Learning Neural Networks under Input-Output Specifications

Zain ul Abdeen[1], He Yin[2], Vassilis Kekatos[1], Ming Jin[1]

***Abstract*— In this paper, we examine an important problem of learning neural networks that certifiably meet certain specifications on input-output behaviors. Our strategy is to find an inner approximation of the set of admissible policy parameters, which is convex in a transformed space. To this end, we address the key technical challenge of convexifying the verification condition for neural networks, which is derived by abstracting the nonlinear specifications and activation functions with quadratic constraints. In particular, we propose a reparametrization scheme of the original neural network based on loop transformation, which leads to a convex condition that can be computationally enforced during learning. The theoretical construction is validated in an experiment that specifies reachable sets for different regions of inputs.**

I. Introduction

The advances in deep learning (DL) has impacted many areas, most prominently computer vision and natural language processing [1]–[3]. However, the use of DL for safety-critical tasks in the real world is challenged by its opaqueness, fragility, and vulnerability [4]. For example, an imperceptible but carefully engineered perturbation in the input can easily mislead DL systems [5]. Notably, DL models are rarely used in a standalone manner but as part of a larger pipeline. Thus, specifications on the model decisions are required to capture the true constraints on their physical and social ramifications. These specifications include but are not limited to safety [6], stability [7], [8], privacy [9], fairness [10], and interpret-ability [11]. Up till now, verification of NN has been primarily focused on adversarial robustness, and can be divided into exact approaches and inexact approaches: exact approaches calculate the output set without any approximation, whereas inexact methods seek to approximate the output set for computational tractability [12]. Moreover, deriving guarantees for nonlinear, large-scale, complex policies such as NN is a significant technical challenge, and there have been increasing efforts in the realm [14]–[16].

The method that we propose for learning NN under specifications belongs to the large family of convex relaxation techniques. We demonstrate that for a specific class of specification problems, learning can be done by solving a semidefinite program (SDP). In particular, we note that the integral quadratic constraint (IQC) framework [18] has been applied in post-hoc verification of robustness for a fixed NN [17]. We address the main challenge of existing methods—the nonconvexity of the learning condition with respect to the policy parameters—to develop a computationally efficient procedure.

Contribution: We propose a framework to learn a NN that satisfies specifications on input-output behaviors. We overcome a major technical hurdle by deriving a convex condition that can be imposed during the learning process. The key to our method is to compute a convex inner approximation to the nonconvex set of admissible policy parameters. To this end, we characterize the behaviors of the nonlinear activation and input-output specifications by quadratic constraints. For convexification, we design a new reparametrization scheme based on loop transformation [19, Chap. 4] and $\mathcal{S}$-lemma [20], [13]. The one-to-one correspondence between the transformed parameters and the original parameters is guaranteed for a two-layer NN. Hence, we can efficiently learn in the reparameterized space and recover the original parameters, leading to a NN that certifiably satisfy the desired properties.

Related work: IQC-based analysis of NN has been explored in [17] for a verification setting; however, the corresponding condition is nonconvex in the policy parameters, preventing its application in a learning setting. As neural networks become popular in control tasks, safety and robustness of NN controlled systems have been examined in [22]–[25]. It is also possible to concatenate an optimization layer to a DNN to satisfy hard output constraints (with an additional computational cost to solve an optimization problem every time an output is produced) [29]. An approach for NN verification against convex-relaxable specifications is proposed in [21], which shares the line of thinking with the present work to move towards general specifications (beyond adversarial robustness). The present work is inspired by [8], in which the authors proposed a method to synthesize a neural network controller with stability and safety guarantees through imitation learning; a recent extension to policy gradient for reinforcement learning is presented in [22].

The rest of the paper is organized as follows. Section II describes the problem setup. Section III briefly reviews results for verification of a fixed NN. The main method to obtain a convex learning condition is presented in Section IV. Section V validates the approach in a reachability setup for different regions of inputs. Section VI concludes the paper with some future directions.

Notation: We denote $\mathbb{S}^n$, $\mathbb{S}^n_+$, $\mathbb{S}^n_{++}$ as the sets of $n \times n$ symmetric, positive semi-definite and positive definite matrices, respectively. For any matrix $A \in \mathbb{S}^n$, the inequality $A \succeq 0$ and $A \succ 0$ indicates positive semi-definiteness and

[1]The Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA. Emails: {zabdeen, kekatos, jinming}@vt.edu

[2]Department of Mechanical Engineering, University of California, Berkeley, CA, USA. Email: he_yin@berkeley.edu

This work was supported by the U.S. National Science Foundation under grant 2034137.

positive definiteness, respectively.

