Enhancing Robustness in Deep Reinforcement Learning: A Lyapunov Exponent Approach

Rory Young[*] **Nicolas Pugeault**
School of Computing Science
University of Glasgow

Abstract

Deep reinforcement learning agents achieve state-of-the-art performance in a wide range of simulated control tasks. However, successful applications to real-world problems remain limited. One reason for this dichotomy is because the learnt policies are not robust to observation noise or adversarial attacks. In this paper, we investigate the robustness of deep RL policies to a single small state perturbation in deterministic continuous control tasks. We demonstrate that RL policies can be deterministically chaotic, as small perturbations to the system state have a large impact on subsequent state and reward trajectories. This unstable non-linear behaviour has two consequences: first, inaccuracies in sensor readings, or adversarial attacks, can cause significant performance degradation; second, even policies that show robust performance in terms of rewards may have unpredictable behaviour in practice. These two facets of chaos in RL policies drastically restrict the application of deep RL to real-world problems. To address this issue, we propose an improvement on the successful Dreamer V3 architecture, implementing Maximal Lyapunov Exponent regularisation. This new approach reduces the chaotic state dynamics, rendering the learnt policies more resilient to sensor noise or adversarial attacks and thereby improving the suitability of deep reinforcement learning for real-world applications.

1 Introduction

Deep neural networks (DNNs) have revolutionised reinforcement learning (RL) [25], enabling agents to excel in a diverse set of simulated control tasks [12, 16, 22, 23, 24]. However, trained deep RL policies are not robust controllers as DNNs are vulnerable to adversarial attacks [7, 26]. Adding a small amount of noise to each observation can cause these policies to make poor decisions, considerably degrading their overall performance [10, 11, 13]. This lack of stability poses a significant threat when applying deep RL to real-world environments, where inaccurate sensors can easily introduce noise [5]. In this work, we argue that even high-performing deep RL policies are not robust controllers as they can create a chaotic closed-loop control system [4, 14]. These systems are characterised by a high sensitivity to initial conditions, with small changes in initial system states producing vastly different long-term outcomes.

In this paper, we use the spectrum of Lyapunov Exponents [15] to empirically measure the stability of the policies learnt by state-of-the-art deep RL approaches subject to small state perturbations. We show that these controllers can produce chaotic state and reward dynamics when controlling continuous environments. Consequently, a single noisy observation has a dramatic long-term impact on these control systems, with the subsequent approximate state and reward trajectories diverging significantly (Figure 1). This instability poses two problems for the safe deployment of deep RL in real-world environments.

[*]Corresponding email: R.Young.4@research.gla.ac.uk

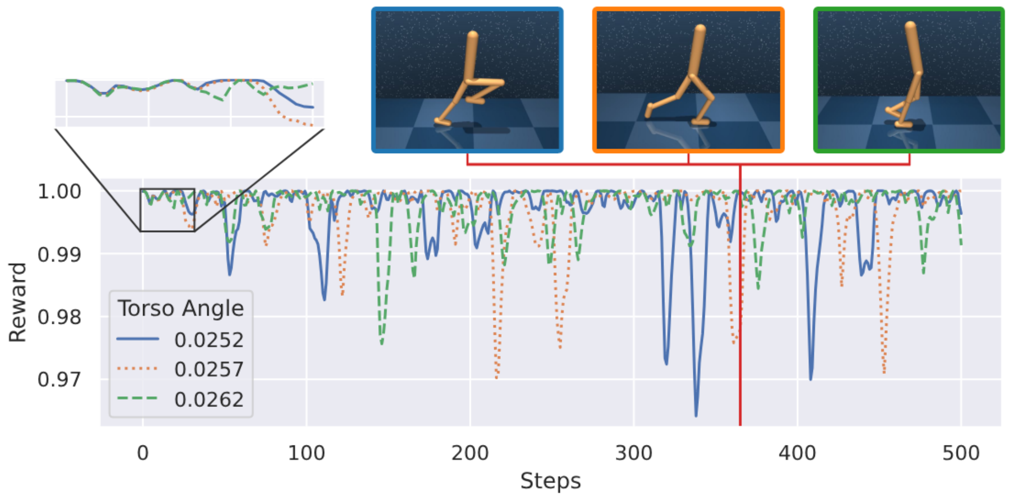


Figure 1: Reward attained when a trained deterministic Soft Actor-Critic [8] agent controls the deterministic *Walker Walk* environment. Each system has the same initial configuration other than the torso angle, which is perturbed by $\pm 5 \times 10^{-4}$ degrees. This small perturbation causes the systems to significantly diverge after 50 steps due to the chaotic nature of the control interaction. Consequently, this affects overall performance as there is significant variation in the total reward attained.

1. The chaotic state dynamics create a fractal return surface [28] which is highly sensitive to small changes in the system state. The high-frequency oscillations in this function cause a lack of robustness as small state perturbations can produce significantly different total rewards.
2. Even for high-performing policies with stable returns, it remains impossible to accurately predict the long-term behaviour of these chaotic control systems as they rely on noisy partially observable sensors to attain an observation. This unpredictability means that safe and reliable behaviour cannot be guaranteed.

To address these issues, we propose Maximal Lyapunov Exponent regularisation for Dreamer V3 [9]. This novel technique estimates the local state divergence using the Recurrent State Space Model and incorporates this term into the policy loss. We demonstrate that this regularisation term significantly reduces the chaotic dynamics produced by this state-of-the-art deep RL controller. This increased stability dramatically improves the robustness of the policy, thus improving the feasibility of deep RL agents for real-world continuous control tasks.

2 Background

2.1 Robust reinforcement learning

Reinforcement learning provides a data-driven method for solving sequential decision-making problems. This control interaction is represented by a deterministic Markov Decision Process (MDP) with state space $\mathcal{S} \subseteq \mathbb{R}^n$, action space $\mathcal{A} \subseteq \mathbb{R}^m$, scalar reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, state transition function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and initial state distribution $\rho_0 \subseteq \mathcal{S}$. The objective of an RL agent is to learn a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ which maximises the sum of discounted returns (Equation 1) for a given discount factor $\gamma \in [0, 1)$.

$$J(\theta) = \mathop{\mathbb{E}}_{s_0 \sim \rho_0} \left[\sum_{t=0}^{\infty} \gamma^t \times r(s_t, a_t) \,\middle|\, s_{t+1} = f(s_t, a_t),\ a_t = \pi_\theta(s_t) \right] \tag{1}$$

RL policies are said to be robust if they can maintain consistent behaviour and reliable performance in the face of noise or adversarial attacks. This stability is crucial for the safe deployment of deep RL

Table 1: Stability of a dynamical system for different values of the Maximal Lyapunov Exponent (λ_1) and Sum of Lyapunov Exponents (λ_Σ).

