LUND UNIVERSITY

SafeDeep: A Scalable Robustness Verification Framework for Deep Neural Networks

Baninajjar, Anahita; Hosseini, Kamran; Rezine, Ahmed; Aminifar, Amir

Published in:
2023 IEEE International Conference on Acoustics, Speech and Signal Processing

2023

Document Version:
Peer reviewed version (aka post-print)

Link to publication

Citation for published version (APA):
Baninajjar, A., Hosseini, K., Rezine, A., & Aminifar, A. (Accepted/In press). SafeDeep: A Scalable Robustness Verification Framework for Deep Neural Networks. In *2023 IEEE International Conference on Acoustics, Speech and Signal Processing*

Total number of authors:
4

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Download date: 17. May. 2023

SafeDeep: A Scalable Robustness Verification Framework for Deep Neural Networks

Anahita Baninajjar*, Kamran Hosseini†, Ahmed Rezine†, and Amir Aminifar*
*Department of Electrical and Information Technology, Lund University, Lund, Sweden
†Department of Computer and Information Science, Linköping University, Linköping, Sweden
Email: anahita.baninajjar@eit.lth.se, kamran.hosseini@liu.se, ahmed.rezine@liu.se, amir.aminifar@eit.lth.se

***Abstract*—The state-of-the-art machine learning techniques come with limited, if at all any, formal correctness guarantees. This has been demonstrated by adversarial examples in the deep learning domain. To address this challenge, here, we propose a scalable robustness verification framework for Deep Neural Networks (DNNs). The framework relies on Linear Programming (LP) engines and builds on decades of advances in the field for analyzing convex approximations of the original network. The key insight is in the on-demand incremental refinement of these convex approximations. This refinement can be parallelized, making the framework even more scalable. We have implemented a prototype tool to verify the robustness of a large number of DNNs in epileptic seizure detection. We have compared the results with those obtained by two state-of-the-art tools for the verification of DNNs. We show that our framework is consistently more precise than the over-approximation-based tool ERAN and more scalable than the SMT-based tool Reluplex.**

***Index Terms*—DNNs, verification, approximation, refinement, linear programming, robustness**

I. Introduction

The current difficulties in providing formal correctness guarantees for the behaviours of DNNs are a fundamental barrier for their adoption in safety-critical systems, e.g., cyber-physical systems or smart medical devices [1]. Examples of such machine-learning-based devices include implantable and wearable devices to detect heart-attacks [2] or epileptic seizures [3]. These systems continuously monitor the patients and inform emergency units for rescue in case of such adverse events. Therefore, it is, for instance, unacceptable for the epileptic seizures detection/classification algorithms used for real-time monitoring to change their classification decision from "seizure" to "typical or non-seizure" after minor and "irrelevant" changes. Such safety requirements led researchers to explore ways to generate adversarial examples, i.e., examples falsifying the desired robustness property [4]–[7]. The existence of such examples proves the possibility of violating the safety property.

The state-of-the-art techniques for formally establishing safety properties (e.g., robustness) for DNNs are either exact (i.e., establish correctness iff it holds) [8]–[10] or based on over-approximation [11]–[16] and/or simplified models (e.g., linear models) [17], [18]. In general, exact methods have difficulties to scale as they explore all possible behaviours, i.e., generally exponential with the size of the model [10]. On the other hand, over-approximation-based approaches scale better at the price of precision [12]. For instance, abstract-interpretation-based methods use abstract domains to over-approximate the possible values and speed-up the verification process [11]. This results in a faster and more scalable verification approach but may generate false positives, i.e., cases where the approach concludes the safety property may be violated when it actually holds. In general, gaining more precision for such approaches requires changing the abstract domain to perform the verification from scratch.

This work is partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation and by the European Union (EU) Interreg Program.

