Formal Verification of Neural Network Safety Beyond Toy Examples:

Three Distinct Gaps and a Specification-First Research Agenda

Authors

Lead Researcher: Pranay M Mahendrakar

Mail: pranaymahendrakar2001@gmail.com

Contact: +91 6361723454

Company: SONYTECH

Date: April 2026

Abstract

Formal verification of neural networks has matured into a real research area. Tools such as alpha-beta-CROWN, ERAN, and Marabou now reliably verify L-infinity robustness properties of ReLU networks at the ResNet scale, with the annual VNN-COMP competition documenting steady year-over-year progress. The dominant framing of what remains undone — "scale verification to bigger models" — captures only one of three distinct gaps that separate current capability from useful guarantees on frontier AI systems. This paper makes three claims. First, the scale gap (verifiers handle networks with millions but not billions of parameters), the architecture gap (standard techniques handle ReLU well but degrade sharply on transformers with softmax, attention, and layer normalization), and the specification gap (we lack formal definitions of "safe LLM output" comparable to L-infinity robustness for image classifiers) are independent obstacles that need independent attention. Second, the specification gap is the most underweighted of the three and may be the binding constraint: even with verifiers that scaled to trillion-parameter models, we would not have formal specifications of harmlessness, honesty, or non-deception to verify against. Third, the most productive near-term frontier is not direct verification of LLM weights but verification of the systems that contain LLMs — LLM-as-policy verification, runtime monitors, agent-action guardrails. We propose a research agenda that reorients around the specification problem and the system-level frontier rather than continuing to pursue parameter-count scaling as the central goal.

Keywords

Formal verification, neural network verification, LLM safety, alpha-beta-CROWN, robustness certification, runtime monitoring, formal specifications

1. Introduction

Formal verification of neural networks is, for a research area sometimes dismissed as marginal, in considerably better shape than its critics allow. Tools developed over the past eight years — Reluplex (Katz et al., 2017), Marabou (Katz et al., 2019), ERAN (Singh et al., 2019), and the alpha-beta-CROWN family (Wang et al., 2021; Zhang et al., 2022) — now provide complete and sound verification for L-infinity robustness properties on ReLU networks at the ResNet scale. The annual International Verification of Neural Networks Competition (VNN-COMP), held since 2020, has documented steady year-over-year progress, with alpha-beta-CROWN winning each iteration from 2021 through 2025. This is not a paper field; it is producing usable artefacts.

It is also the case that no current technique can formally verify safety properties of frontier large language models. The framing usually offered for this gap is that verification needs to scale: more parameters, more layers, more compute. We argue the framing is incomplete. The distance between what current verifiers can do and what useful frontier-AI guarantees would require involves three distinct gaps, only one of which is about scale. The architecture gap (transformers, with softmax and attention, are structurally harder to verify than ReLU networks) and the specification gap (we lack formal definitions of "harmlessness" or "honesty" that would be amenable to verification even on a model verifiers could reach) are at least as important, and arguably more binding.

This paper makes three claims. First, the three gaps — scale, architecture, specification — are independent and require independent research strategies; conflating them under "verification needs to scale" obscures the most important one. Second, the specification gap is the most underweighted and may be permanent: even with a

hypothetical verifier capable of analysing trillion-parameter transformers, we would not have formal specifications of LLM safety properties to feed it. Third, the most productive near-term work is not direct verification of LLM weights but verification of the systems that contain LLMs — verifying that an LLM-driven agent never executes certain actions, verifying that a runtime monitor catches certain output classes, verifying that a synthesised behavioural policy satisfies stated constraints — areas where recent work (Gross et al., 2025; VeriGuard, 2025) has begun to make concrete progress.

Section 2 reviews where neural network verification actually stands. Section 3 develops the three-gap distinction. Section 4 argues that the specification gap is central. Section 5 identifies three productive frontiers that do not depend on scaling traditional verification to LLM weights. Section 6 proposes a research agenda. The contribution is structural — a clearer framing of what is solved, what is open, and what may be permanently out of reach — rather than new empirical results or new verifiers.

2. Where Neural Network Verification Stands

2.1 What Modern Verifiers Actually Verify

Modern verifiers operate, almost without exception, on a specific class of properties: bounded-input perturbation properties on classifier networks. The canonical formulation: given a classifier f, an input x, a perturbation budget epsilon, and a class label y, verify that for all x' with the L-infinity distance between x and x' at most epsilon, f(x') predicts y. This is L-infinity robustness, and it is the workhorse property for which verification techniques are most mature. The same techniques also handle related properties: linear specifications on outputs, monotonicity, basic safety properties for low-dimensional control systems such as the canonical ACAS Xu collision-avoidance benchmark.

alpha-beta-CROWN, the consistent VNN-COMP winner, supports a substantial range of architectures: fully connected networks, convolutional networks, residual connections, average and max pooling, several non-ReLU activations (sigmoid, tanh, gelu), and explicitly claims transformer support including self-attention. The bound-propagation engine (auto_LiRPA) underlying it is genuinely general. The state of the art is more capable than the field's critics often acknowledge.

