ART : Abstraction Refinement-Guided Training for Provably Correct Neural Networks

No Institute Given

Abstract. Artificial Neural Networks (ANNs) have demonstrated remarkable utility in various challenging machine learning applications. While formally verified properties of their behaviors are highly desired, they have proven notoriously difficult to derive and enforce. Existing approaches typically formulate this problem as a *post facto* analysis process. In this paper, we present a novel learning framework that ensures such formal guarantees are *enforced by construction*. Our technique enables training provably correct networks with respect to a broad class of safety properties, a capability that goes well-beyond existing approaches, *without* compromising accuracy. Our key insight is that we can integrate an optimization-based abstraction refinement loop into the learning process and operate over dynamically constructed partitions of the input space that considers accuracy and safety objectives synergistically. The refinement procedure iteratively splits the input space from which training data is drawn, guided by the efficacy with which such partitions enable safety verification. We have implemented our approach in a tool (ART) and applied it to enforce general safety properties on unmanned aviator collision avoidance system ACAS Xu dataset [1]. We have also evaluated our approach on ensuring sophisticated safety properties in cyber-physical applications using reinforcement learning models. Importantly, we empirically demonstrate that realizing safety does not come at the price of accuracy. Our methodology demonstrates that an abstraction refinement methodology provides a meaningful pathway for building both accurate and correct machine learning networks.

Keywords: Certified Machine Learning · Neural Network Training · Abstraction Refinement.

1 Evaluation

We have performed a comprehensive evaluation of our approach to validate the feasibility of building neural networks that are correct-by-construction over a range of sophisticated correctness properties. All experiments reported in this section were performed on a Ubuntu 16.04 system with 3.2GHz CPU and NVidia GTX 1080 Ti GPU with 11GB memory. All experiments uses the DeepPoly abstract domain [7] implemented on Python 3.6 and PyTorch 1.0.1 [5].

1.1 ACAS Xu Benchmark

Our first evaluation study centers around the network architecture and correctness properties described in the Airborne Collision Avoidance System for Unmanned Aircraft (ACAS Xu) dataset [1, 2]. A family of 45 neural networks are used in the avoidance system; each of these networks consists of 6 hidden layers with 50 neurons in each hidden layer. ReLU activation functions are applied to all hidden layer neurons. All 45 networks take a feature vector of size 5 as input that encodes various aspects of an airborne environment. The outputs of the networks are prediction scores over 5 advisory actions to select the advisory action.

In the evaluation, we reason about sophisticated correctness conditions of the ACAS Xu system in terms of its aggregated ability to preserve a total of 10 correctness properties distributed among all 45 networks following the specifications in [2]. Each network is supposed to satisfy some subset of these properties.

All correctness properties Φ can be formulated in terms of input ($\Phi_{\texttt{in}}$) and output ($\Phi_{\texttt{out}}$) predicates. Informal notions of input predicates such as "*intruder is distant*" and "*intruder is significantly slower than ownship*" can be concretely interpreted in value bounds on input feature vectors. Common output predicates such as "*Strong Right is advised*" can be formalized by linear equality formulas.

Setup Unfortunately, the training and test set of ACAS Xu is not publicly available online. We, therefore, uniformly sample a total of 10k training set data points and 5k test set data points for each of the provided 45 networks in evaluation. To demonstrate the effectiveness of Art , we train each network using its safety specification and the prepared training set. Whether the trained networks is correct-by-construction is recorded, as well as their accuracy evaluated on sampled test sets.

As reported in [1], 36 out of 45 provided networks are reported with safety property violations and 9 are reported safe. We evaluate Art on those 36 unsafe networks to demonstrate the effectiveness of generating correct-by-construction networks. The test sets from unsafe networks may contain unsafe points and are thus unauthentic, so we apply Art on those 9 already safe networks to demonstrate the accuracy overhead when enforcing the safety properties.

Applying Art During each training epoch, our implementation refines up to 200 abstractions at a time that expose the largest correctness losses. The Adam optimizer [3] is used in both training tasks and runs up to 100 epochs with learning rate 0.001 and a learning rate decay policy if the loss has been stable for some time.