In this paper, we propose SafeDeep, a scalable robustness verification framework for deep neural networks. The key insight in SafeDeep is in the on-demand incremental refinement of the convex approximations. Our proposed framework leverages on over-approximation for verification of the robustness property. Unlike the abstract-interpretation-based techniques, rather than restarting the verification process from scratch upon inconclusive results, each time the obtained convex over-approximation is not sufficient to establish the property, we incrementally refine the underlying convex approximation. The refinement process can be parallelized allowing for a scalable usage of the decades of advances in LP. We have conducted several experiments using DNNs with 760 neurons each and trained them for the classification of epileptic seizures in real-time setting using the e-Glass wearable system [3]. Our experiments on several hundreds formulations of the robustness property for these DNNs show that our framework could verify more formulations than the two established tools, i.e., ERAN [13], which is based on over-approximation, and the Reluplex [10], which is based on Satisfiability Modulo Theories (SMT).

II. SafeDeep

In this section, we propose a novel framework to verify the robustness of neural networks. We will explain our framework step by step using a running example.

A. Deep Neural Network Model

A neural network is a function composed of a sequence of linear transformations and non-linear activation functions to compute outputs based on trained weights and biases. Assume a neural network with N layers such that n_i is the number of neurons in layer i for $i \in \{1, \dots N\}$. The network takes

as input a vector $\mathbf{x}^{(1)} \in \mathbb{R}^{n_1}$. Let $f(.) : R^{n_{k-1}} \rightarrow R^{n_k}$ be a nonlinear activation function from layer $k-1$ to layer k. The relation between layer $k-1$ and layer k is defined as $\mathbf{x}^{(k)} = f(\mathbf{W}^{(k)}\mathbf{x}^{(k-1)} + \mathbf{b}^{(k)})$, for all $k \in \{2, ..., N\}$, where $\mathbf{W}^{(k)} \in R^{n_k \times n_{k-1}}$ and $\mathbf{b}^{(k)} \in R^{n_k}$ are matrices of weights and vectors of biases, respectively. Note that $x_i^{(k)}$ shows the value of the i^{th} neuron of the k^{th} layer.

Fig. 1 depicts a fully-connected feed-forward neural network with two neurons in each hidden layer and Rectified Linear Unit (ReLU) activation functions. This network is a binary classifier that classifies input $\mathbf{x}^{(1)}$ into Class 0, if $x_1^{(4)} > x_2^{(4)}$ holds, and Class 1 otherwise. The network is trained and its weights and biases are shown in Fig. 1, on the arrows and above/below each neuron, respectively. In this example, the nonlinear activation function f is the ReLU function resulting in $x_i^{(k)} = f(\hat{x}_i^{(k)})$ for layer k. Each neuron in the hidden layers is split in two, shown with pink dashed boxes, to illustrate the relations between neurons before and after applying the ReLU activation function. For the neural network of Fig. 1, starting from the input $x_1^{(1)} = 2$, $x_2^{(1)} = 0$ and propagating through the network, we obtain $x_1^{(4)} = -9$ and $x_2^{(4)} = 0$ as output, which means the input belongs to Class 1, because $x_1^{(4)} < x_2^{(4)}$.

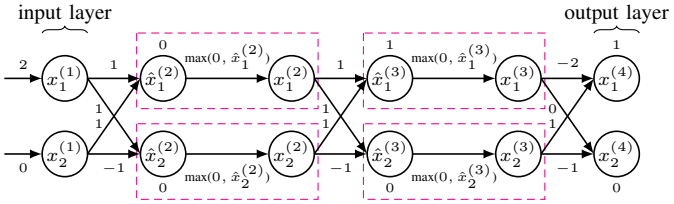


Fig. 1: A four-layer fully-connected feed-forward neural network with ReLU activation functions

