DOPT: D-learning with Off-Policy Target toward Sample Efficiency and Fast Convergence Control

Zhaolong Shen and Quan Quan*

***Abstract*— In recent times, Lyapunov theory has been incorporated into learning-based control methods to provide a stability guarantee. However, merely satisfying the Lyapunov conditions does not fully leverage the capabilities of the Neural Network (NN) controller. Furthermore, training an effective Lyapunov candidate requires substantial data, which inherently results in sample inefficiency. To address these limitations, we propose an off-policy variant of the vanilla D-learning method that uses current and historical data to iteratively enhance the NN controller within the framework of Lyapunov theory. Our method outperforms the Deep Deterministic Policy Gradient (DDPG) and D-learning in terms of stability, sample efficiency, and the quality of the trained controllers and Lyapunov candidates. Link to code: github.com/Shenzhaolong1330/DOPT**

I. INTRODUCTION

Learning-based Lyapunov control and stability verification methods have been proposed and widely adopted in robotics and safety-critical control systems. Studies in [1]–[4] concentrate on learning rigorous Lyapunov stability certificates and estimating a maximal Region of Attraction (RoA) for a given dynamical system. Besides stability certificates, [5]–[8] use Lyapunov conditions as an effective guide in optimization to search for controllers that can stabilize the system. In the Reinforcement Learning (RL) community, a survey conducted in [9] has introduced various applications of Lyapunov theory in safe learning. Lyapunov theory is leveraged to achieve two goals: acquiring stable policy and ensuring training safety. Lyapunov functions assume various roles across different studies. Specifically, they are employed as critic functions in the works of [10], [11], where they contribute to the evaluation of policy performance. In [12], [13], Lyapunov functions are integrated as additional constraints in their optimization frameworks to ensure system stability. Furthermore, [14], [15] innovate by incorporating Lyapunov conditions into the reward structure, aligning stability with the agent's learning goals.

Notwithstanding the advancements in Learning-based Lyapunov Control (LLC) methods, two significant areas remain ripe for improvement. **1) The Sample and Data Efficiency.** LLC methods are subject to constraints, including the necessity for copious training data and an inherent sample inefficiency. To accurately model an expressive Lyapunov landscape, it is imperative to implement a comprehensive and uniform sampling across the Region of Interest (RoI) [6], [7]. Sampling in the form of trajectories is also feasible [15], [16], as evidenced by its practical applications in the real world [14], [17]. However, the sparsity of trajectory samples may present a challenge for Lyapunov candidate training. To provide the most accurate stability certificate, once the controller is changed, the system dynamics evolve, and the Lyapunov candidate must be retrained using newly sampled data [15]. This determines the on-policy characteristic, which subsequently leads to sample inefficiency. **2) The Enhancement of NN Controllers.** The current literature [1]–[8] on LLC methods emphasizes that strictly satisfying the Lyapunov conditions is essential for stability. Nevertheless, merely meeting these conditions is inadequate for fully exploiting the potential of NN controllers. Consequently, these approaches may fall short of meeting the demands of scenarios that require agility in responding to disturbances or rapid convergence to the equilibrium. The formal enhancement of NN controllers in LLC methods for accelerated convergence remains an open question.

In light of these requirements, we propose the D-learning with Off-Policy Target (DOPT) method as a potential solution. This method is specifically developed to train controllers that ensure accelerated convergence and are equipped with stability certificates while improving the efficiency of training sample utilization. D-learning [18], a sample-based model-free approach for continuous control learning, is a parallel method to Q-learning [19] in RL. It collects system data and improves controllers with trained Lyapunov functions and D-functions. Prior research in [18] has provided a rig-

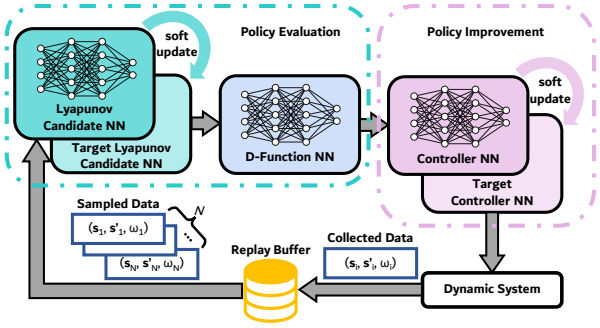

Fig. 1. Overview of the D-learning with Off-Policy Target (DOPT) algorithm: there are five trainable parts, the Lyapunov candidate and its target network, the D-function network, the controller and its target network. Initialized and optimized controllers will run in a simulation environment and the state data will be recorded and stored in the replay buffer. Training data batches are randomly selected from the replay buffer and used for the next epoch of policy evaluation and policy improvement.

Zhaolong Shen and Quan Quan (corresponding author) are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, 100191, P.R. China. E-mail: shenzhaolong@buaa.edu.cn; qq_buaa@buaa.edu.cn.

