Bit-Precise Neural Network Verification

Edoardo Manino

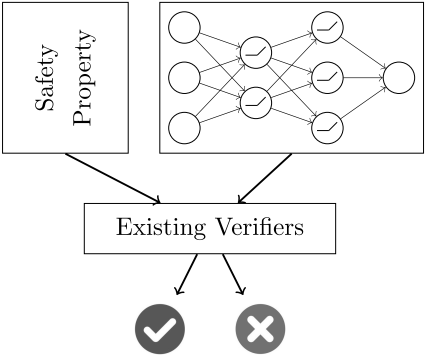
The University of Manchester

21 May 2024

EnnCore

MANCHESTER 1824 The University of Manchester

Background: neural network verification



Mainstream approach (à la VNN-COMP)

- Neural network in high-level format (ONNX, PyTorch. . .)
- Input-output safety property in FOL (pre- and post-conditions)
- Large focus on **robustness** properties

The abstraction ladder (1)

The "classic ML" mindset

- Define a neural net as $f : \mathbb{R}^n \to \mathbb{R}^m$
- Gradient descent, auto differentiation
- Data manifold, regularizers, . . .

What's the implicit assumption?

- We live in a mathematician's world
- At a very high level of abstraction
- And operations have infinite precision

Very effective, most of the time

The abstraction ladder (2)

Quantisation efforts

- 16-bit floating point
- 16-bit, 8-bit, 4-bit integers
- Binarized neural networks

Parallel execution

- GPU, SIMD instructions, TPU, FPGAs
- Distributed/federated learning

What's the implicit assumption?

- We make a lot of optimisations
- But the result doesn't really change

The abstraction ladder (3)

The software safety mindset

- Buffer overflow, division by zero
- Data race, deadlock, use-after-free
- Infinite loops, side effects

What's the implicit assumption

- Every innocent bug
- Can introduce a vulnerability

Just a problem for library makers?


Let's Play
Floating Point

Implementation effects (1)

Can we expect consistent behaviour across devices?

- Cidon et al., *Characterizing and taming model instability across edge devices*, 2021
- Wang et al, *SysNoise: exploring and benchmarking trainin-deployment system inconsistency*, 2023
- Schlögl et al., *Causes and Effects of Unanticipated Numerical Deviations in Neural Network Inference Frameworks*, 2023

Many low-level sources of noise!

- Pre-processing: .jpg→tensor (iDCT, interpolation, colour)
- Model inference: convolutions, upsampling, floats, quantize
- Post-processing: tensor→bounding box (rounding coordinates)

Up to 6% accuracy fluctuation[1]

[1] Cidon [2021] runs MobileNetV2 on photos taken from five different phones.

Implementation effects (2)

Can we trust NN verifiers?

- VNN-COMP compares the best neural network verifiers
- Let's reproduce one of their results!

Benchmark: reach_prob_density/robot_11

- A ReLU network with architecture $5 \times 64 \times 64 \times 64 \times 5$
- Input assumption: $x_0 \in [-1.8, -1.2] \wedge x_1 \in [-1.8, -1.2] \ldots$
- Output assertion: $y_0 \geq 0.27 \wedge y_1 \in [-0.17, 0.17] \ldots$

Five tools return a counterexample!

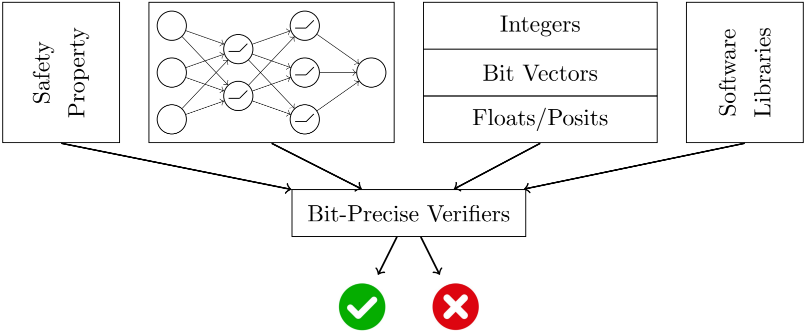
- $\alpha\beta$-CROWN, Marabou, `nnenum`, VeriNet, Peregrinn

But none of them violates the output assertion[2]

[2] With the plain C code from `onnx2c` and the MinGW-w64 compiler.


