Stabilizing Neural Control Using Self-Learned Almost Lyapunov Critics

Ya-Chien Chang Sicun Gao

Abstract— **The lack of stability guarantee restricts the practical use of learning-based methods in core control problems in robotics. We develop new methods for learning neural control policies and neural Lyapunov critic functions in the model-free reinforcement learning (RL) setting. We use sample-based approaches and the Almost Lyapunov function conditions to estimate the region of attraction and invariance properties through the learned Lyapunov critic functions. The methods enhance stability of neural controllers for various nonlinear systems including automobile and quadrotor control.**

I. Introduction

Ensuring stability of neural control policies is critical for the practical use of learning-based methods for control design in robotics. There has been exciting progress towards introducing control-theoretic approaches for enhancing stability in reinforcement learning and imitation learning, such as using Lyapunov methods [1]–[8], control barrier functions [9]–[11], or control-theoretic regularization [12]–[14]. However, three major questions are still open. First, while giving direct control-theoretic prior or guidance can improve performance, to what extent can a learning agent *self-supervise* to achieve certifiable stability? Second, stability requirements should be inherent to a system and not affected by the choice of reward functions, which often do not focus on stability. Can we achieve stability without restricting the flexibility of reward engineering in reinforcement learning? Third, the typical formulation of stability in learning is in the form of reducing expected violation of certain constraints asymptotically, which does not easily translate to certifiable behaviors when the learned policies are practically deployed. Is it possible to obtain stronger stability claims in the standard control-theoretic sense, such as for estimating region of attraction and forward invariance properties? Our goal is to positively answer these questions to improve the reliability and usability of learning-based methods for practical control problems in robotics.

We propose new methods for incorporating Lyapunov methods in deep reinforcement learning. We design a model-free policy optimization process in which the agent attempts to formulate a Lyapunov-like function through *self-supervision*, in the form of a special critic function that is then used for improving stability of the learned policy, without using or being affected by the environment reward. Our work follows the framework of actor-critic Lyapunov methods recently proposed in [5] and extends it in the following ways. We allow the agent to self-learn the Lyapunov critic function by minimizing the Lyapunov risk [7] over its experience buffer without accessing the rewards. This Lyapunov critic function is represented as a generic feed-forward neural network, randomly initialized. This design differs from the typical choice of using a positive definite neural network to construct the Lyapunov candidate, to capture values learned from appropriately designed cost functions that only reflect stability objectives, as in [5], [15]. The lack of restriction and guidance turns out to be beneficial. In our approach, the learning agent can formulate Lyapunov landscapes that better enforce stability, compared to existing methods. In fact, this form of self-learned Lyapunov candidate can often be shown to satisfy the Almost Lyapunov conditions [16], which allows us to use sample-based analysis to estimate its region of attraction and forward invariance properties. We show that the new design is important for learning certifiably stable control policies for practical control problems such as automobile path-tracking and quadrotor control.

Our work builds on the recent progress of model-free and sample-based Lyapunov methods [5], [15]–[17]. Such methods allow us to use sampled Lie derivatives of candidate Lyapunov functions to estimate stability properties of neural control policies without using analytic forms of the system dynamics. We exploit the expressiveness of neural networks for capturing Lyapunov landscapes that are too complex for conventional choices such as sum-of-squares polynomials. We believe the proposed methods further advance the promising direction of developing rigorous neural control and certification methods in reinforcement learning.

In all, we make the following contributions. We propose new methods for training neural Lyapunov critic functions (Section IV.A) and use it to improve stability properties of neural control policies (Section IV.B) in a model-free policy optimization setting. We show that the learned Lyapunov critic function can be analyzed through sample-based methods based on the Almost Lyapunov conditions, for estimating its region of attraction and invariance properties (Section IV.C). We demonstrate the benefits of the proposed methods in comparison with standard policy optimization and existing Lyapunov-based actor-critic methods (Section V).

II. Related Work

Besides the most closely related work [5] discussed above, our work is connected to various recent progress in safe reinforcement learning, Lyapunov methods in reinforcement learning, and data-driven methods for the analysis of control and dynamical systems. Safe reinforcement learning is now a

This material is based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-18-C-0092, AFOSR YIP FA9550-19-1-0041, NSF Career CCF 2047034, and NSF NRI 1830399.

Y.-C. Chang, S. Gao are with the Department of Computer Science and Engineering, UC San Diego, USA. (Email: yac021@eng.ucsd.edu; sicung@eng.ucsd.edu).

large and active field [18], [19] focusing on learning with various forms of soft or hard safety constraints, often formulated as Constrained Markov Decision Processes (CMDP) [20]. Lyapunov methods have seen various applications in this context. It was first introduced in RL by the work of [2], which uses predefined controllers and Lyapunov-based methods for learning a switching policy with safety and performance guarantees. [1] proposed a model-based RL framework that uses Lyapunov functions to guide safe exploration and uses Gaussian Process models of the dynamics to obtain high-performance control policies with provable stability certificates. [3] developed methods for constructing Lyapunov function in the tabular setting that can guarantee global safety of a behavior policy during training. The approach is extended to policy optimization for the continuous control setting in [4], showing benefits in various high-dimensional control tasks. The work in [6] formulates the state-action value function for safety costs as candidate Lyapunov functions and model its derivative with Gaussian Processes with statistical guarantees on the control performance. Similarly in [5], candidate Lyapunov functions are constructed from value functions with benefit in stabilizing the control performance. In the work of [7], [15], neural networks are used to learn Lyapunov functions for establishing certificates for stability and safety properties.

