2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Transfer Learning for Barrier Certificates

Alireza Nadali, Ashutosh Trivedi, and Majid Zamani

Abstract— **A principled approach to safety verification of dynamical systems demands formal guarantees. Barrier certificates are an effective tool for searching safety proofs in the form of inductively verifiable invariants. However, finding barrier certificates is an expensive and time-consuming process that demands human expertise in selecting various templates, hyperparameters, and decision procedures. Is it possible to transfer the knowledge gained in finding a barrier certificate and control algorithm from a given environment (*source environment*) to a different but related environment (*target environment*)? This paper presents a *transfer learning* approach to adapt the barrier certificates (of any template) in the form of neural networks from the source to the target environment. We derive a validity condition to formally guarantee the correctness of network by leveraging its Lipschitz continuity. To demonstrate the effectiveness of our approach, we apply it to two case studies, namely the inverted pendulum, DC motor and Room temperature control. Our results show that transfer learning can successfully adapt barrier certificates from the source to the target environment, reducing the need for human expertise and speeding up the verification process.**

I. Introduction

The importance of safety in hybrid systems has grown significantly due to the rising prevalence of autonomous systems in safety-critical infrastructure such as self-driving cars, implantable medical devices, robotics, and industrial control systems. Barrier certificates [1]–[3] facilitate a deductive approach to safety by enabling an easy-to-verify condition that the system stays clear from unsafe regions. Intuitively, a barrier certificate is a real-valued function over the state space whose level sets can separate the reachable state space from the unsafe space, acting as a "barrier" between two regions. The search for a barrier certificate with a desired template can be automated using powerful decision procedures such as sum-of-squares optimization [4] or satisfiability modulo theory (SMT) solvers [5]. However, this process still requires considerable human insight and remains a computationally and intellectually challenging task that demands expertise in control theory and optimization.

This paper proposes the use of tools from *transfer learning* [6], [7] as a practical and efficient method to lift safety guarantees enabled by barrier certificates from one system to a different but related environment. By leveraging existing barrier certificates and transferring them to new systems, we can reduce the time and compute resources [8] required for synthesizing new certificates. Our approach aims to facilitate the safe deployment of autonomous systems in new environments while reducing the verification burden.

Transfer Learning for Safety. While for traditional software systems it is safe to assume that guarantees established at design time will continue to hold, this assumption does not hold for software controlling hybrid dynamical systems. The dynamics of the environment may evolve over time due to several factors, including mechanical wear and tear, changes in operating conditions (such as ambient temperature and humidity), and changes in requirements. As a result, it is not always possible to rely on barrier certificate-based guarantees established at design time to continue to hold for all future manifestations of the system. However, oftentimes, the changes in system dynamics are not drastic and may not require a costly redesign of the entire system. This underscores the need for efficient and systematic approaches to transfer guarantees from previous versions of the system to adapt to new realities. In this paper, we explore *transfer learning* approaches to leverage existing barrier certificates to provide safety guarantees for new versions of the system without requiring a full-scale verification of the entire system.

Advances in Transfer Learning. Transfer learning is a mature sub-field of AI that utilizes a previously learned knowledge in *source* domain in order to apply it to *target* domain. Transfer learning started with using previously learned weights of a neural network to speed up the training of a new one [9]. Recently, new training methods for neural networks were introduced, including domain adversarial training [10] and adversarial discriminative domain adaptation [11]. These methods provide invariance to small shifts in data distribution by extracting robust features that do not depend on a specific data set. Adversarial learning methods are a promising tool in training robust neural networks, by which they attempt to mitigate the domain shift. Fine-tuning and retraining a network from scratch for similar domains is computationally expensive, or labeled data is sparse or non-existent in target domain [10], [11]. Though transfer learning has achieved astounding results in practice [10], [11], it does not provide formal guarantee.

Neural Network Based Barrier Certificates. Neural networks are universal approximators that can be trained using input-output data to learn to approximate arbitrary Borel-measurable function with arbitrary precision. This has revolutionized the capability of autonomous systems due to their applications in perception, adaptation via reinforcement learning, and control. More recently, neural networks has been exploited to develop data-driven methodology to learn barrier certificates for systems where explicit dynamics in unavailable.

This work was supported in part by the NSF under grants CNS-2145184, CNS-1952223, and A22-0123-S003.

