VNN-COMP 2023

4th International Verification of Neural Networks Competition

Stanley Bak

Christopher Brix

Taylor Johnson

Changliu Liu

David Shriver

CAV '23

https://sites.google.com/view/vnn2023

FoMLAS

Outline and Session Plans

- VNN-COMP overview & background (Taylor)
- VNN-COMP’23 results (Christopher)

- Discussion: feedback on this year, past years, and plans for the future

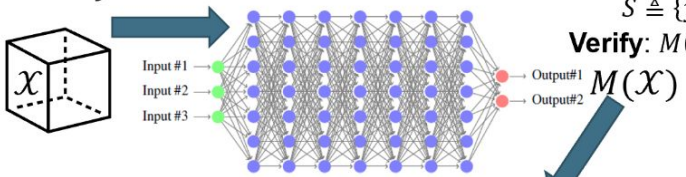
What is VNN-COMP?

- Similar to other formal methods and verification competitions (SV-COMP, SMT-COMP, SAT-COMP, ARCH-COMP, etc.)
- Given a neural network M and a specification S, try to prove that M satisfies S
  - S: essentially constraint over input space and output space of M, specified in VNN-LIB, similar to SMT-LIB
  - M: neural network of various forms, specified in the ONNX format
  - Possible results: prove S holds, falsify S to show it does not hold, unknown, or timeout
  - Challenges: don't necessarily know ground truth, particularly for large models; scalability; formats and model representations; compute; …
- Tool participants: those who registered and submitted results
  - 2023: abCROWN, Fastbatllnn, Marabou, NeuralSAT, nnenum, NNV, PyRAT
- Benchmarks: proposed by community and tool participants
  - Each tool participant can nominate 2 benchmarks to be scored
- All networks are feedforward in various flavors (fully-connected, convolutional, resnets, some transformers, …)
  - No RNNs/LSTMs, no closed-loop systems (see ARCH-COMP AINNCS), …

Neural Network Verification Example

Given a NN $M: \mathbb{R}^n \mapsto \mathbb{R}^m$ & an input set $\mathcal{X} \subseteq \mathbb{R}^n$, the **output reachable set** of M is $\mathrm{Y} = \{y \mid y = M(x), \forall x \in \mathcal{X}\} \subseteq \mathbb{R}^m$

M: simple feedforward NN with 3 inputs, 2 outputs, 7 hidden layers of 7 neurons each, ReLU activations; $M: \mathbb{R}^3 \to \mathbb{R}^2$

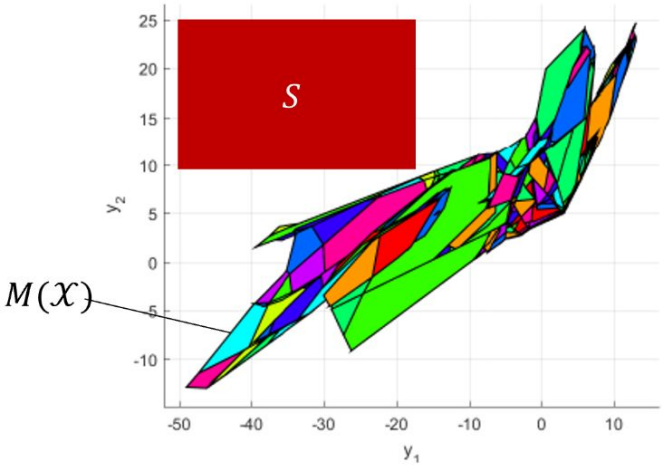


Input set: $\mathcal{X} \triangleq \{x \in \mathbb{R}^3 \mid \|x\|_\infty \leq 1\}$

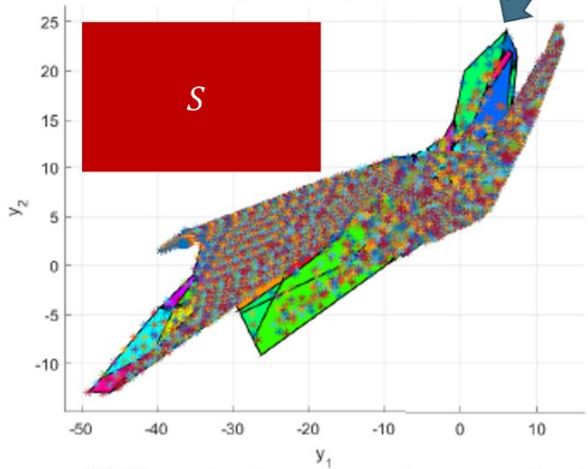
Specification:

$S \triangleq \{y \in \mathbb{R}^2 \mid -50 \leq y_1 \leq -20 \wedge 10 \leq y_2 \leq 25\}$

Verify: $M(\mathcal{X}) \cap S = \emptyset$?



Output reachable set $Y = M(\mathcal{X})$: union of 1250 polytopes, shown in different colors



8000 randomly generated outputs (evaluating M on points, e.g., $M(x)$ for 8000 points $x \in \mathcal{X}$)

Scalability Challenge: for ReLU activations, this problem is NP-complete

Intuition: number of polytopes may grow exponentially in number of ReLUs due to case splitting

History and Overview

- 2020: first iteration, no standardized evaluation (results incomparable)
- 2021: standardized hardware
- 2022: standardized submission system
- 2023: many new benchmarks

2020 - 2022: Summary published
First three years of the international verification of neural networks competition (VNN-COMP)
STTT: C. Brix, M. Müller, S. Bak, T. Johnson, C. Liu

https://doi.org/10.1007/s10009-023-00703-4

Similarities, Changes and Differences from Past Iterations

- Similar organization, same infrastructure as last year, plus updates and general improvements

- Many benchmarks this year are new and have different architectures than in the past: this year perhaps more of a challenge than a competition

- Some AI/ML and verification/formal methods community differences in terms of what is a competition

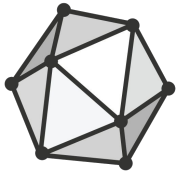
Goal 1: Foster Community

- Create examples and benchmarks for different classes of neural networks
- Participants often then use these in papers (some unintended consequences in reviews sometimes :-( ), so reward this work that needs to be done anyway
- Celebrate and reward tool development
- This session and feedback from community

Some significant broader impact:
- highlights in US NSF FMitF PI meetings, feeding input to other programs (SLES)
- DARPA Assured Autonomy and ANSR programs
- highlights in start-up funding rounds/press releases
- technology transfer into commercial products (Mathworks, …)
- etc.