Many other control-theoretic methods have also been proposed to improve reinforcement learning [8]–[14]. These methods typically involve introducing strong control-theoretic priors, or use the neural policies as an oracle to extract low-variance policies in simpler hypothesis classes. For instance, the work in [9] uses control barrier functions to ensure safety of learned control policies. The work in [12] proves stability properties throughout learning by taking advantage of the robustness of control-theoretic priors. Our goal is to turn control-theoretic methods into general self-supervision methods for enhancing stability.

III. Preliminaries

Definition 1 (Dynamical Systems)**.** An n-dimensional controlled dynamical system is defined by

$$\dot{x}(t) = f(x(t),u(t)),\ u(t) = g(x(t)),\ x(0) = x_0, \qquad (1)$$

where $f : D \to \mathbb{R}^n$ is a Lipschitz-continuous vector field, $g : D \to \mathbb{R}^m$ is a control function, and $D \subseteq \mathbb{R}^n$ with $0, x_0 \in D$ defines the state space of the system. Each $x(t) \in D$ is called a state vector and $u(t) \in \mathbb{R}^m$ is a control vector.

Definition 2 (Stability)**.** We say that the system of (1) is stable at the origin if for any $\varepsilon \in \mathbb{R}^+$, there exists $\delta(\varepsilon) \in \mathbb{R}^+$ such that if $\|x(0)\| < \delta$ then $\|x(t)\| < \varepsilon$ for all $t \geq 0$. The system is asymptotically stable at the origin if it is stable and also $\lim_{t\to\infty} x(t) = 0$ for all $\|x(0)\| < \delta$.

Definition 3 (Lie Derivatives)**.** Consider the system in (1) and let $V : D \to \mathbb{R}$ be a continuously differentiable function. The Lie derivative of V over f is defined as

$$\mathsf{L}_f V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \frac{\mathrm{d}x_i}{\mathrm{d}t} = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \dot{x}_i(t). \qquad (2)$$

It measures the rate of change of V over time along the direction of the system dynamics of $x(t)$.

Definition 4 (Lyapunov Conditions for Asymptotic Stability)**.** Consider a controlled system (1) with an equilibrium at the origin, i.e., $\exists\ u \in \mathbb{R}^m$ s.t. $f(0,u) = 0$. Suppose there exists a continuously differentiable function $V : D \to \mathbb{R}$ satisfying $V(0) = 0$, and $\forall x \in D \setminus \{0\}, V(x) > 0$, and $\mathsf{L}_f V(x) < 0$. Then V is a Lyapunov function. The system f is asymptotically stable at the origin if such Lyapunov function V can be found.

The recent work [16] proposed an approximate notion of Lyapunov methods named *Almost Lyapunov functions*, which allows the Lyapunov conditions to be violated in restricted subsets of the space while still ensuring stability properties. It is the basis of our sample-based approach for validating learned Lyapunov candidates. We will discuss this notion and our approach in detail in Section IV.C.

We will use the standard notations for reinforcement learning in Markov Decision Processes (MDP) with state space S, action space A and transition model P: $S \times S \times A \to [0,1]$. A reward function defines the reward for taking action a in state s and transitioning into s'. Let π_ϕ denote a stochastic policy parameterized by ϕ. The goal of the learning agent is to maximize the expected γ-discounted cumulative return $J(\phi) = \mathbb{E}_{s_0,a_0,\ldots}[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t,s_{t+1})]$. Policy optimization methods [21]–[24] estimate policy gradient and use stochastic gradient ascent to directly improve policy performance. A standard gradient estimator is

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\phi \log \pi_\phi(a_t|s_t) \hat{A}_t \right], \qquad (3)$$

where π_ϕ is a stochastic policy and $\hat{A}_t$ estimates the advantage that represents the difference between the Q value of an action compared with the expected value of a state, to indicate whether an action should be taken more frequently in the future. The gradient steps will move the distribution over actions in the right direction accordingly. The expectation $\hat{\mathbb{E}}_t[\ldots]$ is estimated by the empirical average over finite batch of samples. The proximal policy optimization algorithm (PPO) [25] applies clipping to the objective function to remove incentives for the policy to change dramatically, using:

$$J^{\text{CLIP}}(\phi) = \hat{\mathbb{E}}_t \left[\min\left(r_t(\phi)\hat{A}_t, \text{clip}(r_t(\phi), 1-\varepsilon, 1+\varepsilon)\hat{A}_t \right) \right], \qquad (4)$$

where $r_t(\phi) = \pi_\phi(a_t|s_t)/\pi_{old}(a_t|s_t)$ and ε is a hyperparameter. The clipping ensures the gradient steps do not overshoot in the policy parameter space in each policy update.

