Safety Verification and Repair of Deep Neural Networks

Xiaodong Yang [1] Tom Yamaguchi [2] Bardh Hoxha [2] Danil Prokhorov [2] Taylor T Johnson [1]

Abstract

This paper presents the Veritex tool for safety verification and repair of deep neural networks (DNNs). Veritex includes methods for exact (sound and complete) analysis and over-approximative (sound and incomplete) reachability analysis of DNNs using novel set representations, including the facet-vertex incidence matrix, face lattice, and $\mathcal{V}$-zono. In addition to sound and complete safety verification of DNNs, these methods can also efficiently compute the exact output reachable domain, as well as the exact unsafe input space that causes safety violations of DNNs in the output. More importantly, based on the exact unsafe input-output reachable domain, Veritex can repair unsafe DNNs on multiple safety properties with negligible performance degradation. The repair is conducted by updating the DNN parameter through retraining. The approach also works in the absence of the safe model reference and the original dataset for learning. Veritex primarily addresses the issue of constructing provably safe DNNs, which is not yet significantly addressed in most of the current formal methods for trustworthy artificial intelligence (AI). The utility of Veritex is evaluated for these two aspects, specifically safety verification and DNN repair. Benchmarks for verification include the ACAS Xu networks, and benchmarks for the repair include unsafe ACAS Xu networks and an unsafe agent trained in deep reinforcement learning (DRL).

1. Introduction

Deep neural networks (DNNs) have been widely utilized in safety-critical systems with learning-enabled components, such as autonomous vehicles. Despite successful applications in many areas, their trustworthiness remains a major concern in realizing reliable autonomy due to their black-box nature with complex nonlinear characteristics. It has been shown that slight perturbations in their inputs can cause unpredictable misbehavior in the output. Recently, much effort has been made to develop techniques for formal analysis of DNNs, such as their safety certification (Tran et al., 2020; Katz et al., 2019; Shriver et al., 2021; Dutta et al., 2018; Tran et al., 2019; Singh et al., 2019; Yang et al., 2021a; Botoeva et al., 2020; Sotoudeh & Thakur, 2021; Wang et al., 2020). However, these methods that conduct post-training verification of DNNs can not address the problem of producing provable safe DNNs when they violate safety specifications.

In this paper, we introduce a tool called Veritex that performs set-based reachability analysis of DNNs, safety certification, and repair of unsafe DNNs. The reachability analysis includes the computation of both the exact and over-approximated output reachable domain for an input domain, and also the computation of the exact unsafe input space that leads to the safety violation in the output. Here, the reachable domain contains all the possible reachable states of a system given an input bounded domain. It is a union of reachable sets, which refer to bounded convex polytopes. A variety of efficient set representations are utilized to construct the reachable set, such as facet-vertex incidence matrix (FVIM) (Yang et al., 2021a), face lattice (FLattice) (Yang et al., 2021b) and $\mathcal{V}$-zono (Yang et al., 2022). If the exact output reachable domain does not intersect with specified unsafe domains, the DNN is determined to be safe by Veritex. Otherwise, the DNN is unsafe and Veritex computes the entire unsafe input space. The repair in Veritex is a retraining process. Based on the unsafe input-output reachable domain computed in the reachability analysis, Veritex can repair an unsafe DNN on multiple safety properties with negligible impact on its original performance. Here, the safety property refers to specifications that describe a desired or unsafe output domain of a DNN for an input domain.

Veritex primarily supports feedforward neural networks (FFNNs) which are commonly used as controllers in learning-enabled control systems. It can perform reachability analysis, safety verification and unsafe network repair. It also supports the reachability analysis and safety verifi-

[*]Equal contribution [1]Vanderbilt University, Nashville, USA [2]TRINA, Toyota NA R&D, Ann Arbor, MI, USA. Correspondence to: Bardh Hoxha <bardh.hoxha@toyota.com>, Taylor T Johnson <taylor.johnson@vanderbilt.edu>.

1^{st} *Workshop on Formal Verification of Machine Learning*, Baltimore, Maryland, USA. Colocated with ICML 2022. Copyright 2022 by the author(s).

