Automated Reachability Analysis of Neural Network-Controlled Systems via Adaptive Polytopes

Taha Entesari   TENTESA1@JHU.EDU
Department of Electrical and Computer Engineering, Johns Hopkins University, USA

Mahyar Fazlyab   MAHYARFAZLYAB@JHU.EDU
Department of Electrical and Computer Engineering, Johns Hopkins University, USA

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Over-approximating the reachable sets of dynamical systems is a fundamental problem in safety verification and robust control synthesis. The representation of these sets is a key factor that affects the computational complexity and the approximation error. In this paper, we develop a new approach for over-approximating the reachable sets of neural network dynamical systems using adaptive template polytopes. We use the singular value decomposition of linear layers along with the shape of the activation functions to adapt the geometry of the polytopes at each time step to the geometry of the true reachable sets. We then propose a branch-and-bound method to compute accurate over-approximations of the reachable sets by the inferred templates. We illustrate the utility of the proposed approach in the reachability analysis of linear systems driven by neural network controllers.

Keywords: Template Polytopes, Branch and Bound, Neural Network Verification, Reachability Analysis

1. Introduction

As the use of neural networks has expanded into safety-critical applications such as autonomous systems and automated healthcare, there is a growing need to develop efficient and scalable methods to rigorously verify neural networks against input uncertainties. The canonical problem is to verify that for a bounded set of inputs, the reachable set of a trained model does not intersect with an unsafe set. From an optimization perspective, this problem can be formulated as a constraint satisfaction feasibility problem, where the goal is to either verify the constraint or find a counter-example.

Going beyond machine learning, neural networks also arise as function approximators in feedback control. Compared to open-loop settings, verification of neural networks in closed-loop systems is a more challenging problem, as it requires an explicit characterization of the reachable sets at each iteration. Computation of reachable sets for dynamical systems is a fundamental problem that arises in, for example, safety verification and robust control synthesis–see Althoff et al. (2021) for an overview. Formally, given the dynamical system $x^{k+1} = F(x^k)$ $x^0 \in \mathcal{X}^0$, where $\mathcal{X}^0$ is a bounded set of initial conditions, the reachable set at time $k + 1$ is defined as

$$\mathcal{X}^{k+1} = F(\mathcal{X}^k) = \{F(x) \mid x \in \mathcal{X}^k\}, \tag{1}$$

Since it is generally difficult to compute the reachable sets exactly, these sets are often over-approximated iteratively by a sequence of *template sets* $(\bar{\mathcal{X}}_k)_{k \geq 0}$: starting with $\bar{\mathcal{X}}^0 = \mathcal{X}^0$, we over-

approximate the image of $\bar{\mathcal{X}}^k$ under F using a *set propagation algorithm* to ensure that $F(\bar{\mathcal{X}}^k) \subseteq \bar{\mathcal{X}}^{k+1}$ for $k = 0, 1, \cdots$. If the over-approximated sets do not intersect with a set of unsafe states (e.g., obstacles), then safety can be guaranteed. However, the over-approximation error can quickly accumulate over time (known as the wrapping effect Neumaier (1993)), leading to overly conservative bounds for long time horizons. This challenge becomes even more pronounced when neural networks are involved in the feedback loop (e.g., when F is itself a neural network approximation of an ODE).

The choice of template sets can have a crucial effect on the accuracy of computations. Indeed, any potential mismatch between the shape of the template set and the actual reachable set can lead to conservative bounds (shape mismatch error). To minimize this error, it is essential to use *dynamic* template sets that can adapt to the geometry of the reachable sets based on the structure of F. Furthermore, the method by which we propagate the sets through F can incur conservatism due to the underlying relaxations (propagation error). This error can be mitigated by using less conservative relaxations in the propagation method and/or by partitioning the input set.

Our Contributions In this paper, we propose a novel method for reachability analysis of discrete-time affine systems in feedback with ReLU[1] neural network controllers. Using bounded polyhedra to represent the template sets, we propose a method that dynamically adapts the geometry of the template sets to that of reachable sets based on the structure of the closed-loop map. Thus, this approach eliminates the manual selection of the template directions, making the procedure fully automated. Based on the chosen template directions, we then compute tight[2] polyhedral over-approximations of the reachable sets using efficient branch-and-bound (BnB) algorithms. Our method is modular in that it can incorporate any bound propagation method for neural networks. Our code is available at https://github.com/o4lc/AutomatedReach.git.

1.1. Related Work

Open-loop Verification Verifying piecewise linear networks can be cast as an MILP with the binary variables describing the ReLU neurons Cheng et al. (2017); Tjeng et al. (2017); Dutta et al. (2018b); Lomuscio and Maganti (2017); Fischetti and Jo (2018). These problems can be solved globally using generic BnB methods, in which the optimization problem is recursively divided into sub-problems by branching the binary variables. The optimal value of each sub-problem is then bounded using convex (linear) relaxations, leading to provable bounds on the optimal objective value of the original problem. However, generic BnB solvers do not exploit the underlying structure of the problem and hence, may be inefficient. As such, state-of-the-art methods for complete verification develop customized BnB methods in which efficient bound propagation methods are used Wang et al. (2021); Bunel et al. (2020); De Palma et al. (2021); Xu et al. (2020); Kouvaros and Lomuscio (2021); Ferrari et al. (2022). Branching is done either on binary variables that describe the ReLU activations or on the input set. Vincent and Schwager (2021) branches using the activation pattern of the ReLU nonlinearities and uses polyhedral sections in the input space, making the neural network an affine function in each polyhedron.