IV. Policy Optimization with Lyapunov Critics

We now describe how to use self-learned candidate Lyapunov functions to improve stability of neural control policies. We will name these candidate Lyapunov functions as *Lyapunov critics*, following [5]. We will first describe how to learn Lyapunov critics through sampled trajectories, and then how to integrate this critic values in advantage estimation for policy optimization. We describe how sampling-based certification of stability using Lyapunov critics, following the framework of Almost Lyapunov functions. The full procedure

is as shown in Algorithm 1. The overall loop is close to standard PPO [25], [26] with only additional steps at Line 11-12, for learning the Lyapunov critic, and Line 17, for policy optimization guided by the new critic. We will explain these steps in the following sections.

Algorithm 1 Policy Optimization with Self-Learned Almost Lyapunov Critics (POLYC)

1: Initialize policy (actor) network π_ϕ and the reward value function network V^r_η
2: Initialize Lyapunov function network V_θ randomly
3: Initialize replay buffer B as empty set
4: **for** episodes $= 1, \ldots, K$ **do**
5: **for** $t = 1, \ldots, T$ **do**
6: Sample $a_t \sim \pi_\phi(a_t|s_t)$
7: Sample $s_{t+1} \sim P(s_t+1|s_t, a_t)$
8: $B \leftarrow B \cup (s_t, a_t, r_t, s_{t+1})$
9: **end for**
10: Sample mini-batches of size N from B
11: Compute Lyapunov risk $R_{f,N,\rho,\Delta t}(\theta)$ under $\pi_{\phi_{new}}$
12: $\theta \leftarrow \theta - \alpha_\theta \nabla_\theta R_{f,N,\rho,\Delta t}(\theta)$
13: **for** each policy optimization step **do**
14: Sample mini-batches of transitions from B
15: $\delta_t \leftarrow r_t + \gamma V^r_\eta(s_{t+1}) - V^r_\eta(s_t)$ ▷ cf. [26]
16: $\hat{A}(s_t, a_t) \leftarrow \delta_t + \gamma\delta_{t+1} + \cdots + \gamma^{T-t+1}\delta_{T-1}$
17: $\hat{A}^L_\beta \leftarrow \beta \min(0, -\mathsf{L}_{f_{\pi_\phi},\Delta t} V_\theta(s_t)) + (1-\beta)\hat{A}(s_t, a_t)$
18: $r(\phi) \leftarrow \pi_{\phi_{new}}(a|s)/\pi_\phi(a|s)$
19: $J^{\text{CLIP}}(\phi, \beta) \leftarrow \hat{\mathbb{E}}\left[\min\left(r(\phi)\hat{A}^L_\beta, \text{clip}(r(\phi), 1-\varepsilon, 1+\varepsilon)\hat{A}^L_\beta\right)\right]$
20: $\phi \leftarrow \phi + \alpha_\phi \nabla_\phi J^{\text{CLIP}}(\phi, \beta)$
21: $\eta \leftarrow \eta - \alpha_\eta \nabla_\eta \hat{\mathbb{E}}[(V^r_\eta(s_t) - G(s_t))^2]$ ▷ $G(s_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$
22: **end for**
23: **end for**

A. Self-Learning Lyapunov Critics

We represent candidate Lyapunov functions using neural networks, draw samples of the system states, and use gradient descent to learn network parameters that maximize the satisfaction of the Lyapunov conditions in Definition 4. Following [7], such learning can be achieved by minimizing the following loss function named as the Lyapunov risk:

Definition 5 (Empirical Lyapunov Risk [7])**.** Consider dynamical system f with domain D. Let $V_\theta : D \to \mathbb{R}$ be a continuously differentiable function parameterized by θ. The empirical Lyapunov risk of V_θ over f with a sampling distribution $\rho(D)$, written as $R_{f,N,\rho}(\theta)$, is defined as:

$$\frac{1}{N}\sum_{i=1}^{N}\left(\max(-V_\theta(s_i), 0) + \max(0, \mathsf{L}_f V_\theta(s_i))\right) + V^2_\theta(0) \tag{5}$$

where $s_1, \ldots, s_N$ are states sampled according to the sampling distribution ρ. The empirical Lyapunov risk is nonnegative, and when V_θ is a true Lyapunov function for f, this empirical risk attains its global minimum $R_{f,N,\rho}(\theta) = 0$, regardless of the choice of the sample size N and distribution ρ.

In this original formulation, evaluating the Lyapunov risk requires the full knowledge of the system dynamics f for computing $\mathsf{L}_f V$. Our current setting is different. The learning agent does not know the system dynamics f and can only approximate the Lie derivative $\mathsf{L}_f V$ along sampled trajectories of the system, through finite differences:

$$\mathsf{L}_{f,\Delta t} V(s) = \frac{1}{\Delta t}\Big(V(s') - V(s)\Big), \tag{6}$$

where s and s' are two consecutive states and Δt is the time difference between them and $\lim_{\Delta t \to 0} \mathsf{L}_{f,\Delta t} V(s) = \mathsf{L}_f V(s)$. We define the corresponding discretized Lyapunov risk objective $R_{f,N,\rho,\Delta t}$ where the only change from (5) above is that the Lie derivative $\mathsf{L}_f V_\theta(s)$ is replaced by the sampled estimate $\mathsf{L}_{f,\Delta t} V(s)$ for all states.