II. Problem Formulation

A. Problem statement

We consider specifications on outputs in relation to inputs that vary across instances. Formally, we define a multi-layer feed-forward neural network NN mapping $\Psi(\cdot;\theta):\mathcal{X}\to\mathcal{Y}$ parameterized by weight vector $\theta\in\mathbb{R}^{n_\theta}$. Sets $\mathcal{X}\subseteq\mathbb{R}^{n_x}$ and $\mathcal{Y}\subseteq\mathbb{R}^{n_y}$ are respectively the sets wherein NN inputs and outputs can lie. We also define an m-way specification $F:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}^m$, and its associated specification set $\mathcal{S}(x):=\{y\in\mathbb{R}^{n_y}:F(x,y)\leq 0\}$. Our aim is to find a parameter θ such that

$$\Psi(x;\theta)\in\mathcal{S}(x),\quad\forall x\in\mathcal{X}. \tag{1}$$

To simplify notation, we henceforth omit the dependence of Ψ on θ. Note that the admissible set $\Theta:=\{\theta\in\mathbb{R}^{n_\theta}:\text{(1) is satisfied}\}$ is nonconvex in general due to the nonlinearity of Ψ and specifications F. The above formulation can incorporate a family of problems in machine learning and control, such as fairness [10], adversarial robustness [17], [21]; and reachability analysis [17], [26]. Granted that searching within the nonconvex admissible set Θ is intractable; hence, our strategy is to compute an convex inner approximation. To this end, we propose a semidefinite convexification approach to specify the convex set.

B. Isolating NN nonlinearities

The input-output mapping of a feed-forward NN with l layers can be described by the recursive equations:

$$\begin{aligned} x^0&=x\\ x^{k+1}&=\phi^k(W^kx^k+b^k),\qquad k=0,1,\dots,l-1\\ \Psi(x)&=W^lx^l+b^l \end{aligned} \tag{2}$$

where $x\in\mathcal{X}$ is the NN input; $W^k\in\mathbb{R}^{n_{k+1}\times n_k}$ and $b^k\in\mathbb{R}^{n_{k+1}}$ are the weight matrix and bias vector of the $(k+1)$-th layer, respectively; and $n=\sum_{k=1}^{l}n_k$ neurons. The mapping ϕ^k applies a nonlinear scalar activation function $\psi:\mathbb{R}\to\mathbb{R}$ on each one of the entries of its vector argument $W^kx^k+b^k$. The mapping can be defined as:

$$\phi^l(x)=[\psi(x_1)\ \psi(x_2)\ \dots\ \psi(x_{n_k})]^T. \tag{3}$$

Common choices for the scalar activation function include the hyperbolic tangent $\tanh$, the sigmoid, and the rectified linear unit (ReLU). The output $\Psi(x)$ is application dependent. For example, in classification, $\Psi(x)$ is the logit input to a softmax function; in feedback control, $\Psi(x)$ is the control to the plant at state x.

To facilitate subsequent derivations, let us isolate the nonlinear and linear components of a NN as in [8], [17]. Let $v^k=W^kx^k+b^k$ denote the input to the activation function at layer $k+1$. Then, the NN defined in (2) can be rewritten as

$$\begin{bmatrix}v^k\\ \Psi(x)\end{bmatrix}=N\begin{bmatrix}x^k\\ x^l\end{bmatrix}+\begin{bmatrix}b^k\\ b^l\end{bmatrix} \tag{4}$$

$$x^l=\phi(v^k) \tag{5}$$

where matrix N depends on the NN weights and can be partitioned as follows:

$$\begin{aligned} N&=\left[\begin{array}{c|ccccc} W^0 & 0 & \dots & 0 & 0 & 0\\ 0 & W^1 & \dots & 0 & 0 & 0\\ \vdots & \vdots & \dots & \vdots & W^{l-1} & 0\\ \hline 0 & 0 & \dots & 0 & 0 & W^l \end{array}\right]\\ &=\begin{bmatrix}N_{vx} & N_{vx^1}\\ N_{fx} & N_{fx^1}\end{bmatrix}. \end{aligned}$$

III. Specification analysis for a fixed NN

We now briefly review the analysis conducted in [17] based on the framework quadratic constraints. Note that in this section, we consider the NN parameters as fixed (i.e., not subject to learning).

A. Input set

We focus on the type of input sets that can be characterized as follows [17], [26].

Definition 3.1 (Quadratic Constraints): Let $\mathcal{X}$ be a nonempty set and $\mathcal{P}\subset\mathbb{S}^{n_x+1}$ be the set of all symmetric (but possibly indefinite matrices) P such that the following quadratic constraint (QC) holds for all $x\in\mathcal{X}$:

$$\begin{bmatrix}x\\1\end{bmatrix}^T P\begin{bmatrix}x\\1\end{bmatrix}\geq 0. \tag{6}$$

Then, we say that $\mathcal{X}$ satisfies the QC defined by $\mathcal{P}$.

Remark 1: By definition, $\mathcal{P}$ is a convex cone (i.e., for any $P_1,P_2\in\mathcal{P}$, we have $c_1P_1+c_2P_2\in\mathcal{P}$, for any $c_1,c_2\in\mathbb{R}^+$).