Closed-loop Verification When the choice of set representation is template polyhedra, the problem of reachability analysis reduces to finding the maximum output along the normal directions

---

1. Rectified Linear Unit
2. We globally solve the corresponding non-convex problems within an arbitrary accuracy.

Dutta et al. (2018a). Some approaches to solve this problem approximate the overall network using polynomials Dutta et al. (2019); Huang et al. (2019). Other methods abstract the input-output relationship of neurons using intervals Clavière et al. (2021), star sets Tran et al. (2019), Taylor models Ivanov et al. (2021), polynomial zonotopes Kochdumper et al. (2022), hybrid zonotopes Zhang and Xu (2022), and integration of Taylor models with zonotopes Schilling et al. (2022). Sidrane et al. (2022) explore the extension to nonlinear systems by abstracting nonlinear functions with a set of optimally tight piecewise linear bounds. Hu et al. (2020) over-approximates the reachable sets using semidefinite relaxations, which can be used to propagate ellipsoids in addition to polytopes. Everett et al. (2021) use a looser convex relaxation Zhang et al. (2018) but bridge the gap by partitioning in the state space. Entesari et al. (2022) propose a BnB framework using Lipschitz bounds. Chen et al. (2022a) characterize the conditions for the set propagation algorithm under which unrolling the dynamics yields less conservative bounds on the reachable sets. Rober et al. (2022) propose a framework for closed-loop verification based on backward reachability analysis.

Set Representation   Axis-aligned hyperrectangles or oriented hyperrectangles, acquired by running principal component analysis (PCA) on samples trajectories Stursberg and Krogh (2003), can be convenient choices as the number of facets grows only linearly with the ambient dimension. However, they can become too conservative. On the other hand, there exist more complex methods like Bogomolov et al. (2017) that solve convex problems to derive template polytope directions or Ben Sassi et al. (2012) that use a first-order Taylor approximation of F (assuming F is differentiable) to propose dynamic directions.

1.2. Notation

For a real number r, $(r)_+ = \max(r, 0)$ is the non-negative part of that number. For any vector $x \in \mathbb{R}^n$, $\text{ReLU}(x) = [(x_1)_+, \cdots, (x_n)_+]^\top$ where x_i is the ith element of x. I_n denotes the n by n identity matrix. $0_{n \times m}$ denotes the n by m matrix of all zeros. We use $\text{Poly}(A, d) = \{x \in \mathbb{R}^n \mid Ax \le d\}$ to denote polyhedrons, in which we denote the i-th row of A by $a_i^\top$. This is called the $\mathcal{H}$−representation of polyhedra as it uses the intersection of half-spaces to represent the set. If such a set is bounded, we call it a polytope. Given matrices $A_i \in \mathbb{R}^{n \times n_i}$ and vectors $a_i \in \mathbb{R}^n$, $[A_1, \cdots, A_m, a_1, \cdots, a_k]$ denotes the horizontal concatenation of the elements into a single matrix. For a given function $f : \mathbb{R}^n \to \mathbb{R}^n$, $f^{(k)}$ denotes the k-the composition of f with itself.

2. Problem Statement

Consider a discrete-time autonomous system

$$x^{k+1} = F(x^k) = Ax^k + Bf(x^k) + e, \tag{2}$$

with state $x^k \in \mathbb{R}^n$, control input $u^k \in \mathbb{R}^m$, and system dynamics given by $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $e \in \mathbb{R}^n$ is a known exogenous constant. We assume that the control policy is given by $f(x^k)$, where $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ ($n_0 = n, n_L = m$) is a fully-connected ReLU network as follows

$$x_0 = x, \quad x_{i+1} = \text{ReLU}(W_i x_i + b_i), \quad i = 0, \cdots, L-1, \quad f(x) = W_L x_L + b_L, \tag{3}$$

where $x_i \in \mathbb{R}^{n_i}$, $W_i \in \mathbb{R}^{n_{i+1} \times n_i}$, and $b_i \in \mathbb{R}^{n_{i+1}}$. We assume that the initial state x^0 belongs to a bounded box $\mathcal{X}^0$ of initial conditions, $\mathcal{X}^0 = \{x \in \mathbb{R}^n \mid \underline{x} \le x \le \bar{x}\}$, where $\underline{x}, \bar{x} \in \mathbb{R}^n$. Starting

from this set, the reachable set at time $k+1$ is defined as in (1). For piecewise linear dynamics and polyhedral initial sets, these sets are non-convex.

In this paper, we use bounded polyhedra to parameterize the template sets, $\bar{\mathcal{X}}^k = \text{Poly}(C^k, d^k)$, where the rows of the template matrix $C^k \in \mathbb{R}^{m_k \times n}$ are the normal directions that define the facets of the polyhedron and $d^k \in \mathbb{R}^{m_k}$ is their offset. We use the superscript k for the template matrices to emphasize that the template may not be the same for all the polyhedra. Starting from $\bar{\mathcal{X}}^0 = \mathcal{X}^0$, our goal is to compute the pair (C^k, d^k) such that $\bar{\mathcal{X}}^k$ over-approximates $\mathcal{X}^k$ as closely as possible.

3. Proposed Method

In this section, we present the details of the proposed method. We first cast the closed-loop map as an equivalent ReLU network (§3.1). Then assuming that the template matrices C^k are all given, we elaborate on computing the offset vectors d^k (§3.2). We will finally discuss how to choose the template matrices C^k, the choice of which will not depend on d^k (§3.3).