B. Robustness Verification

Now, we define *robustness* of a classifier at input $\tilde{\mathbf{x}}^{(1)} \in \mathbb{R}^{n_1}$ w.r.t. a bound δ. Let c be the class associated to $\tilde{\mathbf{x}}^{(1)}$. A box around $\tilde{\mathbf{x}}^{(1)}$ contains all inputs $\mathbf{x}^{(1)} \in \mathbb{R}^{n_1}$ satisfying:

$$\|\mathbf{x}^{(1)} - \tilde{\mathbf{x}}^{(1)}\|_\infty \leq \delta. \tag{1}$$

Such an input $\mathbf{x}^{(1)}$ is said to be a perturbed input. The neural network is said to be robust at $\tilde{\mathbf{x}}^{(1)}$ against perturbation δ if, considering all outputs $\mathbf{x}^{(N)}$ corresponding to all possible perturbed inputs $\mathbf{x}^{(1)}$ satisfying Inequality (1), we have that:

$$\min\left(x_c^{(N)} - x_i^{(N)}\right) > 0, \qquad \forall i \in \{1, ..., n_N\}, i \neq c. \tag{2}$$

In other words, applying any perturbations of at most δ around $\tilde{\mathbf{x}}^{(1)}$ does not change the class of the outputs.

Next, we explain how to bound the value of $x_c^{(N)} - x_i^{(N)}$ for perturbed inputs. Towards this, we obtain the lower and upper bounds for each neuron $x_i^{(k)}$ (i.e., i^{th} neuron of the k^{th} layer), which are captured by $\underline{x}_i^{(k)}$ and $\overline{x}_i^{(k)}$, respectively. For each neuron of the input/first layer, we have $\underline{x}_i^{(1)} = \tilde{x}_i^{(1)} - \delta$ and $\overline{x}_i^{(1)} = \tilde{x}_i^{(1)} + \delta$, for all $i \in \{1, 2, ..., n_1\}$. The lower and upper bounds of neurons in other layers can be obtained as follows:

$$\underline{x}_i^{(k)} = \sum_{j:W_{j,i}^{(k)}>0} W_{j,i}^{(k)} \cdot \underline{x}_j^{(k-1)} + \sum_{j:W_{j,i}^{(k)}<0} W_{j,i}^{(k)} \cdot \overline{x}_j^{(k-1)} + b_i^{(k)},$$

$$\overline{x}_i^{(k)} = \sum_{j:W_{j,i}^{(k)}>0} W_{j,i}^{(k)} \cdot \overline{x}_j^{(k-1)} + \sum_{j:W_{j,i}^{(k)}<0} W_{j,i}^{(k)} \cdot \underline{x}_j^{(k-1)} + b_i^{(k)}, \tag{3}$$

where $i \in \{1, 2, ..., n_k\}$ and $j \in \{1, 2, ..., n_{k-1}\}$.

Let us reconsider the example in Fig. 1 and suppose we would like to check whether the network is robust at input $\tilde{x}_1^{(1)} = 2$ and $\tilde{x}_2^{(1)} = 0$ w.r.t. a perturbation $\delta = 0.5$. Recall input $\tilde{\mathbf{x}}^{(1)}$ belongs to Class 1. The values $x_1^{(1)}$ and $x_2^{(1)}$ of any perturbed input freely range in $[1.5, 2.5]$ and $[-0.5, 0.5]$, respectively. If the neural network is robust against perturbation $\delta = 0.5$ at $\tilde{\mathbf{x}}^{(1)}$, then $x_1^{(4)} < x_2^{(4)}$ holds for all $(x_1^{(1)}, x_2^{(1)}) \in [1.5, 2.5] \times [-0.5, 0.5]$, which means all inputs belong to the Class 1.

