 2021). It is natural to ask whether similar results hold true for Deep RL.

3. Experimental Setup

We study two classic RL problems — pendulum swing-up and Atari Pong — to investigate the research questions outlined in Section 1. Our code and training details are available on GitHub[1,2] and in Appendix A, respectively.

Pendulum Swing-up aims to swing a physical pendulum to its upright equilibrium based on the quadratic reward $r_t = -(\alpha_t^2 + 0.1\dot{\alpha}_t^2 + 0.001 u_t^2)$, where α_t is the pendulum angle (wrapped to $[-\pi, \pi]$) and u_t is the pendulum torque. The optimal policy is well-known to have sharp decision boundaries which make it susceptible to chattering and instability under small measurement perturbations, delays, or uncertainty. We trained MLP policies (6) without any bounds on the network Lipschitz constant, and Lipschitz-bounded policies using the Sandwich parameterization (9). We investigated the robustness of each policy to two sources of perturbations: a) sample delays; and b) adversarial attacks with constrained ℓ_2 norm. Adversarial attacks were computed by solving (4) over a sequence of four 50-sample windows with gradient descent.

Atari Pong is a video game in which two players each control a paddle that can move up and down and try to deflect a puck into their opponent's goal (Fig. 6). The reward is the net game score, with a maximum score of 21 goals to nil. An automated "computer" player controls the left paddle and the RL policy controls the right paddle. It is a commonly-studied benchmark in Deep RL (Mnih et al., 2015; Russo & Proutiere, 2021). The game can be written as an RL problem (3) where the states x_t are grayscale

[1] https://github.com/nic-barbara/Lipschitz-RL-MJX
[2] https://github.com/nic-barbara/Lipschitz-RL-Atari

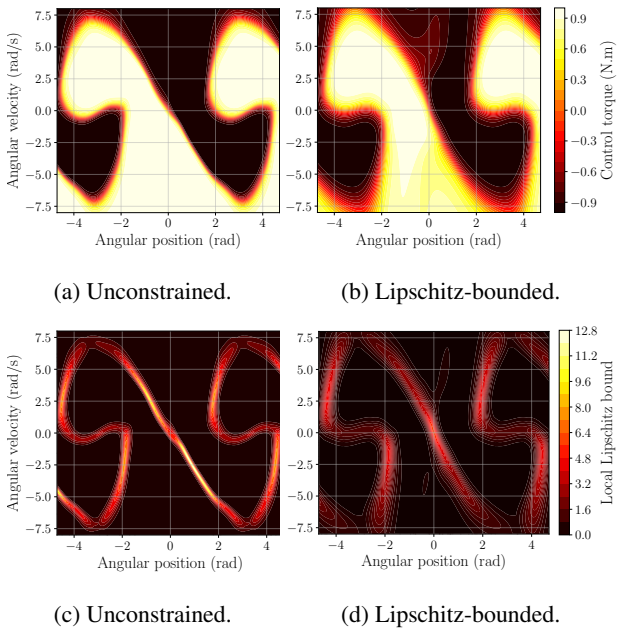
gameplay images and the control actions u_t are discrete paddle movements. We trained a classic, unconstrained CNN model (6) and four different Lipschitz-bounded policy architectures (SN, AOL, Sandwich, Cayley) with various Lipschitz bounds. We compared the robustness of each policy network to: a) uniform random noise; b) PGD attacks (5) with constrained ℓ_2 norm; and c) PGD attacks (5) with constrained ℓ_∞ norm.

4. Results and Discussion

We first study the advantages of Lipschitz-bounded policy networks in terms of robustness to perturbations and adversarial attacks, using pendulum swing-up as an illustrative example (Sec. 4.1). We extend this study to vision-based feedback control in Pong (Sec. 4.2) and compare the benefits of different Lipschitz-bounded policy architectures. In Figures 4 and 5 and Table 1, Lipschitz lower bounds $\underline{\gamma}$ were computed for each policy by performing gradient ascent on the model inputs to maximize the local Lipschitz constant.