3.1. Neural Network Representation of the Closed-loop Map

We start by observing that the closed-loop map is essentially a neural network with a skip connection as shown in Fig. 1-(a). State-of-the-art bound propagation methods for neural network verification operate on sequential neural networks, i.e., neural networks without any skip connections. To take advantage of existing bound propagation methods with minimal intervention, we convert the closed-loop map into an equivalent sequential network without skip connections. To be precise, note that we can write $x^k = \text{ReLU}(x^k) - \text{ReLU}(-x^k)$. This implies that $x^k = (\text{ReLU} \circ \cdots \circ \text{ReLU})(x^k) - (\text{ReLU} \circ \cdots \circ \text{ReLU})(-x^k)$, i.e., x^k can be routed through a sequential ReLU network. By concatenating this path on top of f, we obtain the following equivalent representation of F,

$$y_0 = \begin{bmatrix} I_{n_0} \\ -I_{n_0} \\ I_{n_0} \end{bmatrix} x^k + \begin{bmatrix} 0_{n_0 \times 1} \\ 0_{n_0 \times 1} \\ 0_{n_0 \times 1} \end{bmatrix}, \quad y_i = \text{ReLU}(\hat{y}_i), \quad i = 1, \cdots, L,$$

$$\hat{y}_{i+1} = \begin{bmatrix} I_{n_0} & 0_{n_0 \times n_0} & 0_{n_0 \times n_i} \\ 0_{n_0 \times n_0} & I_{n_0} & 0_{n_0 \times n_i} \\ 0_{n_{i+1} \times n_0} & 0_{n_{i+1} \times n_0} & W_i \end{bmatrix} y_i + \begin{bmatrix} 0_{n_0 \times 1} \\ 0_{n_0 \times 1} \\ b_i \end{bmatrix}, \quad i = 0, \cdots, L-1, \tag{4}$$

$$y_{L+1} = \begin{bmatrix} I_{n_0} & 0_{n_0 \times n_0} & 0_{n_0 \times n_L} \\ 0_{n_0 \times n_0} & I_{n_0} & 0_{n_0 \times n_L} \\ 0_{n_{L+1} \times n_0} & 0_{n_{L+1} \times n_0} & W_L \end{bmatrix} y_L + \begin{bmatrix} 0_{n_0 \times 1} \\ 0_{n_0 \times 1} \\ b_L \end{bmatrix}, x^{k+1} = \begin{bmatrix} A & -A & B \end{bmatrix} y_{L+1} + e.$$

We use F_{eq} to denote the neural network in (4) (see Fig. 1-(b)). In summary, we can rewrite the closed-loop system (2) equivalently as

$$x^{k+1} = Ax^k + Bf(x^k) + e = F_{eq}(x^k). \tag{5}$$

Formulating the closed-loop system in the form of (4) is also useful in that it eliminates the need to compute the Minkowski sum of the two polytopes enclosing $A\bar{\mathcal{X}}^k + e$ and $Bf(\bar{\mathcal{X}}^k)$. Not only has the process of finding the Minkowski sum of two polytopes proven to be challenging Althoff et al. (2021), this would amount to the unnecessary enlargement of the approximated reachable set $\bar{\mathcal{X}}^{k+1}$, resulting in less accurate overapproximations of the actual reachable sets. We will subsequently work with the equivalent representation (5).
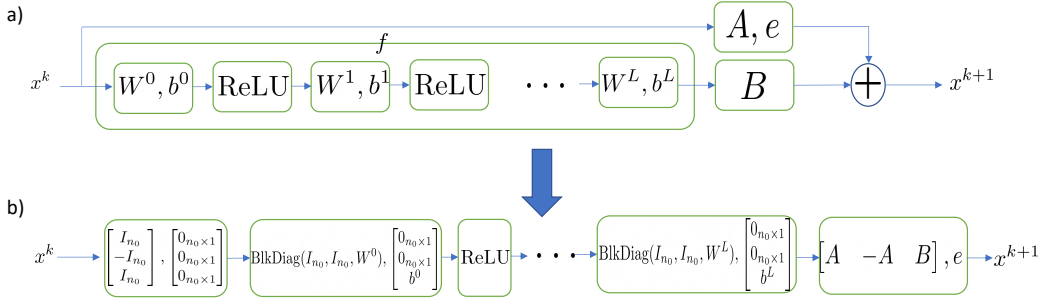
Figure 1: Closed-loop system block diagrams. The first matrix in each block represents the weights matrix of that linear layer, while the second matrix represents the bias of that layer. a) The original system as proposed by (2). b) Conversion of the system dynamics into an equivalent ReLU network, as explained by (4).

3.2. Solving the Optimization Problem Using Branch and Bound

Suppose that the polytope $\bar{\mathcal{X}}^k$ has been computed and that the template matrix C^{k+1} is given. To over-approximate $F(\bar{\mathcal{X}}^k)$ by $\bar{\mathcal{X}}^{k+1} = \text{Poly}(C^{k+1}, d^{k+1})$, the offset vector d^{k+1} of the minimal volume $\bar{\mathcal{X}}^{k+1}$ enclosing $F(\bar{\mathcal{X}}^k)$ must satisfy

$$d_i^{k+1} = \sup\{c_i^{k+1\top} F(x) \mid x \in \bar{\mathcal{X}}^k\} = \sup\{c_i^{k+1\top} F_{eq}(x) \mid x \in \bar{\mathcal{X}}^k\} = \quad i = 1, \cdots, m_{k+1}. \quad (6)$$

where the second equality follows from the equivalence in (5), i.e., $F(x) = F_{eq}^k(x)$ for all $x \in \mathbb{R}^{n_x}$. Computation of the maximal values in (6) involves solving an MILP with the binary variables describing the ReLU neurons Tjeng et al. (2017). Generic MILP solvers may not efficiently utilize the underlying problem structure. Therefore, custom BnB methods aim to exploit highly specialized bound propagation methods in the bounding stage. However, these bound propagation techniques could prove inefficient for general convex polyhedra input sets, such as the polytope $\bar{\mathcal{X}}^k$, as they require solving linear programs (LPs) to compute the bounds. Furthermore, general convex polytopes do not lend themselves to efficient partitioning.

