Provably Bounding Neural Network Preimages

Suhas Kotha[1*], Christopher Brix[2*], Zico Kolter[1,4],
Krishnamurthy (Dj) Dvijotham[3†], Huan Zhang[1†]

[1]Carnegie Mellon University, Pittsburgh PA, 15213, USA.
[2]RWTH Aachen University, Aachen, 52056, Germany.
[3]Google Research, Brain Team.
[4]Bosch Center for AI.

*Corresponding authors. E-mails: suhask@andrew.cmu.edu; brix@cs.rwth-aachen.de;
†These authors contributed equally to this work.

Abstract

Many safety-critical applications require the computation of preimages of neural networks for a given output. We present INVPROP, an algorithm to compute a convex overapproximation of these preimages that does not rely on the use of LP solvers and can be executed using GPUs. The experimental evaluation demonstrates that some computed overapproximations are over **2500**× tighter and **2.5**× faster than prior work.

1 Introduction

In safety-critical applications, it is often of interest to determine the set of inputs that produces a given set of outputs. Examples include autonomous robots that should reach a target area ("What location must the robot be in, to reach the target in n timesteps?") or out-of-distribution (OOD) detection ("What inputs will cause an OOD-alert?") Since neural networks are not invertible, computing the required exact input set may be computationally infeasible. Instead, we propose a technique to compute a convex hull of the input set in a highly efficient, GPU-supported fashion. Importantly, this separates our approach from state-of-the-art techniques that rely on LP solvers. We demonstrate significant speed and precision improvements over the state-of-the-art [1, 2] using our technique.

2 Problem Statement

For a given network $f : \mathcal{X} \subseteq \mathbb{R}^{\text{in}} \to \mathbb{R}^{\text{out}}$ and an output set $\mathcal{S}_{\text{out}} \subseteq \mathbb{R}^{\text{out}}$, we need to compute a convex overapproximation of $f^{-1}(\mathcal{S}_{\text{out}}) \subseteq \mathcal{X}$. We assume that the output set can be described by linear constraints: $\mathbf{H}f(\boldsymbol{x}) + \mathbf{d} \leq 0$.

The convex hull of $f^{-1}(\mathcal{S}_{\text{out}})$ can be described as $\bigcap_{\boldsymbol{c}\in\mathbb{R}^{\text{in}}}\{\boldsymbol{x} : \boldsymbol{c}^\top \boldsymbol{x} \geq \min_{f(\boldsymbol{x'})\in\mathcal{S}_{\text{out}}} \boldsymbol{c}^\top \boldsymbol{x'}\}$, As an approximation, a finite set of $\boldsymbol{c}$ can be used. Thus, we need to solve several instances of the problem

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^\top \boldsymbol{x} \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{x} \in \mathcal{X}; \quad \mathbf{H}f(\boldsymbol{x}) + \mathbf{d} \leq 0 \tag{1b}$$

3 INVPROP

The constraint $\mathbf{H}f(\boldsymbol{x}) + \mathbf{d} \leq 0$ can be moved into the optimization objective by introducing a dual variable γ. Inverting the order of min and max, we get a lower bound:

$$\max_{\boldsymbol{\gamma}} \min_{\boldsymbol{x}} \quad \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\gamma}^\top (\mathbf{H}f(\boldsymbol{x}) + \mathbf{d})$$
$$\text{s.t.} \quad \boldsymbol{x} \in \mathcal{X}; \quad \boldsymbol{\gamma} \geq 0$$

The inner minimization is only constrained by $\boldsymbol{x} \in \mathcal{X}$, so a lower bound can be efficiently computed by Auto-LiRPA [3]. The result is a relaxed expression with additional parameters $\boldsymbol{\alpha} \in [0, 1]$:

$$\max_{\boldsymbol{\gamma}} \quad \text{AutoLiRPA}(\boldsymbol{\alpha}, \boldsymbol{\gamma})$$
$$\text{s.t.} \quad \boldsymbol{x} \in \mathcal{X}; \quad \boldsymbol{\gamma} \geq 0; \quad 0 \leq \boldsymbol{\alpha} \leq 1$$

We note that each instantiation of $\boldsymbol{\gamma} \geq 0$ yields a valid lower bound. Both λ and α can then be optimized using projected gradient ascent using PyTorch, which iteratively tightens the lower bound of the original problem (1).