4.1. Illustrative Example — Pendulum Swing-up

Let us first consider the effect of small Lipschitz bounds on unperturbed policy networks. Figure 2 compares the policy landscape and (empirically-estimated) local Lipschitz constant in phase space for an unconstrained MLP policy and a Lipschitz-bounded policy composed of Sandwich layers with $\gamma = 4$. The unconstrained policy has sharp decision boundaries, either side of which is flips the sign of the control torque (limited to ± 1 N.m). While these sharp changes are optimal in the unperturbed case, it is clear that any small uncertainty in the pendulum's position or velocity will cause the network to apply a drastically different control action, potentially driving the system to instability. In contrast, imposing a bound on the Lipschitz constant with Sandwich layers visibly smooths the decision boundaries with negligible penalty to the final test reward (-153 for unconstrained and -157 for Lipschitz-bounded, see Figure 8).



(a) Unconstrained. (b) Lipschitz-bounded.

(c) Unconstrained. (d) Lipschitz-bounded.

Figure 2: Contours in phase space of control actions (a,b) and local Lipschitz bounds (c,d) for an unconstrained (MLP) and a Lipschitz-bounded (Sandwich, $\gamma = 4$) policy show how Lipschitz bounds control a policy's smoothness.

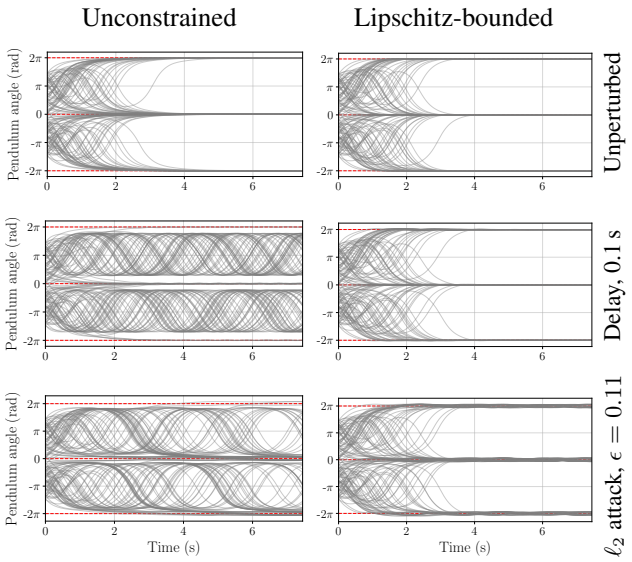
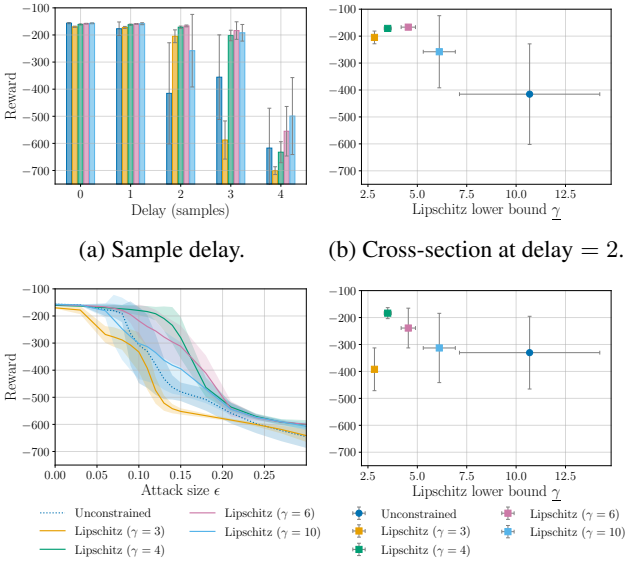


Figure 3: Pendulum trajectories generated by unconstrained (MLP) and Lipschitz-bounded (Sandwich, $\gamma = 4$) policies in nominal operation, with sample delays, and with ℓ_2 adversarial attacks. Red dashed lines indicate the target.

Figure 3 illustrates the effect of this smoothing on each policy's robustness to sample delays and ℓ_2-optimal adversarial attacks. The unperturbed trajectories in Figure 3 indicate that both the unconstrained and Lipschitz-bounded policy networks perform similarly in nominal operation. However, when introducing a small sample delay (2 time samples, 0.1 s) or a small adversarial attack ($\epsilon = 0.11$), the unconstrained policy is unable to hold the pendulum stable and upright, whereas the Lipschitz-bounded policy is successful and only exhibits minor oscillations about the equilibrium under adversarial attacks.