Thus, we can over-approximate $\mathcal{X}$ with the intersection of possibly infinite number of sets:

$$\mathcal{X}\subseteq\bigcap_{P\in\mathcal{P}}\left\{x\in\mathbb{R}^{n_x}:\begin{bmatrix}x\\1\end{bmatrix}^T P\begin{bmatrix}x\\1\end{bmatrix}\geq 0\right\} \tag{7}$$

In this paper, we only consider ellipsoid as input sets: $\mathcal{X}=\{x\in\mathbb{R}^{n_x}:\|Ax+b\|_2\leq 1\}$ with $A\in\mathbb{S}^{n_x}$ and $b\in\mathbb{R}^{n_x}$, which satisfies QC with

$$\mathcal{P}_{\mathcal{X}}=\left\{\lambda\begin{bmatrix}-A^TA & A^Tb\\ -b^TA & 1-b^Tb\end{bmatrix},\ \ \lambda\geq 0\right\}. \tag{8}$$

Nonetheless, as in [17], [26], other types of sets such as polytopes, hyper-rectangles and zonotopes can be also incorporated.

B. Specification set

The desirable properties to be verified are formulated as a specification set $\mathcal{S}(x)$ in the output space of the neural network. However checking the condition, $\Psi(x)\in\mathcal{S}(x)$ for all $x\in\mathcal{X}$, is a challenging task as it requires an exact computation of the non-convex set of outputs. Instead, our goal is to find a non-conservative over-approximation the output set and verify the safety properties with respect to the new sets. We assume the safety set is represented by the intersection of finitely many quadratic inequalities.

$$\mathcal{S}(x)=\bigcap_{i=1}^{m}\left\{y\in\mathbb{R}^{n_y}:\begin{bmatrix}x\\y\\1\end{bmatrix}^T S_i\begin{bmatrix}x\\y\\1\end{bmatrix}\leq 0\right\} \tag{9}$$

where the $S_i \in \mathbb{S}^{n_x+n_y+1}$ are given. For instance, if the output set is specified by an ellipsoid, i.e., $\mathcal{S}(x) = \{y \in \mathbb{R}^{n_y} : \|Cy + d\| \leq 1,\ y = \Psi(x)\}$, then we can choose

$$S_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & C^TC & C^Td \\ 0 & d^TC & d^Td - 1 \end{bmatrix}. \tag{10}$$

C. Abstraction of activation functions

One of the challenges in the analysis of NN is the composition of nonlinear activation functions. By exploiting the common patterns of activation functions, a viable approach is to employ sector bounds [8], [17]. We begin with a formal definition.

Definition 3.2 (QC for functions): Let $\phi : \mathbb{R}^n \to \mathbb{R}^n$ and suppose $\mathcal{Q} \subset \mathbb{S}^{2n+1}$ is the set of all symmetric indefinite matrices Q such that the following inequality holds for all $x \in \mathbb{R}^n$:

$$\begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix}^T Q \begin{bmatrix} x \\ \phi(x) \\ 1 \end{bmatrix} \leq 0. \tag{11}$$

Then, we say that ϕ satisfies the QC defined by $\mathcal{Q}$.

The derivation of quadratic constraints is function specific but there are certain heuristics that can be utilized depicted below.

Definition 3.3: Let $\alpha \leq \beta$ be given. The function $\psi : \mathbb{R} \to \mathbb{R}$ lies in the sector $[\alpha, \beta]$ if:

$$(\psi(x) - \alpha x)(\beta x - \psi(x)) \geq 0, \quad \forall x \in \mathbb{R}.$$

The interpretation of sector $[\alpha, \beta]$ is that $y = \psi(x)$ lies in the region formed by the lines $y = \alpha x$ and $y = \beta x$ passing through origin (see Fig. 1).

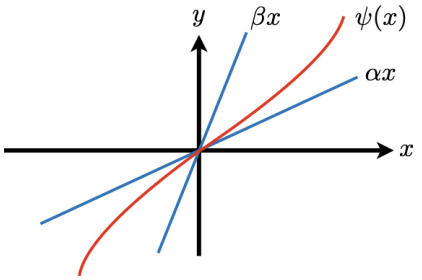

Fig. 1. Illustration of sector bounds for nonlinearity, i.e., $\alpha \leq \frac{\psi(x)}{x} \leq \beta$.

Local sector constraints can also be defined for vector valued function ϕ. These local sectors can be concatenated in the form of vectors $\alpha_\phi, \beta_\phi \in \mathbb{R}^{n_\phi}$.

