Neural Networks Verification as Piecewise Linear Optimization

Tu Anh-Nguyen   Joey Huchette

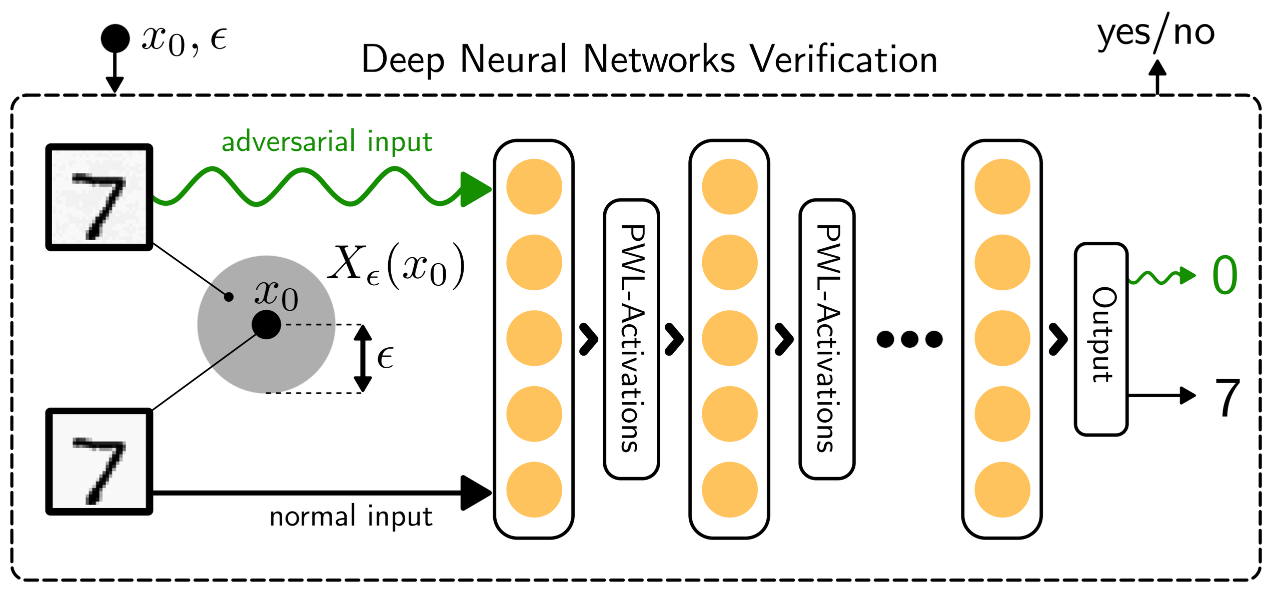
RICE

DIMACS
Rutgers University
May 23-26, 2022

Abstract

In this work, we provide a strong (ideal) MIP formulation for the neural network verification task, which extends the work of Anderson et al. [1] Our contributions are summarized as follows.

- We derive a polynomial algorithm to obtain facet-defining hyperplane for the Cayley embedding of the graph of activation functions.
- Empirically, our formulation is shown to give tighter LP relaxations for relaxed verifiers and improves the performance of exact verifiers.



MIP Formulation of NN Verification using Cayley Embedding

output of a neural net

$$\text{Is} \max_{x \in X_\epsilon(x_0)} c \cdot \boxed{M(x)} \leq \xi?$$

expanded form

Cayley Embedding

$$\max\ c_1 y_1 + \cdots + c_{n_2} y_{n_2}$$
$$(x_1, \ldots, x_i) \in \mathrm{gr}(g_i(x_1, \ldots, x_{i-1})) \quad \forall i \in \{n_1+1, \ldots, N\}$$
$$(x_1, \ldots, x_{n_1}) \in X_\epsilon(x_0)$$
$$y = (x_{N+1-n_2}, \ldots, x_N),$$

non-linear constraints

- The convex hull of the Cayley embedding is infact a polytope with an exponential number of faces. Hence, we need to derive an efficient separation procedure.

