Optimal decision-making with trained NN embedded

Shiqiang Zhang, Juan S Campos, Christopher Hojny, Francesco Ceccon, Jordan Jalving, Joshua Haddad, Alexander Thebelt, Calvin Tsay, Carl D Laird, Ruth Misener

Funding EPSRC EP/P016871/1 & EP/T001577/1
Sandia LDRD program
Institute for the Design of Advanced Energy Systems

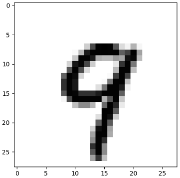
13 September 2024

Papers

1. Botoeva, Kouvaros, Kronqvist, Lomuscio, Misener, *AAAI*, 2000.
2. Kronqvist, Misener, Tsay, *CPAIOR*, 2021. **Best Paper**
3. Tsay, Kronqvist, Thebelt, Misener, *NeurIPS*, 2021.
4. Ceccon*, Jalving*, Haddad, Thebelt, Tsay, Laird†, Misener†, *JMLR MLOSS*, 2022.
5. Zhang, Campos, Feldmann, Walz, Sandfort, Mathea, Tsay, Misener, *NeurIPS*, 2023.
6. Hojny*, Zhang*, Campos, Misener, *ICML*, 2024.

Optimization challenges to analyze trained neural networks

Example: Classification of MNIST digits [Tsay et al., 2021]



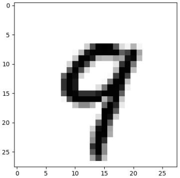
Given . . . .

Trained NN	
Image	$\bar{\boldsymbol{x}}$
Label	$j = 9$
Adversary?	$k = 4$

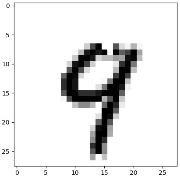
- **Verification** *[Feasibility]* Is there an adversary labeled k within a given perturbation (e.g., by ℓ_1- or ℓ_∞-norm)?
- **Optimal adversary** [Anderson et al., 2020] What image within a perturbation radius maximizes the prediction difference?
- **Minimally distorted adversary** [Croce and Hein, 2020] Smallest perturbation over which NN can predict adversarial label k?
- **Lossless compression** [Serra et al., 2020] Can I safely remove NN nodes or layers?

Optimization challenges to analyze trained neural networks

Example: Classification of MNIST digits [Tsay et al., 2021]



↓ ℓ_1



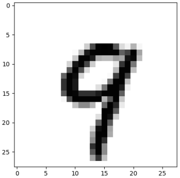
$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_1 = 4$

Given	
Trained NN	
Image	$\bar{\boldsymbol{x}}$
Label	$j = 9$
Adversary?	$k = 4$

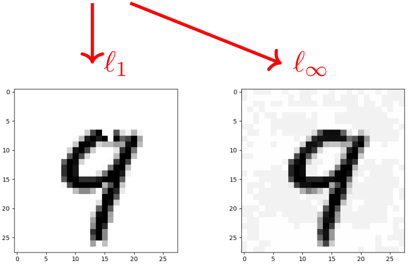
- **Verification** *[Feasibility]* Is there an adversary labeled k within a given perturbation (e.g., by ℓ_1- or ℓ_∞-norm)?
- **Optimal adversary** [Anderson et al., 2020] What image within a perturbation radius maximizes the prediction difference?
- **Minimally distorted adversary** [Croce and Hein, 2020] Smallest perturbation over which NN can predict adversarial label k?
- **Lossless compression** [Serra et al., 2020] Can I safely remove NN nodes or layers?

Optimization challenges to analyze trained neural networks

Example: Classification of MNIST digits [Tsay et al., 2021]



Given	....
Trained NN	
Image	$\bar{\boldsymbol{x}}$
Label	$j = 9$
Adversary?	$k = 4$



$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_1 = 4$ $\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_\infty = 0.05$

- **Verification** *[Feasibility]* Is there an adversary labeled k within a given perturbation (e.g., by ℓ_1- or ℓ_∞-norm)?
- **Optimal adversary** [Anderson et al., 2020] What image within a perturbation radius maximizes the prediction difference?
- **Minimally distorted adversary** [Croce and Hein, 2020] Smallest perturbation over which NN can predict adversarial label k?
- **Lossless compression** [Serra et al., 2020] Can I safely remove NN nodes or layers?

Verification [sign check only] & Optimal Adversary

$$
\begin{aligned}
\max \quad & f_k(\boldsymbol{x}^L) - f_j(\boldsymbol{x}^L) \\
\text{s.t.} \quad & x_i^\ell = \max\left(0, \left(\left(\boldsymbol{w}_i^{\ell-1}\right)^T \boldsymbol{x}^{\ell-1} + b\right)\right) \quad \forall \ell \in \{1, \dots, L\} = \text{Layer},\ i \in \text{Node}^\ell \\
& \boldsymbol{x} \in \mathcal{X}
\end{aligned}
$$

Here, f_k and f_j correspond to the k- and j-th elements of the neural network output layer L, respectively. $\mathcal{X}$ defines the domain of perturbations.

International Verification of Neural Networks Competition

Specialized codes win ▪ Branch & bound on GPUs (α-β CROWN) ▪ Thoughtful heuristics

Software tools?

Neural network verification

- **MIP** MIPVerify [Tjeng et al., 2017] ▪ NSVerify [Akintunde et al., 2018]
- **SMT** Reluplex [Katz et al., 2017] ▪ marabou [Katz et al., 2019]
- **CP + MIP + Other** CROWN & Variants [Zhang et al., 2018, Xu et al., 2020, Salman et al., 2019, Xu et al., 2021, Wang et al., 2021, Zhang et al., 2022b,a]

Software tools?

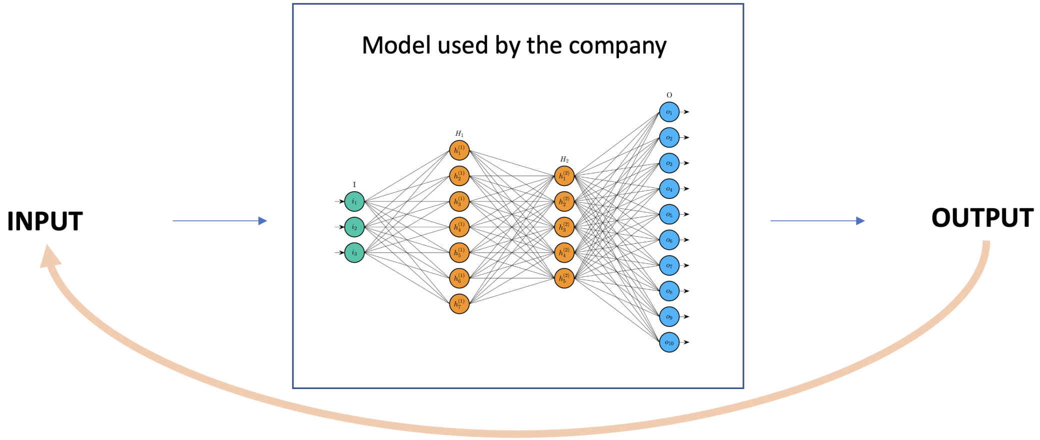
Neural network verification

- **MIP** MIPVerify [Tjeng et al., 2017] ▪ NSVerify [Akintunde et al., 2018]
- **SMT** Reluplex [Katz et al., 2017] ▪ marabou [Katz et al., 2019]
- **CP + MIP + Other** CROWN & Variants [Zhang et al., 2018, Xu et al., 2020, Salman et al., 2019, Xu et al., 2021, Wang et al., 2021, Zhang et al., 2022b,a]

Optimization over ML models

- *MeLOn* [Schweidtmann and Mitsos, 2019] dense sigmoid NNs, reduced-space formulation,
- *JANOS* [Bergman et al., 2022] dense ReLU NNs & logistic regression, Gurobi formulation,
- *reluMIP* [Lueg et al., 2021] dense ReLU NNs, Pyomo big-M formulation,
- *OptiCL* [Maragno et al., 2021] mixed-integer formulations of its own surrogates,
- *OMLT* Dense & convolutional NNs, Gradient-boosted trees, Competing formulations

Solve inverse problems over trained neural networks

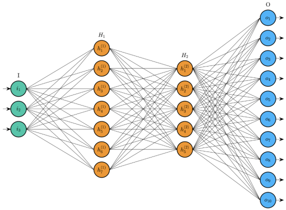


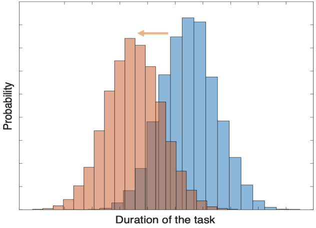
What is the input that achieves the desired output?

nPlan: Construction Start-Up

Task characteristics (given by the project manager)

- Windows installation
- Season
- Number of workers



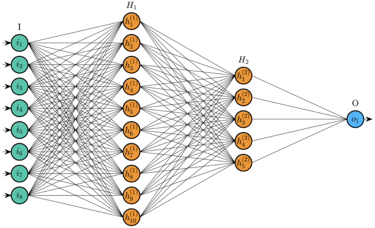


Can we find a combination of the characteristics of the task that reduce the time span?

Finance

Individual/Company features

- Salary
- Homeowner
- Value of other assets
- Value of other credits
- Age
- Credit history
- Dependent family members
- Other income
- Other debtors/guarantors



Granted

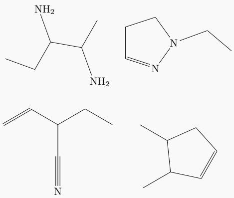
- What changes can be done in the individual features for the credit to be granted?
- What is the minimum value of the assets such that the credit is granted?

Computer-aided molecular design [with BASF]

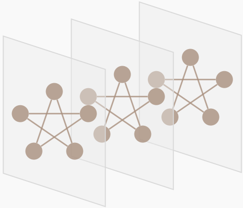
[Zhang et al., 2023]

prediction (forward)

molecules



GNN



properties

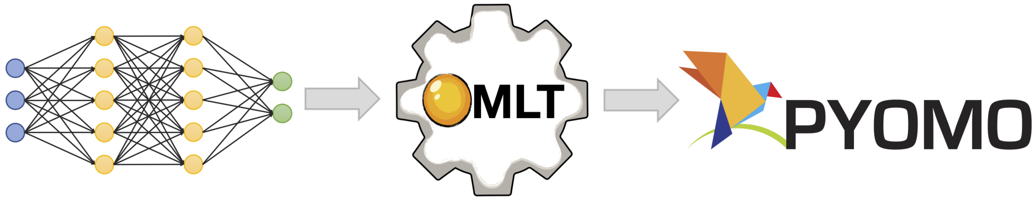
quantum mechanics
physical chemistry
biophysics
physiology
...

optimization (backward)

OMLT: Optimization & Machine Learning Toolkit [Ceccon et al., 2022]

https://github.com/cog-imperial/OMLT

CI passing | codecov 94% | docs passing



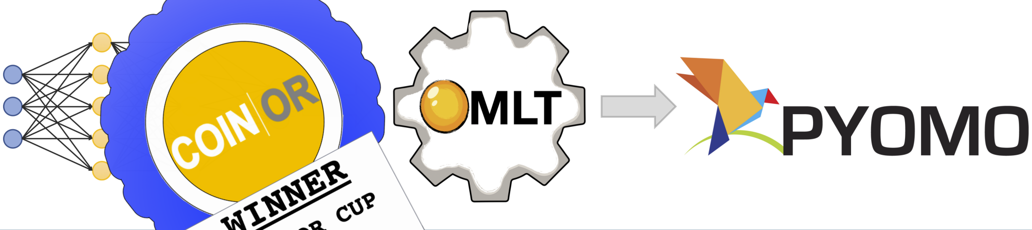
Why represent trained machine learning models as Pyomo formulations?