Lemma 3.4 ([8]): Let α_ϕ, β_ϕ, $\underline{x}$, $\bar{x} \in \mathbb{R}^{n_\phi}$ be given with $\alpha_\phi \leq \beta_\phi$. And ϕ satisfies the local sector $[\alpha_\phi, \beta_\phi]$ coordinate-wise for all $x \in [\underline{x}, \bar{x}]$. If $\mu \in \mathbb{R}^{n_\phi}$ with $\mu \geq 0$ then:

$$\begin{bmatrix} x \\ \phi(x) \end{bmatrix}^T \begin{bmatrix} -2A_\phi B_\phi M & (A_\phi + B_\phi)M \\ (A_\phi + B_\phi)M & -2M \end{bmatrix} \begin{bmatrix} x \\ \phi(x) \end{bmatrix} \geq 0 \tag{12}$$

where $A_\phi = diag(\alpha_\phi)$, $B_\phi = diag(\beta_\phi)$, and $M = diag(\mu)$.

D. Admissibility analysis of NN

Based on quadratic constraint abstractions and $\mathcal{S}$-procedure, the following result provides the admissibility condition of a fixed NN [17].

Theorem 3.1: Consider a two-layer NN $\Psi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ described by (2), with nonlinear activation function sector bounded as in (12). Consider the following matrix inequality

$$M_{\mathcal{X}}(P) + M_{\mathcal{Y}}(S) + M_{\Psi}(Q) \preceq 0 \tag{13}$$

where

$$M_{\mathcal{X}}(P) = \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T P \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{\Psi}(Q) = \begin{bmatrix} W^{0^T} & 0 & 0 \\ 0 & I_{n_1} & 0 \\ b^{0^T} & 0 & 1 \end{bmatrix} \begin{bmatrix} -2A_\phi B_\phi M & (A_\phi + B_\phi)M \\ (A_\phi + B_\phi)M & -2M \end{bmatrix} \begin{bmatrix} W^0 & 0 & b^0 \\ 0 & I_{n_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{\mathcal{Y}}(S) = \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & W^{1^T} & 0 \\ 0 & b^{1^T} & 1 \end{bmatrix} S \begin{bmatrix} I_{n_0} & 0 & 0 \\ 0 & W^1 & b^1 \\ 0 & 0 & 1 \end{bmatrix}$$

If (13) is feasible for (P, Q, S), then $\Psi(x) \in \mathcal{S}(x)$, for all $x \in \mathcal{X}$.

The above theorem implies that for a given NN, if (13) is feasible, then we can certify the admissibility. However, it is seldom the case that an NN learned with an unconstrained approach can satisfy the specified constraints. To reliably learn an admissible NN, it seems straightforward to impose the specifications as constraints. Nevertheless, the analysis condition (13) is nonconvex with respect to both NN weights and multipliers (P, Q, S), making the problem computationally intractable to solve.

IV. Convexified Learning under Specifications

The key idea is to reparametrize the NN such that the condition becomes convex in a transformed space. To streamline the presentation, we derive our results for a two-layer NN with $l = 1$ hidden layer (i.e., $\Psi(x) = W^1\phi(W^0x + b^0) + b^1$). In this case, (2) can be rewritten as:

$$\begin{bmatrix} v^0 \\ \Psi(x) \end{bmatrix} = N \begin{bmatrix} x \\ x^1 \end{bmatrix} + \begin{bmatrix} b^0 \\ b^1 \end{bmatrix} \tag{14}$$

$$x^1 = \phi(v^0), \tag{15}$$

where the matrix N depends on the weights as follows:

$$N = \begin{bmatrix} N_{vx} & N_{vx^1} \\ N_{fx} & N_{fx^1} \end{bmatrix} = \begin{bmatrix} W^0 & 0 \\ 0 & W^1 \end{bmatrix} \tag{16}$$

A. Loop transformation

Loop transformation is a standard linear fractional transformation manipulation in the control literature. Through loop transformation, we obtain a new representation that convexifies the learning condition without imposing restrictions on sector bounds α_ϕ and β_ϕ of the activation function. In particular loop transformation normalizes the nonlinearity $\tilde{\phi}$ to lie in the sector $[-1_{n_\phi \times 1}, 1_{n_\phi \times 1}]$. Thereby $\tilde{x}^1 = \tilde{\phi}(\nu^0)$ satisfies the quadratic constraint

$$\begin{bmatrix} \nu^0 \\ \tilde{x}^1 \end{bmatrix}^T \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} \nu^0 \\ \tilde{x}^1 \end{bmatrix} \geq 0, \ \forall \nu^0 \in [\underline{\nu}, \bar{\nu}], \tag{17}$$

where $M = diag(\mu)$. The input to N is transformed by the following algebraic equation.

$$x^1 = \frac{B_\phi - A_\phi}{2} \tilde{x}^1 + \frac{B_\phi + A_\phi}{2} v^0 \tag{18}$$

