veriFIRE: Verifying an Industrial, Learning-Based Wildfire Detection System

Guy Amir[1(✉)], Ziv Freund[2], Guy Katz[1], Elad Mandelbaum[2], and Idan Refaeli[1]

[1] The Hebrew University of Jerusalem, Jerusalem, Israel
{guyam,guykatz,idan0610}@cs.huji.ac.il
[2] Elbit Systems—EW & SIGINT—Elisra Ltd., Holon, Israel
{ziv.freund,elad.mandelbaum}@elbitsystems.com

Abstract. In this short paper, we present our ongoing work on the veriFIRE project—a collaboration between industry and academia, aimed at using verification for increasing the reliability of a real-world, safety-critical system. The system we target is an airborne platform for wildfire detection, which incorporates two deep neural networks. We describe the system and its properties of interest, and discuss our attempts to verify the system's consistency, i.e., its ability to continue and correctly classify a given input, even if the wildfire it describes increases in intensity. We regard this work as a step towards the incorporation of academic-oriented verification tools into real-world systems of interest.

1 Introduction

In recent years, *deep neural networks* (DNNs) [16] have achieved unprecedented results in a variety of fields, such as image recognition [44], speech analysis [39], and many others [7,23,32,37,43]. This success has led to the integration of DNNs in various safety-critical systems [10].

A particular safety-critical application of DNNs is within *wildfire detection* systems [31,34,42,51], whose goal is to detect and alert first responders to situations that could later become life threatening. One such airborne system, which is currently being considered by Elbit Systems for use on aerial vehicles, is based on Infra-Red (IR) sensors that feed their inputs, usually a series of image frames, to multiple neural networks—which then determine whether the images contain a wildfire. Naturally, it is possible that (a) the system will mistakenly issue an alert when a wildfire does not exist, or, worse, that (b) the system will fail to issue an alert when the images do indicate the existence of a wildfire. The second kind of failure is clearly very dangerous, and could potentially jeopardize human lives. Consequently, potential users of the system require it to be extremely reliable.

Although DNN-based systems are highly successful, prior research has shown that even complex and highly-accurate DNNs are prone to errors. For example, small input perturbations, due to either random noise or adversarial attacks,

All authors contributed equally.

M. Chechik et al. (Eds.): FM 2023, LNCS 14000, pp. 648–656, 2023.
https://doi.org/10.1007/978-3-031-27481-7_38

are known to cause modern DNNs to fail miserably [17,30,38]. Such issues raise serious concerns regarding the trustworthiness of a DNN-based wildfire detection system, and could delay or prevent its deployment.

In order to address such issues and facilitate the certification of DNNs, the formal methods community has recently suggested various tools and approaches for *formally verifying* the correctness of DNNs [5,11,15,19,21,22,24,25,27,35,36,45,47,49,50], based on reachability analysis and abstract interpretation [15,35,46], SMT-solving [3,12,18,19,24,26,29], and other methods. Given a DNN and a specification, these techniques allow us to formally prove that the DNN satisfies the specification for *any* possible input of interest (see Appendix A for additional details). However, despite the rapid improvement in DNN verification technology, there remains a gap between the capabilities of verification tools developed by academia, and the actual needs of industrial teams. First, academic tools often face scalability issues, and may be unsuitable for verifying industrial-sized DNNs with millions of neurons. Second, academic-oriented verification tools may not support the various DNN specifications used in industry. Consequently, practitioners often resort to using various forms of testing, and not verification, when attempting to certify real-world DNNs.

In this paper, we describe our ongoing work on the *veriFIRE* project—a collaboration between Elbit Systems and the Hebrew University, aimed at formally verifying the correctness of the aforementioned wildfire detection system. As part of this project, our goals are to (1) produce formal specifications for this system, which could then be formulated into DNN verification tools; and (2) enhance and extend existing verification technology, so that it can be successfully applied to this system.

