Design-while-Verify: Correct-by-Construction Control Learning with Verification in the Loop

ABSTRACT

In the current control design of safety-critical cyber-physical systems, formal verification techniques are typically applied *after* the controller is designed to evaluate whether the required properties (e.g., safety) are satisfied. However, due to the increasing system complexity and the fundamental hardness of designing a controller with formal guarantees, such an open-loop process of *design-then-verify* often results in many iterations and fails to provide the necessary guarantees. In this paper, we propose a correct-by-construction control learning framework that integrates the verification into the control design process in a closed-loop manner, i.e., *design-while-verify*. Specifically, we leverage the verification results (computed reachable set of the system state) to construct feedback metrics for control learning, which measure how likely the current design of control parameters can meet the required reach-avoid property for safety and goal-reaching. We formulate an optimization problem based on such metrics for tuning the controller parameters, and develop an approximated gradient descent algorithm with a difference method to solve the optimization problem and learn the controller. The learned controller is formally guaranteed to meet the required reach-avoid property. By treating verifiability as a first-class objective and effectively leveraging the verification results during the control learning process, our approach can significantly improve the chance of finding a control design with formal property guarantees. This is demonstrated via a set of experiments on both linear and non-linear systems that use model-based or neural network based controllers.

1 INTRODUCTION

Safety-critical cyber-physical systems(CPSs), such as avionics systems and self-driving vehicles often operate in highly dynamic environments with significant uncertainties and disturbances. It is critical yet challenging to formally ensure their safety, especially for the control and decision making modules. Thus, while there has been increasing interest in applying machine learning techniques (e.g., reinforcement learning (RL) [17] with neural networks) to control and general decision making, their adoption in safety-critical systems is hindered by the challenges in formally ensuring system properties [15].

In this work, we address system safety and goal-reaching ability in control design with a *reach-avoid property* [8], which intuitively represents whether the system can "reach its goal without entering unsafe states" (formal definition in Section 2). It is a fundamentally hard problem to design a controller with formal guarantees for such property. Even in linear systems, the similar "hyper-plane hitting problem" is proved to be NP-hard and it is unclear whether the problem is decidable or not [3]. The complexity continues to increase for non-linear and hybrid systems [11]. Moreover, for emerging neural network-based controllers, synthesizing them with formal guarantees is extremely challenging. A few recent works intended to address it but came with strong limitations, such as only applying to discrete control input [12], or ReLU activation functions [19].

The common process for controller design and verification follows an open-loop *design-then-verify* pattern. The designers first design a controller using either model-based methods such as linear quadratic regulator [2], or model-free approaches such as RL with neural networks [17]. Formal verification tools [4, 13, 14, 16] are then leveraged to evaluate whether the designed controller satisfies the required properties. However, due to the above-mentioned difficulty in designing a controller with formal guarantees, such process might result in many iterations between design and verification, with numerous attempts in tuning the control design and parameter settings, and may still fail to provide the necessary guarantees. For neural network-based controllers, this could be even more challenging, as tuning the design and learning parameters often has an unpredictable impact on the control property [10].

In this work, to address the above challenges, we propose an offline (i.e., design-time; not runtime) correct-by-construction control learning framework that integrates verification in a closed-loop manner, i.e., *design-while-verify*, to formally guarantees that the learned controller satisfies the required reach-avoid property. In our framework, we leverage the verification results, particularly the computed reachable set of the system state, to construct two different types of feedback metrics (based on geometric and Wasserstein distances) that reflect the system's potential ability to meet the reach-avoid property. We then formulate the control learning as an optimization problem of the control parameters based on either metric, and develop an approximated gradient descent algorithm with a difference method for tuning the control parameters until a feasible solution is obtained or iteration limit is reached. Our approach can be applied to *both model-based controllers and neural network based ones*, and formally guarantees that the learned controller can meet the required reach-avoid property.

Related work: Our work is related to the safety verification of controlled dynamical systems [4, 7, 13, 14, 16], which typically relies on the computation of the system reachable set containing all possible states that the system may visit within a time horizon. Our approach leverages these verification tools (called *verifiers* in the rest of the paper), and develops novel metrics and method to integrate them into the control design process. Falsification is another technique that can be leveraged for closed-loop controller design [6]. However, the falsification-driven process does not provide formal guarantees as in our approach.

Our work is also related to control synthesis with reach-avoid guarantees for model-based controllers [5, 8] or neural network-based controllers [12, 19]. However, these recent works are still limited to either linear systems, specific activation functions such as ReLU, or discrete control input; while our framework can address linear and non-linear systems, all types of activation functions and their mixture, and continuous state feedback controllers, as long as their reachable sets can be computed.

