Proceedings of Machine Learning Research vol 242:52–67, 2024 6th Annual Conference on Learning for Dynamics and Control

Learning to Stabilize High-dimensional Unknown Systems Using Lyapunov-guided Exploration

Songyuan Zhang SZHANG21@MIT.EDU and **Chuchu Fan** CHUCHU@MIT.EDU
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA

Editors: A. Abate, M. Cannon, K. Margellos, A. Papachristodoulou

Abstract

Designing stabilizing controllers is a fundamental challenge in autonomous systems, particularly for high-dimensional, nonlinear systems that can hardly be accurately modeled with differential equations. The Lyapunov theory offers a solution for stabilizing control systems, still, current methods relying on Lyapunov functions require access to complete dynamics or samples of system executions throughout the entire state space. Consequently, they are impractical for high-dimensional systems. This paper introduces a novel framework, **LY**apunov-**G**uided **E**xploration (LYGE), for learning stabilizing controllers tailored to high-dimensional, unknown systems. LYGE employs Lyapunov theory to iteratively guide the search for samples during exploration while simultaneously learning the local system dynamics, control policy, and Lyapunov functions. We demonstrate its scalability on highly complex systems, including a high-fidelity F-16 jet model featuring a 16D state space and a 4D input space. Experiments indicate that, compared to prior works in reinforcement learning, imitation learning, and neural certificates, LYGE reduces the distance to the goal by 50% while requiring only 5% to 32% of the samples. Furthermore, we demonstrate that our algorithm can be extended to learn controllers guided by other certificate functions for unknown systems.[1]

Keywords: Lyapunov-guided Exploration, High-dimensional Unknown Systems, Machine Learning

1. Introduction

Designing stabilizing controllers for high-dimensional systems with potentially unknown dynamics is essential in autonomous systems, where Lyapunov-based control design plays a significant role (Parrilo, 2000). In recent years, methods have been proposed to automatically construct Lyapunov functions and control Lyapunov functions (CLFs) for systems of varying complexity, encompassing both optimization and learning-based methods (Giesl and Hafstein, 2015; Dawson et al., 2022a). However, existing methods encounter two major challenges: *scalability*, *i.e.*, applicability to high-dimensional systems, and *model transparency*, *i.e.*, the necessity of knowing the system dynamics.

Traditional optimization-based control design involves finding controllers and Lyapunov functions by solving a sequence of semi-definite programming (SDP) problems (Parrilo, 2000; Majumdar et al., 2013; Ahmadi and Majumdar, 2016). However, scalability to high-dimensional systems is hindered by the exponential growth of the number of decision variables with system dimension (Lofberg, 2009) and numerical problems (Permenter and Parrilo, 2018), including strict feasibility and numerical reliability issues. Although recent advancements have produced scalable SDP solvers (Yurtsever et al., 2021), they depend on assumptions such as the sparsity of the decision matrix. Neural network (NN)-based representations of Lyapunov functions have gained popularity (Chang et al., 2019; Han

---

[1] Project website: https://mit-realm.github.io/lyge-website/. The appendix can be found on the project website.

et al., 2020; Chang and Gao, 2021; Dawson et al., 2022b) and can to some extent alleviate dimensionality limitations when finding a CLF. However, due to their reliance on state-space sampling, learning techniques still face exponential growth in sample complexity for high-dimensional systems.

Additionally, most optimization-based and learning-based methods require system dynamics to be known as ordinary differential equations (ODEs), limiting their applicability and practicality for real-world systems. For instance, the F-16 fighter jet model (Heidlauf et al., 2018) investigated in this paper is represented as a combination of look-up tables, block diagrams, and C programs. Accurately describing the complex behavior of the system using an ODE is highly challenging. For systems with unknown dynamics, previous works have attempted to first use system identification, (*e.g.*, learn the ODE model with NNs), then find a controller with a CLF (Dai et al., 2021; Zhou et al., 2022). However, using a single NN to fit the dynamics of the entire state space of a high-dimensional system requires a vast number of training samples to cover the entire state space, and these surrogate NN models can exhibit substantial prediction errors in sparsely sampled regions of the state-action space.

Our work is motivated by the fact that for high-dimensional systems, collecting data across the entire state space is both infeasible and unnecessary, as only a small subset of the state space is reachable for the agent starting from a set of initial conditions. Therefore, obtaining a model for the entire state space is excessive (Kamalapurkar et al., 2016). Instead, we learn a model valid only in the reachable subset of the state space and update the controller according to guidance, e.g., the learned CLF, to continuously expand the subset towards the goal. Constructing such a subset is non-trivial, so we assume access to some imperfect and potentially unstable demonstrations as initial guidance for reachable states. Starting from these demonstrations, our objective is to update the controller, guide exploration of necessary regions in the state space, and ultimately stabilize the system at the goal.

To achieve this, we propose a novel framework, **LY**apunov-**G**uided **E**xploration (LYGE), to jointly learn the system dynamics in the reachable subset, a controller, and a CLF to guide the exploration of the controller in high-dimensional unknown systems. We iteratively learn the dynamics in the reachable states using past experience, update the CLF and the controller, and perform exploration to expand the subset toward the goal. Upon convergence, we obtain a stabilizing controller for the high-dimensional unknown system. The main **contributions** of the paper are: 1) We propose a novel framework, LYGE, to learn stabilizing controllers for high-dimensional unknown systems. Guided by a learned CLF, LYGE explores only the *useful* subset, thus addressing the scalability and model transparency problems; 2) We show that the proposed algorithm learns a stabilizing controller; 3) We conduct experiments on benchmarks including Inverted Pendulum, Cart Pole, Cart II Pole (Brockman et al., 2016), Neural Lander (Shi et al., 2019), and the F-16 model (Heidlauf et al., 2018) with two tasks. Our results show that our learned controller outperforms other reinforcement learning (RL), imitation learning (IL), and neural-certificate-based algorithms in terms of stabilizing the systems while reducing the number of samples by 68% to 95%.