To demonstrate the importance of abstraction refinement mechanism, we also compare between the results with and without refinement (as done in existing works [4]). For all experiments with refinement enabled, refinement operations are applied to derive up to 5k refined input space abstractions before training starts. The detailed results are shown in Table 1. For completeness, we record the correct-by-construction enforced rate (Safe%) and the evaluated accuracy statistics for both tasks among multiple runs.

	Safe%	Min Accuracy	Mean Accuracy	Max Accuracy
36 unsafe nets via ART	100%	92.68%	96.20%	98.70%
36 unsafe nets via ART without refinement	94.44%	89.18%	94.30%	98.34%
9 safe nets via ART	100%	93.68%	96.21%	99.92%
9 safe nets via ART without refinement	88.89%	86.32%	94.39%	99.92%

Table 1: Using ART to build correct-by-construction networks for the ACAS Xu dataset.

Note that ART successfully generates correct-by-construction networks for all scenarios with only minimal loss in accuracy. On the other hand, if refinement is disabled, it fails to generate correct-by-construction networks for all cases, and displays lower accuracy than the refinement-enabled instantiations.

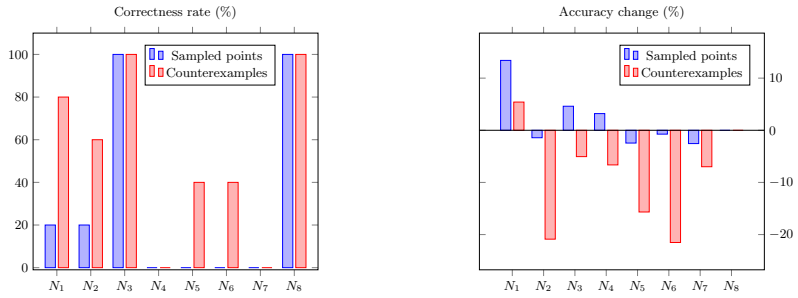

Fig. 1: Correctness rate and accuracy change of *post facto* training using sampled points or counterexamples. Results are normalized based on the baseline networks.

Comparison with post facto *training loop* We also consider a comparison of our abstraction refinement-guided training for correct-by-construction networks against a *post facto* training loop that feed concrete correctness related data points to training loops.

Such concrete points may be the sampled points from provided specification or the counterexamples collected from an external solver. The results on 8 representative networks comparing to the same baseline are shown in Figure 1. These 8 networks belong to a representative set of networks that cover all provided safety properties. For the experiment using sampled data points, 5k points sampled from correctness properties are used during training. For the experiment using counterexamples, all counterexamples from correctness queries to external verifier ReluVal [8] are collected and used during training. In both experiments, the points from original training set are used for jointly training to preserve accuracy. We concluded the experiments using counterexamples after 20 epochs since no improvement was seen after this point.

Both experiments fail to enforce correctness properties in most cases and they may impose great impact to model accuracy compared to the baseline network. We believe this result demonstrates the difficulty of applying a counterexample-guided training loop strategy for generating safe networks compared an abstraction-guided methodology.

1.2 Cyber-Physical System Benchmarks

We additionally evaluated ART on 7 cyber-physical system benchmarks adapted from [9]. For each benchmark, the goal is to apply ART to train a verifiable neural network controller with respect to a safety constraint (*e.g.* a pendulum's angle and angular velocity must be operated within a safe range).

In our experiments, each benchmark is modeled as a closed-loop state transition system whose transition function G is governed by the physical law of the system (defined as linearized system dynamics). G takes as input the current state and a control action and produces the next state. Given the state $s_i \in \mathbb{R}^d$ at timestep i, its next state $s_{i+1} \in \mathbb{R}^d$ at timestep $i+1$ can be derived by:

$$s_{i+1} = G_F(s_i) = G\left(s_i, F(s_i)\right)$$

where F is the feed-back neural network controller of the system that generates a *continuous* control action based on the current state. Therefore the transition function G is parameterized by F. Formally, given the initial constraint $\Phi_{\texttt{init}}$, the safety constraint $\Phi_{\texttt{safe}}$, a timestep bound K, a neural network controller F is K-bounded safe with respect to a state transition function G, denoted as $F \models (\Phi_{\texttt{init}}, \Phi_{\texttt{safe}}, G, K)$, iff:

$$\forall s_0 \models \Phi_{\texttt{init}},\ \forall i \in 1 \ldots K,\ s_i = \underbrace{G_F \circ \ldots \circ G_F}_{i \text{ times}}(s_0) \Rightarrow s_i \models \Phi_{\texttt{safe}}.$$