In summary, our work makes the following contributions:

- We propose an offline correct-by-construction control learning framework that integrates verification in a closed-loop *design-while-verify* manner, which formally ensures that with the learned controller, the system satisfies the reach-avoid property for safety and goal-reaching.
- Our framework includes novel formulation of the verification-in-the-loop control learning problem based on two different metrics (using geometric or Wasserstein distance) and an approximate gradient descent algorithm with a difference method for solving the problem.
- Our approach can be applied to both linear and non-linear systems under traditional model-based or emerging neural network based controllers. Experiments on a linear adaptive cruise control system, a non-linear oscillator and another 3D numerical system demonstrate that our approach significantly outperforms the baseline methods in convergence rate, safe control rate, goal-reaching rate, and ability to provide formal guarantees.

The paper is organized as follows. Section 2 presents the system model. Section 3 introduces our verification-in-the-loop control learning framework, with analysis on its optimality, soundness, and incompleteness. Experiments and conclusion are presented in Sections 4 and 5, respectively.

2 SYSTEM MODEL

System Dynamics and Controller: We consider a continuous system that can be expressed as a tuple $(X, U, f, \kappa_\theta, X_0, \delta)$. Specifically, the system dynamics is modeled as

$$\dot{x} = f(x, u), \tag{1}$$

where $x \in X \subseteq \mathbb{R}^n$ is the system state vector with X as the state space. $u \in U \subseteq \mathbb{R}^m$ is the control input variable with U as the control input space. $f : X \times U \longrightarrow X$ is a locally Lipschitz-continuous function that can be either linear or non-linear, ensuring there exists a unique solution $x(t)$ to ODE (1). X_0 is a set containing all possible initial states $x(0)$.

Such a system can be controlled by a feedback controller $\kappa_\theta : X \rightarrow U$, parameterized by θ in the following way. Given a sampling period δ, the controller κ reads the system state $x(i\delta)$ at time $t = i\delta (i = 1, 2, \cdots)$, and computes the control input as $u(i\delta) = \kappa_\theta(x(i\delta))$. Then, the system state evolves as $\dot{x} = f(x, u(i\delta))$ within the time slot $[i\delta, (i+1)\delta]$.

Remark 1. *Our approach can deal with a variety of controller types, such as linear controllers, and fully-connected feed-forward neural network controllers that are expressed as*

$$\kappa_\theta(x) = \kappa_L(\kappa_{L-1} \cdots (\kappa_1(x; W_1, b_1); W_{L-1}, b_{L-1}); W_L, b_L),$$

where $\theta = \{W_i, b_i (i = 1, \cdots, L)\}$, W_i, b_i, κ_i are the weights, bias parameters and non-linear activation function for layer i and L is the number of layers.

Flow and Reach-avoid Property: A *flow* function $\varphi(x(0), t) : X_0 \times \mathbb{R}_+ \rightarrow X$ maps some initial state $x(0)$ to the system state $\varphi(x(0), t)$ at time t. Mathematically, φ satisfies 1) $\varphi(x(0), 0) = x(0)$ 2) φ is the solution of the $\dot{x} = f(x, u(i\delta))$ in the time interval $t \in [i\delta, i\delta + \delta]$ 3) $u(i\delta) = \kappa_\theta(\varphi(x(0), i\delta)), \forall i = 1, 2, \cdots$ Based on the *flow* definition, the system reach-avoid property is defined as follows.

Definition 1. ***(Reach-avoid property)*** *Starting from $x(0)$, the system is considered to be reach-avoid if and only if its flow $\varphi(x(0), t)$ 1) never enters into an unsafe set X_u(safety) and 2) reaches a goal set X_g(goal-reaching) within a finite time horizon T.*

$$\begin{cases} \forall\, T \geq t \geq 0,\ \varphi(x(0), t) \cap X_u = \emptyset (safety) \\ \exists\, 0 \leq t' \leq T,\ \varphi(x(0), t') \cap X_g \neq \emptyset (goal - reaching) \end{cases}$$

Verifier and Control Learning: We consider a verifier as a formal tool $\Psi(f, X_0, \kappa_\theta)$ that takes input of system dynamics f, initial state set X_0, and controller κ_θ, and outputs the feedback concerning reach-avoid property (reachable set in this paper). Leveraging such verifier, we define the closed-loop control learning problem with reach-avoid guarantee as follows.

Problem 1. ***(Offline verification-in-the-loop control learning)*** *Given a continuous control system described as Eq (1), find a feasible solution of controller parameters θ and initial region $X_I \subseteq X_0$ with the reachable set computed from verifier $\Psi(f, X_0, \kappa_\theta)$, such that the reach-avoid property is satisfied for every possible $x(0) \in X_I \subseteq X_0$ with controller κ_θ.*

3 VERIFICATION-IN-THE-LOOP CONTROL LEARNING

Our verification-in-the-loop approach leverages the feedback from the verifier to guide the control learning process. It includes the following major components: the computation of the system state reachable set from the verifier (Section 3.1); the two different definitions of a distance metric over the reachable set for evaluating the current control design and the formulation of an optimization problem for control learning (Section 3.2); and an approximated gradient descent algorithm for solving the optimization problem, including the computation of an initial state set for ensuring goal-reaching (Section 3.3). The optimality, soundness, and incompleteness of our approach are also analyzed (Section 3.4).