2. Related Work

Control Lyapunov Functions. Our work builds on the widely used Lyapunov theory for designing stabilizing controllers. Classical CLF-based controllers primarily rely on hand-crafted CLFs (Choi et al., 2020; Castaneda et al., 2021) or Sum-of-Squares (SoS)-based SDPs (Parrilo, 2000; Majumdar et al., 2013; Ahmadi and Majumdar, 2016; Long et al., 2023). However, these approaches require known dynamics and struggle to generalize to high-dimensional systems due to the exponential growth of decision variables and numerical issues (Lofberg, 2009; Permenter and Parrilo, 2018). To alleviate these limitations, recent work uses NN to learn Lyapunov functions (Richards et al., 2018; Abate et al., 2020, 2021; Gaby et al., 2021) and stabilizing controllers (Chang et al., 2019; Mehrjou et al., 2021; Dawson et al., 2022b; Farsi et al., 2022; Zhang et al., 2023; Wang et al., 2023; Min et al., 2023). Most of these works sample states in the entire state space and apply supervised learning to enforce the CLF conditions. They either assume knowledge of the dynamics or attempt to fit the entire state space's dynamics (Dai et al., 2021; Zhou et al., 2022), making it difficult to generalize to high-dimensional real-world scenarios where the number of required samples grows exponentially with the dimensions. In contrast, our algorithm can handle unknown dynamics and does not suffer from the curse of dimensionality caused by randomly sampling states in the entire state space.

Reinforcement Learning (RL) and Optimal Control. RL and optimal control have demonstrated strong capabilities on problems without knowledge of the dynamics, particularly in hybrid systems (Schulman et al., 2015, 2017; Rosolia and Borrelli, 2019; So and Fan, 2023). However, they struggle to provide results of the closed-loop system's stability. Additionally, hand-crafted reward functions and sample inefficiency impede the generalization of RL algorithms to complex environments. Recent works in the learning for control domain aim to solve this problem by incorporating certificate functions into the RL process (Berkenkamp et al., 2017; Chow et al., 2018; Cheng et al., 2019; Han et al., 2020; Chang and Gao, 2021; Zhao et al., 2021; Qin et al., 2021). Nevertheless, they suffer from limitations such as handcrafted certificates (Berkenkamp et al., 2017) and balancing CLF-related losses with RL losses (Han et al., 2020; Chang and Gao, 2021). Unlike these approaches, our algorithm learns the CLF from scratch without prior knowledge of the dynamics or CLF candidates and provides a structured way to design loss functions rather than relying on reward functions. Furthermore, we can demonstrate the stability of the closed-loop system using the learned CLF.

Imitation Learning (IL). IL is another common tool for such problems. However, classical IL algorithms like behavioral cloning (BC) (Pomerleau, 1991; Bain and Sammut, 1995; Schaal, 1999; Ross et al., 2011), inverse reinforcement learning (IRL) (Abbeel and Ng, 2004; Ramachandran and Amir, 2007; Ziebart et al., 2008), and adversarial learning (Ho and Ermon, 2016; Finn et al., 2016; Fu et al., 2018) primarily focus on recovering the exact policy of the demonstrations, which may result in poor performance when given imperfect demonstrations. A recent line of work on learning from suboptimal demonstrations offers a possible route to learn a policy that outperforms the demonstrations. However, they either require various types of manual supervision, such as rankings (Brown et al., 2019; Zhang et al., 2021), weights of demonstrations (Wu et al., 2019; Cao and Sadigh, 2021), or have additional requirements on the environments (Brown et al., 2020; Chen et al., 2021), demonstrations (Tangkaratt et al., 2020, 2021), or the training process (Novoseller et al., 2020). Moreover, none of them can provide results about the stability of the learned policy. Another line of work learns certificates from demonstrations (Ravanbakhsh and Sankaranarayanan, 2019; Robey et al., 2020; Chou et al., 2020; Boffi et al., 2021), but they need additional assumptions such as known dynamics, perfect demonstrations, or the ability to query the demonstrator. In contrast, we leverage the CLF as natural guidance for the exploration process and do not require additional supervision or assumptions about the demonstrations or the environment to learn a stabilizing policy.

3. Problem Setting and Preliminaries

We consider a discrete-time unknown dynamical system

$$x(t+1) = h(x(t), u(t)), \tag{1}$$

where $x(t) \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ represents the state at time step t, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ denotes the control input at time step t, and $h : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the *unknown* dynamics. We assume the state space $\mathcal{X}$ to be compact and h is Lipschitz continuous in both (x, u) with constant $L_h > 0$ following Berkenkamp et al. (2017). Our objective is to find a control policy $u(\cdot) = \pi(x(\cdot))$, where $\pi : \mathcal{X} \to \mathcal{U}$, such that from initial states $x(0) \in \mathcal{X}_0$, under the policy π, the closed-loop system asymptotically stabilizes at a goal $x_{\text{goal}} \in \mathcal{X}$. In other words, $\forall x(t)$ starting from $x(0) \in \mathcal{X}_0$ and satisfying Equation (1) with $u(t) = \pi(x(t))$, we have $\lim_{t\to\infty} \|x(t) - x_{\text{goal}}\| = 0$.

We assume that we are given a set of N_{D^0} demonstration transitions $\mathcal{D}^0 = \{(x_i(t), u_i(t), x_i(t+1))\}_{i=1}^{N_{D^0}}$ generated by a demonstrator policy π_d starting from states $x_i(0) \in \mathcal{X}_0$. In contrast to the assumption of stabilizing demonstrators in classical IL works (Ho and Ermon, 2016), our demonstrations may not be generated by a stabilizing controller.

Lyapunov theory is widely used to prove the stability of control systems, and CLFs offer further guidance for controller synthesis by defining a set of stabilizing control inputs at a given point in the state space. Following Grüne et al. (2017), we provide the definition of CLF for discrete system (1).