This work was supported in part by the National Natural Science Foundation of China under Grant 62450127.

orous theoretical framework of D-learning and a promising offline implementation. Nevertheless, D-learning is not without shortcomings. **1) Complexity in Optimization**. In D-learning, the presence of multiple objectives and constraints within a single optimization problem impedes the efficiency and effectiveness of the solution process. **2) Instability during Training**. Sparse and inhomogeneous samples may result in an incompletely trained Lyapunov function, which could mislead the controller optimization and further cause fluctuations in the training process. **3) Sample Inefficiency**. D-learning is essentially an on-policy LLC method that requires a large amount of the most recent data to obtain a Lyapunov function, which is time-consuming in practice.

In DOPT, key promotions include:

- To address the sample inefficiency, an off-policy strategy is employed that recycles historical data to train Lyapuonv candidates. Better results are achieved with half of the original sampled data volume.
- Beyond satisfying Lyapunov conditions, in DOPT, NN controllers are enhanced by using the trained D-function as a reference. The convergence of the closed-loop system is optimized locally and globally.
- Complex optimization problems in D-learning are simplified and transformed into a series of unconstrained optimization problems, each of which can be efficiently solved using gradient descent.
- For the purpose of ensuring a steady and smooth training process, the soft update and target networks are employed in training Lyapunov candidates and controllers.

In Section III, a detailed elaboration of DOPT is presented, and the trained Lyapunov candidate is analyzed based on the almost Lyapunov conditions [20]. In Section IV, controllers trained with DDPG, D-learning, and DOPT are evaluated in dynamical systems. Comparisons of these algorithms are presented along three dimensions: training stability, sample efficiency and the performance of the trained controller.

II. BACKGROUND

A. Stability of Dynamical System

A general autonomous closed-loop dynamical system is in the form

$$\dot{x} = f(x,u), \quad u = c(x), \quad x(0) = x_0 \tag{1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the system state, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input, and $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$ is a Lipschitz-continuous flow map, $c : \mathcal{X} \mapsto \mathcal{U}$ is a control function. We can assume $x_0 = 0$ without the loss of generality.

Given a closed-loop dynamical system (1), for some region $\mathcal{D}$ around the origin (specifically an open subset of $\mathbb{R}^n$ containing the origin), if we can construct a scalar, continuously-differentiable function $V(x)$, such that

$$\begin{aligned} &V(x) > 0, \forall x \in \mathcal{D}\backslash\{0\} \quad V(0) = 0, \text{ and} \\ &\dot{V}(x) = \frac{\partial V}{\partial x} f(x, c(x)) \leq 0, \forall x \in \mathcal{D}\backslash\{0\} \quad \dot{V}(0) = 0, \end{aligned} \tag{2}$$

then the origin x_0 is stable in the sense of Lyapunov (i.s.L.). If, additionally, we have

$$\dot{V}(x) = \frac{\partial V}{\partial x} f(x, c(x)) < 0, \forall x \in \mathcal{D}\backslash\{0\}, \tag{3}$$

then the origin is (locally) asymptotically stable.

B. D-learning

D-learning is introduced as a parallel method to Q-learning [19] in RL. It optimizes controllers within the framework of Lyapunov theory for enhanced stability and performance. D-learning can be primarily divided into two steps: the policy evaluation step and the policy improvement step.

In D-learning, assume we have a data set $\mathcal{P}_c = \{(x_i, \dot{x}_i), i = 1, \cdots, N\}$, a stabilizing controller $c_\phi(x)$, and a Lyapunov candidate $V_\theta(x)$. $\mathcal{P}_c$ is collected from observing the state of a stabilized system (1) under the controller $c_\phi(x)$. The Lyapunov candidate satisfies $V_\theta(x_0) = 0$, $V_\theta : \mathcal{X}/\{x_0\} \to \mathbb{R}_+$. The derivative of $V_\theta(x)$ is expressed as

$$\dot{V}_\theta(x) = \frac{\partial V_\theta}{\partial x} f(x,u), \tag{4}$$

on $\mathcal{P}_c$, the derivative of the Lyapunov function satisfies

$$\left.\frac{\partial V_\theta}{\partial x}\right|_{x=x_i} \dot{x}_i \leq -\eta \|x_i\|^2, \tag{5}$$

where $\eta \in \mathbb{R}$ is expected to be positive. Furthermore, to avoid knowing any of the system dynamics, we rewrite $\dot{V}_\theta(x)$ as

$$D_\delta(x,u) = \frac{\partial V_\theta}{\partial x} f(x,u), \tag{6}$$

this new formulation is called D-function since it represents the *derivative* of the Lyapunov candidate and is expected to be *decreasing* in optimization. To train an effective D-function network $D_\delta(x,u)$, an optimization problem can be defined on the data set $\mathcal{P}_c$

$$\left| D_\delta(x_i, c_\phi(x_i)) - \left.\frac{\partial V_\theta}{\partial x}\right|_{x=x_i} \dot{x}_i \right| \leq b. \tag{7}$$