- **Adversarial examples** Verification, optimal adversary, minimally-distorted adversary, lossless compression
- **Machine learning** Maximize a neural acquisition function, Bayesian optimization
- **Engineering** Machine learning models may replace complicated constraints or serve as surrogates in larger design & operations problems.

OMLT: Optimization & Machine Learning Toolkit [Ceccon et al., 2022]

https://github.com/cog-imperial/OMLT

CI passing | codecov 94% | docs passing



Why represent traine… models as Pyomo formulations?

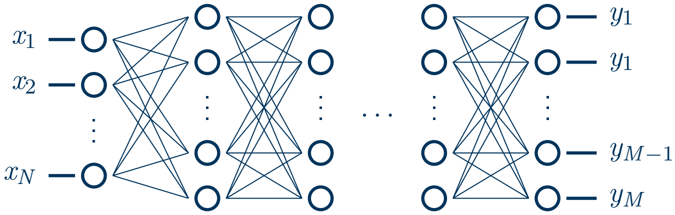
- **Adversarial exan…** …ptimal adversary, minimally-distorted adversary, lossless compression
- **Machine learning** M…imize a neural acquisition function, Bayesian optimization
- **Engineering** Machine learning models may replace complicated constraints or serve as surrogates in larger design & operations problems.

What type of optimization problem do we want to solve?

Hybridize mechanistic, model-based optimization with surrogate models learned from data

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad f_0(\boldsymbol{x}, \boldsymbol{y})$$

$$f_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0 \quad \forall i \in \{1, 2, \ldots, C\}$$

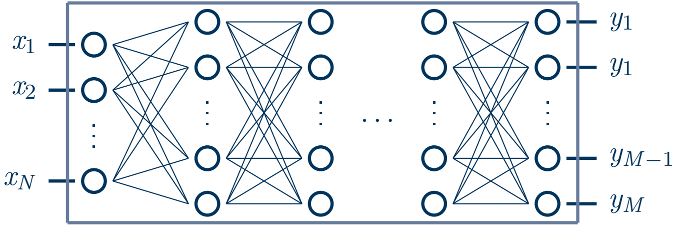


What type of optimization problem do we want to solve?

Hybridize mechanistic, model-based optimization with surrogate models learned from data

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad f_0(\boldsymbol{x}, \boldsymbol{y})$$

$$f_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0 \quad \forall i \in \{1, 2, \dots, C\}$$



What type of optimization problem do we want to solve?

Hybridize mechanistic, model-based optimization with surrogate models learned from data

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{y}} \quad & f_0(\boldsymbol{x}, \boldsymbol{y}) \\
& f_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0 \quad \forall i \in \{1, 2, \ldots, C\}
\end{aligned}
$$

What type of optimization problem do we want to solve?

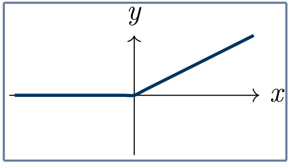
Hybridize mechanistic, model-based optimization with surrogate models learned from data

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad f_0(\boldsymbol{x}, \boldsymbol{y})$$
$$f_i(\boldsymbol{x}, \boldsymbol{y}) \leq 0 \quad \forall i \in \{1, 2, \ldots, C\}$$

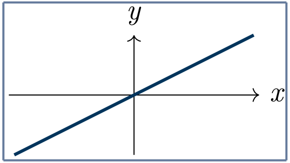
The `OmltBlock` abstraction encapsulates neural networks (NN) & trees

Dense NN ▪ CNN ▪ GNN (MPNN) ▪ Gradient boosted trees (GBT) ▪ Linear model trees

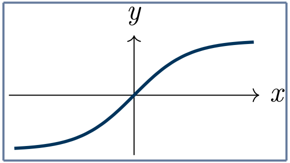
How NN activation functions map onto OMLT formulations . . .



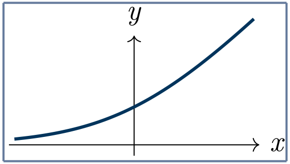
ReLU



linear

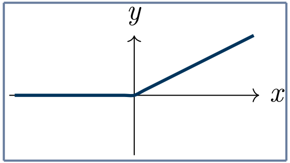


tanh

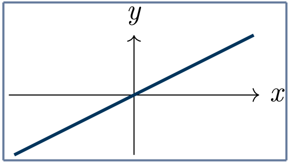


softplus

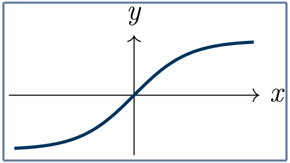
How NN activation functions map onto OMLT formulations . . .



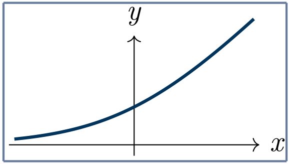
ReLU



linear



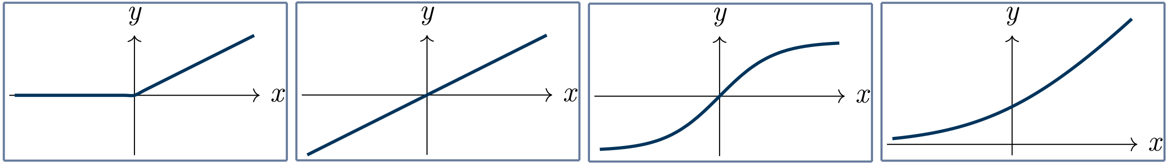
tanh



softplus

`ReluBigMFormulation`
`ReluComplementarityFormulation`
`ReluPartitionFormulation`

How NN activation functions map onto OMLT formulations . . .

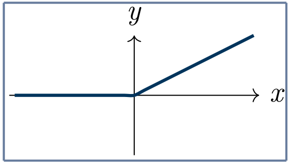


ReLU linear tanh softplus

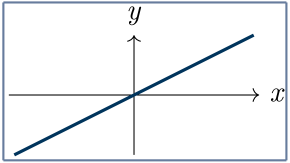
`ReluBigMFormulation`
`ReluComplementarityFormulation`
`ReluPartitionFormulation`

`FullSpaceSmoothNNFormulation`
`ReducedSpaceSmoothNNFormulation`

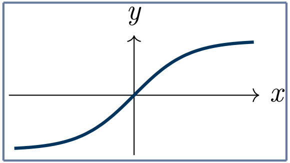
How NN activation functions map onto OMLT formulations . . .



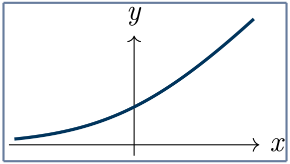
ReLU



linear



tanh



softplus

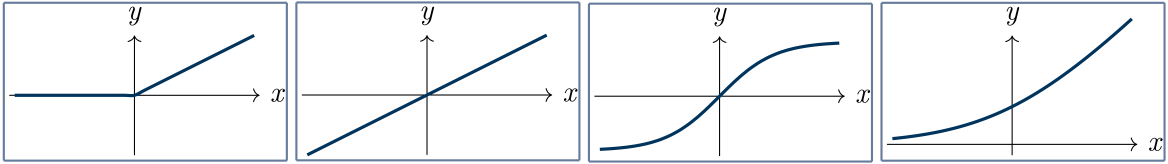
`ReluBigMFormulation`
`ReluComplementarityFormulation`
`ReluPartitionFormulation`

`FullSpaceSmoothNNFormulation`
`ReducedSpaceSmoothNNFormulation`

Formulations [Schweidtmann and Mitsos, 2019, Anderson et al., 2020, Tsay et al., 2021, Yang et al., 2021]

Non-smooth [`ReluBigMFormulation`, `ReluComplementarityFormulation`, `ReluPartitionFormulation`] ReLU ▪ **Smooth** [`{Full,Reduced}SpaceSmoothNNFormulation`] Linear ▪ Tanh ▪ Sigmoid ▪ Softplus ▪ Smooth monotonic

How NN activation functions map onto OMLT formulations . . .



ReLU linear tanh softplus

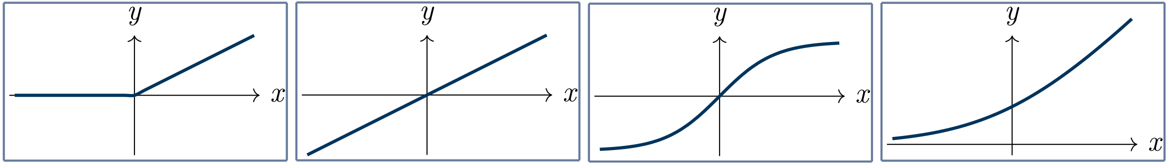
ReluBigMFormulation
ReluComplementarityFormulation
ReluPartitionFormulation

FullSpaceSmoothNNFormulation
ReducedSpaceSmoothNNFormulation

Optimization solver software — EPL ≡ Eclipse Public License; Prop ≡ Proprietary

Mixed-integer linear [Relu{BigM,Partition}Formulation] CBC [EPL] ▪ Gurobi [Prop] ▪ Xpress [Prop] ▪ CPLEX [Prop] **Nonlinear** [{Full,Reduced}SpaceSmoothNNFormulation, ReluComplementarityFormulation] Ipopt [EPL] ▪ SNOPT [Prop] ▪ MINOS [Prop]

How NN activation functions map onto OMLT formulations . . .



