



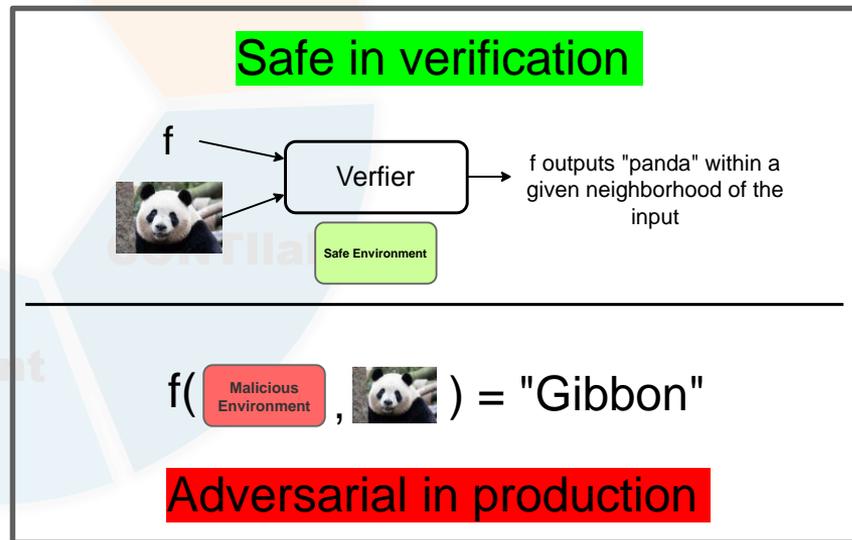
Name: Attila Szász, PhD student

Project type: laboratory project

Topic: On the Challenges of Sound Verification of Neural Networks

Supervisors: Dr. Balázs Bánhelyi

- Quantizing models to lower precision yields significant performance gains; however, many studies have shown that quantized models often suffer from considerable **accuracy degradation**.
- The main cause of this phenomenon is the effect of numerical representation on the accumulation of numerical errors, which highlights the importance of **clearly defining different environments** (e.g., training, evaluation, etc.).
- In this work, we highlighted the importance of precisely defining environments in the context of **verification**.
- Main idea: **The verifier operates under the assumption of an environment that differs from the one used in production.**
- This is a serious problem that can be exploited by an attacker.



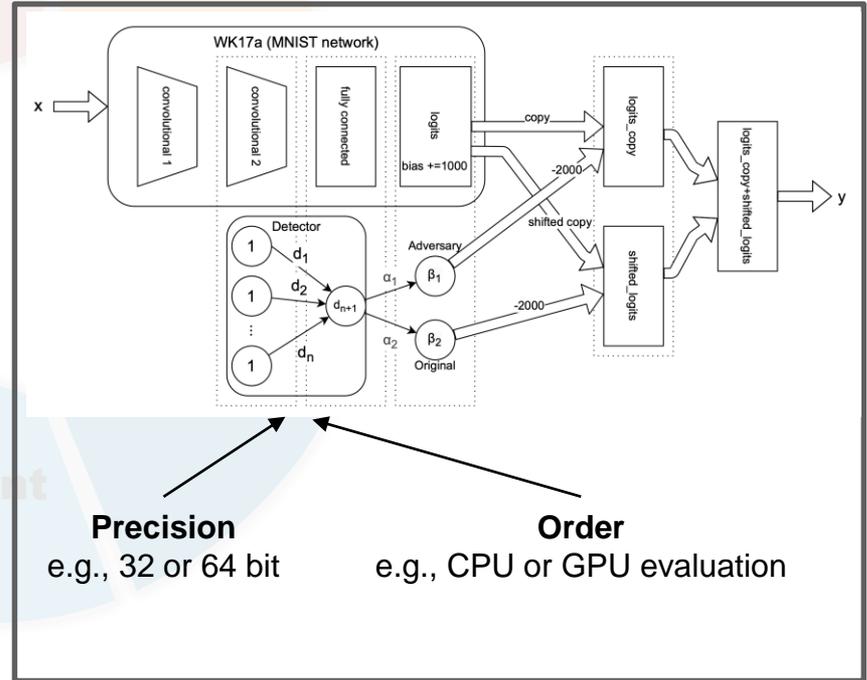
Backdoors

Research

- We defined backdoors that detect the underlying implementation (e.g., numerical representation or the order of summation) and **trigger malicious behavior only under specific environments**. (Typically **differs from the one used during verification**.)
- Main idea: Bound propagation algorithms used in verification are sensitive to numerical precision and operation order.