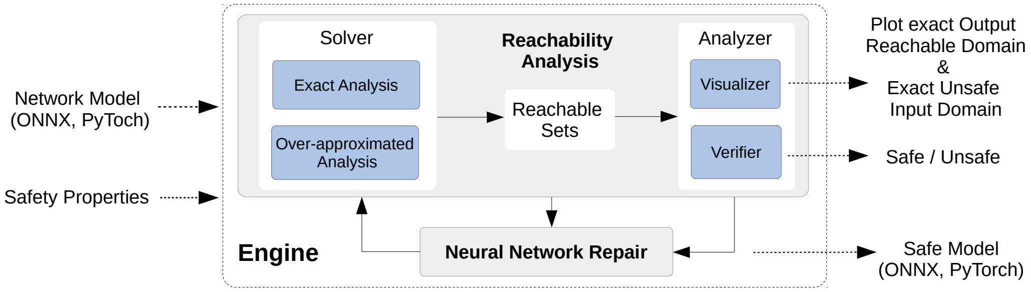


Figure 1. An overview of Veritex architecture.

cation of convolutional neural networks (CNNs). To speed up its computation, we also design a work-stealing parallel framework. It is a well known scheduling algorithm for dynamic multi-threaded computation. In the evaluation, two cases studies are presented. They include the safety verification and repair of ACAS Xu networks (Katz et al., 2017), and an unsafe DNN agents for a rocket-lander system in DRL (roc). The experimental results show that Veritex has the highest efficiency in the safety verification of the ACAS Xu networks compared to all 13 related works, and that it can repair all unsafe DNNs with negligible performance degradation. Veritex currently supports DNNs with low-dimensional inputs. It has been shown that the exact analysis of DNNs is a NP-complete problem (Katz et al., 2017). Existing exact analysis methods, including our approaches based on FVIM and Flattice, are only scalable to neural networks with small input ranges. To handle limitations of the exact analysis, our tool also includes an over-approximation method based on activation-function linearization, which maintains the possibility of supporting large-scale DNNs for image classification in the future.

2. Overview and Features

Veritex is an object-oriented software programmed in Python. It takes in two inputs as shown in Fig. 1, the network model and safety properties. Veritex supports the standardized format ONNX and PyTorch for the network and the unified format VNN-LIB[1] for the safety properties. In DNN verification, VNN-LIB is the emerging standard that can specify safety properties of a DNN by defining their input domains and their corresponding unsafe output domains. Roughly for specifications, it is an extension of SMT-LIB with additional assumptions. With the network model and its safety properties, Veritex can compute the exact or over-approximated output reachable domain and also the entire unsafe input space if exists. It supports the plotting of 2 or 3-dimensional polytopes. When the repair option is enabled, it will produce a provable safe network in ONNX or PyTorch format. Unlike tools (era; ver; cro; Bak, 2021; Tran et al., 2020; Katz et al., 2019), Veritex does not involve LP problems in the reachability analysis and verification of DNNs. Therefore, it does not require any commercial optimization solvers, which makes its installation straightforward. The main features of Veritex are summarized in Table 1.

Table 1. Overview of primary features in Veritex. FC stands for fully-connected layers. CONV stands for convolutional layer. BN stands for batch normalization.

Feature	Exact Analysis	Over-approximation Analysis
Set representations	FVIM, FLattice	$\mathcal{V}$-zono
Safety Verification	Sound and complete	Sound
Network Repair	Provably safe networks (FFNNs)	
Activation Function	ReLU	ReLU, Sigmoid, Tanh
Layer Types	FC, CONV, MaxPool, BN	
Parallel Computing	Work-stealing parallel	

2.1. Engine and Components

The engine of Veritex contains two main modules: reachability analysis of DNNs and DNN repair, as shown in Fig. 1. The former contains functions to compute the reachable domains of a DNN. The latter contains functions to repair unsafe DNN on multiple safety properties.

2.1.1. Reachability Analysis Module

The module includes a solver for the computation of the reachable domain and an analyzer for the safety verification and and reachable-domain visualization. The solver constructs the incoming network model and its safety properties with a network object and a set of property objects. It can compute the exact or over-approximated output reachable domain of the network. It can also compute the exact unsafe input space using the backtracking algorithm (Yang et al., 2021a) in the exact analysis.