At each iteration of the policy update (the main loop from Line 4 to 23 in Algorithm 1), we collect trajectories of samples from the controlled system f_π, and perform stochastic gradient descent to minimize the discretized Lyapunov risk $R_{f,N,\rho,\Delta t}(\theta)$ to learn the Lyapunov critic function V_θ (Line 11-12). Note that the dependency of the dynamics f on the behavior policy π_ϕ is important. When the policy is updated, the controlled system changes its dynamics, and the candidate Lyapunov function should be learned correspondingly. Thus, we need to be able to sample from the buffer of previous trajectories, and estimate their Lie derivative values using the new policies (Line 10-11). This dependency makes our approach inherently on-policy in the current formulation, in the sense that the critic is always learned from the behavior policy π_ϕ and thus not very sample-efficient. Off-policy learning of the Lyapunov critic is possible through importance sampling and keeping track of the policy and Lyapunov critic updates, which is a promising direction that we leave open.

B. Stabilization via Policy Optimization

Once the Lyapunov critic is formulated, we use the learned candidate Lyapunov functions as an additional critic value for policy optimization. For a temporarily fixed candidate Lyapunov function V_θ, the only term in the Lyapunov risk that is affected by policy change is the Lie derivative term $\mathsf{L}_{f,\Delta t} V_\theta$. Thus, in policy optimization we now impose the additional goal of ensuring $\mathsf{L}_{f,\Delta t} V(\theta) < 0$ to encourage the policy update to improve the Lyapunov landscape for stabilization. We combine the standard advantage estimate $\hat{A}(s_t, a_t)$ [25] and a clipped Lie derivative term in each policy optimization step:

$$\hat{A}^L_\beta(s_t, a_t) = (1-\beta)\hat{A}(s_t, a_t) + \beta \min(0, -\mathsf{L}_{f_{\pi_\phi},\Delta t} V(s_t)) \tag{7}$$

where $\beta \in [0, 1]$ balances the weights. With policy gradient methods on this advantage estimate, the second term penalizes actions that produce a positive Lie derivative which makes $\min(0, -\mathsf{L}_{f_{\pi_\phi},\Delta t} V(s_t)) < 0$. When the Lie derivative is negative, it does not bias the advantage estimate. We emphasize that although the Lie derivative term is dependent on π_ϕ, it does not have a functional form but only accessed through sampled values based on Equation (6). The true dynamics of the system remains unknown to the learner.

With the definition of the advantage with Lyapunov critic in (7), we can easily replace the standard advantage estimators in various on-policy algorithms. For instance, the standard policy gradient estimator becomes $\mathbb{E}_{\pi_\phi}[\nabla_\phi \log \pi_\phi(a_t|s_t)\hat{A}^t_\beta(s_t, a_t)]$. In Algorithm 1 we use the PPO version [25] of policy update

(Line 19-20), which is what we use in the experiments. Optionally, we can update the β parameter as a Lagrange multiplier, by also taking gradient steps on β at some learning rate α using $\beta \leftarrow \beta - \alpha \mathbb{E}[\mathsf{L}_{f_{\pi_\phi},\Delta t} V(s)]$, clipped between $[1,0]$. We have not observed much performance difference in doing so, and have excluded this step in Algorithm 1 for simplicity.

C. Validating Almost Lyapunov Conditions

A major benefit of the proposed approach is that the self-learned Lyapunov critic allows us to estimate the region of attraction of the controlled system when learning is successful. Since we do not have access to the dynamics of the system and can only estimate the Lyapunov risk at sampled states, we can not expect to certify that the learned Lyapunov critic functions are true Lyapunov functions in the standard sense. However, recent progress in relaxed conditions for Lyapunov methods enabled the use of sample-based analysis to find region of stability. In particular, the Lyapunov critic functions can be analyzed through sampling using the Almost Lyapunov conditions [16] defined as follows:

Proposition 1 (Almost Lyapunov Conditions [16])**.** *Consider a dynamical system in (1) defined by f with domain $D \subseteq \mathbb{R}^n$ and a continuously differentiable positive definite function $V : D \to [0,\infty)$. Let $c_1, c_2 > 0$ be two constants and define B as the region between two sublevel sets $B = \{x \in D : c_1 \le V(x) \le c_2\}$ for V. Let $\Omega \subseteq B$ be a measurable set. Suppose for some $a > 0$, $\max_{x \in B} \mathsf{L}_f V(x) < a \min_{x \in B} V$, and $\forall x \in \Omega, \mathsf{L}_f V(x) \ge -aV(x)$ and $\forall x \in \Omega \setminus B, \mathsf{L}_f V(x) < -aV(x)$. Then there exists $\hat{\varepsilon} > 0$ such that for any $\varepsilon \in (0, \hat{\varepsilon})$, if the volume of each connected component Ω^* of Ω satisfies $\mathrm{vol}(\Omega^*) \le \varepsilon$, then there exists $T > 0$ such that for any $x_0 \in D$ with $V(x_0) < c_2 - r(\varepsilon)$, $x(t)$ stays within B for all $t > 0$ and moreover, it converges to the sublevel set $V(x) \le c_1 + r(\varepsilon)$ for any $t > T$. Here $r(\varepsilon) = h\varepsilon^{1/n} + g\varepsilon$ for some constants h and g.*