3.1 Verifier Reachable Set Computation

During the verification-in-the-loop control learning process, the verifier $\Psi(f, X_0, \kappa_\theta)$ computes a reachable set of the system state based on the current controller design κ_θ, defined as:

Definition 2. *A state x_r of system $(X, U, f, \kappa_\theta, X_0, \delta)$ is called reachable at time $t \geq 0$, if and only if there $\exists\, x(0) \in X_0$ such that $x_r = \varphi(x(0), t)$ under controller κ_θ. The reachable set X_r^T with time horizon T for initial set X_0 is defined as*

$$X_r^T = \{\varphi(x(0), t) \mid \forall\, x(0) \in X_0,\ \forall\, 0 \leq t \leq T\}$$

For computing this reachable set, we consider two cases: linear systems under linear controllers, and non-linear systems under non-linear controllers such as neural network based ones.

Linear System with Linear Controller: For a linear time-invariant (LTI) system as $\dot{x} = Ax + Bu$, its reachable set under a linear controller within a finite time interval can be evaluated recursively. Specifically, we consider its discretized LTI system as $x[t+1] = A_d x[t] + B_d u[t]$ with a linear feedback controller $u[t] = \theta^T x[t]$, where $A_d = e^{A\delta}$, $B_d = \int_0^{\delta} e^{At} B dt$ with sampling period δ. Note that for continuous LTI systems, as long as the controller is periodically updated and zero-order hold is applied in each period, it can always be discretized. The initial set X_0 is considered as a polyhedron. In this case, the reachable set of each time step t, denoted as $X_r[t]$, is also a polyhedron, and can be derived recursively from X_0 by polyhedron operation $X_r[t+1] = (A_d + B_d\theta)X_r[t]$ with $X_r[0] = X_0$. The overall reachable set can be obtained as $X_r^T = \bigcup_{t=0}^{T} X_r[t]$. It can also be computed by verification tools, such as Flow* [4] used in the experiment.

Non-linear System with Neural Network Controller : Due to the black-box nature of the neural network, many previous works apply the overly function approximator(e.g. polynomials) to the neural network controller and then compute the over approximation of reachable set for the transformed function. Typically, to ensure the soundness, the output range of the neural network controller under some reachable set at time $t (t \geq 0)$ is bounded as $u = \kappa_\theta(x) \in G_{\kappa_\theta}(x) + [-\epsilon(x), \epsilon(x)], \forall\, x \in X_r^t$, where $G_{\kappa_\theta}(x)$ and $\epsilon(x)$ are the function approximator and remainder of the neural network controller κ_θ within space X_r^t, respectively. Formal verification tools then iterative compute the reachable set of $G_{\kappa_\theta}(x) + [-\epsilon(x), \epsilon(x)]$, as an over approximation for the reachable set of the neural network controller.

In the experiment part, we tried with ReachNN [14] and POLAR [13]. ReachNN leverages the Bernstein polynomials as the function approximator and estimates the approximation remainder by a novel sampling method. POLAR utilizes the Taylor model to approximate the neural network and tighten the approximation by a symbolic remainder.

3.2 Distance Metric Definitions over Reachable Set and Control Learning Formulation

We define two different types of metrics for evaluating the current control design based on the computed reachable set from the verifier, one based on the intuitive geometric distance and one on the Wasserstein distance for its convexity.

Geometric Distance based Metrics: We define a geometric distance d_θ^u that measures the distance between the reachable set X_r^T and the unsafe region X_u as

$$d_\theta^u = \begin{cases} -|X_r^T \cap X_u|, if\ X_r^T \cap X_u \neq \emptyset \\ \inf(||x_r - x_u||^2), \forall x_r \in X_r^T, \forall x_u \in X_u, Otherwise \end{cases} \tag{2}$$

where $|\cdot|$ measures the size of a set. For instance in Fig. 1 with a 2-dimensional system, $|X_r^T \cap X_u|$ is the intersection area between blue and red regions. Intuitively, the system is safe within time horizon T if and only if d_θ^u is positive. Moreover, the larger the d_θ^u is, the further the system stays away from the unsafe region.

Following the same idea, we define another geometric distance d_θ^g for the goal-reaching property as

$$d_\theta^g = \begin{cases} \left|X_r^T \cap X_g\right|, if\ X_r^T \cap X_g \neq \emptyset \\ -\inf(||x_r - x_g||^2),\ \forall x_r \in X_r^T, \forall x_g \in X_g, Otherwise \end{cases} \tag{3}$$

The system satisfies the goal-reaching property if and only if d_θ^g is positive. Similarly in Fig. 1, the larger the d_θ^g is, the better it is for the goal-reaching property. To have formal guarantee on goal-reaching, a searching algorithm for the initial set $X_I \subseteq X_0$ is proposed and detailed later.