2.2 What Modern Verifiers Cannot Verify

Three categories of limitation are worth being explicit about. First, network size. The largest networks routinely verified in VNN-COMP are in the millions-of-parameters range — image classifiers up to ResNet-class. Verifying a billion-parameter transformer for any non-trivial property is, on current techniques, computationally infeasible. The exponential blow-up in case analysis as networks deepen is not a bug to be tuned away; it is a property of the problem.

Second, property class. Almost all verified properties are local: bounds on output behaviour for inputs near a fixed point. Global properties — "the network never outputs harmful content over any input" — are not handled by current techniques and would, in general, be undecidable for any sufficiently expressive specification.

Third, specification. Even where size and property class are favourable, writing down what to verify is itself non-trivial. For image classifiers, L-infinity robustness has a clean formal statement that everyone agrees on. For LLMs, what does it mean to verify "safety"? The community does not have a consensus answer, and section 4 argues the absence is not accidental.

3. Three Distinct Gaps

3.1 The Scale Gap

The scale gap is the one most often cited and the easiest to characterise. Verification techniques based on bound propagation, abstract interpretation, or SMT solving all face exponential or near-exponential complexity in network depth and width. Considerable algorithmic progress has been made — the bound-tightening techniques in alpha-beta-CROWN are substantially more efficient than naive interval propagation — but the underlying complexity class has not changed. Verifying a billion-parameter network for L-infinity robustness at non-trivial epsilon is not a matter of waiting for Moore's law; it is plausibly intractable for current techniques regardless of compute.

Scale-gap research is real and useful: better bound propagation, GPU acceleration, branch-and-bound improvements, learned heuristics for case-splitting. The honest position is that this work pushes the frontier upward at a polynomial rate while the targets — frontier LLMs — grow at a faster rate. The gap may close in some absolute sense while widening relative to deployment-relevant model sizes.

3.2 The Architecture Gap

Transformers were designed to be easy to optimise, not easy to verify. Softmax produces a non-piecewise-linear, non-monotonic activation pattern that bound propagation handles less efficiently than ReLU. Attention introduces a quadratic dependency between tokens that is structurally different from the layer-by-layer composition standard

verifiers exploit. Layer normalisation introduces another global coupling. Each of these adds either tightness loss in bound propagation or new sources of branching that compound the scale gap.

Recent verifier development has worked on this. alpha-beta-CROWN claims transformer support; this is real but should be read carefully — "supports the architecture" is not the same as "verifies properties of frontier-scale instances of the architecture in tractable time." The combination of architecture and scale produces multiplicative difficulty, not additive. A researcher who solves the architecture gap on small transformers has not solved the combined problem.

3.3 The Specification Gap

Even granting hypothetical solutions to the previous two gaps — verifiers that handle billion-parameter transformers — the question of what to verify remains open. For an image classifier, L-infinity robustness is a meaningful and crisp property. For an LLM, what is the analogue? "Never produce harmful content" requires defining harmful content, which is not a property of the model output alone but of the output's relationship to a context, a recipient, and a normative framework. "Always answer truthfully" requires defining truth in a way the model and the verifier can agree on. "Never deceive the user" requires a formal notion of deception that the philosophical literature has wrestled with for centuries.

This is not a problem that more research on bound propagation will solve. The specification gap is a problem of formalising informal concepts well enough that an automated tool can check compliance. Where similar problems have been solved in other domains — protocol verification, hardware verification — it is because the specifications were themselves formal artefacts (cryptographic protocols, hardware semantics) and the gap was bridged by skilled formal-methods practitioners writing specifications by hand. The analogous practice for LLM safety properties would require formal-methods experts and AI-safety experts collaborating to produce specifications, and the practice barely exists.

4. The Specification Gap Is Central

4.1 What Crisp Specifications Buy You

Where the specification is crisp, formal verification is genuinely useful. L-infinity robustness on safety-critical image classifiers (medical imaging, autonomous driving perception) is well-defined and verifiable; useful guarantees can be obtained. ACAS Xu collision-avoidance properties are specified as conditions on the controller's output given specific aircraft-state regions; these are routinely verified. Monotonicity properties on credit-scoring models are crisp and verifiable. The pattern: when the specification can be written down without ambiguity, the verification machinery — within scale and architecture limits — delivers what it promises.