A similar optimization problem can be defined to compute bounds on the values of neurons in intermediate neurons. Tighter bounds on those neurons allow for a more precise relaxation in Auto-LiRPA, further improving the bounds on problem (1).

4 Results

4.1 Backward Reachability Analysis for Neural Feedback Loops

Prior work [1, 2] defines a benchmark where the objective is to compute states that a robot must not be in in order to avoid an obstacle. The next position is determined by a three layer network with 12, 7, and 2 neurons. Multiple time steps can be modeled by composing this function. We leverage the fact that bounds computed for time step t can be reused for time step $t+1$.

The SOTA method suffers from increasingly weak bounds for longer time horizons, as is evident in Figure 1. Their implementation cannot tighten the bounds of neurons in intermediate layers with respect to the output set, which is possible with our method. By tightening those bounds, we can in turn improve the bounds on the input layer.

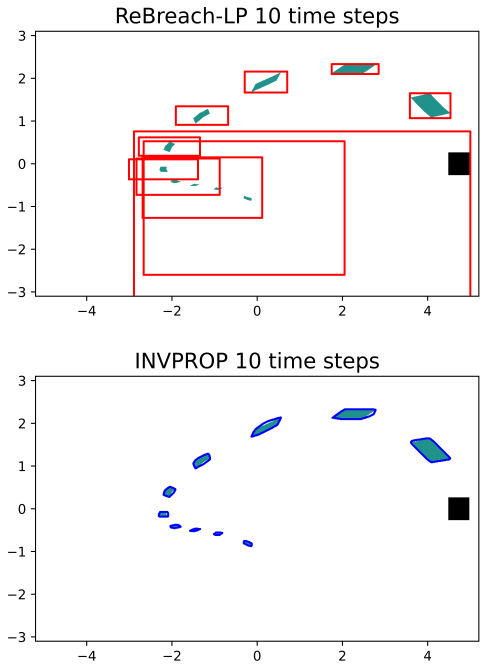

Fig. 1 Black: Obstacle; green: approximated states that would result in an collision within 10 time steps. Top: Prior work (ReBreach-LP) has increasingly weak bounds for larger time horizons (42.86s for time step 10); bottom: INVPROP is almost perfectly tight even for ten time steps (17.89s for time step 10).

4.2 OOD Detection

For a dense feed-forward network with 200, 200 and 3 neurons, that was trained to predict label 2 for every OOD input, we define the set of in-domain data as $\max\{y_0, y_1\} > y_2$, pictured in green in Figure 2. For our model, this set is non-convex, leading to a weak approximation if a convex hull is used. However, using input space branching, this non-convexity can be removed: Each quadrant of the input space allows for a precise convex hull of the target set. Therefore, non-convexity can easily be handled by INVPROP.

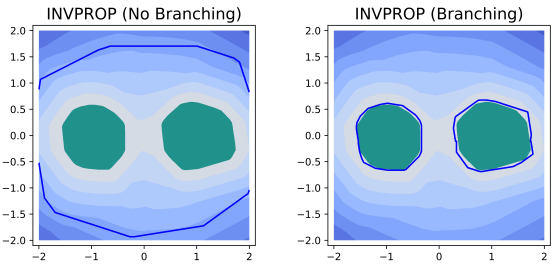

Fig. 2 Green: Approximated ID-input space (1 million samples); blue border: Approximation using INVPROP. Left: One convex hull (weak); right: union of four convex hulls, one per quadrant (strong)

References

[1] Rober, N., Everett, M., Zhang, S., How, J.P.: A hybrid partitioning strategy for backward reachability of neural feedback loops. arXiv preprint arXiv:2210.07918 (2022)

[2] Rober, N., Katz, S.M., Sidrane, C., Yel, E., Everett, M., Kochenderfer, M.J., How, J.P.: Backward reachability analysis of neural feedback loops: Techniques for linear and nonlinear systems. arXiv preprint arXiv:2209.14076 (2022)

[3] Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.-W., Huang, M., Kailkhura, B., Lin, X., Hsieh, C.-J.: Automatic perturbation analysis for scalable certified robustness and beyond. Advances in Neural Information Processing Systems **33** (2020)