Toward Continuous Verification of DNNs

Shubham Ugare [1] **Debangshu Banerjee** [1] **Tarun Suresh** [1] **Sasa Misailovic** [1] **Gagandeep Singh** [1 2]

Abstract

Deep neural network (DNN) verification is used to assess whether a DNN meets a desired trustworthy property (e.g., robustness, fairness) on an infinite input set. While there have been significant advancements in scalable verifiers for individual DNNs, they suffer from inefficiency when developers update a deployed DNN (e.g. through quantization, pruning, or fine-tuning) to enhance its inference speed or accuracy. The verification is inefficient because the developers need to re-run the computationally expensive verifier from scratch on the updated DNN. To address this issue, we propose an incremental DNN verification framework, leveraging novel theory, data structures, and algorithms. Our previous works Ugare et al. (2023a) and Ugare et al. (2022) focused on incremental verification, and in this paper, we provide a summary of these advancements and explore the potential of incremental verification in enabling real-world DNN verification systems.

1. Introduction

Deep neural networks (DNNs) are increasingly being used in safety-critical applications, such as autonomous driving (Bojarski et al., 2016), healthcare (Amato et al., 2013), and aviation (Julian et al., 2018). However, DNNs are vulnerable to adversarial examples, which are inputs that have been slightly modified to cause the DNN to make incorrect predictions (Szegedy et al., 2014; Madry et al., 2017). This vulnerability has raised concerns about the trustworthiness of DNNs in safety-critical applications.

To address these concerns, researchers have developed a number of methods for verifying the trustworthiness of DNNs (see (Urban & Miné, 2021; Albarghouthi, 2021) for a survey). Verification methods use formal proofs to show that DNNs will behave correctly on a set of inputs. This is in contrast to standard test-set accuracy measurements, which only check DNN performance on a finite set of inputs.

Verification methods can be broadly classified as either complete or incomplete. Complete methods are guaranteed to provide an exact answer for the verification task, while incomplete methods may fail to prove or disprove a trustworthiness property (Gehr et al., 2018a; Singh et al., 2018b; 2019b;a; Zhang et al., 2018; Xu et al., 2020; Salman et al., 2019). Complete methods are more desirable, but they are also more computationally expensive (Bunel et al., 2020a;b; Bak et al., 2020; Ehlers, 2017; Ferrari et al., 2022; Fromherz et al., 2021; Wang et al., 2021; Palma et al., 2021).

Problem of Existing Work: Deploying DNNs on real-world systems has opened the question of optimizing the computational cost of inference: to reduce this cost, researchers have devised various techniques for *approximating* DNNs, which simplify the structure of the network (typically post-training), while maintaining high accuracy and robustness. Common approximation techniques include quantization – reducing numerical precision of the weights (Gholami et al., 2021; Laurel et al., 2021); pruning – removing weights (Frankle & Carbin, 2019); operator approximation – e.g. approximating convolutions (Figurnov et al., 2016; Sharif et al., 2021)); and others.

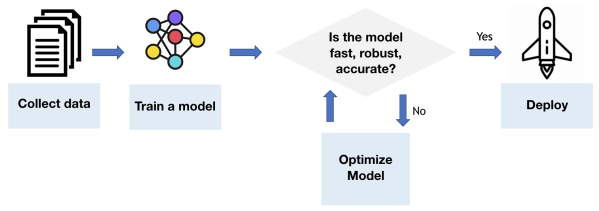

Figure 1. A common deployment cycle for a deep neural network.

Figure 1 illustrates the DNN deployment process, which iteratively optimizes the network. Checking whether the optimized networks are robust is an important step in this process, but currently developers mostly rely on empirical testing due to the high cost of verification techniques. A major limitation of almost all existing approaches for verifying deep neural networks is that the verifier needs to be run from scratch end-to-end every time the network is even slightly modified. Running precise verification from

[1]Department of Computer Science, University of Illinois, Urbana-Champaign, USA [2]VMware Research, Palo Alto, USA. Correspondence to: Shubham Ugare <sugare2@illinois.edu>.

