VNN: Verification-Friendly Neural Networks with Hard Robustness Guarantees

Anahita Baninajjar [1] **Ahmed Rezine** [2] **Amir Aminifar** [1]

Abstract

Machine learning techniques often lack formal correctness guarantees, evidenced by the widespread adversarial examples that plague most deep-learning applications. This lack of formal guarantees resulted in several research efforts that aim at verifying Deep Neural Networks (DNNs), with a particular focus on safety-critical applications. However, formal verification techniques still face major scalability and precision challenges. The over-approximation introduced during the formal verification process to tackle the scalability challenge often results in inconclusive analysis. To address this challenge, we propose a novel framework to generate Verification-Friendly Neural Networks (VNNs). We present a post-training optimization framework to achieve a balance between preserving prediction performance and verification-friendliness. Our proposed framework results in VNNs that are comparable to the original DNNs in terms of prediction performance, while amenable to formal verification techniques. This essentially enables us to establish robustness for more VNNs than their DNN counterparts, in a time-efficient manner.

1. Introduction

The state-of-the-art machine learning techniques suffer from lack of formal correctness guarantees. It has been demonstrated by a wide range of "adversarial examples" in the deep learning domain (Moosavi-Dezfooli et al., 2016; Kurakin et al., 2018; Liang et al., 2022) and presents a major challenge in the context of safety-critical applications (Huang et al., 2020). To address this challenge, several verification frameworks for DNNs have been proposed in the state of the art (Katz et al., 2017; Tjeng et al., 2017; Huang et al., 2017; Singh et al., 2019b). However, the majority of state-of-the-art verification techniques suffer from poor scalability. As a result, there have been several excellent initiatives to improve the scalability of the existing verification frameworks for DNNs (Katz et al., 2019; Singh et al., 2019a; Baninajjar et al., 2023).

In contrast to the majority of the state-of-the-art verification techniques, here, we propose a new generation of DNNs that are amenable to verification, or simply verification-friendly. Verification-Friendly Neural Networks (VNNs) allow more efficiency, in terms of time, and more verified samples, while maintaining on-par prediction performance with their DNN counterparts. The key observation is that most verification frameworks are based on over-approximation and this over-approximation accumulates/amplifies along the forward pass. To reduce the over-approximation, we define an optimization problem to obtain VNNs by enforcing sparsity on the DNNs, constrained to satisfy/guarantee performance/robustness requirements. As such, our framework allows verification tools to establish robustness for a wider range of samples. While the main goal of VNNs is to be verification-friendly, our proposed framework also plays a regularization role, leading to more robust VNNs owing to their sparsity.

We evaluate our proposed framework based on the classical MNIST image dataset (LeCun, 1998) and demonstrate that VNNs are indeed amenable to verification, while maintaining accuracy as their DNN counterparts. The verification time of VNNs is always less than the original DNNs (up to one-third in some cases) due to the sparsity of the VNNs. At the same time, and more importantly, we observe that VNNs allow verification of up to 76 times more samples, besides their advantage in terms of verification time efficiency.

To evaluate our framework on real-world safety-critical applications, we consider two datasets from medical domain, namely, CHB-MIT (Shoeb, 2010), and MIT-BIH (Goldberger et al., 2000), for real-time epileptic seizure detection and real-time cardiac arrhythmia detection, respectively. Failure to detect an epileptic seizure or a cardiac arrhythmia episode in time may have irreversible consequences and potentially lead to death (Panelli, 2020; Stroobandt et al., 2019). For such safety-critical applications, we demonstrate

[1]Department of Electrical and Information Technology, Lund University, Lund, Sweden, [2]Department of Computer and Information Science, Linköping University, Linköping, Sweden. Correspondence to: Anahita Baninajjar <anahita.baninajjar@eit.lth.se>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

that the number of verified samples with VNNs is up to 24 and 34 times more than DNNs, considering the CHB-MIT and MIT-BIH datasets, respectively.

Finally, we compare the performance of VNNs with two state-of-the-art techniques, namely, Magnitude-Based Pruning (MBP) and Sparse Optimization Pruning (SOP) (Manngård et al., 2018). MBP and SOP approaches aim at introducing sparsity, without taking the robustness requirements into consideration. As a result, the sparse network is not guaranteed to satisfy the robustness requirements. The experiments confirm this explanation and show that VNNs are up to 46, 19, and 27 times more verification-friendly than MBP models on DNNs trained on MNIST, CHB-MIT, and MIT-BIH datasets, respectively. Moreover, VNNs are up to 51 times more verification-friendly than SOP models trained on the MNIST dataset.

2. Verification-Friendly Neural Networks

In this section, we discuss our proposed framework to generate VNNs. The framework comes with an optimization problem that takes a trained model as input (i.e., the original DNN) and results in a VNN. In the following, we introduce the optimization problem and explain the formulation of the objective function and constraints.

2.1. Deep Neural Networks

Let us first formalize our notation for DNNs. A DNN is a series of linear transformations and nonlinear activation functions to generate outputs using trained weights and biases. We consider $f_k(.) : R^{n_{k-1}} \rightarrow R^{n_k}$ as the function that generates values of the kth layer from its previous layer, where n_k shows the number of neurons in the kth layer. Therefore, the values of the kth layer, $\boldsymbol{x}^{(k)}$, are given by $\boldsymbol{x}^{(k)} = f_k(\boldsymbol{x}^{(k-1)}) = act(\boldsymbol{W}^{(k)}\boldsymbol{x}^{(k-1)} + \boldsymbol{b}^{(k)})$, such that $\boldsymbol{W}^{(k)}$ and $\boldsymbol{b}^{(k)}$ are the weights and biases of the kth layer and act is the activation function.

2.2. Layer-Wise Optimization Problem

Our optimization problem to obtain a VNNs is performed layer-by-layer. Therefore, here, we focus on the lth layer of an N-layer DNN. To obtain a VNN, our aim is to minimize the number of non-zero elements of weight $\tilde{\boldsymbol{W}}^{(l)}$ and bias $\tilde{\boldsymbol{b}}^{(l)}$ of the lth layer of an N-layer DNN to have a model that is more sparse while still accurate. To minimize the number of non-zero elements of $\tilde{\boldsymbol{W}}^{(l)}$ and $\tilde{\boldsymbol{b}}^{(l)}$, we consider the L_0 norm $\|.\|_0$, also known as the sparse norm, as the objective function. The L_0 norm counts the number of non-zero elements in a vector or matrix. Therefore, the objective function is $\|\tilde{\boldsymbol{W}}^{(l)}\|_{0,0} + \|\tilde{\boldsymbol{b}}^{(l)}\|_0$, to find the minimum number of non-zero elements in $\tilde{\boldsymbol{W}}^{(l)}$ and $\tilde{\boldsymbol{b}}^{(l)}$. In the following, we outline our proposed constrained optimization problem to obtain a VNN:

$$\min_{\tilde{\boldsymbol{W}}^{(l)}, \tilde{\boldsymbol{b}}^{(l)}} \|\tilde{\boldsymbol{W}}^{(l)}\|_{0,0} + \|\tilde{\boldsymbol{b}}^{(l)}\|_0, \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{x}^{(k)} = f_k(\boldsymbol{x}^{(k-1)}),\ \forall k \in \{1, \ldots, N\} \tag{2}$$

$$\tilde{\boldsymbol{x}}^{(l)} = \tilde{f}_l(\boldsymbol{x}^{(l-1)}), \tag{3}$$

$$\tilde{\boldsymbol{x}}^{(m)} = f_m(\tilde{\boldsymbol{x}}^{(m-1)}), \forall m \in \{l+1, \ldots, N\}, \tag{4}$$

$$\|\tilde{\boldsymbol{x}}^{(l)} - \boldsymbol{x}^{(l)}\|_\infty \leq \epsilon, \tag{5}$$

$$\arg\max \boldsymbol{x}^{(N)} = \arg\max \tilde{\boldsymbol{x}}^{(N)} = y, \tag{6}$$

$$\forall \boldsymbol{x}^{(0)} \in \boldsymbol{D}_{val}. \tag{7}$$

Consider the constraints presented in Equations (2)–(7). Equation (2) establishes the neurons' values of the original DNN with the trained weights and biases. For setting up the neurons' values after optimization, we define $\tilde{f}_l(.) : R^{n_{l-1}} \rightarrow R^{n_l}$ as a new nonlinear function that transfers neurons' values of the $(l-1)$th layer to the target layer l using the optimized weights and biases. Therefore, $\tilde{\boldsymbol{x}}^{(l)}$ is given by $\tilde{\boldsymbol{x}}^{(l)} = \tilde{f}_l(\boldsymbol{x}^{(l-1)}) = act(\tilde{\boldsymbol{W}}^{(l)}\boldsymbol{x}^{(l-1)} + \tilde{\boldsymbol{b}}^{(l)})$ in Equation (3). The change of neurons' values in layer l may result in different values for the neurons of the next layers, hence we introduce Equation (4). Equation (4) computes updated values for neurons in layers from $l+1$ to the end of the DNN, which is layer N, using the new values of neurons in layer l. Equation (5) ensures that $\tilde{\boldsymbol{x}}^{(l)}$ takes a value within the neighborhood of ϵ around $\boldsymbol{x}^{(l)}$ to avoid extreme changes and an excessive loss of accuracy. If $\epsilon = 0$, the value of each neuron is not allowed to change after optimization, i.e., $\tilde{\boldsymbol{x}}^{(l)} = \boldsymbol{x}^{(l)}$. If $\epsilon > 0$, then $\tilde{\boldsymbol{x}}^{(l)}$ can be different from $\boldsymbol{x}^{(l)}$ and the optimization has more freedom to find a more sparse VNN.

In Equation (6), the optimization problem is constrained to keep the same class as the one derived by the DNN prior to the optimization. Consider that the input $\boldsymbol{x}^{(0)}$ belongs to class c. It means $\boldsymbol{x}_c^{(N)} > \boldsymbol{x}_i^{(N)}$ for all neurons $i \neq c$ in the last layer $l = N$. To obtain the same class after optimization, the same holds for $\tilde{\boldsymbol{x}}_i^{(N)}$, i.e., $\tilde{\boldsymbol{x}}_c^{(N)} > \tilde{\boldsymbol{x}}_i^{(N)}$ for all neurons $i \neq c$, which is formulated in the context of our linear optimization problem. Moreover, introducing a constant robustness margin M to this inequality converts it to $\tilde{\boldsymbol{x}}_c^{(N)} > \tilde{\boldsymbol{x}}_i^{(N)} + M$ to provide hard guarantees for enhancing the robustness of the DNN. However, for the simplicity of the presentation, we adopt the compact form of this constraint in Equation (6) in the remainder of this paper. In Equation (6), y is the label of each $\boldsymbol{x}^{(0)}$ in the validation set $\boldsymbol{D}_{val}$. All constraints need to hold for all inputs in the validation set $\boldsymbol{D}_{val}$ (Equation (7)). In the optimization problem, we consider the validation set instead of the training set to avoid over-fitting in the VNN.

2.3. Relaxation of Objective Function

The L_0 norm is expressed in the form of an L_p norm, but to be precise, it is not a norm (Chou et al., 2022). However, optimizing the L_0 norm poses significant challenges due to the non-convex nature of the L_0 norm. Here, we elaborate on how to overcome this difficulty.

The L_1 norm also referred to as the Manhattan norm, is the tightest convex relaxation of the L_0 norm in convex optimization (Zass & Shashua, 2006). While the L_0 norm counts the number of non-zero elements of a matrix, the L_1 norm sums the absolute values of the elements. Being a convex function, the L_1 norm can be used in convex optimization problems, which is not possible with the L_0 norm. Therefore, the optimization objective is reformulated as follows to be convex:

$$\min_{\tilde{\boldsymbol{W}}^{(l)},\tilde{\boldsymbol{b}}^{(l)}} \quad \|\tilde{\boldsymbol{W}}^{(l)}\|_{1,1} + \|\tilde{\boldsymbol{b}}^{(l)}\|_1. \tag{8}$$

2.4. Reformulation of Constraints

The constraints of the optimization problem are nonlinear, which is caused by the activation functions in $\tilde{f}_l(.)$ and $f_m(.)$. Henceforth, we perform the analysis on the ReLU activation function, which is the most popular activation function in DNNs, but our framework can be extended to any nonlinear activation function that can be presented in a piece-wise linear form.