Goal 2: Standardized Comparison

- Unified format for NNs: onnx
- Unified format for specifications: Vnnlib
- Equal cost hardware (AWS)



ONNX

VNN-LIB

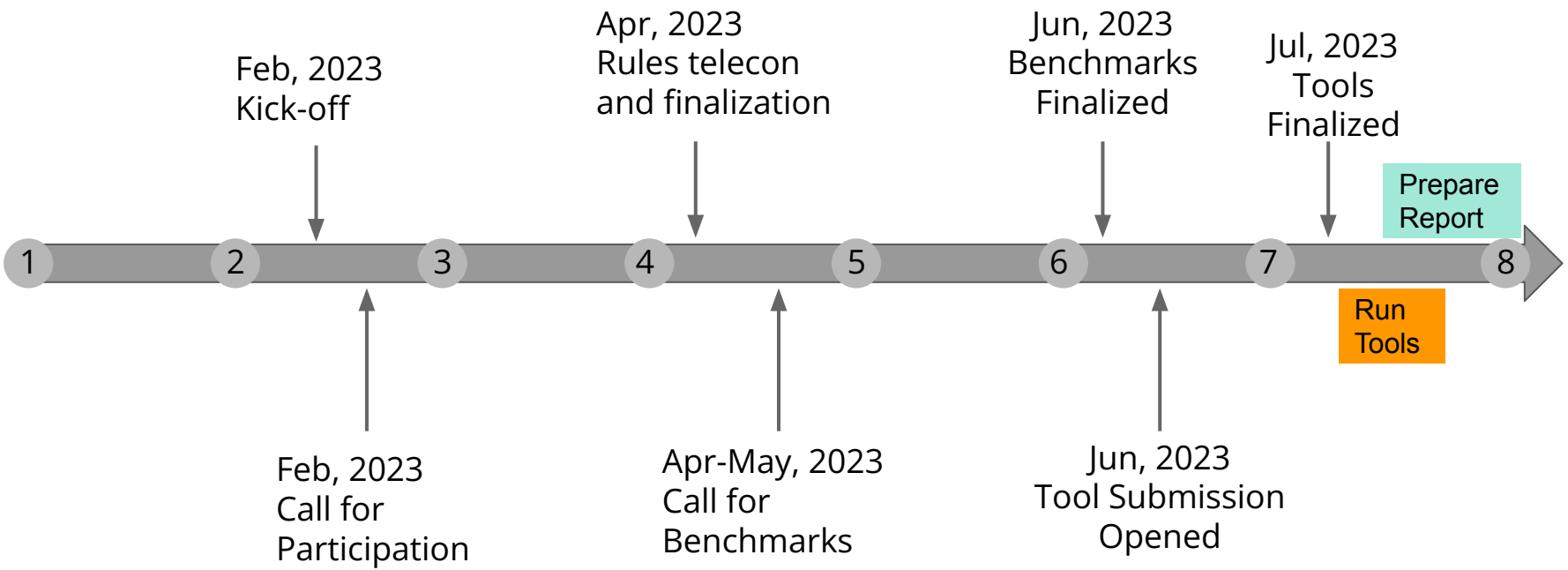
Verification of Neural Networks

aws

AWS Credits kindly provided by
- Lu Jin Family Foundation
- Intelligent Control Lab @ CMU

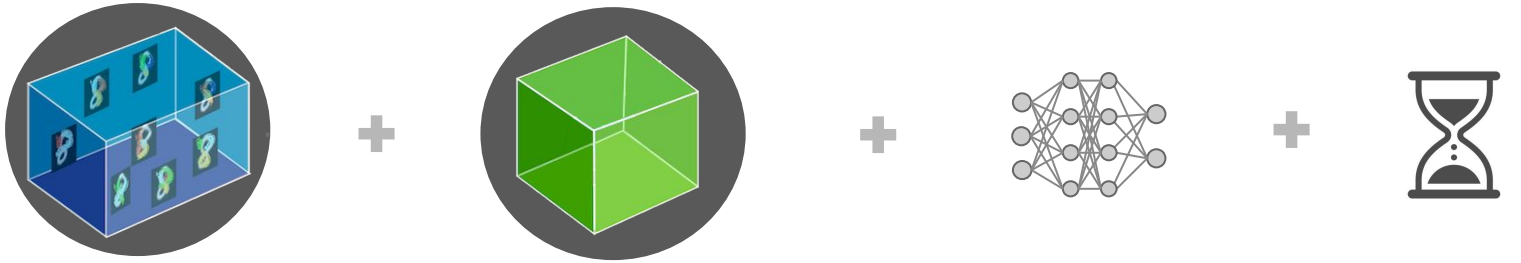
Carnegie Mellon University

Timeline


Feb, 2023
Kick-off
Apr, 2023
Rules telecon
and finalization
Jun, 2023
Benchmarks
Finalized
Jul, 2023
Tools
Finalized
Prepare
Report
1
2
3
4
5
6
7
8
Run
Tools
Feb, 2023
Call for
Participation
Apr-May, 2023
Call for
Benchmarks
Jun, 2023
Tool Submission
Opened

Rules: Terminology

Benchmark Instance:



Input Specification + Property + Network + Timeout

VNN-LIB ONNX

Benchmark:

- Set of related instances
- Total timeout < 6h

Rules: Scoring

For a benchmark instance:

GT \ Returned results	Holds	Violated	Timeout	Error	Unknown
Holds	+10 pts	-150 pts	0 pts		
Violated	-150 pts	+10 pts			

No Time Bonus

Decided by witness

Overall benchmark score:

- Percent of maximum points reached by any tool

Total score:

- Sum of all benchmarks (incentivizes supporting as many as possible)

Rules: Time Runtime Calculation

Preparation Script:

- Untimed (max 10 minutes)
- Parsing of network and property for conversion to other format

Overhead:

- Measured on minimal test instances

Rules: Tool Tuning

- Tuning per benchmark permitted, but not per network
- Instances are randomized
  - One set provided for testing
  - Different set used for evaluation
- Tools may not have substantially overlapping codebases

Evaluation Hardware

Choice of 3 AWS instances (same for all benchmarks):

Name	Focus	GPUs	VRAM	# CPU Threads	Clock Speed	Memory	$ / hour
m5.16xlarge	CPU	-	-	64	3.1 GHz	256 GB	3.072
g5.8xlarge	mixed	A10G*	24 GB	32	3.3 GHz	128 GB	2.448
p3.2xlarge	GPU	V100	16 GB	8**	2.7 GHz	61 GB	3.06

* A10G has good float32 performance, but poor float64 performance compared to V100

** No instance with strong GPU and better CPU available

Benchmarks 2023

Name	Network Type	# Parms	Input Dim	Sparsity	# Regions
nn4sys	Conv, FC, Residual + ReLU, Sigmoid	33k - 37M	1-308	0-66%	1 - 11k
VGGNet 16	Conv + ReLU + MaxPool	138M	150k	0-99%	1
Collins Rul CNN	Conv + ReLU, Dropout	60k - 262k	400-800	50-99%	2
TLL Verify Bench	FC + ReLU	17k - 67M	2	0%	1
Acas XU	FC + ReLU	13k	5	0-20%	1-4
cGAN	FC, Conv, ConvTranspose, Residual + ReLU, BatchNorm, AvgPool	500k-68M	5	0-40%	2
Dist Shift	FC + ReLU, Sigmoid	342k-855k	792	98.9%	1
ml4acopf	FC, Residual + ReLU, Sigmoid	4k-680k	22-402	0-7%	1-600
Traffic Signs Recognition	Conv, FC + MaxPool, Sign	905k-1.7M	2.7k-12k	0%	42
ViT	Conv, FC, Residual + ReLU, Softmax, BatchNorm	68k-76k	3072	0%	9