The full proof is in [16] where all constants are explicitly constructed. Conceptually, the theorem relaxes the standard Lyapunov conditions to allow a set Ω that contains the violation states where the Lie derivative can be positive ($\forall x \in \Omega, \mathsf{L}_f V(x) \ge -aV(x)$). As long as each component of Ω is small enough (with volume less than ε), the violations do not affect stability. Under mild conditions, the appropriate region between sublevel sets of the system (written as B in the definition) defines a forward invariant set for all trajectories, and an approximate form of contraction where the trajectory converges to near the lower level set of the region B.

With the Almost Lyapunov conditions, we can use an ε-net over the space, chosen based on the Lipschitz constant of the Lie derivative, such that using sampled $\mathsf{L}_{f,\Delta t} V$ value at the center of each cell we can identify the set Ω of states where the Lie derivatives violate the standard conditions. In Figure 1, we show the results of such computation to compare four different types of candidate Lyapunov functions for the inverted pendulum controlled by neural network policies. Each plot visualizes the sign of the Lie derivative of the proposed Lyapunov candidate at uniformly sampled points over the $\theta/\dot{\theta}$ space. The grey dots represent cells where the candidate Lyapunov function has negative Lie derivative values ($\mathsf{L}_f V < -aV(x)$ as required in Proposition 1), and the red dots indicate cells that violate such condition. The value of the Lyapunov candidate itself can be shown to be always nonnegative in the domain in all four cases, and the black contours represent the level sets of increasingly positive values. The red dots indicate states where the standard Lyapunov conditions are violated, and the patterns are different.

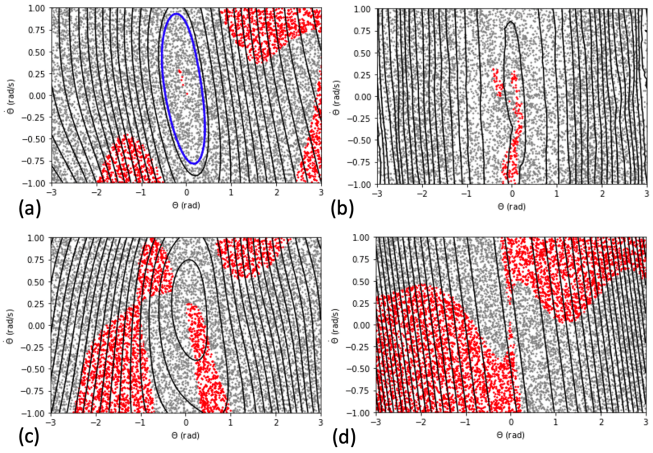

Fig. 1. The landscapes of four different Lyapunov candidates for the inverted pendulum controlled by neural network policies.

- Figure 1(a) shows the self-learned Lyapunov critic obtained after learning the neural control policy using Algorithm 1. We see that we can find a sublevel set of the Lyapunov critic (inside the blue circle) where the Lie derivative is positive only at a very small number of sparse regions (a few red dots near the center). This landscape satisfies the Almost Lyapunov conditions and the sublevel set defines forward invariant set with attraction (Proposition 1).
- Figure 1(b) shows the landscape generated by the Lyapunov actor-critic method (LAC) [5]. We see that the Lyapunov conditions are satisfied more globally, although with more violations closer to the origin. The function can also be established as an Almost Lyapunov function where the violation is sparse, which does not include the innermost sublevel set. The level sets is also much larger than those in (a), making it harder to find sublevel sets that are forward invariant. On the other hand, as shown later in the next section, the learned controller does always stabilize the system in all sampled trajectories. This indicates that there may exist better Lyapunov candidates for certifying the stability.
- Figure 1(c) shows the Lyapunov candidate fit for the control policy learned by standard PPO. We see that much more violation states are observed and it does not allow us to find a region where Almost Lyapunov Conditions can be validated. This indicates that the lack of Lyapunov critic makes it hard to enforce stability properties.
- Figure 1(d) uses the quadratic Lyapunov function for the linearized inverted pendulum obtained through LQR directly as a Lyapunov candidate for the neural controller trained by Algorithm 1, same as the one used in (a). We see that the simple Lyapunov function does not satisfy the

Almost Lyapunov conditions and fails to capture the stability properties of the learned controller.

