Global Robustness Verification Networks

Weidi Sun  
School of Mathematical  
Peking University  
Haidian Qu 100871,Beijing, China  
weidisun@pku.edu.cn

Coauthor  
Yuteng Lu  
School of Mathematical  
Peking University  
Haidian Qu 100871,Beijing, China  
1701210111@pku.edu.cn

Coauthor  
Xiyue Zhang  
School of Mathematical  
Peking University  
Haidian Qu 100871,Beijing, China  
zhangxiyue@pku.edu.cn

Coauthor  
Zhanxing Zhu  
School of Mathematical  
Peking University  
Haidian Qu 100871,Beijing, China  
zhanxing.zhu@pku.edu.cn

Coauthor  
Meng Sun  
School of Mathematical  
Peking University  
Haidian Qu 100871,Beijing, China  
Center for Quantum Computing, Peng Cheng Laboratory  
Shenzhen Shi 518055, China  
sunmeng@math.pku.edu.cn

Abstract

The wide deployment of deep neural networks, though achieving great success in many domains, has severe safety and reliability concerns. Existing adversarial attack generation and automatic verification techniques cannot formally verify whether a network is globally robust, i.e., the absence or not of adversarial examples in the input space. To address this problem, we develop a global robustness verification framework with three components: 1) a novel rule-based “back-propagation” finding which input region is responsible for the class assignment by logic reasoning; 2) a new network architecture Sliding Door Network (SDN) enabling feasible rule-based “back-propagation”; 3) a region-based global robustness verification (RGRV) approach. Moreover, we demonstrate the effectiveness of our approach on both synthetic and real datasets.

1 Introduction

Deep Neural Networks (DNNs) have been applied to a variety of domains and achieved great success. Reliance on DNNs’ decisions makes their behavior reliability of high importance. Recent research has shown that the safety of DNNs is threatened by their susceptibility to human-imperceptible adversarial perturbations [14, 3, 1].

To explore the adversarial robustness of neural networks, two aspects have been considered: crafting adversarial examples and automatic verification. Given an input sample, adversarial example generation techniques [13, 2, 10, 15, 6] fail to guarantee that no adversarial example exists around

Preprint. Under review.
the given input, when they cannot generate adversarial examples for it. The efforts in automatic
testification mainly focus on the guarantee of local robustness [9, 5, 16, 11, 3], i.e., the robustness of
an input’s neighborhood. These verification approaches can provide a rigorous local robustness proof
if adversarial examples do not exist in a local region. However, the local robustness only takes a
small part of the input space into account, and thus cannot guarantee reliability of the whole network
for every possible input.

Some attempts have been made towards the verification and evaluation of global robustness, i.e.,
finding out whether no adversarial example exists in the input space of a network [12, 6]. Though
the SMT/SAT-based method in [6] takes global robustness into account, the definition for global
robustness in [6] cannot be satisfied by the inputs near the classification boundary. In other words, no
network can satisfy this definition. The technique developed in [12] evaluates the local robustness
of each sample in a test dataset and treats the expected value of evaluation results as the indicator
of “global robustness”. The technique in [12] can be considered as finding expected maximum safe
radius over the test dataset. Thus, the selection of the test dataset directly influences the estimation
in [12], and the global robustness cannot be formally guaranteed in general. We can easily identify
two stumbling blocks on the path of global robustness verification: the complex activation patterns
and large input space. It is computationally prohibited to analyze all possible activation patterns or
traverse input space to guarantee global robustness. Thus existing testing and verification techniques
are infeasible to handle the global robustness verification for DNNs.

In this paper, we develop a feasible global verification framework with three components: 1) a
novel rule-based “back-propagation” which is used for mapping classification rules from output to
input to find which input region is responsible for the corresponding class assignment [1], 2) a new
network design Sliding Door Network (SDN) that enables feasible rule-based “back-propagation”, 3) a region-based global robustness verification (RGRV) approach by finding “adversarial regions”.

Particularly, we address the “two stumbling blocks” by two means. Firstly, we design a new activation
function Sliding Door Activation (SDA), with which the number of possible activation patterns is
dramatically reduced to circumvent the complexity issue. Secondly, instead of treating a single input
as the foundation “atom” of global robustness analysis, we cluster the input space into multiple
classification regions to address the input space explosion challenge. To the best of our knowledge,
this is the first work that can achieve global robustness formally with only slight drop of classification
accuracy compared with classic DNNs. We evaluate the effectiveness of our framework on the
MNIST [8] dataset. We also design a synthetic case study to show the feasibility of our global
verification method.