To overcome these obstacles, we propose an end-to-end reachability analysis on the unrolled dynamics, following a similar approach to Chen et al. (2022a). Observing that the reachable set at time $k + 1$ can be written as $\mathcal{X}^{k+1} = \{F^{(k+1)}(x) \mid x \in \mathcal{X}^0\}$, we can directly over-approximate $\mathcal{X}^{k+1}$ by $\bar{\mathcal{X}}^{k+1} = \text{Poly}(C^{k+1}, d^{k+1})$, where d^{k+1} is now obtained by solving

$$d_i^{k+1} = \sup\{c_i^{k+1\top} F^{(k+1)}(x) \mid x \in \mathcal{X}^0\} = \sup\{c_i^{k+1\top} F_{eq}^{(k+1)}(x) \mid x \in \mathcal{X}^0\} \; i = 1, \cdots, m_{k+1}. \quad (7)$$

Thus we have converted the unrolled dynamics $F^{(k+1)}$, which is essentially a neural network with skip connections, to an equivalent neural network $F_{eq}^{(k+1)}$ without any skip connections.

Based on the assumption that $\mathcal{X}^0$ is a hyperrectangle, we propose to solve each of the optimization problems in (7) using a BnB method based on recursive partitioning of $\mathcal{X}^0$, and bounding the optimal value d_i^{k+1} efficiently over the partition. To be precise, given any sub-rectangle $\mathcal{X} \subset \mathcal{X}^0$, the bounding subroutine produces lower $\underline{d}_i(\mathcal{X})$ and upper bounds $\bar{d}_i(\mathcal{X})$ such that $\underline{d}_i(\mathcal{X}) \leq \sup\{c_i^{k+1\top} F_{eq}^{(k+1)}(x) \mid x \in \mathcal{X}\} \leq \bar{d}_i(\mathcal{X})$. Now, given a rectangular partition of the input set $\mathcal{X}^0$ and the corresponding bounds at each iteration of the BnB algorithm, we can update the

lower (respectively, upper) bound on d_i^{k+1} by taking the minimum of the lower (respectively, upper) bounds over the partition of $\mathcal{X}^0$. For the next iteration, the partition is refined non-uniformly based on a criterion, and those sub-rectangles that cannot contain the global solution are pruned. Overall, the algorithm produces a sequence of non-decreasing lower and non-increasing upper bounds on the objective value, and under mild conditions, it terminates in a finite number of iterations with a certificate of ϵ-suboptimality (see Boyd and Mattingley (2007) for an overview of BnB methods).

The above framework in the context of the problem presented in this paper has been developed recently in Entesari et al. (2022), in which the bounds over the partition are computed by utilizing upper bounds on the Lipschitz constant of the objective function (here $c_i^{k+1\top} F_{eq}^{(k+1)}$). Although any bound propagation method can be incorporated into our framework (on account of the conversion in (7)), in this paper, we use the fast and scalable method of DeepPoly Singh et al. (2019) within the branching strategy of Entesari et al. (2022). DeepPoly can handle larger networks and consequently, longer time horizons in our case. For details of the branching strategy, see Entesari et al. (2022).

Remark 1 (Zonotope Initial Set) Throughout the paper, we focus on hyper-rectangle input sets $\mathcal{X}^0$. However, it is fairly simple to adapt our method to zonotope initial sets $\mathcal{X}^0 = \{x \mid x = Gz, \|z\|_\infty \leq 1\}$. To do so, we substitute $x = Gz$ in (2) and absorb G into the first linear layer of f, i.e., use $W^0 G$ in lieu of W^0 in (3). We then apply the BnB method on this modified network and the input set $\mathcal{Z} = \{z \mid \|z\|_\infty \leq 1\}$.

3.3. Adaptive Template Polytopes

As discussed previously, the choice of the template sets can have a crucial effect on the accuracy of computations. The advantage of using fixed template polyhedra is that geometric operations such as union or intersection can be performed more efficiently Althoff et al. (2021). However, choosing a flexible template a priori that can capture the shape of the reachable sets is challenging. In this subsection, we propose a method that can adapt the geometry of the template polyhedra to the shape of the reachable sets. We outline the building blocks of the method for a single ReLU plus affine layer. Extension to a full neural network is straightforward and is provided in Algorithm 1.

Affine Layers Consider an affine layer $y = Wx + b$, where $W \in \mathbb{R}^{n_1 \times n_0}, b \in \mathbb{R}^{n_1}$, and suppose $x \in \mathcal{X} = \mathrm{Poly}(C, d), C \in \mathbb{R}^{m \times n_0}$. The image of $\mathcal{X}$ under the affine layer is given by

$$\mathcal{Y} = W\mathcal{X} + b = \{y \in \mathbb{R}^{n_1} \mid y = Wx + b, Cx \leq d\}.$$

In principle, we can find an exact $\mathcal{H}$-representation of $\mathcal{Y}$ using quantifier elimination methods such as Fourier-Motzkin elimination or Ferrante and Rackoff's method Bradley and Manna (2007). However, these methods typically have doubly exponential complexity. For example, using Ferrante and Rackoff's method, the complexity is $O(2^{2^{pn_0}})$ for some fixed constant p. Here we propose a heuristic based on singular value decomposition of W to compute an efficient while good approximation of the shape of the output polytope $\mathcal{Y}$.