Definition 1 *Consider the system* (1) *and a goal point* x_{goal}, *and let* $\mathcal{G} \subseteq \mathcal{X}$ *be a subset of the state space such that* $x_{\text{goal}} \in \mathcal{G}$. *A function* $V : \mathcal{G} \to \mathbb{R}$ *is called a CLF on* $\mathcal{G}$ *if there exists functions* $\underline{\alpha}, \bar{\alpha} \in \mathcal{K}_\infty$[2] *and a constant* $\lambda \in (0, 1)$ *such that the following hold:*

$$\underline{\alpha}(\|x - x_{\text{goal}}\|) \le V(x) \le \bar{\alpha}(\|x - x_{\text{goal}}\|) \tag{2a}$$

$$\inf_{u\in\mathcal{U}} V\left(h(x, u)\right) \le \lambda V(x) \tag{2b}$$

The set of input $\mathcal{K}(x) = \{u \in \mathcal{U} \mid V\left(h(x, u)\right) \le \lambda V(x)\}$ is called *stablizing control inputs*. It is a standard result that if $\mathcal{G}$ is forward invariant[3] and the goal point $x_{\text{goal}} \in \mathcal{G} \subseteq \mathcal{X}$, then starting from initial set $\mathcal{X}_0 \subseteq \mathcal{G}$, any control input $u \in \mathcal{K}$ will make the closed-loop system asymptotically stable at x_{goal} (Grüne et al., 2017). The details are provided at Appendix A.

4. LYGE Algorithm

Notation: Let τ be the current iteration step in our algorithm. A *dataset* $\mathcal{D}^\tau$ is a set of N_{D^τ} transitions collected by the current iteration step. Recall that $\mathcal{D}^0$ is the set of given demonstrations. Let $\mathcal{D}^\tau_x \subset \mathcal{X}$ be the projection of $\mathcal{D}^\tau$ on the first state component of $\mathcal{D}^\tau$ defined as $\mathcal{D}^\tau_x = \{x \mid (x, \cdot, \cdot) \in \mathcal{D}^\tau\}$. A *trusted tunnel* $\mathcal{H}^\tau$ is defined as the set of states at most $\gamma > 0$ distance away from the dataset $\mathcal{D}^\tau_x$, *i.e.*, $\mathcal{H}^\tau = \{x \mid \exists x_i \in \mathcal{D}^\tau_x, \|x - x_i\| \le \gamma\}$.

We first outline our algorithm **LY**apunov-**G**uided **E**xploration (LYGE), which learns to stabilize high-dimensional unknown systems, then provide a step-by-step explanation (see Appendix B for detailed theoretical analysis). Given a dataset of imperfect demonstrations $\mathcal{D}^0$ (which may not contain x_{goal}), we firstly employ IL to learn an initial controller π_{init} (which may be an unstable controller). Then, during each iteration τ, we learn a CLF V^τ_θ and the corresponding controller π^τ_ϕ using samples from $\mathcal{D}^\tau_x$, and use the learned controller π^τ_ϕ to generate closed-loop trajectories as additional data added to $\mathcal{D}^\tau$ to obtain $\mathcal{D}^{\tau+1}$. At each iteration, the learned CLF V^τ_θ ensures that the

---

[2] A function $\alpha : [0, +\infty) \to [0, +\infty)$ is said to be class-$\mathcal{K}_\infty$ if α is continuous, strictly increasing with $\alpha(0) = 0$, and $\lim_{s\to+\infty} \alpha(s) = +\infty$.

[3] A set $\mathcal{G}$ is forward invariant if $x(0) \in \mathcal{G} \implies x(t) \in \mathcal{G}$ for all $t > 0$.

newly collected trajectories get closer to x_{goal}, as each V_θ^τ is designed to reach the global minimum at x_{goal}. In this way, the trusted tunnel $\mathcal{H}^\tau$ grows and includes states closer and closer to x_{goal} with more iterations. Upon convergence, LYGE returns a stabilizing controller π^* that can be trusted within the converged trusted tunnel $\mathcal{H}^*$, where $\mathcal{H}^*$ contains all trajectories starting from $\mathcal{X}_0$.

Learning from Demonstrations: At iteration 0, our approach starts with learning an initial policy from imperfect and potentially unstable demonstrations $\mathcal{D}^0$. We use existing IL methods (*e.g.*, BC (Bain and Sammut, 1995)) to learn the initial policy $\pi_{\text{init}}(\cdot)$. Since the IL algorithms directly recover the behavior of the demonstrations, the initial policy could lead to unstable behaviors.

Learning Local Dynamics: At iteration τ, we learn a discrete NN approximation of the dynamics $\hat{h}_\psi^\tau(x,u) : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ parameterized by ψ using the transition dataset $\mathcal{D}^\tau$. We train $\hat{h}_\psi^\tau$ by minimizing the mean square error loss from transitions sampled from the dataset $\mathcal{D}^\tau$ using Adam (Kingma and Ba, 2014). Let $\omega > 0$ be the maximum error of the learned dynamics on the training data: $\|\hat{h}_\psi^\tau(x_i(t), u_i(t)) - x_i(t+1)\| \leq \omega$ for all $(x_i(t), u_i(t), x_i(t+1)) \in \mathcal{D}^\tau$.