Lemma 1. *Given $(\hat{x}, \hat{y}, \hat{z})$, if the optimal value of the following problem is greater than $\hat{y}$ then $(\hat{x}, \hat{y}, \hat{z})$ is feasible, otherwise, the optimal solution α^* corresponds to a hyperplane that cuts off $(\hat{x}, \hat{y}, \hat{z})$*

$$\text{min} \quad s\textstyle\sum_{i=1}^{k} z_i\left(\sum_{j=1}^{n} u_j|w_j|\bar{\beta}_j^i - \sum_{j=1}^{n} l_j|w_j|\bar{\gamma}_j^i + (h_i - b)\bar{\theta}_1^i - (h_{i-1} - b)\bar{\theta}_2^i\right) + s\sum_{j=1}^{n} x_j|w_j|\bar{\alpha}_j$$

$$\text{subject to} \quad \begin{bmatrix} A & 0 & \ldots & 0 & I_n \\ 0 & A & \ldots & 0 & I_n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & A & I_n \end{bmatrix} \begin{bmatrix} \bar{\beta}^1 \\ \bar{\gamma}^1 \\ \bar{\theta}^1 \\ \vdots \\ \bar{\beta}^k \\ \bar{\gamma}^k \\ \bar{\theta}^k \\ \bar{\alpha} \end{bmatrix} = \begin{bmatrix} \frac{a_1}{s}\bar{w} \\ \frac{a_2}{s}\bar{w} \\ \frac{a_3}{s}\bar{w} \\ \vdots \\ \frac{a_k}{s}\bar{w} \end{bmatrix}, \text{ and } \begin{bmatrix} \bar{\beta}^1 \\ \bar{\gamma}^1 \\ \bar{\theta}^1 \\ \vdots \\ \bar{\beta}^k \\ \bar{\gamma}^k \\ \bar{\theta}^k \end{bmatrix} \geq \mathbf{0}.$$

Theorem 1. *The separation procedure can be done in $O(n\log(n + \max(k, n))$ time complexity*

Motivating Example: Staircase Function

A univariate piecewise linear function $f : \mathbb{R} \to \mathbb{R}$ with k pieces is a staircase function if there exists $s \in \mathbb{R}$ such that every pieces' slope $a_i \in \{0, s\}$.