Expression tree	$(2^{53}+1) + 1$	$(1 + 2^{53}) + 1$	$(1 + 1) + 2^{53}$
FP64 result	2^{53}	2^{53}	$2^{53} + 2$
Exact value	$2^{53} + 2$	$2^{53} + 2$	$2^{53} + 2$
Interval bounds	$[2^{53}, 2^{53} + 4]$	$[2^{53}, 2^{53} + 4]$	$[2^{53}+2, 2^{53} + 2]$

- Precision detector: $2^{24} + 1 - 2^{24}$
- Order detector: $\underbrace{1 + \dots + 1}_{h \times} + w - w$



Results – ICML 2025 Spotlight - Vancouver

Verifier	Ver. Env.	Bounding	Precision	Order1	Order2	Order3
MIPVerify (Tjeng et al., 2017)	64-bit, CPU	IBP	unsound	sound	unsound	unsound
MN-BAB (Ferrari et al., 2022)	64-bit, GPU	Polyhedra	unsound	sound	unsound	unsound
β -CROWN BaB (Wang et al., 2021)	32-bit, CPU	Polyhedra	unsound	sound	unsound	unsound
β -CROWN BaB (Wang et al., 2021)	64-bit, CPU	Polyhedra	unsound	sound	unsound	unsound
β -CROWN BaB (Wang et al., 2021)	64-bit, GPU	Polyhedra	unsound	sound	unsound	unsound
GCP-CROWN (Zhang et al., 2022)	64-bit, CPU	Polyhedra	unsound	sound	unsound	unsound
DeepPoly (Singh et al., 2019a)	64-bit, CPU	Polyhedra	unsound	sound	sound	unsound
RefinePoly (Singh et al., 2019a)	64-bit, CPU	Polyhedra	unsound	sound	sound	unsound
DeepZono (Singh et al., 2018)	64-bit, CPU	Zonotope	unsound	sound	sound	unsound
RefineZono (Singh et al., 2019b)	64-bit, CPU	Zonotope	unsound	sound	sound	unsound

No Soundness in the Real World: On the Challenges of the Verification of Deployed Neural Networks

Attila Szász¹ Balázs Bánhegyi¹ Márk Jelasity^{1,2}

The ultimate goal of verification is to guarantee the safety of deployed neural networks. However, we claim that all the state-of-the-art verifiers are an order of magnitude slower than they should be. We argue that the bottleneck of verification is not full precision solving, but comparing with floating point ones using provably correct methods. Showing the floating point outputs in a particularly interesting environment. We prove this observation for the approximations that are commonly used in safety-critical hardware verification, such as interval analysis and bit-sets. We also argue that underlying practical verification is still inherently linear complexity. We report on state-of-the-art verifiers as well as evaluating our novel verification methods. We conclude that verifiers, we create advanced methods that detect and exploit features of the deployment environment, such as the order and precision of floating point operations. We demonstrate that all the neural verifiers are vulnerable to our deployment-specific attacks, which prove that they are not practically sound.

1. Introduction

The formal verification of an artificial neural network provides a mathematical proof that the network has (or does not) meet a certain safety property. One common property to verify is the adversarial robustness (Szegedy et al., 2014) of classical networks, where we wish to prove that a subset of inputs are all assigned the same class label.

