Robustness Verification of Traffic Sign Recognition: Practical Feasibility from a Safety Perspective*

Marvin Schneider[1], Jörn Schneider[1], Holger Voos[2] and Jürgen Kaiser[3]

[1]Dept. of Computer Science, Trier University of Applied Sciences, Trier, Germany
[2]Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
[3]3E-motion Ventures GmbH, Speyer, Germany

Abstract

The use of Machine Learning in safety-relevant applications such as Traffic Sign Recognition (TSR) in automated cars introduces new verification challenges. This paper presents an empirical study investigating the practical feasibility of formal verification of Convolutional Neural Network (CNN) based TSR. We systematically evaluated the local robustness of four architectures with increasing complexity using the α,β-CROWN verifier. Our experiments explore the multi-dimensional interplay between model capacity, training paradigms, and varying perturbation bounds to map the current verification frontier of TSR. We conclude that while formal verification provides essential insight into local robustness, it remains constrained by a significant semantic gap and computational scalability issues. Thus, L_∞-norm robustness verification should not be used as a stand-alone method but should be incorporated into an established safety strategy.

Keywords

Formal Verification, Machine Learning (ML), Robustness, Automotive Safety, Traffic Sign Recognition (TSR)

1. Introduction

Modern TSR systems are often based on CNNs and achieve high accuracy on benchmarks such as the German Traffic Sign Recognition Benchmark (GTSRB) [1, 2]. However, achieving high test accuracy is good but not sufficient for deployment in safety-relevant applications like modern Software-Defined Vehicles (SDVs).

Neural Networks (NNs) are prone to adversarial perturbations. For instance, minor changes to input pixels can alter the classification output [3]. A misclassification in an Automated Driving System (ADS)-equipped vehicle (i. e. SAE level 3 or above) could have serious consequences in the real-world. From an industrial perspective, the necessity for robustness extends beyond the classifier itself. The TSR system constitutes a critical link within the complex *cause-effect chains* of a vehicle's Electronic/Electric (E/E) architecture.

Safety standards such as ISO 21448 (SOTIF) require evidence that the residual risk from functional insufficiencies under reasonably foreseeable conditions is acceptable. Formal verification of neural networks has emerged as one approach to derive mathematical statements about bounded properties of a network. A current research challenge is to determine what guarantees today's tooling can deliver on a realistic NN pipeline, and which safety-relevant questions remain unaddressed upon completion of the verification process. The present work contributes to this question with an empirical study that pairs a state-of-the-art verifier with a verifiable TSR on local robustness.

The local L_∞ robustness we verify is a property of an individual image instance: it bounds how far the pixels of one captured image may deviate while the prediction is held fixed. What matters for safety is whether the sign is reliably correctly recognized. Keeping these two levels apart is essential for interpreting what local robustness verification contributes to a safety case.

1st Workshop on Explainability, Transparency, and Safety (ExTraSafe)

*This work is partially funded by the European Union and the State of Rhineland-Palatinate within the project "Model-based Safety of Automated Electric Vehicles" (SafeAIV).

marv.schneider@inf.hochschule-trier.de (M. Schneider); j.schneider@inf.hochschule-trier.de (J. Schneider); holger.voos@uni.lu (H. Voos); juergen.kaiser@3e-motion.com (J. Kaiser)

0009-0008-4213-8098 (M. Schneider); 0000-0001-8452-9216 (J. Schneider); 0000-0002-9600-8386 (H. Voos)

The contributions of this paper are the following:

- A reproducible empirical study of local L_∞-robustness verification on four CNN architectures of increasing complexity for TSR
- A comparison of the verification between standard Adam-trained and Projected Gradient Descent (PGD) adversarial trained models
- An application-oriented reflection on what such verification empirically establishes for an automotive safety argument and what it leaves unaddressed

The remainder of this paper is structured as follows. Section 2 situates the work in the verification, adversarial-training, and TSR literature. Section 3 details the architectures, the training experiments, and the verification property. Section 4 reports the observations across all experiments. Section 5 interprets these observations, with particular attention to the limitations of an L_∞-bounded specification for a real-world safety argument. Section 6 concludes and offers an outlook on future research.

2. Related Work

Modern TSR systems are often CNN-based classifiers with very high accuracies on the GTSRB for both standard deep architectures and lightweight, embedded-oriented models [1, 2, 4, 5].

Several formal verifiers (e. g. α,β-CROWN [6, 7, 8, 9], Marabou [10], NeuralSAT [11], etc.) are commonly applied to neural networks for robustness analysis. We select α,β-CROWN for this study as a state-of-the-art verifier with support for CNNs used for TSR [12, 13].

We selected α,β-CROWN for this study as a state-of-the-art verifier with support for CNNs. It has repeatedly achieved the highest overall score in the International Verification of Neural Networks Competition (VNN-COMP) [14, 12, 13], which makes it a reasonable proxy for what the current tool landscape resolves.

The inverse relation between model depth or width and verification tractability is well documented in recent surveys and assessments of the field [15, 16]. The present work does not aim for novelty on this axis. Rather, it provides a reproducible, application-specific quantification of the trade-off in a realistic TSR setting.

Direct prior work on formal verification of TSR is comparatively thin. A VNN-COMP 2023 contribution introduced a TSR-specific track focused on binarized neural networks, drawing three randomly selected test images per model from the GTSRB test set and evaluating perturbations at $\epsilon \in \{1, 3, 5, 10, 15\}/255$ across 45 specification files in total. α,β-CROWN performed strongest among the participating tools while still proving robustness for none of the TSR test-set instances (zero verified outcomes) [14, 12].