The rest of this paper is structured as follows. We introduce the rule-based “back-propagation” in
Section 2. The network design and the corresponding rule-based back-propagation method are
described in Section 3. Section 4 presents the RGRV approach. We evaluate the usefulness of SDN
and effectiveness of RGRV in Section 5. Section 6 summarizes our work.

2 Rule-based back-propagation for DNNs

Due to the colossal input space of DNNs, exhausting all possible inputs with traditional testing
methods is infeasible. Thus we develop a family of classification rules to divide the input space
into several regions. These regions can simplify the global robustness verification significantly. The

Figure 1: A DNN with two hidden lay-
ers and one output layer.

Note that this “back-propagation” is entirely different from typical use of back-propagation for evaluating
the gradient with respect to the weight parameters in DNNs.
classification rules in the input space could be achieved by the proposed rule-based back-propagation, as elaborated below. Before introducing the classification rules in detail, we first present a warm up example.

**Example 1.** Considering the one-layer network in Figure 2(a), a classification rule in the output space is \((y_0 > y_1)\) which represents a blue region in output space shown as Figure 2(b). The back-propagation we proposed aims for mapping classification rules to the input space. For example, the activation pattern “all neurons are active” means that \((y_0 = y'_0 \land y_1 = y'_1 \land y'_0 > 0 \land y'_1 > 0)\). As \((y_0 = y'_0 \land y_1 = y'_1 \land y'_0 = x_1 \land y'_1 = x_0)\), \((y_0 > y_1)\) is equivalent to \((x_1 > x_0)\) and \((y'_0 > 0 \land y'_1 > 0)\) is equivalent to \((x_1 > 0 \land x_0 > 0)\). Thus the mapping result of output space \((y_0 > y_1)\) to input space is \((x_0 > 0 \land x_1 > 0 \land x_1 > x_0)\). If we change the activation pattern to "\(y_0\) is active and \(y_1\) is inactive", the equivalent condition of this activation pattern is \((y_0 = y'_0 \land y_1 = 0 \land y'_0 > 0 \land y'_1 < 0)\), because ReLU assigns 0 to \(y_1\). Thus \((y'_0 > 0 \land y'_1 < 0)\) is equivalent to \((x_1 > 0 \land x_0 < 0)\); \((y_0 > y_1)\) is equivalent to \((x_1 > 0)\); the mapping result is \((x_0 < 0 \land x_1 > 0)\). Obviously, the activation pattern determines the mapping result. The mapping result \((x_0 > 0 \land x_1 > 0 \land x_1 > x_0)\) represents a blue region in input space shown in Figure 2(c). We name this blue region as classification region, indicating its responsibility to the class assignment.

With the intuition from the warm up example, we now elaborate the rule-based back-propagation layer by layer for deep neural networks.

There are many inequations recorded as \(\mathbb{P}^{h}_{\gamma \eta}\) in \(\text{Layer}^h\). These inequations make up the disjunctive normal form \(\lor \land \mathbb{P}^{h}_{\gamma \eta}\), which describes how neural networks classify the inputs. For simplicity, if we select a \(\mathbb{P}^{h}_{\gamma \eta}\) in \(\text{Layer}^h\) this \(\mathbb{P}^{h}_{\gamma \eta}\) will be denoted as \(\mathbb{P}^h = \sum c_i x_i^h + b > 0\). We can easily provide the classification rules in output space. For example, “the output belongs to class \(k\)” is \(\land_{j(j \neq k)} y_k > y_j\) where \(y_j\) (0 \(\leq j < m\)) are the output values and \(m\) is the number of classes for output. To obtain the classification rules in input space, we take a typical DNN\(^2\) as an example and propose a back-propagating function. During the back-propagation from \(\text{Layer}^h\) to \(\text{Layer}^{h-1}\), the \(\mathbb{P}^h\) should be substituted and the \(\land\)s and \(\lor\)s should be retained to \(\text{Layer}^{h-1}\), thus we apply our function to each \(\mathbb{P}^h\) instead of the conjunctive normal form. The recursive call of this function can back-propagate the classification rules layer by layer to the input space of the network. Since the output layer and the hidden layers should be treated in different ways, we divide the function into two parts: the output and hidden layer part.

