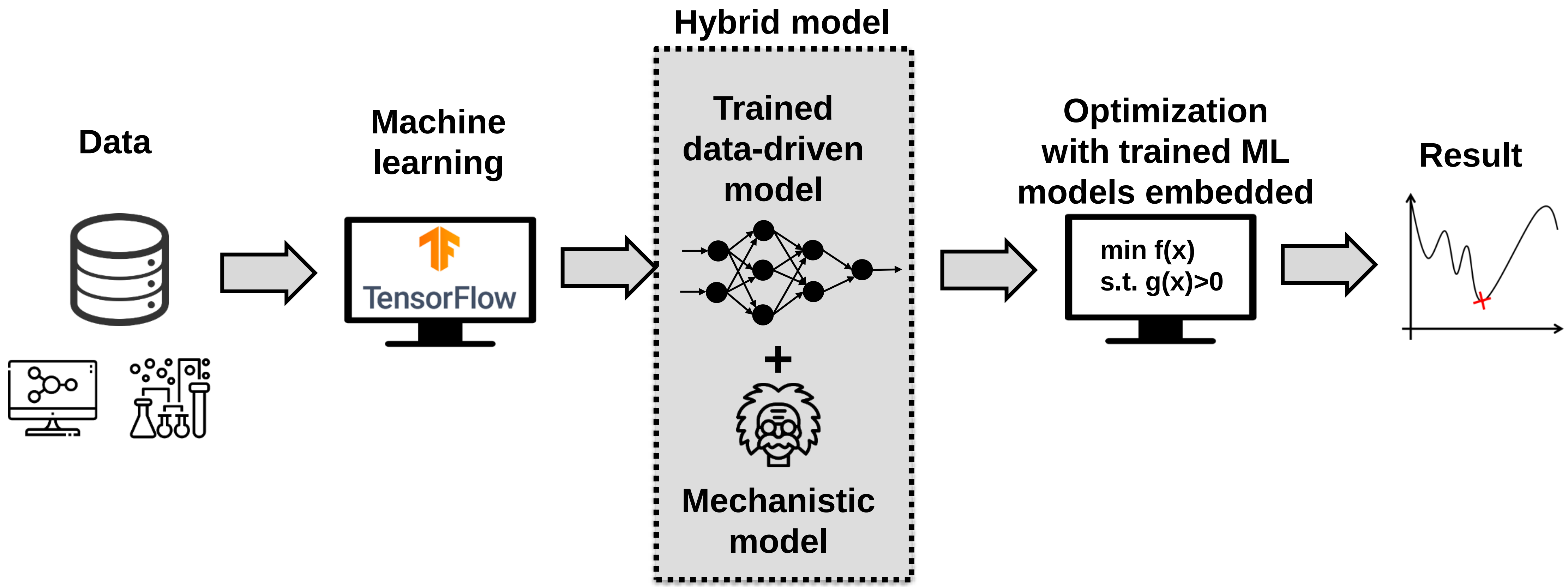
reluMIP: Open-Source Tool for MILP Optimization of ReLU Neural Networks