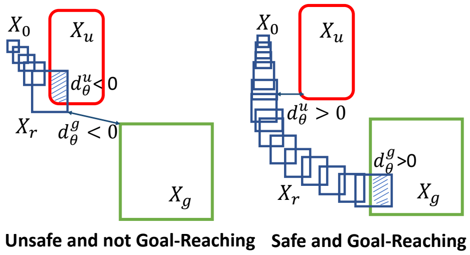

Figure 1: Geometric distances d_θ^u, d_θ^g for safe and goal-reaching properties defined on reachable set (blue) with respect to unsafe region (red) and goal set (green).

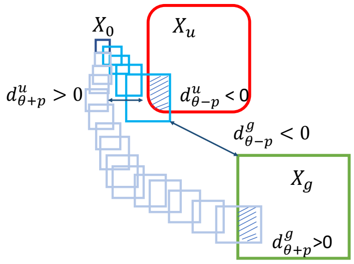

Figure 2: Approximate the gradient for tuning controller parameters by the difference method with perturbation p.

Based on these two metrics, an optimization problem of controller parameters θ for the control learning with reach-avoid property can be formulated as

$$\begin{cases} \max_\theta\ d_\theta^u + d_\theta^g, \\ s.t.\ d_\theta^u \geq 0, d_\theta^g \geq 0. \end{cases}$$

Overall, a feasible solution θ of this problem should make both d_θ^g and d_θ^u positive, which indicates the reach-avoid property is formally assured.

Wasserstein Distance based Metric: Wasserstein distance is defined on two distributions $z(x)$ and $v(y)$ as

$$W(z, v) = \inf_{\gamma \in \Gamma(x,y)} \int d(x, y) d\gamma(x, y), \tag{4}$$

where Γ denotes the collections of all joint distributions with margins as $z(x)$ and $v(y)$. $d(x, y)$ is a distance measure function over x, y, such as norms.

We view the last step of the reachable set X_r^{Tl} as a uniform distribution $r_\theta(x)$, i.e.,

$$r_\theta(x) = \begin{cases} \frac{1}{|X_r^{Tl}|}, if\ x \in X_r^{Tl}, \\ 0, otherwise \end{cases}$$

The same applies to the goal set X_g as $g(x)$ and the unsafe set X_u as $u(x)$. With this transformation, Wasserstein distance is naturally defined on $r_\theta(x), g(x)$ and $r_\theta(x), u(x)$. In this case, the system is reach-avoid if and only if we can determine that $X_r^T \cap X_g \neq \emptyset$ and $X_r^T \cap X_u = \emptyset$. Therefore, the optimization problem based on the

Wasserstein distance over controller parameters θ is defined as

$$\begin{cases}\min_\theta \mathcal{W}(r_\theta) = W(r_\theta(x), g(x)) - W(r_\theta(x), u(x)) \\ s.t. X_r^T \cap X_g \neq \emptyset, X_r^T \cap X_u = \emptyset\end{cases}$$

3.3 Approximated Gradient Descent Algorithm for Control Learning

Based on the computed reachable set and the defined distance metrics, we develop an approximated gradient descent algorithm for the control learning, shown in Algorithm 1.

Algorithm 1 Verification-in-the-loop Control Learning.

```
Require: A verifier $\Psi(f, X_0, \kappa_\theta)$ that computes the reachable set
    $X_r^T$, system dynamics $f$, initial set $X_0$, controller $\kappa_\theta$.
Randomly initialize $\theta$; set the maximum number of updates $N$,
control horizon $T$, step lengths $\alpha, \beta$ and $i = 0$.
while $i \leq N$ and $X_r^T$ is not reach-avoid do
    Generate perturbation $p$ and compute the reachable sets for
    each perturbation as $\Psi(f, X_0, \kappa_{\theta-p})$ and $\Psi(f, X_0, \kappa_{\theta+p})$.
    Compute geometric distances $\left[d^u_{\theta-p}, d^u_{\theta+p}, d^g_{\theta-p}, d^g_{\theta+p}\right]$
    with Eq (2) and (3) or Wasserstein distances
    $\left[W(r_{\theta+p}, g), W(r_{\theta-p}, g), W(r_{\theta+p}, u), W(r_{\theta-p}, u)\right]$ with
    Eq (4).
    Approximate the gradients $\nabla^u_\theta$ and $\nabla^g_\theta$ by Equation (5).
    $\theta = \theta - \alpha\nabla^u_\theta + \beta\nabla^g_\theta$.
    $i \leftarrow i + 1$.
Search reach-avoid initial set $X_I$ using Algorithm 2.
Return: Learned controller $\kappa_\theta$ and $X_I$.
```