Unscored Benchmarks

Name	Network Type	# Parms	Input Dim	Sparsity	# Regions
CCTSDB YOLO	Conv, Skip + ReLU	100k	12k	99.9%	1
Collins YOLO Robustness	Conv, FC, Skip + LeakyReLU, MaxPool	1.7M	1.2M	99.3%	19
Metaroom	Conv, FC + ReLU	466k-7.4M	5376	97-100%	19
Yolo	Conv, Skip + ReLU, AvgPool	92k	8112	0-17%	1

Selection: https://github.com/stanleybak/vnncomp2023/issues/2#issuecomment-1595747102

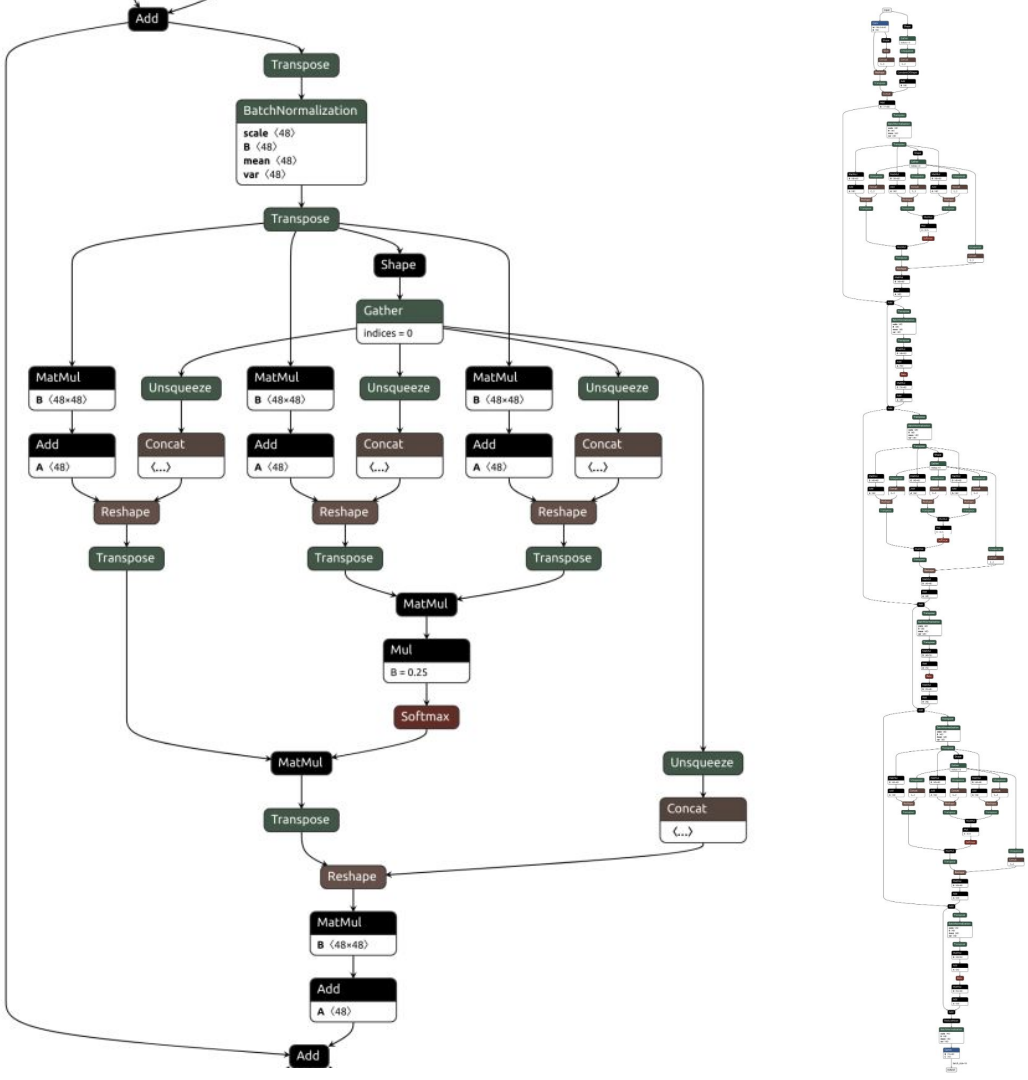
Tool participants may nominate up to 2 benchmarks to be scored

Eliminated rule based on discussion that at least 2 tools must submit results for benchmark to be scored, based on discussions and that we forgot to include rule in official rules

Some participants submitted results for unscored benchmarks

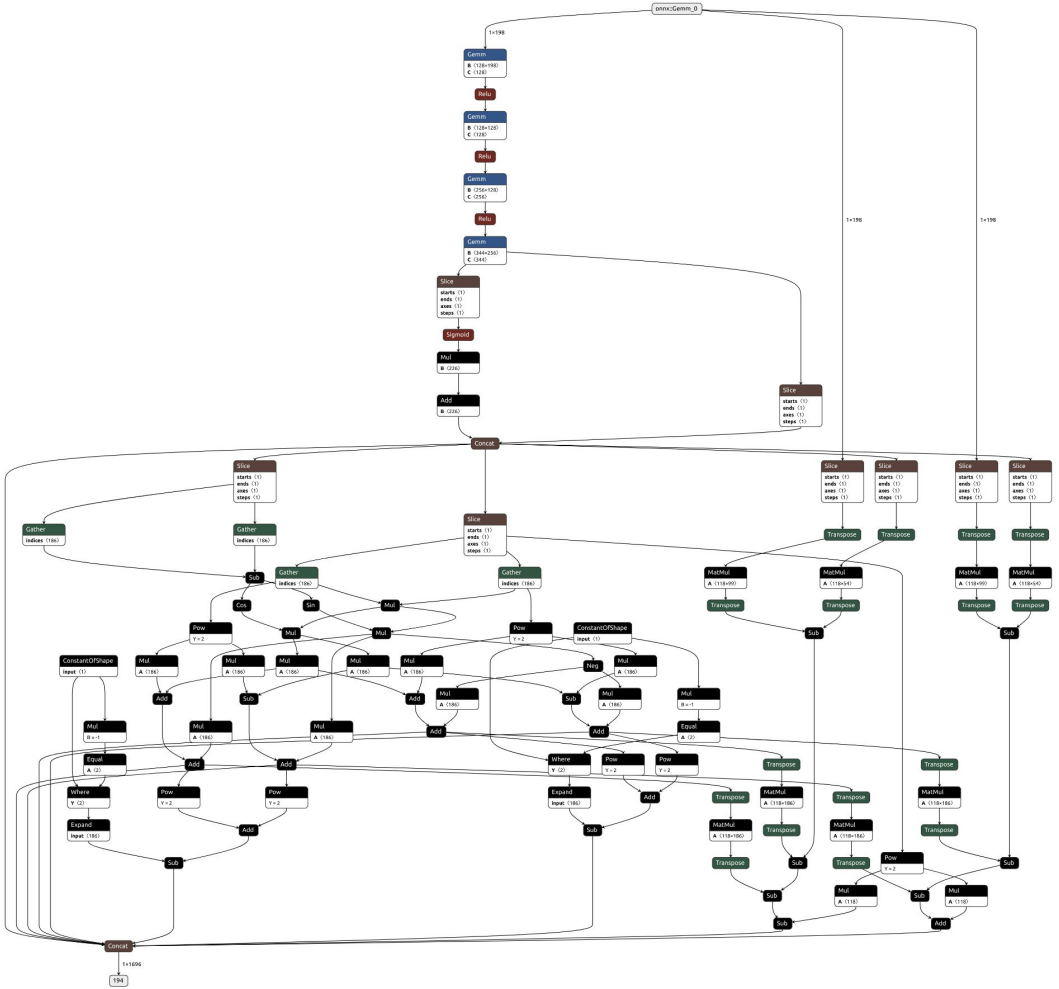
ViT

- Vision Transformers
- Self-Attention
- BatchNorm instead of LayerNorm
- CIFAR-10 with eps=1/255
- Excluding trivial instances