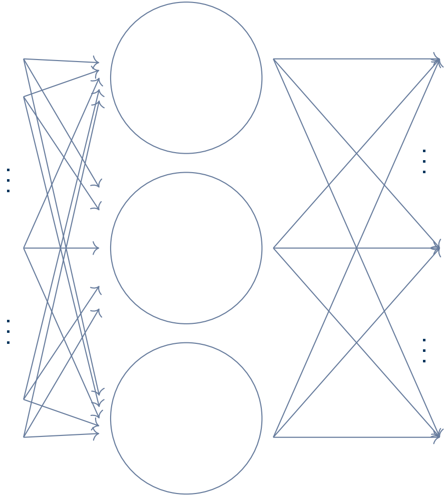
ReLU linear tanh softplus

Optimization solver software EPL ≡ Eclipse Public License; Prop ≡ Proprietary

Mixed-integer linear [Relu{BigM,Partition}Formulation] CBC [EPL] ▪ Gurobi [Prop] ▪ Xpress [Prop] ▪ CPLEX [Prop] **Nonlinear** [{Full,Reduced}SpaceSmoothNNFormulation, ReluComplementarityFormulation] Ipopt [EPL] ▪ SNOPT [Prop] ▪ MINOS [Prop]

Big-M formulation of a learned ReLU neural network

Lomuscio and Maganti [2017], Fischetti and Jo [2018]



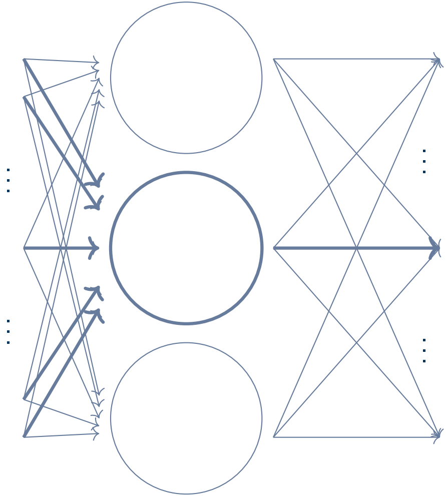
$$y \geq (\boldsymbol{w}^T \boldsymbol{x} + b)$$
$$y \leq (\boldsymbol{w}^T \boldsymbol{x} + b) - (1 - \sigma) LB^0$$
$$0 \leq y \leq \sigma UB^0$$
$$\sigma \in \{0, 1\}$$

Big-M coefficients $LB^0, UB^0 \in \mathbb{R}$

$(\boldsymbol{w}^T \boldsymbol{x} + b) \in [LB^0, UB^0]$

Big-M formulation of a learned ReLU neural network

Lomuscio and Maganti [2017], Fischetti and Jo [2018]



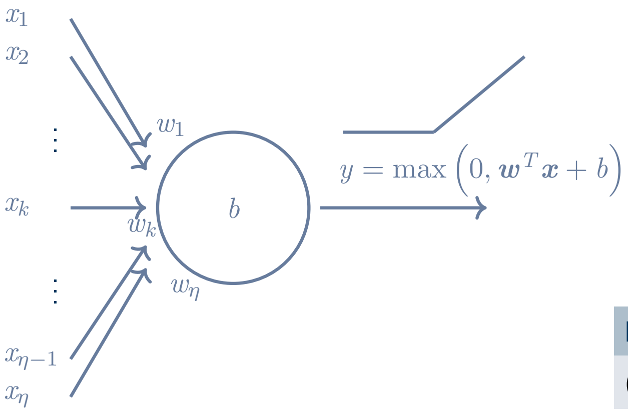
$$y \geq (\boldsymbol{w}^T \boldsymbol{x} + b)$$
$$y \leq (\boldsymbol{w}^T \boldsymbol{x} + b) - (1 - \sigma) LB^0$$
$$0 \leq y \leq \sigma\, UB^0$$
$$\sigma \in \{0, 1\}$$

Big-M coefficients $LB^0, UB^0 \in \mathbb{R}$

$(\boldsymbol{w}^T \boldsymbol{x} + b) \in [LB^0, UB^0]$

Big-M formulation of a learned ReLU neural network

Lomuscio and Maganti [2017], Fischetti and Jo [2018]



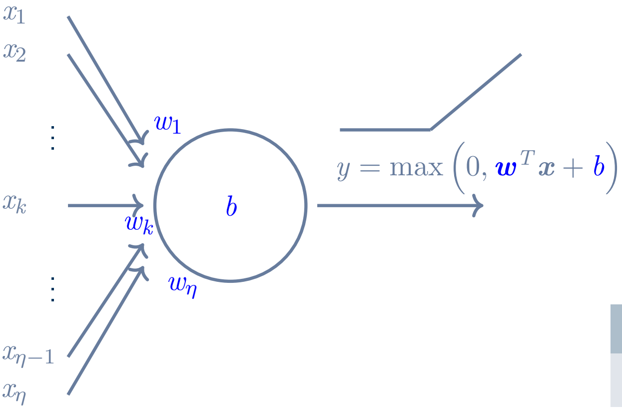
$$y \geq (\boldsymbol{w}^T\boldsymbol{x} + b)$$
$$y \leq (\boldsymbol{w}^T\boldsymbol{x} + b) - (1-\sigma)LB^0$$
$$0 \leq y \leq \sigma\, UB^0$$
$$\sigma \in \{0, 1\}$$

Big-M coefficients $LB^0, UB^0 \in \mathbb{R}$

$(\boldsymbol{w}^T\boldsymbol{x} + b) \in [LB^0, UB^0]$

Big-M formulation of a learned ReLU neural network

Lomuscio and Maganti [2017], Fischetti and Jo [2018]



$$y \geq (\boldsymbol{w}^T\boldsymbol{x} + b)$$

$$y \leq (\boldsymbol{w}^T\boldsymbol{x} + b) - (1-\sigma)LB^0$$

$$0 \leq y \leq \sigma\, UB^0$$

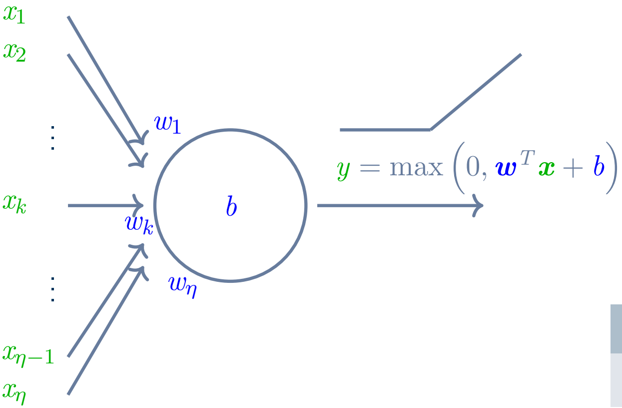
$$\sigma \in \{0, 1\}$$

Big-M coefficients $LB^0, UB^0 \in \mathbb{R}$

$(\boldsymbol{w}^T\boldsymbol{x} + b) \in [LB^0, UB^0]$

Big-M formulation of a learned ReLU neural network

Lomuscio and Maganti [2017], Fischetti and Jo [2018]



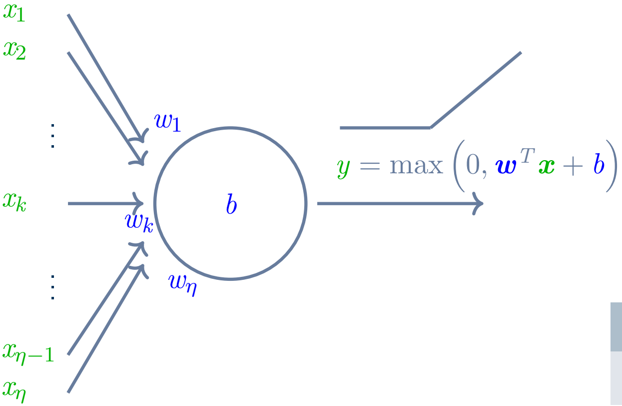
$$y \geq (\boldsymbol{w}^T\boldsymbol{x} + b)$$
$$y \leq (\boldsymbol{w}^T\boldsymbol{x} + b) - (1 - \sigma)LB^0$$
$$0 \leq y \leq \sigma\, UB^0$$
$$\sigma \in \{0, 1\}$$

Big-M coefficients $LB^0, UB^0 \in \mathbb{R}$

$(\boldsymbol{w}^T\boldsymbol{x} + b) \in [LB^0, UB^0]$

Big-M formulation of a learned ReLU neural network

Lomuscio and Maganti [2017], Fischetti and Jo [2018]



$$y \geq (\boldsymbol{w}^T\boldsymbol{x} + b)$$

$$y \leq (\boldsymbol{w}^T\boldsymbol{x} + b) - (1 - \sigma)LB^0$$

$$0 \leq y \leq \sigma\, UB^0$$

$$\sigma \in \{0, 1\}$$

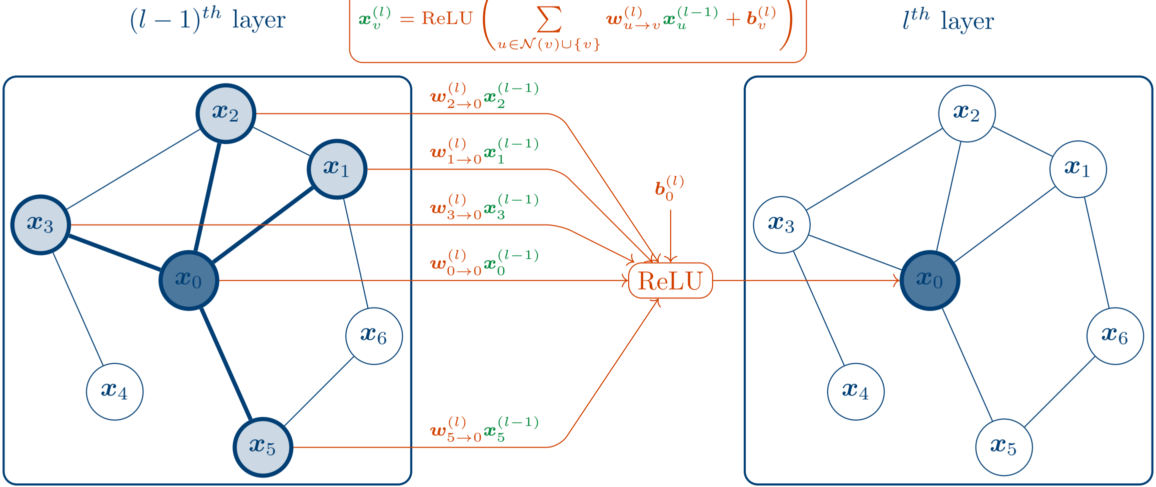
Big-M coefficients $LB^0, UB^0 \in \mathbb{R}$

$(\boldsymbol{w}^T\boldsymbol{x} + b) \in [LB^0, UB^0]$

Message passing with fixed graph structure

[Hojny et al., 2024]

$$x_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} w_{u \to v}^{(l)} x_u^{(l-1)} + b_v^{(l)}\right)$$



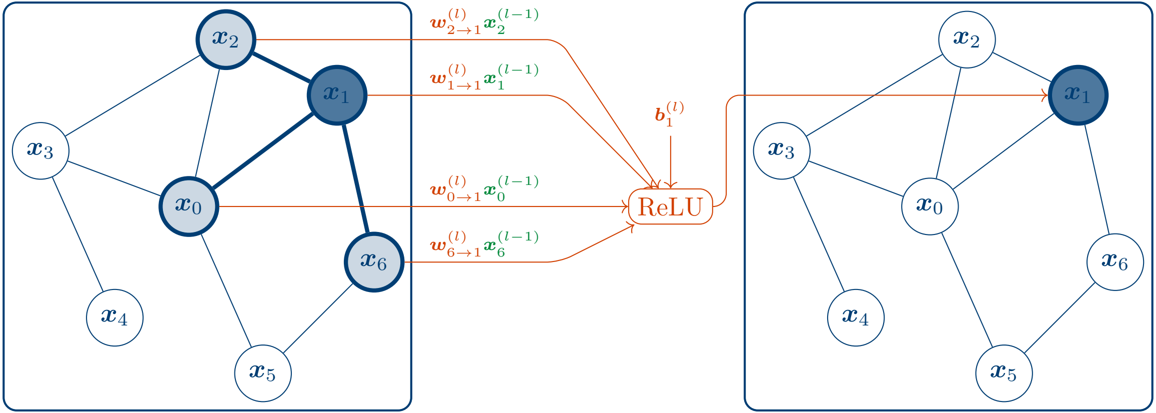
Message passing with fixed graph structure

[Hojny et al., 2024]