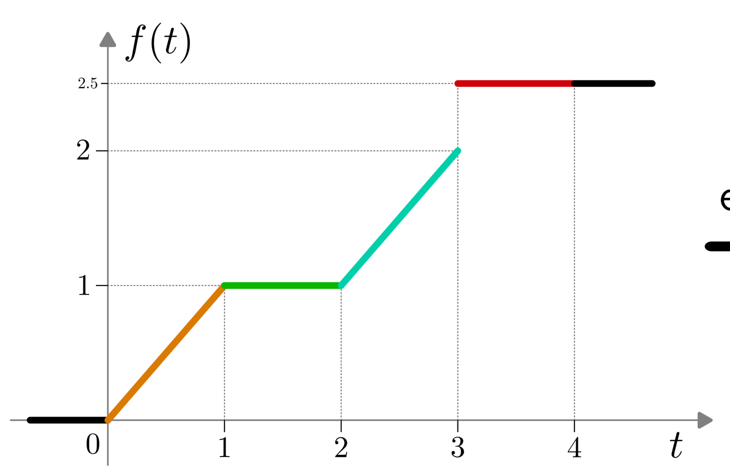


Fig. 1: 1D Staircase Function

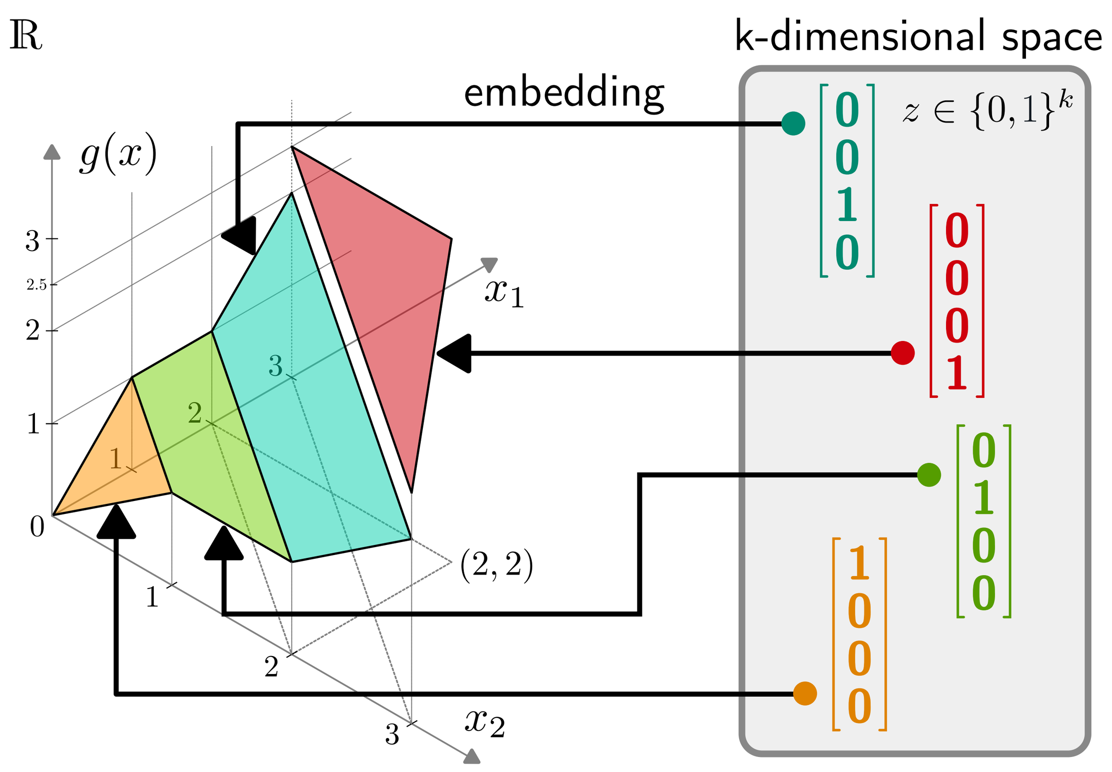


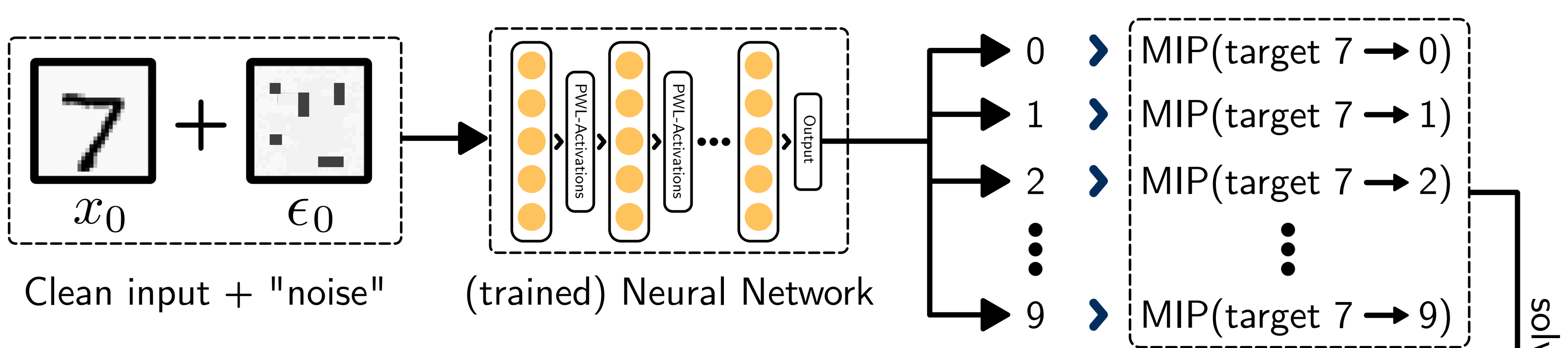
Fig. 2: 2D Staircase Function and Cayley Embeddings

