Principled Rewriting of ONNX Operators for Reluctant Solvers

A proof of concept for the Argmax operator

Guilhem Ardouin
Julien Girard-Satabin
Alban Grastien

CEA LIST

2026-07-24

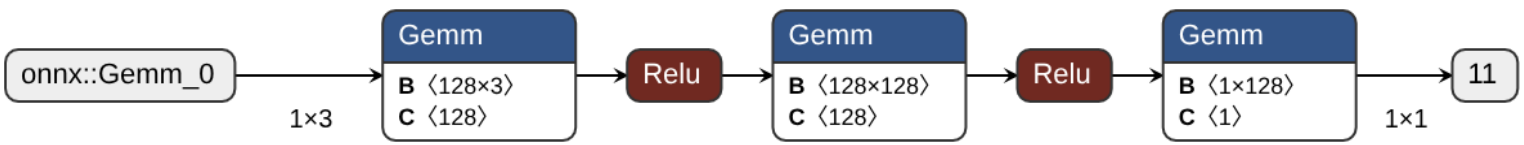
This work was supported partially by CARNOT CHAINS and PEPR SAIF

Motivation

Subject of interest: formal verification of Neural Networks by multiple solvers

Multiple solvers: VNN-COMP, stronger confidence from different, concording answers (and people from certification agencies are used to that!)

Open Neural Network eXchange (ONNX) format



- Dataflow representation of neural network computations
- Nodes are operators, semantics loosely defined[1] from standard *https://onnx.ai*
- 196 operators encoding traditional machine learning pipelines
- several provers have different support range for ONNX operators

[1] ask me about the Conv operator during break for fun and profit

ONNX support discrepencies

Solver	Supported ops (/196)
nnenum [1]	17
$\alpha - \beta - \mathrm{CROWN}$ [2]	22
Marabou [3]	16
Maraboupy [3]	24
PyRAT [4]	54

Mutually exclusive support of operators!

Do we have to write custom program transformations for provers?

Existing work

DNNV ([5], [6]): ONNX simplifications

ONNX runtimes and `onnxsimplifier` perform compiler optimizations like constant-folding which results in simpler, faster models

$\alpha - \beta - \mathrm{CROWN}$ [2], Marabou [3] , PyRAT [4] have their own (often undocumented) heuristics on simplificatons

The ArgMax

An ubiquitous operator

Case-study

Sound rewriting the ArgMax (Argument of Maximum value) operator

Case-study

Sound rewriting the ArgMax (Argument of Maximum value) operator

1. Ubiquituous in classification tasks (selecting the best performing class)
2. Useful for rewrites for hyper-properties [7]
3. Can be decomposed onto a subgraph of simpler ONNX operators

Often absent from VNN-COMP benchmarks, yet often found in production models

ArgMax semantic

ArgMax

Given a tensor t with shape $\mathrm{sh} = [d_0, d_1, ..., d_{n-1}]$
ArgMax returns a tensor a of shape $[d_0, d_1, ..., d_{n-2}, 1]$ such that if $a[i_1, ..., i_{n-2}, 1] = j$ then $t[i_1, ..., i_{n-2}, j] > t[i_1, ..., i_{n-2}, k] \forall k \neq j$

Example with a vector

$$v = \begin{pmatrix} 3 \\ 4 \\ -6 \\ 2 \end{pmatrix} \qquad \mathrm{sh}(v) \overset{\text{def}}{=} (4)$$

Example with a vector

$$v = \begin{pmatrix} 3 \\ \boxed{4} \\ -6 \\ 2 \end{pmatrix} \qquad \mathrm{sh}(v) \overset{\text{def}}{=} (4) \qquad a(v) \overset{\text{def}}{=} [2]$$

Example with a matrix

$$m = \begin{pmatrix} 3 & 3 \\ 1 & 3 \\ 8 & 6 \end{pmatrix} \qquad \mathrm{sh}(m) \stackrel{\mathrm{def}}{=} (3, 2)$$