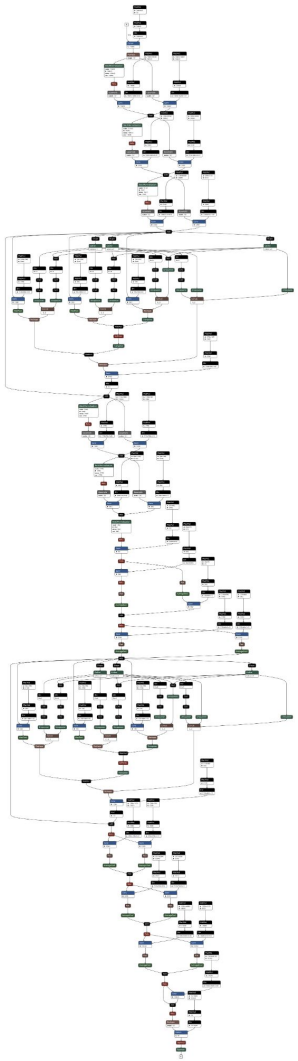
ml4acopf

- Machine Learning for AC Optimal Power Flow
- Simulates different number of buses, generators, loads and lines
- Tests for power balance Violation on perturbations of the reference active and reactive load



cGAN

- Conditional generative adversarial networks
- Generate camera images with vehicle obstacle at specific distance
- CNN and Transformer
- Sizes 32x32, 64x64
- Test combination of generator and discriminator

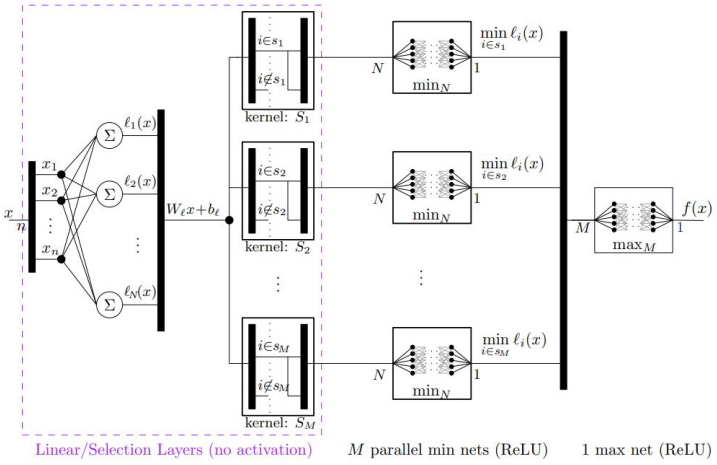


TLL Verify Bench

Two-Level Lattice networks represented as FC

Architecture:

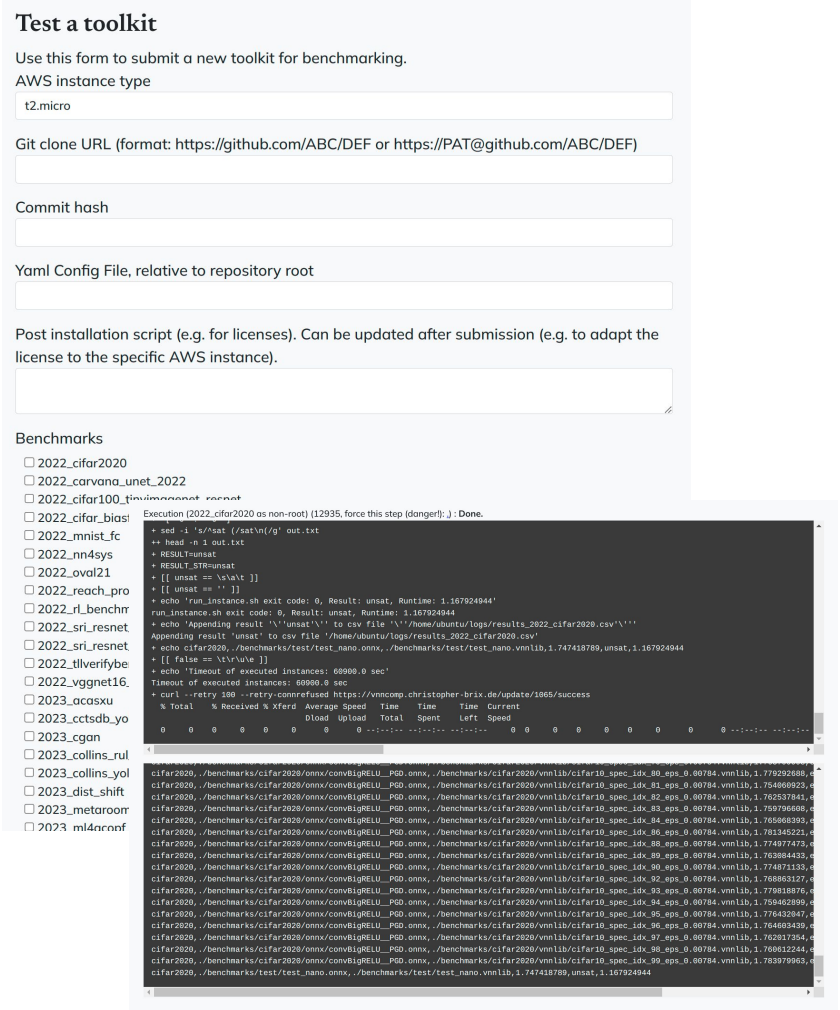
- Deep: 6 - 12 ReLU layers
- Low dim in- and output
- Large intermediate dimensions (8k)
- Consecutive affine layers



Tool Submission Process:

- Git URL + Hash
- Committed config file
- Installation + Post-Installation scripts
- Automated evaluation

⇒ Testing can be done by teams themselves



Automatic Evaluation - Lessons Learned

Last Year

- ✓ Teams could test their submission
- ✓ Simple final evaluation
- ✓ External licenses
- ✓ Closed source tools
- ✓ Reasonable usage by teams
- ✓ Ready for reuse

This Year

- ✓ Quota allocation process at AWS
- ✓ Better frontend
- ✓ Dev environment
- ✓ Little downtime