λ_1	λ_Σ	Stability
-	-	Stable
+	-	Chaotic
+	+	Unstable

to real-world applications where observation noise is inevitable. Recently, a wide range of attack methods have been developed which can easily compromise the performance of deep RL policies with relatively low levels of intervention. In their seminal work, Huang et al. [10] extended the Fast Gradient Sign Method [7] attack to deep RL policies, demonstrating that the addition of a small amount of strategic noise to every observation is sufficient to degrade the performance of trained deep RL agents. Furthermore, Kos & Song [11] showed that this attack does not need to occur at every step, as less frequent attacks with small Gaussian noise still produce a drop in performance. Additionally, they demonstrated that only perturbing the observation when the value function surpasses a set threshold still produces poor performance while significantly decreasing the frequency of the attack. A similar result was established by Lin et al. [13], who showed that specifically attacking when the relative performance gain between best and worst action surpasses a defined threshold can successfully degrade performance. These findings demonstrate that consistent and inconsistent small perturbations to the system can easily confuse deep RL policies, resulting in unintended and detrimental behaviour. Note that small observation noise is frequent in real-world systems, and therefore, this lack of robustness severely limits the application of deep RL to real-world environments.

2.2 Measuring stability: Lyapunov Exponents

Lyapunov Exponents [15, 20] provide a method for quantifying the stability of complex, non-linear, high-dimensional systems by measuring the deformation rate of a small hyperellipsoid under the effects of a transition function. In general, for a dynamical system with N degrees of freedom, there are N Lyapunov Exponents, each representing the exponential growth rate of a unique principal axis of the hyperellipsoid. Given a set of N ordered exponents ($\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N$), the volume of the hyperellipsoid grows proportionally to $e^{(\lambda_1+\lambda_2+...\lambda_N)t}$ and the length grows proportionally to $e^{\lambda_1 t}$. From this definition, the Maximal Lyapunov Exponent (MLE) (Equation 2) and the Sum of Lyapunov Exponents (SLE) (Equation 3) are used to determine if a system is stable, chaotic or unstable, as outlined in Table 1.

$$\lambda_1 = \lim_{t\to\infty} \lim_{\hat{s}_0\to s_0} \frac{1}{t} \ln\left(\frac{|s_t - \hat{s}_t|}{|s_0 - \hat{s}_0|}\right) \qquad (2)$$

$$\lambda_\Sigma = \sum_{i=0}^{N} \lambda_i \qquad (3)$$

Dynamical systems with a negative MLE ($\lambda_1 \leq 0$) are stable as all principal axes of the hyperellipsoid exponentially decrease to zero [21]. In these systems, any trajectories produced from similar initial positions converge to the same trajectory given sufficient time. Conversely, systems with positive MLE ($\lambda_1 > 0$) and SLE ($\lambda_\Sigma > 0$) are unstable as the resulting state trajectories diverge at an exponential rate. However, for a positive MLE ($\lambda_1 > 0$) and negative SLE ($\lambda_\Sigma < 0$), similar trajectories will diverge at an exponential rate but remain confined to a subregion of the phase space known as a *chaotic attractor* [4, 14]. This bounded exponential divergence means trajectories in this region of the state space are unstable and only replicable given the exact same starting state: small perturbations to the starting state will produce significantly different long-term outcomes which appear random and uncorrelated. As a result, *it is impossible to predict the long-term behaviour of a chaotic system given an approximation of the initial state*.

To measure the stability of a known dynamical system, the full spectrum of Lyapunov Exponents can be estimated using the approach outlined by Benettin et al. [2, 3]. This method represents the spectrum as a set of small perturbation vectors which are iteratively updated using the known transition function. To avoid all vectors collapsing in the direction of maximal growth, they are periodically

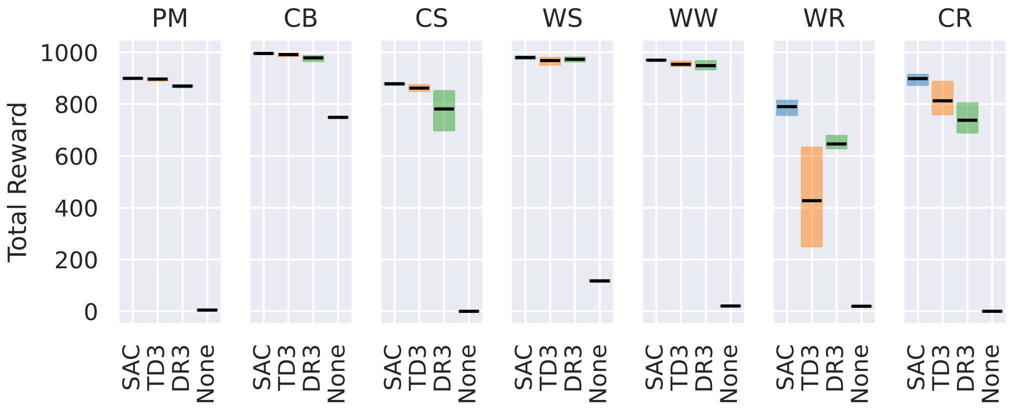


Figure 2: Total episode reward for the *Pointmass Easy* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR) and *Cheetah Run* (CR) environments when controlled by trained instances of SAC, TD3, Dreamer V3 (DR3) and an agent which takes no actions (None). Each policy-environment combination is independently trained with three random seeds and the average interquartile episode reward with a bootstrapped 95% confidence interval is reported over 80 evaluation episodes each with a fixed length of 1000 steps.

Gram-Schmidt orthonormalised so that each vector maintains a unique direction. Performing this orthonormalisation allows for the detection of both positive and negative Lyapunov exponents up to the dimension of the phase space. The spectrum of Lyapunov Exponents is then determined as the average log rate of divergence of the perturbation vectors as they are Gram-Schmidt orthonormalised. Estimation of the full spectrum of Lyapunov Exponents using this method allows for the estimation of λ_1 and λ_Σ which can quantify the stability of the dynamical system.