$(l-1)^{th}$ layer

$$\boldsymbol{x}_v^{(l)} = \text{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$

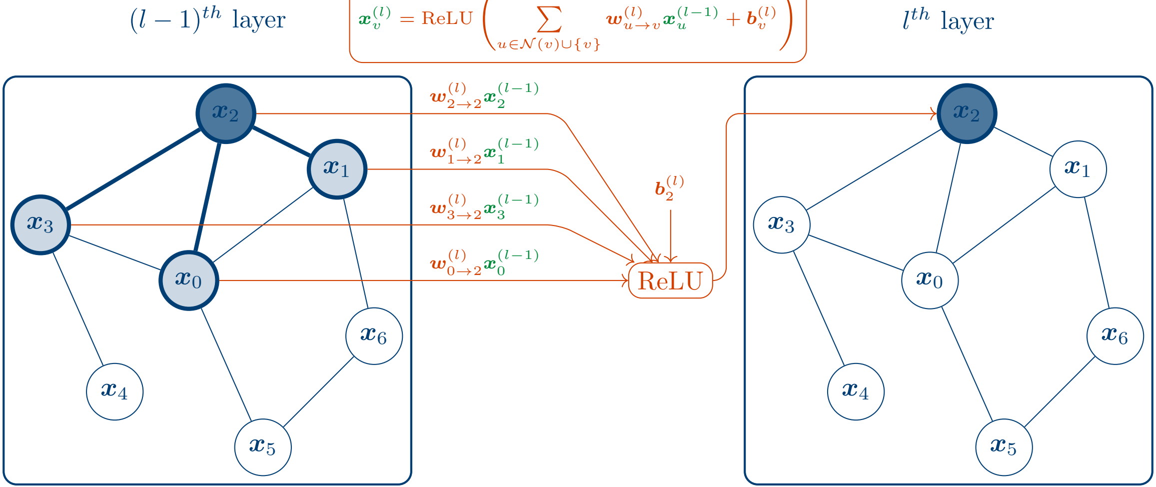
l^{th} layer



Message passing with fixed graph structure

[Hojny et al., 2024]

$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v)\cup\{v\}} \boldsymbol{w}_{u\to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$

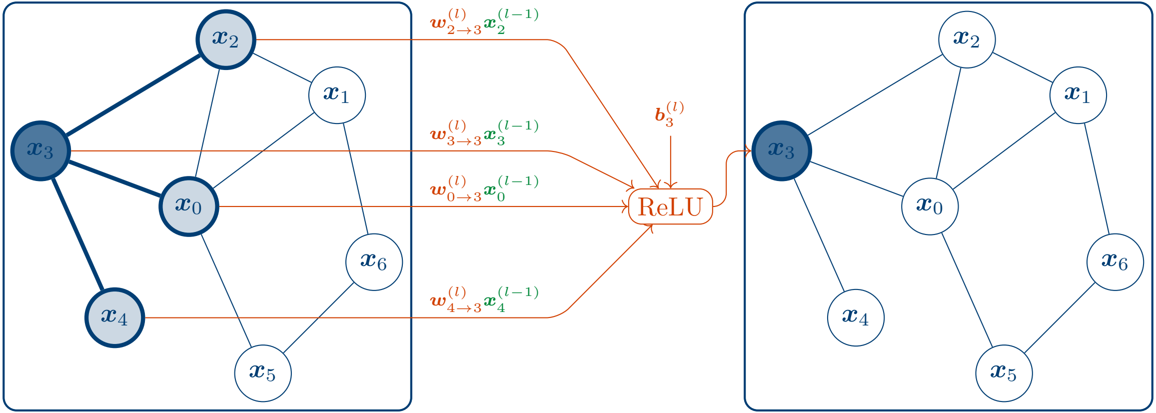


Message passing with fixed graph structure [Hojny et al., 2024]

$(l-1)^{th}$ layer

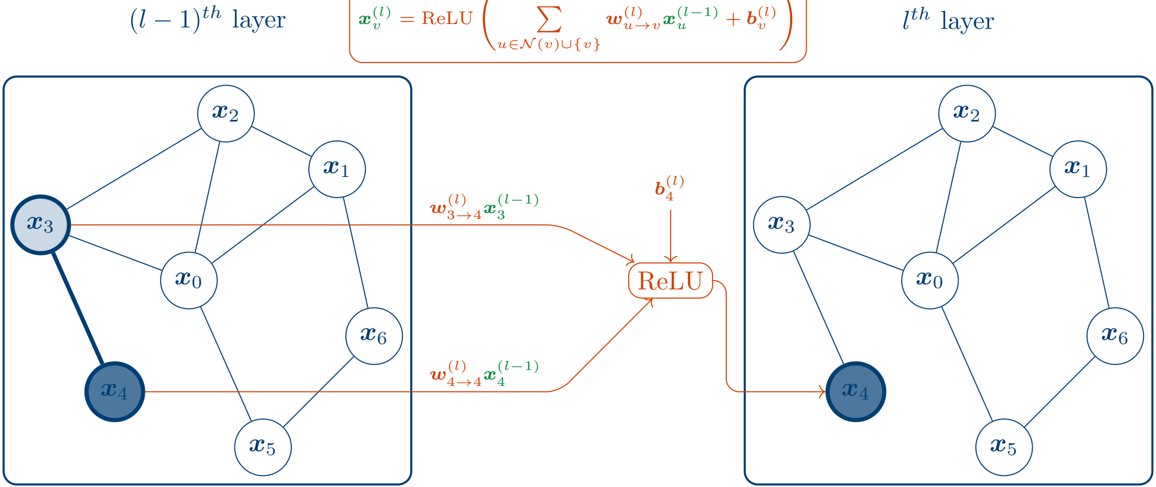
$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u\in\mathcal{N}(v)\cup\{v\}} \boldsymbol{w}_{u\to v}^{(l)}\boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$

l^{th} layer



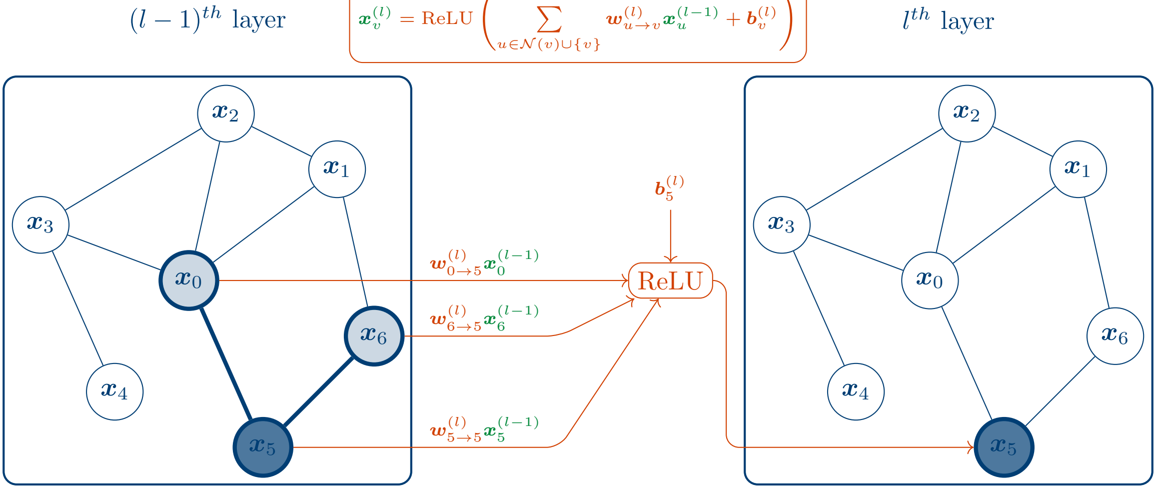
Message passing with fixed graph structure [Hojny et al., 2024]

$$x_v^{(l)} = \text{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} w_{u \to v}^{(l)} x_u^{(l-1)} + b_v^{(l)}\right)$$



Message passing with fixed graph structure [Hojny et al., 2024]

$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{w}_{u\to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$

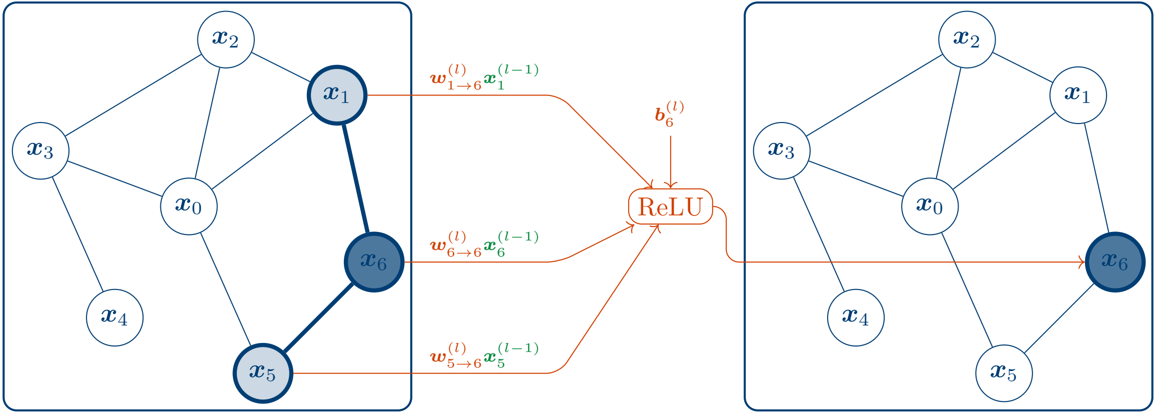


Message passing with fixed graph structure [Hojny et al., 2024]

$(l-1)^{th}$ layer

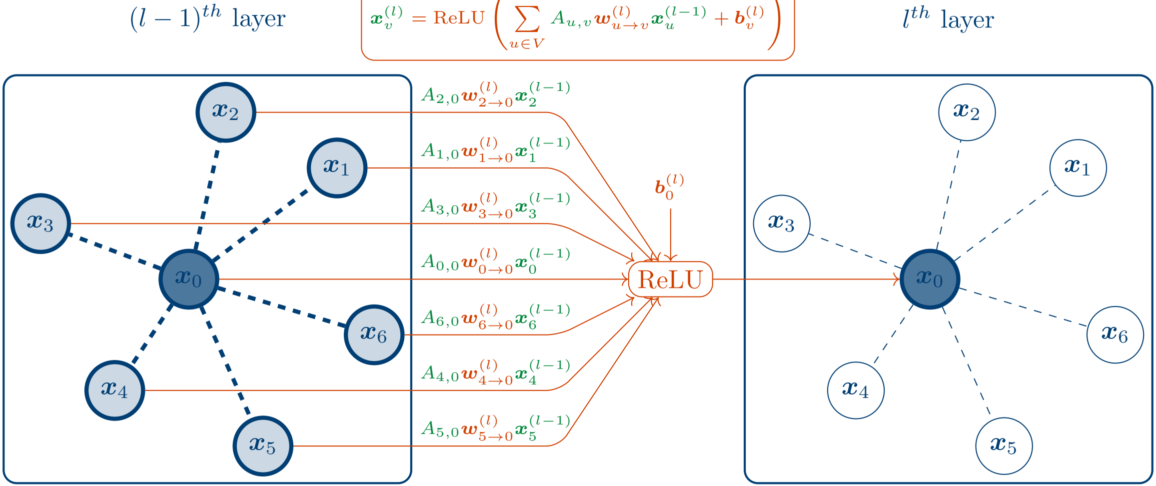
$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$

l^{th} layer



Message passing with unknown graph structure

$$x_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in V} A_{u,v} w_{u \to v}^{(l)} x_u^{(l-1)} + b_v^{(l)}\right)$$