What went wrong

- × No interactive debugging
- × Small issues with benchmarks

Tool Participants

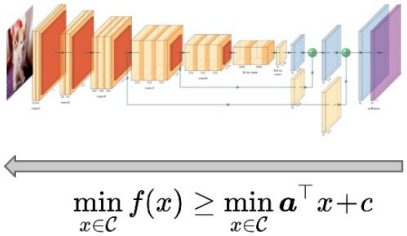
Tool Overview

Name	Institution	Open Source	AWS Instance	Additional License
alpha-beta-CROWN	UIUC, CMU, UCLA, Drexel, Columbia, RWTH Aachen University	yes	g5.8xlarge	GUROBI
FastBATLLNN	UCI	yes	m5.16xlarge	–
Marabou	Stanford	yes	m5.16xlarge	GUROBI
NeuralSAT	GMU	no*	g5.8xlarge	GUROBI
nnenum	Stony Brook University	yes	m5.16xlarge	GUROBI
NNV	Vanderbilt, University of Nebraska-Lincoln	yes	m5.16xlarge	Matlab
PyRAT	Université Paris-Saclay, CEA, List	no	m5.16xlarge	–

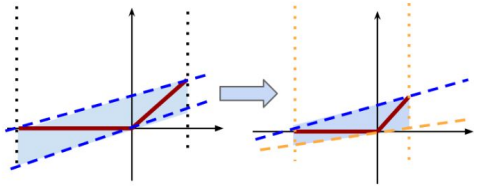
*Will be open sourced

α,β-CROWN (abcrown.org)

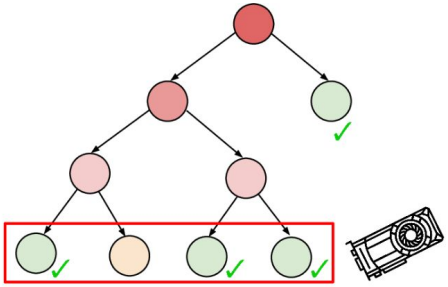
Team from
UIUC/CMU/UCLA/Columbia/Drexel/
RWTH Aachen University



linear bound propagation (CROWN) arxiv 1811.00866



Tight incomplete verifier:
α-CROWN arxiv 2011.13824
GCP-CROWN arxiv 2208.05740



Complete verification via branch and bound: (β-CROWN) arxiv 2103.06624

Implementation highlights: PyTorch + GPU; verify *general* computations beyond NNs

NEW in 2023:

- **Branch and bound for general nonlinear func**; better branching heuristics (in submission)
- Handling output constraints (arxiv 2302.01404)
- Enhanced support for bounding general computation functions (arxiv 2002.12920)

FastBATLLNN

Fast Box Analysis of Two-Level Lattice Neural Networks

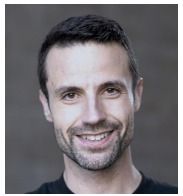
→ Verifies only *box-like (hyper-rectangle) output properties* for *Two-Level Lattice (TLL) NNs*

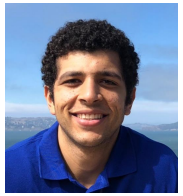
↪ competes only on `tllverifybench` benchmark

→ Fast verification using the ***unique semantics of TLLs*** and box-like output constraints

↪ Converts TLL verification into *region enumeration for a hyperplane arrangement* (worst case)

↪ Each hyperplane specified directly from TLL and problem (i.e. O(1) time)


James Ferlez


Haitham Khedr


Yasser Shoukry

CPS Lab Resilient

UCI University of California, Irvine

James Ferlez, Haitham Khedr, and Yasser Shoukry. Fast BATLLNN: Fast Box Analysis of Two-Level Lattice Neural Networks. In *Hybrid Systems: Computation and Control 2022 (HSCC'22)*. ACM, 2022.

The Marabou Framework for Formal Analysis of Neural Networ

Tool submission authors: Haoze Wu[1], Guy Katz[2], Clark Barrett[1]
[1] Stanford University [2] Hebrew University of Jerusalem
Full author list:
https://github.com/NeuralNetworkVerification/Marabou/blob/master/AUTHORS

Python/C++ Interface

- Supports tf, onnx format
  - Piecewise-linear/transcendental activations
- Supports arbitrary linear properties

Verification Query

Parallelization

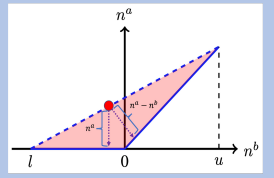
- Split-and-Conquer [FMCAD'20]
- Portfolio
  - PGD attack
  - Sampling
  - Verification Engine

Main Engine

CEGAR for Sigmoids [SaTML'23]

BaB shell

Convex solver:
DeepSoI [TACAS'22]



Network-level reasoner:

- Abstract interpretation
  - Supports transformers [AISTATS'23]
- (MI)LP-based tightening [FMCAD'22]
- Forward-backward analysis [OOPSLA'22]

SAT
UNSAT

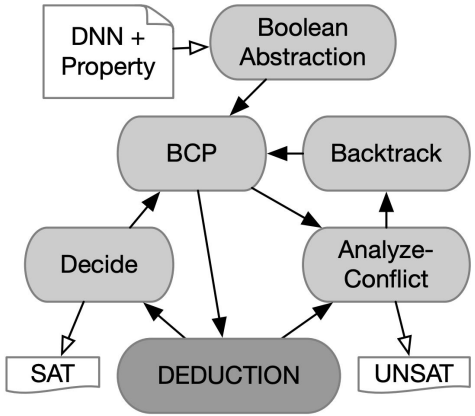
Tool available on GitHub:
https://github.com/NeuralNetworkVerification/Marabou

NeuralSAT

DynaROARS Lab @ GMU

dynaroars.github.io

- A DPLL(T)-based Constraint Solving Approach to Verifying DNNs
- Core algorithms:
  - DPLL/CDCL
  - Abstraction
  - Adversarial attack



nnenum - Neural Network Enumeration Tool

nnenum (pronounced en-en-en-um) is a high-performance neural network verification tool. Multiple levels of abstraction are used to quickly verify ReLU networks without sacrificing completeness. Analysis combines three types of zonotopes with star set (triangle) overapproximations, and uses efficient parallelized ReLU case splitting. The tool is written in Python 3, uses GLPK and Gurobi for LP solving (you can choose either) and directly accepts ONNX network files and vnnlib specifications as input. The ImageStar trick allows sets to be quickly propagated through all layers supported by the ONNX runtime, such as convolutional layers with arbitrary parameters.

NNV *[Tran et al., **CAV'20**]*

The Neural Network Verification (NNV) is a Matlab toolbox that implements reachability methods for analyzing neural networks.

VNN-COMP 2023 approach

- Set Representation: **Star, ImageStar**
- General verification approach (all participated benchmarks)
  1) Simulation-guided verification (SAT?)
  2) Reachability analysis (UNSAT?)
     a) Refinement from sound ("relax", "approx") approaches to sound and complete ("exact").