The exact analysis utilizes set representations FVIM and Flattice to compute output reachable sets whose union is

[1] http://www.vnnlib.org/standard.html

the exact output reachable domain. These reachable sets can be sent to the verifier for a sound and complete safety verification, which returns either ”safe” or ”unsafe”. The over-approximation utilizes the set representation $\mathcal{V}$-zono to over approximate the output reachable domain. This reachable domain can be sent to the verifier for a sound but incomplete safety verification, which returns either ”safe” or ”unknown”. The visualizer plots a reachable domain by projecting it into a 2 or 3-dimensional space. This visualization is critical for the analysis of the impact of repair methods on DNN reachability.

2.1.2. DNN Repair Module

This module eliminates safety violations through optimization of a loss function in the retraining of a DNN. In each iteration of repair, it interacts with the reachability analysis module. Given a DNN and its violated safety properties, they are first fed into the reachability analysis module, where its exact unsafe input-output reachable domain over these properties are computed. Recall that the reachable domain consists of reachable sets, which are convex polytopes. Then, the vertices of these sets are selected as representative data pairs $(\mathbf{x}, \mathbf{y})$ to fully represent this reachable domain. They distribute over this domain, including all its extreme points. They are used to construct the distance between the unsafe reachable domain and the safe domain of the DNN. By minimizing this objective function, the repair can gradually eliminate the unsafe reachable domain, generating a provably safe DNN. When there is a safe model as a reference for the repair, adversarial $\mathbf{x}$ can be fed into this model to generate safe and correct $\hat{\mathbf{y}}$ for the repair. Otherwise, $\hat{\mathbf{y}}$ is set to the closet safe output to $\mathbf{y}$ for the minimal modification.

In addition to the objective function above, the repair also incorporates another objective function into the loss function, which aims to minimize the DNN parameter deviation. This is because slight changes in the parameter can cause unexpected performance degradation. This function minimizes the difference between the predicted output of the repaired network for the training data and the true output in the training data. A weighted-sum method is applied for this multi-objective optimization problem. Two positive real-valued α and β represent the weights of each objective function and $\alpha + \beta = 1$. This repair is named the minimal repair. If the original dataset is not available, it can be sampled from the original network. The sampled data are purified by removing unsafe data before the training. Or users can set $\alpha = 1$ and $\beta = 0$ to transform the optimization into a single-objective optimization. Then, only the objective function for repair is considered, which is named the non-minimal repair.

In practice, the solving of the minimal repair is less efficient than the non-minimal repair due to the Pareto optimality issue in the multi-objective optimization, where one objective function cannot be optimized without worsening the optimization of other objective functions.

2.2. Work-stealing Parallel computation

In the exact analysis, different linearities that the ReLU activation function exhibits over its input ranges $x \geq 0$ and $x < 0$ are separately considered. Therefore, when an input reachable set to one ReLU neuron spans its two input ranges, this set will be divided into two subsets which are separately processed with respect to the linearity in that range. Afterward, these two subsets will be input sets to another neuron. Here, the state $\mathcal{S} = (P, l, N)$ is defined for this computation, where P is a reachable set, l denotes the index of that layer, and N denotes a list of neurons in the layer that will process $\mathcal{S}$. After one neuron, the state $\mathcal{S}$ spawns at most two states $\mathcal{S}'$s with updated P's and N's. This state concept is also applied in the max-pooling layer. One pooling operation normally contains more linearities than the ReLU neuron and thus spawns more states. In the affine-mapping layer, such as fully-connected layer and convolutional layer, P in the state will be transformed to one new reachable set P' accordingly.

In the work-stealing parallel computing, each processor computes their states and store additional states in a local queue for future processes. One processor becomes idle when its local queue is empty. Then this processor steals states from other processors with a globally-shared queue as the agent, such that it can enable the full use of the processors. The process of states will be terminated once they reach the end of the DNN, where different callback functions can be invoked. In this phase, the reachable set P in the state is an output reachable set of the DNN. The callback functions include the safety verification and the computation of unsafe input space with the backtracking algorithm.