Adversarial training has established itself as one practical method to improve robustness against bounded perturbations. Goodfellow et al. introduced training on adversarial examples generated by Fast Gradient Sign Method (FGSM) [3], and Madry et al. established PGD-based adversarial training [17], which we will also use. Two possible side effects of adversarial training are a clean-accuracy respectively generalization reduction [18] and an overfitting tendency to the training perturbations [19]. Recent surveys provide a comprehensive overview [20].

A parallel research line characterizes physical-world adversarial threats for TSR empirically, including printed stickers that reliably induce misclassification under realistic viewing conditions [21], surveys of the practical attack surface for TSR [22], and evaluations of physical-world adversarial robustness [23]. This line is complementary to formal verification rather than competing with it: physical-attack work documents failure modes empirically, while verification establishes formally-guaranteed regions for the well-defined perturbation models it can express. Our case study contributes to the verification part of this multi-method picture, and we record the scope of our claims, including the boundary toward physically realizable perturbations, explicitly in Section 5.

In summary, the related work above shows that scaling formal verification, raising the verifiable fraction through adversarial training, and constraining architectures for verifiability are active research

lines. Empirical evidence on how a state-of-the-art verifier behaves on a realistic, application-near traffic-sign-recognition pipeline, however, remains comparatively sparse.

3. Methodology

This section describes the experimental setup, two TSR training experiments on a common architecture family and a verification protocol with explicit property and outcome categories.

3.1. Verification Property

The property covered by our verification is local robustness under the L_∞-norm. Given an input x with predicted class $c = f(x)$ for the network f, the property holds if and only if

$$\forall x' \text{ with } \left\|x' - x\right\|_\infty \leq \epsilon : \quad f(x') = f(x).$$

So, every perturbed instance x' within the L_∞-ball of radius ϵ around x is assigned the same class as x by f [24]. Crucially, this is a purely local statement about the specific instance x and does not characterize the model's behavior globally.

3.2. Dataset and Classifier Architectures

Experiments use the GTSRB [25] (43 traffic-sign classes, approximately 50 000 images). All images are resized to 32×32 pixels and normalized channel-wise. To prevent data leakage from consecutive video frames and ensure independent (i.i.d.) partitions, we split the GTSRB data at the track level in an 80/20 train/validation ratio, retaining the official test set for evaluation and verification. Standard augmentation (rotation, color jitter, affine, and random erasing) is applied to the training split and is held identical across both training experiments.

We use four CNNs of increasing capacity: *XS*, *S*, *M*, and *L*[1]. Each network is a feed-forward stack of convolutional blocks (`Conv–BatchNorm–ReLU–AvgPool`) followed by an adaptive average pool and a fully connected classifier (softmax). Layer types are restricted to `Conv`, `Linear`, `BatchNorm`, `ReLU`, and `AvgPool`, an operator subset that α,β-CROWN (among others) supports with linear bound propagation and that PGD-based adversarial training can handle well without architectural modification. The restriction to CNNs is deliberate. They remain a dominant architecture family for TSR [1, 2, 4]. To get a good comparability, we only wanted to change the model sizes, not the architecture itself.

3.3. Training Process

Two training experiments are run on identical architectures, dataset, split, and augmentation pipeline. The *baseline experiment* is configured with Adam, a learning rate of 10^{-3}, a batch size of 64, cross-entropy loss, and cosine learning-rate annealing over 100 epochs. The *adversarial experiment* shares every hyperparameter with the standard experiment except that each training batch is replaced by its 7-step PGD adversarial counterpart with step size $\alpha = 2/255$, perturbation budget $\epsilon_{\text{train}} = 8/255$, and random initialization within the L_∞-ball.

3.4. Verification Protocol

Verification is performed at four perturbation budgets, $\epsilon \in \{1, 2, 4, 8\}/255$, expressed in raw pixel units. The values span the standard sweep used in the empirical-robustness literature [12, 13]. The complete experimental matrix consists of 4 architectures $\times$ 4 budgets $\times$ 2 training experiments $= 32$ cells. For each cell, 10 distinct images per class are drawn randomly from the held-out GTSRB test

[1]The naming *XS–L* reflects relative complexity within this study and is not tied to externally defined size classes. A detailed specification of the models can be found in the Appendix A.

split, yielding $10 \times 43 = 430$ verification instances per cell[2]. The same instance set is reused across all 32 cells, so that comparisons across architectures, budgets, and training experiments share identical inputs. The per-item verification timeout is set to 300 s. The verifier returns one of three outcomes per instance. *Verified* means the property is proven to hold for the entire L_∞-ball of radius ϵ, *falsified* means a concrete perturbation within the ball was found that changes the prediction, and *timeout* means that neither result was reached within the time limit.

4. Results

This section reports experimental observations across the four architectures, budgets, and training experiments. We present the training metrics, the verification outcomes, and how the verifier behaves on these instances. Detailed classifier performance, runtime and resource profiles and additional tables and plots are provided in Appendix A[3].

4.1. Training

Table 1
Test-set accuracy and macro-averaged F1 of the four CNN sizes for both training experiments.

	XS	*S*	*M*	*L*
Baseline Training				
Test accuracy	94.8%	97.2%	97.5%	98.1%
Test F1 (macro)	0.9128	0.9573	0.9571	0.9745
Adversarial Training				
Test accuracy	41.2%	64.8%	79.5%	84.4%
Test F1 (macro)	0.2743	0.6450	0.8022	0.8448

Table 1 reports test-set accuracy and macro-F1 for the four classifiers under both training experiments. Baseline training reaches good performance and adversarial training exposes the well-known robustness-accuracy trade-off [18], leaving the smallest model with very poor performance. The cost of adversarial training decreases with capacity, from 53.6 percentage points for *XS* to 13.7 for *L*, so the two training experiments differ least where the models are largest.