In all, the policy evaluation step can be summarized with an optimization problem formulated as

$$\begin{aligned} \min_{\theta,\delta,a,b>0,\eta\in\mathbb{R}} \quad & -\eta + w_1 a + w_2 b, \\ \text{s.t.} \quad & -D_\delta(x_i, c_\phi(x_i)) - \eta\|x_i\|^2 \geq 0 \\ & \left| D_\delta(x_i, c_\phi(x_i)) - \left.\frac{\partial V_\theta}{\partial x}\right|_{x=x_i} \dot{x}_i \right| \leq b \\ & F(\theta,\delta) \leq a \end{aligned} \tag{8}$$

where w_1 and w_2 are weights, F is a constraint on the trained parameters, and effective Lyapunov candidate and D-function can be obtained by solving this problem.

With an effective D-function network, the optimization problem in the policy improvement step is presented as

$$\begin{aligned} \min_{\phi,\eta\in\mathbb{R}} \quad & -\eta, \\ \text{s.t.} \quad & -D_\delta(x_i, c_\phi(x_i)) - \eta\|x_i\|^2 \geq 0 \\ & F(\delta,\phi) \leq a \end{aligned} \tag{9}$$

in which the controller is optimized toward decreasing the value of the D-function network on the $\mathcal{P}_c$.

III. PROPOSED APPROACH

This section provides a detailed description of the implementation of the DOPT, where NNs are employed as Lyapunov candidates, D-functions, controllers, and target networks. Furthermore, the trained Lyapunov candidates are analyzed with the almost Lyapunov conditions, proving that the trained Lyapunov candidates are valid.

A. D-learning with Off-Policy Target

Algorithm 1 D-learning with Off-Policy Target (DOPT)

1: Randomly initialize Lyapunov candidate network $V_{\theta_0}(x)$ and D-function network $D_{\delta_0}(x,u)$
2: Initialize a stabilizing controller $\pi_{\phi_0}(x)$
3: Initialize target networks $V'_{\theta'_0}(x)$ and $\pi'_{\phi'_0}$ with $\theta'_0 \leftarrow \theta_0$, $\phi'_0 \leftarrow \phi_0$
4: Initialize replay buffer $\mathcal{R}$
5: **for** episodes = $1,\ldots,K$ **do**
6: $\quad\omega \leftarrow \gamma^{k-1}$
7: $\quad$Sample $\mathcal{D}_k = \{(x_i, x'_i, \omega_i), i = 1 \ldots T\}$ with $\pi'_{\phi'_{k-1}}$
8: $\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{D}_k$
9: $\quad$Randomly sample a batch of N (x_i, x'_i, ω_i) from $\mathcal{R}$
$\quad$**Policy Evaluation:**
10: $\quad$Train $V_{\theta_k}(x)$ by minimizing $L_V(\theta_k)$ (Eq.10)
11: $\quad$Update the target Lyapunov candidate

$$\theta'_k \leftarrow \tau\theta_k + (1-\tau)\theta'_{k-1}$$

12: $\quad$Train $D_{\delta_k}(x,u)$ by minimizing $L_D(\delta_k)$ (Eq.12)
$\quad$**Policy Improvement:**
13: $\quad$Improve $\pi_{\phi_k}(x)$ by decreasing $PII(\phi_k)$ (Eq.13)
14: $\quad$Update the target controller

$$\phi'_k \leftarrow \tau\phi_k + (1-\tau)\phi'_{k-1}$$

15: **end for**

The complete procedure of DOPT is delineated in *Algorithm 1*. The policy evaluation step Eq.8 in D-learning is divided and transformed into two unconstrained optimization problems, which are training an effective Lyapunov candidate (line 10) and training a D-function to imitate the derivative of the target Lyapunov candidate (line 12). The policy improvement step Eq.9 is simplified to decrease the $PII(\phi_k)$ (line 13). This algorithm repeats in episodes until the performance of the trained controller converges.

1) Initialization Stage: We first initialize the Lyapunov candidate network, D-function network, and target Lyapunov candidate network with random parameters. As with other LLC methods, a stabilizing controller is a prerequisite. To obtain an NN controller with stabilizing parameters, we pre-train the network using input-output data to imitate an already stabilizing controller. The replay buffer is initialized with the maximum capacity.

We then sample the closed-loop system data with the target controller $\pi'_{\phi'_0}$ from the RoI, the collected data is formatted as (x_i, x'_i, ω_i), where x_i and x'_i denote the state and the subsequent state, ω_i is the sample weight decaying at a constant rate as episodes repeated. The collected data will be stored in the replay buffer.