1) "relax" → 2) "approx" → 3) "exact"

Fastest, most conservative → Reduced overapproximation → Slowest, no overapproximation

Upcoming events

- Presentation on Thursday (9am session)
  - *[Manzanas Lopez et al., "NNV 2.0: The Neural Network Verification Tool", CAV'23]*
- NNV Tutorial at EMSOFT'23
  - ESWeek (half-day hands-on tutorial)

Team members: Diego Manzanas Lopez, Neelanjana Pal, Hoang-Dung Tran, Taylor T. Johnson

https://github.com/verivital/nnv

PyRAT

- **From:** Université Paris-Saclay, CEA, List
- **Type:** Abstract Interpretation based
  - **Domains:** Box, Zonotopes, Polyhedra
- **Support:**
  - Architectures: DNN, CNN, Residual connection
  - Format: ONNX, Keras, NNet, PyTorch
  - Activations: ReLU, Sigmoid, Tanh, Softmax
- Input splitting optimisation: ReCIPH
- Written in Python

Results

Results Preface

Some discrepancies in rules understanding (all benchmarks proposed vs. all those nominated by tools); eliminated rule that at least 2 tools must provide results for a benchmark to be scored (see GitHub discussion for long overview)

Tools could select on which benchmarks to evaluate

Reran 2022 benchmarks as well (not scored)

Some possible numerical issues with counterexamples (some tool disagreement, possibly numerical precision or some other issue)

Need to audit and double check everything still, so take results with this in mind at this time

Please help us audit them and ensure results for your tool are accurate

Total Score

#	Tool	Score
4	NeuralSAT	535.7
5	nnenum	430.5
6	NNV	257.1
7	FastBATLLNN	100.0

Total Score

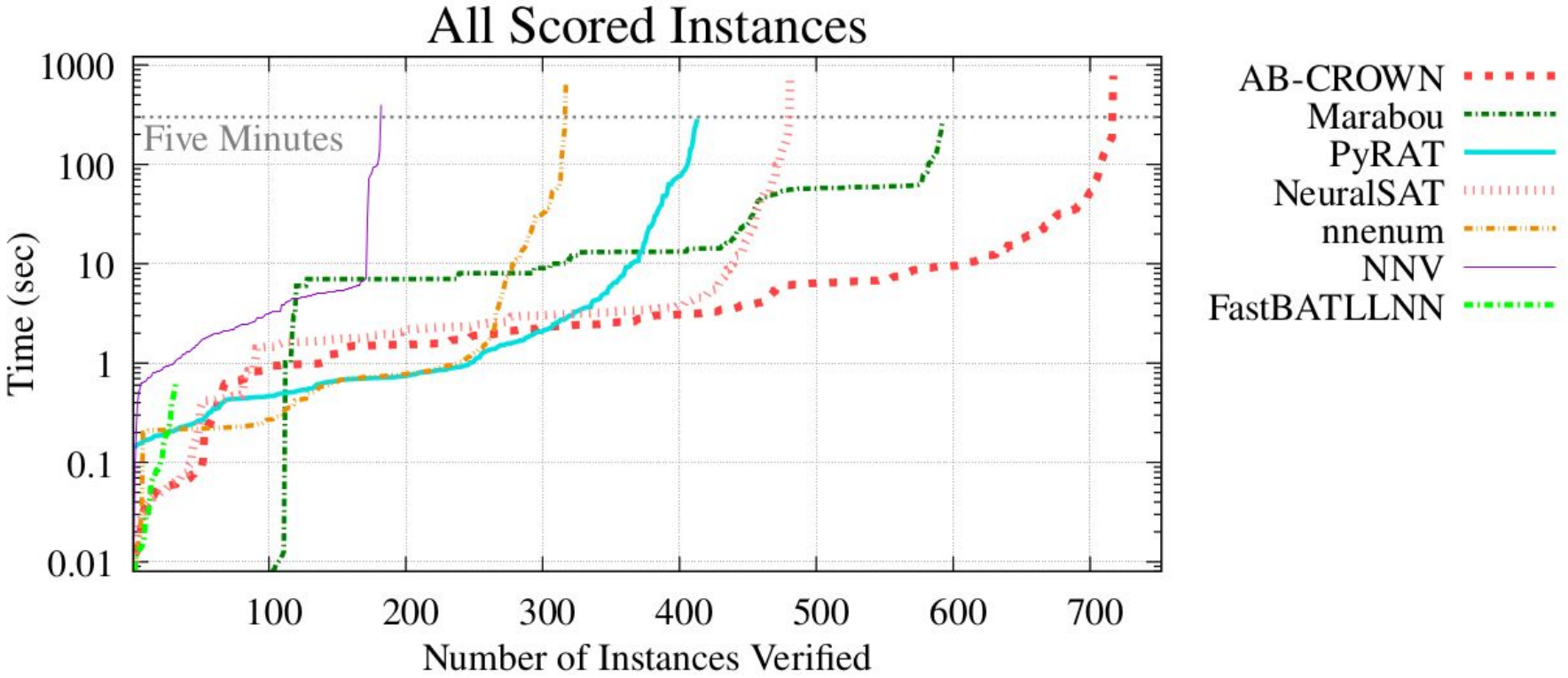
#	Tool	Score
3	PyRAT	550.7
4	NeuralSAT	535.7
5	nnenum	430.5
6	NNV	257.1
7	FastBATLLNN	100.0