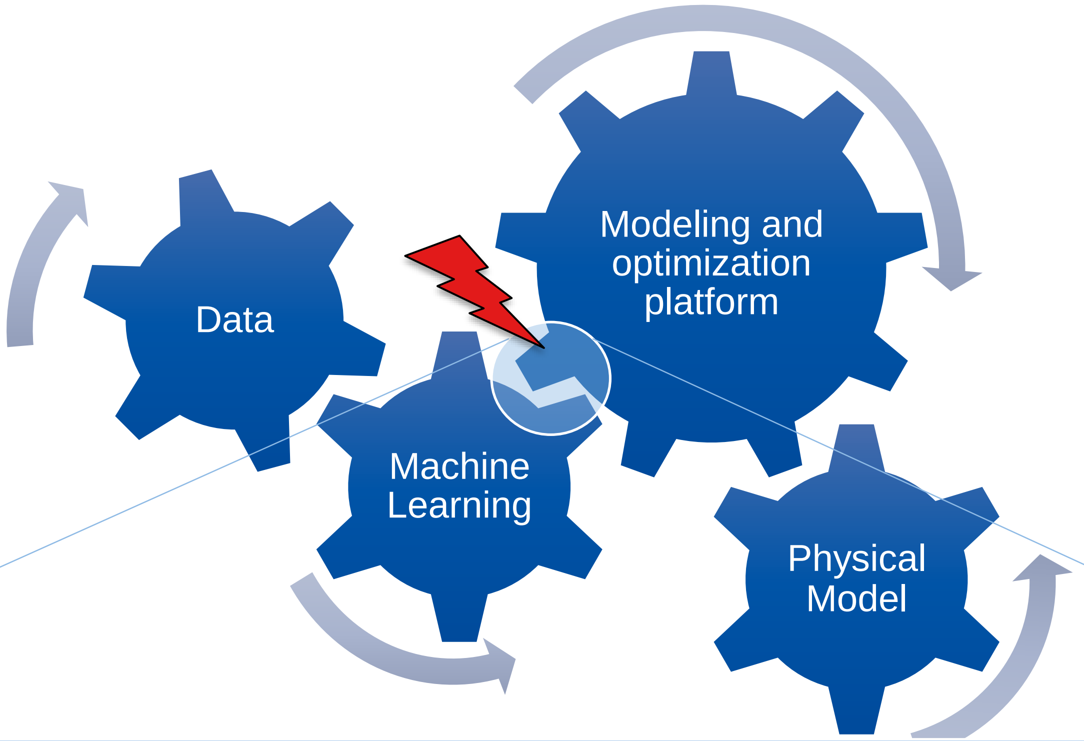
Laurens Lueg, Bjarne Grimstad, Alexander Mitsos, Artur M. Schweidtmann

October 26, 2021

Machine learning and optimization approach in chemical engineering



We aim for an integration of ML and optimization tools



TensorFlow

Keras

How to formulate ML model in optimization problem?