A. Nadali, A. Trivedi and M. Zamani are with the Department of Computer Science, University of Colorado Boulder, Boulder, USA. Email:{a_nadali,ashutosh.trivedi, majid.zamani}@colorado.edu

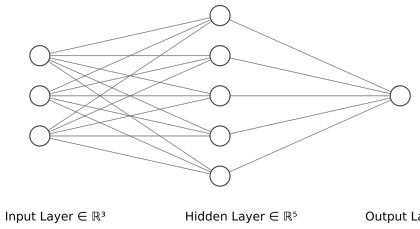

Fig. 1. A simple neural network with 1 hidden layer

They provide formal [12] and statistical [13] guarantees on the barrier certificate based on various assumptions (Lipschitz-continuity) about the underlying system. Recently, new methods have emerged in order to provide formal guarantee for neural networks by utilizing semidefinite constraints, yet these methods are still plagued with high complexity [14]. Our work builds upon and complements these approaches by transferring barrier certificates of any template (neural networks or polynomial functions) as a neural network based barrier certificate [15], [16] and establishes formal guarantees on its properties.

Contributions. We propose a novel transfer learning method by leveraging previously learned knowledge to ensure safety of the system. We provide formal guarantee by deriving a validity condition based upon Lipschitz continuity. In addition, we implement the validity condition in neural network training by enforcing small Lipschitz constant on the neural network. Hence, there is no need for posterior verification. Finally, we illustrate suitability of our method by applying it to two relevant case studies.

II. Preliminaries

Let $\mathbb{R}^n$ be the n-dimensional Euclidean space equipped with infinity norm $\|\cdot\|$. We write $\mathbb{N}$ and $\mathbb{N}_{>0}$ to denote the sets of non-negative and positive integers, respectively. A rectified linear unit (ReLU) is a commonly used activation function $\sigma : \mathbb{R} \to \mathbb{R}$ defined as $\sigma(x) = \max\{x, 0\}$. We can generalize this function from scalars to vectors as $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ in a straightforward fashion by applying ReLU component-wise.

A. Feedforward Neural Networks

A feedforward neural network (henceforth, network) is a weighted directed acyclic graph (see Figure 1) organized in a sequence of layers performing linear mappings followed by non-linear activations. We focus on networks with k fully-connected layers where each layer i is characterized with a weight matrix W_i and a bias vector b_i of appropriate size and is followed by a ReLU.

A *neural network* [17] can be viewed as a function $F : \mathbb{R}^{n_i} \to \mathbb{R}^{n_o}$. Given an input $y_0 \in \mathbb{R}^{n_i}$, a neural network will compute an output $y_k \in \mathbb{R}^{n_o}$ as follows:

$$\begin{aligned} x^{(1)} &= W_1 y_0 + b_1, & y_1 &= \sigma(x^{(1)}) \\ x^{(2)} &= W_2 y_1 + b_2, & y_2 &= \sigma(x^{(2)}) \\ &\vdots \\ x^{(k)} &= W_k y_{k-1} + b_k, & y_k &= \sigma(x^{(k)}) \end{aligned}$$

We call y_{i-1} and y_i the input and output of the i-th layer, respectively, and $x^{(i)}$ the intermediate values at layer i. It is easy to see that neural networks with ReLU activations represent a Lipschitz continuous functions.

Backpropagation [18] is a popular algorithm to learn various parameters of the network $(W_i, b_i)_{i=1}^k$ from input-output data driven by a *loss function* L_F characterizing the closeness of F with the dataset.

B. Safety via Barrier Certificates

We consider discrete-time control systems operating under a given feedback control defined as the following.

Definition 2.1 (Discrete-Time Control System): A discrete-time control system (henceforth, system) $\mathfrak{S}$ is a tuple (X, X_0, U, f) where $X \subseteq \mathbb{R}^n$ is the bounded set of states, $X_0 \subseteq X$ is the set of initial states, $U \subseteq \mathbb{R}^m$ is the set of inputs, and $f : X \times U \to X$ is the state transition function.