Performing function $ReLU(x) = \max(0, x)$ results in two possible states for each neuron $x_i^{(l)}$. Suppose $f_{l,i}(.)$ generates the output of $f_l(.)$ for neuron $x_i^{(l)}$; The neuron either remains continuously active if $f_{l,i}(\boldsymbol{x}^{(l-1)}) = \boldsymbol{W}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + b_i^{(l)} > 0$, or becomes permanently inactive when $f_{l,i}(\boldsymbol{x}^{(l-1)}) = 0$ or $\boldsymbol{W}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + b_i^{(l)} \leq 0$. It reflects that the ReLU function is a combination of linear segments, creating a piece-wise linear function. If we constrain the neuron to remain within one of these linear segments, the ReLU function behaves linearly, making it possible to handle the proposed optimization. Therefore, the neurons in the target layer l must remain in the same state (or in the same linear segment). It is formulated as follows for $\tilde{x}_i^{(l)}$:

$$\begin{cases} \tilde{\boldsymbol{W}}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + \tilde{b}_i^{(l)} \geq 0 & \text{if } f_{l,i}(\boldsymbol{x}^{(l-1)}) > 0, \\ \tilde{\boldsymbol{W}}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + \tilde{b}_i^{(l)} \leq 0 & \text{otherwise.} \end{cases} \tag{9}$$

The above introduces a new constraint for each neuron $\tilde{x}_i^{(l)}$, where if the neuron has been originally active ($f_{l,i}(\boldsymbol{x}^{(l-1)}) = \boldsymbol{W}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + b_i^{(l)} > 0$), we constrain the new weights and bias to also remain active ($\tilde{\boldsymbol{W}}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + \tilde{b}_i^{(l)} \geq 0$); otherwise (if the neuron has originally been inactive, i.e., $f_{l,i}(\boldsymbol{x}^{(l-1)}) = 0$ or $\boldsymbol{W}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + b_i^{(l)} \leq 0$), we constrain the new weights and bias to also remain inactive ($\tilde{\boldsymbol{W}}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + \tilde{b}_i^{(l)} \leq 0$).

In this framework, the values of the neurons in layers $k < l$ remain unchanged during the optimization process, but the output of layer l may change. Changing neurons' values of layer l may cause an alteration in the values of any neuron of the following layers $m > l$, which could potentially influence the classification result. Ensuring the consistency of the assigned class for each sample involves capturing the entire DNN. To do so, we leverage on the piece-wise linearity of the ReLU function once more. Suppose $f_{m,j}(.)$ produces the output of $f_m(.)$ for neuron $\tilde{x}_j^{(m)}$. We formulate the updated value of neuron $\tilde{x}_j^{(m)}$ in layer $m > l$ as follows:

$$\begin{cases} \boldsymbol{W}_{j,:}^{(m)}\tilde{\boldsymbol{x}}^{(m-1)} + b_j^{(m)} \geq 0 & \text{if } f_{m,j}(\tilde{\boldsymbol{x}}^{(m-1)}) > 0, \\ \boldsymbol{W}_{j,:}^{(m)}\tilde{\boldsymbol{x}}^{(m-1)} + b_j^{(m)} \leq 0 & \text{otherwise.} \end{cases} \tag{10}$$

2.5. Reformulated Layer-Wise Optimization Problem

After relaxing the objective function and the constraints, we state our proposed layer-wise optimization problem as a linear program:

$$\begin{aligned} &\min_{\tilde{\boldsymbol{W}}^{(l)},\tilde{\boldsymbol{b}}^{(l)}} \|\tilde{\boldsymbol{W}}^{(l)}\|_{1,1} + \|\tilde{\boldsymbol{b}}^{(l)}\|_1, \\ &\text{subject to:} \\ &\boldsymbol{x}^{(k)} = f_k(\boldsymbol{x}^{(k-1)}),\ \forall k \in \{1,\dots,N\} \\ &\begin{cases} \tilde{x}_i^{(l)} = \tilde{\boldsymbol{W}}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + \tilde{b}_i^{(l)} \geq 0 & \text{if } f_{l,i}(\boldsymbol{x}^{(l-1)}) > 0, \\ \tilde{x}_i^{(l)} = 0, \tilde{\boldsymbol{W}}_{i,:}^{(l)}\boldsymbol{x}^{(l-1)} + \tilde{b}_i^{(l)} \leq 0 & \text{otherwise} \end{cases} \\ &\forall i \in \{1,\dots,n_l\}, \\ &\begin{cases} \tilde{x}_j^{(m)} = \boldsymbol{W}_{j,:}^{(m)}\tilde{\boldsymbol{x}}^{(m-1)} + b_j^{(m)} \geq 0 & \text{if } f_{m,j}(\tilde{\boldsymbol{x}}^{(m-1)}) > 0, \\ \tilde{x}_j^{(m)} = 0, \boldsymbol{W}_{j,:}^{(m)}\tilde{\boldsymbol{x}}^{(m-1)} + b_j^{(m)} \leq 0 & \text{otherwise} \end{cases} \\ &\forall j \in \{1,\dots,n_m\}, \forall m \in \{l+1,\dots,N\}, \\ &\|\tilde{\boldsymbol{x}}^{(l)} - \boldsymbol{x}^{(l)}\|_\infty \leq \epsilon, \\ &\arg\max \boldsymbol{x}^{(N)} = \arg\max \tilde{\boldsymbol{x}}^{(N)} = y, \\ &\forall \boldsymbol{x}^{(0)} \in \boldsymbol{D}_{val}. \end{aligned} \tag{11}$$

2.6. End-to-End Optimization Problem

The optimization problem in Equation (11) only targets a single layer l. Here, we propose an end-to-end process to generate all layers of VNNs iteratively. We initiate the process with an already-trained model along with a validation set. The process proceeds layer-by-layer to obtain sparse weights and biases for the DNN. To find a verification-friendly model within our framework, we start from the

first hidden layer, $l = 1$, and run the optimization problem in Equation (11) to find the sparse matrix of weights and vector of biases such that the outcome of the classification on the validation set remains unchanged. Then, the weights and biases of the first hidden layer, $\tilde{\boldsymbol{W}}^{(1)}$ and $\tilde{\boldsymbol{b}}^{(1)}$, remain unaltered, while the optimization is performed on the second hidden layer to detect the sparse $\tilde{\boldsymbol{W}}^{(2)}$ and $\tilde{\boldsymbol{b}}^{(2)}$. This process continues until the last layer of the DNN.

3. Evaluation

In this section, we assess the performance of our proposed framework to generate VNNs compared to state-of-the-art techniques. Our framework is implemented using the Gurobi solver (Gurobi Optimization, LLC, 2023) on a MacBook Pro with an 8-core CPU and 32 GB of RAM[1].

3.1. Datasets

We consider three public datasets for evaluation of VNNs, namely, the MNIST dataset (LeCun, 1998) and two datasets from safety-critical medical applications. The first one is about epileptic seizure detection based on the CHB-MIT Scalp EEG database (Shoeb, 2010) and the second one concerns cardiac arrhythmia detection based on the MIT-BIH Arrhythmia database (Goldberger et al., 2000).

MNIST image dataset (LeCun, 1998) is composed of gray-scale handwritten digits such that each digit is represented by a 28×28 pixel image. Similar to (Singh et al., 2019b), we consider the first 400 images of the test set and equally split them into validation and test sets for the optimization and evaluation of DNNs, respectively.

