Towards a study of performance for safe neural network training

Nishant Kheterpal[1][0000−0001−5868−7843] and
Jean-Baptiste Jeannin[2][0000−0001−6378−1447]

[1] Robotics Institute, University of Michigan
nskh@umich.edu
[2] Department of Aerospace Engineering, University of Michigan
jeannin@umich.edu

Abstract. Neural networks perform well in complex, data-driven tasks, although guaranteeing their performance remains a challenge. Networks that perform well on training and test sets are not guaranteed to generalize to unseen or untested settings; arbitrary behavior may occur. Despite recent progress in verifying correctness properties of neural networks, it is often unclear how to repair networks when a verifier finds a counterexample. We present work in progress examining correct-by-construction neural networks and the performance impact of guaranteeing safety during training. Training a neural network amounts to exploring a high-dimensional, non-convex parameter space to find the best-performing point; we aim to study the impact of constraining such exploration during training on convergence, training time, and the final loss. We propose a study of safe training methods on the ACAS Xu neural network compression task, for which explicit correctness properties have been stated.

Keywords: formal verification, neural network verification, reachability

1 Introduction

Considerable research attention has been directed towards the use of neural networks due to their suitability for complex, data-driven tasks such as image classification, generative modeling, or other decision-making tasks. However, the universal function approximation characteristics of neural networks make it challenging to guarantee their performance. Generalization is not guaranteed, even if a network performs well on a training or test set. In safety-critical domains, such unintended behavior may cost lives or damage valuable equipment. As such, verification for these systems is a key research area.

Recent progress has been made in verifying neural networks with abstract interpretation [3]. Abstract domains like zonotopes, convex polytopes, and more are used to verify invariant specifications in recent work such as PRIMA [14]. Due to the complexity of neural networks and their applications, a performant verifier will likely find counterexamples (cases where the network does not perform as

intended), which must be addressed by retraining. However, it is not obvious how to retrain and guarantee that the same counterexamples will not occur, or that retraining will not introduce new counterexamples; after all, there is no formal guarantee that training examples will all be classified correctly. We seek to study how to guarantee correct behavior while integrating verifiers into the training loop.

We are motivated to study this problem based on the following challenge: training a neural network is a search problem in a high-dimensional vector space over a non-convex surface, and constraining the search by enforcing safety likely will impact performance. Unconstrained gradient descent methods may, for example, escape local optima and discover more globally optimal parameters using strategies like momentum [11]. However, if the network must satisfy safety properties throughout the entirety of training, the gradient steps may be limited to a small domain of the loss landscape.

In the following section, we present a series of safe training approaches which we aim to study using the ACAS Xu neural network compression task, a well-studied benchmark for which formally verified safety properties have been stated before [8,9]. This benchmark task provides clear input-output invariants that can be propagated through the layers of the network using abstract interpretation approaches and, correspondingly, used to constrain training to maintain safety throughout. Note that enforcing "correct" behavior for neural networks is a highly challenging task in which optimal performance may be computationally infeasible due to high dimensionality, non-convexity with respect to training parameters, and problem complexity. We seek to ground this open-ended question of safe training performance by beginning with a well-known benchmark task that has been studied extensively in past work.

We propose to perform several safe training experiments to assess three ways of constraining training to ensure the network takes safe actions. Two approaches may not guarantee correct results or training may terminate early: the first penalizes unsafety in the loss term and the second incorporates an "exploration budget" into training. The third approach directly modifies the parameters of the neural network during training using the intermediate representations computed by a reachability-oriented verifier. We intend to comprehensively study which properties were successfully verified and explore the impact of maintaining safety during training on final performance metrics such as loss relative to the original ACAS Xu lookup table.

1.1 ACAS Xu: A Case Study

The ACAS Xu collision avoidance system issues advisories to avoid near-misses or mid-air collisions between two aircraft viewed from a top-down perspective; as such advisories take forms like "Strong Left," "Weak Right," "Clear of Conflict," and others. Correctness can be checked geometrically in this top-down view: there are optimal advisories to maximally avoid a near-miss based on their configuration. The original ACAS Xu system chooses advisories based on a large

lookup table ranking 5 actions over 120 million states, which can be computationally demanding to store on embedded settings found in the low-resourced unmanned aircraft for which ACAS Xu is intended. Past work examined compressing the lookup table by approximating it first with a single and then multiple neural networks [8]. In the course of that work, those authors generated explicit verification conditions to check the correctness of the advisories generated by their neural network. They found counterexamples for properties dictating, for example, when the system should yield a "weak left" advisory, in which the trained deep neural network failed to return the correct advisory [9].

1.2 Safe Training Approaches

We intend to study the performance of several approaches for safe training; note that some approaches are heuristics that may not guarantee correctness. Our goal in this work is to study the effectiveness of certain approaches to safe training; as such, we will consider non-differentiable or heuristic approaches to compare them to other approaches with stronger guarantees.