2nd *Workshop on Formal Verification of Machine Learning*, Honolulu, Hawaii, USA. Colocated with ICML 2023. Copyright 2023 by the author(s).

scratch is an expensive operation and cannot keep up with the rate at which the networks are modified during deployment. Overcoming this main limitation requires addressing a fundamental problem in verifier design:

Can we reuse the proofs obtained when verifying a given network to speed up the verification of its perturbed versions?

Incremental Verification: Formal methods research has developed numerous techniques for incremental verification of programs, that reuse the proof from previous revisions for verifying the new revision of the program (Johnson et al., 2013; Lakhnech et al., 2001; O'Hearn, 2018; Stein et al., 2021). The main challenge in building an incremental verifier on top of a non-incremental one is to determine which information to pass on and how to effectively reuse this information. Often the classical programs' commits are local changes that affect only a small part of the big program. In contrast, most DNN updates result in weight perturbation across one or many layers of the network. This poses a different and more difficult challenge than incremental program verification.

Next, we summarize our work on incremental complete (Section 2) and incomplete (Section 3) verification of DNNs.

2. Incremental Complete Verification

In our recent work (Ugare et al., 2023a), we address the fundamental limitation of existing complete verifiers by presenting IVAN, the first general technique for incremental and complete verification of DNNs. An original network and its updated network have similar behaviors on most of the inputs, therefore the proofs of property on these networks are also related. IVAN accelerates the complete verification of a trustworthy property on the updated network by leveraging the proof of the same property on the original network. IVAN can be built on top of any Branch and Bound (BaB) based method. The BaB verifier recursively partitions the verification problem to gain precision. It is currently the dominant technology for constructing complete verifiers (Wang et al., 2018; Bunel et al., 2020a;b; Bak et al., 2020; Ehlers, 2017; Ferrari et al., 2022; Fromherz et al., 2021; Wang et al., 2021; Palma et al., 2021; Anderson et al., 2019).

Our Solution: DNN complete verifiers employ distinct heuristics for branching. A key challenge is to develop a generic method that incrementally verifies a network perturbed across multiple layers and is applicable to multiple complete verification methods, yet can provide significant performance benefits.

IVAN computes a specification tree – a novel tree data structure representing the trace of BaB – from the execution of the complete verifier on the original network. We design new algorithms to refine the specification tree to create a more compact tree. At a high level, the refinement involves reordering the branching decisions such that the decisions that worked well in the original verification are prioritized. Besides, it removes the branching decisions that worked poorly in the original verification by pruning nodes and edges in the specification tree. IVAN also improves the branching strategy in BaB for the updated network based on the observed effectiveness of branching choices when verifying the original DNN. The compact specification tree and the improved branching strategy guide the BaB execution on the updated network to faster verification, compared to non-incremental verification that starts from scratch.

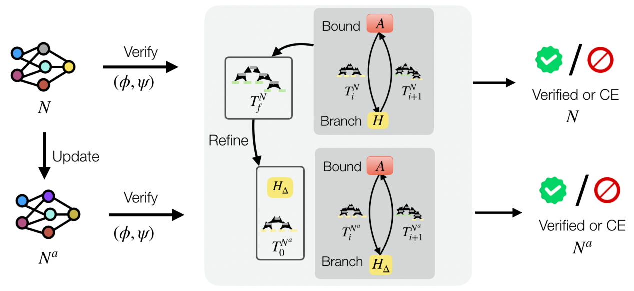

Figure 2. Workflow of IVAN from left to right. $IVAN$ takes the original network N, input specification ϕ and output specification ψ. It is built on top of a BaB-based complete verifier that utilizes an analyzer A for the bounding, and heuristic H for branching. IVAN refines a specification tree T_f^N, result of verifying N, to create a compact tree $T_0^{N^a}$ and updated branching heuristic H_Δ. IVAN performs faster verification of N^a exploiting both $T_0^{N^a}$ and H_Δ.

Workflow: Figure 2 illustrates the high-level idea behind the workings of IVAN. It takes as input the original neural network N, the updated network N^a, a local or global input region ϕ, and the output property ψ. The goal of IVAN is to check whether, for all inputs in ϕ, the outputs of networks N and N^a satisfy ψ. N and N^a have similar behaviors on the inputs in ϕ, therefore the proofs of the property on these networks are also related. IVAN accelerates the complete verification of the property (ϕ, ψ) on N^a by leveraging the proof of the same property on N.