YOU PROVED THAT A ML MODEL IS SAFE
YOUR GUARANTEE IS IMPLEMENTATION DEPENDENT
imgflip.com

Bit-precise neural network verification



We need more precision!

- Bit-precise machine arithmetic (float or integer)
- Exact order of operations (requires knowledge of software)
- Guarantees sound proofs (unless the hardware is misbehaving)


Develop yet another software verifier
Check the performance of existing ones
imgflip.com

NeuroCodeBench (1)

Benchmarking goals

- Representativeness $\rightarrow$ realistic use cases
- Compatibility $\rightarrow$ SV likes plain C code
- Variety $\rightarrow$ from small to “large” instances
- Correctness $\rightarrow$ known ground truth

Our work is inspired by

- Microcontroller software

Short paper available!

- Manino, Menezes, Shmarov, Cordeiro. *NeuroCodeBench: a plain C neural network benchmark for software verification*. 2023
- `https://arxiv.org/abs/2309.03617`

NeuroCodeBench (2)

Benchmark Category	Safe	Unsafe	Ground Truth
`math_functions`	33	11	A Priori
`activation_functions`	40	16	A Priori
`hopfield_nets`	47	33	A Priori
`poly_approx`	48	48	Brute Force
`reach_prob_density`	22	13	VNN-COMP'22
`reinforcement_learning`	103	193	VNN-COMP'22
Total	293	314	

Table: Overview of *NeuroCodeBench*. The "Unsafe" column comprises all properties for which a counterexample exists. The "Ground Truth" column reports the source of our verdicts.


COMPETITION
TIME!

A short story (1)

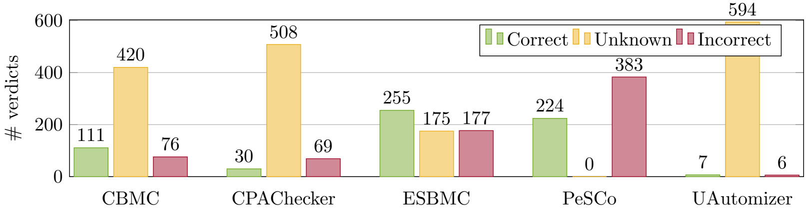


Figure: top software verifiers after 900 seconds **(August 2023)**.

Experiments with off-the-shelf verifiers

- We pick the top scoring tools from SV-COMP 2022
- We keep the same settings of the reachability category
- Some of these tools have competed for **decades**
- Variety of techniques: BMC, automata, portfolios

A short story (2)

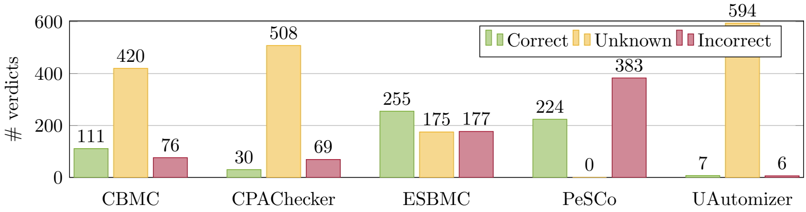


Figure: top software verifiers after 900 seconds **(August 2023)**.

Our opinion

- No support of mathematical libraries → incorrect results
- Cannot scale to large programs → unknown result (timeout)

Is there anything else at play here?


You discover that verifiers are broken
THIS IS FINE.
But that means more future work!
imgflip.com

A short story (3)

Reproducibility goal

- Submit *NeuroCodeBench* to SV-COMP 2023
- Experiments run by independent team
- Tool authors have a chance to fix bugs

Community engagement (October 2023)

- After some discussion[3] *NeuroCodeBench* is approved
- All future editions of SV-COMP will use it

Improve ESBMC (November 2023)

- At Manchester, we develop one of the top software verifiers
- *NeuroCodeBench* is breaking our own tool too!

[3] https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/issues/1396

A short story (4)

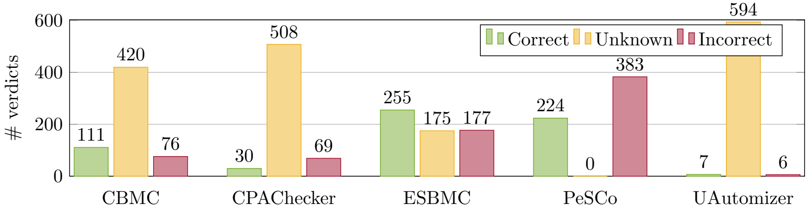


Figure: top software verifiers before SV-COMP **(August 2023)**.