Substituting (18) in (14), we get

$$v^0 = N_{vx} x + N_{vx^1} \left(\frac{B_\phi - A_\phi}{2} \tilde{x}^1 + \frac{A_\phi + B_\phi}{2} v^0 \right) + b^0 \tag{19}$$

$$f(x) = N_{fx} x + N_{fx^1} \left(\frac{B_\phi - A_\phi}{2} \tilde{x}^1 + \frac{A_\phi + B_\phi}{2} v^0 \right) + b^1. \tag{20}$$

By solving (19), we obtain the expression for v^0,

$$v^0 = (I - C_2)^{-1} N_{vx} x + (I - C_2)^{-1} C_1 \tilde{x}^1 + (I - C_2)^{-1} b^0 \tag{21}$$

Substituting v^0 in (20) yields

$$\begin{aligned} f(x) = &\left(N_{fx} + C_4 (I - C_2)^{-1} N_{vx} \right) x + C_4 (I - C_2)^{-1} b^0 \\ &+ \left(C_3 + C_4 (I - C_2)^{-1} C_1 \right) \tilde{x}^1 + b^1, \end{aligned} \tag{22}$$

with

$$\begin{aligned} C_1 = N_{vx^1} \frac{B_\phi - A_\phi}{2}, \quad C_2 = N_{vx^1} \frac{B_\phi + A_\phi}{2}, \\ C_3 = N_{fx^1} \frac{B_\phi - A_\phi}{2}, \quad C_4 = N_{fx^1} \frac{B_\phi + A_\phi}{2} \end{aligned}$$

.

After applying the loop transformation on neural network controller the new representation of NN is equivalent to

$$\begin{bmatrix} v^0 \\ f(x) \end{bmatrix} = \tilde{N} \begin{bmatrix} x \\ \tilde{x}^1 \end{bmatrix} + \begin{bmatrix} \tilde{b}^0 \\ \tilde{b}^1 \end{bmatrix} \tag{23}$$

$$\tilde{x}^1 = \tilde{\phi}(v^0). \tag{24}$$

where $\tilde{b}^0 = (I - C_2)^{-1} b^0$, $\tilde{b}^1 = C_2 (I - C_2)^{-1} b^0 + b^1$, and

$$\tilde{N} = \begin{bmatrix} (I - C_2)^{-1} N_{vx} & (I - C_2)^{-1} C_1 \\ N_{fx} + C_4 (I - C_2)^{-1} N_{vx} & C_3 + C_4 (I - C_2)^{-1} C_1 \end{bmatrix}. \tag{25}$$

It can be seen that $\tilde{N}$ is in general a nonlinear function of N. To solve the equation, an ADMM algorithm is developed in [8]. Also, it is important to note that $\tilde{N}$ depends indirectly on N through the sector bounds (A_ϕ, B_ϕ). Specifically, suppose both N and the state bounds are given. Then $\tilde{N}$ is constructed by: (i) propagating bounds through NN to compute bounds $\underline{\nu}, \bar{\nu}$ on the activation inputs, (ii) compute local sector bounds A_ϕ, B_ϕ consistent with the activation bounds, and (iii) performing steps to compute $\tilde{N}$ from (N, A_ϕ, B_ϕ). Hereafter we treat $\tilde{N}$ as decision variable instead of N (i.e., $\tilde{N}$ is the reparametrization of N).

B. Admissibility condition after loop transformation

Now we analyze the admissibilty of NN after loop transformation. Consider input and output sets to be ellipsoids. Based on the new representation of NN, the matrix inequality (13) can be rewritten as:

$$\tilde{M}_{\mathcal{X}} + \tilde{M}_{\Psi} + \tilde{M}_{\mathcal{Y}} \preceq 0, \tag{26}$$

with

$$\tilde{M}_{\mathcal{X}} = \begin{bmatrix} I_{n_0} & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -A^T \mu A & -A^T \mu b \\ -b^T \mu A & \mu(1 - b^T b) \end{bmatrix} \begin{bmatrix} I_{n_0} & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}^T \tag{27}$$

$$\begin{aligned} \tilde{M}_{\Psi} = &\begin{bmatrix} \tilde{N}_{vx}^T & 0 & 0 \\ \tilde{N}_{vx^1}^T & I_{n_1} & 0 \\ \tilde{b}^{0^T} & 0 & 1 \end{bmatrix} \begin{bmatrix} M & 0 & 0 \\ 0 & -M & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &\begin{bmatrix} \tilde{N}_{vx} & \tilde{N}_{vx^1} & \tilde{b}^0 \\ 0 & I_{n_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{28}$$

$$\begin{aligned} \tilde{M}_{\mathcal{Y}} = &\begin{bmatrix} I_{n_0} & \tilde{N}_{fx}^T & 0 \\ 0 & \tilde{N}_{fx^1}^T & 0 \\ 0 & \tilde{b}^{1^T} & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & A_y^T A_y & A_y^T b_y \\ 0 & b_y^T A_y & b_y^T b_y - 1 \end{bmatrix} \\ &\begin{bmatrix} I_{n_0} & 0 & 0 \\ \tilde{N}_{fx} & \tilde{N}_{fx^1} & \tilde{b}^1 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{29}$$