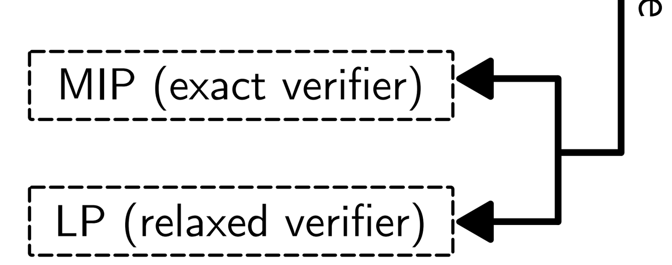
A piecewise linear function $f : \mathbb{R}^n \to \mathbb{R}$ is a k-piece staircase function if $f = g(w \cdot x)$ where g is a univariate staircase function.

Let $D^i := \{x \in D \mid h_{i-1} \leq w \cdot x + b \leq h_i\}$, the Cayley Embedding [2] for the closure of graph of f is:

$$S_{\text{Cayley}}(f) := \textstyle\bigcup_{i=1}^{k} \{(x, y, z) \mid x \in D^i,\ y = f(x),\ z = e^i\}$$

Neural Networks Verification Procedure





References

[1] Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. *Strong Mixed-Integer Programming Formulations for Trained Neural Networks.* Mathematical Programming, pages 1–37, 2020.

[2] Juan Pablo Vielma. *Embedding Formulations and Complexity for Unions of Polyhedra.* Management Science, 64(10):4721–4734, 2018.