CHB-MIT dataset (Shoeb, 2010) comprises 23 individuals diagnosed with epileptic seizures. This dataset is recorded in the international 10-20 system of EEG electrode positions and nomenclature. We utilize two electrode pairs, namely F7-T7 and F8-T8, which are frequently employed in seizure detection research (Sopic et al., 2018). The recordings are sampled at a rate of 256 Hz. For each patient, we consider 60%, 20%, and 20% of the dataset for the training, validation, and test sets, respectively. The set sizes vary among patients, and each patient's entire dataset consists of 44 to 1009 samples, with an average of 252 samples.

MIT-BIH dataset (Goldberger et al., 2000) contains ECG signals that have been digitized with a sampling rate of 360 Hz. To be able to formulate a classification problem, we select the 14 subjects with at least two distinct types of heartbeats. Similar to the CHB-MIT dataset, each model is trained using a training set, optimized with a validation set, and evaluated using a test set. The training set includes 75% of the entire dataset. The remaining 25% are equally partitioned among the test set and the validation set. The size of the datasets may differ among the patients and each patient's entire set consists of 400 to 1800 samples, with an average of 957 samples.

[1]The code is available on https://github.com/anahitabn94/VNN.

3.2. Deep Neural Networks

We consider Fully-connected Neural Networks (FNNs) and Convolutional Neural Networks (CNNs) to investigate the performance of our proposed framework.

FNNs. We adopt FNNs trained by (Singh et al., 2019b) and (Ugare et al., 2022) on MNIST to explore the performance of our framework. In this paper, the notation of $n \times m$ is used for the architecture of a FNN comprising n hidden layers, each containing m neurons. We examine six FNNs with 2×50, 2×100, 5×100, 5×200, 8×100, and 8×200 architectures generated by (Singh et al., 2019b), where each dense layer is followed by a ReLU activation function. Besides, we consider three FNNs with 6×500 architectures, two of which are generated using Projected Gradient Descent (PGD) adversarial training, i.e., defended against adversarial attacks (Singh et al., 2019b). Moreover, a set of pruned FNNs with 7×200 architecture, developed by (Ugare et al., 2022), is employed in our study. The input and output layers of all the FNNs have 784 and 10 neurons, respectively.

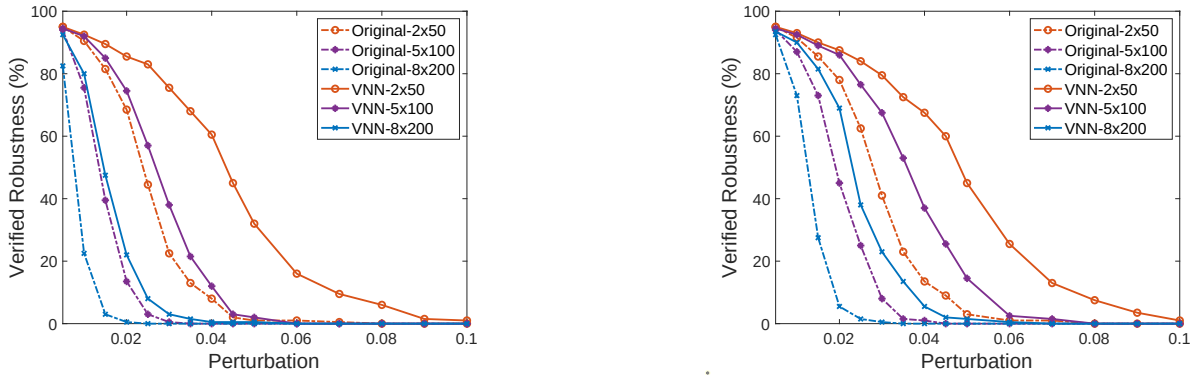
CNNs. We train CNNs on CHB-MIT and MIT-BIH datasets to study the compatibility of our framework with CNNs. For the CHB-MIT dataset, we train 23 personalized CNNs, one for each patient. Each model consists of three hidden layers, including two convolutional layers with 3 and 5 filters and kernel sizes of 100 and 200, respectively, each followed by a max-pooling layer, as well as a dense layer with 40 neurons. The input layer has 2048 neurons. For the MIT-BIH dataset, we train 14 individual CNNs, each with an input layer of 320 neurons, a convolutional layer including 3 filters with a kernel size of 64, and a dense layer with 40 neurons. The output layer of all CNNs has two output neurons to classify the input into negative (non-seizure/normal) or positive (seizure/arrhythmia) classes.

3.3. Robustness Properties

The robustness property is formulated in terms of the L_∞ norm and parameterized by a constant δ. The adversarial region includes all perturbed inputs such that each input neuron has a maximum distance of δ from the corresponding neuron's value of the original input. The range of perturbation is considered differently for different datasets, as it depends on the robustness of each DNN w.r.t. its structure. The maximum studied perturbations for MNIST, CHB-MIT, and MIT-BIH are 0.1, 0.01, and 0.2, respectively.

Table 1: Verified robustness (%) and average processing time (s) of original DNNs and VNNs ($\epsilon = 0$) on the MNIST dataset for six FNNs with different network structures using ERAN and SafeDeep. VNNs are optimized models that are more verification-friendly and time-efficient than their corresponding original DNNs.

		ERAN (Singh et al., 2019b)										SafeDeep (Baninajjar et al., 2023)									
		$\delta = 0.01$		$\delta = 0.02$		$\delta = 0.03$		$\delta = 0.05$		$\delta = 0.07$		$\delta = 0.01$		$\delta = 0.02$		$\delta = 0.03$		$\delta = 0.05$		$\delta = 0.07$	
	Model	Org	VNN	Org	VNN	Org	VNN	Org	VNN	Org	VNN	Org	VNN	Org	VNN	Org	VNN	Org	VNN	Org	VNN
Robustness (%)	2×50	90.5	**92.5**	68.5	**85.5**	22.5	**75.5**	1	**32**	0.5	**9.5**	91.5	**93**	78	**87.5**	41	**79.5**	3	**45**	1	**13**
	2×100	85	**92.5**	33.5	**85**	3.5	**70**	0	**16.5**	0	**2**	90.5	**92.5**	67.5	**86.5**	15.5	**77.5**	0	**33.5**	0	**7**
	5×100	75.5	**92**	13.5	**74.5**	0.5	**38**	0	**2**	0	0	87	**92.5**	45	**86**	8	**67.5**	0	**14.5**	0	**1.5**
	5×200	27	**84**	0.5	**29.5**	0	**4.5**	0	0	0	0	76	**90**	7	**73**	0	**27**	0	**1**	0	0
	8×100	67	**85**	17.5	**60.5**	0.5	**28**	0	**3.5**	0	0	84	**90**	40	**79**	11.5	**56**	0	**16.5**	0	**1.5**
	8×200	22.5	**80**	0.5	**22**	0	**3**	0	**0.5**	0	0	73	**90**	5.5	**69**	0.5	**23**	0	**1.5**	0	0
Average Time (s)	2×50	0.2	0.2	0.2	0.2	0.2	0.2	0.3	**0.2**	0.3	**0.2**	0.8	**0.6**	1.2	**0.7**	1.9	**0.7**	2.2	**1.1**	2.2	**1.1**
	2×100	0.5	**0.4**	0.6	**0.5**	0.7	**0.5**	0.8	**0.6**	0.8	**0.7**	2.7	**1.7**	7.6	**1.9**	7.6	**2.6**	10	**4.2**	10	**4.2**
	5×100	1.7	**1.5**	2.3	**1.6**	2.5	**1.9**	2.5	**2.2**	2.5	**2.2**	16	**9**	36	**13**	44	**19**	48	**29**	48	**30**
	5×200	8	**6**	9	**7**	9	**8**	9	**8**	9	**8**	106	**35**	180	**68**	180	**88**	180	**94**	180	**94**
	8×100	3	3	4	**3**	5	**4**	5	**4**	5	5	46	**21**	93	**31**	111	**51**	123	**60**	123	**87**
	8×200	16	**11**	19	**15**	20	**17**	20	**17**	20	**17**	240	**94**	325	**140**	335	**165**	335	**173**	335	**173**