The evolution of $\mathfrak{S}$ from an initial state $x_0 \in X_0$ under a feedback control $u : X \to U$ is the infinite state sequence $X_{x_0,u} = \langle x(0), x(1), x(2), \ldots \rangle$ where $x(0) = x_0$ and

$$x(t+1) = f(x(t), u(t)) \text{ for all } t \in \mathbb{N}. \tag{1}$$

We write $\mathfrak{S}_u$ for a system $\mathfrak{S}$ evolving under the controller $u : X \to U$. When clear from context, we write $X_{x_0,u}$ to also denote the set of states in the evolution $X_{x_0,u}$. We define the set of *reachable states* of under u as the set

$$X_u = \bigcup_{x_0 \in X_0} X_{x_0,u}.$$

Given a system $\mathfrak{S}$, a feedback controller $u : X \to U$, and a set of unsafe states $X_\dagger \subseteq X$, we say that system $\mathfrak{S}_u$ is safe if $X_u \cap X_\dagger = \emptyset$. Barrier certificates [1] are real-valued functions over X whose zero-level sets separate reachable and unsafe states and can be used to guarantee safety.

Definition 2.2: Consider a system $\mathfrak{S} = (X, X_0, U, f)$. A function $B : X \to \mathbb{R}$ is a barrier certificate for $\mathfrak{S}$ under a controller $u : X \to U$ against unsafe states $X_\dagger \subseteq X$, if there exists $\eta \leq 0$ such that

$$\begin{aligned} \mathrm{B}(x) &\leq \eta \quad \text{for all } x \in X_0, \\ \mathrm{B}(x) &> -\eta \quad \text{for all } x \in X_\dagger, \text{ and} \\ \mathrm{B}(f(x, u(x))) - \mathrm{B}(x) &\leq \eta \quad \text{for all } x \in X. \end{aligned} \tag{2}$$

The following lemma characterizes the correctness of the barrier certificates.

Lemma 1: Consider a system $\mathfrak{S} = (X, X_0, U, f)$. The existence of a barrier certificate $B : X \to \mathbb{R}$ satisfying (2) for $\mathfrak{S}$ under a controller $u : X \to U$ against unsafe states $X_\dagger \subseteq X$ implies that $\mathfrak{S}_u$ is safe.

Proof: Note that the zero sublevel set of the barrier certificate $B_{\leq 0} = \{x : B(x) \leq 0\}$ characterizes an inductive invariant over the reachable state space. It is easy to verify that every evolution $\langle x(0), x(1), x(2), \ldots \rangle$ remains in $B_{\leq 0}$:

- the initial states $x(0) \in B_{\leq 0}$ as $B(x(0)) \leq \eta$, and
- for every t assuming $x(t) \in B_{\leq 0}$ implies that $x(t+1) \in B_{\leq 0}$ as $\mathrm{B}(f(x, u(x))) - \mathrm{B}(x) \leq \eta$.

Finally noting that no state in $X_\dagger$ is in $B_{\leq 0}$, as $\mathrm{B}(x) > -\eta$ for all $x \in X_\dagger$, completes the proof. ∎

Hence, in order to provide formal guarantee for safety of the system $\mathfrak{S}$, it is sufficient to find a controller and the corresponding barrier certificate. Given a controller, one restricts the search for a barrier certificate within a given template (e.g., polynomial functions of a fixed degree) and employs decision procedures such as sum-of-squares programming [19] or satisfiability-modulo-theory (SMT) solver [20]. In practice, one can combine combine the search for control and barrier certificates [2], [3].

Despite over twenty years of research in synthesizing barrier certificates, the search for barrier certificates require human insight in selecting the template, decision procedures, and requires considerable computational resources in establishing the barrier certificate conditions. More often than not, fixing the barrier certificate's template is too restrictive and one may not be able to find a barrier certificate. To tackle both challenges, feedforward neural networks have been proposed [15], [16] to represent barrier certificates.

III. Transfer Learning for Barrier Certificates

We are interested in the following problem of transferring barrier certificates from the source system to the target.

Definition 3.1 (Barrier Transfer): Consider two different but related systems: the *source* $\mathcal{S} = (X, X_0, U, f)$ and the *target* $\mathcal{T} = (X, X_0, U, \widehat{f})$. Assume that the control law $u : X \to \mathbb{R}$ and barrier certificate $B : X \to \mathbb{R}$ for $\mathcal{S}$ are given. The *barrier transfer problem* is to find a barrier certificate $\widehat{B} : X \to \mathbb{R}$ demonstrating the safety of $\mathcal{T}$ under u.