2 The VeriFIRE Project

The Platform. The veriFIRE project is a recent and ongoing collaboration between Elbit Systems and the Hebrew University. It involves an airborne wildfire detection system, designed to be mounted on aerial vehicles (AVs)—from small drones, to large manned or unmanned aircraft—being manufactured by Elbit Systems (see Fig. 1). The airborne system consists of the following components: (i) a set of infra-red (IR) sensors, located at different spots on the AV, and pointing at different angles. These sensors produce temporal image streams of the *background* surrounding the AV; (ii) a first, convolutional DNN, which receives the image streams generated by the IR sensors, and produces candidate detections, based on temporal changes as detected when compared to previous images of the background. Each candidate detection is a stream of slices (through time) taken from the background image streams, around the suspicious areas; and (iii) a second convolutional DNN, which receives a candidate detection, produced by the first DNN, and determines whether it is a wildfire (at its early stages), or a false detection of the first DNN. The goal of the veriFIRE project is to ensure the overall reliability of the system, by verifying the correctness of its DNN components.

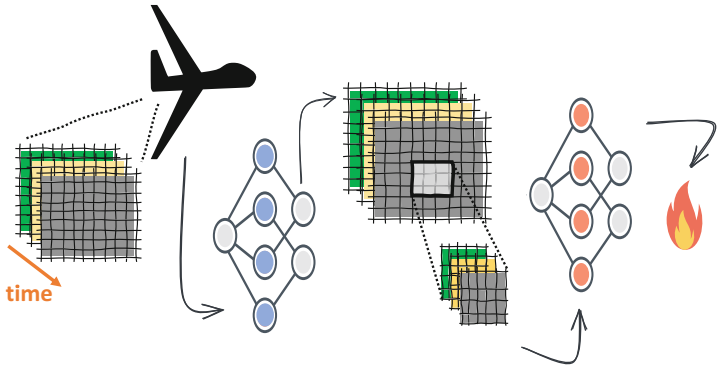

Fig. 1. A scheme of the airborne wildfire detection system. At first, an airborne platform takes multiple IR images, and uses the first DNN to detect candidate areas, in which a wildfire is suspected. Next, these candidates are passed to a second DNN, which determines whether a wildfire has truly occurred, or not.

Training the wildfire detection platform is performed using a proprietary simulator that automatically generates synthetic images, by adding simulated wildfire images to recorded background images. Given two datasets, one containing only normalized wildfire signals ($\mathcal{S}$) with no background, and another for background images ($\mathcal{B}$) which do not contain any wildfires, the simulator creates a new dataset of synthetic images, each one generated by combining a wildfire image with a background image, in a process referred to as *planting*. More formally, for any $x_s \in \mathcal{S}$, $x_b \in \mathcal{B}$, the simulator uses a planting function p to produce a realistic image $I = p(\epsilon \cdot x_s, x_b)$, which contains the wildfire with intensity ϵ. At its early stages, a wildfire is a sub-pixel in the sensor's field of view, and thus the planting function can be treated as a linear combination of the wildfire image and the background image. We note that this methodology is common practice, and is acceptable to Elbit Systems' clients.

Although the dataset is large enough to produce sufficiently many test samples, statistical testing alone is inadequate for guaranteeing the platform's reliability. Specifically, clients may wish to guarantee that some performance features are not random—for example, it is required that if a small wildfire is detected by the platform in a given scenario, a stronger wildfire will definitely be detected as well. Thus, we began by focusing on formally verifying the correctness of the second DNN used, which we term N. This network can be regarded as a mapping $N : \mathbb{R}^{n \times k} \rightarrow \mathbb{R}$, where n is the number of pixels in each image, and k is the number of time-steps observed. When presented with a stream of input images $x \in \mathbb{R}^{n \times k}$, N computes a score, $N(x)$; and if this score exceeds a threshold δ, then N classifies x as an image containing a wildfire. The value of δ is determined according to the clients' needs, as a balancing point between the empirical false-alarm rate and its tradeoff with the empirical positive-detection rate, after a short evaluation period. The network N is comprised of three convolution

layers [28,44], each one followed by a max-pooling layer and two fully-connected layers. In the last layer, the network has a single output node with a sigmoid activation, which serves as the output of the entire DNN.