(a) Verified robustness (%) using ERAN.

(b) Verified robustness (%) using SafeDeep.

Figure 1: Verified robustness (%) of the MNIST dataset w.r.t. different values of perturbations for original DNNs and their corresponding VNNs using ERAN and SafeDeep.

3.4. Verification Frameworks

We consider ERAN (Singh et al., 2019b) and SafeDeep (Baninajjar et al., 2023) to evaluate the effectiveness of our proposed framework in this paper. ERAN and SafeDeep are over-approximation-based frameworks designed to verify the robustness of DNNs. ERAN, or more specifically DeepPoly, works based on the principles of abstract interpretation. SafeDeep, on the other hand, works based on incremental refinement of convex approximations.

3.5. Results and Analysis

In this section, we evaluate our proposed framework in terms of the provability of robustness. We shall first assess the proposed VNNs considering the MNIST dataset.

3.5.1. MNIST.

We start by evaluating the VNNs generated by our proposed framework against pre-trained original DNNs on the MNIST dataset. Table 1 illustrates the verified robustness and average processing time of FNNs evaluated by ERAN and SafeDeep. Here, we show five different perturbation values, δ, covering a broad range of perturbations. Based on the results in Table 1, the number of samples whose robustness could be established is considerably larger for the VNNs than that of the original DNNs Moreover, as the perturbation increases, the difference in the number of verified samples becomes even more significant. Furthermore, the average processing time of VNNs is up to three times less than their corresponding original DNNs.

In Figure 1 we select three FNNs with 2×50, 5×100, and 8×200 architectures as representatives of small, medium,

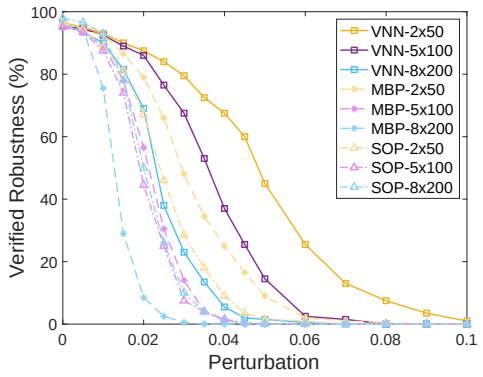


Figure 2: Comparison with state of the art: MBP and SOP (Manngård et al., 2018).

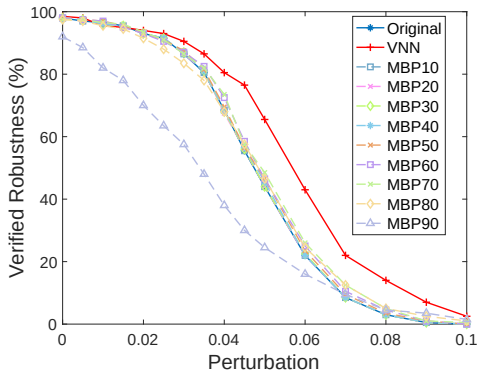


Figure 3: Comparison with state of the art: MBP (Ugare et al., 2022).

and large FNNs, respectively, to study a wider range of perturbation. Based on the results in Figures 1a and 1b using ERAN and SafeDeep, we observe that the accuracy of VNNs is within the same range as their corresponding original DNNs, which is shown in the absence of perturbation $\delta = 0$. In addition, the verification tools consistently demonstrate the robustness of a greater number of samples and for larger perturbations in the case of VNNs.

Comparison with State-of-the-Art Pruning. Here, we compare our framework with two state-of-the-art pruning approaches in terms of compatibility with the verification techniques: (1) Magnitude-Based Pruning (MBP) is widely used to enhance the scalability of DNNs by forcing small values of weights and biases to become zero (Guidotti et al., 2020), and (2) Sparse Optimization Pruning (SOP) aims to enhance sparsity of DNNs during training while preserving accuracy (Manngård et al., 2018).

Figure 2 shows the verified robustness for three FNNs with 2×50, 5×100, and 8×200 architectures pruned by the MBP and SOP compared to our proposed VNNs. Accuracy is depicted on the y-axis where $\delta = 0$. The results demonstrate that the accuracy of the MBP and SOP models are within the same range as VNNs. However, the VNNs offer substantially higher verified robustness than the corresponding MBP and SOP models.

Next, we conduct another experiment using the publicly available pruned DNNs proposed by (Ugare et al., 2022) using MBP method. The authors shared an original and 9 pruned FNNs with 7×200 architecture. The pruned DNNs are made by discarding the smallest weights at each layer with pruning rates ranging from 10% to 90%. The findings in Figure 3 indicate that, despite their impressive accuracy shown on the y-axis where $\delta = 0$, these models do not demonstrate higher verification-friendliness compared to the original one.