By substituting (27)–(29) in (26), and after simplification, we obtain:

$$\begin{aligned} &\begin{bmatrix} \star \end{bmatrix}^T \begin{bmatrix} M & 0 \\ 0 & I_y \end{bmatrix} \begin{bmatrix} \tilde{N}_{vx} & \tilde{N}_{vx^1} & \tilde{b}^0 \\ \tilde{N}_{fx} C & \tilde{N}_{fx^1} C & \tilde{b}^1 C + d \end{bmatrix} \\ &\quad - \begin{bmatrix} A^T \lambda A & 0 & A^T \lambda b \\ 0 & M & 0 \\ b^T \lambda A & 0 & \lambda(b^T b - 1) + 1 \end{bmatrix} \preceq 0. \end{aligned} \tag{30}$$

Applying Schur complements yields an equivalent condition:

$$\begin{bmatrix} A^T \lambda A & 0 & A^T \lambda b & \tilde{N}_{vx}^T & \tilde{N}_{fx}^T C^T \\ 0 & M & 0 & \tilde{N}_{vx^1}^T & \tilde{N}_{fx^1}^T C^T \\ b^T A \lambda & 0 & \lambda(b^T b - 1) + 1 & \tilde{b}^0 & \tilde{b}^{1T} C^T + d^T \\ \tilde{N}_{vx} & \tilde{N}_{vx^1} & \tilde{b}^0 & T^{-1} & 0 \\ C \tilde{N}_{fx} & C L_2 & C \tilde{b1} + d & 0 & I_{ny} \end{bmatrix} \succeq 0 \tag{31}$$

The above inequality (31) is linear in NN weights and bias vectors, but still nonconvex in T. Now multiply (31) on the left and right sides by $diag\left(\begin{bmatrix} I_{n_x} & 0 \\ 0 & T^{-1} \end{bmatrix}, I_{1+n1+n_y} \right)$ to get the required convex condition,

$$\begin{bmatrix} A^T\lambda A & 0 & A^T\lambda b & \tilde{N}_{vx}^T & \tilde{N}_{fx}^T C^T \\ 0 & Q_1 & 0 & L_1^T & L_2^T C^T \\ b^T A\lambda & 0 & \lambda(b^T b-1)+1 & \tilde{b}^0 & \tilde{b}^{1T} C^T + d^T \\ \tilde{N}_{vx} & L1 & \tilde{b}^0 & Q_1 & 0_{n1\times n_y} \\ C\tilde{N}_{fx} & CL_2 & C\tilde{b}1+d & 0_{ny\times n1} & I_{ny} \end{bmatrix} \succeq 0 \tag{32}$$

where $Q_1 = (T)^{-1} \succ 0$, $L_1 = \tilde{N}_{vx1} Q_1$, and $L_2 = \tilde{N}_{fx^1} Q_1$. Now the above constraint (32) is convex in the decision variables $Q_1, \lambda, L_1, L2, \tilde{N}_v x, \tilde{N}_{fx}, \tilde{b}^0$, and $\tilde{b}^1$. As a result, we can efficiently search over the admissible NN parameters by imposing this condition during learning.

C. Algorithm

The learning procedure is as follows, which involves finding a feasible solution to the LMI condition (32), and recovering the NN parameters from the numerical solutions:

1. Formulate and abstract the provided input-output specifications according to Sections III-A and III-B.
2. Numerically solve for a feasible solution (L, Q) to the LMI (32).
3. If a feasible solution is found successfully, compute the NN weights by solving (25).

If a feasible solution is found in step 2, then by Theorem 4.1, the corresponding NN recovered in step 3 certifiably meet the input-output specifications; nevertheless, infeasibility of (32) in general does not imply the emptiness of the admissible set—a limitation due to the potential conservativeness of convex relaxation approaches.

D. Multi-layer neural network

The extension to a multi-layer NN is straightforward. Define $x = [x^0, x^1, \ldots, x^l]$, where $l \geq 1$ is the number of hidden layers, and $x^k = E^k x$ for $k = 0, \ldots, l$, where E^k is the entry selector matrix. Also, denote

$$A = \begin{bmatrix} W^0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & W^1 & 0 & \ldots & 0 & 0 \\ . & . & W^2 & \ldots & . & . \\ . & . & . & \ldots & . & . \\ . & . & . & \ldots & . & . \\ 0 & 0 & 0 & \ldots & W^{l-1} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & I_{n_1} & 0 & \ldots & 0 & 0 \\ . & . & I_{n_2} & \ldots & . & . \\ . & . & . & \ldots & . & . \\ . & . & . & \ldots & . & . \\ 0 & 0 & 0 & \ldots & 0 & I_{n_l} \end{bmatrix}, b = \begin{bmatrix} b^0 \\ b^1 \\ . \\ . \\ . \\ b^{l-1} \end{bmatrix}$$

The following result provides a convex condition for learning a multi-layer NN under specifications.