MIP encoding of MPNNs

$$\boldsymbol{x}_v^{(l)} = \max\{\underbrace{\bar{\boldsymbol{x}}_v^{(l)}}, \mathbf{0}\} \longleftrightarrow \begin{cases} x_{v,f}^{(l)} \geq 0 \\ x_{v,f}^{(l)} \geq \bar{x}_{v,f}^{(l)} \\ x_{v,f}^{(l)} \leq \bar{x}_{v,f}^{(l)} - lb(\bar{x}_{v,f}^{(l)}) \cdot (1 - \sigma_{v,f}^{(l)}) \\ x_{v,f}^{(l)} \leq ub(\bar{x}_{v,f}^{(l)}) \cdot \sigma_{v,f}^{(l)} \end{cases}$$

$$\downarrow$$

$$\bar{\boldsymbol{x}}_v^{(l)} = \sum_{u \in V} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_{u \to v}^{(l-1)} + \boldsymbol{b}_v^{(l)}$$

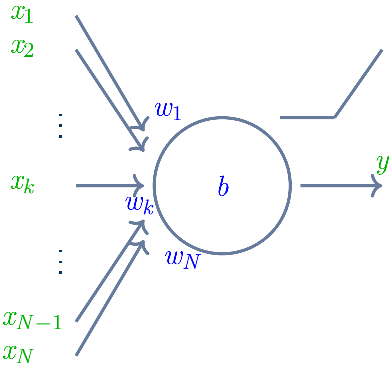
$$\uparrow$$

$$\overbrace{\boldsymbol{x}_{u \to v}^{(l-1)}} = A_{u,v} \boldsymbol{x}_u^{(l-1)} \leftrightarrow \begin{cases} x_{u \to v,f}^{(l-1)} \geq lb(x_{u,f}^{(l-1)}) \cdot A_{u,v} \\ x_{u \to v,f}^{(l-1)} \leq ub(x_{u,f}^{(l-1)}) \cdot A_{u,v} \\ x_{u \to v,f}^{(l-1)} \leq x_{u,f}^{(l-1)} - lb(x_{u,f}^{(l-1)}) \cdot (1 - A_{u,v}) \\ x_{u \to v,f}^{(l-1)} \geq x_{u,f}^{(l-1)} - ub(x_{u,f}^{(l-1)}) \cdot (1 - A_{u,v}) \end{cases}$$

OMLT puts *optimization formulations* in competition [Ceccon et al., 2022]

Key idea One optimization formulation may be more effective than another

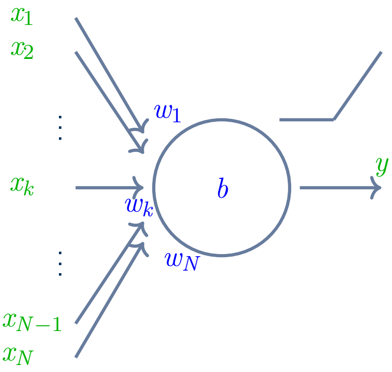
- Algebraic modelling languages, e.g., Pyomo, make switching optimization *solvers* easy
- OMLT makes switching formulations as easy as changing a couple lines of code



OMLT puts *optimization formulations* in competition [Ceccon et al., 2022]

Key idea One optimization formulation may be more effective than another

- Algebraic modelling languages, e.g., Pyomo, make switching optimization *solvers* easy
- OMLT makes switching formulations as easy as changing a couple lines of code



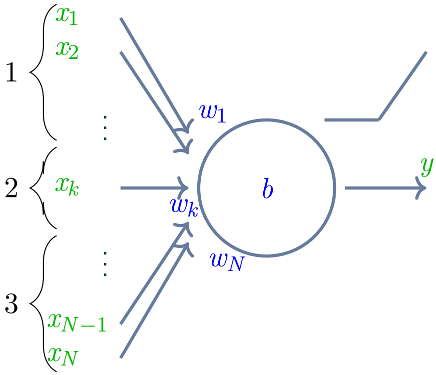
Big-M formulation [Anderson et al., 2020]

```
formulation = ReluBigMFormulation(net_relu)
```

OMLT puts *optimization formulations* in competition [Ceccon et al., 2022]

Key idea One optimization formulation may be more effective than another

- Algebraic modelling languages, e.g., Pyomo, make switching optimization *solvers* easy
- OMLT makes switching formulations as easy as changing a couple lines of code



Big-M formulation [Anderson et al., 2020]

```
formulation = ReluBigMFormulation(net_relu)
```

Partition-based formulation [Tsay et al., 2021]

```
P = 3
split_func = lambda w: partition_split_func(w, P)
formulation = ReluPartitionFormulation(
    net_relu, split_func=split_func)
```

What's next? Embedding trained ML models into optimal decision-making

Wish list

Algorithms	Addressing nonconvexity ▪ Managing problem size ▪ Proposing formulations
Applications	Lots more!
Models	Skip connections for NN? ▪ Recurrent NN
Software	OMLT back-end to other algebraic modeling languages ▪ Tree input

Challenges & opportunities

Nonconvexity	Nonconvex activation functions ▪ Discrete on/off adjacency matrix
Size	Activation function at every node? At every edge?

Adversarial attack v.s. Certifiable robustness

Machine learning models are vulnerable: small input changes could lead to wrong predictions.

Denote f as a model, assume $\mathcal{P}(X^*)$ is the admissible perturbations on input X^*.

Adversarial attack	Certifiable robustness
$\exists X \in \mathcal{P}(X^*)$, s.t., $f(X) \neq f(X^*)$	$f(X) = f(X^*),\ \forall X \in \mathcal{P}(X^*)$

Besides input features, the graph structure involved in graph neural networks (GNNs) provides more options to attack, while makes it harder to be verified (certified robustness).

Problem definition

Given a trained GNN f for graph/node classification task, where the predicted label corresponds to the maximal logit. Given an input (X^*, A^*) consisting of features X^* and adjacency matrix A^*, denote its predictive label as c^*. The worst case margin between predictive label c^* and attack label c under perturbations $\mathcal{P}(\cdot)$ is:

$$\begin{aligned} m(c^*, c) := \min_{(X,A)} \ & f_{c^*}(X, A) - f_c(X, A) \\ s.t. \ & X \in \mathcal{P}(X^*), \ A \in \mathcal{P}(A^*). \end{aligned} \tag{1}$$

A positive $m(c^*, c)$ means that the logit of class c^* is always larger than class c.

Let $\mathcal{C}$ be the set of all classes. If $m(c^*, c) > 0, \forall c \in \mathcal{C} \setminus \{c^*\}$, then any admissible perturbation can not change the predictive label, i.e., this GNN is robust at (X^*, A^*).

Admissible perturbations

Perturbations on features, i.e., $\mathcal{P}(X^*)$, are usually defined as a l_p norm ball around X^*. The choice of norm is quite flexible for attack since one feasible attack is sufficient. For verification, l_∞ norm is most commonly used since it defines bounds for each feature separately.

Remark: If only feature perturbations are allowed, then verifying a GNN is equivalent to verifying a NN since the connections between layers are fixed.

New challenges for GNN verification:

- Perturbations on graph structure, e.g., add edges/remove edges/inject nodes, directly change the connections between layers.
- Perturbations on one node indirectly attack other nodes via message passing or graph convolution.

Verification of message passing neural networks (MPNNs)

Motivation: classic and general GNN framework, but few certificates.

Tool: a recently developed mixed-integer programming (MIP) formulation for MPNNs.

Definition: consider a MPNN with l-th layer defined as:

$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in V} A_{u,v} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right), \ \forall v \in V \tag{2}$$

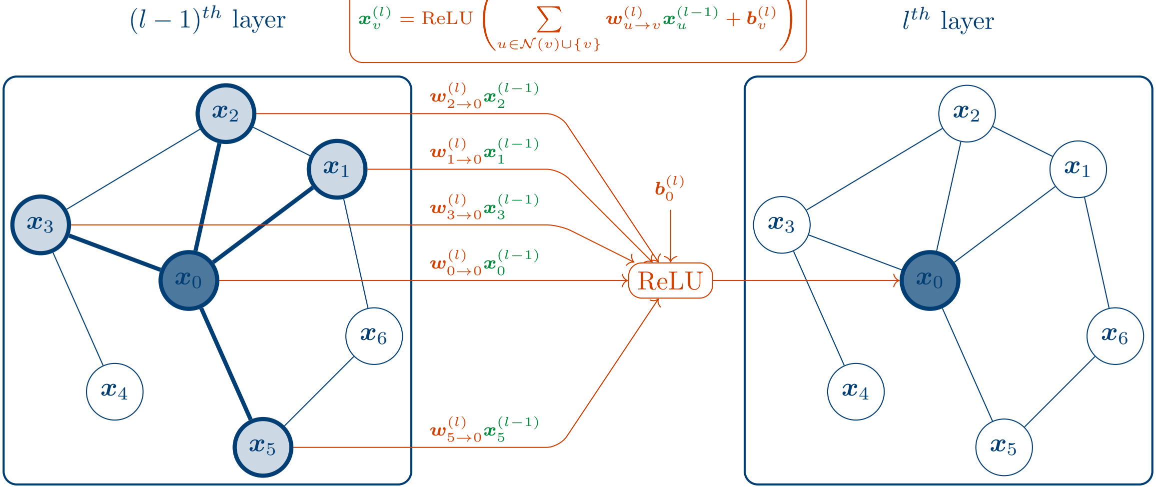
where $V = \{0, 1, \ldots, N-1\}$ is the node set, N is the number of nodes, $A_{u,v} \in \{0, 1\}$ denotes the existence of edge $u \to v$.

Perturbations:

- Graph classification: remove/add edges with global/local budgets.