Because the verifier is often complex and does not have an analytical form, we propose a difference method to approximate the gradients for safety and goal-reaching metrics, as shown in Fig. 2. For each update iteration, we generate some perturbations p to the controller θ, and then compute their reachable set and also corresponding metrics. Thus, for the geometric and Wasserstein metrics, the gradients can be approximated respectively as

$$\begin{cases}\nabla^u_\theta \approx \frac{d^u_{\theta+p} - d^u_{\theta-p}}{2p}, \nabla^g_\theta \approx \frac{d^g_{\theta+p} - d^g_{\theta-p}}{2p}, \\ \nabla^u_\theta \approx \frac{W(r_{\theta+p}, u) - W(r_{\theta-p}, u)}{2p}, \nabla^g_\theta \approx \frac{W(r_{\theta+p}, g) - W(r_{\theta-p}, g)}{2p},\end{cases} \tag{5}$$

and thus the controller parameters are updated accordingly. Note that if the reach-avoid property is true for some initial space $X_I \subseteq X_0$, we can directly break from the iteration and return the learned controller. Finally, we search for X_I to complete the algorithm.

Reach-avoid Initial Set Searching: Once Algorithm 1 successfully learns a controller, safety can be ensured to the entire initial set X_0. However, goal-reaching is not guaranteed for X_0 because of the intersection operator we used in the metrics and also due to the over-approximation computation of reachable set. Thus, we further propose the searching Algorithm 2 to obtain the reach-avoid initial set $X_I \subseteq X_0$ such that $\forall x(0) \in X_I$, the reach-avoid property is formally verified to hold. Specially, we partition the initial space X_I to many X_p and compute each reachable set $\Psi(f, X_p, \kappa_\theta)$. If there exist some time $t > 0$, such that X_p's reachable set at time t, $\Psi(f, X_p, \kappa_\theta)|_t \subseteq X_g$, then goal-reaching is formally satisfied for X_p under the learned controller κ_θ. A collection of X_p builds up X_I.

Algorithm 2 X_I Searching.

```
Require: Verifier $\Psi(f, X_0, \kappa_\theta)$, system dynamics $f$, initial set $X_0$,
    learned controller $\kappa_\theta$, $X_I = \emptyset$, $P = 1$.
while $X_I$ not converged do
    Evenly partition $X_0$ into sub-spaces $X_p (p = 1, \cdots, P)$.
    for $p \in (1, \cdots, P)$ do
        if $\exists t > 0, \Psi(f, X_p, \kappa_\theta)|_t \subseteq X_g$ then
            $X_I = X_I \cup X_p, X_0 = X_0 - X_p$.
    Increase $P$.
```

3.4 Approach Optimality, Soundness, and Incompleteness

In this section, we introduce the optimality analysis, soundness theorem, and incompleteness clarification of our approach.

Optimality. The Wasserstein distance $\mathcal{W}(r_\theta)$ is believed to be convex and almost everywhere differentiable in the distribution r_θ [1, 18]. Due to its convexity, it holds that

$$\mathcal{W}(r_{\theta_1}) - \mathcal{W}(r_{\theta_2}) \leq \langle \delta\mathcal{W}/\delta r_{\theta_1}, r_{\theta_1} - r_{\theta_2} \rangle, \forall r_{\theta_1}, r_{\theta_2} \in \mathcal{R},$$

where $\langle \cdot, \cdot \rangle$ is the inner-product.

Let $\hat{\theta}$ be an ϵ-stationary point of objective $\mathcal{W}(r_\theta)$. In most cases of the experiments, our approach can reach such stationary points. It then holds with some constant number M that

$$\nabla_\theta \mathcal{W}(r_{\hat{\theta}})^T v \leq \epsilon, \forall v \in \mathcal{B} = \left\{\theta \in \mathbb{R}^n, ||\theta||^2 \leq M\right\}$$

We assume that r_θ is a differentiable function of θ. By the chain rule, it then holds that

$$\nabla_\theta \mathcal{W}(r_{\hat{\theta}})^T v = \left\langle \delta\mathcal{W}/\delta r_{\hat{\theta}}, \left(\frac{dr_\theta}{d\theta}\bigg|_{\theta=\hat{\theta}}\right)^T v \right\rangle \leq \epsilon, \forall v \in \mathcal{B}$$

Let r^* be a global minimizer of $\mathcal{W}(r)$ for $r \in \mathcal{R}$ with corresponding θ^*, we then have

$$\mathcal{W}(r_{\hat{\theta}}) - \mathcal{W}(r^*) \leq \left\langle \delta\mathcal{W}/\delta r_{\hat{\theta}}, r_{\hat{\theta}} - r^* \right\rangle$$