4.2 What Vague Specifications Cost You

Where the specification is vague, formal verification cannot help no matter what verifier is used. "The model should not produce harmful outputs" is not a specification in any formal sense; it is a research programme of its own. The same applies to most informally stated LLM safety properties. The current empirical safety toolkit — red-teaming, adversarial testing, alignment evaluations — is not a substitute for formal verification, but it is the appropriate response to specifications that are inherently informal, because empirical methods can probe behaviour without requiring formal definitions of what constitutes a violation.

4.3 Recent Attempts to Bridge

Several research lines have attempted to use LLMs themselves to bridge the specification gap. SpecVerify (Liu et al., 2025) uses LLMs to generate formal specifications from natural-language requirements, then formally verifies the resulting specifications. The Lahiri (2024) line of work on LLM-driven user-intent formalization for verification-aware languages such as Dafny and F* operates in a similar spirit. AutoSVA's GPT-4-based extension (Orenes-Vera et al., 2023) uses LLMs to generate SystemVerilog Assertions for hardware verification. Each of these takes informal user intent and produces formal specifications that traditional verifiers can act on.

These efforts have a limitation worth being explicit about. They reduce the specification gap from "write the formal specification by hand" to "verify that the LLM correctly translated user intent into formal specification." The latter is itself a verification problem, and the LLM-generated specifications are subject to hallucination and partial misunderstanding of the user's intent. The pattern is closer to specification-assistance than specification-elimination. It is useful, but it does not dissolve the specification gap; it relocates it.

5. Three Productive Frontiers That Do Not Require Scaling Verification of LLM Weights

5.1 Verify the Agent, Not the Model

LLMs are increasingly deployed as components of larger agentic systems: a model selects actions, those actions are executed in an environment, and the system makes consequential decisions over time. This system view enables

a different verification target: verify properties of the action sequence the LLM-driven agent produces, regardless of the LLM's internal computations. Gross, Spieker, and Gotlieb (2025) demonstrated this concretely for memoryless sequential decision-making, encoding the LLM as an opaque policy in a Markov decision process and applying probabilistic computation tree logic (PCTL) verification via the Storm model checker to derive formal safety guarantees for the resulting action sequences.

This is a substantively different problem from verifying the LLM weights. The LLM is treated as a black box that maps states to actions; verification operates over the abstracted policy. This sacrifices some expressive power — properties that depend on the LLM's internal reasoning are not addressable — but gains tractability and engages with a more useful question for many deployment contexts: not "is the LLM safe in some abstract sense" but "will the agent driven by this LLM ever execute the following bad action."

5.2 Runtime Monitors with Formal Guarantees

Where ahead-of-time verification of an LLM is infeasible, runtime monitoring of LLM outputs can provide formal guarantees over the deployed system. VeriGuard (2025) develops this approach for LLM agents specifically: the system synthesises a behavioural policy and formally verifies it against safety specifications, then the verified policy serves as a runtime monitor that validates each proposed agent action before execution. Watchdogs and Oracles (Liu et al., 2025) develops a similar architectural pattern, integrating runtime verification monitors with LLM-driven systems where the LLM is the subject of monitoring rather than the verifier.

The trade-off is clear: runtime monitors provide guarantees only for the properties the monitor was designed to check, not for the LLM's behaviour across all possible inputs. They do not catch the kinds of failures that the monitor's specification did not anticipate. This is a limitation, not a defect — the same trade-off applies to runtime monitoring of conventionally engineered software, where it has been productive for decades.

5.3 Probabilistic Verification as Middle Ground

Strict formal verification asks for guarantees that hold over all inputs in a specified set. Probabilistic verification asks for guarantees that hold with high probability over a specified input distribution. The latter is weaker but often more tractable and, for many AI safety questions, more honest about the kind of guarantee that is actually achievable. Statistical verification approaches — randomised smoothing, certified probabilistic robustness, abstract probabilistic interpretation — sit in this middle ground. They give weaker guarantees than full formal verification but extend further into the deployment-relevant regime.

6. A Research Agenda

6.1 Methodological Commitments

Three commitments would help the field. First, papers on neural network verification should explicitly identify which of the three gaps they address. A paper that improves bound propagation efficiency addresses the scale gap; one that handles softmax better addresses the architecture gap; one that proposes a formalisation of an LLM safety property addresses the specification gap. The current habit of framing all verification work as "making verification scale" obscures which problem is actually being solved.

Second, claims of "safety verification" for AI systems should report the specification used. "We verified that the model is safe" is not a publishable claim; "we verified that the model satisfies property phi where phi is [precise formal statement]" is. The pattern of using the word "safety" as a placeholder for unspecified properties has done genuine damage to the field's epistemic standards.