Results: IVAN yields up to 43x speedup over the baseline based on state-of-the-art non-incremental verification techniques (Henriksen & Lomuscio, 2021; Bunel et al., 2020a; Singh et al., 2018b). It achieves a geometric mean speedup of 2.4x across challenging fully-connected and convolutional networks over the baseline. IVAN is generic and can work with various common BaB branching strategies in the literature (input splitting, ReLU splitting). Appendix A.1 describes the methodology for the evaluation.

Table 1. Overall IVAN speedups across all properties for various quantizations.

Model	Approximation	IVAN *Sp*	IVAN *+Solved*
FCN-MNIST	int16	4.43x	0
	int8	2.02x	0
CONV-MNIST	int16	3.09x	2
	int8	1.71x	4
CONV-CIFAR	int16	2.52x	2
	int8	1.78x	0
CONV-CIFAR-WIDE	int16	1.87x	2
	int8	1.53x	2
CONV-CIFAR-DEEP	int16	3.21x	0
	int8	1.25x	1

Ugare et al. (2023a) evaluate IVAN on multiple approximations and BaB baselines. Here, we summarize part of the evaluation that uses quantization. Table 1 presents the comparison of the technique used in IVAN for each mode. Column *+Solved* displays the number of extra verification problems solved by the technique in comparison to the baseline. Columns in IVAN present the results on using IVAN. Column Sp demonstrates the overall speedup obtained compared to the baseline.

3. Incremental Sound and Incomplete Verification

This Work: Our recent paper (Ugare et al., 2022) presents FANC, the first general approach for transferring proofs between a given network and its multiple approximate versions. Our approach is generic and can be combined with any existing state-of-the-art *Split*incomplete verifier (Gehr et al., 2018b; Katz et al., 2017; 2019; Lu & Kumar, 2020; Salman et al., 2019; Singh et al., 2018a; 2019c) to improve the speed of verifying the approximate versions of the given network with fully-connected and convolutional layers and various activation functions. FANC guarantees to be as precise as the chosen verifier.

FANC first generates a set of *templates* – connected symbolic shapes (e.g., boxes, polyhedra) at an intermediate layer – by running a verifier through the original network. The templates capture the proof of the property to be verified on the original network. Next, FANC transfers these templates to the approximate networks by incremental template modification so that they capture the proof on the approximate versions. Finally, it runs the verifier on the approximate networks but only until the layer where the template is employed. If the generated templates capture the intermediate proof, then we have proved the property without running the verifier end-to-end.

Workflow: Figure 3 illustrates the high-level idea of FANC.

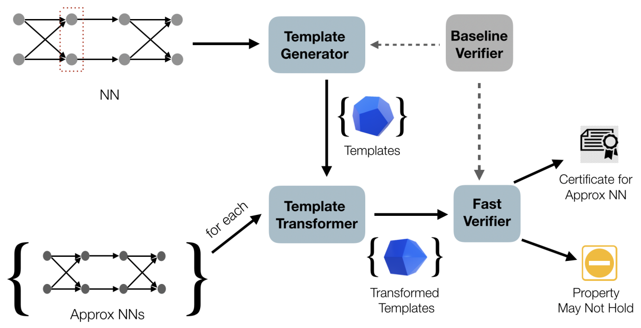

Figure 3. Workflow of FANC from left to right. FANC consists of three components: template generator, template transformer, and fast verifier. First, the template generator takes the network N as input and creates a set of templates. For each approximate network, the template transformer transforms the templates. This transformed template is used by our fast verifier to verify the approximate network. The fast verifier either successfully verifies the network and generates a certificate or reports that the property may not hold.

It takes as input a neural network N, its approximate version N^{app}, a set of input regions ϕ as a precondition, and the output property ψ as the postcondition. The neural network verification problem corresponding to the property (ϕ,ψ) involves proving that for all network inputs in ϕ, the corresponding network output satisfies the postcondition ψ. The goal of FANC is to speed up the proof of the property (ϕ, ψ) on the approximate network by leveraging intermediate proofs obtained by verifying N. FANC proves the same number of properties as a vanilla baseline verifier (in our case DeepZ) without templates but significantly faster. We get up to 4.1x speedup on robustness certification tasks.