3. Reachability Analysis and Set Representations

3.1. Reachability Analysis

The reachability analysis in Veritex includes the computation of exact or over-approximated output reachable domain and exact unsafe input subspace for a bounded input domain. This computation can be formulated by

$$\mathcal{L}(P) = (\mathcal{E}_n \circ \cdots \circ \mathcal{E}_2 \circ \mathcal{E}_1 \circ \mathcal{T})(P)$$
$$\mathcal{N}(P) = (\mathcal{L}_n \circ \cdots \circ \mathcal{L}_2 \circ \mathcal{L}_1)(P)$$

where $\mathcal{L}$ denotes the reachable-set computation in one layer, P denotes an input reachable set, $\mathcal{E}$ denotes the computation in one ReLU neuron and $\mathcal{T}$ denotes the preceding affine mapping. The reachable sets are computed layer by layer until the last layer. Similarly, in the computation of CNNs, $\mathcal{E}$ also refers to one pooling operation in the max-pooling layer, and $\mathcal{T}$ also refers to the convolutional computation or the batch normalization. The non-linearity of ReLU DNNs origins from piecewise linearity of the max function in the ReLU activation function and max-pooling operation. In the exact analysis, different linearities are separately considered for the reachable-set computation. Therefore, an output reachable set is actually the output of a linear region of the DNN. A **linear region** refers to a maximal convex subspace of the input domain, on which the DNN is linear.

3.2. Set Representations

The set representation encodes geometric information of a convex polytope, which directly affects the efficiency of reachability analysis. Veritex includes multiple set representations, FVIM, FLattice and $\mathcal{V}$-zono.

3.2.1. Facet-vertex Incidence Matrix (FVIM)

FVIM is an efficient representation to encode the combinatorial structure of a polytope. Since this set representation tracks the vertices (extreme points) of a polytope, any LP problems involved can be avoided. There are two types of operations on FVIM in the reachable-set computation. The first one is the affine mapping from the weight parameter in the fully-connected layer, the convolutional layer and the batch normalization. This operation will only modify the value of vertices but preserve the FVIM of a polytope. Therefore, its implementation in Veritex is straightforward. The other operation is the process by the max function in ReLU neurons and Max-pooling layers, whose details are discussed in (Yang et al., 2021b). In brief, the vertex adjacency can be efficiently deduced from the encoded facet-vertex relation, which facilitates the update of reachable sets in the max function.

With this representation, Veritex computes the exact output reachable domain. Furthermore, the computation also tracks the affine-mapping relation between an output reachable set and its linear region. Therefore, Veritex can backtrack exact unsafe input subspace that causes safety violation in the output. This set representation can be only applied to simple polytopes (Yang et al., 2021a). The common input interval domain to a DNN is a simple polytope. Affine mapping does not change this attribute. A reachable set computed from a simple polytope in the max function is still a simple polytope if none of its vertices lies in the boundary distinguishing the linearities of this max function. In practice, this situation extremely unlikely happens because of floating-point computation. Veritex can also detect this situation. In case, Veritex implements another set representation, Face Lattice.

3.2.2. Face Lattice (FLattice)

Compared to FVIM, the face lattice structure encodes the complete combinatorial structure of a polytope, describing all the containment relation between different-dimensional faces. Therefore, it is scalable to represent general polytopes. FLattice is also for the exact analysis of ReLU DNNs. The affine-mapping operation on it is the same as FVIM. Similarly, in the process of the max function, the vertex adjacency also needs to be achieved for the reachable-set update. Since FLattice has more face-containment relation to process, its efficiency is slightly lower than FVIM. This set representation also can backtrack exact unsafe input space given an unsafe output domain, the same strategy as FVIM. Overall, FLattice is compatible with the operations on FVIM in the reachable-set computation, and it is a convenient and effective alternative to address the issue in FVIM.

3.2.3. $\mathcal{V}-$zono

$\mathcal{V}$-zono is an enhanced vertex-representation of zonotope, which is used to construct the over-approximated reachable set in the linear relaxation of activation functions, such as ReLU, Sigmoid and Tanh. This zonotope-based reachability method computes the over-approximated output reachable domain of a DNN and can be used for sound but incomplete safety verification. Since it does not consider different linearities in the activation function, this method is faster than the exact analysis. However, the approximation error is accumulated with respect to each neuron, which can yield a conservative approximation. Normally, this method is used for safety verification with small input domains. Veritex also combines the exact analysis with this method in the safety verification and the computation of unsafe input-output reachable domain of DNNs. Because the over approximation method can quickly filter out spaces that does not contain unsafe elements in the beginning of its computation and significantly improve the computational efficiency.