Example with a matrix

$$m = \begin{pmatrix} 3 & 3 \\ 1 & 3 \\ 8 & 6 \end{pmatrix} \qquad \mathrm{sh}(m) \stackrel{\text{def}}{=} (3, 2)$$

$$a(m)_{j=0} \stackrel{\text{def}}{=} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Example with a matrix

$$m = \begin{pmatrix} 3 & 3 \\ 1 & 3 \\ 8 & 6 \end{pmatrix}$$

$$\mathrm{sh}(m) \stackrel{\text{def}}{=} (3, 2)$$

$$a(m)_{j=0} \stackrel{\text{def}}{=} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Example with a matrix

$$m = \begin{pmatrix} 3 & 3 \\ 1 & 3 \\ 8 & 6 \end{pmatrix}$$

$$\mathrm{sh}(m) \stackrel{\text{def}}{=} (3, 2)$$

$$a(m)_{j=0} \stackrel{\text{def}}{=} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Example with a matrix

$$m = \begin{pmatrix} 3 & 3 \\ 1 & 3 \\ 8 & 6 \end{pmatrix}$$

$$\mathrm{sh}(m) \stackrel{\text{def}}{=} (3, 2)$$

$$a(m)_{j=0} \stackrel{\text{def}}{=} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

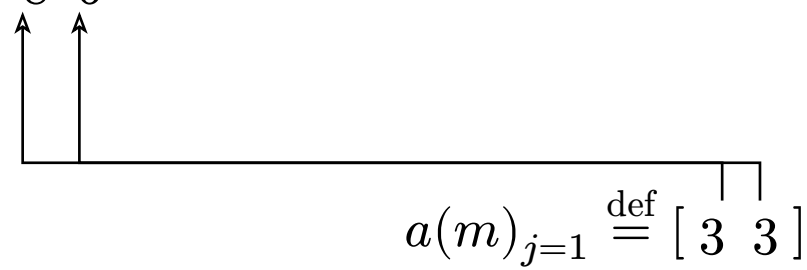
Example with a matrix

$$m = \begin{pmatrix} 3 & 3 \\ 1 & 3 \\ 8 & 6 \end{pmatrix} \qquad \mathrm{sh}(m) \stackrel{\text{def}}{=} (3, 2)$$

$$a(m)_{j=1} \stackrel{\text{def}}{=} [\, 3 \;\; 3 \,]$$

Example with a matrix

$$m = \begin{pmatrix} 3 & 3 \\ 1 & 3 \\ 8 & 6 \end{pmatrix} \qquad \mathrm{sh}(m) \stackrel{\text{def}}{=} (3, 2)$$



$$a(m)_{j=1} \stackrel{\text{def}}{=} [\, 3 \;\; 3 \,]$$

Tensors

Tensor t is a multi-dimensional array, of shape $\mathrm{sh}(\mathrm{t}) = (\mathrm{dim}_1, ..., \mathrm{dim}_r)$; $r = \mathrm{len}(\mathrm{sh})$ is the rank of t, dim_l is the size of the $l - \mathrm{th}$ dimension.

Vector of positions $\mathbf{i} = [i_1, ..., i_r] \stackrel{\mathrm{def}}{=} [\mathrm{dim}_1] \times ... \times [\mathrm{dim}_t]$

Sound decomposition

Idea of the decomposition

Use basic ONNX operators to implement pairwise-comparison for each couple of coordinates given the target axis