If the system $\mathcal{T}$ is close to the system $\mathcal{S}$, we should be able to adapt the previously acquired knowledge in B to learn the barrier certificate $\widehat{B}$. This is the central thesis behind transfer learning [21]. We develop a solution to the barrier transfer problem by designing a novel architecture combining the ideas from *generative adversarial networks* (GANs) [22] and a recent transfer learning approach known as *adversarial discriminative domain adaptation* (ADDA) [11].

The GAN is an adversarial training approach to learn a generative model where two networks (the generator and the discriminator) have interdependent loss functions such that the goal of the generator network is to increasingly learn a more refined concept, while the goal of the discriminator network is to continue to improve in finding errors in generator network. As they are trained iteratively, both networks get increasingly performant in their respective tasks. The ADDA approach, on the other hand, is a transfer learning approach to transfer a classifier from one setting to another. The key idea is based on the common wisdom [23] that the initial layers of a neural network tend to extract features, while the latter layers perform a classification task [24] based on the learned features. In this approach, network is forced to learn robust features that do not depend on a specific domain. Hence, if we can learn the to extract features from the new environment such that a discriminator network cannot distinguish if the features is coming from the source environment or the target, the feature extraction is learned from the target environment in such a manner that is consistent with the source environment.

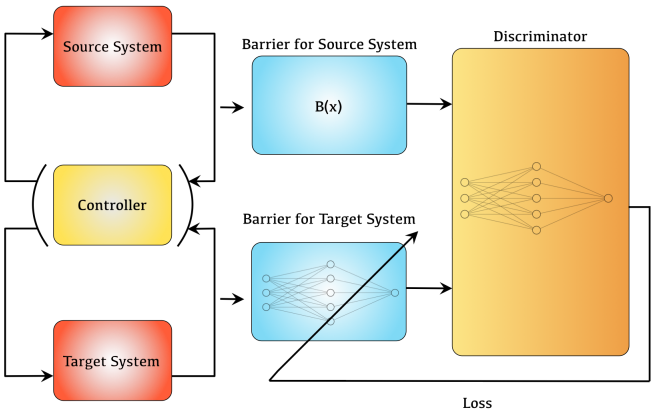

Fig. 2. Transferring formal guarantee. Our proposed method consists of two neural networks, discriminator and generator. Generator is learning how to represent a barrier certificate for target system while discriminator is distinguishing between source and target barriers.

Thus, one can use the classifier network (which maps the features into labels) from source domain in target domain.

Our work combines GAN and ADDA approaches to transfer barrier certificates from the source system to the target. Our architecture is depicted in Figure 2 where the generator network for the barrier certificate (in blue) and the discriminator network (in gold) distinguishing between source and target systems by only observing barrier function values. As the transfer learning approaches do not provide formal guarantees, we enclose our learning approach within a formal framework of data-driven learning of barrier certificates to establish formal guarantees. In particular, we sample the state space by picking data points using a fine grid and exploit Lipschitz continuity argument to establish that a barrier certificate faithful to the grid point remains faithful to all unseen points within the grid.

The rest of the paper is organized a follows. We begin (Section III-A) by talking about of data generation, followed by the validity condition (Section III-B) to ensure the correctness of barrier certificates. Our key contribution of the architecture of our transfer learning is discussed in Section III-C followed by some optimization to the architecture in Section III-D.

A. Data Generation

Our approach to barrier transfer problem is to learn $\widehat{B}$ as a neural network by forming a uniform grid on state space.

$$X_{\text{grid}} = \langle x_1, \ldots, x_N \rangle, \tag{3}$$

where N is the total number of grid points. we form the data sets $(\mathcal{I}, \mathcal{U}, \mathcal{E})$ as

$$\begin{aligned} \mathcal{I} &= X_{\text{grid}} \cap X_0, \\ \mathcal{U} &= X_{\text{grid}} \cap X_\dagger, \text{ and} \\ \mathcal{E} &= \langle (x_i, f(x_i, u(x_i)), \widehat{f}(x_i, u(x_i))) \rangle_{i=1}^{N}. \end{aligned} \tag{4}$$

Using grid points, we construct cover sets consist of hyper-rectangles

$$R_i\left(x_i, \epsilon_i\right) := \{x \in X \mid 0 \leq \|x - x_i\| \leq \epsilon_i\} \tag{5}$$

centered at sampled points (x_i), by defining $\hat{\epsilon} = \max_i ||\epsilon_i||$, then for all $x \in X$, there exists x_i, $i \in \{1, \ldots, N\}$, such that $||x - x_i|| \leq \hat{\epsilon}$. Therefore, data sets $\mathcal{I}$, $\mathcal{U}$, and $\mathcal{E}$ corresponding to X_0, $X_\dagger$ and X are constructed by considering corresponding representative points, which networks is trained on.