4.2. Verification Outcomes

Figures 1–3 report the verification outcomes across all 32 experimental cells, each of which comprises 430 instances. Every outcome is reported as a rate, so the three figures jointly account for all instances of a cell. The complete breakdown, including absolute counts and the runtimes is provided in Appendix A.

The results reveal a contrast between training objectives: while standard-trained models are only partially verifiable at the smallest budget ($\epsilon = 1/255$) and collapse to near-zero verified rates at $\epsilon \geq 2/255$, the PGD adversarially trained models retain non-trivial verified rates up to $\epsilon = 4/255$. At $\epsilon = 8/255$ only the *XS* model retains a small verified fraction (1.9%), while *S*, *M*, and *L* verify no instance under either training experiment. For the model sizes S–L, adversarial training increases the verified rate at every budget up to $\epsilon = 4/255$. At $\epsilon = 1/255$, the total number of verified instances ranges from 14 to 301, with the largest gain from PGD training observed for the *L* model ($14 \rightarrow 248$, i. e. $3.3\% \rightarrow 57.7\%$).

Figure 1 shows the verified rate in both trainings. Under baseline training the rate decreases monotonically with capacity at $\epsilon = 1/255$, from 29.5% (*XS*) to 3.3% (*L*). Under adversarial training it's the

[2]This sample size was chosen to maintain a manageable overall runtime.
[3]We used the High Performance Computing Cluster at the RPTU Kaiserslautern-Landau (Elwetritsch) for our experiments, which is coordinated by the Alliance for High Performance Computing Rhineland-Palatinate (AHRP).

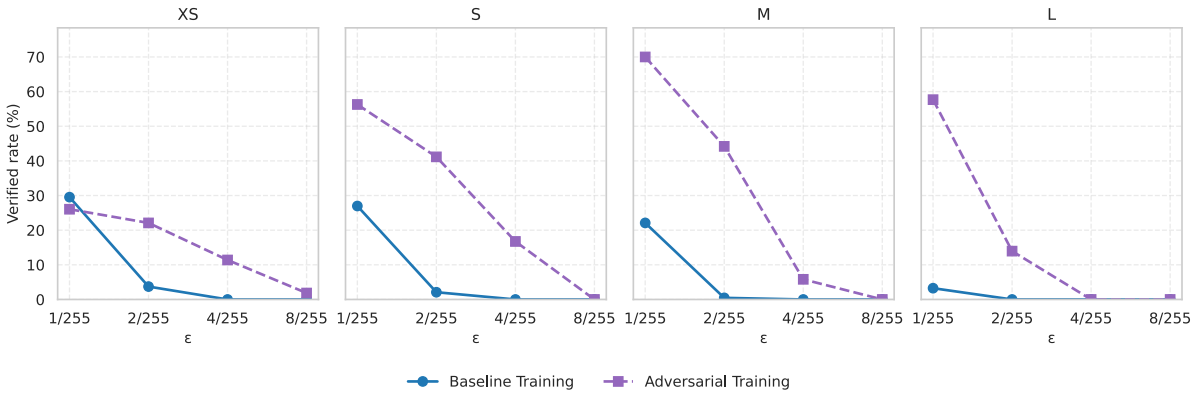

Figure 1: Verified rate over the perturbation budget ϵ, reported as a fraction of the 430 instances per cell. Each panel corresponds to one model size, the solid curve denotes baseline training and the dashed curve adversarial training.

same monotonically decrease and the rate peaks at the *M* model with 70.0% and falls to 26.0% for *XS* (for $\epsilon = 1/255$). The *XS* model seems to be too small to learn features that are simultaneously accurate and robust.

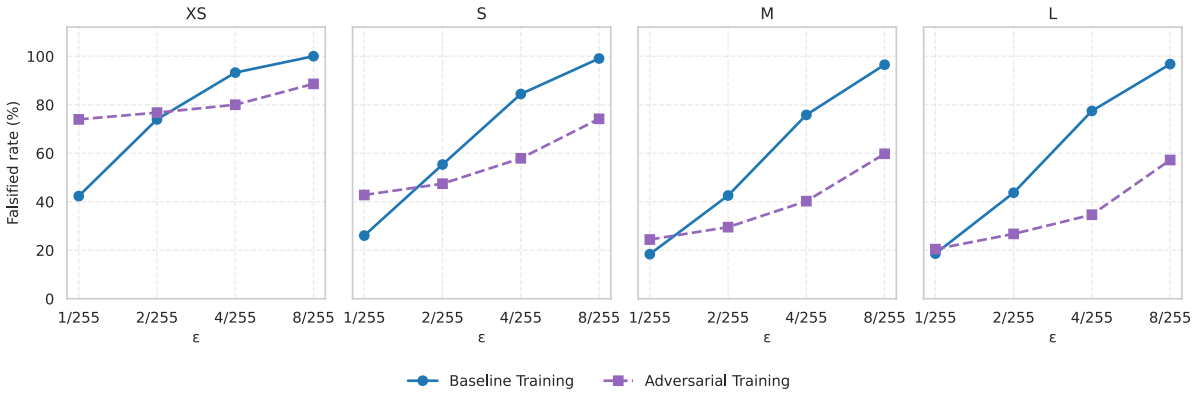

Figure 2: Falsified rate over the perturbation budget ϵ, in the same layout as Figure 1.