Results: We evaluate the effectiveness of FANC by verifying robustness of challenging fully-connected and convolutional networks approximated with quantization and pruning against different attacks. We considered four network architectures, robustly trained on the popular MNIST and CIFAR10 datasets. We verified the robustness of the networks quantized using float16, int16, and int8 strategies against five different adversarial attacks: adversarial patches, L_0-random, L_0-center, rotation and brightening. We used the state-of-the-art DeepZ (Singh et al., 2018a) as the baseline verifier. FANC has significantly improved verification time for the quantized networks, by up to 4.1x, with a median speedup of 1.55x over DeepZ. We verified the robustness of the networks with 10-90% pruning rates against the adversarial patch attack. FANC has improved verification time up to 2.8x, with a median speedup of 1.48x over DeepZ. Appendix A.2 describes the methodology for the evaluation.

Table 2. Average FANC speedups for verification for L_0-random, L_0-center and patch attacks.

Model	Approximation	L_0-random	L_0-center	patch
FCN7-MNIST	float16	3.03	2.63	2.25
	int16	3.28	2.23	2.24
	int8	3.93	1.57	2.63
CONV2-MNIST	float16	1.72	1.53	1.41
	int16	1.72	1.52	1.4
	int8	1.61	1.43	1.34
FCN7-CIFAR	float16	1.5	1.51	1.64
	int16	1.75	1.75	1.47
	int8	1	1.14	1.13
CONV4-CIFAR	float16	1.37	1.28	1.41
	int16	1.39	1.25	1.39
	int8	1.19	1.26	1.28

Ugare et al. (2022) evaluate FANC on pruning and qunatization. Here, we summarize part of the evaluation that uses quantization. Table 2 summarizes the verification results for different quantization approximations. Each column presents the speedup obtained by FANC's core verification with proof transfer compared to the baseline (without the template creation). In each case, the time in the table is averaged over all the images used in the experiment.

4. Related Work

Incremental Program Verification. Incremental verification has significantly improved the scalability of the traditional program verification (Johnson et al., 2013; Lakhnech et al., 2001; O'Hearn, 2018; Stein et al., 2021). Incremental program analysis tasks achieve faster analysis of individual commits by reusing partial results (Yang et al., 2009), constraints (Visser et al., 2012), and precision information (Beyer et al., 2013) from previous runs. However, often the program commits are local changes that affect only a small part of the big program. In contrast, most DNN updates result in weight perturbation across one or many layers of the network. This poses a different and more difficult challenge than incremental program verification. Thus, new algorithms and tools are required for incremental DNN verification.

Incremental DNN Verification. Several methods have been introduced in recent years to certify the properties of DNNs deterministically (Tjeng et al., 2017; Bunel et al., 2020a; Katz et al., 2017; Wang et al., 2021; Laurel et al., 2022) and probabilisticly (Cohen et al., 2019). More recently, few works used incremental verification to improve the scalability of DNN verification (Fischer et al., 2022; Ugare et al., 2022; Wei & Liu, 2021; Ugare et al., 2023a) – these works apply complete and incomplete formal verification.

5. Discussion and Future Work

More Aggressive Weight Perturbation. Both IVAN and FANC have limitations in reusing proofs under certain levels of perturbation. As the perturbation increases, the benefit decreases. There is a point where the approximate DNN is semantically different from the original DNN. Further research is needed to determine the precise class of DNNs for which proof reusability is feasible. This knowledge will facilitate the utilization of proofs for more aggressive perturbations, such as those encountered during subsequent training iterations.

Incremental Probabilistic Verification. The techniques presented in this paper provide incremental deterministic certification, but they lack scalability for high-dimensional inputs like ImageNet or state-of-the-art models such as ResNet. To support such models and datasets, it is important that future research develops incremental frameworks to enhance the speed of probabilistic methods like Randomized Smoothing. Our first steps toward this goal are presented in (Ugare et al., 2023b).