Total Score

#	Tool	Score
	And the winner is…	
3	PyRAT	550.7
4	NeuralSAT	535.7
5	nnenum	430.5
6	NNV	257.1
7	FastBATLLNN	100.0

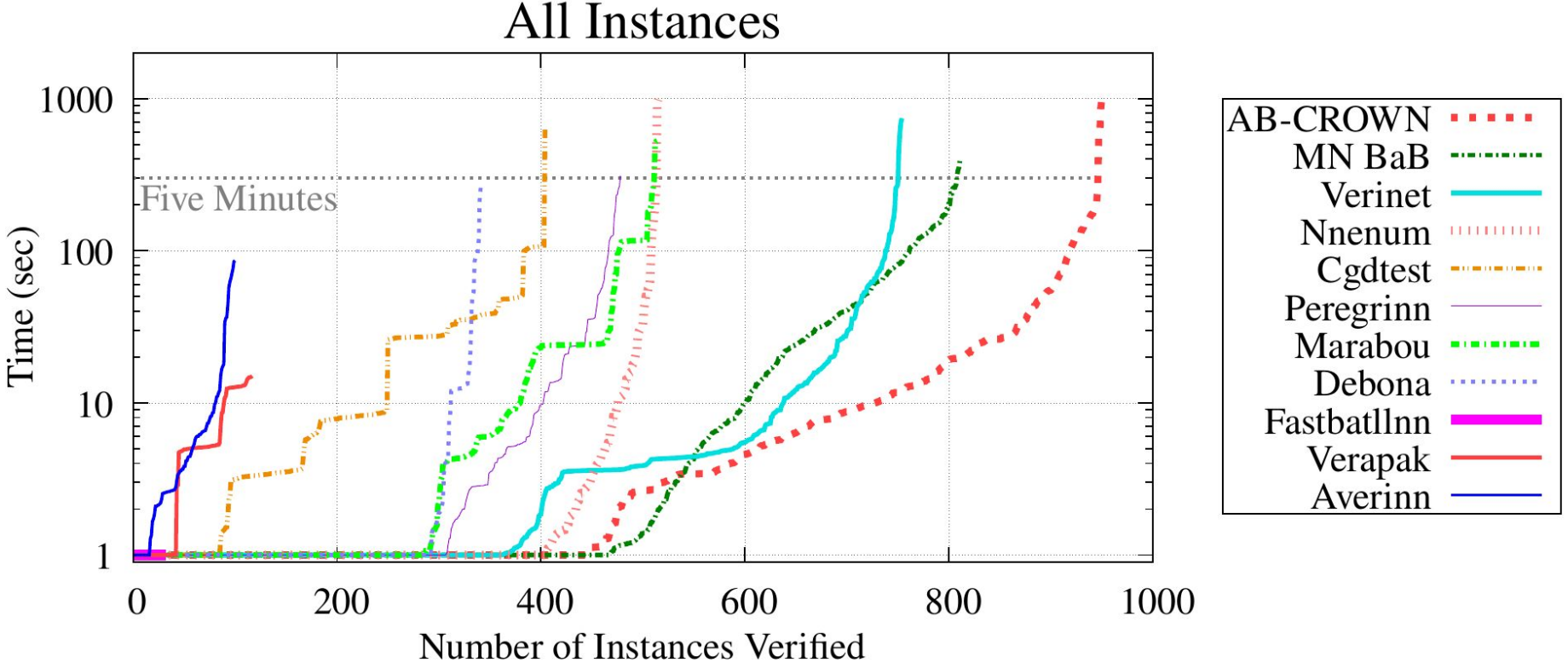
Total Score

#	**Tool**	**Score**
1	α-β-CROWN	839.5
2	Marabou	642.4
3	PyRAT	550.7
4	NeuralSAT	535.7
5	nnenum	430.5
6	NNV	257.1
7	FastBATLLNN	100.0



Note that #Instances per Benchmark varies

2022 Results for Comparison



Note that #Instances per Benchmark varies

Results by Benchmark

Table 2: Benchmark `2023-acasxu`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	nnenum	139	47	0	0	1860	100.0%
2	α-β-CROWN	139	47	0	0	1860	100.0%
3	NeuralSAT	138	46	0	0	1840	98.9%
4	PyRAT	137	45	0	0	1820	97.8%
5	Marabou	135	46	0	0	1810	97.3%
6	NNV	68	27	0	0	950	51.1%

Table 3: Benchmark `2023-cgan`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	8	13	0	0	210	100.0%
2	PyRAT	8	11	0	0	190	90.5%
3	nnenum	6	11	0	0	170	81.0%
4	NeuralSAT	3	11	0	0	140	66.7%
5	NNV	3	11	0	0	140	66.7%
6	Marabou	0	13	0	0	130	61.9%

Table 4: Benchmark `2023-collins-rul-cnn`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	nnenum	35	27	0	0	620	100.0%
2	NeuralSAT	35	27	0	0	620	100.0%
3	Marabou	35	27	0	0	620	100.0%
4	α-β-CROWN	35	27	0	0	620	100.0%
5	PyRAT	33	27	0	0	600	96.8%
6	NNV	23	27	0	0	500	80.6%

Table 5: Benchmark `2023-dist-shift`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	67	5	0	0	720	100.0%
2	PyRAT	66	5	0	0	710	98.6%
3	NeuralSAT	66	5	0	0	710	98.6%
4	Marabou	55	5	0	0	600	83.3%
5	NNV	0	4	0	0	40	5.6%

Table 6: Benchmark `2023-ml4acopf`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	18	1	0	0	190	100.0%

Table 7: Benchmark `2023-nn4sys`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	192	0	0	2	1620	100.0%
2	NeuralSAT	81	0	0	2	510	31.5%
3	Marabou	33	0	0	1	180	11.1%
4	PyRAT	38	0	0	2	80	4.9%
5	nnenum	20	0	0	2	-100	0%
6	NNV	0	2	0	8	-1180	0%

Table 8: Benchmark `2023-tllverifybench`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	NeuralSAT	15	17	0	0	320	100.0%
2	FastBATLLNN	15	17	0	0	320	100.0%
3	α-β-CROWN	15	17	0	0	320	100.0%
4	PyRAT	10	12	0	0	220	68.8%
5	Marabou	5	17	0	0	220	68.8%
6	nnenum	2	16	0	0	180	56.2%
7	NNV	1	16	0	0	170	53.1%

Note that the TLL (Two-Level Lattice) networks were represented as fully connected networks, but can be encoded more efficiently using a more complex structure

Table 9: Benchmark `2023-traffic-signs-recognition`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	Marabou	0	18	0	1	30	100.0%
2	PyRAT	0	7	0	1	-80	0%
3	NeuralSAT	0	31	0	4	-290	0%
4	α-β-CROWN	0	39	0	3	-60	0%

Table 10: Benchmark `2023-vggnet16`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	15	0	0	0	150	100.0%
2	nnenum	14	0	0	0	140	93.3%
3	PyRAT	13	1	0	0	140	93.3%
4	NeuralSAT	5	1	0	0	60	40.0%
5	Marabou	2	1	0	0	30	20.0%

Table 11: Benchmark `2023-vit`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	Marabou	200	0	0	0	2000	100.0%
2	α-β-CROWN	79	0	0	0	790	39.5%

Table 34: Incorrect Results (or Missing CE)

#	Tool	Count
1	NNV	8
2	NeuralSAT	6
3	α-β-CROWN	5
4	PyRAT	3
5	nnenum	2
6	Marabou	2

⇒ Incentivising tools to provide a witness makes determining the ground truth much easier

Table 29: Overhead

#	Tool	Seconds
1	FastBATLLNN	0.3
2	nnenum	0.9
3	Marabou	1.1
4	PyRAT	3.4
5	NeuralSAT	3.6
6	α-β-CROWN	5.7
7	NNV	15.9

Table 30: Num Benchmarks Participated

#	Tool	Count
1	α-β-CROWN	10
2	Marabou	9
3	PyRAT	8
4	NeuralSAT	8
5	nnenum	6
6	NNV	6
7	FastBATLLNN	1

Other Metrics

Table 31: Num Instances Verified

#	Tool	Count
1	α-β-CROWN	717
2	Marabou	592
3	NeuralSAT	481
4	PyRAT	413
5	nnenum	317
6	NNV	182
7	FastBATLLNN	32

Table 32: Num SAT

#	Tool	Count
1	α-β-CROWN	149
2	NeuralSAT	138
3	Marabou	127
4	PyRAT	108
5	nnenum	101
6	NNV	87
7	FastBATLLNN	17

Table 33: Num UNSAT

#	Tool	Count
1	α-β-CROWN	568
2	Marabou	465
3	NeuralSAT	343
4	PyRAT	305
5	nnenum	216
6	NNV	95
7	FastBATLLNN	15

Results Audit (please help)

Results and scoring scripts:
https://github.com/ChristopherBrix/vnncomp2023_results

- No code execution necessary

Report bugs

- Code similar to last year, but double-check
- Errors can be fixed for final report

Example from the Log

Row: ['NN_rul_full_window_20-monotonicity CI shift5 w20', '1.0 (v)', '1.0 (v)', '1.0 (v)', '1.0 (v)', '5.1 (v)', '3.6 (v)'] (THIS IS THE LIST OF RUNTIMES FOR EACH TOOL, 1 sec is minimum, 'v' is violated and 'h' is holds)