Consistency. One main challenge in the veriFIRE project is to produce formal specifications for N. Ideally, we would like to prove that N correctly identifies any possible wildfire within any possible image, but this is difficult to formulate rigorously. Current state-of-the-art verification tools focus primarily on verifying local adversarial robustness [8,15,18,33,36,40,46,48], i.e., on proving that a DNN continues to correctly classify an input in the presence of slight perturbations; but we have observed that this kind of property is of limited interest to potential clients of the system. Thus, a new kind of specification is required for this process. With that in mind, we introduce the definition for *local consistency*:

Definition 1 (Local Consistency). *Given a deep neural network $N : \mathbb{R}^{n\times k} \rightarrow \mathbb{R}$, a wildfire signal image stream $x_s \in \mathcal{S}$, and an input background image stream $x_b \in \mathcal{B}$, we say that N is (x_s, x_b)-locally-consistent if for every $\epsilon_1 \geq \epsilon_2$, it holds that $N(p(\epsilon_1 \cdot x_s, x_b)) \geq N(p(\epsilon_2 \cdot x_s, x_b))$, where $p : \mathbb{R}^{n\times k} \times \mathbb{R}^{n\times k} \rightarrow \mathbb{R}^{n\times k}$ is a planting function, such that $p(s, b)$ plants the signal s into the background b.*

Intuitively, local consistency in this context means that if the original image x was determined to contain a wildfire (i.e., $N(x)$ exceeded the threshold δ), then any image stream with a *stronger signal*, e.g., a larger wildfire, will also be determined to contain a wildfire. If this property holds, then there is a specific wildfire magnitude threshold, above which the system will be reliable. For our purposes, we use the linear planting function: $p(s, b) = s + b$, as a good approximation to the full generation function, as it approximately represents real wildfire signals at their early stages on the background images.

The above definition only considers a single pair of a signal image stream and a background image stream. Ideally, we would like to verify consistency for *all* possible background images containing wildfires. Thus, we define *global* consistency, as follows:

Definition 2 (Global Consistency). *Given a deep neural network $N : \mathbb{R}^{n\times k} \rightarrow \mathbb{R}$, we say that N is globally-consistent if for every $x_s \in \mathcal{S}$ and $x_b \in \mathcal{B}$, N is (x_s, x_b)-locally-consistent.*

We note that the sets $\mathcal{S}$ and $\mathcal{B}$ are not necessarily finite, and may represent *all* possible wildfire signal images and *all* possible background images, respectively. Thus, global consistency is significantly more complex to prove than local consistency.

3 Conclusion and Remaining Challenges

This paper presents a collaboration between academia and industry, with the goal of verifying an airborne system for wildfire detection. Our work so far has

focused on devising novel kinds of specifications of interest, which are better suited for this domain than the specifications commonly supported by academia-oriented verification tools. Moving forward, we plan to formulate such properties for the remaining parts of the system, and also to enhance existing verification engines so that they become sufficiently expressive and scalable to tackle the networks in question.

Acknowledgement. This work was supported by a grant from the Israel Innovation Authority. The work of Amir was also supported by a scholarship from the Clore Israel Foundation.

Appendices

A Background: DNNs and Their Verification

Deep Neural Networks. A deep neural network (DNN) [16] is a computational, directed graph, comprised of layers. The network computes a value, by receiving inputs and propagating them through its layers until reaching the final (output) layer. These output values can be interpreted as a classification label or as a regression value, depending on the kind of network in question. The actual computation depends on each layer's *type*. For example, a node y in a *rectified linear unit* (*ReLU*) layer calculates the value $y = \text{ReLU}(x) = \max(0, x)$, for the value x of one of the nodes in its preceding layer. Additional layer types include weighted sum layers, as well as layers with various non-linear activations. Here, we focus on *feed-forward* neural networks, i.e., DNNs in which each layer is connected only to its following layer.