Figure 2 reports the falsified rate. At $\epsilon = 8/255$ adversarial training lowers the rate substantially, from 96.5–100% under baseline training to 57.2–88.6%. At $\epsilon = 1/255$ it raises the rate for every architecture instead, most strongly for *XS* (42.3% → 74.0%) and least for *L* (18.6% → 20.5%), because instances that timed out under baseline training are now decided and a substantial share of them is decided negatively. The gain in decisiveness from adversarial training is therefore not exclusively a gain in verified outcomes. Under baseline training the falsified rate at $\epsilon = 1/255$ additionally falls with capacity, from 42.3% for *XS* to 18.4% for *M*, before levelling off, so larger models are not only harder to verify but also harder to falsify.

Figure 3 makes the complementary quantity explicit. Since verified and falsified outcomes sum to the decided fraction, the decisiveness of the verifier at $\epsilon = 1/255$ falls from 71.9% to 21.9% across the baseline models, whereas it stays between 100% and 78.1% across the adversarially trained ones. Summed over all four budgets, adversarial training reduces the number of timeouts for *XS* (246 → 83), *S* (456 → 273), and *M* (619 → 542), but increases it for *L* (689 → 807), the highest count in the entire matrix. Adversarial training therefore improves decisiveness only while the model remains small enough. At the largest capacity, the instances that are no longer quickly falsified are not resolved within the timeout either.

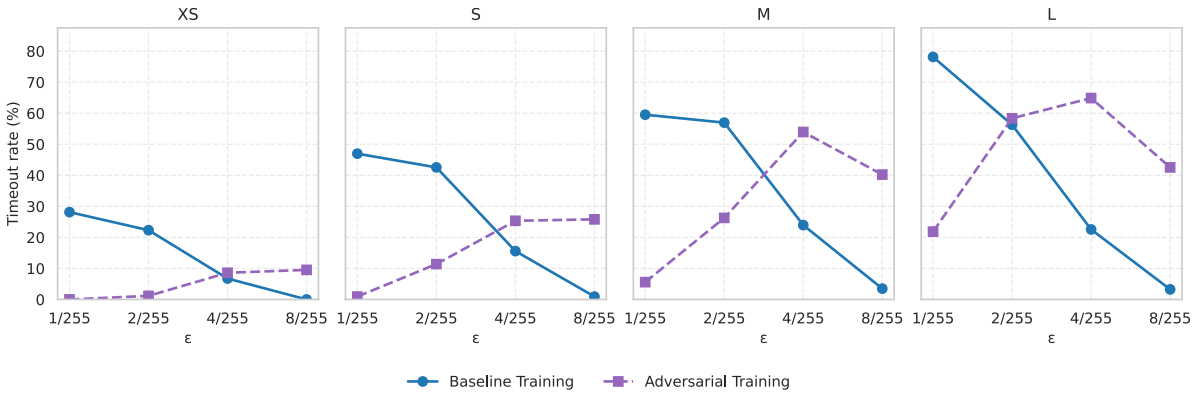

Figure 3: Timeout rate over the perturbation budget ϵ, in the same layout as Figure 1.

4.3. Runtime Behaviour

The outcome rates characterise what the verifier achieved within the given time limit, but not how it used that limit. This subsection therefore turns to the runtime side of the experiments, first to the distribution of the results over the available computation time and then to a structural property of the instances that indicates how much work a proof may require.

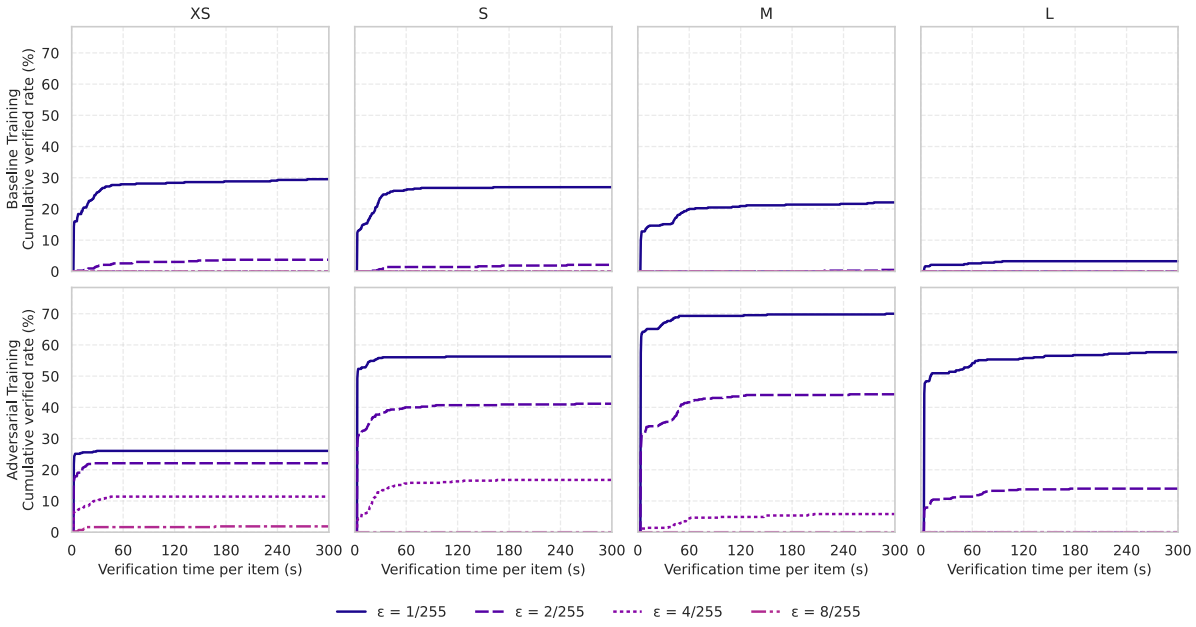

Figure 4: Cumulative verified rate as a function of the per-item verification time, up to the 300 s timeout. Columns correspond to model size, rows to the training experiment, and the four curves within each panel to the perturbation budgets ϵ.