2) Policy Evaluation: The policy evaluation step includes training the Lyapunov candidate and the D-function, which are implemented separately in DOPT. The Lyapunov candidate is trained by minimizing a Lyapunov risk [5] like weighted loss function $L_V(\theta)$, designed as

$$\begin{aligned} L_V(\theta_k) =& \frac{1}{N}\sum_{i=1}^{N}\omega_i\left(\max(-V_{\theta_k}(x_i),0) + \max(\dot{V}_{\theta_k}(x_i),0)\right) \\ &+V^2_{\theta_k}(0) + \eta\|\theta_k\|^2, \end{aligned} \tag{10}$$

in which, current and historical data sampled from the replay buffer are used. The off-policy strategy is accomplished using the replay buffer, the temporal weight labeling method and the novel weighted Lyapunov risk function $L_V(\theta)$.

In real physical systems, the derivative of state is hard to acquire. Instead, we use the sampled difference to approximate the $\dot{V}_{\theta_k}(x)$

$$\dot{V}_{\theta_k}(x_i) \approx \left(V_{\theta_k}(x'_i) - V_{\theta_k}(x_i)\right)/\Delta t. \tag{11}$$

The rationale behind the $L_V(\theta)$ design is based on two different perspectives. First, from the perspective of ensuring the positive definiteness of the Lyapunov candidate over the sampled domain, the incorporation of historical data compensates for the sampling sparsity and inhomogeneity, thus facilitating regional positive definiteness in a sample-based way. Second, from the perspective of ensuring a negative derivative, the use of historical data does not compromise the plasticity of the Lyapunov candidate trained with current data, provided that the system converges faster after each episode. In exceptional cases where historical data might undermine the current training Lyapunov candidate, the use of episode-dependent weights mitigates the historical data violation and ensures the dominance of the current data.

Inspired by [21], [22], the soft updates for the target Lyapunov candidate network and the subsequent target controller network are executed in lines 11 and 14, respectively, τ is the update rate. In the first episode, the target Lyapunov candidate network must be directly updated, as it is only initialized with random parameters (line 3). Soft updates make the value changes of the target networks smoother, thereby facilitating the training process more stable.

The D-function $D(x,u)$ is a scalar function to state x and control input u. To capture the derivative of the target Lyapunov candidate $\dot{V}'_{\theta'}(x)$, the D-function network $D_\delta(x,u)$ is trained with sampled states, with the objective of minimizing the loss

$$\begin{aligned} L_D(\delta_k) =& \frac{1}{N}\sum_{i=1}^{N}\left(\left(D_{\delta_k}(x_i,\pi_{\phi_k}(x_i)) - \dot{V}'_{\theta'_k}(x_i)\right)^2\right) \\ &+\eta\|\delta_k\|^2. \end{aligned} \tag{12}$$

As shown in Eq. (6), the D-function incorporates the gradient of Lyapunov and system dynamics, such modeling

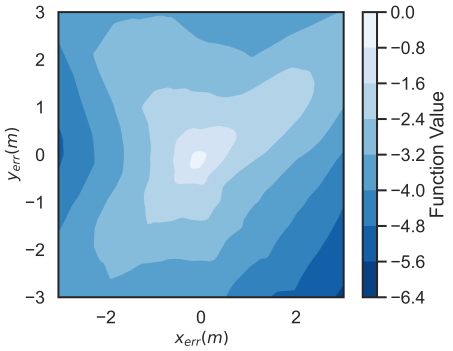

Fig. 2. The landscape of $D_\delta(x, \pi_\phi(x))$ (the trained D-function with trained controller network) for the single-track car system, the coordinate axis is about the position error.

avoids the system identification. Likewise, in Q-learning, Q-function $Q(s, a)$ is the action-state value function which calculates its derivative of action a without going through the system [23]. They are in a similar structure, which contributes to the property of being model-free.

Fig. 2 shows the landscape of a well-trained D-function network with a trained controller network. In the RoI, the value of the D-function is negative, which implies that the Lyapunov value decreases over time.

3) Policy Improvement: The controller is accelerated in the policy improvement step. Since the D-function is directly related to the changing rate of the Lyapunov value over time along the direction of the system dynamics, optimizing the controller network $\pi_{\phi_k}(x)$ based on the D-function value can improve the convergence performance of the system. This step is operated by decreasing the value of an elaborated cost function Policy Improvement Index (PII), defined as

$$PII(\phi_k) = \lambda_1 \max_{i=1\dots N} (D_\delta(x_i, u_i)) \tag{13a}$$

$$+\lambda_2 \sum_{i=1}^{N} \max(D_\delta(x_i, u_i), 0) \tag{13b}$$

$$+\frac{\lambda_3}{N} \sum_{i=1}^{N} D_\delta(x_i, u_i) \tag{13c}$$

$$+\lambda_4 \|\phi_k - \phi_{k-1}\|^2. \tag{13d}$$

In $PII(\phi_k)$, λ_1,λ_2,λ_3 and λ_4 are weights, $u_i = \pi_{\phi_k}(x_i)$, (13a) serves the local improvement by decreasing the minimum upper bound of the sampled D-function values. It plays a role in mitigating steady-state errors. In most cases, the state with the highest D-function value lies near the equilibrium point, as shown in Fig. 2. The following (13b) and (13c) address global optimization. Leveraging all the samples, (13b) forces D-function values to be negative, and (13c) reduces the average of sampled D-function values, ensuring and accelerating the convergence in the RoI. (13d) is designed to limit the deviation from the previous controller and prevent overfitting. The updated controller π_{ϕ_k} must not deviate significantly from the controller $\pi'_{\phi'_{k-1}}$ used for sampling, thus the gradient descent step size m and the policy improvement learning rate must be chosen judiciously to ensure steady and smooth training.