2.3 Chaos in reinforcement learning

Previous studies have investigated the chaotic state and reward dynamics produced by reinforcement learning policies. Rahn et al. [19] showed that the policy optimisation landscape can contain high-frequency discontinuities in the vicinity of a trained policy. A similar result was established by Wang et al. [28], who proved that control systems with Lipschitz continuous reward and transition functions only have a Lipschitz continuous objective function if $\lambda_1 < -\ln(\gamma)$. When $\lambda_1 > -\ln(\gamma)$, the objective function is a fractal and is α-Hölder continuous with holder exponent $\alpha = -\ln(\gamma)/\lambda_1$. Consequently, a single update to the policy in these chaotic control systems can produce substantially different total rewards. Furthermore, Parmas et al. [17] demonstrated that chaos exists in model-based RL methods due to repeated nonlinear predictions and this instability causes gradients to explode during training.

While these works use chaos theory to highlight an important issue in the field of RL policy learning, they are focused primarily on the stability of the policy subject to *policy parameter* perturbations during training. Our work uses similar concepts but instead focuses on the stability of fully trained RL policies subject to *state perturbations* and the adverse effect this has on the total reward attained in realistic environments. To our knowledge, we are the first to use the spectrum of Lyapunov Exponents to estimate the level of chaos produced by trained DNN policies in continuous control tasks.

3 Chaotic state dynamics

In this section, we use Lyapunov Exponents to identify the level of chaos produced by various state-of-the-art deep reinforcement learning policies in continuous control environments. We claim that the presence of chaos in the MDP implies that the policies are not robust controllers, as trivial changes to the system state produce significantly different long-term state trajectories. This instability poses a significant problem for real-world control systems where consistent and predictable behaviour is necessary.

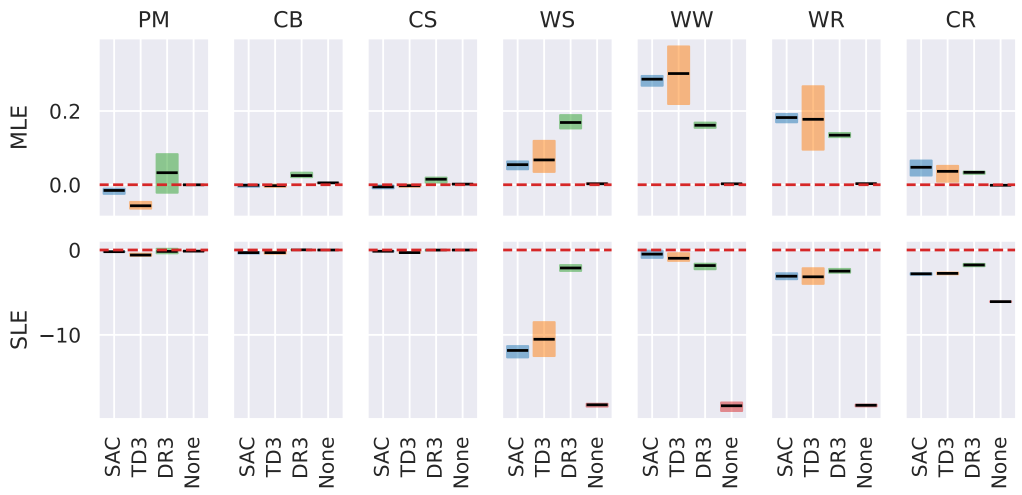
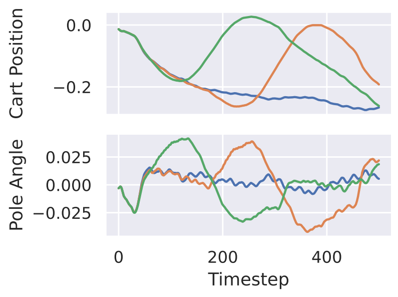


Figure 3: Estimated Maximal Lyapunov Exponent (MLE) and Sum of Lyapunov Exponents (SLE) for the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR) and *Cheetah Run* (CR) environments when controlled by a trained instance of SAC, TD3, Dreamer V3 (DR3) and an agent which takes no actions (None). Each policy-environment combination is independently trained with three random seeds and the interquartile average MLE & SLE for each seed is calculated using 20 initial states. A bootstrapped 95% confidence interval is included to show the variation in MLE and SLE across random seeds.

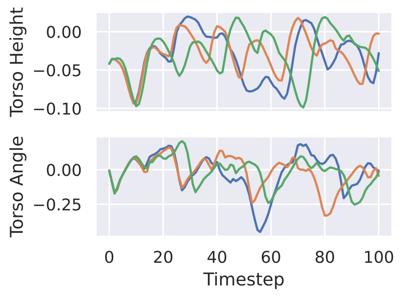
To closely match real-world applications, we estimate the stability of tasks from the DeepMind Control Suite [27] when controlled by deep RL policies. These simulated environments provide a range of deterministic control problems with continuous state spaces, as outlined in Appendix A.1. For each control task we independently train three instances of Soft Actor-Critic (SAC) [8], Twin Delayed Deep Deterministic Policy Gradients (TD3) [6] and Dreamer V3 [9], as these represent state-of-the-art off-policy and model-based methods for continuous control tasks. Furthermore, to determine if trained deep RL policies directly influence the stability of each control system, we also introduce a passive controller that takes no actions. All models are based on the Stable Baselines 3 [18] implementation with the parameters outlined in Appendix A.2 and trained using an Intel Core i7-8700 CPU workstation with an Nvidia RTX 2080 Ti GPU and 32GB of RAM. The average interquartile reward and bootstrapped 95% confidence interval [1] for each policy type are reported in Figure 2. These results show that all three algorithms learn policies which performed well and significantly better than the no-action baseline. However, this does not speak to the types of dynamics produced or how robust these policies are to local state perturbations.

To identify the stability of each policy-environment interaction, we estimate the full spectrum of Lyapunov Exponents using the method proposed by Benettin et al. [2, 3]. The spectrum is calculated using states sampled from the initial state distribution for each environment. A perturbation vector is initialised for each state dimension at a distance of 10^{-4} from the sample state. This represents an arbitrarily small change to the control system without introducing numerical precision errors. The spectrum is calculated over 1000 environment steps and perturbation vectors are orthonormalised every 10 steps to prevent divergence saturation. This process is repeated with 20 initial states and the average value for each exponent is used to calculate the MLE and SLE. Ablation studies for these constants are provided in Appendix B.