Message passing with fixed graph structure [Hojny et al., 2024]

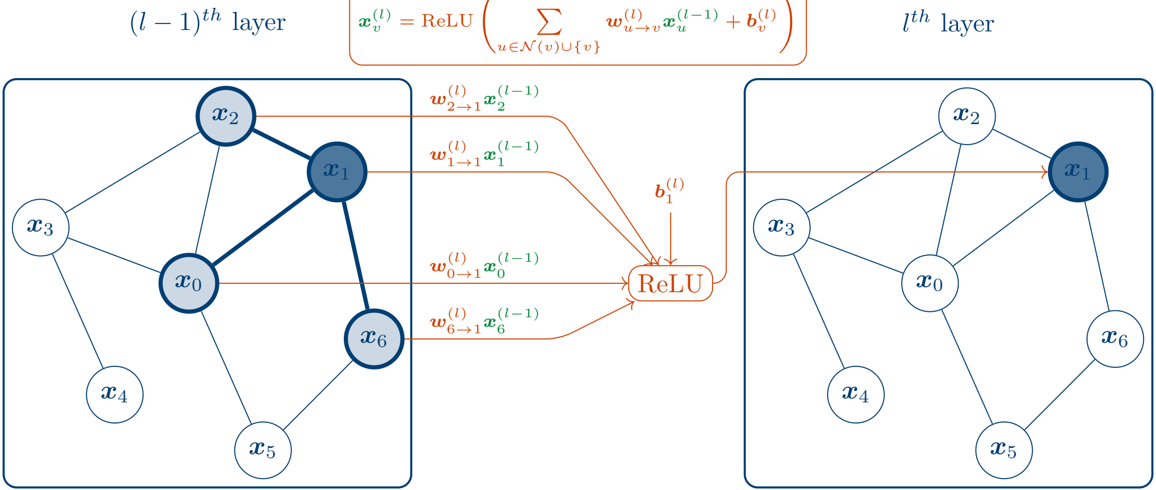
$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$



Message passing with fixed graph structure

[Hojny et al., 2024]

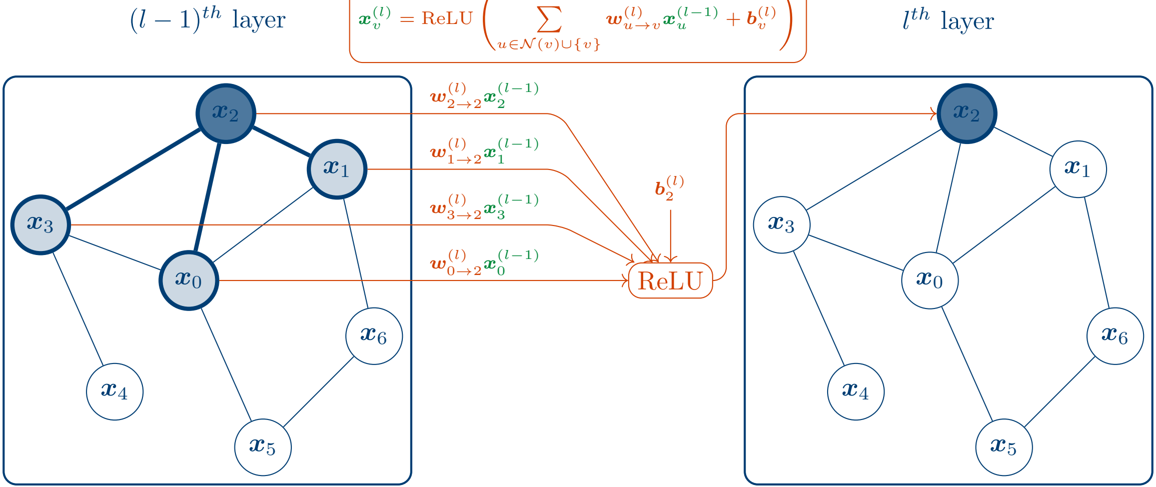
$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$



Message passing with fixed graph structure

[Hojny et al., 2024]

$$\boldsymbol{x}_v^{(l)} = \text{ReLU}\left(\sum_{u \in \mathcal{N}(v)\cup\{v\}} \boldsymbol{w}_{u\to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$



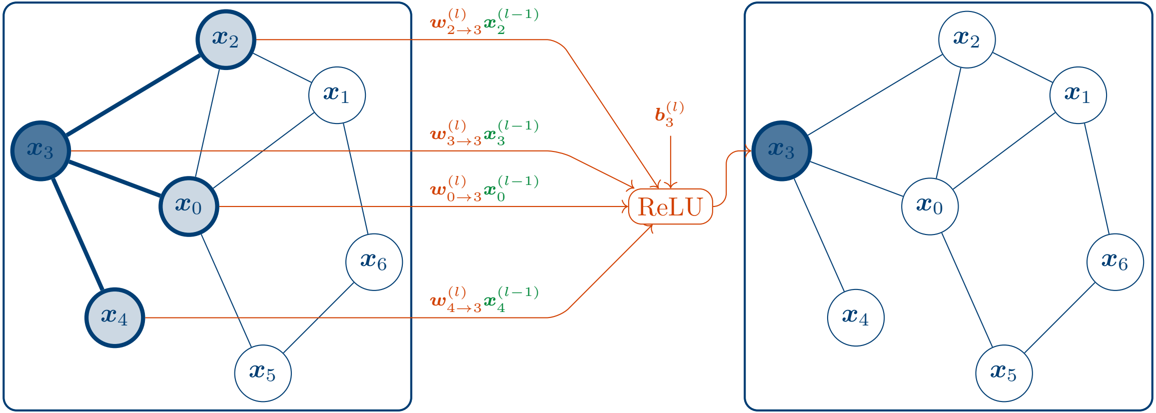
Message passing with fixed graph structure

[Hojny et al., 2024]

$(l-1)^{th}$ layer

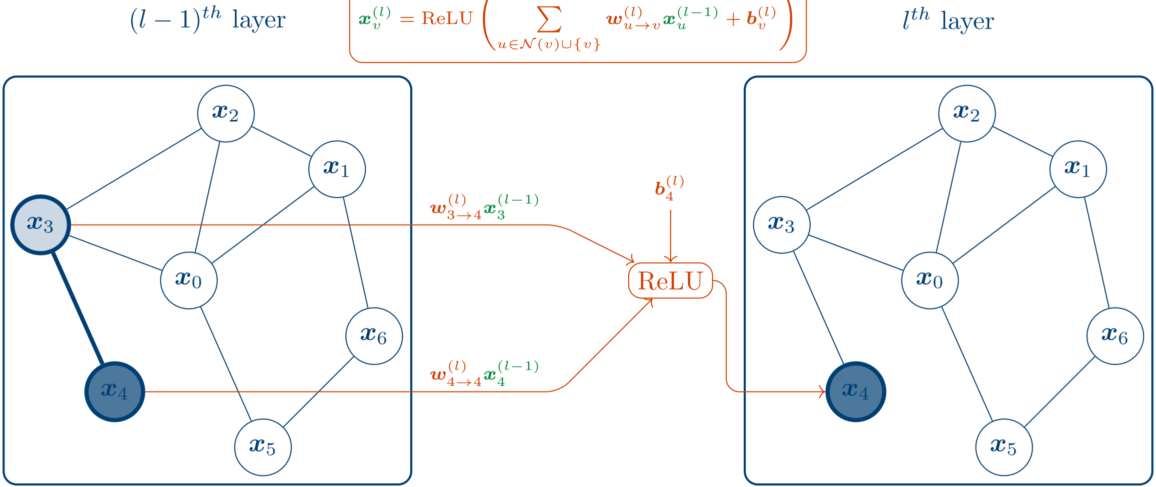
$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)} \right)$$

l^{th} layer



Message passing with fixed graph structure [Hojny et al., 2024]

$$x_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} w_{u \to v}^{(l)} x_u^{(l-1)} + b_v^{(l)}\right)$$



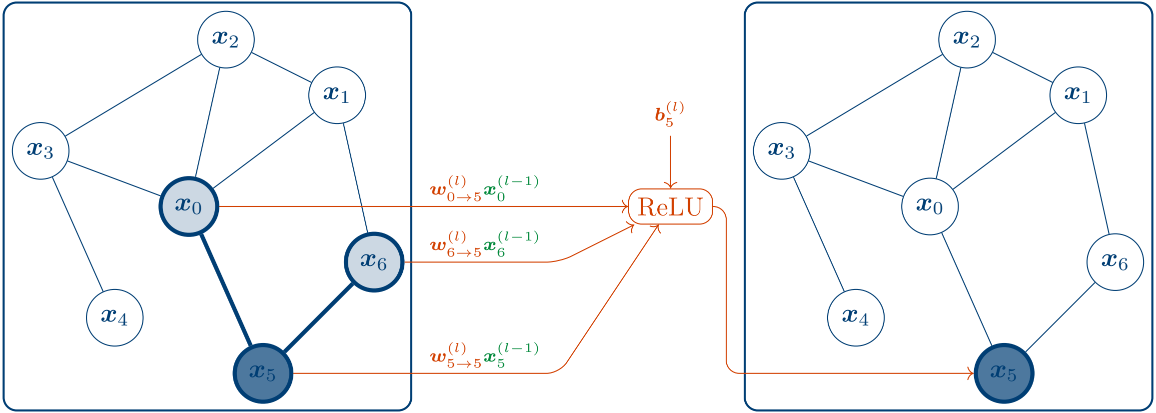
Message passing with fixed graph structure

[Hojny et al., 2024]

$(l-1)^{th}$ layer

$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)} \right)$$

l^{th} layer

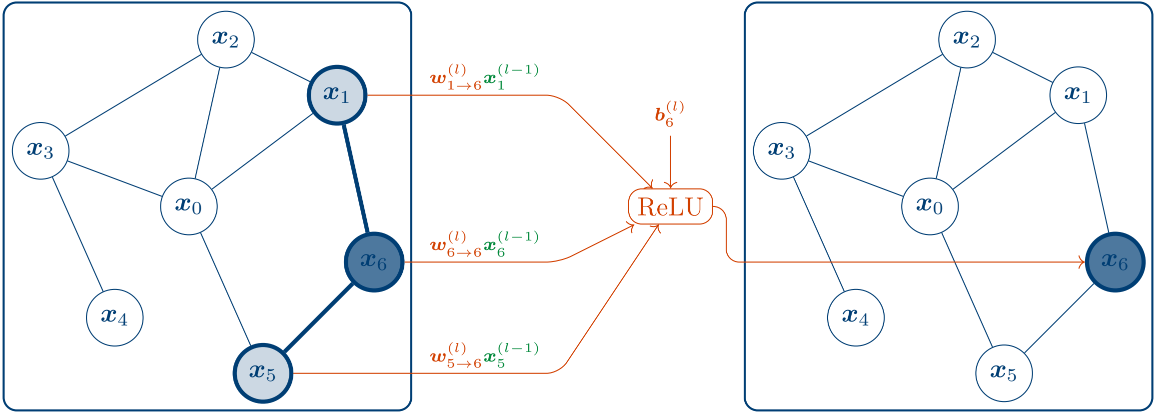


Message passing with fixed graph structure [Hojny et al., 2024]