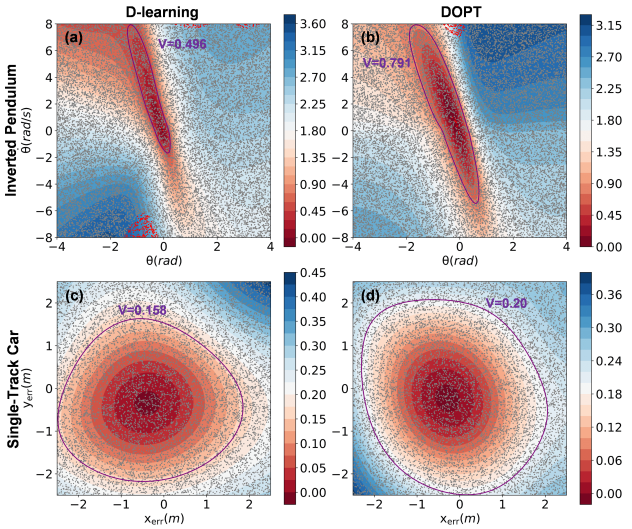

Fig. 3. The landscapes of trained Lyapunov candidates with D-learning and DOPT for inverted pendulum and single-track car systems, sample-based almost Lyapunov verification and estimated RoA (the maximum sublevel set inscribed within the RoI) are overlaid.

B. Verifying Almost Lyapunov Conditions

Besides controller improvement, DOPT provides stability guarantees during training and RoA estimations with successfully trained Lyapunov candidate networks. However, due to the reliance on sample-based approximation, historical data, and soft updates, we can not take it for granted that trained Lyapunov candidate networks can be used as accurate Lyapunov functions in the rigorous definition. Almost Lyapunov conditions are proposed in [20] and are considered as relaxed conditions for stability analysis in [15], [24]. This theorem [20] proves stability by allowing a small and finite set of violation states where the derivative of Lyapunov is positive.

The verification of the almost Lyapunov conditions is conducted through a sample-based method. We first intensively sample states from the RoI and compute the derivative of the trained target Lyapunov candidate network with the true dynamics:

$$\operatorname{sgn}\left(\frac{\partial V'_{\theta'_k}}{\partial x}\bigg|_{x=x_i}^{\mathrm{T}} f(x_i, \pi'_{\phi'_k}(x_i))\right), \tag{14}$$

where $\operatorname{sgn}(\cdot)$ is the sign function, positive represents violation.

Verification results are shown in Fig. 3, in which grey dots represent sampled data and red dots represent violation. Fig. 3a and 3c illustrate the landscapes of Lyapunov candidate networks obtained from D-learning, which correspond to the inverted pendulum and single-track car, respectively. The sublevel sets (purple circle) of the Lyapunov candidates can be found in the verification region, where only a few or no violations exist. This implies there are only limited or no violations in the sublevel sets. These two Lyapunov candidate networks satisfy the almost Lyapunov conditions, and the sublevel sets can be used as estimated RoAs for the

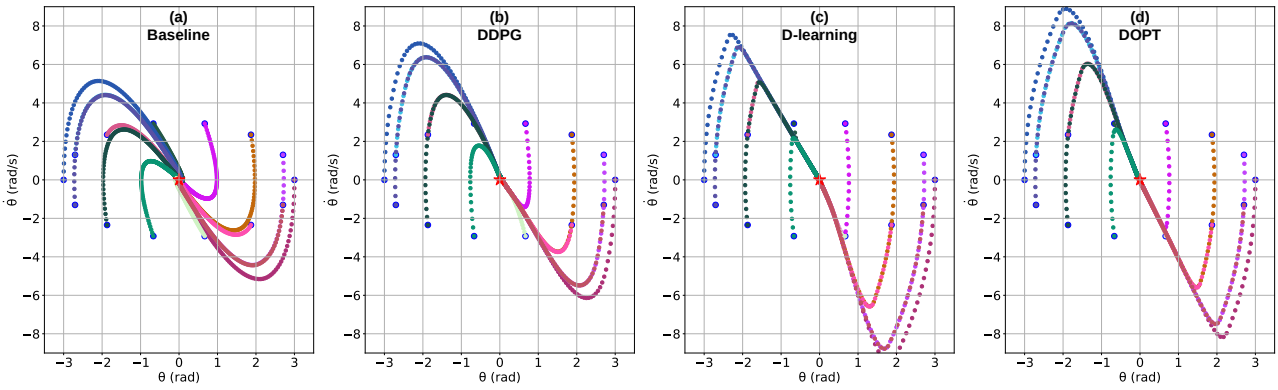

Fig. 4. Sampled trajectories generated by baseline controller and controllers trained with each algorithm in the inverted pendulum environment. Trajectories start with different initial states are represented with different colors, the red star marks the position of the equilibrium point. In this phase portrait, the longer the trajectory stretched along the $\dot{\theta}$ axis, the faster convergence speed is implied.

two studied systems.

