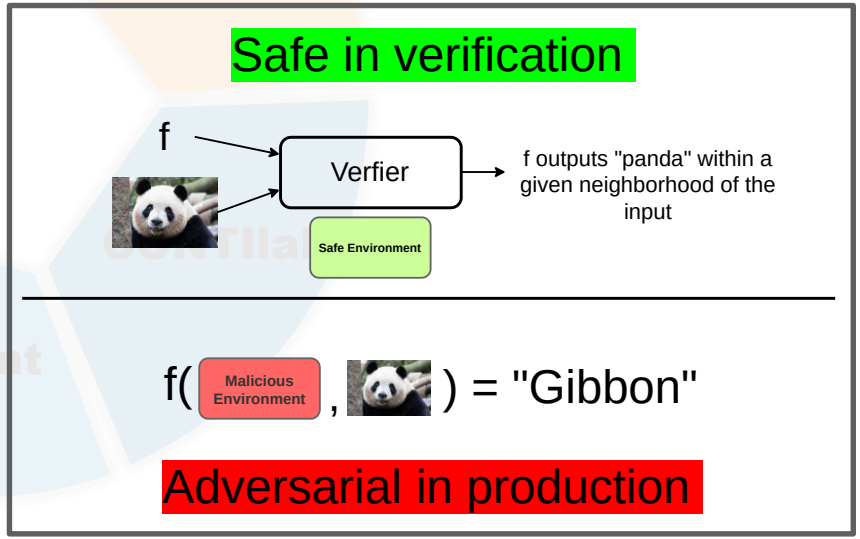
Name: Attila Szász, PhD student

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.

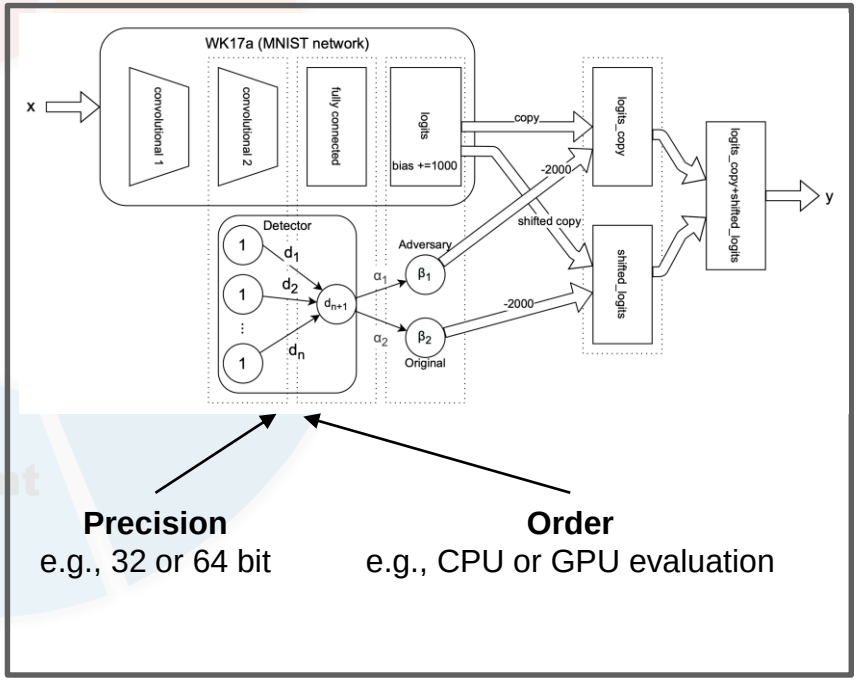


Backdoors

- 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	$[2^{53}, 2^{53}+4]$	$[2^{53}, 2^{53}+4]$	$[\mathbf{2^{53}+2}, \mathbf{2^{53}+2}]$

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



Results

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**

- We showed that none of the existing neural network verifiers are floating-point sound.
- The key issue is that we need to verify the network with the deployed environment.
- We constructed backdoors that practically exploit the misalignment between the production and verified environment.
- **ICML 2025 Spotlight**
- Attila Szász, Balázs Bánhelyi, Márk Jelasity. No Soundness in the Real World: On the Challenges of the Verification of Deployed Neural Networks

Publication

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

Attila Szász [1] Balázs Bánhelyi [1] Márk Jelasity [1 2]

Abstract

The ultimate goal of verification is to guarantee the safety of *deployed* neural networks. Here, we claim that all the state-of-the-art verifiers we are aware of fail to reach this goal. Our key insight is that *theoretical soundness* (bounding the full-precision output while computing with floating point) does not imply *practical soundness* (bounding the floating point output in a potentially stochastic environment). We prove this observation for the approaches that are currently used to achieve provable theoretical soundness, such as interval analysis and its variants. We also argue that achieving practical soundness is significantly harder computationally. We support our claims empirically as well by evaluating several well-known verification methods. To mislead the verifiers, we create adversarial networks that detect and exploit features of the deployment environment, such as the order and precision of floating point operations. We demonstrate that all the tested verifiers are vulnerable to our new deployment-specific attacks, which proves that they are not practically sound.

1. Introduction

The formal verification of an artificial neural network produces a mathematical proof that the network has (or does not have) a certain important property. One important property to verify is the adversarial robustness (Szegedy et al., 2014) of classifier 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 invariably focus on the verification of the theoretical model of the network, that is, they seek to characterize the behavior of the full-precision computation. Most solutions carefully address floating point issues as well, but only as an obstacle in the way of verifying the theoretical model, e.g. (Singh et al., 2019a; 2018).

At the same time, deployment environments affect numeric computation through hardware and software features, often resulting in stochastic behavior (Schlögl et al., 2023; Villa et al., 2009; Shanmugavelu et al., 2024). This means that the verification of the theoretical model and the verification of the deployed network are different problems. Parallel to our work, a similar observation was also made by (Cordeiro et al., 2025), where this problem is referred to as the *implementation gap*.

We formally prove that a verifier that is theoretically sound (i.e., bounds the full-precision output correctly) 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, (Shanmugavelu et al., 2025) demonstrated that adversarial inputs can be generated simply by permuting the order of associative 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 can be changed arbitrarily with *deployment-sensitive backdoors*.

Thus, *the deployment environment should be a fundamental part of any verification effort* because otherwise an attacker can hide potentially harmful behaviors 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 for deployed networks
- We demonstrate that deployed networks may differ arbitrarily from the full-precision model, with the help of backdoors triggered by features of the environment

[1]University of Szeged, Szeged, Hungary [2]HUN-REN–SZTE Research Group on Artificial Intelligence, Szeged, Hungary. Correspondence to: Attila Szász <szasz@inf.u-szeged.hu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

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