There is a wide variety of approaches that address this problem (see Section 2). However, these methods inevitably focus on the verification of the theoretical model of the

University of Szeged, Hungary. *E-mail: szasz@inf.u-szeged.hu
Research Center for Artificial Intelligence, Szeged, Hungary. *E-mail: banhegyi@inf.u-szeged.hu

Proceeding of the 41st International Conference on Machine Learning, Vancouver, Canada, PMLR 240, 2022. Copyright 2022 by the author(s).

full precision computation. Most verifiers carefully address floating point issues as well, but only as an afterthought in the way of verifying the theoretical model, e.g. (Singh et al., 2019a, 2018).

At the same time, deployment environments often introduce complications through hardware and software features, e.g. (Vidali et al., 2020; Shrivastava et al., 2018). This means that the verification of the theoretical network and the deployment of the deployed network are different problems. Parallel to our work, a similar observation was made by (Günther et al., 2022), where this problem is referred to as the environment gap.

We formally prove that a verifier that is theoretically sound does, indeed, the full-precision computation is not necessarily practically sound, that is, it might not bound the actual output correctly in a given deployment environment. The problem has practical implications. Recently, (Chen et al., 2022) demonstrated that adversarial attacks can be generated simply by perturbing the order of activation operations in the deployment environment.

We demonstrate a more severe, practically exploitable vulnerability as well. The deployed network is shown to be fundamentally different from the theoretical network because the behavior of the network on the target hardware is not deployment-robust. In fact, the deployment environment leads to a degradation of the adversarial robustness of the network. Our attack does not only potentially harmful behavior from the verifier if enough information about the deployment environment is available.

We summarize our contributions below:

- We prove that verifiers that are theoretically sound are not necessarily sound in deployed networks.
- We demonstrate that deployed networks may differ significantly from the full-precision models in the order of backpropagation gradients for features of the environment.

Acknowledgements

This work was supported by the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory and project TKP2021-NVA-09, implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

We thank the support by the PIA Project, a collaboration between the University of Szeged and Continental Autonomous Mobility Hungary Ltd. with the goal of supporting students' research in the field of deep learning and autonomous driving. We thank Andras Balogh for his initial feedback and our anonymous reviewers for the relevant



11 February 2026
Author, © Continental AG

Upcoming work and publication

Verifier	Ver. Env.	Bounding	Pr.	O1	O2	O3	Zombori et al. (2021)
MIPVerify (Tjeng et al. (2017))	64-bit, CPU	IBP	U	S	U	U	U
MN-BAB (Ferrari et al. (2022))	64-bit, GPU	Symbolic	U	S	U	U	U
β -CROWN BaB (Wang et al. (2021))	32-bit, CPU	Symbolic	U	S	U	U	[no 32-bit model]
β -CROWN BaB (Wang et al. (2021))	64-bit, CPU	Symbolic	U	S	U	U	S
β -CROWN BaB (Wang et al. (2021))	64-bit, GPU	Symbolic	U	S	U	U	S
GCP-CROWN (Zhang et al. (2022))	64-bit, CPU	Symbolic	U	S	U	U	S
DeepPoly (Singh et al. (2019a))	64-bit, CPU	Symbolic	U	S	S	U	S
RefinePoly (Singh et al. (2019a))	64-bit, CPU	Symbolic	U	S	S	U	U
DeepZ (Singh et al. (2018))	64-bit, CPU	Symbolic	U	S	S	U	S
RefineZono (Singh et al. (2019b))	64-bit, CPU	Symbolic	U	S	S	U	[Gurobi error]
FPSoundIBP (ours)	32-bit, CPU	IBP	S	S	S	S	S
FPSoundSymbolic (ours)	32-bit, CPU	Symbolic	S	S	S	S	S

- We developed an algorithm that is capable of handling the non-associative nature of floating-point operations.
- We implemented a neural network verifier incorporating this algorithm (FPSoundIBP and FPSoundSymbolic) and demonstrated that our approach is uniquely capable of bounding the aforementioned numerical error-based backdoors.
- This work is currently being prepared for publication.