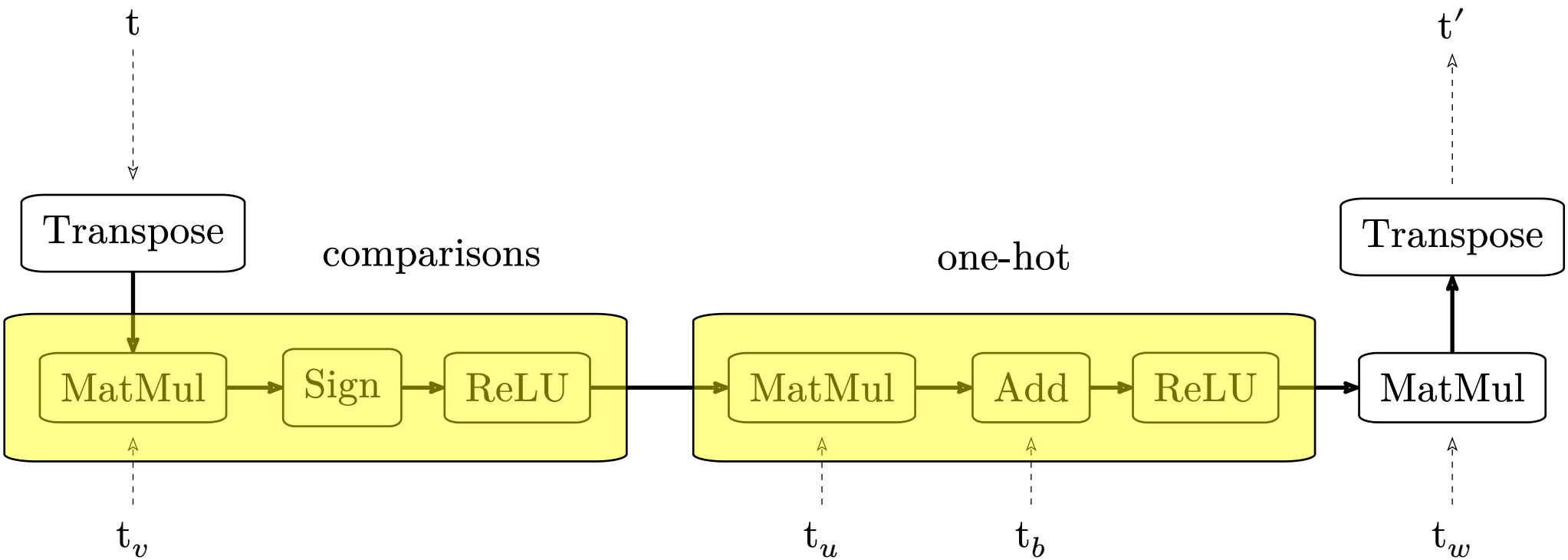
1. select the dimension of t to work on;
2. substract elements pairwise;
3. get the sign of substractions;
4. compute 1 if the sign is positive, 0 otherwise;

Some basic ONNX operators

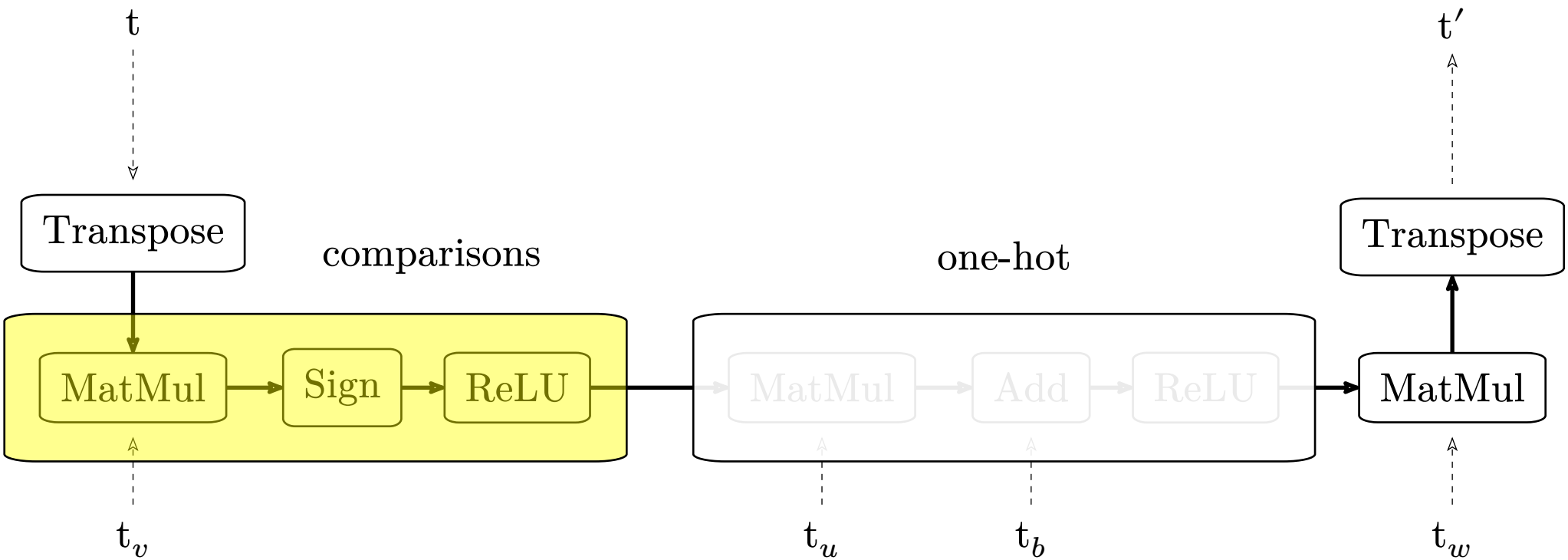
Given an input tensor t:

1. Const_c returns c;
2. Add_c is element-wise addition of t and c;
3. Sign is element-wise sign of t, returns 1, 0, -1 if the element is strictly positive, zero, or strictly negative, respectively;
4. ReLU is element-wise application of function $x \mapsto \max(0, x)$;
5. $\mathrm{Transpose}_p$ permutates indices in t;
6. MatMul_A is matrix multiplication At.

Implementing the decomposition



Implementing the decomposition



Comparison implementation

1. transposition using swapping of indices $p_{l,r}$;
2. compute $C^2_{\dim_l}$, and associate elements of vector u with values of said combinations in index ξ;
3. sub-network ensures $u[\xi(\{j,k\})] = 1$ if $v[k]$ higher than $v[j]$, and 0 otherwise
   <= you can do that by computing $v[k] - v[j]$

Comparison implementation

$v = \begin{pmatrix} 3 & 4 & -6 & 2 \end{pmatrix}$

$\mathrm{sh}(m) \stackrel{\mathrm{def}}{=} (1, 4)$

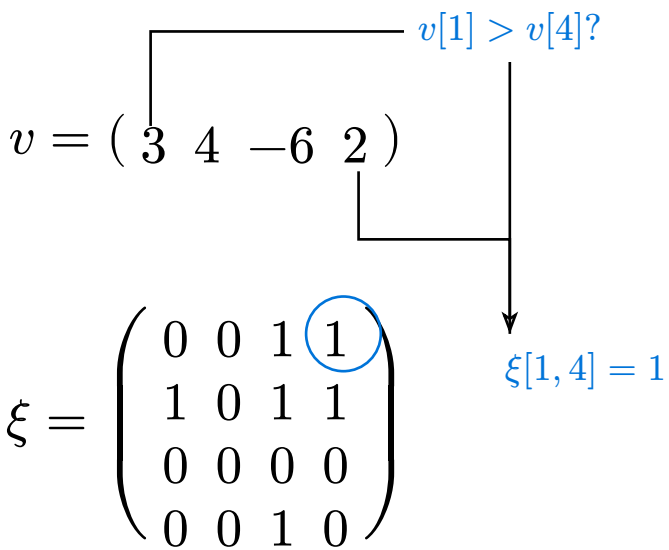
Comparison implementation

$$v = \begin{pmatrix} 3 & 4 & -6 & 2 \end{pmatrix}$$

$$\mathrm{sh}(m) \stackrel{\text{def}}{=} (1, 4)$$

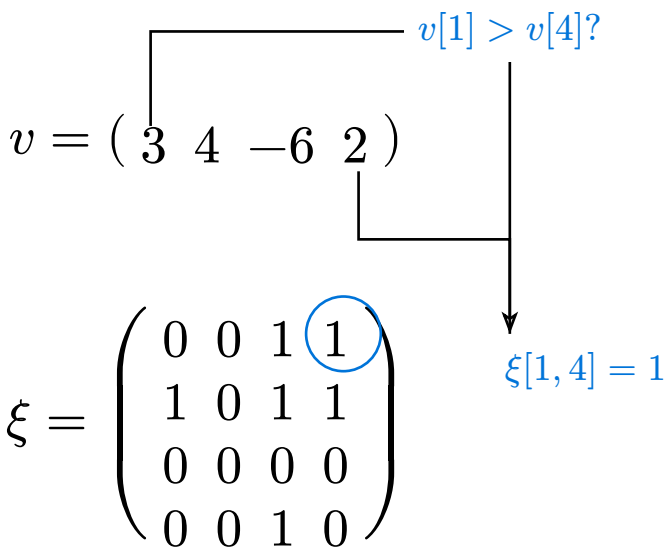
$$\xi = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Comparison implementation