The comparison rules like \(y_k > y_j\) can be directly mapped to the corresponding pre-activation layer based on the order-preserving activation function, e.g., softmax, of the output layer. We replace every variable \(y_j\) in inequations with the corresponding polynomial \(\sum \omega_{ij} y_j x_i^{h-1} + b_j^h\) to obtain the classification rule in \(\text{Layer}^{h-1}\). Thus the output layer part is the function

\[
\text{MAP-OUT}(y_k > y_j) = \sum \omega_{ik} y_i x_i^{h-1} + b_k^h > \sum \omega_{ij} y_j x_i^{h-1} + b_j^h
\]

and the mapping result of \(\land_{j(j \neq k)} y_k > y_j\) is \(\land_{j(j \neq k)} \text{MAP-OUT}(y_k > y_j)\), i.e., the classification rules in \(\text{Layer}^{h-1}\).

The hidden layer part is a function \(\text{MAP-HIDDEN}\). Since each linear inequation in \(L^h\) can be simplified into the form \(\sum c_i x_i^h + b > 0\), we select an inequation \(\mathbb{P}^h = \sum c_i x_i^h + b > 0\) as the input of \(\text{MAP-HIDDEN}\) to show how \(\text{MAP-HIDDEN}\) works. The hidden layers cannot be processed in the same way as output layer because of the activation patterns which determine the mapping result. We denote the set of active neurons’ indexes by \(\Theta\) and use \(\Theta\) to represent the activation pattern. For

\(^2\)The activation functions in hidden layers and output layer of this DNN are ReLU and softmax, respectively.
simplicity, we record the mapping result of $P^h$ under $\Theta^h$ as $MAP\text{-}FIX(\Theta^h, P^h)$ where $\Theta^h$ is the activation pattern of Layer$^h$. $MAP\text{-}FIX(\Theta^h, P^h)$ is the conjunction of some classification rules in Layer$^{h-1}$.

The function $MAP\text{-}HIDDEN$ is shown as follows where $\Delta^h$ denotes all activation patterns of Layer$^h$:

$$MAP\text{-}HIDDEN(\Delta^h, P^h) = \bigvee_{\Theta^h \in \Delta^h} MAP\text{-}FIX(\Theta^h, P^h)$$  \hspace{1cm} (2)

The mapping result of $MAP\text{-}HIDDEN$ is the disjunction of all the classification rules in Layer$^{h-1}$. As each neuron has two activation states, there are $2^{m^h}$ activation patterns in $\Delta^h$ where $m^h$ is the number of neurons in Layer$^h$. The time cost of whole back-propagation is $O(\Pi h 2^{m^h}) = O(2 \sum h m^h)$. Such immense time cost makes the above mapping NP-hard and infeasible.

## 3 Sliding Door Network for Feasible Back-propagation

To handle the complexity issue, we present a novel network design, SDN, and the corresponding rule-based back-propagation method $M_{SDN}$. SDN reduces the size of $\Delta$ by grouping the neurons in each layer to overcome the infeasibility problem in back-propagating classification rules for DNNs.

### 3.1 Sliding Door Network

Compared with typical DNNs, SDN has two different components: a novel activation function $SDA$ and the loss function design for supporting $SDA$.

**Sliding Door Activation.** $SDA$ takes a pre-activation layer into account and divides neurons into several groups evenly. For example, the layer $L^h$ in Figure 3 with 10 neurons is divided into 5 groups which are represented as $G^h_0 \{x^h_i \mid 2j \leq i < 2j + 2 \}$ $(0 \leq j < 5)$. These groups are classified by $SDA$ into three categories: active group with all positive neurons (e.g., $G^h_1$ and $G^h_4$ in Figure 3), inactive group in which all neurons are negative (e.g., $G^h_3$ in Figure 3), and trivial groups with mixing of both positive and negative neurons (i.e., $G^h_0$ and $G^h_2$ in Figure 3).