V. Experiments

We now show experimental results on the proposed methods for various nonlinear control problems. Our implementation follows Algorithm 1 (referred to as LY) and optionally uses an additional entropy term in the advantage estimates. We compare the performance with the closely related work of Lyapunov-based Actor-Critic (LAC) [5], and also standard implementations of Soft Actor-Critic (SAC) [27] and PPO [25]. We use the following control environments: the inverted pendulum, quadrotor control, automobile path-tracking, and Mujoco Hopper and Walker.

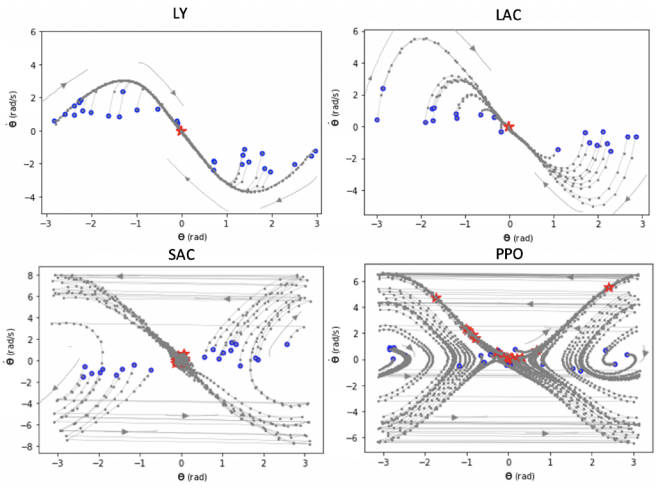

Fig. 2. Sample trajectories generated by the policies learned with each algorithm in the inverted pendulum environment.

Inverted Pendulum. For the standard inverted pendulum (Pendulum-v0 in OpenAI Gym), Figure 2 shows the system trajectories under learned policies plotted in the θ-$\dot{\theta}$ space. The blue dots indicate initial positions and the red dots indicate positions after stabilization. Both LY and LAC learn stable controllers that reach the upright position without falling down, i.e., showing monotonically decreasing θ values. In contrast, the policies trained with SAC and PPO show horizontal lines that indicate falling down and crossing $-\pi$, by first swinging downwards and then back to the upright position. As discussed in the previous section in Fig 1, both LY and LAC learn Almost Lyapunov functions that are consistent with the stable control behaviors.

Quadrotor control. We learn neural controller for the 6-DOF quadrotor model, which has four control inputs and twelve state variables. The control inputs $\Omega_1, \Omega_2, \Omega_3$ and Ω_4 are the angular velocity of each rotor. The state variables are the inertia frame positions (x, y, z), velocities $(\dot{x}, \dot{y}, \dot{z})$, rotation angles (ϕ, θ, ψ) and angular velocities $(\dot{\phi}, \dot{\theta}, \dot{\psi})$. The equations of motion of the quadrotor are given in [28] and Figure 3(a) shows the schematic. The control problem is known to be hard for policy gradient methods, typically requiring imitation learning steps. In Figure 4, we see that the LY method can learn stable tracking controller for the system. Moreover, the learned Lyapunov critic is shown in Fig 4(a) and its Lie derivatives is in Fig 4(b). Almost Lyapunov conditions can be validated with in the blue level set which certifies stable behavior shown in the trajectory. In comparison, LAC fails to learn a working controller.

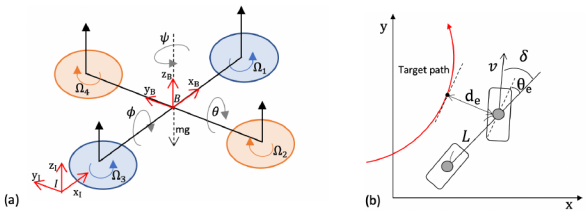

Fig. 3. (a) Schematic of 6-DOF quadrotor system with body frame B and inertia frame I. (b) Schematic of wheeled vehicle.

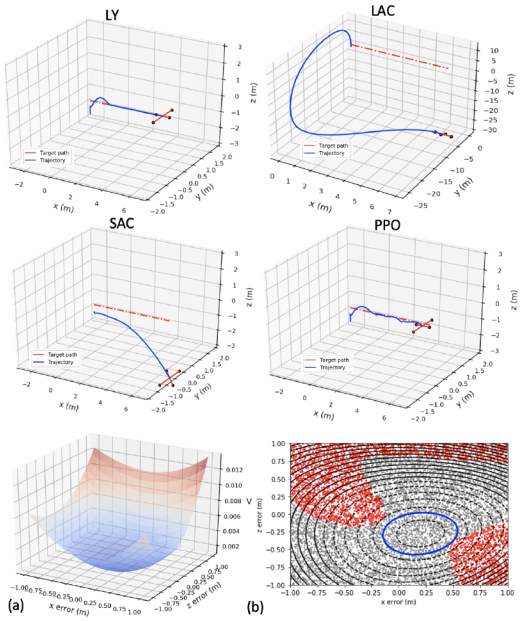

Fig. 4. Quadrotor control for tracking a horizontal path. (a) shows the Lyapunov critic learned in LY and (b) shows where Almost Lyapunov conditions are validated (within the blue level set).