$$\mathrm{sh}(m) \stackrel{\text{def}}{=} (1, 4)$$

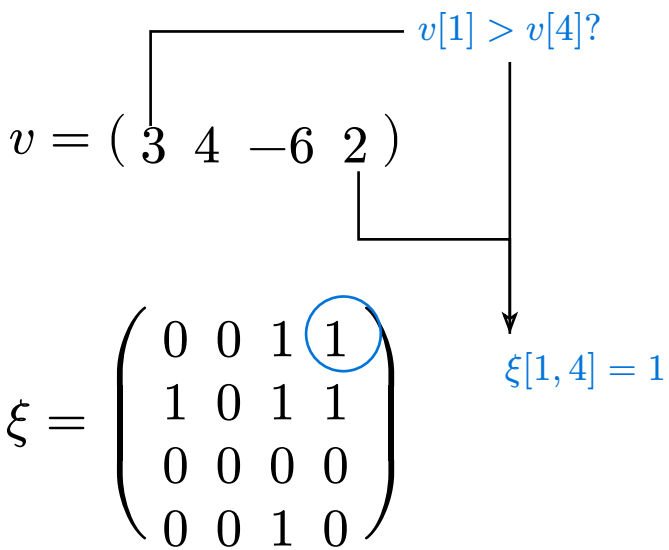
Comparison implementation



$$\mathrm{sh}(m) \stackrel{\text{def}}{=} (1, 4)$$

$$\mathrm{MatMul}\ (\mathrm{v}, \mathrm{t}_v) = \begin{pmatrix} \underbrace{-1}_{v[1]<v[2]} & \underbrace{1}_{v[1]>v[3]} & \underbrace{1}_{v[1]>v[4]} & \underbrace{1}_{v[2]>v[3]} & \underbrace{1}_{v[2]>v[4]} & \underbrace{-1}_{v[3]<v[4]} \end{pmatrix}$$

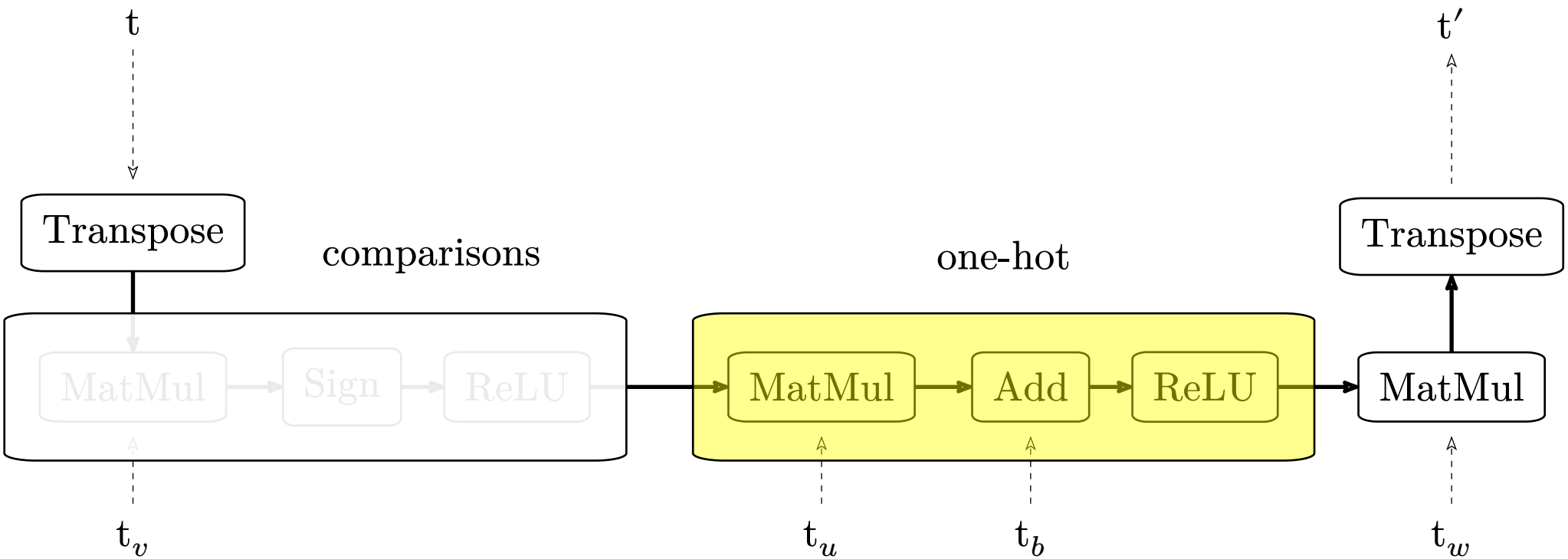
Comparison implementation



$$\mathrm{sh}(m) \stackrel{\text{def}}{=} (1, 4)$$

$$u = (\ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ )$$

One-hot implementation



One-hot implementation

w is a one-hot vector where $w[\hat{j}]$ if $v[\hat{j}]$ is the max value among the dimension, and all others are 0; obtained by computing differences that will be negative