In order to reduce the complexity, we select the first active (inactive) group as active (inactive) door for each pre-activation layer. For example in Figure 3, $G^h_1$ and $G^h_3$ are active door and inactive door respectively. Based on the assigned doors, we define $SDA$ as:

$$x^h = SDA(x^h) = \begin{cases} 0 & \text{if } x^h_i \text{ belongs to inactive door;} \\ \alpha x^h_i & \text{if } x^h_i \text{ belongs to active door;} \\ x^h_i & \text{otherwise.} \end{cases}$$  \hspace{1cm} (3)

To increase the network expressiveness, $SDA$ strengthens the active door by $\alpha$ and assigns 0 to inactive door’s neurons. Other groups are sent to the corresponding activation layer directly. During the training, for each pre-activation layer, the position of the two doors might change instantly up to the states of the groups, behaving like a sliding door, thus the name of our activation function. Figure 4 shows the entire network architecture, replacing the ReLU in classic DNNs with the proposed $SDA$ for each layer.

**Loss function design.** If a pre-activation layer cannot provide active or inactive door, the expressiveness of SDN will be weakened. To avoid this issue, we design regularization term to penalize the absence of either of the two doors. If the active (inactive) door does not appear in $L^h$, we will find the group $G^h_\alpha (G^h_\beta)$ in $L^h$ with most
active (inactive) neurons, and adjust the weights to make the negative (positive) neurons in \( G^h \) tend to be positive (negative) so as to create active (inactive) groups. Thus, besides the typical data fitting loss, we add a regularization term to encourage the emergence of such groups, defined as:

\[
L(W, b) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{h} \left( \sum_{x_i^h \in G^h, x_i^h < 0} (-x_i^h) + \sum_{x_i^h \in G^h, x_i^h > 0} x_i^h \right)
\]

where \( \{W, b\} \) denotes all the weights and biases to be trained, and \( \lambda \) is the user-given penalty parameter.

3.2 Rule-based Back-propagation for SDN

As MAP-OUT can be reused for the back-propagation of SDN’s output layer, we focus on back-propagation between hidden layers in this section. The constructing process of MAP-HIDDEN for SDN is as follows.

We denote the set of neurons in the active door of layer \( L^h \) as \( \Theta^h_A \), the set of neurons in the inactive door as \( \Theta^h_I \), and other neurons are in \( \Theta^h_T \). Considering the condition that a rule in \( L^h \) is \( \Phi^h = \sum_i c_i x_i^h + b > 0 \), and the activation pattern \( \Theta^h \) is fixed where \( \Theta^h = \{ \Theta^h_A, \Theta^h_I, \Theta^h_T \} \), we record the mapping result of \( \Phi^h \) under \( \Theta^h \) as MAP-FIX(\( \Theta^h, \Phi^h \)) with three components:

1) SDNInherit.

\[
SDNInherit = \sum_{i \in \Theta^h_A} a_{ci} \left( (\sum_{i} \omega_i x_i^h - 1) + b_i^h \right) + \sum_{i \in \Theta^h_I} a_{ci} \left( (\sum_{i} \omega_i x_i^h + 1) + b_i^h \right) + b > 0
\]

where we replace the neurons in \( \Phi^h \) belonging to \( \Theta^h_A \) with the corresponding polynomial, i.e., \( \sum_{i} \omega_i x_i^h - 1 + b_i^h \) multiplied by \( a \) due to SDA activation, the neurons in \( \Theta^h_T \) with the corresponding polynomial meanwhile remove the neurons in \( \Theta^h_I \) to obtain SDNInherit.

2) SDNActiveCon.

\[
SDNActiveCon = \bigwedge_{i \in \Theta^h_A} ((\sum_{i} \omega_i x_i^h - 1) + b_i^h > 0)
\]

It describes the rules that “all the corresponding pre-activation neurons of \( \Theta^h_A \) are greater than 0” and we replace these pre-activation neurons with corresponding polynomial.