Using weights and biases of the first layer, it is deduced that $\hat{x}_1^{(2)}$ and $\hat{x}_2^{(2)}$ both range in $[1, 3]$. As the range of $\hat{x}_1^{(2)}$ and $\hat{x}_2^{(2)}$ is positive, we have the same range for $x_1^{(2)}$ and $x_2^{(2)}$. By propagating further these individual bounds along the layers, we obtain that $\hat{x}_1^{(3)} \in [3, 7]$ and $\hat{x}_2^{(3)} \in [-2, 2]$. Then, after applying the ReLU function we have $x_1^{(3)} \in [3, 7]$ and $x_2^{(3)} \in [0, 2]$. In the output/last layer, we have $x_1^{(4)} \in [-13, -3]$ and $x_2^{(4)} \in [-2, 0]$. Hence, $\min\left(x_2^{(4)} - x_1^{(4)}\right) \geq 1$, which is a sufficient lower bound on the difference of neurons of the output/last layer of perturbed inputs to ensure that $x_2^{(4)}$ is always larger than $x_1^{(4)}$. This means that all inputs in $[1.5, 2.5] \times [-0.5, 0.5]$ belong to Class 1, and the neural network is robust against perturbation $\delta = 0.5$ with the input $x_1^{(1)} = 2$ and $x_2^{(1)} = 0$. However, this is not always the case and we may need to refine the lower and upper bounds further to be able to establish robustness. This is introduced in the following section.

C. Linear Approximations

Compared to the bounds obtained by simple propagation of the individual bounds along the network, as described in the previous section, the bounds obtained by enumerating all active/inactive choices are tighter because simple bound propagation ignores the relations between variables. Indeed, tighter bounds on outputs $\mathbf{x}^{(N)} \in \mathbb{R}^{n_N}$ are obtained by considering all active/inactive choices of ReLU functions. However, the number of such decisions is computationally intractable, potentially exponential in the number of ReLU functions. To address this issue, here, we apply a linear optimizer on the neurons' bounds to investigate the robustness of a neural network for perturbed inputs $\mathbf{x}^{(1)} \in \mathbb{R}^{n_1}$ around input $\tilde{\mathbf{x}}^{(1)}$ and perturbation δ. We use linear optimizers to avoid the expensive enumeration of all active/inactive choices while tracking the relations among variables.

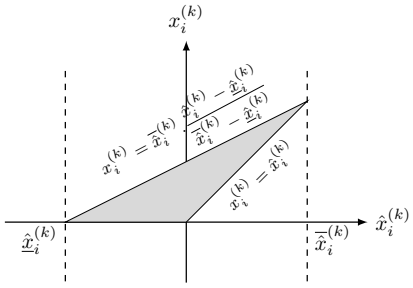


Fig. 2: Minimum area convex approximation for ReLU.

Practically, to calculate the lower and upper bounds of neurons using linear optimization tools, we need to express the ReLU activation function, which is a nonlinear function with linear constraints. Consider $\underline{\hat{x}}_i^{(k)}$ and $\overline{\hat{x}}_i^{(k)}$ as the lower and upper bounds of the neuron before applying ReLU activation function; Applying $ReLU(x) = \max(0, x)$ leads to one of the following states: The neuron is (i) always active, if $\underline{\hat{x}}_i^{(k)} \geq 0$, then $\underline{x}_i^{(k)} = \underline{\hat{x}}_i^{(k)}$ and $\overline{x}_i^{(k)} = \overline{\hat{x}}_i^{(k)}$, (ii) always inactive, if $\overline{\hat{x}}_i^{(k)} \leq 0$, then $\underline{x}_i^{(k)} = \overline{x}_i^{(k)} = 0$, and (iii) *undetermined*, if $\underline{\hat{x}}_i^{(k)} < 0$ and $\overline{\hat{x}}_i^{(k)} > 0$, then the value of the neuron after applying ReLU cannot be exactly captured. Here, we over-approximate the undetermined neurons, because active/inactive choices cannot be ruled out. We use a convex approximation with the minimum area, which is the tightest possible linear approximation for ReLUs, as described in Fig. 2. Thus, when the state of a neuron is undetermined, we introduce the following constraints to model the tightest possible convex approximation for the neuron $\hat{x}_i^{(k)}$ and its ReLU image $x_i^{(k)}$:

$$x_i^{(k)} \leq \overline{\hat{x}}_i^{(k)} . \frac{\hat{x}_i^{(k)} - \underline{\hat{x}}_i^{(k)}}{\overline{\hat{x}}_i^{(k)} - \underline{\hat{x}}_i^{(k)}}, \qquad x_i^{(k)} \geq \hat{x}_i^{(k)}, \qquad x_i^{(k)} \geq 0.$$

Now, we have the lower and upper bounds of all neurons, and we would like to proceed with the robustness of the neural network. To this end, using linear optimization, we investigate whether Inequality (2) holds. If Inequality (2) holds, it is proved that the neural network is robust against all possible perturbations within range δ, when the input is $\tilde{\mathbf{x}}^{(1)} \in \mathbb{R}^{n_1}$ as shown in Inequality (1).