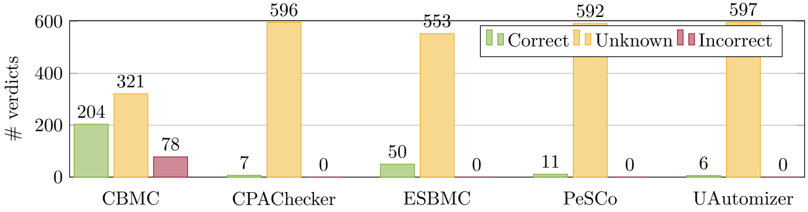


Figure: top software verifiers after SV-COMP **(December 2023)**.

A short story (5)

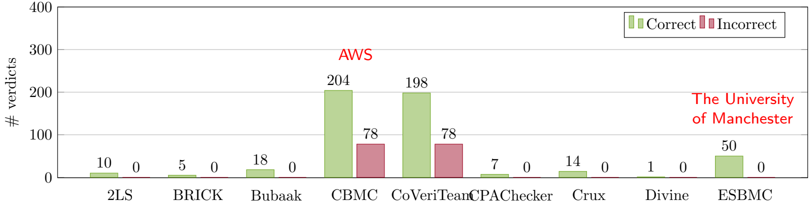


Figure: all software verifiers on *NeuroCodeBench* **(December 2023)**.

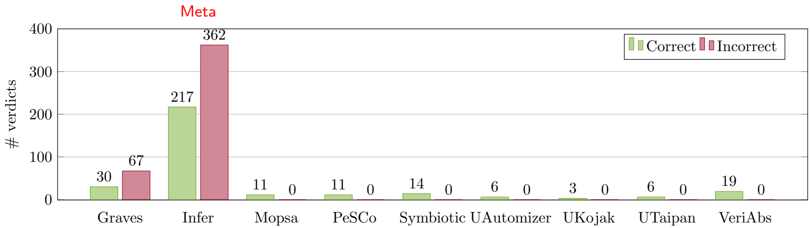
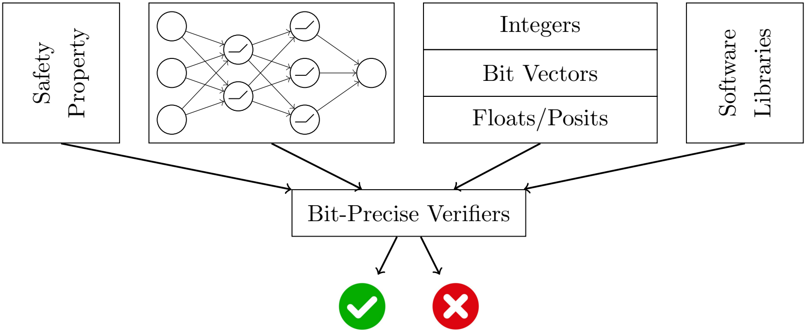


Figure: all software verifiers on *NeuroCodeBench* **(December 2023)**.

or so they thought...

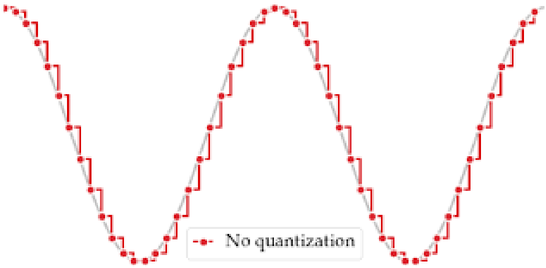
Bit-precise verification: what's next?



We need more precision!

- Support: ML dev pipeline (Python, ONNX, Apache TVM...)
- Scalability: code transformations, encoding, abstractions...
- Synthesis: verifying is not enough, let's *enforce* safety!

Neural Network Quantization (1)



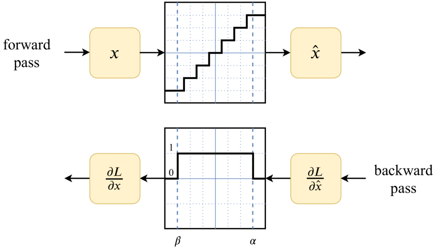
Why quantization?