Let $\varphi_{\hat{\theta}} = \frac{dr_\theta}{d\theta}\Big|_{\theta=\hat{\theta}}$ represents the tangent function of r_θ at $\theta = \hat{\theta}$. Then, by combining the above two equations, we have

$$\begin{aligned}\mathcal{W}(r_{\hat{\theta}}) - \mathcal{W}(r^*) &\leq \epsilon + \left\langle \delta\mathcal{W}/\delta r_{\hat{\theta}}, r_{\hat{\theta}} - r^* - \varphi_{\hat{\theta}}^T v \right\rangle, \forall v \in \mathcal{B} \\ &\leq \epsilon + ||\delta\mathcal{W}/\delta r_{\hat{\theta}}|| \cdot ||r_{\hat{\theta}} - r^* - \varphi_{\hat{\theta}}^T v||, \forall v \in \mathcal{B}\end{aligned}$$

Theorem 1. ***(Optimality Analysis)*** *With the designed Wasserstein distance metric, the optimality bound between a stationary point to the global optimum can be obtained as*

$$\mathcal{W}(r_{\hat{\theta}}) - \mathcal{W}(r^*) \leq \epsilon + ||\delta\mathcal{W}/\delta r_{\hat{\theta}}|| \cdot \inf_{v \in \mathcal{B}} ||r_{\hat{\theta}} - r^* - \varphi_{\hat{\theta}}^T v||$$

This shows that on the Wasserstein distance metrics, our approach is highly likely to reach a stationary point that has a bounded distance to the global optimum.

Soundness. Our approach is sound based on its definitions, i.e., the solution found satisfies the reach-avoid property.

Theorem 2. ***(Soundness)*** *Given a system described as Eq* (1), *let κ_θ and X_I be the controller and the initial space obtained by Algorithm 1 and Algorithm 2, respectively. Then the system with κ_θ will reach* *the target set X_g without entering the unsafe set X_u from any initial state within X_I.*

Incompleteness. Our approach is incomplete, i.e., we cannot guarantee to find a solution if it exists. The incompleteness comes from both controller learning and initial set searching procedure. First, in the learning procedure (Algorithm 1), we use an approximate gradient in each update, which does not guarantee to find a feasible controller even if it exists. Second, after obtaining a controller, we apply Algorithm 2 to find a feasible initial set X_I leveraging a reachable set computation tool. Since existing tools for nonlinear systems are all incomplete, a feasible initial set for the controller is not guaranteed to be found even if it exists.

4 EXPERIMENTAL RESULTS

Test Systems: We evaluate our approach by learning linear controllers for a linear adaptive cruise control (ACC) system [20] and neural network controllers for a Van der Pol's oscillator system [20] and 3-dimensional system [16]. The Baseline methods for comparison include model-based stochastic value gradient (SVG) [9] and model-free deep deterministic policy gradient (DDPG) method [17]. SVG and DDPG follow the *design-then-verify* process while our approach is *design-while-verify*. The reward functions in DDPG and SVG are designed to minimize the Euclidean distance to the goal set center and maximize the distance to the unsafe set center. For the activation function, the neural network controllers have ReLU for the hidden layers and Tanh as the output layer.

ACC. There are two robotic vehicles driving on the road, shown in Fig. 3 with the Webots environment [21]. The front vehicle drives at a velocity v_f while the ego vehicle manages the relative distance by accelerating or braking. The dynamics can be expressed as $\dot{s} = v_f - v, \dot{v} = kv + u$, where $v_f = 40$, $k = -0.2$, $\delta = 0.1$ is the sampling period, and (s, v) is the system state. Initial set $X_0 = [122, 124] \times [48, 52]$, the unsafe set $X_u = \{(s, v) | s \leq 120\}$ and the goal set $X_g = [145, 155] \times [39.5, 40.5]$.

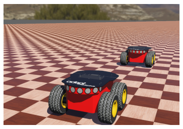

Figure 3: Four-wheeled robot "Pioneer3-AT" in Webots.

Oscillator. Van der Pol's oscillator is a 2D non-linear system, expressed as $\dot{x_1} = x_2, \dot{x_2} = \gamma(1-x_1^2)x_2 - x_1 + u$, sampling period $\delta = 0.1$, and damping coefficient $\gamma = 1$. $X_0 = [-0.51, -0.49] \times [0.49, 0.51]$, $X_g = [-0.05, 0.05] \times [-0.05, 0.05]$, and $X_u = [-0.3, -0.25] \times [0.2, 0.35]$.

3D numerical example. we also test on a 3D system [14], expressed as $\dot{x_1} = x_1^3 - x_2, \dot{x_2} = x_3, \dot{x_3} = u$, sampling period $\delta = 0.2$, $X_0 = [0.38, 0.4] \times [0.45, 0.47] \times [0.25, 0.27]$, $X_g = x_1 \in [-0.5, -0.28] \times x_2 \in [0, 0.28]$, $X_u = x_1 \in [-0.1, 0.2] \times x_2 \in [0.55, 0.6]$.