Composition of Staircase Functions

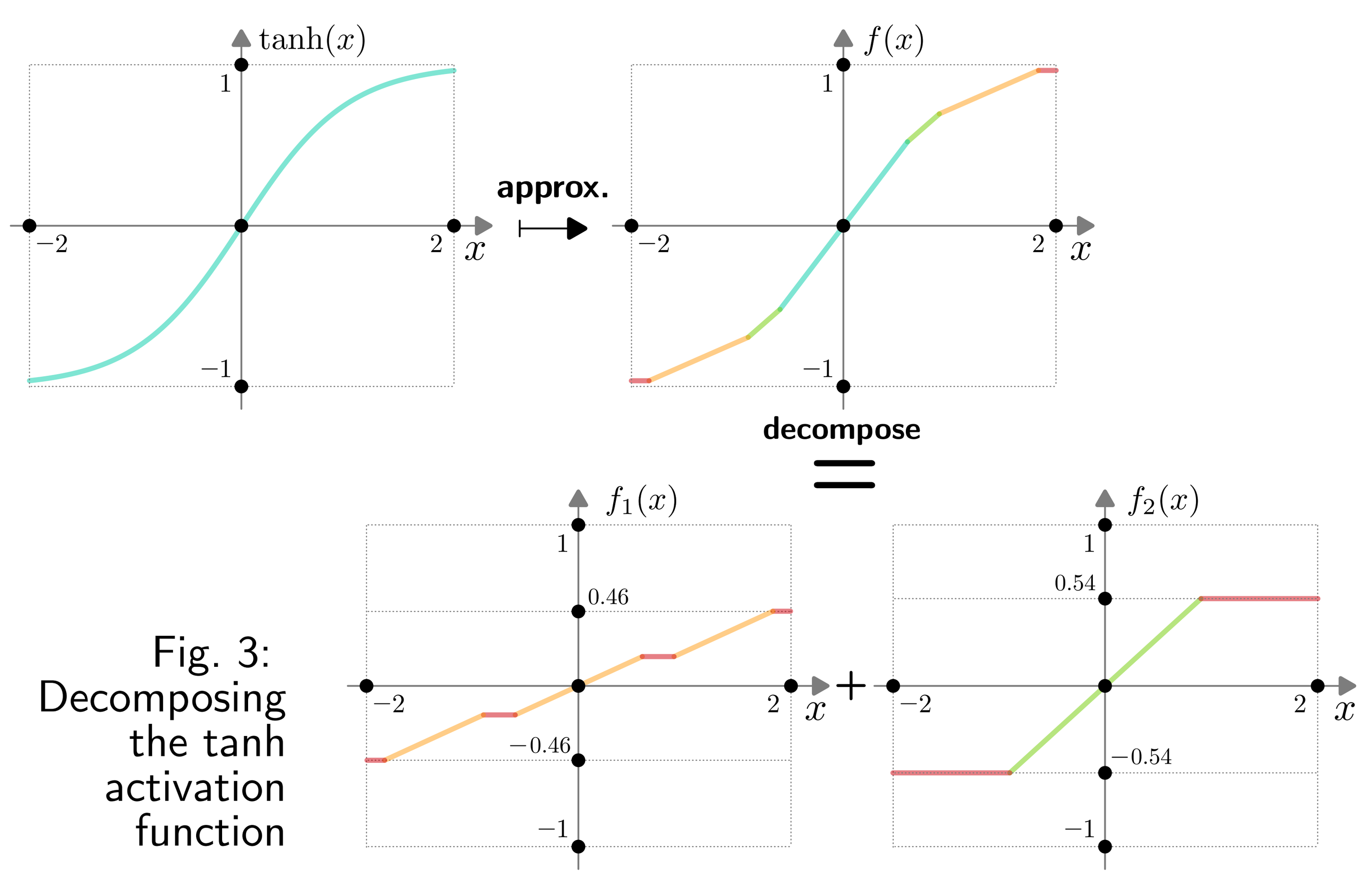


Fig. 3: Decomposing the tanh activation function

Lemma 2. *Let $g = g_1 + \cdots + g_m$ where $g_1, \ldots, g_k$ are staircase functions, then conv$(C(g))$ is the solutions of the following system*

$$y \leq \min_{\bar{\alpha}_1 \in \mathbb{R}^n} \left(\alpha_1 \cdot x + \sum_{i=1}^{k} \left(\max_{x^i \in D^i} (a_i^1 w - \alpha_1) \cdot x^i + b_i\right) z_i\right) + \cdots +$$
$$\min_{\bar{\alpha}_m \in \mathbb{R}^n} \left(\alpha_m \cdot x + \sum_{i=1}^{k} \left(\max_{x^i \in D^i} (a_i^m w - \alpha_m) \cdot x^i + b_i\right) z_i\right)$$
$$y \geq \max_{\underline{\alpha}_1 \in \mathbb{R}^n} \left(\underline{\alpha}_1 \cdot x + \sum_{i=1}^{k} \left(\min_{x^i \in D^i} (a_i^1 w - \underline{\alpha}_1) \cdot x^i + b_i\right) z_i\right) + \cdots +$$
$$\max_{\underline{\alpha}_m \in \mathbb{R}^n} \left(\underline{\alpha}_m \cdot x + \sum_{i=1}^{k} \left(\min_{x^i \in D^i} (a_i^m w - \underline{\alpha}_m) \cdot x^i + b_i\right) z_i\right)$$
$$(x, y, z) \in D \times \mathbb{R} \times \Delta^k.$$