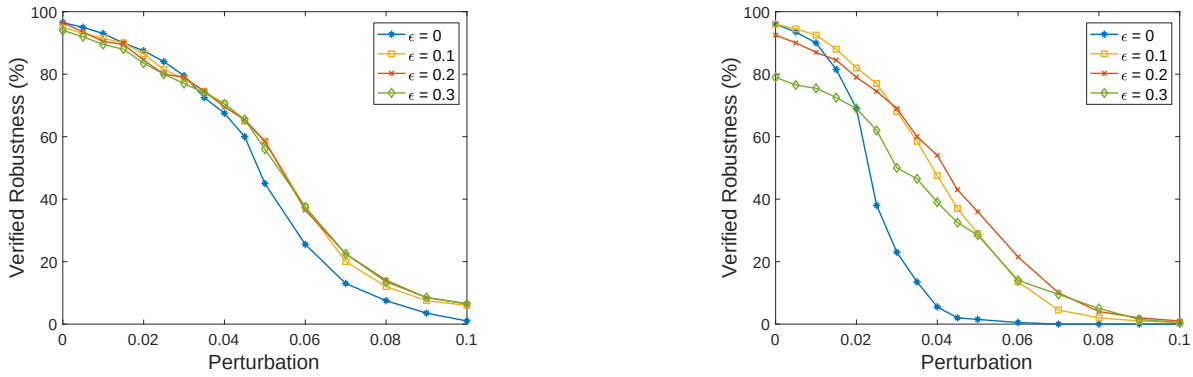
Investigating Robustness of VNNs. Here, we investigate the robustness of VNNs using the exact verification tool Marabou, an SMT-based tool (Katz et al., 2019). Exact verification frameworks, which are sound and complete, suffer from poor scalability. To overcome this issue, we only consider the FNN with 2×50 architecture and opt for an extended timeout of 600 seconds per sample. Table 2 presents the results of the FNN and its corresponding VNN. For larger FNNs, over 75% of cases exceeded the timeout.

Table 2: Robustness of the DNN and VNN.

		VNN		
		Robust	Not Robust	Timeout
DNN	Robust	41	0	0
	Not Robust	269	275	261
	Timeout	927	0	27

The study involves a total of 1800 states, corresponding to 200 samples in the test set, each potentially misclassified among 9 classes, as MNIST has 10 classes. The number of timeout cases in the DNN is three times more than that in the VNN, which implies the VNN is more verification-friendly. On the other hand, in 269 out of the total 1800 cases, the VNN is guaranteed to be robust, while the DNN lacks robustness in these specific cases. Finally, no cases are identified where the DNN exhibits robustness while the VNN does not.

Hyperparameter. Next, we investigate how changing the hyperparameter ϵ impacts verification-friendliness on FNNs characterized by different structures. Note that here, we employ a multiplicative perturbation. The outcomes obtained through SafeDeep are depicted in Figure 4, where the x-axis and y-axis represent perturbation δ and verified robustness, respectively. Note that the y-axis captures accuracy in the absence of a perturbation, which means



(a) 2×50 VNNs.

(b) 8×200 VNNs.

Figure 4: The effect of ϵ for different network structures.

$\delta = 0$. The key idea is to relax the constraints, which leads to an optimization problem with a larger solution space, to obtain a sparser VNN in the optimization problem with fewer non-zero weights/biases. For example, when $\epsilon = 0.1$, the value of each neuron $\tilde{x}_i^{(l)}$ can change in the range of $[0.9x_i^{(l)}, 1.1x_i^{(l)}]$, and the freedom of choices allows for sparser VNNs that are easier to handle by the over-approximation-based verification techniques. However, a validation set with inadequate size, in comparison to the size of the model, may lead to a decrease in the model's prediction performance.

Figure 4a displays the verification results obtained for VNNs of a small network with two dense layers each with 50 hidden neurons. As the DNN is small in comparison to the size of the validation set, which contains 200 samples, the accuracy of its corresponding VNNs remains within the same range, while the verified robustness increases up to 13%. On the other hand, Figure 4b, which illustrates the results of a large network with 9×200 architecture, shows a slight decrease in accuracy when ϵ increases. This trade-off between accuracy and verification-friendliness rises due to the small size of the validation set in relation to the sizes of the VNNs. When $\epsilon = 0$ or $\epsilon = 0.1$, the accuracy is above 90% and the model is more robust against small values of perturbation, while with $\epsilon = 0.3$, the accuracy slightly decreases to 80%. Our experiments show that increasing the size of the validation set increases the prediction performance.

VNN for Adversarially-Trained Models. Our proposed framework can be applied in combination with state-of-the-art adversarial training techniques, which defend against adversarial attacks, such as PGD, and still further improve verification-friendliness.

Figure 5 illustrates the verified robustness for six FNNs, all with 6×500 architectures. We consider two adversarial-trained models, trained via PGD adversarial training with δ values of 0.1 and 0.3, as outlined in (Singh et al., 2019b), denoted as PGD1 and PGD3, respectively. For each model, there are two corresponding VNNs with $\epsilon = 0$ and $\epsilon = 0.1$. They are denoted as VNN0(PGD1) and VNN1(PGD1) for PGD1 and VNN0(PGD3) and VNN1(PGD3) for PGD3. The accuracy is shown on the y-axis when $\delta = 0$ in Figure 5. More concretely, Figure 5 shows that the accuracy values of VNN0(PGD1) and VNN1(PGD1) are comparable to that of PGD1, while exhibiting higher verification-friendliness. The same pattern exists for the PGD-trained model with $\delta = 0.3$, denoted as PGD3, and its corresponding VNNs, namely, VNN0(PGD3) and VNN1(PGD3).

Number of Optimized Layers. Our proposed framework optimizes the entire DNN layer by layer. The number of optimized layers is, therefore, another parameter that affects the performance of VNNs. Figure 6 shows the results of robustness verification for a VNN with 8×200 architecture and $\epsilon = 0.2$ using SafeDeep. The x-axis shows the number of layers that are optimized using our framework, e.g., 3 means the first 3 hidden layers of the VNN are optimized. Each curve shows the robustness verification results w.r.t. a specific value of perturbation δ to be able to compare the effect of increasing the number of optimized layers of VNNs. Figure 6 demonstrates that as the number of optimized layers increases, shown on the x-axis, the proportion of verified cases increases. This phenomenon arises as increasing the number of optimized layers leads to decreasing the number of non-zero neurons and over-approximation of each neuron; thus VNNs become more verification-friendly.

Time Complexity. Our proposed framework to generate VNNs has a polynomial time complexity, since for each layer one linear program is solved. The end-to-end process of generating VNNs also has linear time complexity with the number of layers. Experimentally, the processing time of FNNs with 2×50, 2×100, 5×100, 5×200, 8×100, and 8×200 architecture is 40, 101, 147, 463, 195, and 618

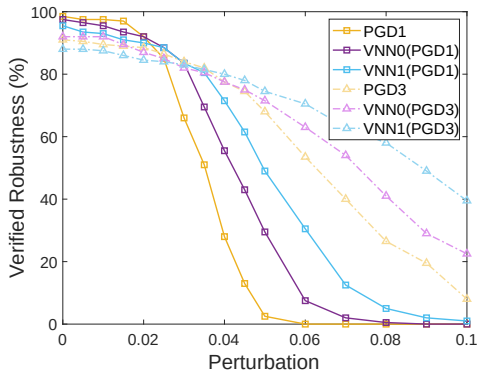


Figure 5: Comparison of PGD-trained networks (Singh et al., 2019b) with VNN-enhanced networks.