Verification Tools: We use Flow* [4] for ACC system to learn linear controllers, and use ReachNN [14] and POLAR [13] for oscillator and 3D systems to learn neural network controllers. ReachNN may perform better in low-dimensional systems while POLAR is more efficient and scalable [13].

Performance Comparison: The comparison between different approaches on the three examples is shown in Table 1. With verification in the loop, our approaches with the Wasserstein metric (Ours(W)) and the geometric metric (Ours(G)) take much fewer convergence iterations (CI) than the DDPG and SVG methods. Considering *experimental* safety and goal-reaching properties, we discretized the system and simulated the system traces with 500 randomly picked initial states $x(0)$ from X_0. Our approach achieves both 100% safe control rate (SC) and goal-reaching (GR) rate, while DDPG and SVG cannot. As shown in Figs. 6, 7, and 8, controllers from our approach are *formally verified to satisfy the reach-avoid property*, while DDPG and SVG cannot provide such guarantees in most cases. Learning processes with geometry and Wasserstein distance metrics are shown in Figs. 4 and 5.

	CI	SC	GR	Verified result
ACC, Linear				
SVG	401(±51)	91%	91%	Unsafe
DDPG	13.6(±2.1)K	99.8 %	99.8%	Unknown
Ours(W, Flow*)	64(±31.6)	100%	100%	reach-avoid
Ours(G, Flow*)	62(±6.1)	100%	100 %	reach-avoid
Oscillator, NN				
SVG	388(±15)	98.2%	98.2%	Unsafe
DDPG	13.7(±6.2)K	100 %	79.2 %	Unknown
Ours(W, ReachNN)	9(±2)	100%	100%	reach-avoid
Ours(G, ReachNN)	11(±1)	100 %	100 %	reach-avoid
Ours(W, POLAR)	9(±2)	100%	100%	reach-avoid
Ours(G, POLAR)	12(±1)	100 %	100 %	reach-avoid
3D systems, NN				
SVG	295(±29)	100%	100%	reach-avoid
DDPG	9(±1.8)K	96%	3.6%	Unsafe
Ours(W, ReachNN)	6(±2)	100%	100%	reach-avoid
Ours(G, ReachNN)	7(±2)	100%	100%	reach-avoid
Ours(W, POLAR)	42(±12)	100%	100%	reach-avoid
Ours(G, POLAR)	18(±8)	100%	100%	reach-avoid

Table 1: Comparison of results for all examples on convergence iterations (CI), safe control rate (SC), goal-reaching rate (GR). The controllers synthesized from our verification-in-the-loop approach are guaranteed to be reach-avoid while those from DDPG or SVG are often unsafe or unknown (due to over-approximation of the reachable set computation). With zero-order hold, our controllers achieve 100% safe control rate and goal-reaching rate with the discretized simulation while DDPG and SVG do not. The learning process converges much faster with our approach than DDPG and SVG.

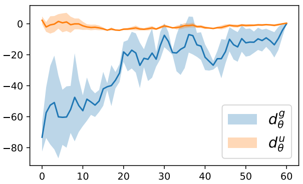

Figure 4: Learning with geometry metric for ACC.

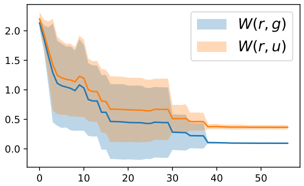

Figure 5: Learning w/ Wasserstein metric for oscillator.

Efficiency Measurement: Table 2 shows the average runtime of each learning iteration in our framework for the three examples

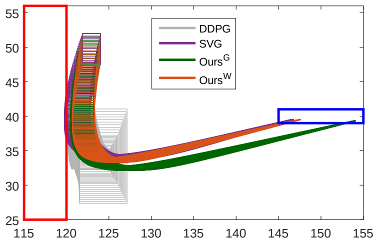

Figure 6: Reachable sets of our approach with two distance metrics and of Baselines for ACC. Goal region is within the blue boundary and the unsafe region is within the red. Our controllers are verified to be reach-avoid with $X_I = X_0$ while DDPG and SVG cannot.

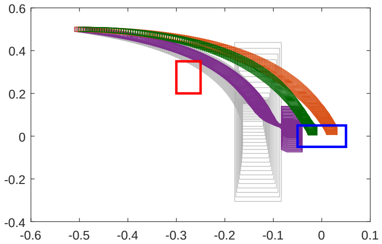

Figure 7: DDPG is verified as unknown due to the over-approximation of reachable set and SVG is unsafe. The learned neural network controllers from our approach are reach-avoid with $X_I = [-0.502, -0.49] \times [0.49, 0.51] (G)$ and $X_I = X_0(W)$.