Learning CLF and Controller: After learning the local dynamics, we jointly learn the CLF and the control policy. Let the learned CLF in τ-th iteration be $V_\theta^\tau(x) = x^\top S^\top S x + p_{\text{NN}}(x)^\top p_{\text{NN}}(x)$, where $S \in \mathbb{R}^{n_x \times n_x}$ is a matrix of parameters, $p_{\text{NN}} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ denotes an NN, and θ encompasses all parameters including S and the ones in p_{NN}. Clearly, $V_\theta^\tau(x)$ is positive by construction. The first term in V_θ^τ models a quadratic function, which is commonly used to construct Lyapunov functions for linear systems. Here we use it to introduce a quadratic prior to the learned CLF. The second term models the CLF's non-quadratic residue. We also parameterize the controller with an NN, *i.e.*, $\pi_\phi^\tau : \mathcal{X} \to \mathcal{U}$ with parameters ϕ. The learned controller aims to direct system trajectories toward the goal, guided by the learned CLF. Additionally, since the learned dynamics $\hat{h}_\psi^\tau$ may be invalid outside the trusted tunnel $\mathcal{H}^\tau$, π_ϕ^τ should not drive the system too far from $\mathcal{H}^\tau$ within the simulation horizon. This can be achieved by penalizing the distance of the control inputs in consecutive iterations. Overall, V_θ^τ and π_ϕ^τ can be synthesized by solving the following optimization problem:

$$\min_{\phi} \quad \mathbb{E}_{x \in \mathcal{H}^\tau} \|\pi_\phi^\tau(x) - \pi_\phi^{\tau-1}(x)\| \tag{3a}$$

$$\text{s.t.} \quad V_\theta^\tau(x_{\text{goal}}) = 0, \qquad V_\theta^\tau(x) > 0, \quad \forall x \in \mathcal{X} \setminus x_{\text{goal}} \tag{3b}$$

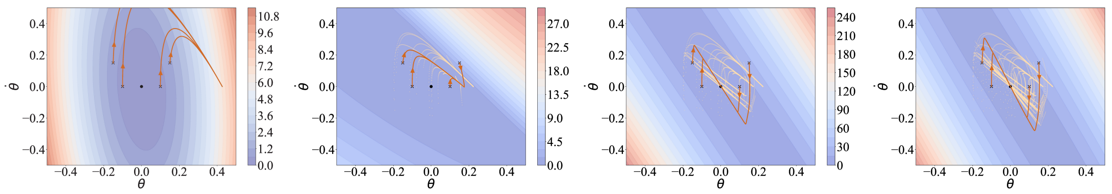
$$V_\theta^\tau\left(\hat{h}_\psi^\tau(x, \pi_\phi^\tau(x))\right) \leq \lambda V_\theta^\tau(x), \quad \forall x \in \mathcal{H}^\tau \tag{3c}$$

where $\lambda \in (0,1)$. We approximate the solution of problem (3) by self-supervised learning with loss $\mathcal{L}^\tau = \mathcal{L}_{\text{CLF}}^\tau + \eta_{\text{ctrl}} \mathcal{L}_{\text{ctrl}}^\tau$, where $\mathcal{L}_{\text{CLF}}^\tau$ and $\mathcal{L}_{\text{ctrl}}^\tau$ correspond to the constraints and the objective of Problem (3), respectively, $\eta_{\text{ctrl}} > 0$ is a hyper-parameter that balances the weights of two losses, and

$$\mathcal{L}_{\text{CLF}}^\tau = V_\theta^\tau(x_{\text{goal}})^2 + \frac{1}{N} \sum_{x \in \mathcal{X} \setminus x_{\text{goal}}} [\nu - V_\theta^\tau(x)]^+ + \frac{\eta_{\text{pos}}}{N} \sum_{x \in \mathcal{D}_x^\tau} \left[\epsilon + V_\theta^\tau\left(\hat{h}_\psi^\tau(x, \pi_\phi^\tau(x))\right) - \lambda V_\theta^\tau(x)\right]^+ \tag{4}$$

$$\mathcal{L}_{\text{ctrl}}^\tau = \frac{1}{N} \sum_{x \in \mathcal{D}_x^\tau} \|\pi_\phi^\tau(x) - \pi_\phi^{\tau-1}(x)\|^2, \tag{5}$$

where $[\cdot]^+ = \max(\cdot, 0)$, $\eta_{\text{pos}} > 0$ is a hyper-parameter that balances each term in the loss, and $\epsilon > 0$ is a hyper-parameter that ensures that the learned CLF V_θ^τ satisfies condition (2b) even if the learned dynamics $\hat{h}_\psi^\tau$ has errors (Studied in Section 5.4). In practice, it is hard to enforce $V_\theta^\tau(x_{\text{goal}})$ to be exactly 0 using gradient-based methods. Therefore, in loss (4) and (5), we instead train the NNs to make $V_\theta^\tau(x_{\text{goal}}) < \nu$, and $V_\theta^\tau(x) \geq \nu$, for all $x \in \mathcal{X} \setminus x_{\text{goal}}$, where ν is a small positive number.



(*a*) Iteration 0 (*b*) Iteration 2 (*c*) Iteration 6 (*d*) Iteration 12

Figure 1: Trajectories generated by LYGE in different iterations in inverted pendulum environment. The counters show the learned CLF. The orange trajectories are generated by the learned controller in the current iteration. The light orange dots are the demonstrations generated in previous iterations, which also indicate the trusted tunnel $\mathcal{H}$. The black dot is the goal.

Exploration: After learning $\hat{h}_\psi^\tau$, V_θ^τ, and π_ϕ^τ in the τ-th iteration, we use the current controller π_ϕ^τ starting from $x_0 \in \mathcal{H}^\tau$ to collect N_D more transitions $\Delta\mathcal{D}^\tau = \{(x_i(t), \pi_\phi^\tau(x_i(t)), x_i(t+1))\}_{i=1}^{N_D}$ and augment the dataset $\mathcal{D}^{\tau+1} = \mathcal{D}^\tau \cup \Delta\mathcal{D}^\tau$. During the exploration, the controller drives the system to states with lower CLF values. Consequently, by collecting more trajectories with the learned policy π_ϕ^τ, we expand the trusted tunnel $\mathcal{H}^\tau$ to states with lower CLF values. As the global minimum of the CLF is at x_{goal}, the system trajectories used to construct the trusted tunnel keep getting closer to the goal in each iteration. The detailed algorithm and the convergence result are provided in Appendix B.