Figure 3 provides the bootstrapped 95% confidence interval for the MLE and SLE produced by each policy-environment pair. These results indicate that all the environments are naturally invariant to small perturbations as the no action baseline has $\lambda_1 = 0$. Conversely, when these environments are controlled by deep RL policies, λ_1 can be non-zero, indicating that DNN controllers directly influence the stability of these control systems.



(a) Cartpole Balance



(b) Walker Walk

Figure 4: Partial state trajectory produced by Dreamer V3 when controlling *Cartpole Balance* and *Walker Walk* subject to a single initial state perturbation. Initially, each system is separated by only 10^{-4} units but the subsequent state trajectories diverge significantly as the control interaction is chaotic.

We find that *SAC* and *TD3* produce stable state dynamics in simple low-dimensional environments (*Pointmass*, *Cartpole Balance* and *Cartpole Swingup*) as they have negative MLE. Therefore, if a small perturbation is made to any of the states in these environments, the subsequent state trajectories converge. This result is consistent with the definition of the reward function for each of these tasks as they provide high rewards for maintaining the system at a fixed location. However, when controlled by Dreamer V3, these simple systems exhibit low levels of chaos as they have positive MLE and negative SLE. As a result, state trajectories in these environments are highly sensitive to initial conditions, with similar states producing significantly different long-term outcomes as shown in Figure 4a. Instead of converging to a single state, these trajectories exhibit chaotic behaviour, continuously orbiting within a region which yields high rewards.

Furthermore, all deep RL methods produce chaotic dynamics in the complex high-dimensional environments (*Walker Stand*, *Walker Walk*, *Walker Run* and *Cheetah Run*), as indicated by the positive MLE and negative SLE. This means that *these policies cannot account for arbitrarily small changes* in the system's state since small changes produce exponentially diverging state trajectories, as illustrated in Figure 4b. This poses a significant problem for real-world applications of RL, as observation perturbations are easily introduced via imperfect measurements or sensor noise. Therefore, for these complex high-dimensional environments controlled by deep RL policies, it is impossible to guarantee stability as the system cannot correct itself after a single inaccurate observation.

4 Chaotic rewards

In this section, we show that chaotic state trajectories can also impact a policy's performance and produce chaotic reward trajectories as determined by the Maximal Lyapunov Exponent. This instability creates a fractal return surface in which small state perturbations produce significantly different total rewards. We argue that adversarial attack methods could leverage these high-frequency oscillations, repeatedly injecting perturbations which cause the agent to follow the state trajectories that attain the lowest total reward. This lack of robustness poses a significant problem for real-world control systems, where the worst-case performance is often more significant than the average performance.

By considering the reward over a state trajectory as a trajectory in a one-dimensional reward space, the stability of the reward can also be measured using Lyapunov Exponents. Given that this space is one-dimensional, only one Lyapunov Exponent exists; however, this single exponent can still be used to reliably identify the stability of reward trajectories. Negative λ_1 values indicate that small perturbations to the system's state still produce converging long-term rewards. Moreover, for a bounded reward function, the reward trajectories cannot diverge indefinitely; thus a positive λ_1 indicates that the long-term reward is chaotic as small changes to the system state produce exponentially diverging bounded reward trajectories.

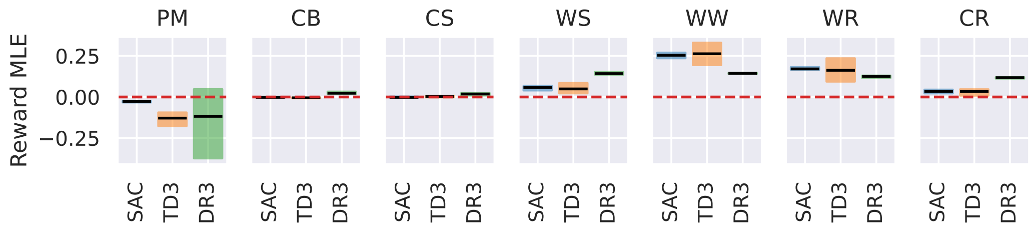


Figure 5: Reward MLE interquartile mean for the *Pointmass* (PM), *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR) and *Cheetah Run* (CR) when controlled by SAC, TD3 and Dreamer V3 (DR3). Each policy-environment combination is independently trained with three random seeds and the reward MLE for each seed is calculated using 20 initial states. A bootstrapped 95% confidence interval is included to show the variation in reward stability across random seeds.

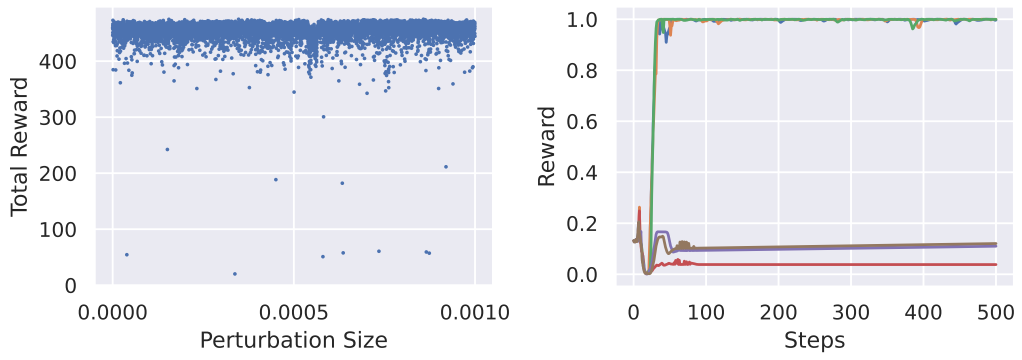


Figure 6: Left: Total reward attained by a deterministic SAC policy when controlling the deterministic *Walker Walk* environment subject to an initial perturbation with fixed direction and varying magnitude. Right: Rewards attained by the three best and three worst state trajectories subject to this perturbation.

Figure 5 provides the average reward MLE and bootstrapped 95% confidence interval for each policy-environment pair defined in Section 3. These graphs indicate that the reward trajectories produced in simple low-dimensional environments (*Pointmass*, *Cartpole Balance*, *Cartpole Swingup*) are stable as they have negative reward MLE. As a result, a single small state perturbation does not harm the total reward attained as the subsequent reward trajectories converge. Conversely, in high-dimensional control systems (*Walker Stand*, *Walker Walk*, *Walker Run* and *Cheetah Run*), the reward MLE is positive, indicating that the reward is highly sensitive to arbitrarily small changes in the system state. Consequently, reward trajectories in these systems are unstable, as similar initial states produce significantly different sequences of rewards. Plotting the reward trajectories for one of these chaotic interactions (Figure 6) shows that a single perturbation with varying magnitude can have a huge detrimental impact on the total reward attained *despite performing well on average*. This poses a significant problem for real-world applications of RL as it is impossible to guarantee worst-case performance in a chaotic control system.