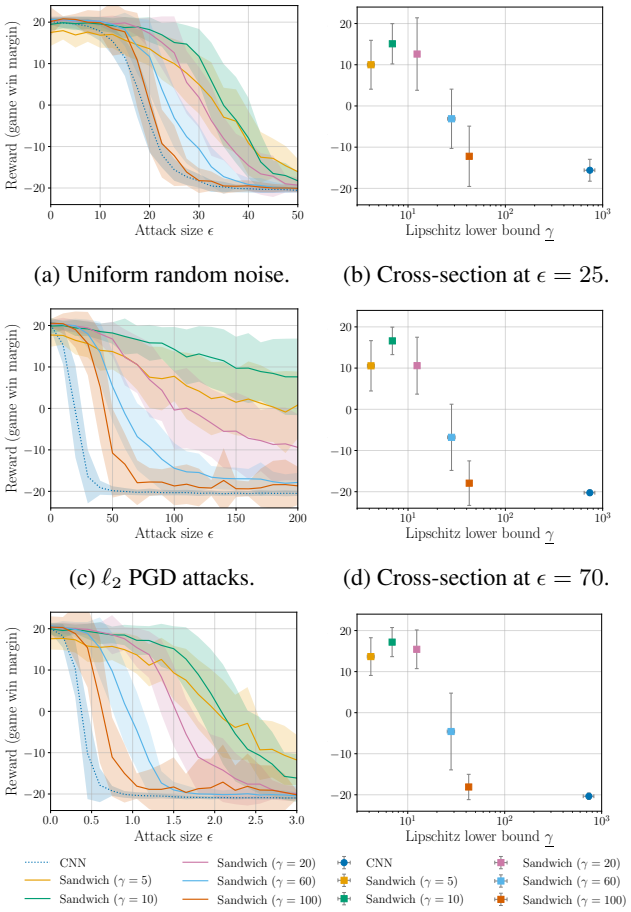
We find that this improvement in robustness is highly correlated with the policy's Lipschitz bound. Comparing unconstrained policies to Lipschitz-bounded policies with various γ in Figure 4, there is a smooth transition from high to low robustness to sample delays and ℓ_2-optimal adversarial



(a) Sample delay. (b) Cross-section at delay $= 2$.

(c) ℓ_2-optimal attacks. (d) Cross-section at $\epsilon = 0.11$.

Figure 4: Robust performance of unconstrained (MLP) and Lipschitz-bounded (Sandwich) policies on pendulum swing-up under sample delays and ℓ_2-optimal adversarial attacks. Panels (b,d) show cross-sections of (a,c) as a function of each model's empirically-estimated Lipschitz lower bound. Bands and error bars show one standard deviation over 10 random model initializations. Sample time is 0.05 s.



(a) Uniform random noise. (b) Cross-section at $\epsilon = 25$.

(c) ℓ_2 PGD attacks. (d) Cross-section at $\epsilon = 70$.

(e) ℓ_∞ PGD attacks. (f) Cross-section at $\epsilon = 1.05$.

Figure 5: Robust performance of unconstrained (CNN) and Lipschitz-bounded (Sandwich) policies for Atari Pong. Bands and error bars show one standard deviation over 4 random model initializations.

attacks as the policy's Lipschitz bound increases. Interestingly, it appears that there is a "best choice" for γ, and that restricting it to very small values ($\gamma = 3$) harms the closed-loop performance. This is to be expected, since the optimal policy for pendulum swing-up is known to be non-smooth, hence with excessive regularization it is likely that the network's parameter space does not contain any high-performing policies. Aside from the policies with $\gamma = 3$, all models were successfully trained to approximately the same final reward, as shown in Figure 8. It is therefore clear from Figures 2 to 4 that, at least in the context of pendulum swing-up, Lipschitz-bounded policy networks significantly improve robustness to disturbances and adversarial attacks over standard, unconstrained networks.

4.2. Comparing Architectures — Atari Pong

In the game of Pong, we expect small amounts of image noise to make very little difference to the state of the game and the optimal action. We would therefore hope that smooth, Lipschitz-bounded policies can improve robustness to perturbations like random noise and adversarial attacks. We see this immediately in Figure 5, where we compare the robustness of unconstrained CNN policies and Lipschitz-bounded Sandwich policies to uniform random noise, ℓ_2 PGD attacks, and ℓ_∞ PGD attacks. The same qualitative results observed for the pendulum in Figure 4 can be seen in Figure 5: there is a smooth transition from high to low robustness as γ decreases; robustness is improved not just for ℓ_2-constrained attacks (which we expect for policies with a small ℓ_2 Lipschitz bound), but also for random noise and ℓ_∞-constrained attacks; and if γ is too small (e.g., $\gamma = 5$ here), the policy's nominal performance and robustness to perturbations is degraded. In this case, it is possible that the $\gamma = 5$ models have not finished training and could perform better if trained over more epochs (Fig. 7d).