Theorem 4.1: Consider a multi-layer NN $\Psi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ described by (2), with nonlinear activation function sector bounded as in (12). Consider the following matrix inequality

$$\tilde{M}_{\mathcal{X}}(P) + \tilde{M}_{\mathcal{Y}}(S) + \tilde{M}_{\Psi}(Q) \preceq 0 \tag{33}$$

where

$$\tilde{M}_{\mathcal{X}}(P) = \begin{bmatrix} E^0 & 0 \\ 0 & 1 \end{bmatrix}^T P \begin{bmatrix} E^0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\tilde{M}_{\Psi}(Q) = \begin{bmatrix} \tilde{A} & \tilde{b} \\ B & 0 \\ 0 & 1 \end{bmatrix}^T Q \begin{bmatrix} \tilde{A} & \tilde{b} \\ B & 0 \\ 0 & 1 \end{bmatrix}$$

$$\tilde{M}_{\mathcal{Y}}(S) = \begin{bmatrix} E^0 & 0 \\ \tilde{W}^{l^T} E^l & \tilde{b}^l \\ 0 & 1 \end{bmatrix}^T S \begin{bmatrix} E^0 & 0 \\ \tilde{W}^{l^T} E^l & \tilde{b}^l \\ 0 & 1 \end{bmatrix}.$$

If (33) is feasible for (P, Q, S), then $\Psi(x) \in \mathcal{S}(x)$, for all $x \in \mathcal{X}$.

V. Numerical Experiments

To validate the proposed method, we consider a reachability problem. For a given input set, the specification dictates that the output of the NN lies within a set. Our task is to learn a set of NN weights and bias vectors that satisfy this requirement. We implement our algorithm in MATLAB and solve the LMI condition with SDPT3 [27]. For the first experiment, we consider two pairs of ellipsoids of the same size as the input-output sets to learn a NN with dimensions $n_x = 2$, $n_y = 2$ and $n_1 = 10$ hidden neurons. The input and output sets are shown in Fig. 2 (note that our method works in a wide range of positions and we only showcase one of typical example here). To test the admissibility of the learned NN, we randomly generate 500 points in the input sets and propagate them through the NN. As expected, all the output points lie within the desired output sets. To "stress test" the proposed method, we conduct another experiment where we consider 3 pairs of input-output sets, but reduce the number of hidden neurons to 5. We also change the sizes of input/output sets. A successful attempt is shown in Fig. 3. In general, as we increase the number of hidden layers, we observe a higher likelihood of finding a feasible solution, which is aligned with the universal approximation theorem [28]. Further quantification of the representation capacity

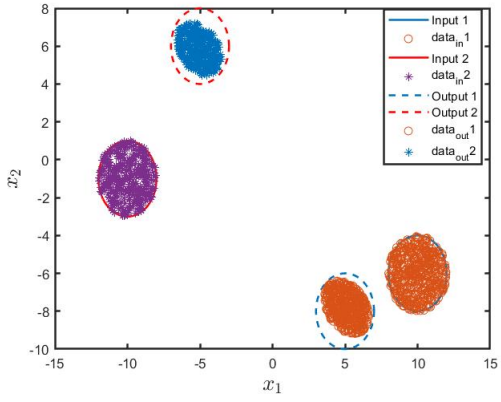

Fig. 2. NN mapping with 10 hidden neurons.

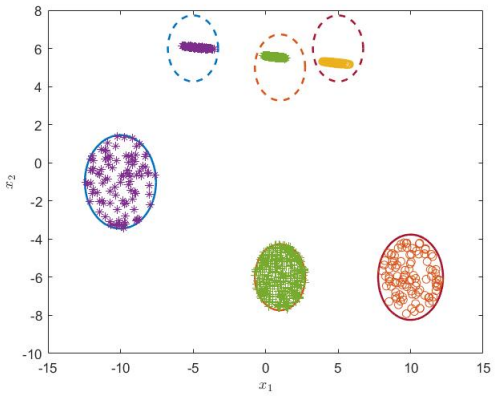

Fig. 3. NN mapping with 5 neurons.

informed by our admissibility condition is an interesting topic to future research.

VI. Conclusion and Future Directions

We address the challenge of learning NNs that certifiably satisfy input-output specifications. To tractably search for admissible weights, we derive a convex inner approximation to the nonconvex set of all admissible parameters. By abstracting the nonlinear specifications and activation functions with QCs, and applying the technique of loop transformation, we are able to derive a convex condition for a multi-layer NN that can be solved via SDP. The theoretical construction is verified by numerical experiments for a reachability-type problem. Building on the present work, there are several directions that we are currently pursuing, including (i) addressing more general forms of specifications, including those that can be approximated by QCs and those with internal dynamics; (ii) extending the theory to address convolutional neural networks, which have wide applications in extracting temporal and spatial correlations within data; and (iii) deriving learning-theoretic guarantees for a sample-based approach to solving problems with a large number of input-output specifications (that would otherwise been challenging to solve within a single SDP).