Some experimental results

ArgMax for confidence-based properties

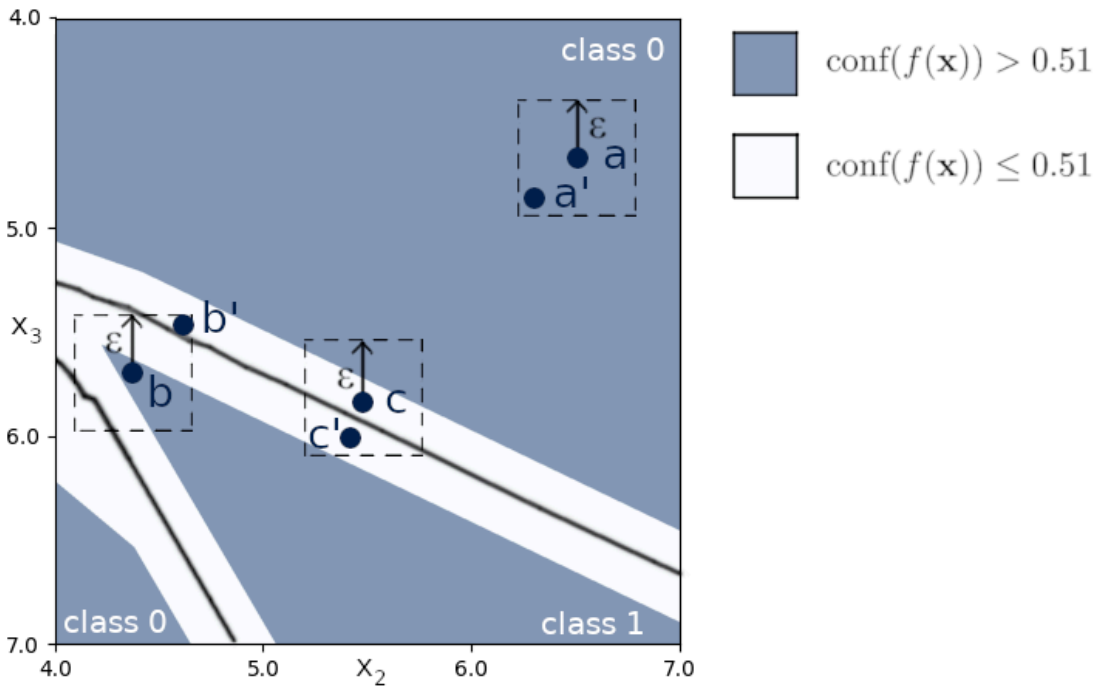


Figure 2: Proposed in [8], refined by [7], needs ArgMax for comparing outputs between two execution traces