5 Maximal Lyapunov Exponent regularisation

In Sections 3 & 4, we established that deep RL policies can produce chaotic state trajectories in continuous control tasks and that this can have a large detrimental impact on performance. To address this, we propose a novel regularisation method which improves the stability of RL policies by constraining the Maximal Lyapunov Exponent during policy updates. This improved stability is a crucial step towards the safe and reliable deployment of RL policies in real-world domains where local perturbations are common.

Algorithm 1 MLE regularisation

```
Require:
   Policy ($\pi_\theta : \mathcal{H} \times \mathcal{Z} \rightarrow \mathcal{P}(\mathcal{A})$)
   Encoder ($q_\phi : \mathcal{S} \times \mathcal{H} \rightarrow \mathcal{P}(\mathcal{Z})$)
   Decoder ($p_\phi : \mathcal{H} \times \mathcal{Z} \rightarrow \mathcal{P}(\mathcal{S})$)
   Dynamics Predictor ($q_\phi : \mathcal{H} \rightarrow \mathcal{P}(\mathcal{Z})$)
   Sequence Model ($f_\phi : \mathcal{H} \times \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{H}$)

   Current State ($s \in \mathcal{S}$)
   Current Hidden State ($h \in \mathcal{H}$)
   Time Horizon ($T \in \mathbb{N}$)

Ensure: $\mathcal{L}^{\lambda_1}(\theta)$
   $\mathcal{L}^{\lambda_1}(\theta) \leftarrow 0$                                  # Initialise the MLE regularisation loss
   $Z = \{z_l \sim q_\phi(s, h)\}_{l=1}^{L}$                          # Update stochastic representation
   for $t = 1, 2, ..., T$ do
       $A \leftarrow \{a_l \sim \pi_\theta(H_l, Z_l)\}_{l=1}^{L}$                 # Generate a set of sample action
       $H \leftarrow \{h_l = f_\phi(H_l, Z_l, A_l)\}_{l=1}^{L}$              # Update hidden representation
       $Z \leftarrow \{z_l \sim q_\phi(H_l)\}_{l=1}^{L}$                       # Update stochastic representation
       $S \leftarrow \{s_l \sim p_\phi(H_l, Z_l)\}_{l=1}^{L}$                  # Generate a set of predicted states
       $\mathcal{L}^{\lambda_1}(\theta) \leftarrow \mathcal{L}^{\lambda_1}(\theta) + \text{Var}(S) + \text{Var}(H)$   # Update the MLE regularisation loss
   end for
   return $\mathcal{L}^{\lambda_1}(\theta)$
```

We base our regularisation on Dreamer V3 [9], a general-purpose model-based RL algorithm which attains state-of-the-art performance across a diverse set of control tasks. To achieve this, Dreamer V3 uses a Recurrent State Space Model (RSSM) consisting of an Encoder ($q_\phi : \mathcal{S} \times \mathcal{H} \rightarrow \mathcal{P}(\mathcal{Z})$), Decoder ($p_\phi : \mathcal{H} \times \mathcal{Z} \rightarrow \mathcal{P}(\mathcal{S})$), Dynamics Predictor ($q_\phi : \mathcal{H} \rightarrow \mathcal{P}(\mathcal{Z})$) and Sequence Model ($f_\phi : \mathcal{H} \times \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{H}$) to predict state trajectories (s_t), bootstrapped λ-return trajectories (R_t^λ) [25] and state value trajectories ($v_\phi(s_t)$) over a short time horizon T. The policy is then trained to maximise the normalised advantage estimates using REINFORCE gradients [29] and an entropy regulariser (H[$\cdot$]) [30] with weighting coefficient η. The full loss function used to train Dreamer V3's policy is outlined in Equation 4.

$$\mathcal{L}^{\text{Dr3}}(\theta) \doteq -\sum_{t=1}^{T} \left[\text{sg}\left(\frac{R_t^\lambda - v_\phi(s_t)}{\max(1, S)} \right) \log \pi_\theta(a_t|s_t) + \eta \text{H}\left[\pi_\theta(a_t|s_t)\right] \right] \tag{4}$$

$$\mathcal{L}^{\lambda_1}(\theta) \doteq \sum_{t=1}^{T} \left[\underset{L}{\text{Var}}(S_t) + \underset{L}{\text{Var}}(H_t) \right] \tag{5}$$

$$\mathcal{L}^{\text{Policy}}(\theta) \doteq \mathcal{L}^{\text{Dr3}}(\theta) + \mathcal{L}^{\lambda_1}(\theta) \tag{6}$$

At its core, Dreamer V3 uses a stochastic RSSM to predict the state and reward trajectories over a predefined time horizon given an initial starting state s_0 and an internal representation h_0. Due to the stochastic nature of this model, repeating the same trajectory predictions $L \in \mathbb{N}$ times produces a set of state trajectories ($S_t = \langle s_{t,1}, s_{t,2}, ..., s_{t,L} \rangle$) and internal representation trajectories ($H_t = \langle h_{t,1}, h_{t,2}, ..., h_{t,L} \rangle$), each of which provides a plausible estimate of the future states. The variance between trajectories ($\text{Var}_L(\cdot)$) thus provides an estimation of the local state divergence as the state perturbation size approaches 0. Therefore, to minimise λ_1 and improve the stability of Dreamer V3 subject to state perturbation, we propose incorporating the regularisation term outlined in Equation 5 into the policy loss (Equation 6). Including this regularisation term as an additional weighted term forces agents to consider the stability of the system during the optimisation process. This incentivises the policy to produce stable state trajectories which attain high rewards instead of solely optimising the expected return. The complete algorithm for calculating the MLE regularisation term is provided in Algorithm 1 using the notation outlined by Hafner et al. [9].

Table 2: Average total reward and average MLE produced when Dreamer V3 (DR3) and Dreamer V3 with MLE regularisation (MLE DR3) when controlling various environments sampled from the *DeepMind Control Suite* [27]. Each policy-environment combination is independently trained with three random seeds using the hyperparameters outlined in Appendix A.2.