The first approach, the unsafety penalty, computes the distance from the trained regions corresponding to each collision advisory to the verified safe regions from past work; this distance is used as part of the loss function used to train. Integrating correctness properties is the most naive approach, though also the simplest. By penalizing the system for the severity of the deviation from the verified region, the training process drives the neural network towards satisfying the geometric properties referenced above. We aim to test a variety of penalty weights and study the effect of the penalty on the performance of the fully trained neural network. The differentiablity of our loss penalty is a challenge here: such penalties must be implemented so that backpropagation can produce gradients with respect to the network weights. We envision building, for example, an interval domain analyzer in Tensorflow and run in the loop when computing loss [1].

The second approach allows the neural network to yield unsafe actions during portions of training, if it eventually converges to safety. Because the loss function is highly nonconvex in the weights and biases of the neural network, there is no guarantee that training will converge directly to optimal, safe behavior. As such, our second tactic will deploy an "exploration budget": a set number of training iterations during which training may yield unsafe actions, like the "temperature" parameter in simulated annealing. If the network violates safety properties after the exploration budget is expended, training begins anew from the last safe parameter set.

In our third safe training strategy, we aim to leverage verification tools that iteratively propagate some safe input set (such as the conditions under which some property of action choice holds) to directly adjust the parameters of the neural network so it always yields correct outputs; by modifying the weights in the final hidden layer, we can shift the output set to lie entirely within a safe output range. There are numerous reachability-based verifiers, like DeepPoly, PRIMA, and CROWN; we would like to study the relative performance of this

correct-by-construction approach using multiple verifiers in the loop. We aim to provide empirical results for the number of verified properties and trained performance for each approach and over various parameter values, such as the weighting of the unsafety penalty, and number of training iterations used as the exploration budget.

Our second and third approaches modify weights outside of the usual optimizer loop, which often includes momentum and other parameters. We aim to empirically study the effects of optimizer parameters. For example, we will evaluate performance with and without resetting momentum in our second approach when weights are modified. For our third approach, we may modify optimizer parameters like learning rate schedules based on the relative magnitude of the change made: if the weights are shifted by a large amount, we may raise the learning rate correspondingly, but small shifts may be better suited to lower learning rates. We will study the performance both with and without these heuristics in order to analyze their effectiveness.

1.3 Evaluation

Our experiments are still in progress, so the precise metrics we will use to measure performance remain uncertain. However, there are a few components of performance with which we are concerned. The primary motivation of this work is seeing whether constraining neural networks to be safe during training prevents them from learning effectively — functionally, whether safety-constrained gradient descent stays trapped in some initial configuration or fails to make progress because it does not satisfy desired properties. As such, we intend to examine the difference in loss compared to an unconstrained neural network. Another natural performance metric is the number or percentage of properties certified as correct, as this work is intended to study methods that guarantee safety. Additionally, we want to study the additional computational burden of running a verifier in the loop for each approach under consideration, and will measure wall-clock time or some other metric for training time.

2 Related Work

Ensuring neural network safety or correctness is a highly open-ended task, and the problem can be made arbitrarily challenging based on, for example, the choice of safety properties, network dimension, benchmark task, and more. As such, much past research has focused on the setting of robustness for neural networks, where generalization properties can be more easily stated and proven. Motivated by the challenge of adversarial attacks, in which imperceptible changes to neural network inputs cause significant errors [19,7], tools for training provably robust neural networks include DiffAI, which leverages an abstract interpretation approach inside the layers of the network itself using a zonotope domain [13,12]; CROWN [24] and its successor β-CROWN [21], which use linear or quadratic functions to bound activation functions and provide a lower bound of distortion,

further improved to handle neuron split constraints; and other tools. Additional work with CROWN leveraged interval bound propagation (IBP) and used schedule hyperparameters that gradually traded off standard and robust loss terms [23]. Small-scale networks with one hidden layer were studied in [15], and a pioneering paper in the field overapproximated activation functions and used linear programming to perform robust optimization [22]. Many tools for robust optimization and learning use custom loss functions to incentivize robustness. The survey [5] provides an overview of approaches for learning under constraints; the work covers input-output constraints of the sort studied for the ACAS Xu benchmark task [9].

Our first approach, which penalizes unsafety, takes a similar approach but defines penalties with respect to safety specifications (in the form of input-output constraints, for example), rather than robustness specifications (regularizing, say, around individual training examples). Differentiable logics may offer a way to encode safety specifications into the loss term in a general fashion, and we look forward to exploring them further [4,25]. Our third safe training strategy, which modifies the weights of a neural network in order to adhere to a safety specification, has similar goals to recent research into Provable Repair for ML systems. Past work has explored the complexity of modifying weights to guarantee performance [6], which is NP-complete at worst but can be simplified to a linear program if only the last layer is analyzed, as we do. The tools PRDNN and APRNN perform Provable Repair and have been applied to some of the ACAS Xu networks to repair violated counterexamples [18,20]. We aim to study PRDNN and APRNN in our future work, though the tools have limits in their input dimensionality.