4. Evaluation

4.1. Safety Verification of ACAS Xu Networks

The performance of Veritex on the safety verification of 45 ACAS Xu networks is compared to the standardized competition results in VNN-COMP’21 (Bak et al., 2021), where most of the state-of-art methods participated. All 13 methods and tools that participated are considered in the comparison. Our hardware configuration is set to the standard configuration, AWS, CPU: r5.12×large, 48vCPUs, 384 GB memory.

Veritex combines the exact analysis and the over-approximation analysis for a fast, sound and complete verification. The verification time of all 186 instances of each

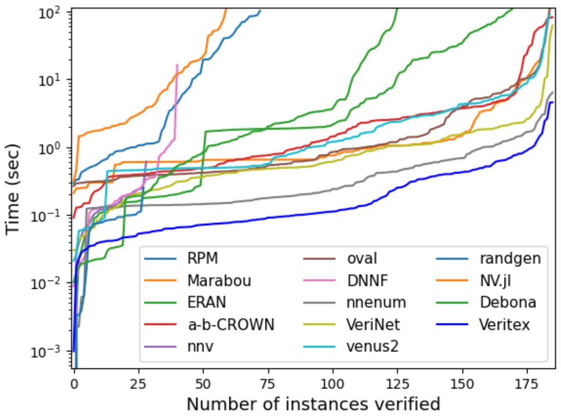


Figure 2. Cactus plot of the running time of the safety verification for ACAS Xu from VNN-COMP'21. The running time of failed instances which return 'unknown' or 'timeout' is not included. Timeout is 116 seconds. Compared to all the related works, Veritex exhibits the highest efficiency.

method is shown in the cactus-plot Fig. 2. We can notice that compared to those 13 methods, Veritex can complete all the verification within the 116-second timeout and exhibits the highest efficiency. There are 10 methods that are over-approximation based fail to verify all the instances due to their conservativeness. There are 3 methods that can also verify all the instances within the timeout, and they are α-β-CROWN (cro), nnenum (Bak, 2021) and VeriNet (Henriksen & Lomuscio, 2020). In terms of the total running time, Veritex is $16.8\times$ faster, $1.8\times$ faster, and $5.0\times$ faster than these 3 methods, respectively. This is because the set representation in Veritex contains vertices of reachable sets, and thus can avoid LP problems that commonly exists in these related works. The other reason is that the incorporation of the over-approximation analysis can quickly filter out safe subspaces in the input domain and thus avoid further computation on them.

4.2. Repair of Unsafe ACAS Xu Networks and Unsafe DNN Agents

Among those 45 networks, there are 35 unsafe networks violating at least one of their safety properties. Their original dataset is not publicly available. Therefore, a set of 5k test data is sampled from each original network for the accuracy analysis of their repaired network, on which the accuracy of these original networks is 100%. Here, the accuracy refers to the ratio of correct predictions on the test data. In this case study, we apply the non-minimal repair and compare Veritex to ART which is a well-known repair method for DNNs.

Table 2. Repair of ACAS Xu neural network controllers. Veritex successfully repairs all 35 unsafe networks with little accuracy degradation.

Methods	Repair Successes	Minimal Accuracy	Mean Accuracy	Maximal Accuracy
Veritex	35/35	98.74%	99.70%	100.0%
Art	34/35	89.08%	94.57%	98.06%
Art-refinement	35/35	88.82%	95.85%	98.64%

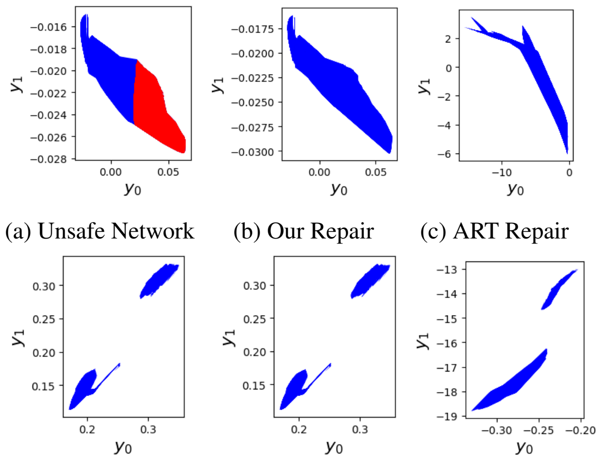
Table 3. Running time (**sec**) of Veritex and ART. Veritex shows a higher efficiency than ART-refinement in most of the instances.