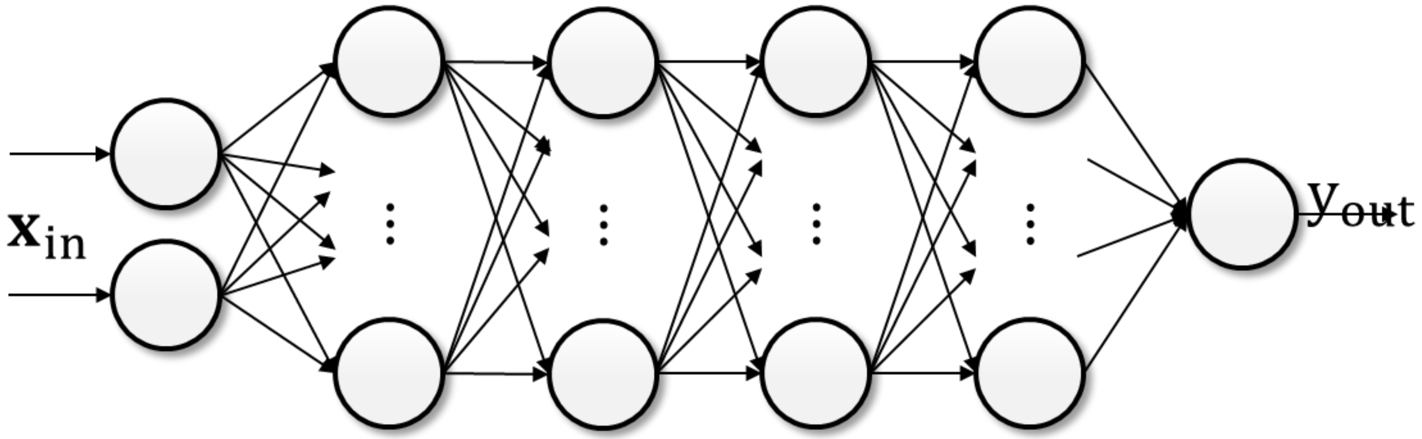
PYOMO

GUROBI OPTIMIZATION

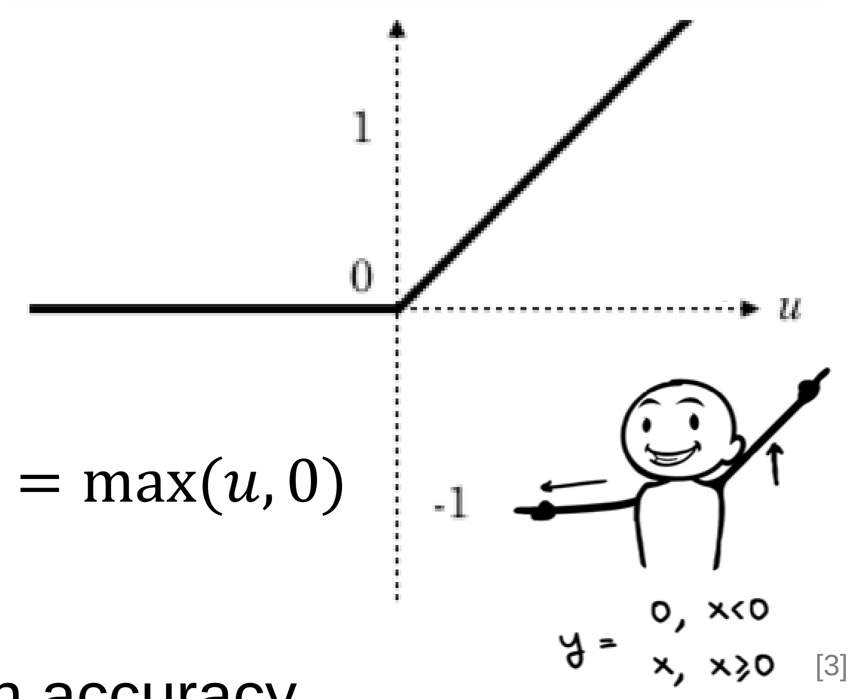
ChemEngAI

Artificial neural networks with ReLU activation[1]

"ReLU" = Rectified Linear Unit



$$f_{ReLU}(u) = \max(u, 0)$$



[3]

→ model any continuous piecewise linear function[2]
→ approximate any continuous function to any given accuracy

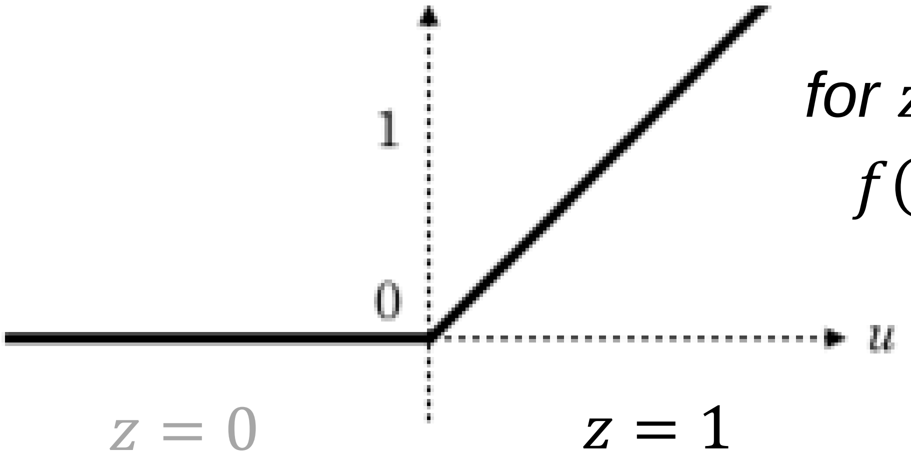
[1] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. JMLR Workshop and Conference Proceedings. [2] Arora, R., Basu, A., Mianjy, P., Mukherjee, A.: Understanding deep neural networks with rectified linear units. arXiv preprint arXiv:1611.01491 (2016) [3] Activation function dance: https://sefiks.com/2020/02/02/dance-moves-of-deep-learning-activation-functions