ArgMax for confidence-based properties

For this property, ArgMax is instanciated as a logical predicate

ArgMax for confidence-based properties

For this property, ArgMax is instanciated as a logical predicate

Quadratic in terms of output dimension 😭

ArgMax for confidence-based properties

For this property, ArgMax is instanciated as a logical predicate

Quadratic in terms of output dimension 😭

But the rewritten ArgMax can be used! 😤

ArgMax for confidence-based properties

		Validity	LOGICAL	ARGMAX	Time difference	
COMPAS	($\kappa = 0.9$)	✓	17.4s	12.3s	−5.1s	(−29.3%)
COMPAS	($\kappa = 0.8$)	✓	18.8s	13.5s	−5.3s	(−28.2%)
COMPAS	($\kappa = 0.7$)	✓	38.3s	30.0s	−8.3s	(−21.7%)
COMPAS	($\kappa = 0.6$)	✓	564.6s	493.7s	−70.9s	(−12.6%)
COMPAS	($\kappa = 0.2$)	✗	63.8s	34.2s	−29.6s	(−46.4%)
COMPAS	($\kappa = 0.1$)	✗	64.0s	34.4s	−29.6s	(−46.2%)
Adult	($\kappa = 0.9$)	✓	13.4s	13.5s	+0.1s	(+0.7%)
Adult	($\kappa = 0.8$)	✓	21.8s	23.0s	+1.2s	(+5.5%)
Adult	($\kappa = 0.7$)	✓	123.2s	138.4s	+15.2s	(+12.3%)
Adult	($\kappa = 0.2$)	✗	78.2s	86.7s	+8.5s	(+10.9%)
Adult	($\kappa = 0.1$)	✗	78.6s	86.8s	+8.2s	(+10.4%)
Industrial	($\kappa = 0.999$)	✓	87.8s	61.9s	−25.9s	(−29.5%)
Industrial	($\kappa = 0.99$)	✓	274.6s	188.1s	−86.5s	(−31.5%)
Industrial	($\kappa = 0.9$)	✓	2058.9s	1389.5s	−669.4s	(−32.5%)
Industrial	($\kappa = 0.2$)	✗	115.9s	68.8s	−47.1s	(−40.6%)
Industrial	($\kappa = 0.1$)	✗	116.2s	68.6s	−47.6s	(−41.0%)

Key takeaway

- ArgMax principled rewriting and proof (in the paper);
- gives access to a whole class of properties for provers not supporting ArgMax;
- with more than 2 classes, simpler operators provides faster verification runs, without comprising soundness!

Future work

- more operator rewrites are planned: ArgMin, Exp, pooling family;
- smaller amount of ReLUs;
- automated proof of the specification within CAISAR with Rocq + extracted OCaml code;

CAISAR

Website: https://caisar-platform.com including manual, use cases and examples

Code: https://git.frama-c.com/pub/caisar

Looking for a research engineer: https://caisar-platform.com/positions

Bibliography

[1] S. Bak, "Nnenum: Verification of ReLU Neural Networks with Optimized Abstraction Refinement," in *NASA Formal Methods*, A. Dutle, M. M. Moscato, L. Titolo, C. A. Muñoz, and I. Perez, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 19–36. doi: 10.1007/978-3-030-76384-8_2.

[2] S. Wang *et al.*, "Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Complete and Incomplete Neural Network Robustness Verification." Accessed: Mar. 04, 2022. [Online]. Available: http://arxiv.org/abs/2103.06624

[3] H. Wu *et al.*, “Marabou 2.0: A Versatile Formal Analyzer of Neural Networks.” 2024. doi: 10.48550/ARXIV.2401.14461.

[4] A. Lemesle, J. Lehmann, T. L. Gall, and Z. Chihani, “Neural Network Verification with PyRAT,” in *Symposium on Static Analysis*, 2025.

[5] D. Shriver, S. Elbaum, and M. B. Dwyer, “DNNV: A Framework for Deep Neural Network Verification,” in *Computer Aided Verification*, A. Silva and K. R. M. Leino, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 137–150. doi: 10.1007/978-3-030-81685-8_6.

[6] D. M. Lopez, S. W. Choi, H.-D. Tran, and T. T. Johnson, “NNV 2.0: The Neural Network Verification Tool,” in *Computer Aided Verification*, C. Enea and A. Lal, Eds., Cham: Springer Nature Switzerland, 2023, pp. 397–412.

[7] G. Ardouin, M. Alberti, and J. Girard-Satabin, “La Confiance Avec Le Contrôle : Spécification et Vérification d'hyperpropriétés Sur Réseaux de Neurones,” in *JFLA 2026 – 37es Journées Francophones Des Langages Applicatifs*, Oberbronn, Alsace, France: Marie Kerjean and Yannick Zakowski, Jan. 2026. Accessed: Feb. 12, 2026. [Online]. Available: https://hal.science/hal-05428158

[8] A. Athavale, E. Bartocci, M. Christakis, M. Maffei, D. Nickovic, and G. Weissenbacher, “Verifying Global Two-Safety Properties in Neural Networks with Confidence,” in *Computer Aided Verification*, A. Gurfinkel

and V. Ganesh, Eds., Cham: arXiv, June 2024, pp. 329–351. doi: 10.48550/arXiv.2405.14400.