- Old technique from signal processing/information theory
- Reduce memory footprint (e.g., store 8-bit weights)
- Reduce latency/power (full integer computation)

Neural Network Quantization (2)

Many Strategies

- Dynamic
- Post-Training
- Q-Aware Training
- Non-Uniform
- ...



Main differences

- Whether the weights and/or the activations are quantized
- Whether the weights are fine-tuned after quantization
- Whether the quantization is uniform (e.g., int 8-bit)

Quantisation and NN Equivalence

		Number of bits														
Safety Prop.		6	7	8	9	10	11	12	13		28	29	30	31	32
Set.	R_{40}	S	S	F	S	S	S	S	S	...	S	S	S	S	S
	R_{50}	S	S	F	F	F	F	F	F	...	F	F	F	F	S
Vers.	R_{20}	S	F	S	S	S	S	S	S	...	S	S	S	S	S
	R_{30}	S	F	S	S	S	S	S	S	...	S	S	S	S	S
	R_{40}	S	F	S	F	F	F	S	S	...	S	S	S	S	S
	R_{50}	S	F	F	F	F	F	F	F	...	F	F	F	F	F
Virg.	R_{20}	S	F	S	S	S	S	S	S	...	S	S	S	S	S
	R_{30}	S	F	S	S	S	S	S	S	...	S	S	S	S	S
	R_{40}	S	F	S	S	F	S	S	S	...	S	S	S	S	S
	R_{50}	S	F	F	F	F	F	F	F	...	F	F	F	F	F

Table: Effects of quantization on the safety of a NN trained on Iris data.

Effects of Quantisation

- Even if the accuracy does not drop, the behaviour may change
- Can we deploy *safe* quantized network?

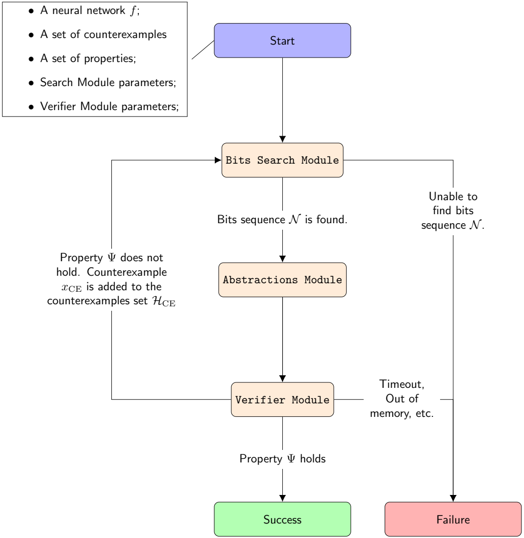
CEG4N: Counterexample-Guided NN Quantisation (1)

Quantisation

- Genetic algorithm
- Minimise bits
- Test equivalence

Verification

- Verify equivalence
- If not, generate counterexample
- Augment testset
- Repeat



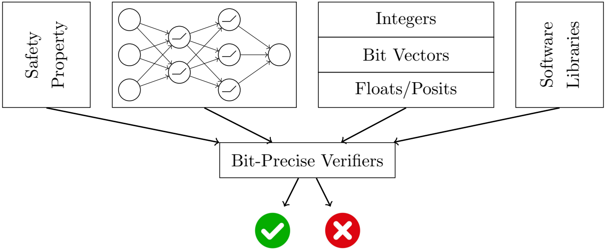
CEG4N: Counterexample-Guided NN Quantisation (2)

Model	Verifier	r	Iter.	Bits	Status
seeds_10x1	ESBMC	0.01	2	3,4	Success
		0.03	13	18,12	Success
		0.05	3	6,4	Timeout
	NNEQUIV	0.01	2	3,4	Success
		0.03	2	4,3	Success
		0.05	2	4,4	Success
seeds_15x1	ESBMC	0.01	4	5,2	Success
		0.03	5	8,6	Timeout
		0.05	7	8,7	Timeout
	NNEQUIV	0.01	2	5,2	Success
		0.03	2	4,4	Success
		0.05	3	5,3	Success

See all results in Batista et al., IEEE TCAD Journal (2023)

- Few iterations are needed to converge to a safe quantisation!

Summary



Bit-precise neural network verification

- Only way to avoid false positives/negatives
- Tricky! Current verifiers may be buggy
- Applications require scalability workarounds

Any questions?