In our example, by increasing perturbation to $\delta = 1$, inputs' ranges change to $x_1^{(1)} \in [1, 3]$ and $x_2^{(1)} \in [-1, 1]$. Then, by propagating through the neural network, we obtain $x_1^{(2)}$ and $x_2^{(2)}$ both range in $[0, 4]$, $x_1^{(3)} \in [1, 9]$, $x_2^{(3)} \in [-4, 4]$, $x_1^{(4)} \in [-17, 3]$, and $x_2^{(4)} \in [-4, 0]$. As $\min(x_2^{(4)} - x_1^{(4)}) \geq 0$ does not hold, the robustness of the network is not guaranteed. We apply a linear optimizer and discover that $\min(x_2^{(4)} - x_1^{(4)}) \geq -1$ on the neural network, which is tighter, but still not sufficient to establish robustness in this example.

D. Refining Bounds

In this section, we further refine the lower and upper bounds of the neurons to reduce the over-approximation and obtain tighter bounds. To avoid unnecessary computations, we do not refine all the neurons since there is no over-approximation for neurons that have already been shown to be in state (i) always active or state (ii) always inactive. Only undetermined neurons need to be over-approximated and targeted by the refinements.

Algorithm 1: Robustness Verification

Input: $\mathbf{W}$, $\mathbf{b}$, and $\mathbf{x}^{(1)}$ of a N-layer neural network
Result: Robustness status

```
1  x^(k) = f(W^(k) x^(k-1) + b^(k)), ∀k ∈ {1, ..., N}
2  for layer l in layers ∈ [3, N − 1] do
3      for neurons in layer l do
4          if state of the neuron is undetermined then
5              Refine bounds using linear optimizers;
6          end
7      end
8      Update neurons' bounds of the next layers;
9      if min(x_c^(N) − x_i^(N)) > 0, ∀i ≠ c in layer N then
10         Return Robustness is verified.
11     end
12 end
13 Return Robustness is not verified.
```

Algorithm 1 shows the SafeDeep refinement procedure. The refinement process proceeds layer-by-layer to obtain tighter convex approximations and bounds for the individual undetermined neurons (Lines 2–11). First, we tighten the lower and upper bounds of the undetermined neurons in the second hidden layer using LP queries, one for each targeted neuron in the layer (Lines 3–7). Then, we update the lower and upper bounds of all neurons in the next layers, based on the narrower bounds in the previous layers (Line 8), using Equation (3).

After re-calculating $\min(x_c^{(N)} - x_i^{(N)})$ on the linear approximations resulting from the obtained bounds (Line 9), if there exists $i \in \{1, ..., n_N\}$ that does not satisfy Inequality (2), we continue tightening bounds for the next layer, up to the last hidden layer; otherwise, the robustness is verified (Line 10). We avoid unnecessary computations and processing time by refining the bounds, one layer at a time, instead of simultaneously refining all undetermined neurons in all layers.

In Fig. 1, $\hat{x}_2^{(3)}$ and its ReLU image $x_2^{(3)}$ are the only neurons for which the initial bounds are insufficient to decide their state. Thus, we tightened the lower and upper bounds of this targeted neuron, and the refined range of $x_2^{(3)}$ is $[-2, 2]$, instead of $[-4, 4]$. If this network had more layers, the new tighter over-approximation could be used to obtain tighter bounds for the next layers, making the approximation more precise. In this example, as there are no more hidden layers, we obtain $\min(x_c^{(N)} - x_i^{(N)}) \geq 1$ by applying linear optimization on the over-approximated system resulting from the new bounds for $x_2^{(3)}$. This is sufficient to establish the robustness property. The robustness of this neural network is, therefore, proved for $\tilde{x}_1^{(1)} = 2$, $\tilde{x}_2^{(1)} = 0$, and $\delta = 1$.