Optimization over trained ReLU artificial neural networks as Mixed-Integer Linear Problem

$$
\begin{aligned}
&\min f_{N+1}(\mathbf{x}_N) \\
&\text{s.t.}\quad x_{i,j} \geq f(\mathbf{x}_{\mathrm{i}-1}) \\
&\qquad x_{i,j} \leq f(\mathbf{x}_{i-1}) - M_{i,j}^{-}(1 - z_{i,j}) \qquad \forall_{i=1\ldots N,\, j=1\ldots m_i} \\
&\qquad x_{i,j} \leq M_{i,j}^{+} z_{i,j} \\
&\qquad (x_{i,j}, z_{i,j}) \in [0, M_{i,j}^{+}] \times \{0,1\} \\
&\qquad x_{0,j} \in [L_{0,j}, U_{0,j}], \qquad \forall_{j=1\ldots m_0}
\end{aligned}
$$

ChemEngAI

Optimization over trained ReLU artificial neural networks



for $z = 1$

$$f(x) = x$$

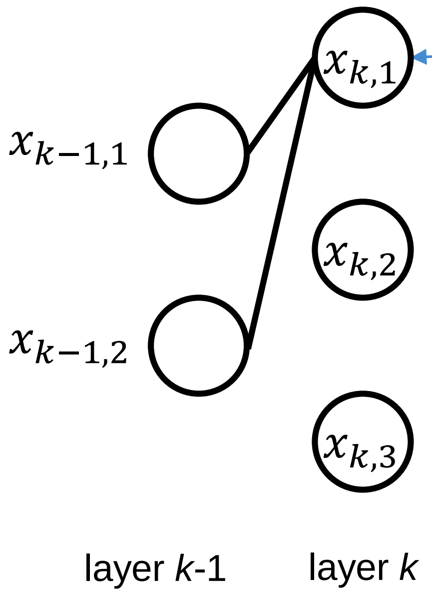
$$
\begin{aligned}
&\min f_{N+1}(\mathbf{x}_N) \\
&\text{s.t.} \quad x_{i,j} \geq f(\mathbf{x}_{i-1}) \\
&\qquad x_{i,j} \leq f(\mathbf{x}_{i-1}) - M^-_{i,j}(1 - z_{i,j}) \qquad \forall_{i=1\ldots N, j=1\ldots m_i} \\
&\qquad x_{i,j} \leq M^+_{i,j} z_{i,j} \\
&\qquad (x_{i,j}, z_{i,j}) \in [0, M^+_{i,j}] \times \{0,1\} \\
&\qquad x_{0,j} \in [L_{0,j}, U_{0,j}], \qquad \forall_{j=1\ldots m_0}
\end{aligned}
$$

- **'Big-M' formulation**[1,2] : not sharp
  - progressive bounds tightening[2,3,4]
- Ideal formulations[1] : additional constraints or auxiliary variables
- Partition-Based Formulations[5]

[1] Anderson et. al (2019) https://arxiv.org/abs/1811.01988v2 [2] Tjeng et. al (2019) http://arxiv.org/pdf/1711.07356v3 [3] Fischetti et. al (2018) http://doi.org/10.1007/s10601-018-9285-6 [4] Grimstad et. al (2019) http://arxiv.org/pdf/1907.03140v3 [5] Tsay, C., Kronqvist, J., Thebelt, A., & Misener, R. (2021). arXiv preprint arXiv:2102.04373.

Optimization-based bound tightening (OBBT)[1,2]

- **Solve two subproblems for each node** to determine M^-/M^+



subproblems node $(k, 1)$:

$M^+_{k,1} =$ max $f_{k,1}(\boldsymbol{x_{k-1}})$

$M^-_{k,1} =$ min $f_{k,1}(\boldsymbol{x_{k-1}})$

Big-M constraints for node $(k, 1)$:

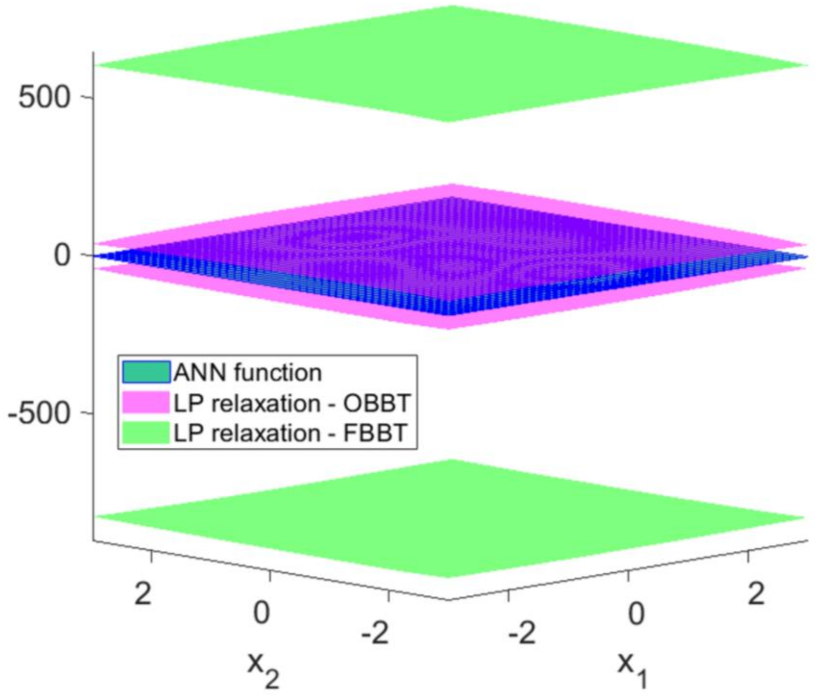
$x_{k,1} \geq f(\mathbf{x_{k-1}})$

$x_{k,1} \leq f(\mathbf{x_{k-1}}) - M^-_{k,1}(1 - z_{k,1})$

$x_{k,1} \leq M^+_{k,1} z_{k,1}$

$(x_{k,1}, z_{k,1}) \in [0, M^+_{k,1}] \times \{0,1\}$

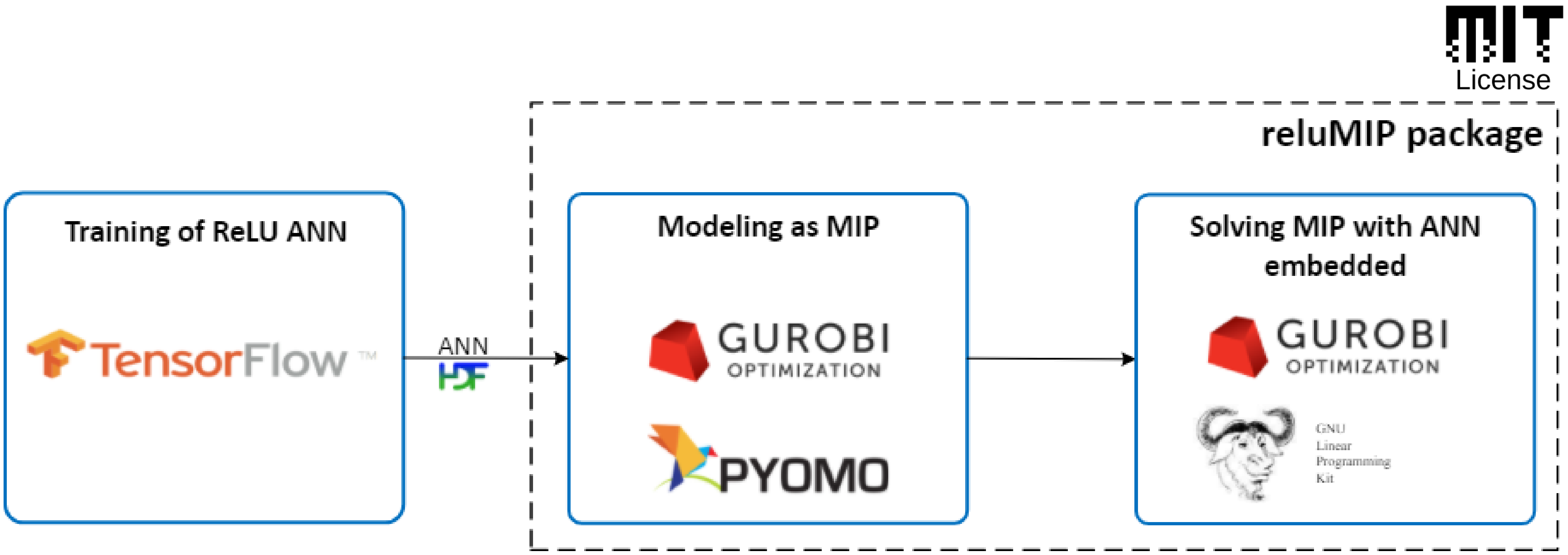
1. Solve to optimality
2. Solve with time limit
3. Solve LP relaxation