Proof Transfer Across Inputs. This work has demonstrated the possibility of proof reuse across networks. Previous work has demonstrated the feasibility of reusing proofs across specifications (Fischer et al., 2022). In a similar vein, future research may explore the potential for reusing proofs across similar inputs. For example, in the case of video input, where subsequent frames exhibit similarity, this approach is likely to be effective.

6. Conclusion

Our research introduces an incremental DNN verification framework to overcome the inefficiencies associated with verifying updated DNNs. By leveraging novel theories, data structures, and algorithms, our framework improves the efficiency of verification processes. We have presented the advancements made in incremental verification in our previous works, as summarized in this paper. The potential of incremental verification in enabling real-world DNN verification systems has been explored. This research opens up new avenues for efficient and practical verification of DNNs, facilitating their reliable deployment in various domains.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments. This research was supported in part by NSF Grants No. CCF-1846354, CCF-1956374, CCF-2008883, CCF-2217144, CCF-2238079, CNS-2148583, USDA NIFA Grant No. NIFA-2024827 and Qualcomm innovation fellowship.

References

Albarghouthi, A. *Introduction to Neural Network Verification*. verifieddeeplearning.com, 2021. http://verifieddeeplearning.com.

Amato, F., López, A., Peña-Méndez, E. M., Vaňhara, P., Hampl, A., and Havel, J. Artificial neural networks in medical diagnosis. *Journal of Applied Biomedicine*, 11 (2):47–58, 2013.

Anderson, G., Pailoor, S., Dillig, I., and Chaudhuri, S. Optimization and abstraction: A synergistic approach for analyzing neural network robustness. In *Proc. Programming Language Design and Implementation (PLDI)*, 2019.

Bak, S., Tran, H., Hobbs, K., and Johnson, T. T. Improved geometric path enumeration for verifying relu neural networks. In Lahiri, S. K. and Wang, C. (eds.), *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I*, volume 12224 of *Lecture Notes in Computer Science*, pp. 66–96. Springer, 2020. doi: 10.1007/978-3-030-53288-8_4. URL https://doi.org/10.1007/978-3-030-53288-8_4.

Balunovic, M. and Vechev, M. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJxSDxrKDr.

Beyer, D., Löwe, S., Novikov, E., Stahlbauer, A., and Wendler, P. Precision reuse for efficient regression verification. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pp. 389–399, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450322379. doi: 10.1145/2491411.2491429. URL https://doi.org/10.1145/2491411.2491429.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Bunel, R., Lu, J., Turkaslan, I., Kohli, P., Torr, P., and Mudigonda, P. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(2020), 2020a.

Bunel, R. R., Hinder, O., Bhojanapalli, S., and Dvijotham, K. An efficient nonconvex reformulation of stagewise convex optimization problems. *Advances in Neural Information Processing Systems*, 33, 2020b.

Chiang, P.-y., Ni, R., Abdelkader, A., Zhu, C., Studor, C., and Goldstein, T. Certified defenses for adversarial patches. In *International Conference on Learning Representations*, 2020.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1310–1320. PMLR, 2019. URL http://proceedings.mlr.press/v97/cohen19c.html.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Ehlers, R. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, 2017.

Ferrari, C., Mueller, M. N., Jovanović, N., and Vechev, M. Complete verification via multi-neuron relaxation guided branch-and-bound. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=l_amHf1oaK.

Figurnov, M., Ibraimova, A., Vetrov, D. P., and Kohli, P. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems 2016*, pp. 947–955, 2016.

Fischer, M., Sprecher, C., Dimitrov, D. I., Singh, G., and Vechev, M. T. Shared certificates for neural network verification. In Shoham, S. and Vizel, Y. (eds.), *Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part I*, volume 13371 of *Lecture Notes in Computer Science*, pp. 127–148. Springer, 2022. doi: 10.1007/978-3-031-13185-1_7. URL https://doi.org/10.1007/978-3-031-13185-1_7.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proc. International Conference on Learning Representations (ICLR)*, 2019.

