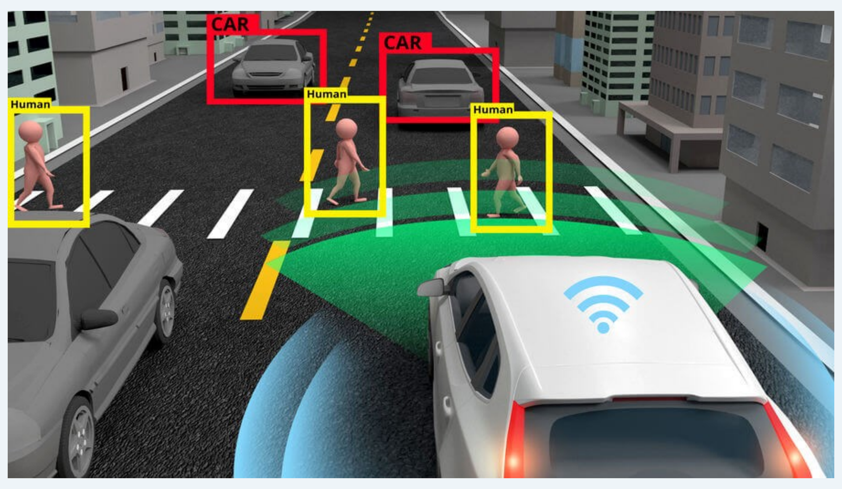
Semantic VS Syntactic Abstraction of Neural Networks

Stefanie Mohr
mohr@in.tum.de
Supervisors: Jan Křetínský & Debarghya Ghoshdastidar
Collaborators: Calvin Chau

CONVEY

Motivation

With the rise of Neural Networks, they are also applied in **safety-critical systems** (e.g. autonomous cars). It is important to prove their safety, however this definition may be. Since the **verification** is currently not even scalable to small NNs, we focus on **abstraction**, i.e. a method to reduce the size of the verification problem.



https://www.rsipvision.com/adas-sensors-rgb-cameras/

Relation to ConVeY

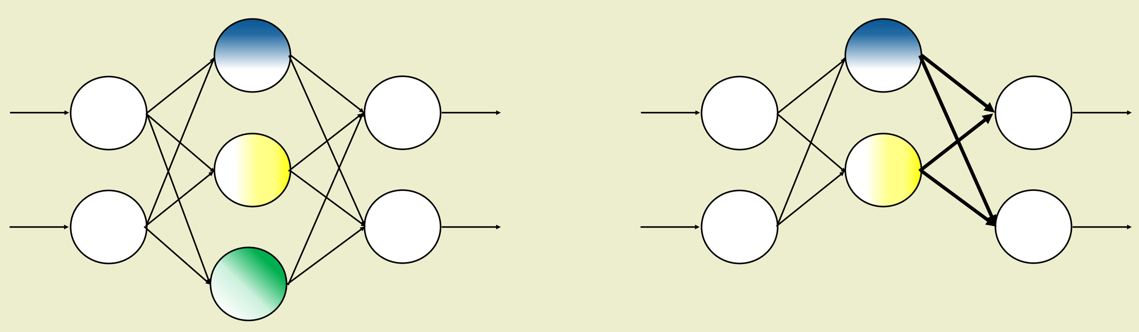
Neural Networks are **continuous** functions and they are often part of Cyber-Physical-Systems. These systems need to be verified or certified. Thus, we also need the **verification of the Neural Networks**. Additionally, there is a wide are of life-long learning and adaptation of Neural Networks, so it might be necessary to have a **continuous verification** of them.