B. Validity Condition

When the training is done, generator would be the barrier for target system, thus it needs to satisfy conditions in (2). To ensure safety, we need to provide formal guarantee for the neural network, since the network is trained on finitely many data points. As it was mentioned before, the state set is discretized with N samples. In order to provide formal guarantee for unseen data, we propose Theorem I.

Theorem 1: Consider a system $\mathfrak{S}$ with controller u, initial and unsafe sets X_0 and $X_\dagger$, respectively. Let B be a Lipschitz continuous barrier for the given system $\mathfrak{S}$ that has been acquired using finitely many samples. Furthermore η^* is a value that satisfies (2) for the said data points. If

$$\mathcal{L}\hat{\epsilon} + \eta^* \leq 0, \tag{6}$$

in which $\mathcal{L}$ is the Lipschitz constant of the barrier with respect to x, then B is valid for the entire state set as it has been mentioned in (2).

Proof: Assuming η^* satisfies (2), following conditions hold.

$$\begin{array}{ll} \mathrm{B}\left(x_i\right) \leq \eta^*, & \forall x_i \in \mathcal{I}, \\ \mathrm{B}\left(x_i\right) > -\eta^*, & \forall x_i \in \mathcal{U}, \\ \mathrm{B}\left(f\left(x_i, u\left(x_i\right)\right)\right) - \mathrm{B}\left(x_i\right) \leq \eta^*, & \forall x_i \in \mathcal{E}, \end{array} \tag{7}$$

for all $i \in \{1, \ldots, N\}$. Based on Lipschitz continuity property of the barrier B we have

$$\mathrm{B}(x_j) - \mathrm{B}\left(x_k\right) \leq \mathcal{L}\left\|x_j - x_k\right\|. \tag{8}$$

where $x_i, x_k \in X$. Furthermore, for all $x \in X_0$ there exists x_i such that $||x - x_i|| \leq \hat{\epsilon}$. Incorporating this property, we get

$$\mathrm{B}(x) - \mathrm{B}\left(x_i\right) \leq \mathcal{L}\left\|x - x_i\right\| \leq \mathcal{L}\hat{\epsilon}, \tag{9}$$

where x_i is a center of a hyper-rectangle. Rearranging the equation

$$\mathrm{B}(x) \leq \mathcal{L}\hat{\epsilon} + \mathrm{B}(x_i). \tag{10}$$

According to barrier definition $\mathrm{B}(x_i) \leq \eta^*$ for all $x_i \in X_0$. Thus

$$\mathrm{B}(x) \leq \mathcal{L}\hat{\epsilon} + \eta^*. \tag{11}$$

which is negative according to (6). Similarly, one can show that $\mathrm{B}(x) > 0$ for all $x \in X_\dagger$ and $\mathrm{B}(f(x, u(x)) - \mathrm{B}(x) \leq 0$ for all $x \in X$. Hence, B satisfies (2) for entire state set. ■

C. Transfer Learning Architecture

Our architecture for barrier transfer consists of two independent networks: the *generator* network $G : \mathbb{R}^n \to \mathbb{R}$ and the *discriminator* network $D : \mathbb{R} \to \mathbb{R}$. Both of these networks are fully-connected with ReLU activation units and the number of hidden layers is left as a hyperparameter in our architecture. The generator network is being trained to capture the barrier certificate for target system. On the other hand, the discriminator network takes a single input, which can be from the generator (barrier for target system) or barrier of the source system. As its name suggests, this network tries to distinguish whether the given input is from target or source barrier. Output of this network is a real number corresponding to the probability of belonging to either source or target barrier. This task of distinguishing data is known as supervised learning, where you assign a label to each data instance [24].