Fromherz, A., Leino, K., Fredrikson, M., Parno, B., and Pasareanu, C. Fast geometric projections for local robustness certification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zWy1uxjDdZJ.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, 2018a.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. T. AI2: safety and robustness certification of neural networks with abstract interpretation. In *IEEE S&P Symposium*, pp. 3–18, 2018b. doi: 10.1109/SP.2018.00058.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. A survey of quantization methods for efficient neural network inference. *CoRR*, abs/2103.13630, 2021. URL `https://arxiv.org/abs/2103.13630`.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2018.

Henriksen, P. and Lomuscio, A. Deepsplit: An efficient splitting method for neural network verification via indirect effect analysis. In Zhou, Z.-H. (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2549–2555. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/351. URL `https://doi.org/10.24963/ijcai.2021/351`. Main Track.

Johnson, K., Calinescu, R., and Kikuchi, S. An incremental verification framework for component-based software systems. In *Proceedings of the 16th International ACM Sigsoft Symposium on Component-Based Software Engineering*, CBSE '13, pp. 33–42, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321228. doi: 10.1145/2465449.2465456. URL `https://doi.org/10.1145/2465449.2465456`.

Julian, K. D., Kochenderfer, M. J., and Owen, M. P. Deep neural network compression for aircraft collision avoidance systems. *CoRR*, abs/1810.04240, 2018.

Julian, K. D., Kochenderfer, M. J., and Owen, M. P. Deep neural network compression for aircraft collision avoidance systems. *Journal of Guidance, Control, and Dynamics*, 42(3):598–608, mar 2019. doi: 10.2514/1.g003724. URL `https://doi.org/10.2514%2F1.g003724`.

Katz, G., Barrett, C. W., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification - 29th International Conference, CAV*, volume 10426, pp. 97–117, 2017. doi: 10.1007/978-3-319-63387-9_5.

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljic, A., Dill, D. L., Kochenderfer, M. J., and Barrett, C. W. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification, CAV*, volume 11561, pp. 443–452, 2019. doi: 10.1007/978-3-030-25540-4_26.

Lakhnech, Y., Bensalem, S., Berezin, S., and Owre, S. Incremental verification by abstraction. In Margaria, T. and Yi, W. (eds.), *Tools and Algorithms for the Construction and Analysis of Systems: 7th International Conference, TACAS 2001*, volume 2031, pp. 98–112, Genova, Italy, April 2001. Springer-Verlag.

Laurel, J., Yang, R., Sehgal, A., Ugare, S., and Misailovic, S. Statheros: Compiler for efficient low-precision probabilistic programming. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 787–792, 2021. doi: 10.1109/DAC18074.2021.9586276.

Laurel, J., Yang, R., Ugare, S., Nagel, R., Singh, G., and Misailovic, S. A general construction for abstract interpretation of higher-order automatic differentiation. *Proc. ACM Program. Lang.*, 6(OOPSLA2), oct 2022. doi: 10.1145/3563324. URL `https://doi.org/10.1145/3563324`.

Lu, J. and Kumar, M. P. Neural network branching for neural network verification. In *International Conference on Learning Representations*, 2020.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

O'Hearn, P. W. Continuous reasoning: Scaling the impact of formal methods. In Dawar, A. and Grädel, E. (eds.), *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pp. 13–25. ACM, 2018. doi: 10.1145/3209108.3209109. URL `https://doi.org/10.1145/3209108.3209109`.

Palma, A. D., Behl, H. S., Bunel, R. R., Torr, P. H. S., and Kumar, M. P. Scaling the convex barrier with active sets. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Paulsen, B., Wang, J., and Wang, C. Reludiff: differential verification of deep neural networks. In *ICSE '20: 42nd International Conference on Software Engineering*, 2020. doi: 10.1145/3377811.3380337.

Salman, H., Yang, G., Zhang, H., Hsieh, C., and Zhang, P. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems 32*, pp. 9832–9842, 2019.

Sharif, H., Zhao, Y., Kotsifakou, M., Kothari, A., Schreiber, B., Wang, E., Sarita, Y., Zhao, N., Joshi, K., Adve, V. S., Misailovic, S., and Adve, S. V. Approxtuner: a compiler and runtime system for adaptive approximations. In *PPoPP '21: 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 262–277, 2021. doi: 10.1145/3437801.3446108.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, volume 31, 2018a.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. Fast and effective robustness certification. *Advances in Neural Information Processing Systems*, 31, 2018b.