Root LP relaxation obtained by OBBT compared to Interval Arithmetic (FBBT)

[1] Tjeng et. al (2019) http://arxiv.org/pdf/1711.07356v3 [2] Grimstad et. al (2019) http://arxiv.org/pdf/1907.03140v3

Open-source software reluMIP

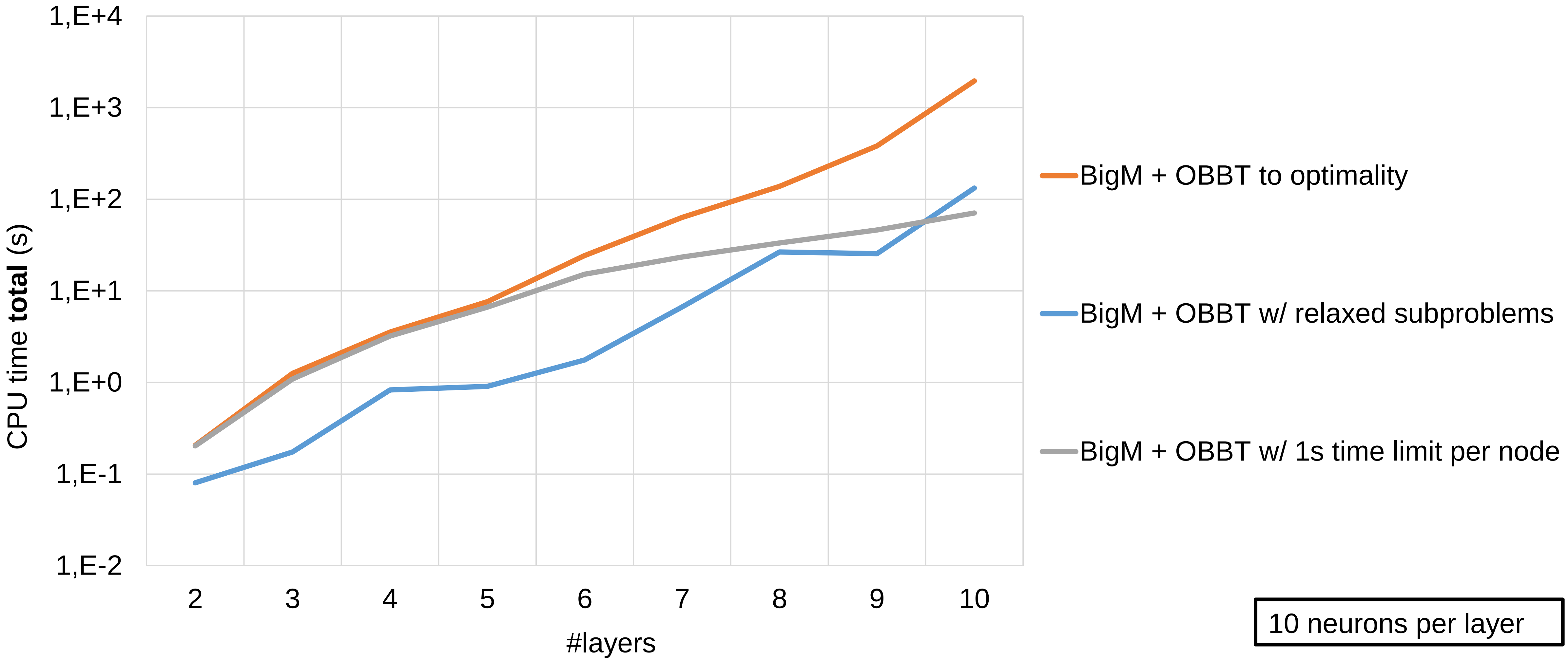


Using reluMIP in Python

```
tf_model  = tf.keras.models.load_model('data/peaks_3x10.h5')
opt_model = gurobipy.Model()
ann_model = reluMIP.AnnModel(tf_model=tf_model, modeling_language='GUROBI')

ann_model.connect_network_input(opt_model, input_vars)
ann_model.connect_network_output(opt_model, output_vars)

ann_model.embed_network_formulation(bound_tightening_strategy='LP')

opt_model.setObjective(output_vars[0], grb.GRB.MINIMIZE)

opt_model.optimize()
```