Figure 4 shows how the verified rate of each cell accumulates over the per-item verification time. Across all configurations the curves rise steeply within the first seconds and then flatten, so the greater part of the final verified rate is already reached long before the timeout. This raises the question of how much additional time would have been required to resolve the remaining instances, which our data cannot answer exactly. A timeout is uninformative in this respect, as it reports that no result was reached but not how close the search was to reaching one. The curves in Figure 4 constrain the answer from one side only. Within the observed range the return per additional second decreases markedly, it

can therefore be assumed that increasing the timeout slightly would not have led to significantly more results. To further support this statement, a quantity that is available independently of the time actually spent is therefore useful, and Figure 5 reports one such quantity.

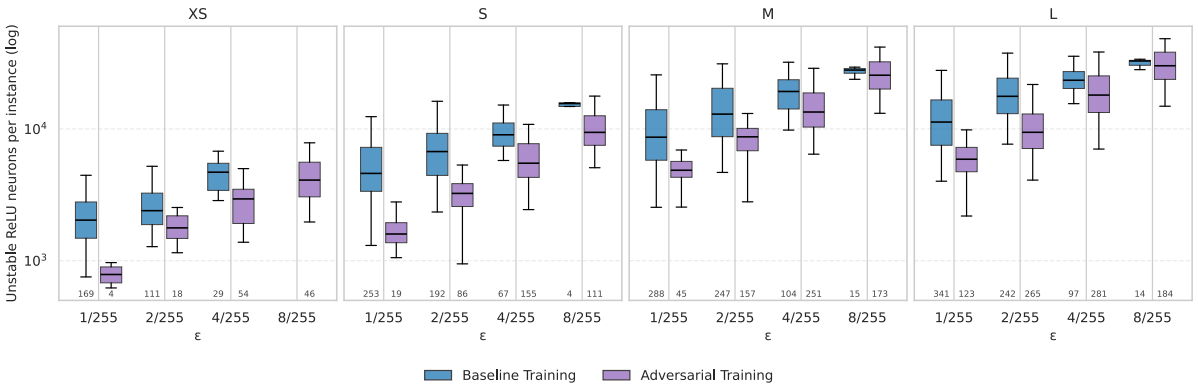

Figure 5: Number of unstable Rectified Linear Unit (ReLU) neurons per instance, that is, neurons for which the propagated bounds admit both an active and an inactive state within the perturbation region. Because bound propagation alone cannot resolve them, these are the neurons the verifier may have to split during branch-and-bound. Each panel corresponds to one model size and each budget carries a pair of boxes. The value below each box states the number of instances it aggregates.

A ReLU neuron is unstable if its pre-activation bounds over the perturbation region straddle zero, so that both states remain possible and the neuron must be over-approximated. Removing that approximation requires splitting the problem into one subproblem per state, so the number of unstable neurons delimits the search space a proof may require.

This makes the quantity an indication of difficulty rather than a measure of it. Many unstable neurons mean that many splits are *possible*, not that they are performed. An instance may still be verified directly if the initial approximation is tight enough, or falsified within seconds by an adversarial example, and the verifier only ever splits a small selection of the candidates. Adversarial training reduces it at a fixed capacity and budget, which is consistent with its effect on the verified rate. Larger models consistently exhibit more unstable neurons, each of which may have to be split before the verifier reaches a result, and since every split doubles the number of subproblems, the search space grows exponentially in the number of candidates. It is therefore to be expected that a moderately larger timeout would not have changed our results substantially.

5. Discussion

Our results highlight the practical tension between predictive performance and formal verifiability on a realistic TSR system. While PGD adversarial training expands the verification frontier, sustaining non-trivial verified rates at larger budgets and reducing verifier timeouts compared to the standard baseline, scalability remains a major bottleneck.

Beyond this verification frontier, a fundamental limitation remains. The semantic gap: even at the widest budget $\epsilon = 8/255$, the verified L_∞-ball covers only very small, pixel-wise bounded deviations that remain imperceptible to a human observer. Verification therefore guarantees invariance against an essentially imperceptible perturbation family (on the verified instances), useful against quantization, read-noise, and small sensor jitter, but offers no guarantees about spatially-correlated phenomena such as rain streaks, partial occlusion, motion blur, etc., which typically induce L_∞ deviations far beyond any tractable ϵ. This distinction matters for the safety argument itself. What this successful verification establishes is sound but narrow. Within a small L_∞-ball that bounds every pixel by ϵ, the predicted class cannot change (if verified), but that ball is a perceptually arbitrary region with no semantic meaning. Even for the single instance, it excludes most changes a human would still read as the same sign: mild

geometric or photometric variations can already correspond to far larger L_∞ distances and therefore fall outside any tractable ϵ. The reasonably foreseeable conditions ISO 21448 calls for thus lie largely outside what an L_∞ guarantee can express.