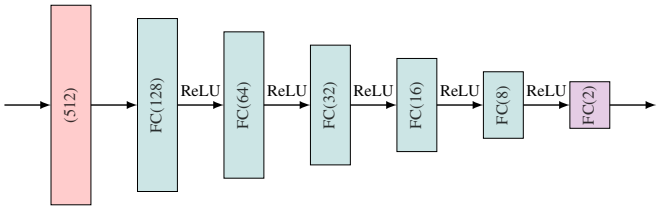


Fig. 3: Architecture of the deep neural network. Here, FC(.) shows a fully-connected layer.

III. Experimental Evaluation

In this section, we evaluate our proposed framework to verify the robustness of deep neural networks against two established tools. We compare SafeDeep to ERAN [12], which is based on over-approximation, and Reluplex [10], which is SMT-based, for a seven-layer neural network shown in Fig. 3. The input size is 512, and the output size is two, as the goal of the network is to classify input data into seizure/non-seizure classes. The activation function for all hidden layers is ReLU. The neural network is fed with the F8-T8 and F7-T7 channels of the EEG signal in CHB-MIT dataset [19], [20], consistent with the e-Glass wearable system for real-time seizure detection [3].

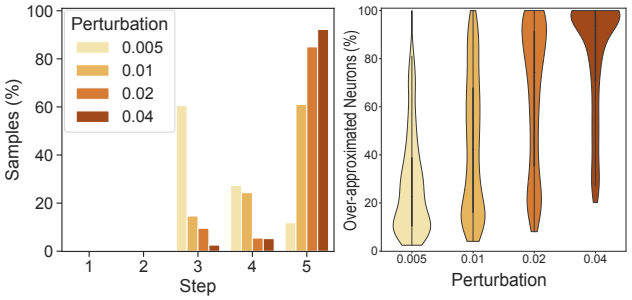
All experiments are conducted using a pre-trained deep neural network shown in Fig. 3 on random samples from the CHB-MIT dataset that are correctly classified by the network. We aim to verify the robustness for the input data against four perturbation δ values: 0.005, 0.01, 0.02, and 0.04. Table I shows our experimental results for comparing SafeDeep against ERAN and Reluplex, where the tool reports whether robustness surely does hold (i.e., robustness guarantee) or might not hold (i.e., inconclusive or, equivalently, robustness could not be proved).

In comparison with ERAN, on average, SafeDeep verifies the robustness for 43% more cases of the 785 random samples selected from CHB-MIT dataset, due to the tighter bounds obtained in our algorithm. It verifies the robustness for the majority of the samples when $\delta = 0.005$. On the other hand, when $\delta = 0.04$, it verifies the robustness for 81 cases, whereas ERAN cannot verify any cases. Using SafeDeep, it takes 32 to 47 seconds on average to run an experiment on this specific deep neural network shown in Fig. 3, while processing time of ERAN is less than one second in these experiments. Therefore,

TABLE I: Verified robustness and processing time (s) by SafeDeep, ERAN, and Reluplex on CHB-MIT dataset.

		Perturbation δ							
		$\delta = 0.005$		$\delta = 0.01$		$\delta = 0.02$		$\delta = 0.04$	
Method	#	Acc	Time	Acc	Time	Acc	Time	Acc	Time
SafeDeep	785	761	32	461	42	140	46	81	47
ERAN		591	0.5	376	0.6	43	0.7	0	0.7
SafeDeep	100	97	33	57	42	14	46	7	48
Reluplex		24	344*	9	774*	5	803*	1	180*

*The average processing time of Reluplex does not include the timeout cases.



(a) Steps required to verify the robustness for samples.

(b) Relative frequency of neurons requiring refinement.

Fig. 4: Results of refining bounds layer-by-layer on over-approximated undetermined neurons.