$(l-1)^{th}$ layer

$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$

l^{th} layer

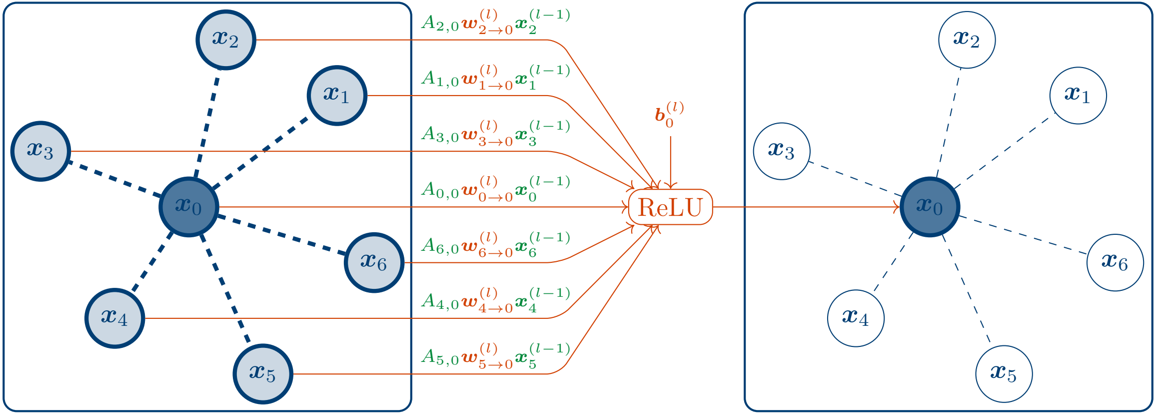


Message passing with unknown graph structure

$(l-1)^{th}$ layer

$$\boldsymbol{x}_v^{(l)} = \text{ReLU}\left(\sum_{u \in V} A_{u,v}\boldsymbol{w}_{u \to v}^{(l)}\boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$

l^{th} layer



MIP encoding of MPNNs

$$\boldsymbol{x}_v^{(l)} = \max\{\underbrace{\bar{\boldsymbol{x}}_v^{(l)}}, \boldsymbol{0}\} \longleftrightarrow \begin{cases} x_{v,f}^{(l)} \geq 0 \\ x_{v,f}^{(l)} \geq \bar{x}_{v,f}^{(l)} \\ x_{v,f}^{(l)} \leq \bar{x}_{v,f}^{(l)} - lb(\bar{x}_{v,f}^{(l)}) \cdot (1 - \sigma_{v,f}^{(l)}) \\ x_{v,f}^{(l)} \leq ub(\bar{x}_{v,f}^{(l)}) \cdot \sigma_{v,f}^{(l)} \end{cases}$$

$$\downarrow$$

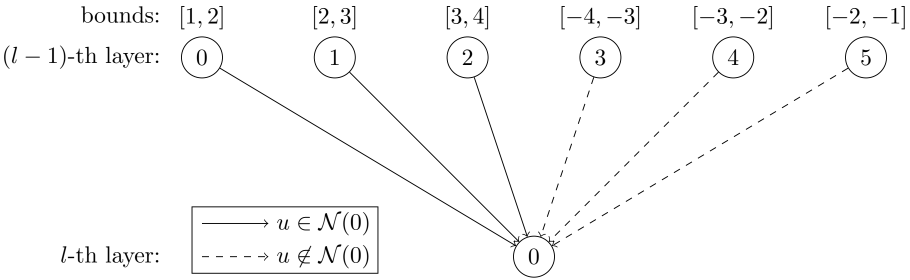
$$\bar{\boldsymbol{x}}_v^{(l)} = \sum_{u \in V} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_{u \to v}^{(l-1)} + \boldsymbol{b}_v^{(l)}$$

$$\uparrow$$

$$\overbrace{\boldsymbol{x}_{u \to v}^{(l-1)}} = A_{u,v} \boldsymbol{x}_u^{(l-1)} \leftrightarrow \begin{cases} x_{u \to v,f}^{(l-1)} \geq lb(x_{u,f}^{(l-1)}) \cdot A_{u,v} \\ x_{u \to v,f}^{(l-1)} \leq ub(x_{u,f}^{(l-1)}) \cdot A_{u,v} \\ x_{u \to v,f}^{(l-1)} \leq x_{u,f}^{(l-1)} - lb(x_{u,f}^{(l-1)}) \cdot (1 - A_{u,v}) \\ x_{u \to v,f}^{(l-1)} \geq x_{u,f}^{(l-1)} - ub(x_{u,f}^{(l-1)}) \cdot (1 - A_{u,v}) \end{cases}$$

Basic bounds tightening (*basic*)

Assume that there are $N = 6$ nodes with only one input and output feature. For simplicity, assume all weights equal to 1 and all biases equal to 0.

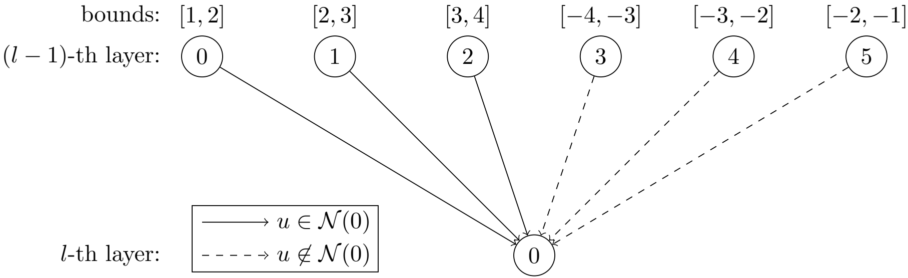


To get the bounds for node 0 in l-th layer, *basic* considers all possibilities of input nodes:

- $lb = \min(0,1) + \min(0,2) + \min(0,3) + \min(0,-4) + \min(0,-3) + \min(0,-2) = -9.$
- $ub = \max(0,2) + \max(0,3) + \max(0,4) + \max(0,-3) + \max(0,-2) + \max(0,-1) = 9.$

Static bounds tightening (*sbt*)

Given that the budget, i.e., the maximal number of modified edges of node 0, is 3. Denote the set of input nodes as $\mathcal{N}'(0)$, then we need to make sure that $|\mathcal{N}'(0)\Delta\mathcal{N}(0)| \leq 3$.

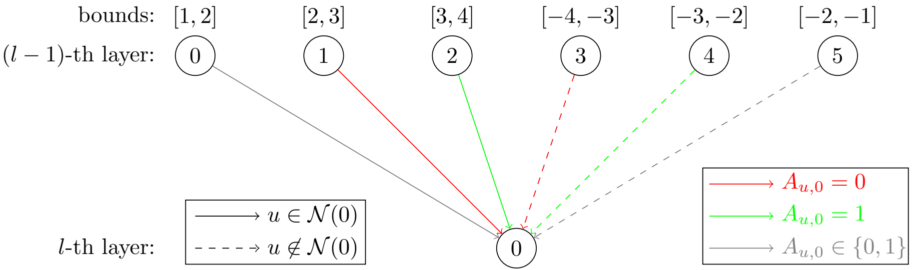


Comparing all possible options gives the *sbt* bounds:

- $lb = 1 + 243 = 4$: $\mathcal{N}'(0) = \{0, 1, 3, 4\}$, i.e., remove node 2 + add node 3 and 4.
- $ub = 2 + 3 + 4 = 9$: $\mathcal{N}'(0) = \mathcal{N}(0)$.

Aggressive bounds tightening (*abt*)

Assume that 4 decisions have been made in current branch-and-bound (B&B) tree node, which are $A_{1,0} = 0, A_{2,0} = 1, A_{3,0} = 0, A_{4,0} = 1$. Then we only have 1 budget left.

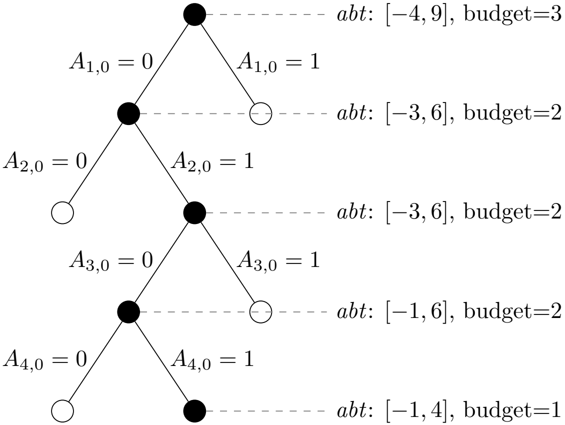


We can (i) change nothing, or (ii) remove node 0, or (iii) add node 5. The *abt* bounds are:

- $lb = 1 + 3 - 3 - 2 = -1$: add node 5.
- $ub = 2 + 4 - 2 = 4$: change nothing.

abt extends *sbt* to each B&B tree node

abt can be interpreted as applying *sbt* to a modified graph with reduced budgets at each B&B tree node. At root node, *abt* = *sbt*.



Numerical results

benchmark	method	all instances			robust instances		
		#	avg-time(s)	# solved	#	avg-time(s)	# solved
ENZYMES	SCIPbasic	5915	605.97	5579	3549	278.58	3444
	SCIPsbt	5915	**230.59**	**5831**	3549	**82.89**	**3528**
	SCIPabt	5915	246.02	5817	3549	88.95	3522
MUTAG	SCIPbasic	1589	679.86	1575	44	798.47	40
	SCIPsbt	1589	**196.07**	**1589**	44	336.41	**44**
	SCIPabt	1589	207.50	**1589**	44	**238.10**	**44**

Conclusion

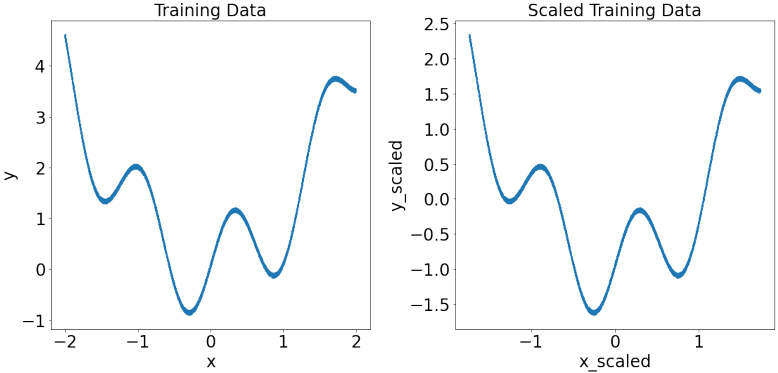
Based on the results of our SCIP implementation, we have the following observations:

- For moderate robust instances, *basic* $<$ *sbt* $\approx$ *abt*.
- For hard robust instances, *basic* $<$ *sbt* $<$ *abt*.
- For non-robust instances, *basic* $<$ *abt* $<$ *sbt*.

For a non-robust instance, the target is not verification but finding an attack. In such cases, tighter bounds derived from more cutting planes could result in slower solving times.