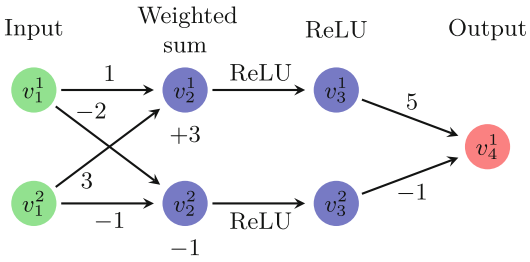

Fig. 2. A toy DNN.

Figure 2 depicts a toy DNN. For input $V_1 = [1, 3]^T$, the second layer computes the values $V_2 = [13, -6]^T$. In the third layer, the ReLU functions are applied, producing $V_3 = [13, 0]^T$. Finally, the network's single output value is $V_4 = [65]$.

DNN Verification. A DNN verification engine [15,19,24,36,47] receives a DNN N, a precondition P that defines a subspace of the network's inputs, and a post-condition Q that limits the network's output values. The verification engine then

searches for an input x_0 that satisfies $P(x_0) \wedge Q(N(x_0))$. If such an input exists, the engine returns `SAT` and a concrete input that satisfies the constraints; otherwise, it returns `UNSAT`, indicating that no such input exists. The postcondition Q usually encodes the *negation* of the desired property, and hence a `SAT` answer indicates that the property is violated, and that the returned x_0 triggers a bug. However, an `UNSAT` result indicates that the property holds.

For example, suppose we wish to verify that the simple DNN depicted in Fig. 2 always outputs a value strictly larger than 25; i.e., for any input $x = \langle v_1^1, v_1^2 \rangle$, it holds that $N(x) = v_4^1 > 25$. This property is encoded as a verification query by choosing a precondition that does not restrict the input, i.e., $P = (true)$, and by setting a postcondition $Q = (v_4^1 \leq 25)$. For this verification query, a sound verification engine will return `SAT`, alongside a feasible counterexample such as $x = \langle 1, 0 \rangle$, which produces $v_4^1 = 20 \leq 25$, proving that the property does not hold for this DNN.

In our work, we used *Marabou* [26]—a sound and complete DNN-verification engine, which has recently been used in a variety of applications [1,2,4,6,9,13,14,20,40,41].

References

1. Amir, G., et al.: Verifying Learning-Based Robotic Navigation Systems (2022). Technical report. https://arxiv.org/abs/2205.13536
2. Amir, G., Schapira, M., Katz, G.: Towards scalable verification of deep reinforcement learning. In: Proceedings 21st International Conference on Formal Methods in Computer-Aided Design (FMCAD), pp. 193–203 (2021)
3. Amir, G., Wu, H., Barrett, C., Katz, G.: An SMT-based approach for verifying binarized neural networks. In: TACAS 2021. LNCS, vol. 12652, pp. 203–222. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72013-1_11
4. Amir, G., Zelazny, T., Katz, G., Schapira, M.: Verification-aided deep ensemble selection. In: Proceedings of the 22nd International Conference on Formal Methods in Computer-Aided Design (FMCAD), pp. 27–37 (2022)
5. Baluta, T., Shen, S., Shinde, S., Meel, K.S., Saxena, P.: Quantitative verification of neural networks and its security applications. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 1249–1264 (2019)
6. Bassan, S., Katz, G.: Towards Formal Approximated Minimal Explanations of Neural Networks, Technical report (2022). https://arxiv.org/abs/2210.13915
7. Bojarski, M., et al.: End to End Learning for Self-Driving Cars, Technical report (2016). http://arxiv.org/abs/1604.07316
8. Casadio, M., et al.: Neural network robustness as a verification property: a principled case study. In: Shoham, S., Vizel, Y. (eds.) CAV 2022. Lecture Notes in Computer Science, vol. 13371, pp. 219–231. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-13185-1_11
9. Corsi, D., Yerushalmi, R., Amir, G., Farinelli, A., Harel, D., Katz, G.: Constrained Reinforcement Learning for Robotics via Scenario-Based Programming, Technical report (2022). https://arxiv.org/abs/2206.09603
10. Dong, S., Wang, P., Abbas, K.: A survey on deep learning and its applications. Comput. Sci. Rev. **40**, 100379 (2021)