Assuming that $W \in \mathbb{R}^{n_1 \times n_0}$ has rank $r \leq \min(n_1, n_0)$, its singular value decomposition can be written as $W = U\Sigma V^\top = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$, where σ_i, u_i and v_i are the ith largest singular value, and the corresponding left and right singular vectors, respectively. We also note that the range space of W is given by $\mathrm{Span}\{u_1, \cdots, u_r\}$ and the null space is given by $\mathrm{Span}\{v_{r+1}, \cdots, v_{n_0}\}$:

$$Wx = \sum_{i=1}^{r}(\sigma_i v_i^\top x)u_i, \quad \text{and} \quad Wv_j = \sum_{i=1}^{r}\sigma_i u_i(v_i^\top v_j) = 0 \quad j = r+1, \cdots, n_0.$$
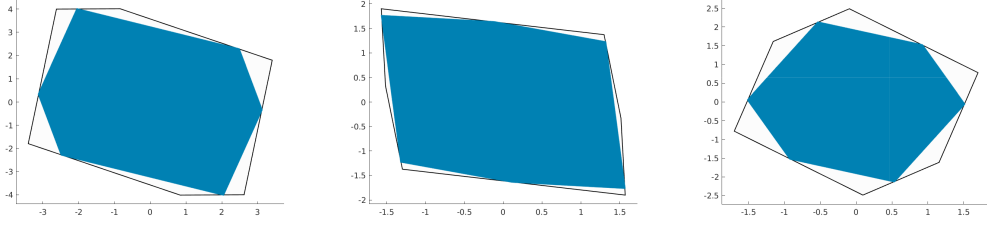
Figure 2: Examples of the polytopes acquired by solving the directions provided by the adaptive polytope method for affine layers with $n_1 = 2, n_0 = 3$. The blue set is the exact image of the transformation whilst the bounding polytope is acquired by solving linear programs $\min_{Cx \leq d} c^\top (Wx + b)$ for cs taken as the rows of $CW^\dagger$.

We distinguish between two cases:

Fat Weight Matrix ($n_1 \leq n_0$): Consider the system of equations $Wx + b = y$. Solving for x, with the knowledge that it is solvable, yields $x = W^\dagger(y-b) + F\zeta$, where the columns of $F \in \mathbb{R}^{n_0 \times (n_0 - r)}$ are the vectors $v_{r+1}, \cdots, v_{n_0}$, i.e., they span the null space of W, and $\zeta \in \mathbb{R}^{(n_0 - r)}$ is a free variable. This can be viewed as $x = \begin{bmatrix} W^\dagger & F \end{bmatrix} \begin{bmatrix} y \\ \zeta \end{bmatrix} - W^\dagger b$. Given $C \begin{bmatrix} W^\dagger & F \end{bmatrix} \begin{bmatrix} y \\ \zeta \end{bmatrix} - CW^\dagger b \leq d$, we are interested in bounding y. This is equivalent to projecting the polytope defined in the (y, ζ) space to the y space and bounding it. To do so, since ζ is a free variable, we absorb it into the constant term $CW^\dagger y \leq d + CW^\dagger b - CF\zeta = d'$. As a result, we propose to use the template matrix $\hat{C} = CW^\dagger$ and leave d' unspecified.

Tall Weight Matrix ($n_1 > n_0$): In this scenario, assuming W has full rank, the system of equations $Wx + b = y$ has a unique solution given by $x = W^\dagger(y - b)$. We now have

$$CW^\dagger(y - b) = Cx \leq d \Rightarrow CW^\dagger y \leq d + CW^\dagger b = d'.$$

As a result, we propose the same directions $CW^\dagger$ as before. Since $u_i^\top W = 0$, $i = n_0 + 1, \cdots, n_1$, we have $u_i^\top y = u_i^\top b$. This shows that y has zero variance in the direction of $u_i, i = n_0 + 1, \cdots, n_1$. At the same time, $CW^\dagger u_i = 0$. Thus, the directions in $CW^\dagger$ have no components in the space of $\mathrm{Span}\{u_{n_0+1}, \cdots, u_{n_1}\}$. Therefore, we simply add $\pm u_i$ for $i = n_0 + 1, \cdots, n_1$ to the list of directions, i.e., $\hat{C} = \begin{bmatrix} (CW^\dagger)^\top, u_{n_0+1}, -u_{n_0+1}, \ldots, u_{n_1}, -u_{n_1} \end{bmatrix}^\top$.

An interesting feature in this scenario is that the polytope defined by $\hat{C}y \leq d'$ is tight, i.e., all points y that reside in this polytope have a corresponding $x = W^\dagger(y - b)$ that satisfies $Cx \leq d$. To see this, it suffices to realize that for all y that $\hat{C}y \leq d'$, by the construction of $\hat{C}$, we know that y is in the range space of W shifted by b. Thus there exists a corresponding $x = W^\dagger(y - b)$. Now, $Cx = CW^\dagger(y - b) \leq d' - CW^\dagger b = d$. Thus, the polytope is tight and the directions of $\hat{C}$ exactly correspond to the directions of the polytope.