Singh, G., Ganvir, R., Püschel, M., and Vechev, M. Beyond the single neuron convex barrier for neural network certification. In *Advances in Neural Information Processing Systems*, 2019a.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL), 2019b.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. T. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL), 2019c. doi: 10.1145/3290354.

Singh, G. et al. Eran. https://github.com/eth-sri/eran, 2018c.

Stein, B., Chang, B. E., and Sridharan, M. Demanded abstract interpretation. In Freund, S. N. and Yahav, E. (eds.), *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, pp. 282–295. ACM, 2021. doi: 10.1145/3453483.3454044. URL https://doi.org/10.1145/3453483.3454044.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations*, 2014.

TFLite. Tf lite post-training quantization, 2017.

Tjeng, V., Xiao, K., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Ugare, S., Singh, G., and Misailovic, S. Proof transfer for fast certification of multiple approximate neural networks. *Proc. ACM Program. Lang.*, 6(OOPSLA):1–29, 2022. doi: 10.1145/3527319. URL https://doi.org/10.1145/3527319.

Ugare, S., Banerjee, D., Misailovic, S., and Singh, G. Incremental verification of neural networks. *Proc. ACM Program. Lang.*, 7(PLDI), jun 2023a. doi: 10.1145/3591299. URL https://doi.org/10.1145/3591299.

Ugare, S., Suresh, T., Banerjee, D., Singh, G., and Misailovic, S. Incremental randomized smoothing certification, 2023b.

Urban, C. and Miné, A. A review of formal methods applied to machine learning, 2021. URL https://arxiv.org/abs/2104.02466.

Visser, W., Geldenhuys, J., and Dwyer, M. B. Green: Reducing, reusing and recycling constraints in program analysis. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, FSE '12, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450316149. doi: 10.1145/2393596.2393665. URL https://doi.org/10.1145/2393596.2393665.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*, 2021.

Wei, T. and Liu, C. Online verification of deep neural networks under domain or weight shift. *CoRR*, abs/2106.12732, 2021. URL https://arxiv.org/abs/2106.12732.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 2018.

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

Yang, G., Dwyer, M. B., and Rothermel, G. Regression model checking. In *2009 IEEE International Conference on Software Maintenance*, pp. 115–124, 2009. doi: 10.1109/ICSM.2009.5306334.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, 2018.

A. Appendix

A.1. IVAN Evaluation Methodology

Table 3. Models and the perturbation ϵ used for the evaluation for incremental verification.

Model	Architecture	Dataset	#Neurons	Training Method	ϵ
ACAS-XU Networks	6×50 linear layers	ACAS-XU	300	Standard (Julian et al., 2019)	-
FCN-MNIST	2×256 linear layers	MNIST	512	Standard	0.02
CONV-MNIST	2 Conv, 2 linear layers	MNIST	9508	Certified Robust (Balunovic & Vechev, 2020)	0.1
CONV-CIFAR	2 Conv, 2 linear layers	CIFAR10	4852	Empirical Robust (Dong et al., 2018)	$\frac{2}{255}$
CONV-CIFAR-WIDE	2 Conv, 2 linear layers	CIFAR10	6244	Certified Robust (Wong & Kolter, 2018)	$\frac{4}{255}$
CONV-CIFAR-DEEP	4 Conv, 2 linear layers	CIFAR10	6756	Certified Robust (Wong & Kolter, 2018)	$\frac{4}{255}$

Networks and Properties. We evaluate IVAN on models with various architectures that are trained with different training methods. Similar to most of the previous literature, we verify L_∞-based local robustness properties for MNIST and CIFAR10 networks and choose standard ϵ values used for evaluating complete verifiers. Table 4 presents the evaluated models and the choice of ϵ for the local robustness properties.

Network Perturbation. Similar to previous works (Paulsen et al., 2020; Ugare et al., 2022), we use quantization to generate perturbed networks. Specifically, we use int8 and int16 post-training quantizations. The quantization scheme has the form (TFLite, 2017): $r = s(q - zp)$. Here, q is the quantized value and r is the real value; s which is the scale and zp which is the zero point are the parameters of quantization. Our experiments use symmetric quantization with $zp = 0$.