	Reward		MLE	
Environment	DR3	MLE DR3	DR3	MLE DR3
Pointmass	869.5	**880.5**	0.0326	**-0.0275**
Cartpole Balance	**978.6**	970.5	0.0249	**0.0231**
Cartpole Swingup	781.4	**866.4**	**0.0149**	0.0235
Walker Stand	**973.0**	961.6	0.1688	**0.0654**
Walker Walk	948.6	**950.7**	0.1614	**0.1405**
Walker Run	646.3	**698.4**	0.1345	**0.1106**
Cheetah Run	**737.7**	675.2	0.0337	**0.0283**

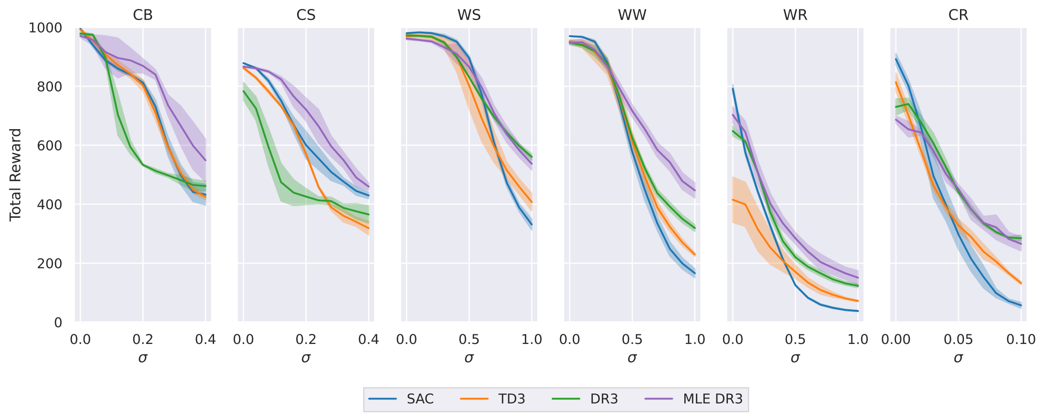
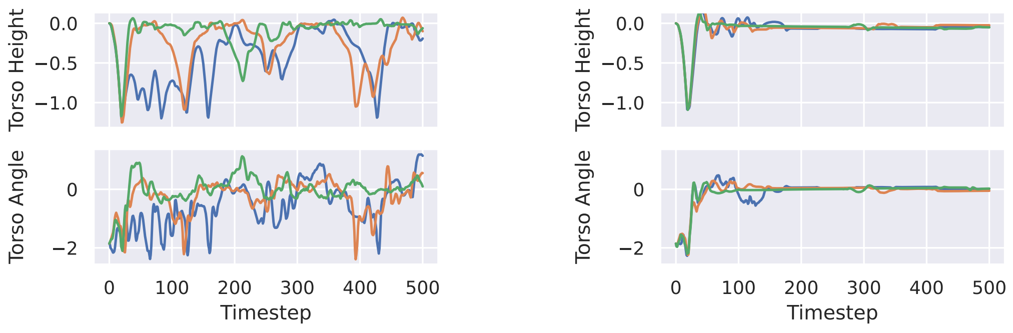


Figure 7: Total episode reward for the *Cartpole Balance* (CB), *Cartpole Swingup* (CS), *Walker Stand* (WS), *Walker Walk* (WW), *Walker Run* (WR) and *Cheetah Run* (CR) environments when controlled by trained instances of SAC, TD3, Dreamer V3 (DR3) and Dreamer V3 with MLE regularisation (MLE DR3) subject to $\mathcal{N}(0, \sigma)$ Gaussian observation noise. Each policy-environment combination is independently trained with three random seeds and the average episode reward with a bootstrapped 95% confidence interval is reported over 80 evaluation episodes each with a fixed length of 1000 steps.

6 Experiments

In this section, we investigate the impact that the proposed MLE regularisation has on Dreamer V3. We show that the inclusion of this term reduces the chaotic state dynamics produced by the control policy and that this improved stability increases performance when noise is introduced. For these experiments, we train three instances of Dreamer V3 with MLE regularisation and reuse the SAC, TD3 and Dreamer V3 policies from Sections 3 & 4. When estimating state divergence, the RSSM predicts $L = 3$ plausible future trajectories over which the state and internal representation variance is measured. Increasing L will produce a more accurate estimate of state divergence; however, this will require more computational resources. Therefore, to maintain a similar training time to that of Dreamer V3, we set $L = 3$. All other hyperparameters are consistent with the Dreamer V3 baseline.

Table 2 provides the average reward and estimated MLE produced by the regularised and unregularised Dreamer V3 models for each control task. This indicates that MLE regularisation successfully minimises the chaotic state dynamics produced by Dreamer V3 while maintaining similar performance. However, as MLE is still positive in the majority of environments, the control interaction still produced chaotic state trajectories. Despite this, the regularised policies are more stable as the rate of divergence has significantly decreased.



(a) Dreamer V3 (b) Dreamer V3 + MLE regularisation

Figure 8: State trajectories produced when Dreamer V3 and Dreamer V3 + MLE regularisation control the *Walker Stand* environment with $\mathcal{N}(0, 0.5)$ Gaussian observation noise.

To identify how robust these regularised Dreamer V3 policies are to continual state perturbations, we measure their performance when Gaussian noise is added to each observation. Each policy is trained without observation noise ($\sigma = 0$) in the fully deterministic variant of each environment and then tested with Gaussian noise ($\mu = 0$ and $\sigma \in [0, 1]$). Performance is measured over 240 episodes with a fixed length of 1000 steps and a maximum reward per step of 1. Figure 7 shows the interquartile mean and bootstrapped 95% confidence interval for the total reward attained by each policy-environment interaction subject to various levels of Gaussian noise. This shows that MLE regularisation significantly improves the performance of Dreamer V3 in four of the noisy control systems, while the other two environments (*Walker Stand* and *Cheetah Run*) attain similar performance to the Dreamer V3 baseline. Furthermore, examining the state trajectories produced by Dreamer V3 and Dreamer V3 + MLE regularisation when controlling the *Walker Stand* task (Figure 8) shows that the regularisation improves the stability of the control interaction as the trajectories do not diverge significantly. These findings indicate that MLE regularisation can improve the robustness of Dreamer V3 subject to observation noise as it produces more consistent and reliable behaviour in the face of uncertainties.