We illustrate the LYGE process in an inverted pendulum environment in Figure 1. In Figure 1(*a*), since the demonstrations are imperfect, our initial controller π_{init} cannot reach the goal. Figure 1(*b*) and Figure 1(*c*) demonstrate that after several iterations, the trusted tunnel $\mathcal{H}^\tau$ (the region around the light orange dots $\mathcal{D}_x^\tau$) is expanded towards the goal, and the closed-loop system progressively approaches the goal. Upon convergence (Figure 1(*d*)), our controller stabilizes the system at the goal.

5. Experiments

We conduct experiments in six environments including Inverted Pendulum, Cart Pole, Cart II Pole, Neural Lander (Shi et al., 2019), and the F-16 jet (Heidlauf et al., 2018) with two tasks: ground collision avoidance (GCA) and tracking. To simulate the imperfect demonstrations, We collect imperfect and potentially unstable demonstrations for each environment using nominal controllers such as LQR (Kwakernaak and Sivan, 1969) for Inverted Pendulum, PID (Bennett, 1996) for Neural Lander and the F-16, and RL controllers for Cart Pole and Cart II Pole. In the first four environments, we collect 20 trajectories as demonstrations, and in the two F-16 environments, we collect 40 trajectories. We aim to answer the following questions in the experiments: 1) How does LYGE compare with other algorithms for the case of stabilizing the system at goal? 2) What is the sampling efficiency of LYGE as compared to other baseline methods? 3) Can LYGE be used for systems with high dimensions? We provide additional implementation details and more results in Appendix C.

5.1. Baselines

We compare LYGE with the most relevant works in our problem setting including RL algorithm PPO (Schulman et al., 2017), standard IL algorithm AIRL (Fu et al., 2018), and algorithms of IL from suboptimal demonstrations D-REX (Brown et al., 2020) and SSRR (Chen et al., 2021). For PPO, we hand-craft reward functions following standard practices in reward function design (Brockman et al., 2016) for stabilization problems. For AIRL, D-REX, and SSRR, we let them learn directly from the demonstrations. For a fair comparison, we initialize all the algorithms with the BC policy. Compared with these baselines, our algorithm has one additional assumption that we know the desired goal point. However, we believe that the comparison is still fair because we do not need the reward function or optimal demonstrations. While LYGE needs more information than D-REX and SSRR, the performance increment from LYGE is large enough that it is worth the additional information.

We also design two other baselines, namely, CLF-sparse and CLF-dense, to show the efficacy of the Lyapunov-guided exploration compared with learning the dynamics model from random samples (Dai et al., 2021; Zhou et al., 2022). These methods require the stronger assumption of being able to sample from arbitrary states in the state space, which is unrealistic when performing experiments outside of simulations. For both algorithms, we follow the same training process as LYGE, but instead of collecting samples of transitions by applying Lyapunov-guided exploration, we directly sample states and actions from the entire state-action space to obtain the training set of the dynamics. We note that CLF-sparse uses the same number of samples as LYGE, while CLF-dense uses the same number of samples as the RL and IL algorithms, which is much more than LYGE (see Table 1).

5.2. Environments

Inverted Pendulum: Inverted Pendulum (Inv Pendulum) is a standard benchmark for testing control algorithms. To simulate imperfect demonstrations, the demonstration data is collected by an LQR controller with noise that leads to oscillation of the pendulum around a point away from the goal.

Cart Pole and Cart II Pole: Both environments are standard RL benchmarks introduced in Open AI Gym (Brockman et al., 2016)[4]. We collect demonstrations using an RL policy that has not fully converged, which makes the cart pole oscillate at a location away from the goal.

Neural Lander: Neural lander (Shi et al., 2019) is a widely used benchmark for systems with unknown disturbances. The state space has 6 dimensions including the 3-dimensional position and the 3-dimensional velocity, with 3-dimensional linear acceleration as the control input. The goal is to stabilize the neural lander at a user-defined point near the ground. The dynamics are modeled by a neural network trained to approximate the aerodynamics ground effect, which is highly nonlinear and unknown. We use a PID controller to collect demonstrations, which makes the neural lander oscillate and cannot reach the goal point because of the strong ground effect.

F-16: The F-16 model (Heidlauf et al., 2018; Djeumou et al., 2021) is a high-fidelity fixed-wing fighter model, with 16D state space and 4D control inputs. The dynamics are complex and cannot be described as ODEs. Instead, the authors of the F-16 model provide many lookup tables to describe the dynamics. We solve the two tasks discussed in the original papers: ground collision avoidance (GCA) and waypoint tracking. In GCA, the F-16 starts at an initial condition with the head pointing at the ground. The goal is to pull up the aircraft as soon as possible, avoid colliding with the ground, and fly at a height between 800 ft and 1200 ft. In the tracking task, the goal of the aircraft is to reach a user-defined waypoint. The original model provides PID controllers. However, the original PID controller cannot pull up the aircraft early enough or cannot track the waypoint precisely.

---

[4] Original names are InvertedPendulum and InvertedDoublePendulum in Mujoco environments

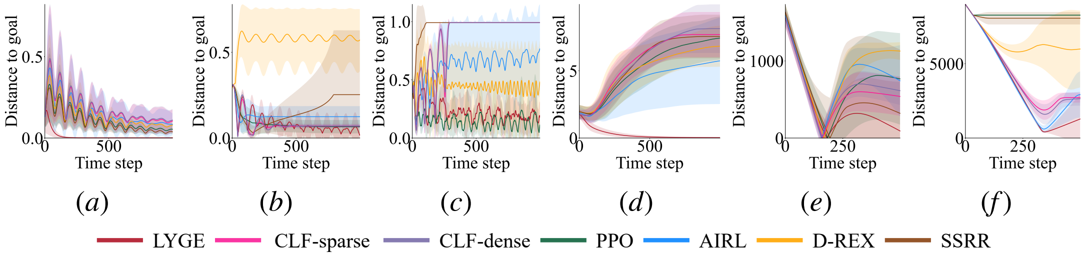


Figure 2: The distance to the goal w.r.t. time step of LYGE and the baselines: (a) Inv Pendulum; (b) Cart Pole; (c) Cart II Pole; (d) Neural Lander; (e) F-16 GCA; (f) F-16 Tracking. The solid lines show the mean distance while the shaded regions show the standard deviation. Note that the curve corresponding to CLF-sparse almost overlaps with the curve corresponding to CLF-dense and so, the curve for CLF-sparse might not be visible in some of the plots.