From a safety perspective, three operationally distinct outcomes matter differently. A *falsified* outcome is an actionable result: the verifier hands the developer an (actionable) counterexample. A *verified* outcome is a strong but narrow guaranty, valid only for the specific instance, ϵ, and norm queried. A *timeout* is the worst outcome for a safety case: it leaves the robustness status of an input genuinely unknown, and our data show that timeouts dominate exactly where they hurt the most, at larger model capacities and ϵ under adversarial training. A safety argument that quotes only an aggregate verified rate therefore hides where the residual risk actually lies. Increasing computational capacity or extending individual timeouts may salvage some unresolved instances, but the observed monotonic growth of timeouts alongside model size points to a fundamental limitation. Because this issue scales systematically with deeper architectures, relying solely on increased computational resources is unsustainable. Complementary system-level safety mechanisms such as fallback strategies, plausibility checks, runtime Out-of-Distribution (OoD) monitoring etc., must be integrated.

It is essential to recognize that robustness does not equate to correctness. A highly robust network might remain invariant to perturbations but consistently predict the wrong class, resulting in being "robustly wrong." Moreover, the severity of misclassifications is highly context-dependent: confusing a *Stop* sign with a *Priority road* poses a catastrophic risk, whereas other errors may be less critical. Because aggregate test accuracy or overall verified rates can quietly mask severe vulnerabilities in individual safety-relevant classes, verification targets and acceptance criteria must be explicitly tailored to the specific Operational Design Domain (ODD) and acceptable risk profile. Crucially, achieving high local robustness for a specific subset of samples does not justify broader claims about the network's global reliability or its behavior across the entire input space.

Finally, these observations point to a problematic trade-off for automotive Artificial Intelligence (AI): developers may be compelled to utilize simpler models to remain within computational verification bounds, potentially sacrificing average-case accuracy for tractable worst-case guarantees. While our study is bounded by specific choices, such as a single verifier (α,β-CROWN), the L_∞ perturbation model, and compact CNN architectures, the observations highlight that we are currently not at a stage where standard CNNs of a sufficient size can be easily verified for local robustness, particularly when moving beyond academic examples toward more realistic, high-dimensional applications. Post-hoc verification of such architectures remains highly difficult for complex TSR systems. The underlying cause, the growth of the complexity of the problem with network capacity (especially the non-linearities), is not specific to the tool we chose. We therefore expect the direction of the observed effect to carry over beyond our setting, given today's tooling landscape. This suggests that future safety-relevant perception pipelines may need to consider verifiability as an upfront constraint at the level of architecture and training, rather than relying solely on retroactive checks. Furthermore, a comprehensive safety argument for deployed TSR systems cannot rely on bounded verification alone, but must integrate complementary evidence such as runtime OoD monitoring, redundant sensing, physical-world threat assessments, etc.

6. Conclusion

This work presented an empirical study of formal L_∞-robustness verification for TSR, systematically evaluating four CNN architectures of increasing capacity across four perturbation budgets under both standard and PGD adversarial training.

Our results yield three concrete takeaways. First, the verification frontier degrades rapidly with model depth: even at the smallest budget $\epsilon = 1/255$, larger architectures produce predominantly unresolved (timeout) outcomes, representing a blind spot that cannot be resolved by simply increasing computational resources for larger models. Second, adversarial training significantly expands the verification frontier and improves verifier decisiveness across all architectures and budgets, but at the cost of clean accuracy, and without eliminating the fundamental scalability problem for larger models.

Third, and most importantly for practitioners, a verified L_∞-property is a sound but narrow claim: it covers only imperceptibly small, uncorrelated pixel-level deviations and leaves a persistent semantic gap to the geometric distortions, weather artifacts, and OoD inputs that constitute the reasonably foreseeable conditions under ISO 21448.

Taken together, these findings suggest that formal verification should not be treated as a standalone safety argument, but as one precisely scoped component within a broader safety strategy. Verifiability should be addressed as an upfront architectural constraint, and formal bounded proofs should be complemented by methods that attach semantic meaning to verified properties and by comprehensive runtime monitoring.

Data Availability Statement

The source code for the training of the different networks and robustness verification, and detailed instructions on how to reproduce the results presented in this paper are available in the Zenodo repository https://zenodo.org/records/20347146.

Declaration on Generative AI

During the preparation of this work, the author(s) used Claude, Writefull and DeepL in order to: Grammar, spelling check and translating words. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] H. Chen, M. A. H. Ali, Y. Nukman, B. A. Razak, S. Turaev, Y. Chen, S. Zhang, Z. Huang, Z. Wang, R. Abdulghafor, Computational methods for automatic traffic signs recognition in autonomous driving on road: A systematic review, Results in Engineering 24 (2024) 103553. doi:10.1016/j.rineng.2024.103553.

[2] X. R. Lim, C. P. Lee, K. M. Lim, T. S. Ong, A. Alqahtani, M. Ali, Recent Advances in Traffic Sign Recognition: Approaches and Datasets, Sensors 23 (2023) 4674. doi:10.3390/s23104674.

[3] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and Harnessing Adversarial Examples, CoRR (2014).

[4] M. Flores-Calero, C. A. Astudillo, D. Guevara, J. Maza, B. S. Lita, B. Defaz, J. S. Ante, D. Zabala-Blanco, J. M. Armingol Moreno, Traffic Sign Detection and Recognition Using YOLO Object Detection Algorithm: A Systematic Review, Mathematics 12 (2024) 297. doi:10.3390/math12020297.

[5] I. Benfaress, A. Bouhoute, A. Zinedine, Advancing Traffic Sign Recognition: Explainable Deep CNN for Enhanced Robustness in Adverse Environments, Computers 14 (2025) 88. doi:10.3390/computers14030088.