Fig. 3b and 3d illustrate the landscapes of DOPT-trained target Lyapunov candidate networks. In the searched sublevel sets, sparse violations can be spotted around $x = [0\ 0]^{\mathrm{T}}$ in Fig. 3b. These two target Lyapunov candidate networks also satisfy the almost Lyapunov conditions, and apparently, the sublevel sets can be used as larger estimated RoAs compared to D-learning.

IV. EXPERIMENTS

In this section, we present experimental results comparing the proposed DOPT method with D-learning [18] and DDPG [21] across stabilization and tracking tasks for nonlinear systems. We use ablation experiments to elucidate the promotions from D-learning to DOPT.

A. Inverted Pendulum

The first example is a classic inverted pendulum stabilization problem. The dynamical system has two states $x = [\theta\ \dot{\theta}]^{\mathrm{T}}$ and one control input torque $u = \tau$. In Fig. 4, sampled trajectories generated by baseline controller and controllers trained with DDPG, D-learning, and DOPT are shown. The baseline controller is a linear quadratic regulator (LQR) [25], as depicted in Fig. 4a. Each algorithm is executed with the same sample volume and same number of iterations. Each controller network is initialized with the same parameters. The reward in DDPG is defined as $-\theta^2 - 0.1\dot{\theta} - 0.001u^2$.

Fig. 4b shows trajectories generated by the DDPG-trained controller. Analyzing together with Fig. 5a, there is a decreasing trend in the convergence steps from the starting position to $\|x\|_2 = 1$ and $\|x\|_2 = 2$, but the trend is unpredictable in the steps to the equilibrium and the accumulated rewards fluctuate as the training processes. Though there is a subtle improvement, pursuing the maximal rewards is not an optimal strategy for training fast converging controllers. RL methods are more suitable for exploration in unknown environments, such as the inverted pendulum transition from unstable to stable.

Fig. 4c and 4d demonstrate the performance of the D-learning and DOPT trained controller. DOPT incorporates a more comprehensive sampling strategy during the optimization process, which results in a more uniformly improved controller across the state space. This is evidenced by the more symmetrical phase trajectories in Fig. 4d compared to those in Fig. 4c. Both Fig. 5b and 5c demonstrate notable reductions in the number of convergence steps, indicating that D-learning and DOPT are effective in training controllers with fast convergence. However, DOPT exhibits steady and consistent decline and narrower variance, with the controller performance reaching convergence after 20 iterations. In contrast, the D-learning process displays greater fluctuations and may not achieve the optimal result.

B. Single-Track Car

A complex nonlinear model of autonomous cars from the CommonRoad benchmarks [26] is chosen to demonstrate the capability of each algorithm in high-dimensional nonlinear control. This model includes seven states $x = [x_e\ y_e\ \delta\ v_e\ \psi_e\ \dot{\psi}_e\ \beta]^{\mathrm{T}}$ and two control inputs $u = [\dot{\delta}\ a]^{\mathrm{T}}$, where x_e, y_e are position error, δ is the steering angle, v_e is the longitudinal velocity error, $\psi_e, \dot{\psi}_e$ are heading angle error and its derivative, β is the lateral slip angle of the vehicle, control inputs are steering angular velocity and longitudinal acceleration. This model considers the nonlinear steering dynamics of a car and the effects of friction between tires and road surface.

We train NN controllers using different algorithms in trajectory tracking scenarios and evaluate their performance in a set of unseen trajectory parameters where a smaller friction coefficient is used to simulate a wet and slippery road surface. The results are presented in Fig. 6. Despite the baseline (LQR) controller and the DDPG-trained controller are able to perform tracking tasks in training scenarios, their performance is significantly degraded by the slippery surface. As depicted in Fig. 6a and 6b, the baseline controller keeps deviating from the trajectory. However, the improvement of DDPG over the baseline can be witnessed through a reduced divergence rate and a decreased tracking error.

In contrast, controllers trained with D-learning and DOPT exhibit stability i.s.L. in this even more challenging task.