In order to train the discriminator, we model the probability using logistic function $v(z) = (1 + e^{-z(l)})^{-1}$ where $z(l)$ is some function of the scaler input l which can be optimized. Since we can classify the inputs with two labels (source and target), we assign $v(l) = 1$ to the source and $v(l) = 0$ to the target data. Assuming $D(l)$ is the output of the discriminator, the probability of l belonging to source and target is $v_{y=1} = \hat{y} = \left(1 + e^{-D(l)}\right)^{-1}$ and $v_{y=0} = 1 - \hat{y}$, respectively. Here y denotes the true label and $\hat{y}$ is the label predicted by the discriminator. Therefore, we employ cross-entropy loss as a measure of dissimilarity between these two

$$H(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}), \tag{12}$$

Note that the loss is zero when $y = \hat{y}$.

Generator is trained on how well it managed to trick the discriminator

$$L_G = H(1, D(G(x))). \tag{13}$$

In addition to (13), we add the Lipschitz constant of the network to its loss to encourage the generator to minimize its Lipschitz constant. Therefore generator's loss is

$$L_G = H(1, D(G(x))) + \mathcal{L}, \tag{14}$$

where $\mathcal{L}$ is Lipschitz constant. Small Lipschitz constant is paramount for our validity condition; which will be discussed later. Similarly, discriminator is trained on how well it manages to distinguish between real barrier and generated barrier output

$$L_D = H(0, D(G(x))) + H(1, D(B(x))), \tag{15}$$

where $\mathrm{B}(x)$ denotes the barrier for source system.

We employ data-driven methods to estimate $\mathcal{L}$. Assume $B : \mathbb{R}^n \to \mathbb{R}$ is a Lipschitz continuous neural network barrier. Firstly, we sample M points in the state space and calculate the absolute value of the slopes

$$s_{ij} = \frac{|\mathrm{B}(x_i) - \mathrm{B}(x_j)|}{|x_i - x_j|} \quad \forall i, j \text{ such that } i \neq j, \tag{16}$$

where x_i and x_j are two sampled points. Then we choose the maximum of s_{ij}. This procedure is repeated $M^{'}$ times and an inverse weibull distribution is fitted to the maximum slopes. The location parameter is an estimate of the Lipschitz constant [25]. One can get a more accurate estimation if $|x_i - x_j| \leq \delta$ for a small $\delta > 0$. Based on proposition 1 in [25], as $M, M^{'} \to \infty$ and for small $\delta > 0$, the estimated Lipschitz constant converges to its true value.

Throughout this paper, we do not consider the error in estimating the Lipschitz constant since we use sufficiently large M and $M^{'}$ and small δ such that the error is negligible.

D. Adaptive Training for Barrier Transfer

In order to facilitate training, we use two different discriminator networks and, consequently, we have two different phases of training. The first phase consists of randomly sampling points from each data set $\mathcal{I}$, $\mathcal{U}$, and $\mathcal{E}$, then generating the corresponding barrier values for each of three conditions. When the generator can satisfy first and second barrier conditions for all points, we move to the second phase.

In the second phase, we directly use the data that are violating the third condition. Using adaptive training, we managed to reduce the number of iteration considerably. Additionally, if most of the points in the data set satisfy the barrier conditions, then the loss would likely be zero or a small value and network will not learn anything. Thus, it is intuitive to use the points that violate barrier conditions to train the network, instead of relying on sampling. Training stops when all three conditions of barrier certificate and $\mathcal{L}\hat{\epsilon} + \eta \leq 0$ for all the data points are satisfied. Algorithm 1 outlines our proposed method.

IV. Case Studies

In this section, we illustrate effectiveness of our method by utilizing two case studies. For all case studies, we trained the neural network on Nvidia RTX 4090 GPU coupled with Intel core I7 13700k CPU. We use Adam optimizer in order to train the neural networks [26].

A. Inverted Pendulum

For the first case study, we consider an inverted pendulum. Dynamics of the system can be described as

$$\mathcal{S}: \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) + \tau\,(x_2(k)) \\ x_2(k) + \tau \frac{g}{l}\left(\sin\left(x_1(k)\right) + \frac{1}{ml^2}u\right) \end{bmatrix},$$

where x_1, x_2 are angular position and velocity, respectively. Constants $m = 1$ and $l = 1$ are the mass and the length of the pendulum, $g = 9.8$ is the gravitational acceleration and sampling time $\tau = 0.01$. The state set is $X = [-\frac{\pi}{4}, \frac{\pi}{4}]^2$, initial state set $X_0 = [-\frac{\pi}{15}, \frac{\pi}{15}]^2$, safe state set $X_s = [-\frac{\pi}{6}, \frac{\pi}{6}]^2$ and unsafe state $X_\dagger = X \setminus X_s$. For the target system, we assume $m = 1.2$ and $l = 1.1$. Additionally, barrier and controller for the source system are obtained using [12], but barrier does not satisfy the third condition in (2) for the target system.