It is interesting to look deeper into the effect of adversarial attacks on these models. Figure 6 shows just how much of an improvement Lipschitz-bounded policy networks provide in Pong over a standard, unconstrained CNN. The CNN loses the game when subject to very small amounts

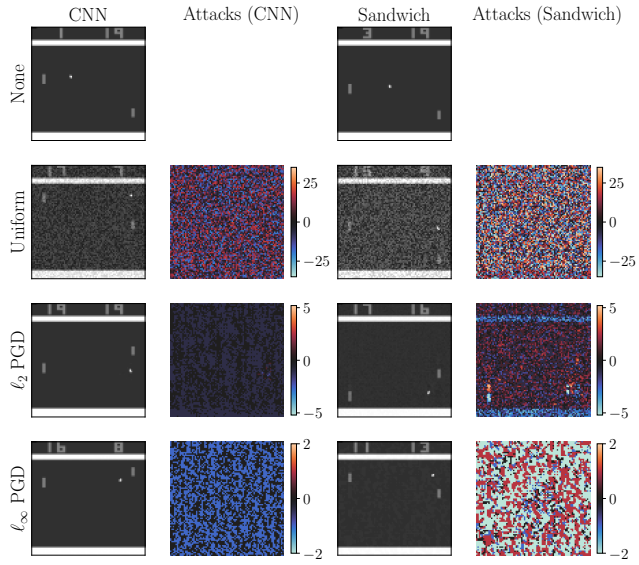
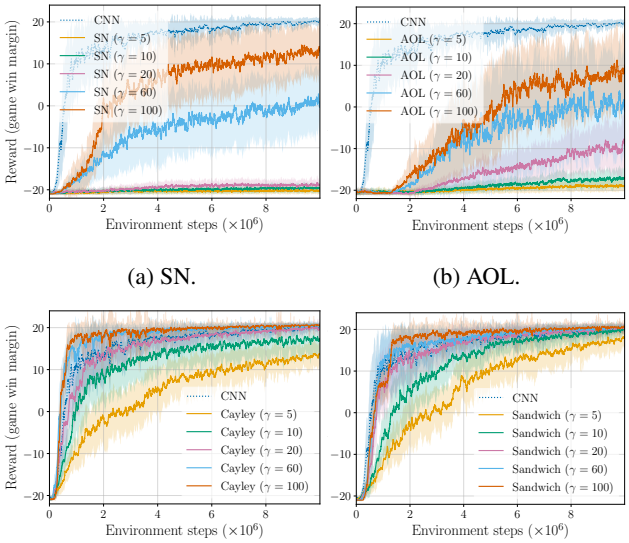


Figure 6: Adversarial examples showing the smallest attacks required to make an unconstrained (CNN) and a Lipschitz-bounded (Sandwich, $\gamma = 10$) policy lose the game. In each image, the "computer" controls the paddle on the left, while the policy controls the paddle on the right. All attacked frames show scenarios where the policy is about to concede a goal (puck moving to the right).



(a) SN. (b) AOL.

(c) Cayley. (d) Sandwich.

Figure 7: Test reward during training on Atari Pong for all layer architectures. Bands show one standard deviation over 4 random model initializations.

of random noise and almost imperceptible adversarial attacks. In contrast, the Lipschitz-bounded policy is only beaten with a level of random noise that would even make the game difficult for a human. Moreover, highly-structured ℓ_2-constrained attacks are required to beat the Lipschitz-bounded policy. Looking closely at Figure 6, successful ℓ_2 PGD attacks try to trick the policy into thinking the opponent's paddle is higher than it actually is while also trying to hide the exact location of the puck. The ℓ_2 attacks also seem to focus on the white boundary walls of the game. It is less clear why the policies should be sensitive to these features, but we hypothesize that the straight lines of the walls may appear similar to a paddle in feature space after passing through convolutional layers. There is no clear structure to the ℓ_∞ PGD attacks and they remain rather small, since the Lipschitz-bounded policies have a constrained ℓ_2 Lipschitz bound, and the ℓ_2 norm is only a loose upper bound of the ℓ_∞ norm in high-dimensional spaces. The additional robustness to ℓ_∞ PGD attacks over CNN policies is a nice bonus.