Third, the distinction between verifying the model and verifying the system should be maintained. Both are valuable; conflating them when reporting results misleads readers about what guarantee has been obtained.

6.2 Specific Empirical Questions

1. On VNN-COMP-style benchmarks restricted to transformer architectures, what is the largest model size at which alpha-beta-CROWN currently produces a sound verification result for non-trivial L-infinity robustness in tractable time? Tracking this number annually would clarify whether the architecture-and-scale combined frontier is moving.
2. For LLM-as-policy verification (Gross-style), what is the practical limit on state-space size before model checking becomes infeasible, and how does that limit interact with prompt complexity?
3. In specification-assistance pipelines (LLM generates spec, traditional verifier checks), what fraction of generated specifications are correct on first attempt, and what kinds of errors dominate the failures?
4. For runtime-monitor approaches like VeriGuard, what is the empirical false-positive and false-negative rate of monitor-detected violations against human-evaluated ground truth on real LLM outputs?
5. Across published "verified safe" claims for AI systems in the past two years, how many include a precise

formal specification of the verified property, and how many use "safety" as an informal placeholder?

6.3 The Specification-First Reorientation

The strongest contribution the broader research community could make is sustained effort on the specification problem. This is not glamorous work — it requires close collaboration between formal-methods practitioners, AI-safety researchers, and domain experts in the application area, and it produces specifications rather than algorithms. But the specifications, if they are good, would unlock the verification work that is currently waiting on having something to verify. The current pattern of building more powerful verifiers without commensurate investment in what to verify produces tools that are increasingly capable of answering questions nobody has formulated.

7. Conclusion

Formal verification of neural networks is a more mature field than its critics acknowledge and a less complete field than its proponents sometimes suggest. The state of the art reliably verifies meaningful properties on networks at the millions-of-parameters scale, has tools that nominally extend to transformers, and has begun to verify properties of LLM-driven systems even where verifying the LLMs themselves remains out of reach. The path forward is not principally about making verifiers handle larger networks. It is about formalising what we want to verify, identifying the system-level abstractions where verification is tractable, and being explicit about which of the three gaps any particular research contribution addresses.

The specification problem is the binding constraint and the most underweighted research direction. It is also the one that most resists the standard machine-learning research playbook, because progress on it is conceptual and collaborative rather than algorithmic and benchmarkable. The community's standard incentives push against the work that is most needed. Acknowledging this honestly is a precondition for changing it.

References

Bak, S., Liu, C., & Johnson, T. (2021). The second international verification of neural networks competition (VNN-COMP 2021): Summary and results. arXiv:2109.00498.

Brix, C., et al. (2024). The fourth international verification of neural networks competition (VNN-COMP 2023): Summary and results. arXiv:2312.16760.

Gross, D., Spieker, H., & Gotlieb, A. (2025). Verifying memoryless sequential decision-making of large language models. arXiv:2510.06756.

Katz, G., Barrett, C., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. International Conference on Computer Aided Verification (CAV).

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., et al. (2019). The Marabou framework for verification and analysis of deep neural networks. CAV 2019.

Lahiri, S. K. (2024). Evaluating LLM-driven user-intent formalization for verification-aware languages. arXiv:2406.09757.

Liu, A., et al. (2025). Watchdogs and oracles: Runtime verification meets large language models for autonomous systems. arXiv:2511.14435.

Liu, X., et al. (2025). Supporting software formal verification with large language models: An experimental study (SpecVerify). arXiv:2507.04857.

Orenes-Vera, M., Manocha, A., Wentzlaff, D., & Martonosi, M. (2023). Using LLMs to facilitate formal verification of RTL. arXiv:2309.09437.

Singh, G., Gehr, T., Püschel, M., & Vechev, M. (2019). An abstract domain for certifying neural networks. POPL 2019, ACM.

Tjeng, V., Xiao, K., & Tedrake, R. (2019). Evaluating robustness of neural networks with mixed integer programming. International Conference on Learning Representations (ICLR).

VeriGuard authors. (2025). VeriGuard: Enhancing LLM agent safety via verified code generation. arXiv:2510.05156.

Wang, S., Pei, K., Whitehouse, J., Yang, J., & Jana, S. (2018). Formal security analysis of neural networks using symbolic intervals. USENIX Security.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., & Kolter, J. Z. (2021). Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. NeurIPS.

Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.-W., Huang, M., Kailkhura, B., Lin, X., & Hsieh, C.-J. (2020). Automatic perturbation analysis for scalable certified robustness and beyond. NeurIPS.

Zhang, H., Wang, S., Xu, K., Li, L., Li, B., Jana, S., Hsieh, C.-J., & Kolter, J. Z. (2022). General cutting planes for bound-propagation-based neural network verification. NeurIPS. (alpha-beta-CROWN)