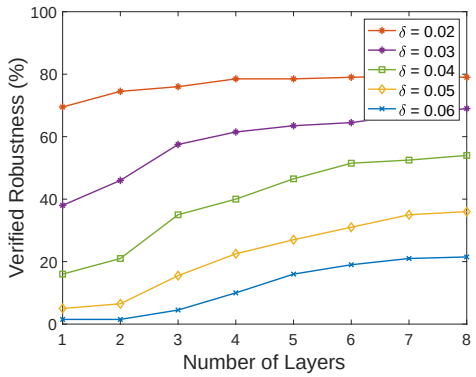


Figure 6: Increasing the number of optimized layers in an FNN with 8×200 architecture and $\epsilon = 0.2$.

seconds, respectively, when $\epsilon = 0$. However, increasing the value of ϵ leads to a decrease in processing time.

Next, we evaluate our proposed framework in the context of two safety-critical medical applications, namely, epileptic seizure detection and cardiac arrhythmia detection, to demonstrate its relevance.

3.5.2. CHB-MIT.

Here, we analyze our proposed framework based on the CHB-MIT dataset. Figures 7a and 7b show the verified robustness curves using ERAN and SafeDeep, where perturbation δ is in the range of $[0.0001, 0.01]$. The y-axis depicts accuracy under conditions of zero perturbation $\delta = 0$. The curves demonstrate the average ratio of samples for which robustness is verified and the shaded areas represent the variance w.r.t. the patients. The verified robustness of 23 individualized CNNs is evaluated on the patients in the CHB-MIT dataset to investigate their behavior for different perturbations. The accuracy ($\mu \pm \sigma^2$) of the original CNNs and VNNs is $85.7\% \pm 3.8\%$ and $82.5\% \pm 4.2\%$, respectively, while the VNNs are substantially more verification-friendly. Our framework generates VNNs with $\epsilon = 0$ that have up to 9 and 24 times more verified robustness using ERAN and SafeDeep, respectively.

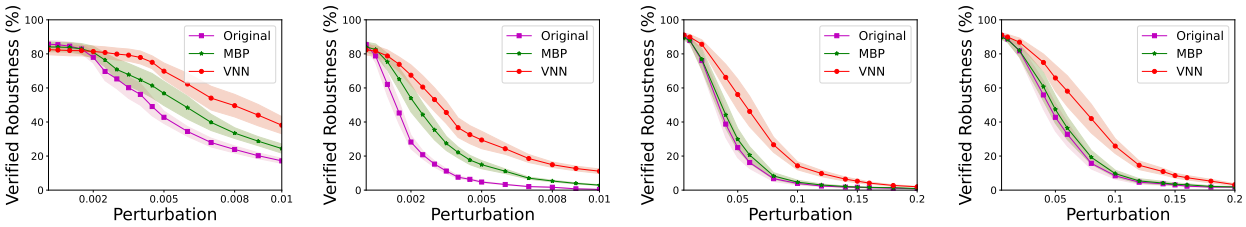
Moreover, we look into MBP models, where the 10% smallest weights and biases of the original DNNs are set to zero (Guidotti et al., 2020). By setting this threshold, 40% to 50% of all weights and biases became zero for each CNN, while higher rates significantly decrease the accuracy of the MBP models. Although MBP improves verification exploring by established over-approximation-based methods, it is not as effective as our proposed optimization framework. Indeed, using a fixed threshold to disregard weights not only does not minimize the number of non-zero elements but also ignores the model's accuracy. Our proposed framework, on the other hand, actively looks for weights and biases that maintain accuracy while forcing as many as possible to become zero. For the CHB-MIT dataset, the accuracy of the MBP models is $84.1\% \pm 4.1\%$, which is only slightly higher than VNNs, but our framework is up to one order of magnitude more verification-friendly. This slight difference in accuracy is because VNNs are approximately 40% more sparse. However, the accuracy of the MBP models drops to $68.4\% \pm 4.3\%$, if we enforce the MBP models to have similar sparsity as our VNNs.

3.5.3. MIT-BIH.

Next, we explore the performance of our framework considering the MIT-BIH dataset using ERAN and SafeDeep. As shown in Figures 7c and 7d, VNNs generated by our framework have a higher rate of verified robustness. Moreover, VNNs demonstrate superior performance when subjected to higher levels of perturbation. Figures 7c and 7d show the verified robustness of original DNNs, MBP models, and VNNs, with the accuracy of $91.5\% \pm 3.1\%$, $90.7\% \pm 3.4\%$, and $92.0\% \pm 3.0\%$, respectively. Note that MBP models are created using the same conditions as CHB-MIT models, wherein a threshold is applied to set the lowest 10% of weights and biases to zero. The experiments show that our proposed framework generates VNNs with $\epsilon = 0$ that have up to 34 and 30 times more verified robustness compared to the original CNNs using ERAN and SafeDeep, respectively. Furthermore, the performance of VNNs is significantly better than MBP models.

4. Related Work

DNNs are known for their vulnerability to adversarial examples (Szegedy et al., 2013). Adversarial examples are obtained by a slight modification of inputs, essentially leading to inputs that are misclassified by the DNN despite their extreme similarity to the original correctly classified inputs. Below, we review several studies aiming at improving the



(a) CHB-MIT, ERAN (b) CHB-MIT, SafeDeep (c) MIT-BIH, ERAN (d) MIT-BIH, SafeDeep

Figure 7: Mean and variance of verified robustness (%) for the CHB-MIT and MIT-BIH datasets. MBP models are generated by a threshold to force small weights to become zero, and VNNs are generated using our proposed framework.

scalability of robustness verification of DNNs against adversarial examples.

One prominent research direction in this domain aims at enhancing the scalability of verification techniques for establishing robustness properties for DNNs. Here, the state-of-the-art techniques for verification robustness of DNNs can be categorized as either precise (Tjeng et al., 2017; Huang et al., 2017; Katz et al., 2017) or based on over-approximation (Singh et al., 2019b; Weng et al., 2018; Bunel et al., 2018; Baninajjar et al., 2023). The exact techniques face fundamental challenges in scalability due to their exhaustive exploration of all potential behaviors, often leading to exponential complexity (Katz et al., 2017). On the other hand, the over-approximation-based approaches prioritize scalability over precision, but are still limited by the inherent trade-off between scalability and precision.

Other studies have sought to establish links between robustness and the architecture of DNNs. Lin et al. (2019) demonstrated that binarization enhances robustness by keeping the noise magnitude small. Another study revealed that quantized DNNs exhibit greater scalability, but their accuracy is compromised as they disregard floating-point values (Henzinger et al., 2021). In (Sietsma & Dow, 1988), it was also shown that robustness against adversarial attacks improves if redundant neurons are eliminated. Guidotti et al. (2020) utilized pruning to reduce the size of DNNs by eliminating non-crucial portions that do not significantly impact their performance, but without any hard performance/robustness guarantees. Tao et al. (2023) presented an approach for architecture-preserving, provable V-polytope repair of DNNs. However, as opposed to the state-of-the-art studies above, our main focus in this paper is not on targeting specific properties, e.g., robustness; rather, the main goal of VNNs is to enhance verifiability, while guaranteeing prediction performance/robustness requirements.