3) SDNInactiveCon.

\[
SDNInactiveCon = \bigwedge_{i \in \Theta^h_I} ((\sum_{i} - \omega_i x_i^h - 1) - b_i^h > 0)
\]

It describes the rules that “all the corresponding pre-activation neurons in \( \Theta^h_I \) are less than 0” and we replace these pre-activation neurons with corresponding polynomial. The function MAP-FIX is shown as follows:

\[
MAP-FIX(\Theta^h, \Phi^h) = SDNInherit \land SDNActiveCon \land SDNInactiveCon
\]

Taking all the activation patterns into account, we can obtain the function MAP-HIDDEN.

\[
MAP-HIDDEN(\Delta^h, \Phi^h) = \bigvee_{\Theta^h \in \Delta^h} MAP-FIX(\Theta^h, \Phi^h)
\]

The combination of MAP-OUT and MAP-HIDDEN is the complete back-propagating function \( M_{SDN} \):

\[
M_{SDN}(\Delta^h, \Phi^h) = \begin{cases} MAP-OUT(\Phi^h) & h = H \\ MAP-HIDDEN(\Delta^h, \Phi^h) & h < H \end{cases}
\]

Thus the mapping result of all the rules \( \bigvee_\gamma \land_\eta \Phi^h_\gamma \eta \) in Layer\(^h \) is \( \bigvee_\gamma \land_\eta M_{SDN}(\Delta^h, \Phi^h_\gamma \eta) \) which is the collection of rules for Layer\(_{\gamma-1} \). Each \( |\Delta^h| \) equals to \( m_h(m_h - 1) + 2m_h + 1 \) where \( m_h \) is the number of groups in \( L^h \). The SDN has \( O(\Pi, m_h^2) \) activation patterns which is greatly less than the
The classification region of With the description of classification graph and limiting ball, we now provide the formal definition of explicit rules. We will show that An adversarial example exists in two types of regions: 1) the region which is isolated, 2) the isolated connected component and Vol
\[ \text{vol}_i \]
(Region-based Global Robustness Analysis)

In this section, we define and address the global robustness verification problem by a region-based global robustness analysis (RGRA) approach. Firstly, we provide the definition for global robustness:

**Definition 1 (Global robustness).** Given a network \( N \), if there is no adversarial example in its input space, \( N \) is a globally robust network.

An adversarial example exists in two types of regions: 1) the region which is isolated, 2) the region which is connected to the correctly classified region. Taking Figure 5 as an example, the adversarial examples belonging to \( C_1 \) but classified as \( C_2 \) tend to exist in \( R_1 \) and \( R_2 \). Before presenting the definitions of protruding region and small-size isolated connected components, we first present a formal definition of the input space classification graph.

**Definition 2 (Classification graph).** Given a result of backpropagation in the form of \( \gamma \cap \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \gamma \γ
The first step of finding the “adversarial regions” is to construct the adjacency relationship between vertexes, i.e., building the classification graph. The construction can be split into two phases: 1) traverse the vertexes \( v_i \) in the classification graph, and treat \( v_j (j \in \{t|t > i\}) \) as the potential adjacent vertexes; 2) find the common boundary shared by \( v_i \) and \( v_j \) if they are adjacent, and provide the proof that they are not adjacent otherwise. Intuitively, a common boundary between \( v_i \) and \( v_j \) means that they stick together on the boundary, formally defined as follows.

**Definition 6 (Common boundary).** Given two vertexes \( v_i \) and \( v_j \) in the classification graph, the corresponding classification regions are \( S_i \) and \( S_j \). A boundary \( \mathbb{F} ^\prime (\sum c_i x_i^0 + b = 0) \) which belongs to \( v_i \) and \( v_j \) is defined as a common boundary iff

\[
\exists x_0 \exists c_i > 0 \forall x \neq x_0 ( x \in \mathbb{F} ^\prime) \land I \cap \mathbb{H} = \emptyset \
\forall x \in (x \in S_i \lor x \in S_j) \land I \cap S_i \neq \emptyset \land I \cap S_j \neq \emptyset
\]

where \( \mathbb{H} = \{ x| \sum c_i x_i^0 + b = 0 \} \) is the set of points on \( \mathbb{F} ^\prime \), \( \mathbb{H} ^\prime \) is the set of points on other boundaries \( \mathbb{F} ^\prime \) of \( v_i \) and \( v_j \), and \( I = \{ x||x - x_0||_\infty < \varepsilon_0 \} \).

As the classification graph is an undirected graph, the first phase only explores the edge \( (v_j, v_j) \) of \( j > i \) to find the potential adjacent vertex \( v_j \). And the following Theorem 1 implies that by traversing the explicit rules of \( v_j \), we can check all the \( v_j (j \in \{t|t > i\}) \) which have a common boundary with \( v_i \), i.e., all the adjacent vertexes.