Experimental Results

Table 1: Relaxed Verifiers

NN Arch.	ϵ	DeepPoly		Big-M Formulation		Cayley Emb. Formulation	
		#Verified	Time (s)	#Verified	Time (s)	#Verified	Time (s)
Dense 2 x 256 Dorefa 2	0.008	118	0.338 ± 0.056	**138**	1.060 ± 0.005	**138**	1.100 ± 0.008
	0.016	59	0.338 ± 0.058	112	1.056 ± 0.006	**113**	1.129 ± 0.086
	0.024	19	0.336 ± 0.055	65	1.075 ± 0.004	**66**	1.139 ± 0.078
	0.032	0	0.326 ± 0.054	28	1.080 ± 0.006	**29**	1.174 ± 0.086
Dense 2 x 256 Dorefa 3	0.008	132	0.339 ± 0.059	**142**	1.056 ± 0.005	**142**	1.102 ± 0.075
	0.016	87	0.340 ± 0.059	125	1.058 ± 0.005	**125**	1.120 ± 0.070
	0.024	11	0.341 ± 0.058	90	1.078 ± 0.005	**91**	1.169 ± 0.079
	0.032	0	0.324 ± 0.052	27	1.080 ± 0.006	**29**	1.210 ± 0.090
Dense 2 x 256 Dorefa 4	0.008	132	0.329 ± 0.055	143	1.082 ± 0.005	**144**	1.113 ± 0.082
	0.016	78	0.329 ± 0.056	126	1.063 ± 0.006	**126**	1.134 ± 0.072
	0.024	6	0.330 ± 0.056	86	1.071 ± 0.006	**90**	1.178 ± 0.086
	0.032	0	0.331 ± 0.056	25	1.100 ± 0.006	**34**	1.286 ± 0.160
Dense 2 x 256 Dorefa 5	0.008	140	0.329 ± 0.056	**143**	1.060 ± 0.006	**143**	1.130 ± 0.083
	0.016	78	0.332 ± 0.056	138	1.087 ± 0.005	**140**	1.169 ± 0.078
	0.024	4	0.331 ± 0.056	98	1.107 ± 0.007	**100**	1.256 ± 0.113
	0.032	1	0.328 ± 0.056	33	1.144 ± 0.007	**44**	1.409 ± 0.190

Table 2: Exact Verifier using Cayley Embedding

NN Arch.	ϵ	Cayley Embedding Formulation			
		#Nodes	Gap (%)	Gurobi Time (s)	User Callbacks (s)
Dorefa 2	0.008	2984.4 ± 1590.1	**0.00**	2.84 ± 0.66	1.14 ± 0.4
Dorefa 3		53277.0 ± 18666.20	**4.19 ± 1.74**	Timeout	17.73 ± 5.12
Dorefa 4		33248.4 ± 268.06	**4.28 ± 1.06**	Timeout	14.09 ± 0.32
Dorefa 2	0.016	45925.4 ± 17338.72	**11.57 ± 5.70**	Timeout	16.51 ± 6.52
Dorefa 3		33406.3 ± 639.79	**12.33 ± 6.09**	Timeout	14.46 ± 0.35
Dorefa 4		42701.2 ± 20587.1	**9.34 ± 6.22**	Timeout	19.63 ± 9.63

Table 3: Exact Verifier using Big-M

NN Arch.	ϵ	Big-M Formulation		
		#Nodes	Gap (%)	Solve Time (s)
Dorefa 2	0.008	3925.5 ± 2326.01	**0.00**	3.11 ± 0.87
Dorefa 3		51285.8 ± 20756.89	5.89 ± 4.37	Timeout
Dorefa 4		33063.8 ± 607.23	4.46 ± 1.64	Timeout
Dorefa 2	0.016	33340.6 ± 427.03	13.09 ± 4.90	Timeout
Dorefa 3		33224.5 ± 317.93	12.48 ± 5.08	Timeout
Dorefa 4		33091.6 ± 406.6	11.41 ± 7.90	Timeout

All neural networks is trained using the quantized network training open-source package Larq. The activation Dorefa κ is a constant piecewise function with 2^κ pieces.

Theoretical Section *Experimental Section*

Contact: tu.na@rice.edu | tu-na.org