Automobile path-tracking control. The control problem of following a target path and speed command is the core of autonomous driving [29]. In the training environment, the state has four dimensions including target velocity V_t, angular error $\theta_e(t)$, distance to the path $d_e(t)$, and the vehicle speed $v(t)$. The action space contains acceleration $a(t)$ and a steering control $\delta(t)$. The car dynamics follows standard bicycle model [29] and Figure 3(b) shows a schematic. In this experiment, we observe that all methods can obtain working control policies in the training environment in Figure 5(a). However, the Lyapunov landscape matters when applying the policy on a different path, as illustrated in the bottom row in Fig 5. LY methods generalizes well to unseen paths because of its region of attraction, validated via the Almost Lyapunov conditions within the blue level set of the learned Lyapunov critic. In Figure 5(c), we see that the Lyapunov critic learned

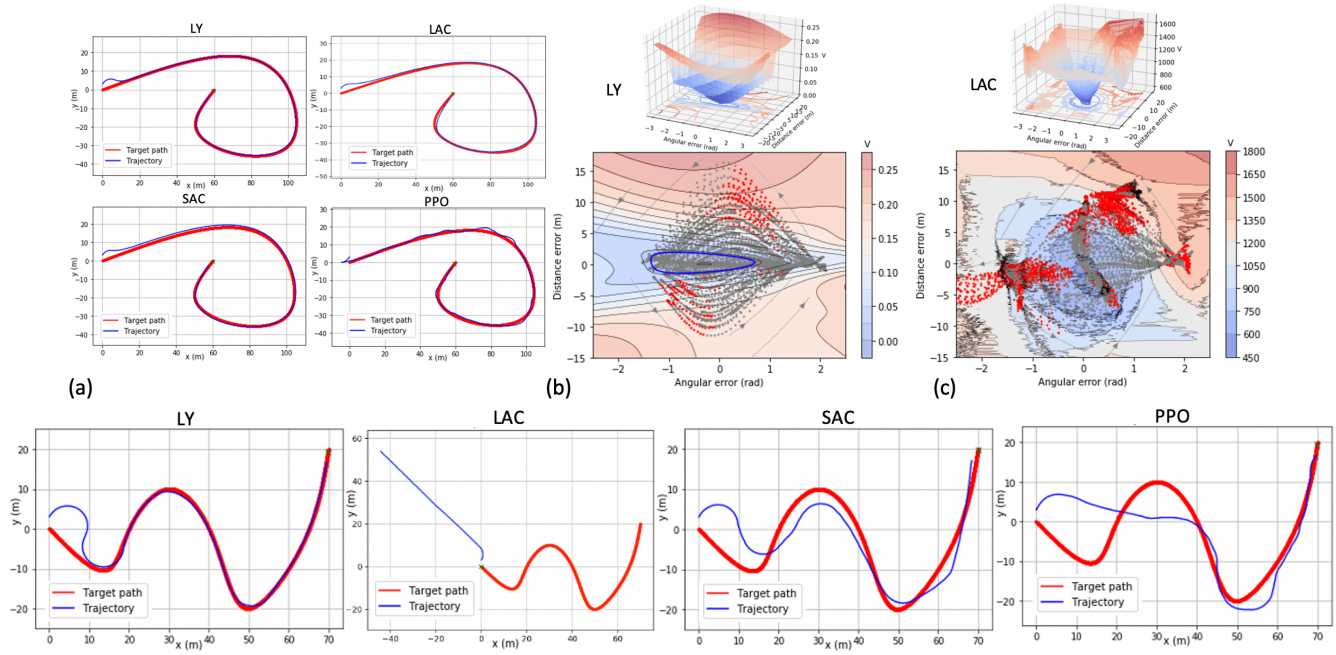

Fig. 5. Experiments for automobile path tracking. (a) All methods learn to control well in the training environment. (b) 3D and 2D landscape generated by the learned Lyapunov function from LY. Almost Lyapunov conditions are validated within the blue level set. (c) Landscape generated by the learned Lyapunov critic from LAC. Second row: the control performance when tracking an unseen path (the red curve). The blue curves indicate the trajectory of the vehicle, starting from the left and going towards the right.