TABLE I

TRAINING EFFICIENCY AND DATA UTILIZATION OF D-LEARNING AND DOPT

Approaches	Environment	Data volume (sample/training)	Training time	Iterations	Final PII	Indicator
D-learning	inverted pendulum	2000/2000	31min46s	40	-610.296	144 (convergence steps)
DOPT	inverted pendulum	1000/2000	28min21s	40	-624.529	136
DOPT	inverted pendulum	2000/4000	34min29s	40	-669.143	122
D-learning	single-track car	2000/2000	63min47.9s	20	-98.4	0.152(steady-state error)
DOPT	single-track car	1000/2000	52min1.6s	20	-110.5	0.124
DOPT	single-track car	2000/4000	69min48.9s	20	-113.7	0.085

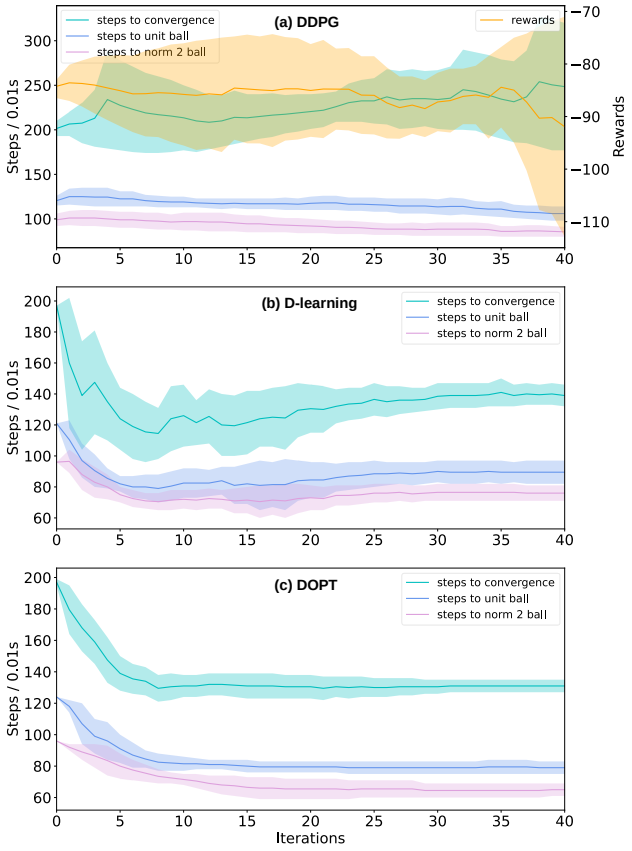

Fig. 5. The statistical training processes for the inverted pendulum controller involve the use of DDPG, D-learning, and DOPT. The steps of three degrees of convergence are recorded. In the legend, from top to bottom, they are steps to equilibrium, steps to $\|x\|_2 = 1$, and steps to $\|x\|_2 = 2$, respectively. In DDPG (a), the accumulated reward is also recorded.

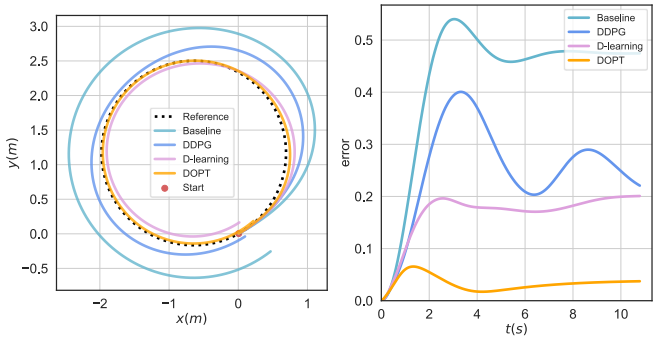

Fig. 6. Single-track car tracking tests in unseen trajectory parameters (road surface with smaller friction) under baseline controller and controllers trained with three algorithms. Results are depicted with trajectory tracking performance (a) and tracking error (b).

Tracking errors of both are confined to an acceptable range, as shown in Fig. 6b. Although some tracking errors remain, the DOPT-trained controller demonstrates greater predictability and minimal steady-state error. The reason for the existence of steady-state error is that adopted NN controllers are essentially static state-feedback controllers, dynamic controllers are needed to achieve zero steady-state error tracking. These results also suggest that the RoA or the forward invariant set of the closed-loop system can be expanded through the DOPT method.

C. Sample Efficiency

Ablation studies are conducted on two systems to evaluate the sample efficiency from the training details shown in Tbl. I. In this online implementation, sample efficiency is reflected in the training efficiency since data sampling is relatively time-consuming. D-learning utilizes the entire sampled data for training, whereas DOPT uses data from a replay buffer that has twice the capacity of the sampled data. For each trial, metrics including training time, final PII, and convergence indicators are documented.

Ablation experiments reveal that the DOPT demonstrates superior sample efficiency and achieves better training results under limited and equal sample volumes. With the same amount of training data, DOPT requires only half the amount of sampled data of D-learning, demonstrating a reduced training time and accelerated convergence controller. Furthermore, when the volume of sampled data is equivalent, DOPT utilizes historical data to more accurately delineate the Lyapunov landscapes, as shown in Fig. 3b and 3d, providing a more valuable optimization reference, and thus yielding faster converging and less steady-state error controllers, larger estimated RoAs, albeit requiring a longer training time. Fig. 5b and 5c depict the training processes of D-learning and DOPT, manifesting that the evolution of the controller in DOPT is more steady and predictable.