Algorithm 1 Learning barrier

Input: $X_0, X_u, X, \hat{\epsilon}, B_s, u_s$
Output B_t, η

```
Construct data sets I, U, and E from X_0, X_† and X.
Initialize η, generator G_t and discriminators D_1, D_2.
while Conditions not satisfied do
    if G_t(I) ≤ η and G_t(U) > −η then
        ind ← {i | G_t(E_i) − G_t(x_i) > η}
        Update loss L_G(X[ind]) and L_{D_2}(X[ind])
    else
        b_X ← Random batch
        Update loss L_G(b_X) and L_{D_1}(b_X)
    end if
end while
Return B_t, η
```

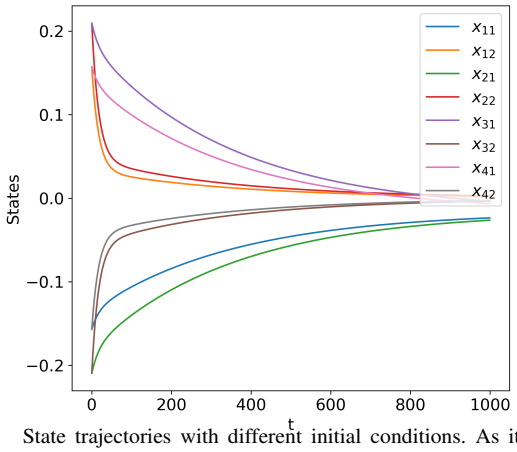

Fig. 3. State trajectories with different initial conditions. As it can be inferred, starting from X_0 system will not reach the unsafe set.

TABLE I
Different Parameters for Target System

m	l	Iterations	Convergence time (min)
1.05	1.05	600	7
1.1	1.1	2458	19
1.2	1.1	3689	25
1.2	1.2	7000	30
1.5	1.5	NA	NA

Our method converged quite fast with roughly 4000 iterations, $\eta^* = -0.002$, $\mathcal{L} = 1.9$ and $\hat{\epsilon} = 0.001$ as final parameters. Four different trajectories of the system are depicted in Figure 3, x_{ij} denotes the jth state of ith trajectory. Table I shows different target systems and number of iterations. The method presented in [12] typically converges in approximately 50 minutes on average, with minor adjustments to the Lipschitz constant estimation. The findings in [12] involve the computation of the global Lipschitz constant, which is computationally intensive. To ensure a fair comparison, we adopted the same sampling technique for Lipschitz constant estimation, resulting in a significant reduction in convergence time.

As we can see, the more parameters change compared to the source system, neural network require more iterations to converge. This trend continues until we reach a point where source and target differ so drastically that neural network cannot converge; meaning one cannot use the same controller for target system.

B. DC Motor

In this case study, we consider a discrete-time DC motor. The dynamics of the system are

$$\mathcal{S}: \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) + \tau\left(\frac{-R}{L}x_1(k) - \frac{K}{L}x_2(k) + \frac{1}{L}u\right) \\ x_2(k) + \tau\left(\frac{K}{J}x_1(k) - \frac{b}{J}x_2(k)\right) \end{bmatrix},$$

where states x_1 and x_2 are the armature current and rotational speed of the shaft, parameters $R = 1$, $L = 0.5$, $J = 0.01$,$\tau = 0.01$ and $b = 0.1$ are the electric resistance, the electric inductance, the moment of inertia of the rotor, sampling time and the friction constant, respectively. In addition, $K = 0.01$ represents both the motor torque and electromotive force constant. For the target system, we consider $R = 1.5$, $L = 0.55$, $J = 0.011$, $b = 0.11$ and $K = 0.011$. Barrier and controller are obtained via [12] for the source system. As before, barrier for the source system violates the third condition in (2) for the target system.