So far, we have compared Lipschitz-bounded policy networks constructed from Sandwich layers with unconstrained networks to illustrate that smoother policies can improve robustness in Deep RL. It turns out that in addition to tuning its upper bound γ, the layer architecture we use to bound the Lipschitz constant $\mathrm{Lip}(\kappa)$ of a policy κ is extremely important. Figure 7 compares the training curves of all four Lipschitz-bounded policy architectures introduced in Section 2.3 with unconstrained CNN policies, while Table 1 summarizes their nominal and robust performance. It is immediately clear from Figures 7a and 7b that the two layer architectures known to have conservative bounds on $\mathrm{Lip}(\kappa)$, SN and AOL, perform poorly when γ is small. When $\gamma \leq 20$, neither architecture ever produces a winning policy. For larger γ, the SN and AOL policies can learn to win the game, but the training dynamics are extremely slow in comparison to the CNN (this is a known problem for AOL, see Bernd & Lampert (2022, Sec. 7)). Interestingly, Table 1 shows that the estimated lower bound $\underline{\gamma}$ on $\mathrm{Lip}(\kappa)$ for these policies is very small when γ is small. This suggests that the conservative parameterization of SN and AOL layers restricts their parameter spaces to a small set of very smooth models which does not include high-performing policy networks. In contrast, Figures 7c and 7d show that the two layer architectures with much tighter bounds on $\mathrm{Lip}(\kappa)$, Cayley and Sandwich, perform quite well even for small choices of γ. The choice of γ still seems to have a strong impact on the training dynamics for these layers — as γ increases, so too does the speed at which the policies converge on a winning strategy. This raises interesting questions about the coupling between a policy's Lipschitz constant and the exploration of its parameter space, which we leave for future work.

Looking closer at Table 1 reveals that simply having a tight Lipschitz bound (i.e., close lower and upper bounds $\underline{\gamma}$ and γ)

Table 1: Averaged results (4 random model initialisations and 20 random game and attack seeds) on performance and robustness of policy networks trained on Atari Pong. γ is the certified Lipschitz upper-bound for a network and $\underline{\gamma}$ is its empirically-estimated lower-bound. The unperturbed reward is the final mean test reward achieved during training. Results for each attack strategy are the smallest average attack size ϵ required to beat the policy (i.e., reward < 0). Attack results are not provided for policies that did not learn a positive reward. **Bold** values indicate the overall best performing models in each column. Values with an * indicate the best performing models for each γ.

Policy	**Lipschitz**		**Reward**	**Smallest winning attack size ϵ (↑)**		
	γ	$\underline{\gamma}$	Unperturbed	Uniform	ℓ_2 PGD	ℓ_∞ PGD
CNN	-	738	20.1	18.7	19.5	0.38
SN	5	1.14	-20.2	-	-	-
	10	2.78	-19.4	-	-	-
	20	6.05	-18.7	-	-	-
	60	24.7	1.21	15.0	9.80	0.21
	100	42.4	14.8	21.5	21.3	0.49
AOL	5	1.40	-18.9	-	-	-
	10	2.95	-16.9	-	-	-
	20	7.62	-9.28	-	-	-
	60	15.3	1.31	12.4	19.2	0.52
	100	17.4	8.85	23.6*	42.3	1.20*
Cayley	5	4.56	13.6	23.7	106	1.52
	10	8.99	17.8	28.7	154	1.59
	20	16.6	20.0	23.1	129*	1.36
	60	34.8	**20.6***	20.9	67.4*	0.98*
	100	47.0	20.4*	17.1	54.0*	0.76
Sandwich	5	4.16	17.5*	33.4*	183*	2.01*
	10	6.90	19.5*	**35.0***	**> 200***	**2.08***
	20	12.4	20.2*	30.8*	99.2	1.53*
	60	28.0	**20.6***	23.9*	58.1	0.94
	100	42.5	20.2	20.1	43.3	0.63