11. Dutta, S., Jha, S., Sankaranarayanan, S., Tiwari, A.: Output range analysis for deep feedforward neural networks. In: Dutle, A., Muñoz, C., Narkawicz, A. (eds.) NFM 2018. LNCS, vol. 10811, pp. 121–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77935-5_9
12. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: D'Souza, D., Narayan Kumar, K. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 269–286. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_19
13. Elboher, Y.Y., Cohen, E., Katz, G.: Neural network verification using residual reasoning. In: chlingloff, B.H., Chai, M. (eds.) Software Engineering and Formal Methods. SEFM 2022. Lecture Notes in Computer Science, vol. 13550, pp. 173–189. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17108-6_11
14. Elboher, Y.Y., Gottschlich, J., Katz, G.: An abstraction-based framework for neural network verification. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12224, pp. 43–65. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53288-8_3
15. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: AI2: safety and robustness certification of neural networks with abstract interpretation. In: Proceedings 39th IEEE Symposium on Security and Privacy (S&P) (2018)
16. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
17. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and Harnessing Adversarial Examples, Technical report (2014). http://arxiv.org/abs/1412.6572
18. Gopinath, D., Katz, G., Păsăreanu, C.S., Barrett, C.: DeepSafe: a data-driven approach for assessing robustness of neural networks. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 3–19. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_1
19. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 3–29. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_1
20. Isac, O., Barrett, C., Zhang, M., Katz, G.: Neural network verification with proof production. In: Proceedings of the 22nd International Conference on Formal Methods in Computer-Aided Design (FMCAD), pp. 38–48 (2022)
21. Ivanov, R., Carpenter, T.J., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verifying the safety of autonomous systems with neural network controllers. ACM Trans. Embedded Comput. Syst. (TECS) **20**(1), 1–26 (2020)
22. Jin, P., Tian, J., Zhi, D., Wen, X., Zhang, M.: Trainify: A CEGAR-driven training and verification framework for safe deep reinforcement learning. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification (CAV), CAV 2022. Lecture Notes in Computer Science, vol. 13371, pp. 193–218. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-13185-1_10
23. Jumper, J., et al.: Highly accurate protein structure prediction with AlphaFold. Nature **596**(7873), 583–589 (2021)
24. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
25. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: a calculus for reasoning about deep neural networks. Formal Methods Syst. Des., 1–30 (2021). https://doi.org/10.1007/s10703-021-00363-7