Idea

Goal
Verification of NNs

Problem
Scalability

Solution
Abstraction



Get semantic information of the neurons

Replace neurons with linear combinations of other neurons

Guarantees

Theorem for Relation between Original and Abstraction

The difference between the original $N^L(\mathbf{x})$ and the abstraction $\tilde{N}^L(\mathbf{x})$ can be bounded by

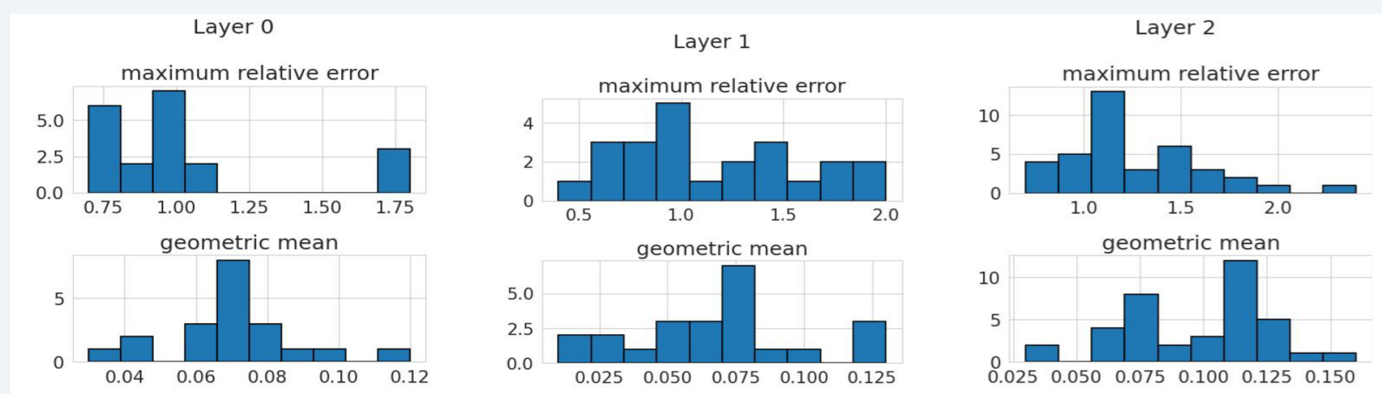
$$\|\tilde{N}^L(\mathbf{x}) - N^L(\mathbf{x})\| \leq b(1-a^{L-1})/(1-a)$$

$a = \lambda(\|W\| + \eta)$
$b = \lambda\|W\|\epsilon$
$\lambda^{(l)}$ *the Lipschitz-constant of the activation function in* l
$\lambda = \max_l \lambda^{(l)}$
$\|W\| = \max_l \|W^{(l)}\|_1$
$\eta = \max_l \eta^{(l)}$
$\epsilon = \max_l \epsilon^{(l)}$

assuming that for all layers $l \in \{1, ..., L\}$ *and for all inputs* $\mathbf{x} \in X$*, we have*

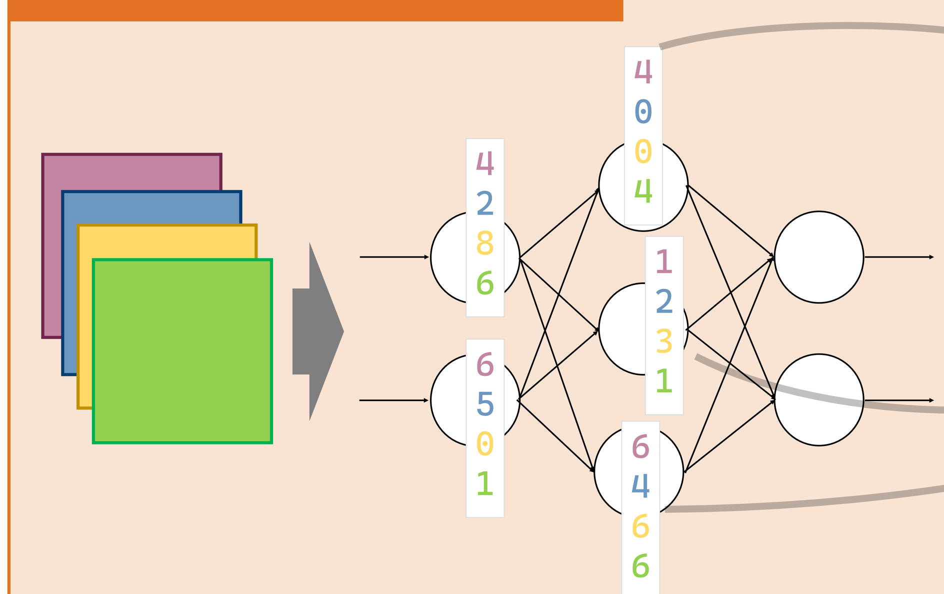
- *for* $i \in I^{(l)}$: $|z_i^{(l)}(\mathbf{x}) - \sum_{j \in B^{(l)}} \alpha_{i,j}^{(l)} z_j^{(l)}(\mathbf{x})| \leq \epsilon^{(l)}$
- $|\sum_{i \in I^{(l)}} W_{*,i}^{(l)} \sum_{t \in B^{(l)}} \alpha_{i,t}^{(l)}| \leq \eta^{(l)}$