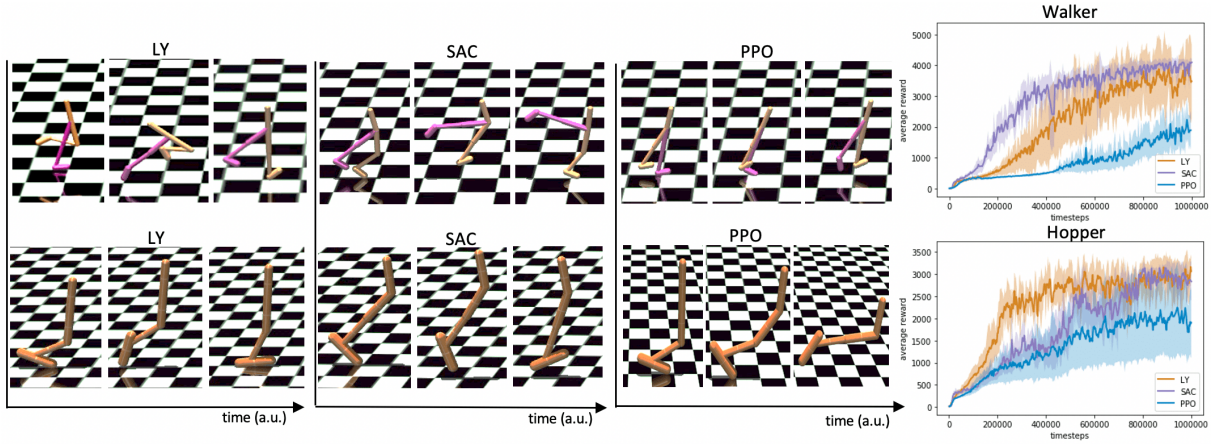

Fig. 6. Control performance in the Walker and Hopper environments and learning curves over 5 random seeds. Our method typically learn faster than PPO, and comparable to SAC. All methods can achieve high rewards, but the learned control behaviors are different.

in LAC does not create a landscape that enforces a region of attraction in the sense of Almost Lyapunov conditions.

Mujoco Walker and Hopper. Walker and Hopper are standard high-dimensional locomotion environment. For both the goal is move forward as fast as possible and not stabilization. The LAC method requires using cost functions for stabilization objectives only, and thus does not work in these environments. We can learn Lyapunov critics that are independent from the reward, just focusing on stabilizing the joint angles to hold the upright positions. As shown in Fig 6, the LY controller maintains better pose and gaits compared to SAC and PPO. The learning curves show that LY does not slow down learning, and can be used in generic high-dimensional control tasks to improve performance.

VI. Conclusion

We proposed new methods for training stable neural control policies using Lyapunov critic functions. We showed that the learned Lyapunov critics can be used to estimate regions of attraction for the controllers based on Almost Lyapunov conditions. We demonstrated the benefits of the proposed methods in various nonlinear control problems. Future work includes further improving sample complexity of the Lyapunov critic learning as well as the validation process for ensuring the Almost Lyapunov conditions.

REFERENCES

[1] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 908–918.

[2] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, pp. 803–832, 2002.

[3] Y. Chow, O. Nachum, E. A. Duéñez-Guzmán, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 2018, pp. 8103–8112.

[4] Y. Chow, O. Nachum, A. Faust, M. Ghavamzadeh, and E. A. Duéñez-Guzmán, "Lyapunov-based safe policy optimization for continuous control," *CoRR*, vol. abs/1901.10031, 2019.

[5] M. Han, L. Zhang, J. Wang, and W. Pan, "Actor-critic reinforcement learning for control with stability guarantee," *IEEE Robotics and Automation Letters and IROS*, 2020.

[6] M. Jin and J. Lavaei, "Control-theoretic analysis of smoothness for stability-certified reinforcement learning," in *57th IEEE Conference on Decision and Control, CDC 2018, Miami, FL, USA, December 17-19, 2018*. IEEE, 2018, pp. 6840–6847.

[7] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, 2019, pp. 3240–3249.

[8] M. Gallieri, S. S. M. Salehian, N. E. Toklu, A. Quaglino, J. Masci, J. Koutník, and F. Gomez, "Safe interactive model-based learning," 2019.

[9] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, 2019, pp. 3387–3395.

[10] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," 2019.

[11] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 2669–2678.

[12] R. Cheng, A. Verma, G. Orosz, S. Chaudhuri, Y. Yue, and J. Burdick, "Control regularization for reduced variance reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 1141–1150.

[13] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[14] A. Liu, G. Shi, S.-J. Chung, A. Anandkumar, and Y. Yue, "Robust regression for safe exploration in control," ser. Proceedings of Machine Learning Research, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Eds., vol. 120. The Cloud: PMLR, 10–11 Jun 2020, pp. 608–619.

[15] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, 2018, pp. 466–476.

[16] "Almost lyapunov functions for nonlinear systems," *Automatica*, vol. 113, p. 108758, 2020.

[17] R. Bobiti and M. Lazar, "Automated-sampling-based stability verification and doa estimation for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3659–3674, 2018.

[18] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, pp. 1437–1480, 2015.

[19] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 22–31.

[20] E. Altman, "Constrained markov decision processes," 1999.

[21] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[22] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML 2016*, 2016, pp. 1928–1937.

[23] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML 2015*, 2015, pp. 1889–1897.

[24] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Second-order optimization for deep reinforcement learning using kronecker-factored approximation," in *NIPS 2017*, 2017, pp. 5285–5294.

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

[26] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2018.

[27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1861–1870.

[28] B. Rubí, R. Pérez, and B. Morcego, "A survey of path following control strategies for uavs focused on quadrotors," *Journal of Intelligent Robotic Systems*, 05 2020.

[29] J. Snider, "Automatic steering methods for autonomous automobile path tracking," 04 2011.