There are several tools, like Reluplex [9], Marabou [10], PRIMA [14], ERAN [16,17], and others, focused on verifying that a neural network obeys a specification, in the form of invariants: inputs in some domain must produce outputs in some range.

The ACAS Xu system is a case study of considerable interest to the neural network community as a benchmark due to its formally verified safety properties; for example, it was examined in a closed-loop control context and found to be unsafe [2].

3 Future Work and Conclusion

Motivated by the challenge of ensuring neural networks obey verification properties, we have presented work in progress investigating the performance of three approaches to safe training of neural networks. Because of the highly open-ended nature of this problem, we focus on a benchmark task of interest to the machine learning verification community: the ACAS Xu neural network compression task. We propose integrating reachability-focused verifiers into the training loop to modify neural networks either via heuristic or direct manipulation of parameters, and present a few metrics we intend to study in order to examine the performance of these approaches. We invite comments on our research

objectives and approach, and look forward to engaging with the broader community focused on deploying machine learning systems with rigorous guarantees in safety-critical contexts.

Acknowledgements. This work was funded in part by a NASA Fellowship.

References

1. Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.
2. Stanley Bak and Hoang-Dung Tran. Neural Network Compression of ACAS Xu Early Prototype is Unsafe: Closed-Loop Verification through Quantized State Backreachability. volume 13260, pages 280–298. 2022. arXiv:2201.06626 [cs, math].
3. Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages - POPL '77*, pages 238–252, Los Angeles, California, 1977. ACM Press.
4. Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. DL2: Training and Querying Neural Networks with Logic. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1931–1941. PMLR, May 2019. ISSN: 2640-3498.
5. Eleonora Giunchiglia, Mihaela Catalina Stoian, and Thomas Lukasiewicz. Deep learning with logical constraints. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, jul 2022.
6. Ben Goldberger, Guy Katz, Yossi Adi, and Joseph Keshet. Minimal Modifications of Deep Neural Networks using Verification. pages 260–240, 2020.
7. Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples, March 2015. arXiv:1412.6572 [cs, stat].
8. Kyle D. Julian, Jessica Lopez, Jeffrey S. Brush, Michael P. Owen, and Mykel J. Kochenderfer. Policy compression for aircraft collision avoidance systems. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–10, Sacramento, CA, USA, September 2016. IEEE.
9. Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks, May 2017. arXiv:1702.01135 [cs].
10. Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, volume 11561, pages 443–452. Springer International Publishing, Cham, 2019. Series Title: Lecture Notes in Computer Science.

11. Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980 [cs].
12. Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3578–3586. PMLR, July 2018. ISSN: 2640-3498.
13. Matthew Mirman, Gagandeep Singh, and Martin Vechev. A Provable Defense for Deep Residual Networks, January 2020. arXiv:1903.12519 [cs, stat].
14. Mark Niklas Müller, Gleb Makarchuk, Gagandeep Singh, Markus Püschel, and Martin Vechev. PRIMA: General and Precise Neural Network Certification via Scalable Convex Hull Approximations. *Proceedings of the ACM on Programming Languages*, 6(POPL):1–33, January 2022. arXiv:2103.03638 [cs].
15. Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified Defenses against Adversarial Examples, October 2020. arXiv:1801.09344 [cs].
16. Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and Effective Robustness Certification. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
17. Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, January 2019.
18. Matthew Sotoudeh and Aditya V. Thakur. Provable repair of deep neural networks. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 588–603, Virtual Canada, June 2021. ACM.
19. Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, February 2014. arXiv:1312.6199 [cs].
20. Zhe Tao, Stephanie Nawas, Jacqueline Mitchell, and Aditya V. Thakur. Architecture-Preserving Provable Repair of Deep Neural Networks. *Proceedings of the ACM on Programming Languages*, 7(PLDI):443–467, June 2023. arXiv:2304.03496 [cs].
21. Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Complete and Incomplete Neural Network Robustness Verification, October 2021. arXiv:2103.06624 [cs, stat].
22. Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope, June 2018. arXiv:1711.00851 [cs, math].
23. Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards Stable and Efficient Training of Verifiably Robust Neural Networks, November 2019. arXiv:1906.06316 [cs, stat].
24. Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient Neural Network Robustness Certification with General Activation Functions, November 2018. arXiv:1811.00866 [cs, stat].
25. Natalia Ślusarz, Ekaterina Komendantskaya, Matthew L. Daggitt, Robert Stewart, and Kathrin Stark. Logic of Differentiable Logics: Towards a Uniform Semantics of DL, May 2023. arXiv:2303.10650 [cs].