7 Conclusion

A key issue preventing the application of deep reinforcement learning to real-world environments is the need for guaranteed stability and performance in the face of noisy observations and adversarial attacks. In this work, we set out to identify the impact a single perturbation has on the long-term behaviour of deep RL policies in continuous control environments. Using the spectrum of Lyapunov Exponents, we established that the MDP can produce chaotic state and reward trajectories which are highly sensitive to initial conditions. This instability poses two threats to the application of deep RL to real-world problems, where it is infeasible to attain an accurate measurement of the system state. First, small state perturbation can have a large impact on the performance of trained deep RL policies, even where the average performance is good. This can create hazards in real-world conditions where observation noise is prevalent and can be exploited by adversarial attack methods. Second, even when deep RL policies perform well, they can produce unpredictable behaviours, which is undesirable in most real-world applications.

To mitigate these chaotic dynamics and improve robustness we propose Maximal Lyapunov Exponent regularisation for Dreamer V3. This novel approach uses the Recurrent State Space Model to estimate the local state divergence and incorporates this into the policy loss. In effect, the agent optimises its confidence in future trajectories jointly with its expectations of rewards. While MLE regularisation helps improve the robustness of the agent's policies, this approach assumes an accurate estimation of the local state divergence. In environments where the RSSM struggles to capture state dynamics, the effectiveness of the proposed regularisation may be diminished. However, in our experiments, we demonstrate that this regularisation improves the stability of the learnt policies, thereby making them more robust to state perturbations.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320. Curran Associates, Inc., 2021.

[2] Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 1: Theory. *Meccanica*, 15:9–20, 1980.

[3] Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 2: Numerical application. *Meccanica*, 15:21–30, March 1980.

[4] Robert Devaney. *A First Course in Chaotic Dynamical Systems: Theory and Experiment*. CRC Press, second edition, 2020.

[5] Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110:2419–2468, 2021.

[6] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 2018.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *3rd International Conference on Learning Representations*, 2015.

[8] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 2018.

[9] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[10] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In *5th International Conference on Learning Representations*, April 2017.

[11] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. In *5th International Conference on Learning Representations*, 2017.

[12] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, San Juan, Puerto Rico, Conference Track Proceedings*, 2016.

[13] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3756–3762, 2017.

[14] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20:130 – 141, 1963.

[15] Aleksandr Lyapunov. *The General Problem of the Stability of Motion*. Control Theory and Applications Series. Taylor & Francis, 1992.

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

[17] Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. PIPPS: Flexible model-based policy search robust to the curse of chaos. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4065–4074. PMLR, 2018.

[18] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22:1–8, 2021.

[19] Nate Rahn, Pierluca D'Oro, Harley Wiltzer, Pierre-Luc Bacon, and Marc Bellemare. Policy optimization in a noisy neighborhood: On return landscapes in continuous control. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 30618–30640. Curran Associates, Inc., 2023.

[20] David Ruelle. Sensitive dependence on initial condition and turbulent behavior of dynamical systems. *Annals of the New York Academy of Sciences*, 316:408–416, 1979.

[21] Hua Shao, Yuming Shi, and Hao Zhu. Lyapunov exponents, sensitivity, and stability for non-autonomous discrete systems. *International Journal of Bifurcation and Chaos*, 28:1850088, 2018.

[22] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362:1140–1144, 2018.

[24] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[25] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[26] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations*, 2014.

[27] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[28] Tao Wang, Sylvia Herbert, and Sicun Gao. Fractal landscapes in policy optimization. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 4277–4294. Curran Associates, Inc., 2023.

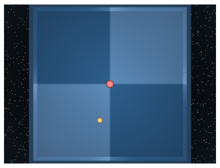
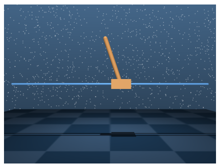
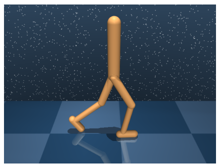
[29] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[30] Ronald J. Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3:241–268, 1991.

A Appendix

A.1 State space composition

Table 3: State space definition for each environment used in this paper. Each dimension of the state space represents a unique aspect of the control system and can contain any real value. Positions are measured in metres (m), angles are measured in radians (rad), velocities are measured in metres per second (m/s) and angular velocities are measured in radians per second (rad/s).

Render	Name	Degrees of freedom	Axis Representations
	Pointmass	4	Point mass x & y position Point mass x & y velocity
	Cartpole Balance Cartpole Swingup	4	Cart position Cart velocity Pole angle Pole angular velocity
	Walker Stand Walker Walk Walker Run	18	Torso x & z position Torso x & z velosity Torso angle Torso angular velocity Left & right hip angle Left & right hip angular velocity Left & right knee angle Left & right knee angular velocity Left & right ankle angle Left & right ankle angular velocity
	Cheetah Run	18	Torso x & z position Torso x & z velocity Torso angle Torso angular velocity Front & back hip angle Front & back hip angular velocity Front & back knee angle Front & back knee angular velocity Front & back ankle angle Front & back ankle angular velocity

A.2 Hyperparameters

Table 4: Hyperparameters used to train SAC, TD3 and Dreamer V3

SAC

Parameter	Value
Environment steps	5×10^6
Buffer size	10^6
Parallel environments	8
Update period	8
Updates per step	8
Discount factor	0.99
Learning rate	0.0003
Batch size	1024
Polyak update coefficient	0.005
Networks activation	Tanh
Networks depth	3
Networks width	256

TD3

Parameter	Value
Environment steps	5×10^6
Buffer size	10^6
Parallel environments	8
Update period	8
Updates per step	8
Discount factor	0.99
Learning rate	0.0003
Batch size	1024
Polyak update coefficient	0.005
Networks activation	Tanh
Networks depth	3
Networks width	256

Dreamer V3

Parameter	Value
Environment steps	10^6
Buffer size	10^5
Parallel environments	8
Update period	8
Updates per step	8
Discount factor	0.99
Learning rate	10^{-4}
Batch size	15
RSSM batch length	64
Imagination horizon	15
Network activation	LayerNorm + SiLU
Networks depth	2
Networks width	512
Recurrent state size	4096
Number of latents	32
Classes per latent	32

B Lyapunov Exponent ablation study

B.1 Default values

Table 5: Parameters used to calculate the spectrum of Lyapunov Exponents using the method outlined by Benettin et al. [2, 3].