V. CONCLUSION

We introduce DOPT, an off-policy variant of D-learning, for the training of NN controllers. DOPT has demonstrated improved sample efficiency, steady evolution, and the ability to effectively capture Lyapunov landscapes as stability certificates. The DOPT-trained controller shows better dynamic and static performance. Future research will integrate more sophisticated NN architectures, such as RNNs or LSTMs, as dynamic controllers within the training framework and explore their application in safety-critical scenarios.

REFERENCES

[1] 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 (CoRL). PMLR. 2018, pp. 466–476.

[2] Shaoru Chen et al. "Learning region of attraction for nonlinear systems". In: 60th IEEE Conference on Decision and Control (CDC). IEEE. 2021, pp. 6477– 6484.

[3] Alessandro Abate et al. "Formal synthesis of Lyapunov neural networks". In: IEEE Control Systems Letters 5.3 (2021), pp. 773–778.

[4] Jun Liu et al. "Towards learning and verifying maximal neural Lyapunov functions". In: 62nd IEEE Conference on Decision and Control (CDC). IEEE. 2023, pp. 8012–8019.

[5] Ya–Chien Chang, Nima Roohi, and Sicun Gao. "Neural Lyapunov control". In: Advances in Neural Information Processing Systems (NeurIPS). Vol. 32. 2019.

[6] Hongkai Dai et al. "Lyapunov-stable neural-network control". In: Robotics: Science and Systems (RSS) XVII. 2021.

[7] Charles Dawson, Sicun Gao, and Chuchu Fan. "Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control". In: IEEE Transactions on Robotics 39.3 (2023), pp. 1749–1767.

[8] Lujie Yang et al. "Lyapunov-stable neural control for state and output feedback: A novel formulation for efficient synthesis and verification". In: arXiv preprint arXiv:2404.07956 (2024).

[9] Lukas Brunke et al. "Safe learning in robotics: From learning-based control to safe reinforcement learning". In: Annual Review of Control, Robotics, and Autonomous Systems 5.1 (2022), pp. 411–444.

[10] Minghao Han et al. "Actor-critic reinforcement learning for control with stability guarantee". In: IEEE Robotics and Automation Letters 5.4 (2020), pp. 6217–6224.

[11] Desong Du et al. "Reinforcement learning for safe robot control using control Lyapunov barrier functions". In: IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2023, pp. 9442–9448.

[12] Yinlam Chow et al. "A Lyapunov-based approach to safe reinforcement learning". In: Advances in Neural Information Processing Systems (NeurIPS). Vol. 31. 2018.

[13] Yinlam Chow et al. "Lyapunov-based safe policy optimization for continuous control". In: arXiv preprint arXiv:1901.10031 (2019).

[14] Milan Ganai et al. "Learning stabilization control from observations by learning Lyapunov-like proxy models". In: IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2023, pp. 2913–2920.

[15] Ya-Chien Chang and Sicun Gao. "Stabilizing neural control using self-learned almost Lyapunov critics". In: IEEE International Conference on Robotics and Automation (ICRA). 2021, pp. 1803–1809.

[16] Ewerton R Vieira et al. "Data-efficient characterization of the global dynamics of robot controllers with confidence guarantees". In: IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2023, pp. 3065–3072.

[17] Alexandre Coulombe and Hsiu-Chin Lin. "Generating stable and collision-free policies through Lyapunov function learning". In: IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2023, pp. 3037–3043.

[18] Quan Quan, Kai–Yuan Cai, and Chenyu Wang. "Control with patterns: a D-learning method". In: Conference on Robot Learning (CoRL). PMLR. 2024.

[19] hristopher JCH Watkins and Peter Dayan. "Q-learning". In: Machine Learning 8 (1992), pp. 279–292.

[20] Shenyu Liu, Daniel Liberzon, and Vadim Zharnitsky. "Almost Lyapunov functions for nonlinear systems". In: Automatica 113 (2020), p. 108758.

[21] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: arXiv preprint arXiv:1509.02971 (2015).

[22] Tuomas Haarnoja et al. "Soft actor-critic algorithms and applications". In: arXiv preprint arXiv:1812.05905 (2018).

[23] Frank L Lewis and Draguna Vrabie. "Reinforcement learning and adaptive dynamic programming for feedback control". In: IEEE Circuits and Systems Magazine 9.3 (2009), pp. 32–50.

[24] Charles Dawson et al. "Safe nonlinear control using robust neural lyapunov-barrier functions". In: Conference on Robot Learning (CoRL). PMLR. 2022, pp. 1724–1735.

[25] Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. Optimal Control. John Wiley & Sons, 2012.

[26] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. "CommonRoad: Composable benchmarks for motion planning on roads". In: IEEE Intelligent Vehicles Symposium (IV). IEEE. 2017, pp. 719–726.