ReLU Layers Consider a ReLU layer $y = \mathrm{ReLU}(x)$, $x \in \mathbb{R}^n$, and suppose $x \in \mathcal{X} = \mathrm{Poly}(C, d)$, $C \in \mathbb{R}^{m \times n}$. Since $y \geq 0$, the ReLU activations might cut off the polytope. Thus we can improve the set of facets by adding the corresponding directions $\tilde{C} = \begin{bmatrix} C^\top, -e_1, \ldots, -e_n \end{bmatrix}^\top$, where e_i is the ith standard basis vector.

Next, we will prove that given a polytope, this proposed method will return a polytope.

Proposition 2 *Given a full-rank matrix W, and a polytope with facets given by C, the adaptive polytope method outlined in Algorithm 1 will yield a polytope, i.e., a bounded polyhedron.*

Proof As the ReLU layers only add a set of facets, they do not affect the bounded property of the polytope as it is already bounded. Thus, we will show that the affine layers preserve the bounded property. Suppose we have $y = Wx + b$ and $x \in \text{Poly}(C, d)$ is bounded. We consider two cases:

1. $n_1 \leq n_0$: If $CW^\dagger y \leq d'$ is unbounded, there exists a point y and direction v such that $y(\alpha) = y + \alpha v$ tends to infinity as α grows larger. The corresponding point in the x space is given by $x(\alpha) = W^\dagger(y + \alpha v - b) + F\zeta = \alpha W^\dagger v + W^\dagger(y - b) + F\zeta$. Since the input is bounded in a polytope, we must have $W^\dagger v = 0$. The null space of $W^\dagger$ is given by $\text{Span}\{u_{r+1}, \cdots, u_{n_1}\}$ where r is the rank of W. But this can not happen since W has full rank $r = n_1$.

2. $n_1 > n_0$: Similarly, since we add the directions not spanned by $CW^\dagger$ to the set of facets, the same argument will apply and the suggested directions define a bounded polytope. ∎

In summary, the method proposed above allows us to approximate the directions of the actual output set. Figure 2 portrays sample results of this algorithm on random single linear layers. The output directions $\hat{C}$ of adaptive polytopes on affine layers will have dimensions $(m + (n_1 - n_0)_+) \times n_1$. Having a consequent ReLU layer will add n_1 rows to $\hat{C}$. For an L layer neural network this would yield $m + \sum_{i=1}^{L-1} n_i + \sum_{i=1}^{L} (n_i - n_{i-1})_+$ facets. However, it is very likely that linear layers with $n_1 \leq n_0$ may produce directions that are similar. To avoid an explosion of the number of facets, we simply remove similar directions. That is, after the algorithm is run for the full neural network, if for two directions c_1 and c_2 the cosine similarity is larger than a certain threshold, i.e., $cos(\theta) = \frac{c_1^\top c_2}{||c_1||_2 ||c_2||_2} > \lambda$, we discard one of the directions.

Remark 3 Although it is possible to employ Algorithm 1 on the equivalent neural network defined by (4), we find, empirically, that it is better to use the algorithm to discern the set of directions

---

Algorithm 1 Adaptive Polytope Directions

Data: Input polytope template matrix C, piecewise linear sequential neural network weights $(W^0, \cdots, W^L)$, similarity tolerance level λ.
Result: Output polytope template matrix D.
$D \leftarrow C$
for $i = 0, \cdots, L$ **do**
    $n_1, n_0 = \text{shape}(W^i)$
    $U, \Sigma, V = \text{SVD}(W^i)$,   /* $U = [u_1, \cdots, u_{n_1}]$ */
    $D = D(W^i)^\dagger$
    if $n_1 > n_0$ **then**
        $D = [D^\top, u_{n_0+1}, -u_{n_0+1}, \cdots, u_{n_1}, -u_{n_1}]^\top$
    end
    $D = [D^\top, -e_1, \cdots, -e_{n_1}]^\top$,   /* $e_i \in \mathbb{R}^{n_1}$. After each layer, there is a ReLU nonlinearity.   */
end
Remove similar directions from D according to cosine similarity at tolerance level λ.
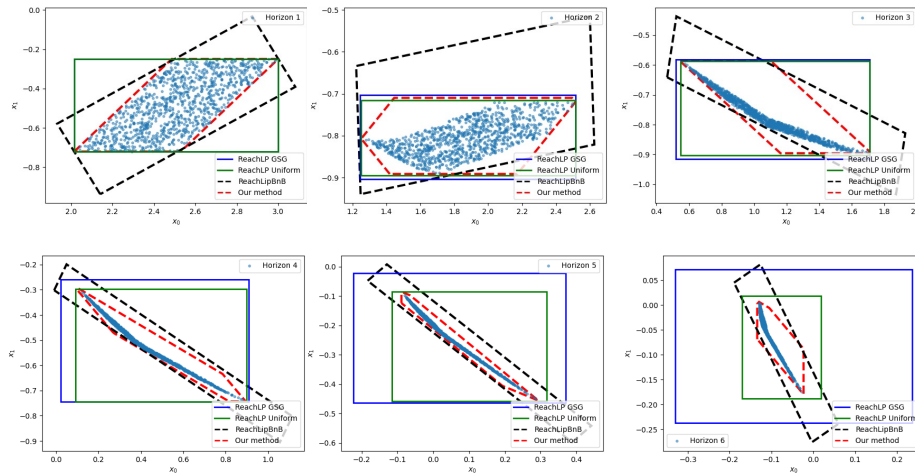**Return** D

Figure 3: Computation of reachable sets for the double integrator system.