Parameter	Value
Total timesteps	1000
Number of iterations	100
Normalisation period	10
Number of samples	20
Perturbation size	0.0001

B.2 Number of iterations ablation study

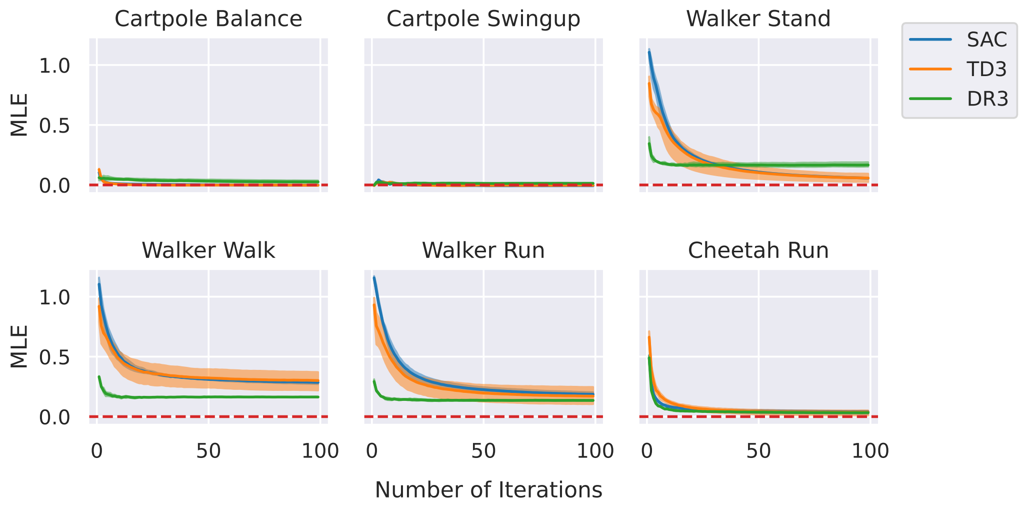


Figure 9: Estimated Maximal Lyapunov Exponent of environments sampled from the *DeepMind Control Suite* when controlled by SAC, TD3 and Dreamer V3 (DR3). Each policy-environment combination is independently trained with three random seeds and evaluated using the parameters in Table 5, except the number of iterations, which varies from 1 to 100. The mean and 95% confidence interval indicate that MLE converges after 100 iterations.

B.3 Number of samples ablation study

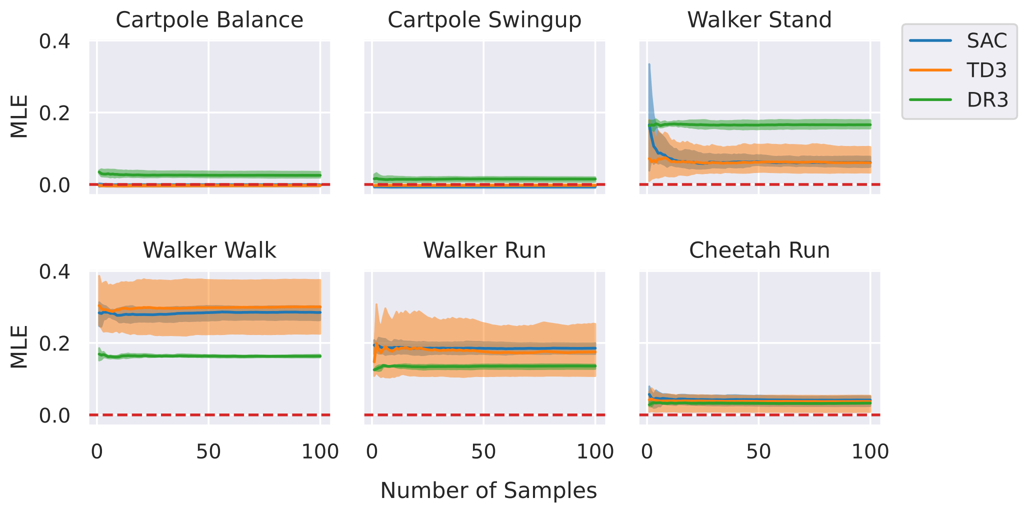


Figure 10: Estimated Maximal Lyapunov Exponent of environments sampled from the *DeepMind Control Suite* when controlled by SAC, TD3 and Dreamer V3 (DR3). Each policy-environment combination is independently trained with three random seeds and evaluated using the parameters in Table 5, except the number of initial samples, which varies from 1 to 20. The mean and 95% confidence interval indicate that MLE converges with 20 initial samples.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim the Markov Decision Process can produce chaotic state trajectories and that this greatly impacts the stability of the reward. These issues are addressed in Sections 3 and 4 respectively. They also claim that we can reduce the chaotic state dynamics produced by Dreamer V3 thereby improving its robustness to adversarial attacks and this is studied in Sections 5 and 6.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: This paper continually highlights an important limitation of deep RL methods, the instability of deep RL controllers. We also propose a method for minimising the chaotic state dynamics and the limitations of this method are discussed in Section 6 and 7.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: This paper discloses the hyperparameters used to train each agent in Appendix A.2, the method for estimating MLE in Section 3 & Appendix B and the method for minimising chaos in Secions 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: This paper provides open access to the code, with instructions to faithfully reproduce the main experimental results outlined in Sections 3 & 5 and Appendix A.2 & B.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All hyperparameters used for training and evaluation are outlined in Appendix A.2 and Appendix B, respectively.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All necessary figures report the mean and bootstrapped 95% confidence interval using the method outlined by Agarwal et al [1]. Full details of this is outlined in the text. Figures 1, 4, 6 and 8 do not include error bars as they explicitly show the variability between state or reward trajectories. Reducing this to a normal distribution removes any nuances produced by the control interaction.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources are outlined in Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   - Our work does not involve human subjects or participants.
   - Our work uses environments [27] and RL agents [18] which are publicly available.
   - We do not publish any new datasets.
   - We ensure reproducibility by including hyperparameters and code.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This paper discusses the important societal issue of safe and reliable control using RL methods.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks as datasets and models are not released.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: This paper uses several existing assets and this is outlined in Section 3. The control environments are sampled from the DeepMind Control Suite [27] and the RL models are implemented using Stable Baselines 3 [18]. The original papers for each model are also cited in the text.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.