The training of F is conducted in our experiments as a supervised imitation learning task [6]. F is learned to imitate the behavior of an expert (a provided oracle). Our augmented training loop, on the one hand, clones the control signals of the oracle at concrete reachable states by reducing the accuracy loss $\ell_{\mathcal{A}}$, on the other hand, enforces F's safety at the same time on the DeepPoly abstract domain. Since G is a linearized function, the above compositional model can be deemed as a chained neural network which naturally extends our definitions. Correct-by-construction of F in these benchmarks creates a challenging task because F can be invoked many times in a control path as opposed to the setting in Sec. 1.1. For each benchmark, our training procedure iteratively increases K to train F and verify the safety of F in large timesteps until a timeout is reached (20 minutes in our experiments).

The results of all benchmarks are shown in Table 2. The architecture of a controller network is recorded in the Network Architecture column in which 64×64 denotes a fully-connected network with 2 hidden layers each with 64 ReLU neurons. In each training iteration, up to 200 abstractions with the largest correctness losses are chosen for refinement (i.e. the hyper-parameter k is set to

Benchmark	Network Architecture	Certified K Steps	Accuracy
Cruise	64×64	300	0.00215
Tape	64×64	300	0.00005
Suspension	64×64	300	0.00545
Quadcopter	30×30	65	0.01619
Pendulum	64×64	84	0.00895
CartPole	30×30	86	0.00156
Oscillator	64×64	189	0.05560

Table 2: Training results of cyber-physical system benchmarks.

200). The Adam optimizer [3] was used with learning rate 0.001 and ran up to 20 minutes. A learning rate decay policy was applied if the loss has been stable for some time. For each trained neural network controller, we evaluated how many steps can it be trained and verified safe when the timeout is reached, as recorded in the Certified K column of Table 2. To demonstrate the accuracy impact of ART as in Section 1.1, we collected 10k input/output pairs of the trained neural network from sampled trajectories of each benchmark system. The L_1-norm distances between ART -generated networks and provided oracles used in imitation learning are recorded in the Accuracy column. We are unaware of any other system capable of statically certifying sophisticated K-bounded safety properties in a correct-by-construction manner (i.e., ensuring safety during training) for these kinds of cyber-physical systems.

References

1. Julian, K.D., Lopez, J., Brush, J.S., Owen, M.P., Kochenderfer, M.J.: Policy compression for aircraft collision avoidance systems. In: 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC). pp. 1–10 (Sep 2016). https://doi.org/10.1109/DASC.2016.7778091
2. Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I. pp. 97–117 (2017). https://doi.org/10.1007/978-3-319-63387-9_5, https://doi.org/10.1007/978-3-319-63387-9_5
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
4. Mirman, M., Gehr, T., Vechev, M.T.: Differentiable abstract interpretation for provably robust neural networks. In: Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. pp. 3575–3583 (2018), http://proceedings.mlr.press/v80/mirman18b.html
5. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.:

Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada. pp. 8024–8035 (2019), http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library

6. Ross, S., Gordon, G.J., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011. pp. 627–635 (2011), http://proceedings.mlr.press/v15/ross11a/ross11a.pdf
7. Singh, G., Gehr, T., Püschel, M., Vechev, M.T.: An Abstract Domain for Certifying Neural Networks. PACMPL **3**(POPL), 41:1–41:30 (2019), https://dl.acm.org/citation.cfm?id=3290354
8. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018. pp. 1599–1614 (2018), https://www.usenix.org/conference/usenixsecurity18/presentation/wang-shiqi
9. Zhu, H., Xiong, Z., Magill, S., Jagannathan, S.: An Inductive Synthesis Framework for Verifiable Reinforcement Learning. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019. pp. 686–701 (2019). https://doi.org/10.1145/3314221.3314638, https://doi.org/10.1145/3314221.3314638