$\hat{C}_1$ and $\hat{C}_2$ for $A\bar{\mathcal{X}}^k$ and $Bf(\bar{\mathcal{X}}^k)$, respectively, and then use $\hat{C}^\top = \begin{bmatrix} \hat{C}_1^\top & \hat{C}_2^\top \end{bmatrix}$ as the final set of directions for $x^{k+1} = Ax^k + Bf(x^k) + e$. To ensure the boundedness of the polytope, we require that A or B be full rank. If not, one can simply add the directions corresponding to axis-aligned hyperrectangles.

4. Experiments

In this section, we present the results of our algorithm on two tasks and compare our results with the BnB method of Entesari et al. (2022) and the ReachLP method of Everett et al. (2021). The experiments are conducted on an Intel Xeon W-2245 3.9 GHz processor with 64 GB of RAM. For specifications of the BnB algorithm, see Entesari et al. (2022). For each work, we use the template polytope directions that the original work has proposed. For Everett et al. (2021), we use their CROWN propagator and test their results with both the uniform and the greedy simulation guidance (GSG) partitioners. We use a $\lambda = 0.98$ for the cosine similarity threshold.

4.1. Double Integrator

We first consider the two-dimensional discrete-time system from Hu et al. (2020), for which we have $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$, and $e = 0$ in (2). We use a trained neural network that approximates the MPC controller. The neural network has the following number of neurons in each layer: (2-10-5-1) (starting from the input layer). The comparison is presented in Fig. 3. For this example, we use the initial set given by $\mathcal{X}^0 = [2.5, 3] \times [-0.25, 0.25]$. As the figure suggests, our method provides considerably less conservative over-approximations, especially for longer time horizons. Entry DI in Table 1 represents the results of the experiment. For this system, with our trained network, our method solves a total of 74 optimization problems, whereas the other methods solve 4 problems per time step. We used an epsilon solve accuracy of 0.01 for ReachLipBnB.
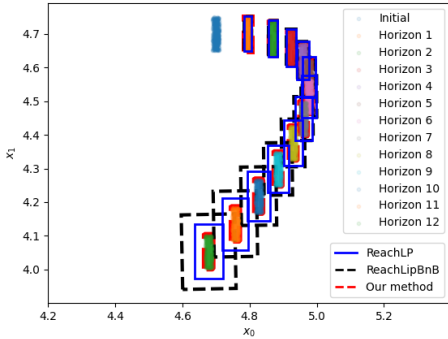
9

Figure 4: Reachable sets of the first two state variables of the quadrotor system. The reachable sets of our method are tight in this experimental setup.

		ReachLipBnB	ReachLP GSG	ReachLP uniform	Our method
DI	Run time [s]	2.45	1.078	5.95	9.96
	RT/ND	0.102	0.045	0.248	0.134
QR	Runt time [s]	2170	18.8	-	213.67
	RT/ND	11.3	0.13	-	0.501

Table 1: Run time statistics for experiments. RT/ND is the total run time of the experiment divided by the total number of optimization problems (or directions) solved.

4.2. 6D Quadrotor

We consider a quadrotor example with 6 state variables. This system is also taken from Hu et al. (2020). For this system, we have $A = I_{6\times6} + \Delta t \times \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix}$, $B = \Delta t \times \begin{bmatrix} & g & 0 & 0 \\ 0_{3\times3} & 0 & -g & 0 \\ & 0 & 0 & 1 \end{bmatrix}^\top$, $u = \begin{bmatrix} \tan(\theta) & \tan(\phi) & \tau \end{bmatrix}^\top$ and $e^\top = \Delta t \times \begin{bmatrix} 0_{5\times1} & -g^\top \end{bmatrix}$ in (2). We train a neural network to approximate an MPC controller. The neural network has 3 linear layers with 32, 32, and 3 neurons, respectively. As the original system is continuous time, we have discretized it with a sampling time of $\Delta t = 0.1$ seconds. The reachability analysis is conducted for 12 time steps. For this example, we let the initial set be defined by $[4.69, 4.71] \times [4.65, 4.75] \times [2.975, 3.025] \times [0.9499, 0.9501] \times [-0.0001, 0.0001] \times [-0.0001, 0.0001]$. For this example, Fig. 4 shows that even after 12 time steps, our method provides tight reachable sets. Entry QR in Table 1 represents the results of the experiment. Our method solves a total of 426 optimization problems, whereas ReachLP and ReachLipBnB solve 12 and 16 problems per time step, respectively. The uniform partitioner for ReachLP timed out after 27 hours. We used an epsilon solve accuracy of 0.001 for ReachLipBnB.

5. Conclusion

We presented a novel method for reachability analysis of affine systems in feedback with ReLU neural network controllers using dynamic template polytopes. We then computed tight polyhedral over-approximation of the reachable sets using a BnB method based on partitioning the set of initial states and unrolling the dynamics. The bounding mechanism of DeepPoly provides fast estimates that scale up to large and deep neural networks; however, the application of our framework for higher dimensional input sets might be restrictive due to the worst-case exponential growth in partitioning the input set. To counteract this, we will explore less conservative bound propagation methods such as Fatnassi et al. (2023), combined with branching in the activation space (as opposed to the state space). Furthermore, we will explore operator splitting methods similar to Chen et al. (2022b) to further improve scalability with respect to long time horizons.

10

References

Matthias Althoff, Goran Frehse, and Antoine Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1), 2021.