Methods	Minimal Time	Mean Time	Maximal Time	Time (N_{19})	Time (N_{29})
Veritex	8.4	77.7	230.2	11250.7	2484.1
Art	63.6	64.6	91.7	67.5	72.6
Art-refinement	83.4	86.0	124.3	82.5	88.4

The result of the ACAS Xu network repair is shown in Table 2. We can notice that Veritex can repair all the 35 unsafe networks. ART can repair 34 networks, and ART-refinement which is an improved version of ART can also repair all the networks. In terms of accuracy, our repaired networks exhibit a much higher accuracy than ART and ART-refinement. Some of our repaired networks even have 100% accuracy, showing much less performance degradation.

Besides the accuracy, we also analyze the reachability change of repaired networks, because the reachability of a network comprehensively reflect its behaviors. A desired repair should fix all safety violations of a network and meanwhile preserve its safe behaviors. Here, we apply Veritex to plot the output reachable domain of the original and repaired network N_{21} on their safety properties, and then analyze their difference. Network N_{21} has safety properties $1, 2, 3, 4$ and it violates the property 2. The output reachable domain of the original network, Veritex-repaired network and ART-refinement-repaired network on the property 1&2 is shown in (a), (b) and (c) in Fig. 3. Their output reachable domains on the property 3&4 are shown in (d), (e) and (f). All domains are projected on $(\mathbf{y}_0, \mathbf{y}_1)$ for visualization. The unsafe reachable domain is plotted in red. We can notice that the unsafe reachable domain on the property 2 is eliminated after the repair by Veritex and ART. We can also notice that compared to ART, Veritex modifies the reachability less. This is also shown by the reachability on the property 3&4 in (d), (e) and (f).

The running time of Veritex and ART is shown in Table 3. The repair of N_{19} and N_{29} by Veritex takes more time than ART. This is because that the exact reachability analysis of networks is an NP-complete problem (Katz et al., 2017). The safety properties of these 2 networks specify very large input domains, therefore, their analysis is more computa-



(a) Unsafe Network (b) Our Repair (c) ART Repair

(d) Unsafe Network (e) Our Repair (f) ART Repair

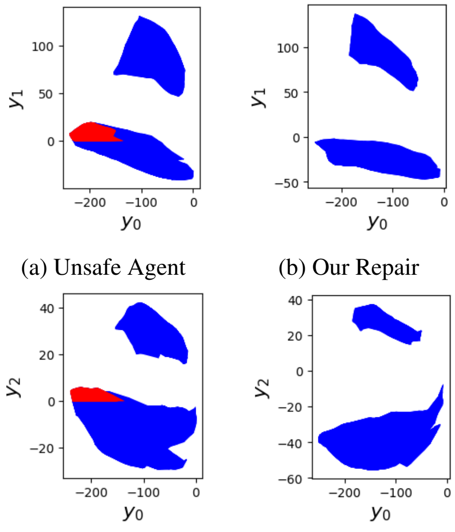
Figure 3. Reachability of the original network and the repaired networks on Properties 1&2 (a,b,c), 3&4 (d,e,f). The output reachable domains are projected on $(\mathbf{y}_0, \mathbf{y}_1)$. Red area represents the unsafe reachable domain. When projected on the lower dimensional space, the unsafe reachable domain overlaps with the safe reachable domain, as shown in (a). The unsafe reachable domain is eliminated by Veritex, but the safe reachable domain is barely changed, as shown in (b).

tional expensive. For the repair of the other 33 networks, Veritex is faster than ART-refinement.

The other case study is repairing an unsafe DNN agent for a rocket-lander system in DRL (roc). This agent has 9 state inputs, 5 hidden layers with each containing 20 ReLU neurons, and 3 outputs with a continuous action space. More details can be found in (Yang et al., 2022). The repair by Veritex takes 304.9 seconds to produce a provable safe agent. The reachability of the original agent and our repaired agent is shown in Fig. 4. Similar to the ACAS Xu network repair, Veritex repairs this unsafe agent without heavily affecting its original reachability. Overall, we can conclude that Veritex can efficiently repair unsafe DNNs on multiple safety properties with trivial impact on the original performance.