is not the only factor contributing to a policy's performance and robustness. Instead, there appears to be an advantage to using "expressive" policy networks, particularly when high robustness (small γ) is required.Table 1 indicates that both the Sandwich and Cayley policies exhibit strong performance and robustness for $\gamma \geq 20$. Either layer architecture is therefore a suitable choice for reasonable improvement over existing methods in robust RL like SN. For smaller $\gamma < 20$, however, Sandwich policies are far superior, and with $\gamma = 10$ they are the most robust of any policy across all three input perturbations while still achieving a strong unperturbed reward of 19.6. This is despite the fact that Cayley policies often have a tighter Lipschitz bound than the Sandwich policies (take $\gamma = 10$ as an example). We suggest that this is due to the less conservative parameterization of the Sandwich layers. Each of the SN, AOL, and Cayley policies are composed of linear layers with a spectral norm of approximately 1 (exactly 1 for Cayley). In contrast, Sandwich layers are nonlinear, have no direct restriction on their spectral norm (Wang & Manchester, 2023, Fig. 4), and instead constrain $\mathrm{Lip}(\kappa)$ via a composition of nonlinear layers which are a complete parameterization of the tightest known bounds on the Lipschitz constant of DNNs (Fazlyab et al., 2019). This allows the Sandwich models to converge on policy networks that are both performant and robust even when their parameter space is restricted by a small value of γ, allowing finer control over the performance-robustness trade-off in Deep RL.

5. Conclusions

This paper has studied the robustness benefits of Lipschitz-bounded policy networks in Deep RL. We have found that policy networks with small Lipschitz bounds are significantly more robust to perturbations such as disturbances, random noise, and targeted adversarial attacks. Moreover, we have observed that choosing a policy with non-conservative Lipschitz bounds and an expressive, nonlinear layer architecture gives the user finer control over the performance-robustness trade-off than existing methods based on spectral normalization. This raises interesting questions for future study, such as whether Lipschitz-bounded policy networks can augment or alleviate the need for adversarial training, and whether the observed benefits can be transferred to real-world robotic systems.

References

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., and Bachem, O. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021.

Araujo, A., Havens, A. J., Delattre, B., Allauzen, A., and Hu, B. A unified algebraic perspective on Lipschitz neural networks. In *International Conference on Learning Representations*, 2023.

Bernd, C. H. P. and Lampert. Almost-orthogonal layers for efficient general-purpose lipschitz networks. *Computer Vision – ECCV 2022*, pp. 350–365, 2022.

Bjorck, J., Gomes, C. P., and Weinberger, K. Q. Towards deeper deep reinforcement learning. *Advances in Neural Information Processing Systems*, 6 2021.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

Ding, G. W., Wang, L., and Jin, X. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. J. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax – a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 6 2021.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *International Conference on Learning Representations*, 2017.

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

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. Scalable deep reinforcement learning for vision-based robotic manipulation. *Proceedings of The 2nd Conference on Robot Learning*, pp. 651–673, 10 2018.

Kumar, A., Levine, A., and Feizi, S. Policy smoothing for provably robust reinforcement learning. *International Conference on Learning Representations*, 2022.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

Megretski, A. and Rantzer, A. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6):819–830, 1997.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518: 529–533, 2 2015.

Nie, B., Ji, J., Fu, Y., and Gao, Y. Improve robustness of reinforcement learning against observation perturbations via ℓ_∞ lipschitz policy networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38:14457–14465, 3 2024. ISSN 2374-3468.

Oikarinen, T., Zhang, W., Meche, M., Megretski, A., Daniel, L., and Weng, T.-W. Robust deep reinforcement learning through adversarial loss. *Advances in Neural Information Processing Systems*, 34:26156–26167, 12 2021.

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. Robust deep reinforcement learning with adversarial attacks. *International Joint Conference on Autonomous Agents and Multiagent Systems*, 3:2040–2042, 12 2017.

Revay, M., Wang, R., and Manchester, I. R. Lipschitz bounded equilibrium networks. *arXiv:2010.01732*, 2020.

Rudin, N., Hoeller, D., Reist, P., and Hutter, M. Learning to walk in minutes using massively parallel deep reinforcement learning. *Proceedings of Machine Learning Research*, 164:91–100, 9 2021.

Russo, A. and Proutiere, A. Towards optimal attacks on reinforcement learning policies. *American Control Conference*, pp. 4561–4567, 5 2021.

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

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *International Conference on Learning Representations*, 2013.

Takase, R., Yoshikawa, N., Mariyama, T., and Tsuchiya, T. Stability-certified reinforcement learning control via spectral normalization. *Machine Learning with Applications*, 10:100409, 12 2022.