Manngård et al. (2018) aimed at enhancing sparsity using the Lagrange multiplier and regularizing the loss function with robustness requirements. However, when a constrained optimization problem is reformulated as an unconstrained optimization problem using the Lagrange multiplier, the final solution obtained may violate the original hard constraints. Leofante et al. (2023) introduced parametric ReLU to reduce over-approximation and improve verification-friendliness, however, this method does not provide any guarantees to generate more verification-friendly DNNs nor does it provide guarantees to satisfy performance/robustness requirements. Hu et al. (2024) introduces a new residual architecture tailored for training neural networks with robustness certification but, similar to the previous work, it lacks any guarantees.

5. Conclusions

The state-of-the-art verification tools currently face major challenges in terms of scalability. In this paper, we presented an optimization framework to generate a new class of DNNs, referred to as VNNs, that are guaranteed to be accommodating to formal verification techniques. VNNs are more time-efficient and verification-friendly while maintaining on-par prediction performance with their DNN counterparts. We formulate an optimization problem on pre-trained DNNs to obtain VNNs while maintaining on-par prediction performance. Our experimental evaluation based on MNIST, CHB-MIT, and MIT-BIH datasets demonstrates that the robustness of our proposed VNNs is established substantially more often and in a more time-efficient manner than their DNN counterparts, without any major loss in terms of prediction performance. In addition, our experiments show that VNNs are not only more verification-friendly, but also more robust compared to their peer DNNs.

Acknowledgements

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

Impact Statement

This paper presents work whose goal is to investigate the important task of providing formal guarantees for DNNs, which presents a significant challenge in the AI/ML domain. This is particularly important in the context of safety-critical medical applications, e.g., epileptic seizure detection and real-time cardiac arrhythmia detection, as we show in this paper. Failure to detect an epileptic seizure or a cardiac arrhythmia episode in time may have irreversible consequences and potentially lead to death. In response to this challenge, this study introduces a framework designed to produce DNNs that align with verification techniques, thereby enhancing the trustworthiness of these networks without sacrificing the prediction performance/robustness. Finally, we would like to highlight that there are no potential ethical impacts and future societal implications/consequences to be highlighted here.

References

Baninajjar, A., Hosseini, K., Rezine, A., and Aminifar, A. Safedeep: A scalable robustness verification framework for deep neural networks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Bunel, R. R., Turkaslan, I., Torr, P., Kohli, P., and Mudigonda, P. K. A unified view of piecewise linear neural network verification. *Advances in Neural Information Processing Systems*, 31, 2018.

Chou, Y.-L., Moreira, C., Bruza, P., Ouyang, C., and Jorge, J. Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications. *Information Fusion*, 81:59–83, 2022.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. MIT-BIH Arrhythmia Database, 2000. URL https://www.physionet.org/content/mitdb/1.0.0/.

Guidotti, D., Leofante, F., Pulina, L., and Tacchella, A. Verification of neural networks: Enhancing scalability through pruning. In *Proceedings of ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 2505–2512. IOS Press, 2020.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

Henzinger, T. A., Lechner, M., and Zikelic, D. Scalable verification of quantized neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 3787–3795, 2021.

Hu, K., Zou, A., Wang, Z., Leino, K., and Fredrikson, M. Unlocking deterministic robustness certification on imagenet. *Advances in Neural Information Processing Systems*, 36, 2024.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. Safety verification of deep neural networks. In *Computer-Aided Verification (CAV)*, pp. 3–29. Springer, 2017.

Huang, X., Kroening, D., Ruan, W., Sharp, J., Sun, Y., Thamo, E., Wu, M., and Yi, X. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37, 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 *Computer-Aided Verification (CAV)*, 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 *Computer-Aided Verification (CAV)*, pp. 443–452. Springer, 2019.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pp. 99–112. Chapman and Hall/CRC, 2018.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun.com/exdb/mnist/*, 1998.

Leofante, F., Henriksen, P., and Lomuscio, A. Verification-friendly networks: the case for parametric relus. In *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2023.

Liang, H., He, E., Zhao, Y., Jia, Z., and Li, H. Adversarial attack and defense: A survey. *Electronics*, 11(8):1283, 2022.

Lin, J., Gan, C., and Han, S. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444*, 2019.

Manngård, M., Kronqvist, J., and Böling, J. M. Structural learning in artificial neural networks using sparse optimization. *Neurocomputing*, 272:660–667, 2018.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582. IEEE, 2016.

Panelli, R. J. Sudep: a global perspective. *Epilepsy & Behavior*, 103:106417, 2020.

Shoeb, A. CHB-MIT Scalp EEG Database, 2010. URL https://physionet.org/content/chbmit/.

Sietsma and Dow. Neural net pruning-why and how. In *IEEE 1988 international conference on neural networks*, pp. 325–333. IEEE, 1988.

Singh, G., Ganvir, R., Püschel, M., and Vechev, M. Beyond the single neuron convex barrier for neural network certification. *Advances in Neural Information Processing Systems*, 32, 2019a.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages (PACMPL)*, 3:1–30, 2019b.

Sopic, D., Aminifar, A., and Atienza, D. e-glass: A wearable system for real-time detection of epileptic seizures. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2018.

Stroobandt, R. X., Duytschaever, M. F., Strisciuglio, T., Van Heuverswyn, F. E., Timmers, L., De Pooter, J., Knecht, S., Vandekerckhove, Y. R., Kucher, A., and Tavernier, R. H. Failure to detect life-threatening arrhythmias in icds using single-chamber detection criteria. *Pacing and Clinical Electrophysiology*, 42(6):583–594, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tao, Z., Nawas, S., Mitchell, J., and Thakur, A. V. Architecture-preserving provable repair of deep neural networks. *Proceedings of the ACM on Programming Languages*, 7(PLDI):443–467, 2023.

Tjeng, V., Xiao, K., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming, 2017.

Ugare, S., Singh, G., and Misailovic, S. Proof transfer for fast certification of multiple approximate neural networks. *Proceedings of the ACM on Programming Languages*, 6 (OOPSLA1):1–29, 2022.

Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Daniel, L., Boning, D., and Dhillon, I. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning (ICML)*, pp. 5276–5285. PMLR, 2018.

Zass, R. and Shashua, A. Nonnegative sparse pca. *Advances in neural information processing systems*, 19, 2006.