Baseline. For proving the local robustness properties, we use LP-based triangle relaxation for bounding (Ehlers, 2017; Bunel et al., 2020a), and we use the estimation based on coefficients of the zonotopes for choosing the ReLU splitting (Henriksen & Lomuscio, 2021).

Experimental Setup. We use 64 cores of an AMD Ryzen Threadripper CPU with the main memory of 128 GB running the Linux operating system. The code for our tool is written in Python. We use the GUROBI (Gurobi Optimization, LLC, 2018) solver for our LP-based analyzer.

A.2. FANC Evaluation Methodology

Networks: Table 4 presents the models used in evaluation. All the models are robustly trained using the training procedure from Chiang et al. (2020). The approximate network versions are generated using post-training quantization and pruning. In the case of CNN, only the weight parameters corresponding to the fully-connected layers are pruned. *Split*For non-robustly trained networks, even the verification on the original network fails most of the time (as we empirically observed) and approximating such networks is unlikely to lead to robust networks. Since the verification is unlikely to succeed, we did not use such networks in our evaluation.

Perturbations: We used the patch, two versions of L_0 attack (L_0-random and L_0-center):

- For the L_0-random attack, we first choose 3 as the threshold for perturbation. Proving robustness against all possible 3-pixel perturbation is currently impractical since it takes $\binom{729}{3}$ verification instance for one MNIST image. Hence, we perform verification against 1000 randomly sampled combinations of 3 pixels.
- For the L_0-center choose all the combinations of 3 pixels from the center of the picture since the center of the image contains an important part of the image in both MNIST and CIFAR10. We selected all $\binom{20}{3}$ set of 3 pixels from the central 5×4 patch in the image.
- For the patch attack, we verify the model against all perturbations in 2×2 patches.

Quantization: In this paper, we consider int8, int16 and float16 post-training quantizations. The quantization scheme is of the form (TFLite, 2017):

$$r = s(q - zp) \quad (1)$$

Here, q is the quantized value and r is the real value. s which is the scale and zp which is the zero point are the parameters of quantization. For our experiments, we use symmetric quantization that uses $zp = 0$. The quantization scheme uses a

Table 4. Models used for the evaluation.

Model	Architecture	Dataset	#Neurons
FCN7-MNIST	7×200 linear layers	MNIST	1400
FCN7-CIFAR	7×200 linear layers	CIFAR10	1400
CONV2-MNIST	2 Conv layers, 4 linear layers	MNIST	2456
CONV4-CIFAR	4 Conv layers, 4 linear layers	CIFAR10	8960

single set of quantization parameters for all values within each layer. At inference, weights are converted from quantized value to the floating point and computations are performed using floating-point kernels. *Split*We randomly selected 25 images from the dataset for the quantization experiments. Each image generates 729 verification instances in case of a patch attack on MNIST, and similarly, there are multiple verification instances generated from a single image for other perturbations (L_0-random, L_0-center, rotation, brightness) as described earlier.

Pruning: At each iteration, we prune the smallest 10% model weights in each layer. Similar to many other iterative pruning techniques, we prune only affine layers (we do not prune weights in the convolution layer or biases, since they are much fewer in number than affine layer weights). We verify the robustness of each model against patch perturbation. We randomly select 10 images from the dataset and create input regions for every possible 2×2 patch perturbation. The number of input regions per image is 729 for MNIST and 961 for CIFAR10. In this experiment, we perform 10 pruning iterations.

Implementation: Code for our tool is written in Python and it is implemented on top of the ELINA library (Singh et al., 2019c; 2018a) for numerical abstractions. For all our experiments, we use DeepZ (Singh et al., 2018a) as the library underlying our analyzer and verifier. We use the zonotope abstract domain for the analysis. However, we store the templates using box abstraction since checking containment in the box is much faster.

Baseline: As the baseline, we use the plain DeepZ verifier for verification without proof transfer. We use the publicly available implementation of DeepZ in ERAN library (Singh et al., 2018c).

Experimental Setup: For all the experiments we use 24 cores of an Intel Xeon E5-2687W CPU with a main memory of 64 GB running Linux operating system.