5.3. Results and Discussions

We train each algorithm in each environment 5 times with different random seeds and test the converged controllers 20 times each. In Figure 2 we show the distance to the goal w.r.t. the simulation time steps. Note that we consider the goal height in the F-16 GCA environment and the goal position in the F-16 Tracking environment. We can observe that LYGE achieves comparable or better results in terms of stabilizing the systems in all environments, especially in high-dimensional complex systems like Neural Lander and F-16. In Neural Lander, none of the baselines can reach the goal as they prioritize flying at a higher altitude to avoid collisions. In F-16 environments, the baselines either pull up the aircraft too late or have large tracking errors. LYGE however, can finish these tasks perfectly.

Specifically, compared with PPO, LYGE achieves comparable results in simpler environments like Cart Pole and Cart II Pole, and behaves much better in complex environments like Neural Lander and F-16. This is due to the fact that PPO is a policy gradient method that approximates the solution of the Bellman equation, but getting an accurate approximation is hard for high-dimensional systems. In PPO the reward function can only describe “where the goal is”, but our learned CLF can explicitly tell the system “how to reach the goal”. Compared with AIRL, which learns the same policy as the demonstrations and cannot make improvements, LYGE learns a policy that is much better than the demonstrations. Compared with ranking-guided algorithms D-REX and SSRR, LYGE behaves better because ranking guidances provide less information than our CLF guidance, and are not designed to explicitly encode the objective of reaching the goal. Compared with CLF-sparse and CLF-dense, LYGE outperforms them because Lyapunov-guided exploration provides a more effective way to sample in the state space to learn the dynamics rather than random sampling. Although CLF-dense uses many more samples than CLF-sparse and LYGE its performance does not significantly improve. This shows that naïvely increasing the number of samples without guidance does little to improve the accuracy of the learned dynamics in high-dimensional spaces.

In Table 1, we show the number of samples used in the training. It is shown that LYGE needs 68% to 95% fewer samples than other algorithms. This indicates that our Lyapunov-guided exploration explores only the necessary regions in the state space and thus improving the sample efficiency.

Table 1: Number of samples (k) used for training in different environments.

Algorithm	Inv Pendulum	Cart Pole	Cart II Pole	Neural Lander	F-16 GCA	F-16 Tracking
LYGE	160	160	480	240	480	560
Baselines	2000	500	5000	5000	2000	10000

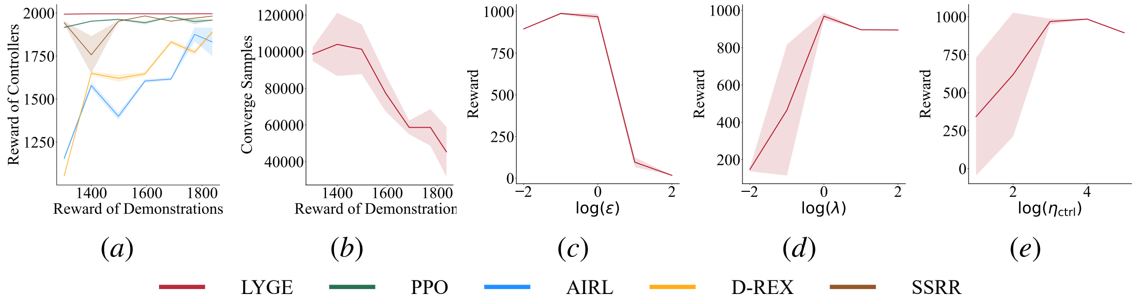


Figure 3: Ablation studies. (a) The converged reward w.r.t. demonstration rewards. (b) The number of samples used before LYGE converges w.r.t. demonstration rewards. (c) The converged reward w.r.t. ϵ. (d) The converged reward w.r.t. λ. (e) The converged reward w.r.t. η_{ctrl}.

5.4. Ablation Studies

We first show the influence of the optimality of the demonstrations. We use the Inverted Pendulum environment and collect demonstrations with different levels of optimality by varying the distance between the actual goal state and the target state of the demonstrator controller. For each optimality level, we train each algorithm 3 times with different random seeds and test each converged controller 20 times. We omit the experiments on CLF-sparse and CLF-dense since their performances are not related to the demonstrations. The results are shown in Figure 3(*a*). We observe that LYGE outperforms other algorithms with demonstrations at different levels of optimality. In addition, we do not observe a significant performance drop of LYGE as the demonstrations become worse. This is because the quality of the demonstrations only influences the convergence speed of LYGE, instead of the controller. PPO's behavior is also consistent since the reward function remains unchanged, but it consistently performs worse than LYGE. IL algorithms, however, have a significant performance drop as the demonstrations get worse because they all depend on the quality of the demonstrations.

The quality of the demonstrations can also influence the convergence speed of LYGE. In the Inv Pendulum environment, we define the algorithm converges when the reward is larger than 1980. We plot the number of samples used for convergence w.r.t. the reward of demonstrations in Figure 3(*b*). It is shown that the better the given demonstrations, the fewer samples LYGE needs for convergence.