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning", MIT press, 2016.

[2] I. Sutskever, O. Vinyals, and QV. Le. "Sequence to sequence learning with neural networks," In Advances in neural information processing systems, pp. 3104-3112. 2014.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems 25 (2012): 1097-1105.

[4] I. Stoica, D. Song, R. A. Popa, D. Patterson, M. W. Mahoney, R. Katz, A. D. Joseph et al. "A berkeley view of systems challenges for ai," arXiv preprint arXiv:1712.05855 (2017).

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. "Intriguing properties of neural networks," arXiv preprint arXiv:1312.6199 (2013).

[6] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. "Concrete problems in AI safety," arXiv preprint arXiv:1606.06565 (2016).

[7] M. Jin, and J. Lavaei. "Stability-certified reinforcement learning: A control-theoretic perspective," IEEE Access 8 (2020): 229086-229100.

[8] H. Yin, P. Seiler, M. Jin, and M. Arcak. "Imitation learning with stability and safety guarantees," IEEE Control Systems Letters (2021).

[9] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, and Z. Lin. "When machine learning meets privacy: A survey and outlook," ACM Computing Surveys (CSUR) 54, no. 2 (2021): 1-36.

[10] N. Mehrabi, Ninareh, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. "A survey on bias and fairness in machine learning" ACM Computing Surveys (CSUR) 54, no. 6 (2021): 1-35.

[11] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang. "A survey on neural network interpretability," IEEE Transactions on Emerging Topics in Computational Intelligence (2021).

[12] S. H. Silva, and P. Najafirad. "Opportunities and challenges in deep learning adversarial robustness: A survey" arXiv preprint arXiv:2007.00753 (2020).

[13] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. "Linear matrix inequalities in system and control theory," Society for industrial and applied mathematics, 1994.

[14] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. "Ai2: Safety and robustness certification of neural networks with abstract interpretation," In 2018 IEEE Symposium on Security and Privacy (SP), pp. 3-18. IEEE, 2018.

[15] E. Wong, F. R. Schmidt, J. H. Metzen, and J. Z. Kolter. "Scaling provable adversarial defenses," arXiv preprint arXiv:1805.12514 (2018).

[16] K. Dvijotham, S. Gowal, R. Stanforth, R. Arandjelovic, B. O'Donoghue, J. Uesato, and P. Kohli. "Training verified learners with learned verifiers," arXiv preprint arXiv:1805.10265 (2018).

[17] M. Fazlyab, M. Morari, and G. J. Pappas. "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming." IEEE Transactions on Automatic Control (2020).

[18] A. Megretski, and A. Rantzer. "System analysis via integral quadratic constraints," IEEE Transactions on Automatic Control 42, no. 6 (1997): 819-830.

[19] S. Sastry. "Nonlinear systems," analysis, stability, and control. Vol. 10. Springer Science and Business Media, 2013.

[20] V. A. Yakubovich, "S-procedure in nolinear control theory," Vestnik Leninggradskogo Universiteta, Ser. Matematika (1971): 62-77.

[21] C. Qin, B. O'Donoghue, R. Bunel, R. Stanforth, S. Gowal, J. Uesato, G. Swirszcz, and P. Kohli. "Verification of non-linear specifications for neural networks," arXiv preprint arXiv:1902.09592 (2019).

[22] X. Yin, Z. Jiang, and Li Pan. "Recurrent neural network based adaptive integral sliding mode power maximization control for wind power systems," Renewable Energy 145 (2020): 1149-1157.

[23] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. "A lyapunov-based approach to safe reinforcement learning," arXiv preprint arXiv:1805.07708 (2018).

[24] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel. "Learning deep neural network policies with continuous memory states," In 2016 IEEE international conference on robotics and automation (ICRA), pp. 520-527. IEEE, 2016.

[25] M. Revay, R. Wang, and I. R. Manchester. "A convex parameterization of robust recurrent neural networks," IEEE Control Systems Letters 5, no. 4 (2020): 1363-1368.

[26] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas. "Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming," In 2020 59th IEEE Conference on Decision and Control (CDC), pp. 5929-5934. IEEE, 2020.

[27] K.C Toh, M. J. Todd, and R. H. Tütüncü. "SDPT3—a MATLAB software package for semidefinite programming, version 1.3," Optimization methods and software 11, no. 1-4 (1999): 545-581.

[28] K. Hornik, "Multilayer feed-forward networks are universal approximators," Artificial neural networks: Approximation and learning theory (1992).

[29] L. P. Donti, D. Rolnick, and J. Z. Kolter. "DC3: A learning method for optimization with hard constraints," arXiv preprint arXiv:2104.12225 (2021).