Error Bound Results



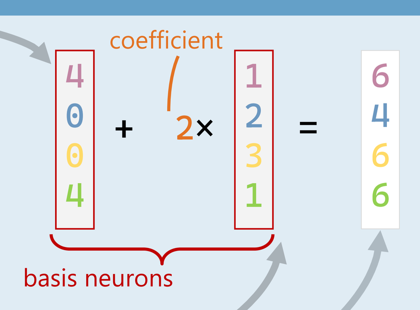
Error of the neurons on the **test-set** of an MNIST network with 3x100 neurons, reduced by 30%.

Semantic Information



Based on an IO-set X, calculate the activation values of the neurons We use the inputs $x \in X$ and feed them to the network. We then capture the activation values (=outputs) of the neurons.

Linear Combination



How to find the **basis** neurons?

- **Greedy** method: Iterate through all neurons in the network and try to replace them. Find the one with the smallesr replacement error
- **Heuristic** method: Sort all neurons with descending variance on the training inputs. Choose the first neurons.

How to find the **coefficients**?

- **Linear Program**: Include slack variables because there is no perfect solution.
- **Orthogonal Projection**: Project neurons that should be replaced in the space of the basis neurons.

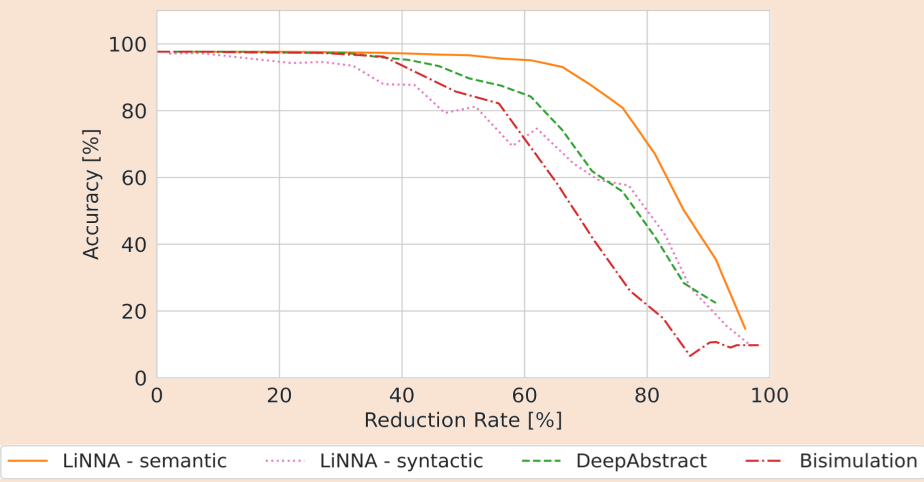
Tool

Linear
Neural
Network
Abstraction

Syntactic VS Semantic

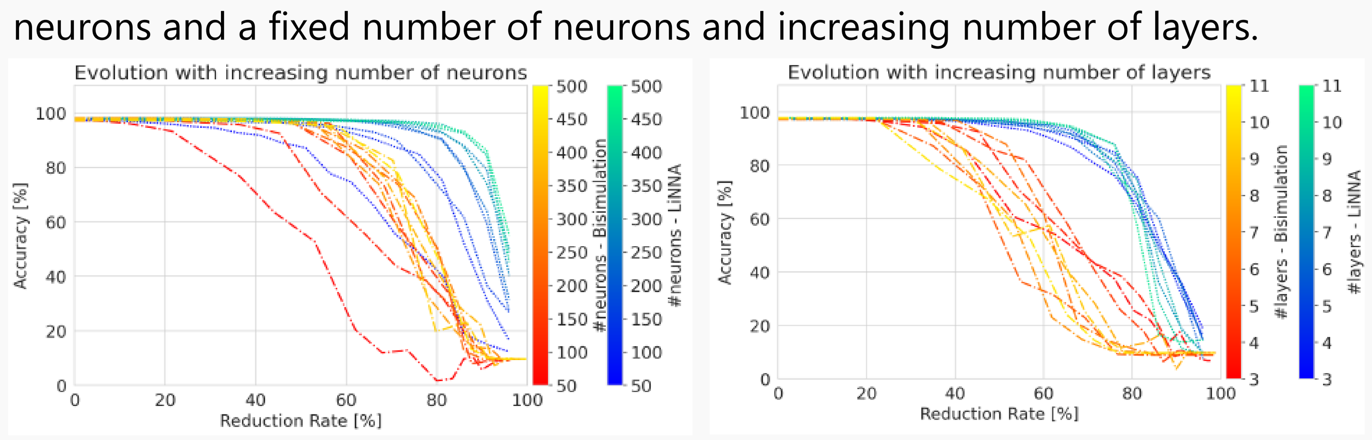
DeepAbstract [1]: Clustering of neurons based on the **semantic information** (many by one)

Bisimulation [2]: Replacing neurons based on the **syntactic information**, i.e. their weights and biases (many by many)



Evolution