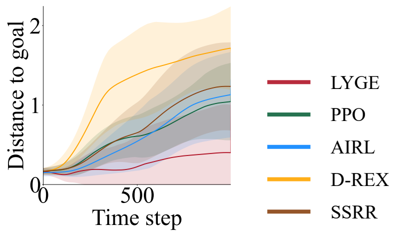
We also do ablations to investigate the influence of the hyperparameter ϵ. We test LYGE in the Cart Pole environment and change ϵ from 0.01 to 100. The results are shown in Figure 3(*c*), which demonstrates that LYGE works well when ϵ is large enough to satisfy the condition introduced in Theorem 7 in Appendix B, and small enough that it does not make the training very hard.

The λ in Equation (4) is another hyperparameter that controls the convergence rate of the learned policy. We test LYGE in the Cart Pole environment and change λ from 0.01 to 100. The results are shown in Figure 3(*d*), demonstrating that LYGE works well with λ in some range. If λ is too small, the convergence rate is too small and the system cannot be stabilized within the simulation time steps. If λ is too large, loss (4) becomes too hard to converge, so the controller cannot stabilize the system.

During training, η_{ctrl} controls the expansion of the trusted tunnel. We do ablations for η_{ctrl} in the Cart Pole environment and change η_{ctrl} from 10 to 10^5. The results are shown in Figure 3(*e*), which suggests that LYGE can work well when the value of η_{ctrl} is within a certain range. If η_{ctrl} is too small, the system leaves the trusted tunnel too early instead of smoothly expanding the trusted tunnel. If η_{ctrl} is too large, exploration is strongly discouraged and the trusted tunnel expands too slowly.

6. Extensions

Our framework is general since it can be directly applied to learn controllers guided by other certificates in environments with unknown dynamics. For example, Control Contraction Metrics (CCMs) are differential analogs of Lyapunov functions (proving stability in the tangent state space). A metric is called CCM if it satisfies a list of conditions, and a valid CCM can guarantee the convergence of tracking controllers. The similarity between CCM and CLF suggests that tracking controllers can also be learned with a similar framework. We change the learning CLF part to learning CCM algorithms (Sun et al., 2021; Chou et al., 2021), and use the same framework to learn the local dynamics, a tracking controller, and a CCM to guide the exploration of the tracking controller. We test this modification in a Dubins car path tracking environment. As shown in Figure 4, our algorithm outperforms the baselines. We explain the details in Appendix E.



(*a*) (*b*)

Figure 4: The tracking error w.r.t. time step in Dubins car path tracking environment.

7. Conclusion

We propose a general learning framework, LYGE, for learning stabilizing controllers in high-dimensional environments with unknown dynamics. LYGE iteratively fits the local dynamics to form a trusted tunnel, and learns a control policy with a CLF to guide the exploration and expand the trusted tunnel toward the goal. Upon convergence, the controller stabilizes the closed-loop system at the goal point. We provide experimental results to demonstrate that LYGE performs comparably or better than the baseline RL, IL, and neural certificate methods. We also demonstrate that the same framework can be applied to learn other certificates in environments with unknown dynamics.

Our framework has a few **limitations**: we require a set of demonstrations for initialization in high-dimensional systems, although they can be potentially imperfect. Without them, LYGE may take a long time to expand the trusted tunnel to the goal. In addition, we need Lipschitz assumptions for the dynamics to derive the theoretical results. If the dynamics do not satisfy the Lipschitz assumptions, the learned CLF might be invalid even inside the trusted tunnel. Moreover, although we observe the convergence of the loss terms in training on all our case studies, it is hard to guarantee that the loss always converges on any system. Finally, if we desire a fully validated Lyapunov function, we need to employ formal verification tools, and we provide a detailed discussion in Appendix D.

Acknowledgments

This work was partly supported by the National Science Foundation (NSF) CAREER Award #CCF-2238030 and the MIT-DSTA program. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and don't necessarily reflect the views of the sponsors.

References

Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.

Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. Fossil: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

Amir Ali Ahmadi and Anirudha Majumdar. Some applications of polynomial optimization in operations research and real-time decision making. *Optimization Letters*, 10(4):709–729, 2016.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

Stuart Bennett. A brief history of automatic control. *IEEE Control Systems Magazine*, 16(3):17–25, 1996.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

Ruxandra Bobiti and Mircea Lazar. Automated-sampling-based stability verification and doa estimation for nonlinear systems. *IEEE Transactions on Automatic Control*, 63(11):3659–3674, 2018.

Nicholas Boffi, Stephen Tu, Nikolai Matni, Jean-Jacques Slotine, and Vikas Sindhwani. Learning stability certificates from data. In *Conference on Robot Learning*, pages 1341–1350. PMLR, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pages 330–359. PMLR, 2020.

Zhangjie Cao and Dorsa Sadigh. Learning from imperfect demonstrations from agents with varying dynamics. *IEEE Robotics and Automation Letters*, 6(3):5231–5238, 2021.

Fernando Castaneda, Jason J Choi, Bike Zhang, Claire J Tomlin, and Koushil Sreenath. Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and dynamics. In *2021 American Control Conference (ACC)*, pages 3683–3690. IEEE, 2021.

Ya-Chien Chang and Sicun Gao. Stabilizing neural control using self-learned almost lyapunov critics. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1803–1809. IEEE, 2021.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Conference on Robot Learning*, pages 1262–1277. PMLR, 2021.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

Jason Choi, Fernando Castañeda, Claire J Tomlin, and Koushil Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. In *Robotics: Science and Systems (RSS)*, 2020.

Glen Chou, Necmiye Ozay, and Dmitry Berenson. Uncertainty-aware constraint learning for adaptive safe motion planning from demonstrations. In *Conference on Robot Learning*, 2020.

Glen Chou, Necmiye Ozay, and Dmitry Berenson. Model error propagation via learned contraction metrics for safe feedback motion planning of unknown systems. *arXiv preprint arXiv:2104.08695*, 2021.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.

Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods. *arXiv preprint arXiv:2202.11762*, 2022a.

Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022b.

Franck Djeumou, Aditya Zutshi, and Ufuk Topcu. On-the-fly, data-driven reachability analysis and control of unknown systems: an f-16 aircraft case study. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–2, 2021.