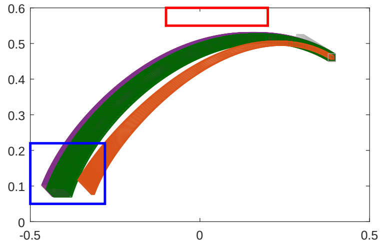

Figure 8: *NAN* occurs for the DDPG controller verification with POLAR after 3 steps while NN controller from our framework formally satisfy the *reach-avoid* property with $X_I = X_0$ for both metrics. SVG is reach-avoid but not guaranteed.

ACC(Flow*)	Os(ReachNN)	Os(POLAR)	3D(ReachNN)	3D(POLAR)
6.05s	516s	72s	195s	23s

Table 2: Average runtime of each iteration for three examples by three verification tools (Os:Oscillator, 3D:3D system).

with three different verification tools, ranging from a few seconds to several minutes. In total, our approaches takes from several minutes to around one hour to learn the controllers for the three examples. In terms of different tools, while utilizing POLAR is much more efficient than ReachNN for neural network controllers.

Discussion on Verification Tightness: The tightness of the over-approximation of the reachable set has a significant impact on our approach. There are adjustable parameters for changing the tightness in verifiers [4, 7, 13, 14, 16]. Intuitively, tighter verification consumes more computation resources and takes more time to finish for each call but may take fewer iterations in our approach. Take the ReachNN as an example, for Wasserstein distance on the oscillator system, the tighter reachable set computation in average takes around 7 steps with near 780 seconds for each step to learn a neural network controller, compared to the lesser tight computation that takes about 9 iterations with 516 seconds for each step.

5 CONCLUSION

In this paper, we propose an offline correct-by-construction control learning framework with reach-avoid guarantees by integrating the verification in a closed-loop manner. Our approach first constructs the control feedback metrics with the reachable sets computed by the verifier, and then iteratively tunes the controller parameters with approximated gradients until a feasible solution is found. Experiments on linear and non-linear systems with linear and neural network controllers demonstrate the effectiveness of our approach on convergence iterations, safe/goal rate, and verified results.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *ICML*. PMLR.

[2] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica* 38, 1 (2002), 3–20.

[3] Vincent D Blondel and John N Tsitsiklis. 2000. A survey of computational complexity results in systems and control. *Automatica* 36 (2000), 1249–1274.

[4] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. 2013. Flow*: An analyzer for non-linear hybrid systems. In *CAV*. Springer, 258–263.

[5] Jerry Ding and Claire J Tomlin. 2010. Robust reach-avoid controller synthesis for switched nonlinear systems. In *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 6481–6486.

[6] Tommaso Dreossi, Daniel J Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A Seshia. 2019. Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *CAV*. Springer, 432–442.

[7] Souradeep Dutta, Xin Chen, and Sriram Sankaranarayanan. 2019. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *HSCC*. 157–168.

[8] Chuchu Fan, Zengyi Qin, Umang Mathur, Qiang Ning, Sayan Mitra, and Mahesh Viswanathan. 2021. Controller synthesis for linear system with reach-avoid specifications. *IEEE Trans. Automat. Control* (2021).

[9] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. 2015. Learning Continuous Control Policies by Stochastic Value Gradients. *NIPs* 28 (2015), 2944–2952.

[10] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *AAAI*.

[11] Thomas A Henzinger, Peter W Kopke, Anuj Puri, and Pravin Varaiya. 1998. What's decidable about hybrid automata? *Journal of computer and system sciences* 57, 1 (1998), 94–124.

[12] Kai-Chieh Hsu, Vicenç Rubies-Royo, Claire J Tomlin, and Jaime F Fisac. 2021. Safety and Liveness Guarantees through Reach-Avoid Reinforcement Learning. In *Robotics: Science and Systems*.

[13] Chao Huang, Jiameng Fan, Xin Chen, Wenchao Li, and Qi Zhu. 2021. POLAR: A Polynomial Arithmetic Framework for Verifying Neural-Network Controlled Systems. *arXiv preprint arXiv:2106.13867* (2021).

[14] Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. 2019. Reachnn: Reachability analysis of neural-network controlled systems. *TECS* 18 (2019).

[15] Radoslav Ivanov, Taylor J Carpenter, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. 2020. Case study: verifying the safety of an autonomous racing car with a neural network controller. In *HSCC*. 1–7.

[16] Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. 2019. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *HSCC*. 169–178.

[17] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning.. In *ICLR (Poster)*.

[18] Gabriel Peyré, Marco Cuturi, et al. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning* 11, 5-6 (2019), 355–607.

[19] Xiaowu Sun and Yasser Shoukry. 2021. Provably Correct Training of Neural Network Controllers Using Reachability Analysis. *arXiv preprint* (2021).

[20] Yixuan Wang, Chao Huang, and Qi Zhu. 2020. Energy-efficient control adaptation with safety guarantees for learning-enabled cyber-physical systems. In *ICCAD*. IEEE, 1–9.

[21] Webots. [n. d.]. http://www.cyberbotics.com. http://www.cyberbotics.com Open-source Mobile Robot Simulation Software.