[6] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, L. Daniel, Efficient neural network robustness certification with general activation functions, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, Curran Associates Inc., Red Hook, NY, USA, 2018, pp. 4944–4953.

[7] K. Xu, Z. Shi, H. Zhang, Y. Wang, K.-W. Chang, M. Huang, B. Kailkhura, X. Lin, C.-J. Hsieh, Automatic perturbation analysis for scalable certified robustness and beyond, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020, pp. 1129–1141.

[8] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, C.-J. Hsieh, Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers, ArXiv (2020).

[9] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, Z. Kolter, Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification, in:

Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS '21, Curran Associates Inc., Red Hook, NY, USA, 2021, pp. 29909–29921.

[10] H. Wu, O. Isac, A. Zeljić, T. Tagomori, M. Daggitt, W. Kokke, I. Refaeli, G. Amir, K. Julian, S. Bassan, P. Huang, O. Lahav, M. Wu, M. Zhang, E. Komendantskaya, G. Katz, C. Barrett, Marabou 2.0: A Versatile Formal Analyzer of Neural Networks, 2024. doi:10.48550/arXiv.2401.14461. arXiv:2401.14461.

[11] H. Duong, T. Nguyen, M. B. Dwyer, NeuralSAT: A High-Performance Verification Tool for Deep Neural Networks, in: R. Piskac, Z. Rakamarić (Eds.), Computer Aided Verification, Springer Nature Switzerland, Cham, 2025, pp. 409–423. doi:10.1007/978-3-031-98679-6_19.

[12] C. Brix, S. Bak, T. T. Johnson, H. Wu, The Fifth International Verification of Neural Networks Competition (VNN-COMP 2024): Summary and Results, 2024. doi:10.48550/arXiv.2412.19985. arXiv:2412.19985.

[13] K. Kaulen, T. Ladner, S. Bak, C. Brix, H. Duong, T. Flinkow, T. T. Johnson, L. Koller, E. Manino, T. H. Nguyen, H. Wu, The 6th International Verification of Neural Networks Competition (VNN-COMP 2025): Summary and Results, 2025. doi:10.48550/arXiv.2512.19007. arXiv:2512.19007.

[14] C. Brix, S. Bak, C. Liu, T. T. Johnson, The Fourth International Verification of Neural Networks Competition (VNN-COMP 2023): Summary and Results, 2023. doi:10.48550/ARXIV.2312.16760.

[15] M. Koenig, A. W. Bosman, H. H. Hoos, J. N. Van Rijn, Critically assessing the state of the art in neural network verification, Journal of Machine Learning Research 25 (2024) 1–53.

[16] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, X. Yi, A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability, Computer Science Review 37 (2020) 100270. doi:10.1016/j.cosrev.2020.100270.

[17] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards Deep Learning Models Resistant to Adversarial Attacks, in: International Conference on Learning Representations, 2018.

[18] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, A. Mądry, Robustness May Be at Odds with Accuracy, arXiv: Machine Learning (2018).

[19] L. Rice, E. Wong, J. Z. Kolter, Overfitting in adversarially robust deep learning, in: Proceedings of the 37th International Conference on Machine Learning, volume 119 of *ICML'20*, JMLR.org, 2020, pp. 8093–8104.

[20] T. Bai, J. Luo, J. Zhao, B. Wen, Q. Wang, Recent Advances in Adversarial Training for Adversarial Robustness, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, Montreal, Canada, 2021, pp. 4312–4321. doi:10.24963/ijcai.2021/591.

[21] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song, Robust Physical-World Attacks on Deep Learning Visual Classification, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, Salt Lake City, UT, USA, 2018, pp. 1625–1634. doi:10.1109/CVPR.2018.00175.

[22] S. Pavlitska, L. Müller, J. M. Zöllner, Evaluating Adversarial Attacks on Traffic Sign Classifiers Beyond Standard Baselines, in: 2024 International Conference on Machine Learning and Applications (ICMLA), 2024, pp. 1390–1395. doi:10.1109/ICMLA61862.2024.00216.

[23] N. Wang, S. Xie, T. Sato, Y. Luo, K. Xu, Q. A. Chen, Revisiting Physical-World Adversarial Attack on Traffic Sign Recognition: A Commercial Systems Perspective, in: Proceedings 2025 Network and Distributed System Security Symposium, Internet Society, San Diego, CA, USA, 2025. doi:10.14722/ndss.2025.230090.

[24] G. Katz, C. Barrett, D. L. Dill, K. Julian, M. J. Kochenderfer, Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks, in: R. Majumdar, V. Kunčak (Eds.), Computer Aided Verification, Springer International Publishing, Cham, 2017, pp. 97–117. doi:10.1007/978-3-319-63387-9_5.

[25] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Neural Networks 32 (2012) 323–332. doi:10.1016/j.neunet.2012.02.016.

A. Appendix

Architectural Specification of the Classifiers

Property	xs	s	m	l
#Parameters	8,683	30,891	116,011	280,619
#Conv blocks	3	3	3	4
Filter sequence	$3 \to 8 \to 16 \to 32$	$3 \to 16 \to 32 \to 64$	$3 \to 32 \to 64 \to 128$	$3 \to 32 \to 64 \to 128 \to 128$
#Pool ops	3	3	3	4
#ReLU (total)	4	4	4	6
#ReLU (conv / FC)	3 / 1	3 / 1	3 / 1	4 / 2
#BatchNorm layers	4	4	4	6
#Dropout layers	1	1	1	2
#FC layers	2	2	2	3
FC widths	$32 \to 32 \to 43$	$64 \to 64 \to 43$	$128 \to 128 \to 43$	$128 \to 128 \to 128 \to 43$

Table 2
Architectural metadata of the four TSR model variants. Reported are the number of trainable parameters, the number of conv blocks together with their filter sequence, the number of pooling operations, ReLU activations, BatchNorm and Dropout layers, and the widths of the fully connected layers.