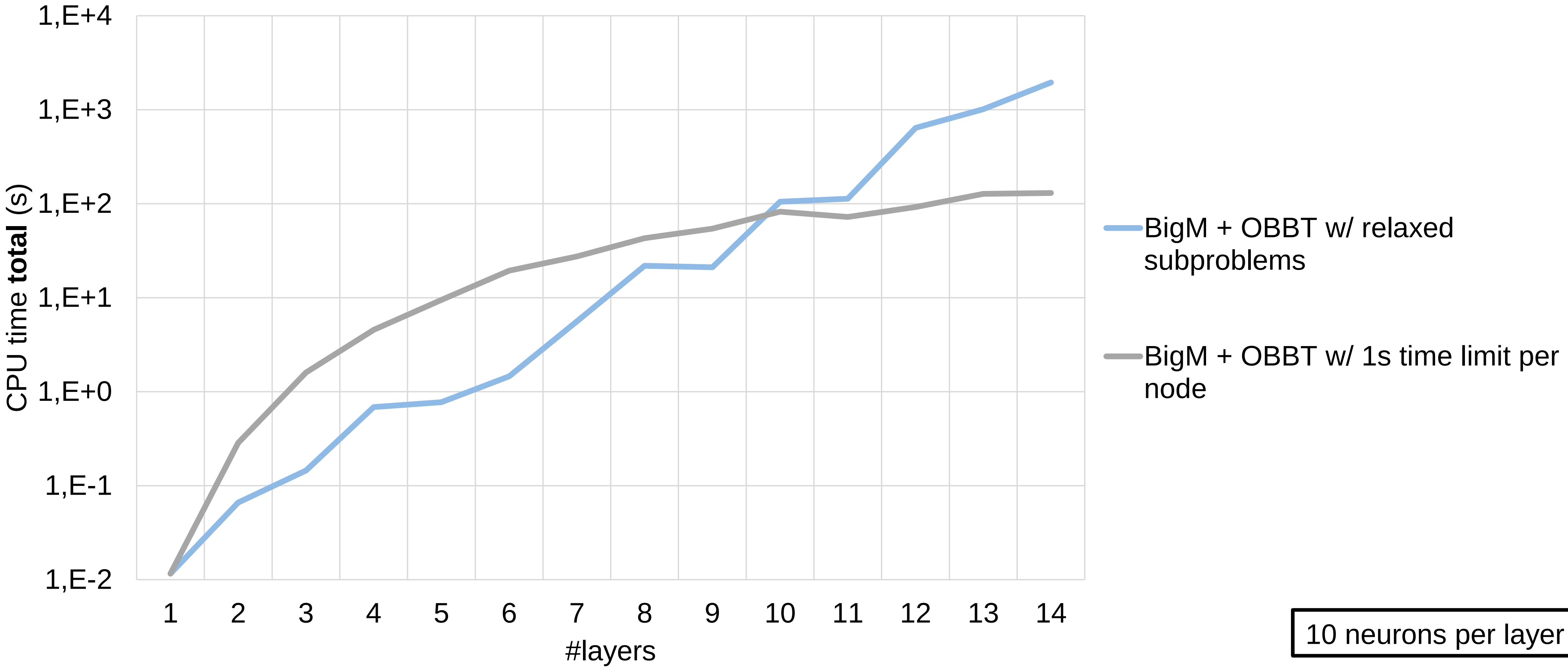

ChemEngAI

MILP vs. NLP – deep networks



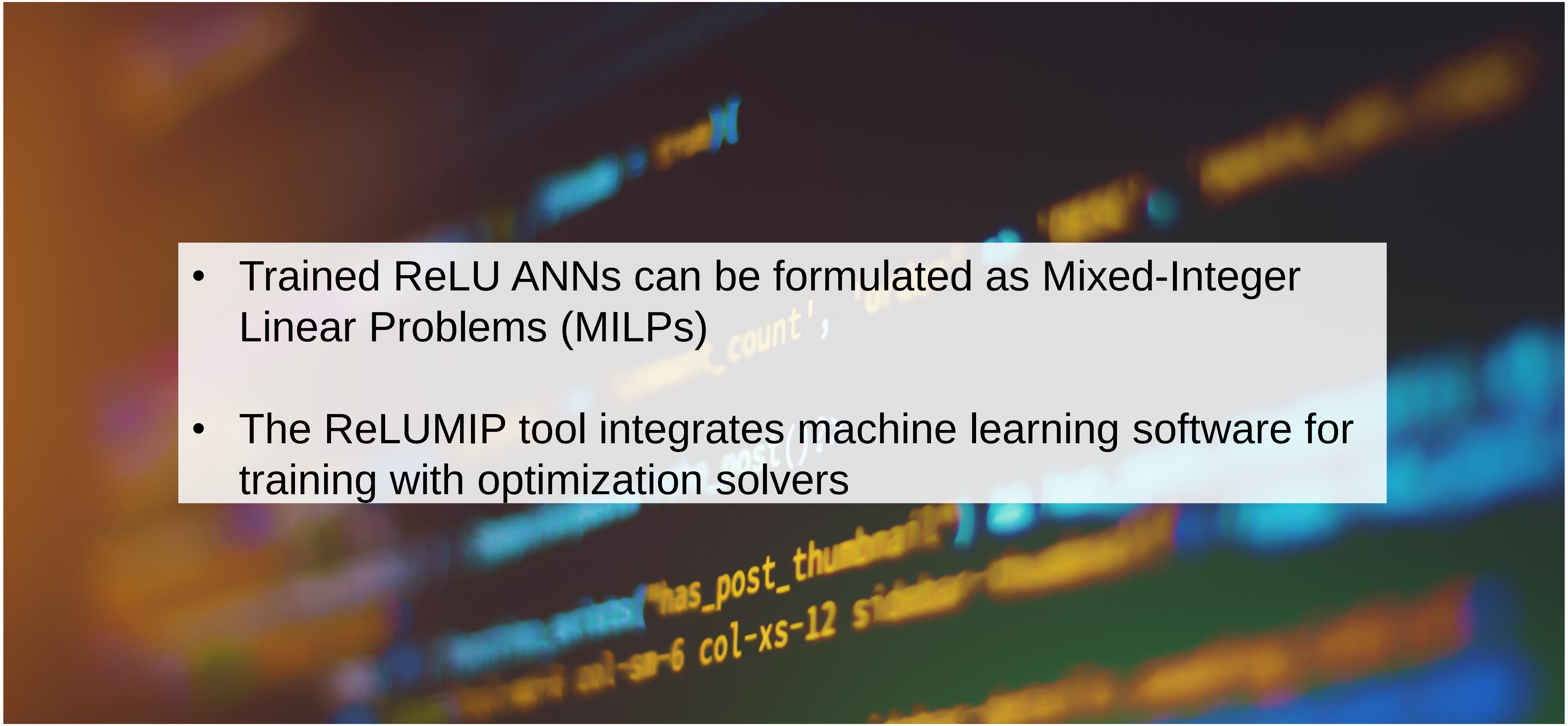
10 neurons per layer

MILP vs. NLP – deep networks



10 neurons per layer

Conclusions



- Trained ReLU ANNs can be formulated as Mixed-Integer Linear Problems (MILPs)
- The ReLUMIP tool integrates machine learning software for training with optimization solvers

ChemEngAI

The authors

Laurens Lueg

ETH

Bjarne Grimstad

NTNU
Norwegian University of Science and Technology

Alexander Mitsos

RWTH AACHEN UNIVERSITY

Artur Schweidtmann

TU Delft
ChemEngAI