Regions of interest are $X_0 = [-0.005, 0.005] \times [-0.05, 0.05]$, $X_u = [0.5, 0.7] \times [0.06, 0.1]$ and $X = [-0.7, 0.7] \times [-0.1, 0.1]$. The input of the system, i.e. u, is the voltage, and it is within the set $U = [-1, 1]$. Our method converged with only 2000 iterations in 5 minutes with $\eta^* = -0.02$, $\mathcal{L} = 19.2$ and $\hat{\epsilon} = 0.001$ as final parameters. The method proposed in [12] typically converges in an average time of 35 minutes, as previously mentioned, with minor adjustments made to the Lipschitz constant estimation.

Remark 1 (Systems with disturbance): In our simulations we have also considered target system to have the same parameters, but with added and multiplicative disturbance for dc motor and inverted pendulum, respectively. For inverted pendulum, the amplitude of the disturbance was 0.05% of the states, and for the DC Motor, amplitude of added disturbance was 0.1. Needless to say, disturbance caused the barrier for the source system to fail. Our proposed method was able to compute a barrier for the target system while adhering to validity condition in (7). Henceforth, one can also utilize this method to provide formal guarantee for a target system that has been plagued with disturbance.

V. Conclusion

Our paper proposes a novel model-free approach to transferring formal guarantee between two iterations of a system. By utilizing neural networks to represent barrier certificates, we can overcome the problem of unknown model and only require data samples from the given system, without any prior mathematical knowledge. Our approach employs two distinct neural networks: a generator and a discriminator, which work together to, respectively, generate and critique the barrier certificate. Furthermore, we proposed a validity criteria that ensures the correctness of the barrier certificate over the entire state set, rather than just the finite data points used for training. We have demonstrated the effectiveness of our method through two relevant case studies, showing that our approach can significantly improve the transfer of safety proofs between different iterations of a system.

For future work, we intend to provide a formal theory for transferring barrier certificates with clear notion of closeness and convergence guarantees.

References

[1] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control*, (Berlin, Heidelberg), pp. 477–492, Springer Berlin Heidelberg, 2004.

[2] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European control conference (ECC)*, pp. 3420–3431, IEEE, 2019.

[3] P. Jagtap, S. Soudjani, and M. Zamani, "Formal synthesis of stochastic systems via control barrier certificates," *IEEE Transactions on Automatic Control*, vol. 66, no. 7, pp. 3097–3110, 2021.

[4] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical programming*, vol. 96, pp. 293–320, 2003.

[5] L. De Moura and N. Bjørner, "Satisfiability modulo theories: introduction and applications," *Communications of the ACM*, vol. 54, no. 9, pp. 69–77, 2011.

[6] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[7] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, pp. 151–175, Oct. 2009.

[8] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264, IGI global, 2010.

[9] B. Neyshabur, H. Sedghi, and C. Zhang, "What is being transferred in transfer learning?," *Advances in neural information processing systems*, vol. 33, pp. 512–523, 2020.

[10] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[11] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," 2017.

[12] M. Anand and M. Zamani, "Formally verified neural network control barrier certificates for unknown systems," in *22nd IFAC World Congress, to appear*, IFAC, 2023.

[13] R. Mazouz, K. Muvvala, A. Ratheesh Babu, L. Laurenti, and M. Lahijanian, "Safety guarantees for neural network dynamic systems via stochastic barrier functions," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9672–9686, 2022.

[14] P. Pauli, N. Funcke, D. Gramlich, M. A. Msalmi, and F. Allgöwer, "Neural network training under semidefinite constraints," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 2731–2736, IEEE, 2022.

[15] H. Zhao, X. Zeng, T. Chen, and Z. Liu, "Synthesizing barrier certificates using neural networks," in *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, pp. 1–11, 2020.

[16] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods," *arXiv preprint arXiv:2202.11762*, 2022.

[17] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1998.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[19] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical programming*, vol. 96, pp. 293–320, 2003.

[20] L. De Moura and N. Bjørner, "Satisfiability modulo theories: introduction and applications," *Communications of the ACM*, vol. 54, no. 9, pp. 69–77, 2011.

[21] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, pp. 151–175, 2010.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014.

[23] M. Nixon and A. Aguado, *Feature extraction and image processing for computer vision*. Academic press, 2019.

[24] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," *Machine learning techniques for multimedia: case studies on organization and retrieval*, pp. 21–49, 2008.

[25] G. Wood and B. Zhang, "Estimation of the lipschitz constant of a function," *Journal of Global Optimization*, vol. 8, pp. 91–103, 1996.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.