SafeDeep outperforms ERAN by a large margin, particularly for larger perturbations, within a reasonable processing time.

To evaluate SafeDeep against Reluplex, we randomly selected a subset of 100 samples from the previous dataset, as the processing time of Reluplex is several orders of magnitude more than SafeDeep and ERAN. We allotted a timeout of one hour for Reluplex, which is more than 70 times higher than the average processing time of our framework. In this experiment, Reluplex does not terminate in more than 80% of samples within the one hour timeout. Therefore, SafeDeep is significantly more efficient than Reluplex and verifies many more cases within a substantially shorter time interval. Note that the average processing time of Reluplex, reported in Table I, exclude timeout cases.

Fig. 4 presents the effect of layer-by-layer refining and selecting undetermined nodes to refine bounds instead of blindly refining the bounds of all neurons. Fig. 4a shows, for each perturbation δ and step m, the relative frequency of samples obtained with perturbation δ that could be verified in m steps. Note that the number of steps is the number of layers that bound refinement is done on their undetermined neurons plus one. In general, the larger the perturbation, the broader the neuron's range, the more the neurons that need to be updated, and the more the steps that are required to verify robustness. For example, when the perturbation is 0.01, about 20% of samples are verified with three steps (i.e., involving two refinement steps). Fig. 4b shows the relative frequency of undetermined neurons w.r.t. each perturbation. These results highlight that SafeDeep avoids unnecessary computations on-demand by over-approximating and refining bounds exclusively for undetermined neurons, only when required.

IV. Conclusions

In this paper, we presented SafeDeep, a scalable framework to verify the robustness of deep neural networks. The key insight in SafeDeep is in the on-demand incremental refinement of the convex approximations. Our experimental evaluation demonstrates that SafeDeep is more precise than ERAN, an over-approximation-based method, and substantially more efficient than Reluplex, an exact SMT-based method.

REFERENCES

[1] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37, 2020.

[2] Dionisije Sopic, Amin Aminifar, Amir Aminifar, and David Atienza. Real-time event-driven classification technique for early detection and prevention of myocardial infarction on wearable systems. *IEEE transactions on biomedical circuits and systems*, 12(5):982–992, 2018.

[3] Dionisije Sopic, Amir Aminifar, and David Atienza. e-Glass: A wearable system for real-time detection of epileptic seizures. In *International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018.

[4] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[5] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582. IEEE, 2016.

[6] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, pages 99–112. Chapman and Hall/CRC, 2018.

[7] Hongshuo Liang, Erlu He, Yangyang Zhao, Zhe Jia, and Hao Li. Adversarial attack and defense: A survey. *Electronics*, 11(8), 2022.

[8] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming, 2017.

[9] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *Computer-Aided Verification (CAV)*, pages 3–29. Springer, 2017.

[10] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer-Aided Verification (CAV)*, pages 97–117. Springer, 2017.

[11] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. AI2: Safety and robustness certification of neural networks with abstract interpretation. In *Symposium on security and privacy (SP)*, pages 3–18. IEEE, 2018.

[12] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages (PACMPL)*, 3:1–30, 2019.

[13] Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin Vechev. Beyond the single neuron convex barrier for neural network certification. *Advances in Neural Information Processing Systems*, 32, 2019.

[14] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *Computer-Aided Verification (CAV)*, pages 3–29. Springer, 2017.

[15] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda. A unified view of piecewise linear neural network verification. *Advances in Neural Information Processing Systems*, 31, 2018.

[16] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning (ICML)*, pages 5276–5285. PMLR, 2018.

[17] Amir Aminifar. Minimal adversarial perturbations in mobile health applications: The epileptic brain activity case study. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1205–1209. IEEE, 2020.

[18] Amir Aminifar. Universal adversarial perturbations in epileptic seizure detection. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2020.

[19] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[20] Ali Hossam Shoeb. *Application of machine learning to epileptic seizure onset detection and treatment*. PhD thesis, Massachusetts Institute of Technology, 2009.