Classifier Performance

Model	Val acc	Val F1	Test acc	Test F1
Baseline Training				
XS	95.6%	0.9321	94.8%	0.9128
S	98.0%	0.9699	97.2%	0.9573
M	98.5%	0.9751	97.5%	0.9571
L	98.8%	0.9848	98.1%	0.9745
Adversarial Training				
XS	42.8%	0.3017	41.2%	0.2743
S	65.6%	0.6204	64.8%	0.6450
M	78.3%	0.7611	79.5%	0.8022
L	82.9%	0.8262	84.4%	0.8448

Table 3
Training and test-set metrics by training method and model size. Val/Test accuracy reported as percentages; F1 is macro-averaged.

Full Verification Runtime Statistics

Model	ε	N	Verif.	Fals.	Timeout	Error	Verif. rate	Total time
Baseline Training								
xs	1/255	430	127	182	121	0	29.5%	10h 58min 6s
xs	2/255	430	16	318	96	0	3.7%	8h 26min 17s
xs	4/255	430	0	401	29	0	0.0%	2h 32min 6s
xs	8/255	430	0	430	0	0	0.0%	4min 51s
s	1/255	430	116	112	202	0	27.0%	17h 33min 33s
s	2/255	430	9	238	183	0	2.1%	15h 41min 46s
s	4/255	430	0	363	67	0	0.0%	5h 43min 55s
s	8/255	430	0	426	4	0	0.0%	25min 16s
m	1/255	430	95	79	256	0	22.1%	22h 20min 14s
m	2/255	430	2	183	245	0	0.5%	20h 49min
m	4/255	430	0	326	103	1	0.0%	8h 48min 6s
m	8/255	430	0	415	15	0	0.0%	1h 20min 45s
l	1/255	430	14	80	336	0	3.3%	1d 4h 34min 10s
l	2/255	430	0	188	242	0	0.0%	20h 30min 46s
l	4/255	430	0	333	97	0	0.0%	8h 16min 22s
l	8/255	430	0	416	14	0	0.0%	1h 16min 12s
Adversarial Training								
xs	1/255	430	112	318	0	0	26.0%	8min 38s
xs	2/255	430	95	330	5	0	22.1%	35min 24s
xs	4/255	430	49	344	37	0	11.4%	3h 19min 43s
xs	8/255	430	8	381	41	0	1.9%	3h 36min 10s
s	1/255	430	242	184	4	0	56.3%	38min 12s
s	2/255	430	177	204	49	0	41.2%	4h 41min 3s
s	4/255	430	72	249	109	0	16.7%	9h 43min 24s
s	8/255	430	0	319	111	0	0.0%	9h 25min 13s
m	1/255	430	301	105	24	0	70.0%	2h 38min 1s
m	2/255	430	190	127	113	0	44.2%	10h 23min 10s
m	4/255	430	25	173	232	0	5.8%	19h 59min 37s
m	8/255	430	0	257	173	0	0.0%	14h 38min 13s
l	1/255	430	248	88	94	0	57.7%	8h 57min 21s
l	2/255	430	60	115	251	4	14.0%	21h 36min 40s
l	4/255	430	0	149	279	2	0.0%	23h 45min 56s
l	8/255	430	0	246	183	1	0.0%	15h 33min 44s

Table 4
Verification outcomes per training method, model size, and perturbation budget. Verif. rate = Verified / N ; Total time = wall-clock sum over all items.

Resource Utilization Statistics

Model	ε	Peak RSS (MB)	Peak GPU mem (MB)	Avg GPU util
Baseline Training				
xs	1/255	20010	32674	23.5%
xs	2/255	23212	30548	16.2%
xs	4/255	21684	32741	55.9%
xs	8/255	1549	27635	33.4%
s	1/255	27411	31754	41.5%
s	2/255	29989	31524	32.0%
s	4/255	29467	32593	71.8%
s	8/255	29918	19633	3.2%
m	1/255	29413	32731	62.9%
m	2/255	31995	32609	46.5%
m	4/255	34193	32731	21.3%
m	8/255	31087	22063	5.4%
l	1/255	27373	32761	62.3%
l	2/255	29808	32761	68.1%
l	4/255	27283	32752	19.6%
l	8/255	29173	27635	5.6%
Adversarial Training				
xs	1/255	1920	20383	39.1%
xs	2/255	9680	27635	50.2%
xs	4/255	15379	23531	9.1%
xs	8/255	23444	25602	8.4%
s	1/255	6851	32418	19.1%
s	2/255	20611	32487	58.8%
s	4/255	24773	32487	23.9%
s	8/255	32767	23316	19.9%
m	1/255	16213	30642	38.4%
m	2/255	23486	32567	69.3%
m	4/255	29716	32729	68.0%
m	8/255	35024	32729	33.0%
l	1/255	21448	32374	46.2%
l	2/255	25832	32567	52.3%
l	4/255	29827	32730	51.9%
l	8/255	34228	32755	67.8%

Table 5
Resource usage by training method, model size, and perturbation budget. All values are per-item maxima (or means for GPU utilization) aggregated over all items. All experiments were run on a server equipped with an Intel Xeon Silver 4215 CPU @ 2.50 GHz (8 Cores), an NVIDIA Tesla V100 GPU with 32 GB memory, and 128 GB of system RAM.