This is a comparison of the behavior of LiNNA and the bisimulation on a Neural Network, for a fixed number of layers and increasing number of neurons and a fixed number of neurons and increasing number of layers.

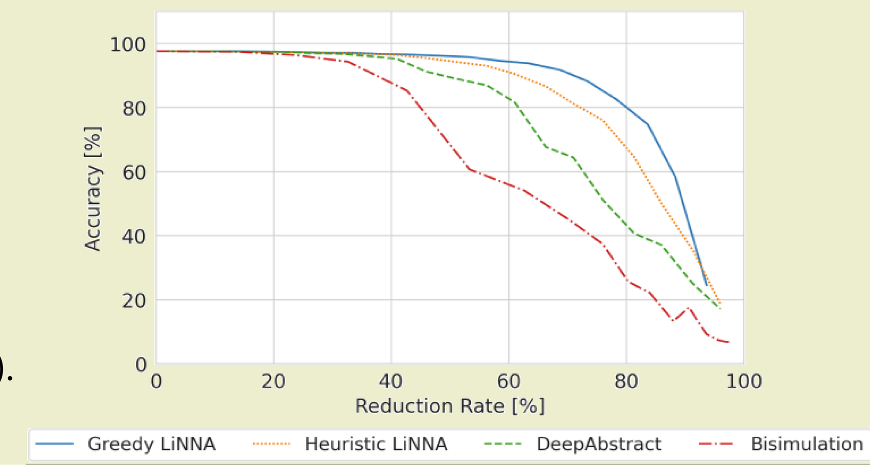


Comparison

The **greedy approach** performs best in terms of accuracy.

LiNNA has a **higher accuracy** than any other approach (even for the heuristic based approach).

Runtime is **magnitudes faster** than the **predecessor** DeepAbstract.



	Greedy LiNNA	Heuristic LiNNA	DeepAbstract	Bisimulation
Mean Runtime [s]	55-199	2-3	187-2420	1-2

[0] Chau, C., Křetínský, J., Mohr, S. Semantic VS Syntactic Linear Abstraction and Refinement of Neural Networks. Under Review.
[1] Ashok, P., Hashemi, V., Křetínský, J., Mohr, S. (2020). DeepAbstract: Neural Network Abstraction for Accelerating Verification. In: Hung, D.V., Sokolsky, O. (eds) Automated Technology for Verification and Analysis. ATVA 2020. Lecture Notes in Computer Science(), vol 12302. Springer, Cham.
[2] Prabhakar, P. (2022). Bisimulations for Neural Network Reduction. In: Finkbeiner, B., Wies, T. (eds) Verification, Model Checking, and Abstract Interpretation. VMCAI 2022. Lecture Notes in Computer Science(), vol 13182. Springer, Cham. https://doi.org/10.1007/978-3-030-94583-1_14