Milad Farsi, Yinan Li, Ye Yuan, and Jun Liu. A piecewise learning framework for control of unknown nonlinear systems with stability guarantees. In *Learning for Dynamics and Control Conference*, pages 830–843. PMLR, 2022.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Nathan Gaby, Fumin Zhang, and Xiaojing Ye. Lyapunov-net: A deep neural network architecture for lyapunov function approximation. *arXiv preprint arXiv:2109.13359*, 2021.

Sicun Gao, Jeremy Avigad, and Edmund M Clarke. δ-complete decision procedures for satisfiability over the reals. In *International Joint Conference on Automated Reasoning*, pages 286–300. Springer, 2012.

Peter Giesl and Sigurdur Hafstein. Review on computational methods for lyapunov functions. *Discrete & Continuous Dynamical Systems-B*, 20(8):2291, 2015.

Lars Grüne, Jürgen Pannek, Lars Grüne, and Jürgen Pannek. *Nonlinear model predictive control*. Springer, 2017.

Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters*, 5(4):6217–6224, 2020.

Peter Heidlauf, Alexander Collins, Michael Bolender, and Stanley Bak. Verification challenges in f-16 ground collision avoidance and other automated maneuvers. In *ARCH@ ADHS*, pages 208–217, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Rushikesh Kamalapurkar, Joel A Rosenfeld, and Warren E Dixon. Efficient model-based reinforcement learning for approximate online optimal control. *Automatica*, 74:247–258, 2016.

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

Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*, volume 1072. Wiley-interscience New York, 1969.

Shenyu Liu, Daniel Liberzon, and Vadim Zharnitsky. Almost lyapunov functions for nonlinear systems. *Automatica*, 113:108758, 2020.

Johan Lofberg. Pre-and post-processing sum-of-squares programs in practice. *IEEE transactions on automatic control*, 54(5):1007–1011, 2009.

Kehan Long, Yinzhuang Yi, Jorge Cortés, and Nikolay Atanasov. Distributionally robust lyapunov function search under uncertainty. In *Learning for Dynamics and Control Conference*, pages 864–877. PMLR, 2023.

Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. In *2013 IEEE International Conference on Robotics and Automation*, pages 4054–4061. IEEE, 2013.

Ian R Manchester and Jean-Jacques E Slotine. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control*, 62(6):3046–3053, 2017.

Arash Mehrjou, Mohammad Ghavamzadeh, and Bernhard Schölkopf. Neural lyapunov redesign. *Proceedings of Machine Learning Research vol*, 144:1–24, 2021.

Youngjae Min, Spencer M Richards, and Navid Azizan. Data-driven control with inherent lyapunov stability. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 6032–6037. IEEE, 2023.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Ellen Novoseller, Yibing Wei, Yanan Sui, Yisong Yue, and Joel Burdick. Dueling posterior sampling for preference-based reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, pages 1029–1038. PMLR, 2020.

Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Frank Permenter and Pablo Parrilo. Partial facial reduction: simplified, equivalent sdps via approximations of the psd cone. *Mathematical Programming*, 171(1):1–54, 2018.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Zengyi Qin, Yuxiao Chen, and Chuchu Fan. Density constrained reinforcement learning. In *International Conference on Machine Learning*, pages 8682–8692. PMLR, 2021.

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(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2586–2591, 2007.

Hadi Ravanbakhsh and Sriram Sankaranarayanan. Learning control lyapunov functions from counterexamples and demonstrations. *Autonomous Robots*, 43(2):275–307, 2019.

Spencer M Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on Robot Learning*, pages 466–476. PMLR, 2018.

Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020.

Ugo Rosolia and Francesco Borrelli. Learning how to autonomously race a car: a predictive control approach. *IEEE Transactions on Control Systems Technology*, 28(6):2713–2719, 2019.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6): 233–242, 1999.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9784–9790. IEEE, 2019.

Oswin So and Chuchu Fan. Solving stabilize-avoid optimal control via epigraph form and deep reinforcement learning. In *Proceedings of Robotics: Science and Systems*, 2023.

Dawei Sun, Susmit Jha, and Chuchu Fan. Learning certified control using contraction metric. In *Conference on Robot Learning*, pages 1519–1539. PMLR, 2021.

Voot Tangkaratt, Bo Han, Mohammad Emtiyaz Khan, and Masashi Sugiyama. Variational imitation learning with diverse-quality demonstrations. In *International Conference on Machine Learning*, pages 9407–9417. PMLR, 2020.

Voot Tangkaratt, Nontawat Charoenphakdee, and Masashi Sugiyama. Robust imitation learning from noisy demonstrations. In *AISTATS*, 2021.

Lizhi Wang, Songyuan Zhang, Yifan Zhou, Chuchu Fan, Peng Zhang, and Yacov A. Shamash. Physics-informed, safety and stability certified neural control for uncertain networked microgrids. *IEEE Transactions on Smart Grid*, pages 1–1, 2023. doi: 10.1109/TSG.2023.3309534.

Steven Wang, Sam Toyer, Adam Gleave, and Scott Emmons. The `imitation` library for imitation learning and inverse reinforcement learning. https://github.com/HumanCompatibleAI/imitation, 2020.

Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pages 6818–6827. PMLR, 2019.

Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021.

Songyuan Zhang, Zhangjie Cao, Dorsa Sadigh, and Yanan Sui. Confidence-aware imitation learning from demonstrations with varying optimality. *Advances in Neural Information Processing Systems*, 34, 2021.

Songyuan Zhang, Yumeng Xiu, Guannan Qu, and Chuchu Fan. Compositional neural certificates for networked dynamical systems. In *Learning for Dynamics and Control Conference*, pages 272–285. PMLR, 2023.

Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural lyapunov control of unknown nonlinear systems with stability guarantees. *arXiv preprint arXiv:2206.01913*, 2022.

Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 3*, pages 1433–1438, 2008.