5. Conclusion

This paper presents a toolbox Veritex which provides a collection of algorithms for the reachability analysis and repair of DNNs. It contains three different set representations for the reachable-set computation. Its reachability analysis can be used for a sound and complete safety verification. Its high efficiency is demonstrated in the ACAS Xu benchmark. The analysis can also be used to compute the exact unsafe input-output reachable domain of DNNs for their repair. The repair algorithm supports the minimal repair and the non-minimal repair. Given an unsafe DNN, it can produce a provable safe version only with slight impacts on the original DNN. Its utility is demonstrated in the repair of unsafe ACAS Xu networks and an unsafe agent in DRL.



(a) Unsafe Agent (b) Our Repair

(c) Unsafe Agent (d) Our Repair

Figure 4. Reachability of the original agent and the repaired agent on Properties 1 & 2. The output reachable domains are projected on $(\mathbf{y}_0, \mathbf{y}_1)$ and $(\mathbf{y}_0, \mathbf{y}_2)$. Red area represents the unsafe reachable domain.

Acknowledgements

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 1910017, 1918450, and 2028001, the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-18-C-0089, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-22-1-0019. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of AFOSR, DARPA, or NSF.

References

alpha-beta-crown. https://github.com/huanzhang12/alpha-beta-CROWN.git.

Eran. https://github.com/eth-sri/eran.git.

Rocket-lander system. https://github.com/arex18/rocket-lander.git.

Verinet. https://github.com/vas-group-imperial/VeriNet.git.

Bak, S. nnenum: Verification of relu neural networks with optimized abstraction refinement. In *NASA Formal Methods Symposium*, pp. 19–36. Springer, 2021.

Bak, S., Liu, C., and Johnson, T. The second international verification of neural networks competition (vnn-comp 2021): Summary and results. *arXiv preprint arXiv:2109.00498*, 2021.

Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., and Misener, R. Efficient verification of relu-based neural networks via dependency analysis. In *AAAI*, pp. 3291–3299, 2020.

Dutta, S., Jha, S., Sankaranarayanan, S., and Tiwari, A. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*, pp. 121–138. Springer, 2018.

Henriksen, P. and Lomuscio, A. Efficient neural network verification via adaptive refinement and adversarial search. In *ECAI 2020*, pp. 2513–2520. IOS Press, 2020.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., et al. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, pp. 443–452. Springer, 2019.

Shriver, D., Elbaum, S., and Dwyer, M. B. Dnnv: A framework for deep neural network verification. In Silva, A. and Leino, K. R. M. (eds.), *Computer Aided Verification*, pp. 137–150, Cham, 2021. Springer International Publishing. ISBN 978-3-030-81685-8.

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):41, 2019.

Sotoudeh, M. and Thakur, A. V. Syrenn: A tool for analyzing deep neural networks. *Tools and Algorithms for the Construction and Analysis of Systems*, 12652:281, 2021.

Tran, H.-D., Musau, P., Lopez, D. M., Yang, X., Nguyen, L. V., Xiang, W., and Johnson, T. T. Star-based reachability analsysis for deep neural networks. In *23rd International Symposisum on Formal Methods (FM'19)*. Springer International Publishing, October 2019.

Tran, H.-D., Yang, X., Lopez, D. M., Musau, P., Nguyen, L. V., Xiang, W., Bak, S., and Johnson, T. T. Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International Conference on Computer Aided Verification*, pp. 3–17. Springer, 2020.

Wang, Z., Albarghouthi, A., and Jha, S. Abstract universal approximation for neural networks. *arXiv e-prints*, pp. arXiv–2007, 2020.

Yang, X., Johnson, T. T., Tran, H.-D., Yamaguchi, T., Hoxha, B., and Prokhorov, D. Reachability analysis of deep relu neural networks using facet-vertex incidence. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, HSCC '21, New York, NY, USA, 2021a. Association for Computing Machinery. ISBN 9781450383394. doi: 10.1145/3447928.3456650. URL https://doi.org/10.1145/3447928.3456650.

Yang, X., Yamaguchi, T., Tran, H.-D., Hoxha, B., Johnson, T. T., and Prokhorov, D. Reachability analysis of convolutional neural networks. *arXiv preprint arXiv:2106.12074*, 2021b.

Yang, X., Yamaguchi, T., Tran, H.-D., Hoxha, B., Johnson, T., and Prokhorov, D. Neural network repair with reachability analysis. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2022.