26. Katz, G., et al.: The marabou framework for verification and analysis of deep neural networks. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 443–452. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_26
27. Könighofer, B., Lorber, F., Jansen, N., Bloem, R.: Shield synthesis for reinforcement learning. In: Margaria, T., Steffen, B. (eds.) ISoLA 2020. LNCS, vol. 12476, pp. 290–306. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61362-4_16
28. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of 26th Conference on Neural Information Processing Systems (NeurIPS), pp. 1097–1105 (2012)
29. Kuper, L., Katz, G., Gottschlich, J., Julian, K., Barrett, C., Kochenderfer, M.: Toward Scalable Verification for Safety-Critical Deep Networks, Technical report (2018). https://arxiv.org/abs/1801.05950
30. Kurakin, A., Goodfellow, I.J., Bengio, S: Adversarial examples in the physical world. In: Artificial Intelligence Safety and Security, pp. 99–112 (2018)
31. Lee, W., Kim, S., Lee, Y.T., Lee, H.W., Choi, M.: Deep neural networks for wild fire detection with unmanned aerial vehicle. In: Proceedings of 2017 IEEE International Conference on Consumer Electronics (ICCE), pp. 252–253 (2017)
32. Lekharu, A., Moulii, K. Y., Sur, A., Sarkar, A.: Deep learning based prediction model for adaptive video streaming. In: Proceedings of International Conference on Communication Systems & Networks (COMSNETS), pp. 152–159 (2020)
33. Levy, N., Katz, G.: RoMA: a Method for Neural Network Robustness Measurement and Assessment, Technical report (2021). https://arxiv.org/abs/2110.11088
34. Li, P., Zhao, W.: Image fire detection algorithms based on convolutional neural networks. Case Stud. Therm. Eng. **19**, 100625 (2020)
35. Lomuscio, A., Maganti, L.: An Approach to Reachability Analysis for Feed-Forward ReLU Neural Networks, Technical report (2017). http://arxiv.org/abs/1706.07351
36. Lyu, Z., Ko, C. Y., Kong, Z., Wong, N., Lin, D., Daniel, L.: Fastened crown: tightened neural network robustness certificates. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI), pp. 5037–5044 (2020)
37. Mnih, V., et al.: Playing Atari with Deep Reinforcement Learning. Technical report (2013). http://arxiv.org/abs/1312.5602
38. Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1765–1773 (2017)
39. Nassif, A., Shahin, I., Attili, I., Azzeh, M., Shaalan, K.: Speech recognition using deep neural networks: a systematic review. IEEE Access **7**, 19143–19165 (2019)
40. Ostrovsky, M., Barrett, C., Katz, G.: An abstraction-refinement approach to verifying convolutional neural networks. In: Bouajjani, A., Holík, L., Wu, Z. (eds.) Automated Technology for Verification and Analysis. ATVA 2022. Lecture Notes in Computer Science, vol. 13505, pp. 391–396 (2022). https://doi.org/10.1007/978-3-031-19992-9_25
41. Refaeli, I., Katz, G.: Minimal multi-layer modifications of deep neural networks. In: Isac, O., Ivanov, R., Katz, G., Narodytska, N., Nenzi, L. (eds.) Software Verification and Formal Methods for ML-Enabled Autonomous Systems. NSV (FoMLAS) 2022. Lecture Notes in Computer Science, vol. 13466, pp. 46–66. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-21222-2_4
42. Sharma, J., Granmo, O.-C., Goodwin, M., Fidje, J.T.: Deep convolutional neural networks for fire detection in images. In: Boracchi, G., Iliadis, L., Jayne, C., Likas, A. (eds.) EANN 2017. CCIS, vol. 744, pp. 183–193. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65172-9_16

43. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
44. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition, Technical report (2014). http://arxiv.org/abs/1409.1556
45. Strong, C.A., et al.: Global optimization of objective functions represented by ReLU networks. J. Mach. Learn., 1–28 (2021). https://doi.org/10.1007/s10994-021-06050-2
46. Tjeng, V., Xiao, K., Tedrake, R.: Evaluating Robustness of Neural Networks with Mixed Integer Programming, Technical report (2017). http://arxiv.org/abs/1711.07356
47. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: Proceedings of the 27th USENIX Security Symposium, pp. 1599–1614 (2018)
48. Weng, T.: Towards Fast Computation of Certified Robustness for ReLU Networks, Technical report (2018). http://arxiv.org/abs/1804.09699
49. Zelazny, T., Wu, H., Barrett, C., Katz, G.: On reducing over-approximation errors for neural network verification. In: Proceedings of the 22nd International Conference on Formal Methods in Computer-Aided Design (FMCAD), pp. 17–26 (2022)
50. Zhang, H., Shinn, M., Gupta, A., Gurfinkel, A., Le, N., Narodytska, N.: Verification of recurrent neural networks for cognitive tasks via reachability analysis. In: Proceedings of the 24th European Conference on Artificial Intelligence (ECAI), pp. 1690–1697 (2020)
51. Zhang, Q., Xu, J., Xu, L., Guo, H.: Deep convolutional neural networks for forest fire detection. In: Proceedings of the International Forum on Management, Education and Information Technology Application (IFMEITA), pp. 568–575 (2016)