38: alpha_beta_crown score: 12, is_ver: False, is_fals: True, is_fastest: True

38: cgdtest score: 12, is_ver: False, is_fals: True, is_fastest: True

38: mn_bab score: 12, is_ver: False, is_fals: True, is_fastest: True

38: nnenum score: 12, is_ver: False, is_fals: True, is_fastest: True

38: peregrinn score: 10, is_ver: False, is_fals: True, is_fastest: False

38: verinet score: 10, is_ver: False, is_fals: True, is_fastest: False

(Note that is fastest is true for 4 tools above, as all times within 0.2 secs are treated the same for scoring)

Example #2 from the Log with mismatched results

```
WARNING: multiple results for index 39. Violated: 1 (['cgdtest']), Holds: 5 (['alpha_beta_crown',
'mn_bab', 'nnenum', 'peregrinn', 'verinet'])

checking counterexample for cgdtest
Checking ce path:
../cgdtest/collins_rul_cnn/NN_rul_full_window_20_monotonicity_CI_shift10_w20.counterexample.gz

L-inf norm difference between onnx execution and CE file output: 14.965118408203125 (limit:
0.0001)  (The stated output in the CE file and the onnxruntime output at the given input are not
the same, so the counterexample is invalid)

were violated counterexamples valid?: {'cgdtest': False}

Row: ['NN_rul_full_window_20-monotonicity_CI_shift10_w20', '1.1 (h)', '1.0 (v)', '1.0 (h)', '1.0
(h)', '35.4 (h)', '3.6 (h)', '*multiple results*']

39: alpha_beta_crown score: 12, is_ver: True, is_fals: False, is_fastest: True
39: cgdtest score: -100, is_ver: False, is_fals: False, is_fastest: False(-100 penalty score)
39: mn_bab score: 12, is_ver: True, is_fals: False, is_fastest: True
39: nnenum score: 12, is_ver: True, is_fals: False, is_fastest: True
39: peregrinn score: 10, is_ver: True, is_fals: False, is_fastest: False
39: verinet score: 10, is_ver: True, is_fals: False, is_fastest: False
```

Report / Proceedings

- A report (officially) co-authored by organizers
- Will be uploaded to arXiv as in the past

We will contact tool authors to write a paragraph about your benchmark / tool (you can add a few citations)

Discussion: Feedback & Future of VNN-COMP

Proposals for next year

- Disallow tool tuning per benchmark?
  - Focus on automated strategy selection
    - if (#inputs > 100) set_settings_image(); else set_settings_control();
  - Beneficial for real-world users
- Require all benchmarks to have at least one sat and one unsat instance
- Filter out / reduce scores for easy **sat** instances (e.g. via adv attack)
- Handling on fast/easy instances:
  - Minimum execution time < 1 second?
  - Batch mode verification?

Proposals for next year

- Separate category for falsification
- Look into effect of (floating point) precision for analysis
- Look into other Hardware/AWS instance options

Discussion: Participation

- How to incentivize tool participation?
  - Minimize barriers to entry: reuse benchmarks already proposed, ease execution platform running (off of AWS to run locally?), …
  - Reuse benchmarks in papers
    - Have seen papers rejected for not comparing to VNN-COMP results: unintended consequence
    - Some pros/cons here (can use cached official VNN-COMP results to lower execution time for artifact/repeatability evaluations, as there is some significant compute cost)
  - Prizes? Sponsorship? Publications?
- About 40% registration-to-results submission (~20 tools registered)
  - Why? Some correspondence with some
- Pros / cons with ONNX and VNN-LIB

Discussion: Benchmarks

- Benchmark proposers may be disappointed if we cannot score their benchmarks in this iteration
  - Rules require tool participants to nominate benchmarks
  - Broader community may have thought any proposed benchmark would be considered
  - But also not reasonable to expect teams to support new architectures, debug issues, etc., in a couple weeks
- How to incentivize benchmark creation? Not always publishable, not fully intending VNN-COMP for new architectures where no other methods can support benchmarks yet (unfair for participants)
- Rules issues: tool participant nomination process, multiple tools (>=2) supporting any benchmark for it to be scored (some controversy this year)
- Please submit benchmarks to AISoLA
  - https://aisola.org/tracks/c1/
  - Different publication options (on-site LNCS proceedings, post-proceedings in LNCS, STTT, etc.), talk to Taylor Johnson / Daniel Neider
  - Still time to get involved for 2023, on Crete in Greece in late October :-)
- Usage in control systems growing: VNN-COMP for open-loop (no plant models), hybrid systems competition ARCH-COMP AINNCS for closed-loop (neural networks + plant models as ODEs)
  - Talk to Taylor Johnson if interested, organizing since 2019
- Some challenges with ONNX and VNN-LIB (generalizations): any suggestions?

Discussion: Rules

- Any suggested changes in rules, process, etc.?
- Rule about multiple tools submitting results on benchmarks for them to be scored? Original goal of this was to incentivize tools supporting multiple benchmarks, and to allow comparisons, as well as ensure fairness with respect to scoring
- One rule aspect to discuss that did not arise before: one tool participant proposed a benchmark (ml4acopf), but did not submit results; probably we should exclude such cases if we continue similar scoring processes and only score benchmarks proposed by tool participants that do successfully participate and submit results
- Will discuss this and look at alternative scorings in the report (all benchmarks, all nominated for scoring, excluding ml4acopf); did not have time to do this for this presentation (some discrepancy in participant understand, e.g., maybe only indicated to run tool on scored benchmarks in submission system)

Discussion: Categories and Meta-Solvers

- First VNN-COMP had categories, based on architecture type (specifically by activation functions)
  - Should we have categories again?
  - How to subdivide?
  - Reuse of prior benchmarks category, newly proposed benchmarks category?
- Quite some interest in meta-solvers from multiple groups: special category?

Discussion: Goals

- Some criticism from community that we are not meeting our goals, e.g., to enable reasonable/fair comparisons
  - See: [König et al, Critically Assessing the State of the Art in CPU-based Local Robustness Verification, AAAI SafeAI'23] **(Best Paper)** https://ceur-ws.org/Vol-3381/
- Some complaints over comparing CPU and GPU methods (using our cost-equivalent instance-hour notion)
  - Could scale things more precisely based on exact cost
  - Challenging problem though, and comparable to what is done in practice in computer architecture for empirical performance comparisons (SPEC benchmarks, etc.)
- Other issues: helping or hurting the community?

Discussion: Organizational Structure

- A lot of work for organizers: change organizational structure and set up a steering committee with different more specific roles for the different organizers?
- Where to host? Verification or AI/ML venue?
- Any volunteers to help organize?
- Community infrastructure to ease barriers to entry (some support provided this year for VNN-LIB, etc.), but other ideas?
  - Open to joint funding proposals, think this would be of interest to a variety of funding agencies, and possibly companies
  - How to open up the execution infrastructure so all can see everything, ease of running, etc. ?

Discussion: Anything Else and Thank You

- Open discussion for any other topics

- Thank you to all the participants, benchmark proposers, organizers, FoMLAS organizers, CAV organizers, and all of you in the audience

- Acknowledgements: AFOSR, DARPA, NSF

RWTH AACHEN UNIVERSITY

Carnegie Mellon University

VANDERBILT UNIVERSITY