Mohamed Amin Ben Sassi, Romain Testylier, Thao Dang, and Antoine Girard. Reachability analysis of polynomial systems using linear programming relaxations. In *International Symposium on Automated Technology for Verification and Analysis*, pages 137–151. Springer, 2012.

Sergiy Bogomolov, Goran Frehse, Mirco Giacobbe, and Thomas A Henzinger. Counterexample-guided refinement of template polyhedra. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 589–606. Springer, 2017.

Stephen Boyd and Jacob Mattingley. Branch and bound methods. *Notes for EE364b, Stanford University*, pages 2006–07, 2007.

Aaron R Bradley and Zohar Manna. *The calculus of computation: decision procedures with applications to verification*. Springer Science & Business Media, 2007.

Rudy Bunel, Alessandro De Palma, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip Torr, and M Pawan Kumar. Lagrangian decomposition for neural network verification. In *Conference on Uncertainty in Artificial Intelligence*, pages 370–379. PMLR, 2020.

Shaoru Chen, Victor M Preciado, and Mahyar Fazlyab. One-shot reachability analysis of neural network dynamical systems. *arXiv preprint arXiv:2209.11827*, 2022a.

Shaoru Chen, Eric Wong, J Zico Kolter, and Mahyar Fazlyab. Deepsplit: Scalable verification of deep neural networks via operator splitting. *IEEE Open Journal of Control Systems*, 1:126–140, 2022b.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 251–268. Springer, 2017.

Arthur Clavière, Eric Asselin, Christophe Garion, and Claire Pagetti. Safety verification of neural network controlled systems. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 47–54. IEEE, 2021.

Alessandro De Palma, Rudy Bunel, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip HS Torr, and M Pawan Kumar. Improved branch and bound for neural network verification via lagrangian decomposition. *arXiv preprint arXiv:2104.06718*, 2021.

Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Learning and verification of feedback control systems using feedforward neural networks. *IFAC-PapersOnLine*, 51 (16):151–156, 2018a.

Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*, pages 121–138. Springer, 2018b.

Souradeep Dutta, Xin Chen, and Sriram Sankaranarayanan. Reachability analysis for neural feed-back systems using regressive polynomial rule inference. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 157–168, 2019.

Taha Entesari, Sina Sharifi, and Mahyar Fazlyab. Reachlipbnb: A branch-and-bound method for reachability analysis of neural autonomous systems using lipschitz bounds. *arXiv preprint arXiv:2211.00608*, 2022.

Michael Everett, Golnaz Habibi, Chuangchuang Sun, and Jonathan P How. Reachability analysis of neural feedback loops. *IEEE Access*, 9:163938–163953, 2021.

Wael Fatnassi, Haitham Khedr, Valen Yamamoto, and Yasser Shoukry. Bern-nn: Tight bound propagation for neural networks using bernstein polynomial interval arithmetic. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2023.

Claudio Ferrari, Mark Niklas Muller, Nikola Jovanovic, and Martin Vechev. Complete verification via multi-neuron relaxation guided branch-and-bound. *arXiv preprint arXiv:2205.00263*, 2022.

Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

Haimin Hu, Mahyar Fazlyab, Manfred Morari, and George 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)*, pages 5929–5934. IEEE, 2020.

Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–22, 2019.

Radoslav Ivanov, Taylor Carpenter, James Weimer, Rajeev Alur, George Pappas, and Insup Lee. Verisig 2.0: Verification of neural network controllers using taylor model preconditioning. In *International Conference on Computer Aided Verification*, pages 249–262. Springer, 2021.

Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. Open-and closed-loop neural network verification using polynomial zonotopes. *arXiv preprint arXiv:2207.02715*, 2022.

Panagiotis Kouvaros and Alessio Lomuscio. Towards scalable complete verification of relu neural networks via dependency-based branching. In *IJCAI*, pages 2643–2650, 2021.

Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint arXiv:1706.07351*, 2017.

Arnold Neumaier. The wrapping effect, ellipsoid arithmetic, stability and confidence regions. In *Validation numerics*, pages 175–190. Springer, 1993.

Nicholas Rober, Sydney M Katz, Chelsea Sidrane, Esen Yel, Michael Everett, Mykel J Kochenderfer, and Jonathan P How. Backward reachability analysis of neural feedback loops: Techniques for linear and nonlinear systems. *arXiv preprint arXiv:2209.14076*, 2022.

Christian Schilling, Marcelo Forets, and Sebastián Guadalupe. Verification of neural-network control systems by integrating taylor models and zonotopes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8169–8177, 2022.

Chelsea Sidrane, Amir Maleki, Ahmed Irfan, and Mykel J Kochenderfer. Overt: An algorithm for safety verification of neural network control policies for nonlinear systems. *Journal of Machine Learning Research*, 23(117):1–45, 2022.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.

Olaf Stursberg and Bruce H Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 482–497. Springer, 2003.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Hoang-Dung Tran, Diago Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T Johnson. Star-based reachability analysis of deep neural networks. In *International symposium on formal methods*, pages 670–686. Springer, 2019.

Joseph A Vincent and Mac Schwager. Reachable polyhedral marching (rpm): A safety verification algorithm for robotic systems with deep neural network components. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9029–9035. IEEE, 2021.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. *arXiv preprint arXiv:2011.13824*, 2020.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.

Yuhao Zhang and Xiangru Xu. Reachability analysis and safety verification of neural feedback systems via hybrid zonotopes. *arXiv preprint arXiv:2210.03244*, 2022.