Neural Network Formulation Example: Data

neural_network_formulations.ipynb

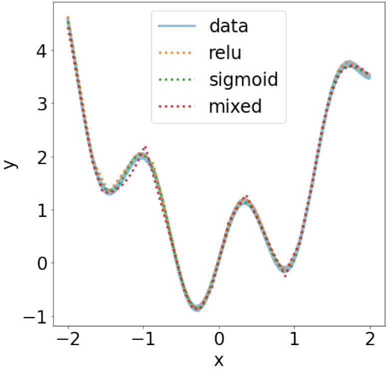


Read in the data 1 input x, 1 output y, 10^4 samples, Scaled has mean 0 & stdev 1

```
df = pd.read_csv("../data/sin_quadratic.csv",index_col=[0]);
```

Neural Network Formulation Example: Trained Neural Networks

`neural_network_formulations.ipynb`

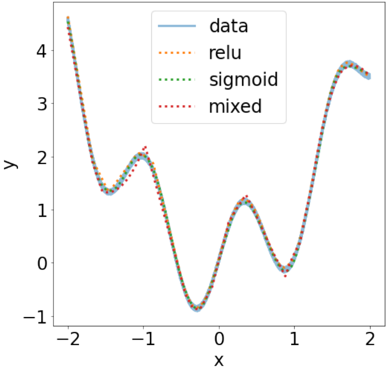


Build a Keras NN with ReLU activation

```
nn = Sequential(name='sin_wave_relu')
nn.add(Input(1))
nn.add(Dense(30, activation='relu'))
nn.add(Dense(30, activation='relu'))
nn.add(Dense(1))
nn.compile(optimizer=Adam(), loss='mse')
history = nn.fit(x=df['x_scaled'], y=df['y_scaled'],
    verbose=1, epochs=75)
```

Neural Network Formulation Example: Trained Neural Networks

neural_network_formulations.ipynb

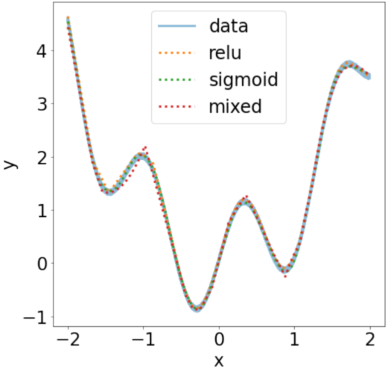


Build a Keras NN with sigmoid activation

```
nn = Sequential(name='sin_wave_sigmoid')
nn.add(Input(1))
nn.add(Dense(50, activation='sigmoid'))
nn.add(Dense(50, activation='sigmoid'))
nn.add(Dense(1))
nn.compile(optimizer=Adam(), loss='mse')
history = nn.fit(x=df['x_scaled'], y=df['y_scaled'],
    verbose=1, epochs=75)
```

Neural Network Formulation Example: Trained Neural Networks

neural_network_formulations.ipynb



Build a Keras NN with mixed (sigmoid/ReLU) activation

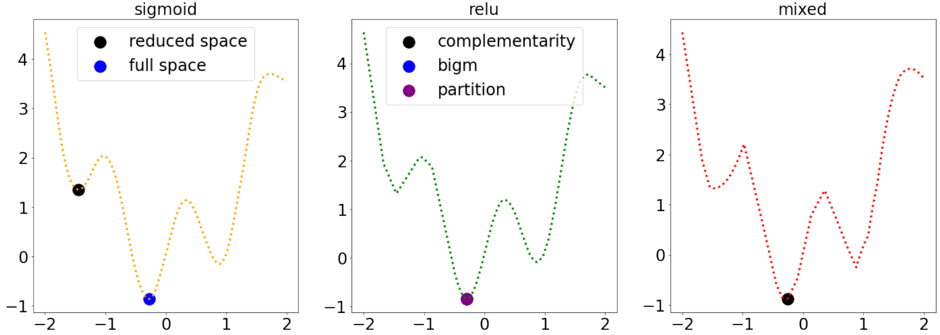
```
nn = Sequential(name='sin_wave_mixed')
nn.add(Input(1))
nn.add(Dense(50, activation='sigmoid'))
nn.add(Dense(50, activation='relu'))
nn.add(Dense(1))
nn.compile(optimizer=Adam(), loss='mse')
history = nn.fit(x=df['x_scaled'], y=df['y_scaled'],
    verbose=1, epochs=150)
```

Neural Network Formulation Example: Set up the optimization problem

```
net_sigmoid = keras_reader.load_keras_sequential(nn,scaler,input_bounds)
model = pyo.ConcreteModel()
model.x = pyo.Var(initialize = 0)
model.y = pyo.Var(initialize = 0)
model.obj = pyo.Objective(expr=(model.y))
model.nn = OmltBlock()
formulation = FullSpaceSmoothNNFormulation(net_sigmoid) #or ReducedSpaceSmoothNNFormulation
model.nn.build_formulation(formulation)

@model.Constraint()
def connect_inputs(mdl):
    return mdl.x == mdl.nn.inputs[0]

@model.Constraint()
def connect_outputs(mdl):
    return mdl.y == mdl.nn.outputs[0]

status = pyo.SolverFactory('ipopt').solve(model, tee=True)
solution = (pyo.value(model.x),pyo.value(model.y))
```

Neural Network Formulation Example: Optimization results

`neural_network_formulations.ipynb`



FullSpaceSmoothNNFormulation [Ipopt]

\# variables: 209, # constraints: 208

$x = -0.28, y = -0.86$

Solve Time: 0.14s

ReducedSpaceSmoothNNFormulation [Ipopt]

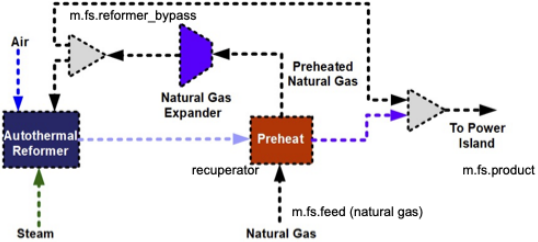
\# variables: 6, # constraints: 5

$x = -1.44, y = 1.36$

Solve Time: 0.08s

Other notebook examples . . .

https://github.com/cog-imperial/OMLT/tree/main/docs/notebooks

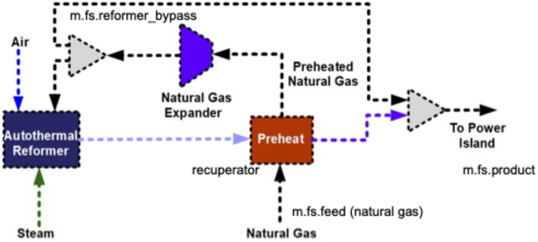


`auto-thermal-reformer{-relu}.ipynb`

develops an NN surrogate with data from a process model built using IDAES-PSE [Lee et al., 2021]

Other notebook examples ...

https://github.com/cog-imperial/OMLT/tree/main/docs/notebooks



auto-thermal-reformer{-relu}.ipynb

develops an NN surrogate with data from a process model built using IDAES-PSE [Lee et al., 2021]

Even more notebook examples ...

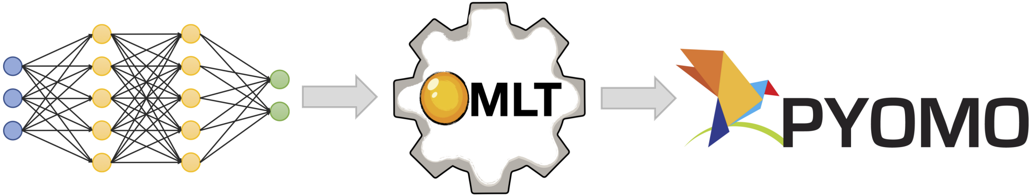
- `import_network.ipynb` imports NN models directly from Keras & ONNX. Using ONNX interoperability, it imports a NN model from PyTorch.
- `build_network.ipynb` builds a `NetworkDefinition` manually.
- `mnist_example_{dense, cnn}.ipynb` train fully dense and convolutional NNs on MNIST [LeCun et al., 2010] and find adversarial examples [Tjeng et al., 2017].
- `bo_with_trees.ipynb` optimizes the Rosenbrock function.

OMLT v 1.0 Summary

https://github.com/cog-imperial/OMLT

Key Contributions

- Automatically translate a trained machine learning model (neural network or gradient boosted tree) into Pyomo optimization constraints
- Achieve interoperability via the ONNX interface
- Easily switch and compare optimization formulations



Team members https://github.com/cog-imperial/OMLT

Francesco Ceccon
Imperial

Jordan Jalving
Sandia

Joshua Haddad
Sandia

Alexander Thebelt
Imperial

Calvin Tsay
Imperial

Carl D Laird
CMU

Ruth Misener
Imperial

You?
Join us on GitHub!

References I

Michael Akintunde, Alessio Lomuscio, Lalit Maganti, and Edoardo Pirovano. Reachability analysis for neural agent-environment systems. In *KR*, pages 184–193, 2018.

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.

David Bergman, Teng Huang, Philip Brooks, Andrea Lodi, and Arvind U Raghunathan. Janos: an integrated predictive and prescriptive modeling framework. *INFORMS Journal on Computing*, 34(2):807–816, 2022.

Michael L Bynum, Gabriel A Hackebeil, William E Hart, Carl D Laird, Bethany L Nicholson, John D Siirola, Jean-Paul Watson, and David L Woodruff. *PyomoOptimization Modeling in Python*, volume 67. Springer Nature, 2021.

F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D Laird, and R. Misener. OMLT: Optimization & machine learning toolkit. *Journal of Machine Learning Research*, 23(349):1–8, 2022.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR, 2020.

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

References II

Christopher Hojny, Shiqiang Zhang, Juan S Campos, and Ruth Misener. Verifying message-passing neural networks via topology-based bounds tightening. In *ICML*, 2024.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, pages 443–452. Springer, 2019.

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Andrew Lee, Jaffer H Ghouse, John C Eslick, Carl D Laird, John D Siirola, Miguel A Zamarripa, Dan Gunter, John H Shinn, Alexander W Dowling, Debangsu Bhattacharyya, et al. The IDAES process modeling framework and model libraryFlexibility for process simulation and optimization. *Journal of Advanced Manufacturing and Processing*, page e10095, 2021.

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

References III

Laurens Lueg, Bjarne Grimstad, Alexander Mitsos, and Artur M. Schweidtmann. reluMIP: Open source tool for MILP optimization of ReLU neural networks, 2021. URL `https://github.com/ChemEngAI/ReLU_ANN_MILP`.

Donato Maragno, Holly Wiberg, Dimitris Bertsimas, S Ilker Birbil, Dick den Hertog, and Adejuyigbe Fajemisin. Mixed-integer optimization with constraint learning. *arXiv preprint arXiv:2111.04469*, 2021.

Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems*, 32: 9835–9846, 2019.

Artur M Schweidtmann and Alexander Mitsos. Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications*, 180(3):925–948, 2019.

Thiago Serra, Abhinav Kumar, and Srikumar Ramalingam. Lossless compression of deep neural networks. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 417–430. Springer, 2020.

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

Calvin Tsay, Jan Kronqvist, Alexander Thebelt, and Ruth Misener. Partition-based formulations for mixed-integer optimization of trained ReLU neural networks. *NeurIPS*, 2021.

References IV

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 complete and incomplete neural network verification. *Advances in Neural Information Processing Systems*, 34, 2021.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33, 2020.

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. In *International Conference on Learning Representations*, 2021.

Dominic Yang, Prasanna Balaprakash, and Sven Leyffer. Modeling design and control problems involving neural network surrogates. *arXiv preprint arXiv:2111.10489*, 2021.

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*, 31:4939–4948, 2018.

Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural network verification. *Advances in Neural Information Processing Systems*, 2022a.

References V

Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. A branch and bound framework for stronger adversarial attacks of ReLU networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 26591–26604, 2022b.

Shiqiang Zhang, Juan S. Campos, Christian Feldmann, David Walz, Frederik Sandfort, Miriam Mathea, Calvin Tsay, and Ruth Misener. Optimizing over trained GNNs via symmetry breaking. In *NeurIPS*, 2023.

Imperial College London
Carnegie Mellon University
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY
Sandia National Laboratories

Optimal decision-making with trained NN embedded

Shiqiang Zhang, Juan S Campos, Christopher Hojny, Francesco Ceccon, Jordan Jalving, Joshua Haddad, Alexander Thebelt, Calvin Tsay, Carl D Laird, Ruth Misener

Funding EPSRC EP/P016871/1 & EP/T001577/1
Sandia LDRD program
Institute for the Design of Advanced Energy Systems

13 September 2024

Paper Ceccon*, Jalving*, Haddad, Thebelt, Tsay, Laird†, Misener†, *arXiv*, 2022.