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

Trockman, A. and Kolter, J. Z. Orthogonalizing convolutional layers with the cayley transform. *International Conference on Learning Representations*, 2021.

Virmaux, A. and Scaman, K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

Wang, R. and Manchester, I. Direct parameterization of lipschitz-bounded deep networks. *International Conference on Machine Learning*, 2023.

Weng, J., Lin, M., Huang, S., Liu, B., Makoviichuk, D., Makoviychuk, V., Liu, Z., Song, Y., Luo, T., Jiang, Y., Xu, Z., and Yan, S. EnvPool: A highly parallel reinforcement learning environment execution engine. *Advances in Neural Information Processing Systems*, 2022.

Wu, F., Li, L., Huang, Z., Vorobeychik, Y., Zhao, D., and Li, B. Crop: Certifying robust policies for reinforcement learning through functional smoothing. *International Conference on Learning Representations*, 2022.

A. Training Details

A.1. Pendulum Swing-up Experiments

The pendulum was modeled in MuJoCo XLA (MJX), a fully-differentiable implementation of the MuJoCo physics simulator (Todorov et al., 2012) written in JAX (Bradbury et al., 2018) that allows users to take gradients through the entire closed-loop system. We trained unconstrained MLP and Lipschitz-bounded (Sandwich) policies using the PPO implementation in Freeman et al. (2021), which leverages the scalability of JAX to massively multi-thread parallel physics simulations on a single workstation GPU. Hyperparameters were tuned by varying each parameter one at a time and choosing the best-performing parameters for an unconstrained MLP. The same hyperparameters were used to train Lipschitz-bounded policies without any further tuning. Our chosen hyperparameters can be found on our GitHub repository[3]. We trained 10 policies for each model architecture and choice of γ, each with a different random seed for model initialization. Results showing robust performance to sample delays and adversarial attacks in Figure 4 were averaged over the 10 policies and 1024 pendulum environments starting from random initial states. MLP networks were composed of 4 linear layers of 32 hidden nodes. Lipschitz-bounded policies were composed of 4 Sandwich layers of 21 hidden nodes to ensure the two model architectures had a similar number of trainable parameters. We chose tanh activations for all policies in accordance with Andrychowicz et al. (2021).

A.2. Atari Pong Experiments

We trained unconstrained CNN models and the Lipschitz-bounded policy architectures (SN, AOL, Sandwich, Cayley) across 4 random model initialisations using the PPO implementation in Huang et al. (2022) and the ALE/Pong-v5 environment from Weng et al. (2022). We used the default hyperparameters chosen for CNN policies in Huang et al. (2022) for all policy architectures. We could not use the reward gradient to directly optimize (4) as the Pong environment is not differentiable, hence we implemented attacks with the PGD method from Ding et al. (2019). Robust performance to noise and adversarial attacks in Figure 5 and Table 1 were averaged over the 4 policies and 20 games of Pong, each with a different random seed. Further details on the network architecture for each layer type can be found on our GitHub repository[4].

B. Additional Results

Figure 8 contains additional results showing the training dynamics for unconstrained and Lipschitz-bounded policy networks on the pendulum swing-up task. All policies successfully learned to swing and hold the pendulum upright. As with Figure 7, restricting γ also leads to slower training dynamics (e.g., $\gamma = 3$). The effect is less extreme in Figure 8 than in Figure 7.

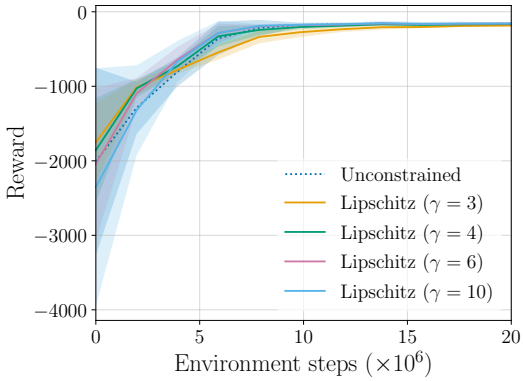


Figure 8: Mean test reward during training on pendulum swing-up for unconstrained (MLP) and Lipschitz-bounded (Sandwich) policies. Bands show one standard deviation over 10 model initializations.

[3] https://github.com/nic-barbara/Lipschitz-RL-MJX/blob/main/scripts/pendulum/train_2_train_models.py
[4] https://github.com/nic-barbara/Lipschitz-RL-Atari/blob/main/liprl/atari_agent.py