**Theorem 1.** Given \( v_i \) and \( v_j (j > i) \), there is a common boundary belonging to the explicit rules of \( v_i \) shared by them iff \( v_i \) and \( v_j \) are adjacent.

Proof: see appendix.

Finally, the global robust verification can be achieved by the following steps:

1. Build classification graph. We can obtain the vertexes from the back-propagation result. Each vertex has some boundaries. For each boundary \( \mathbb{F} \) of \( v_i \), we select some points on it randomly and sample in the tiny neighborhood of these points. By feeding the samples into SDN, we can find out their activation patterns. For example, a sample’s activation pattern represents \( v_j \) as this sample is in the tiny neighborhood of points on \( \mathbb{F} \), \( \mathbb{F} \) belongs to \( v_j \). Thus \( v_i \) and \( v_j \) shares the boundary \( \mathbb{F} \) and we add edge \( (v_i, v_j) \) to the edge set \( E \).

2. Find the limiting ball of classification regions and connected components. For a classification region \( S \), we can obtain the rough upper and lower bound of each dimension. Taking the inequality \( \sum x_i^0 > 0 \) as an example, if \( c_i > 0 \) and \(-2 \leq x_i \leq 2 \), the lower bound of \( x_i^0 > -(\sum_{i|c_i > 0} 2c_i + \sum_{i|c_i < 0} -2c_i) + b/c_i \). All these bounds form a “box” including \( S \). We take samples \( u_t (0 \leq t < n) \) in this “box” and select the samples in \( S \). \( o = \sum u_t/n \) is the estimated center. Denote \( d(u_t, o) \) as the distance between \( u_t \) and \( o \), the estimated radius is \( \max x_k(d(u_t, o)) \). Moreover, we can calculate the volume of the “box” \( \text{vol}(\text{box}) \), if there are \( m \) samples in \( S \), \( \text{vol}(s) = \frac{m}{n} \text{vol}(\text{box}) \). If a connected component consists classification regions \( S_k \)s, the estimated center is \( o = \frac{1}{\sum x_k \text{vol}(S_k)} \sum x_k \text{vol}(S_k) \) and \( \max x_k(d(u_k, o)) \) is the estimating radius where \( u_k \)s are the samples in \( S_k \).

3. Find small-size isolated connected components and protruding regions. Given a connected component, by comparing the radius of limiting ball with the \( R \) given by user, we can find out whether this connected component is a small-size isolated connected component. For a classification region belonging to class \( C \), we take \( n \) samples in the limiting ball. If \( m \) of them belong to \( C \), calculate \( n/m \) and compare it with the user given \( r \). Then we can find out whether this region is a protruding region.

5 Experiments

The evaluation of our work concentrates on three aspects: 1) the utility of SDN on classification tasks, 2) generating adversarial examples, 3) the feasibility of global verification. In the first part, we evaluate our method on MNIST database with typical DNNs as baseline. In the second part, we show the adversarial examples generated from adversarial regions. In the third part, we design a synthetic case study to show the effectiveness of RGRV.
5.1 Utility on classification task

Due to the reduction of activation patterns, the expressive ability of SDN is slightly inferior to typical DNNs with the same architecture. We empirically show the drop of classification accuracy is acceptable based on two groups of case studies. The first group is the comparison of typical DNNs and SDNs on MNIST dataset. The details of SDNs are shown as follows.

Table 1: Classification results on MNIST where sat-rate is the frequency of layers which can provide both active door and inactive door in evaluation

| MNIST    | (16,12) | (24,18) | (32,24) | (40,30) |
|----------|---------|---------|---------|---------|
| SDN accuracy(%) | 95.08   | 95.24   | 95.32   | 95.45   |
| sat-rate(%)   | 81.64   | 85.25   | 93.49   | 96.37   |
| DNN accuracy(%) | 97.80   | 98.12   | 98.21   | 98.28   |

1) Each SDN has two hidden layers Layer$^1$ and Layer$^2$; 2) These SDNs have 16, 24, 32, 40 groups in Layer$^1$, respectively and 12, 18, 24, 30 groups in Layer$^2$, respectively. We name these SDNs as (16,12), (24,18), (32,24), and (40,30) based on their architecture features; 3) Each group in Layer$^1$ has four neurons, and each group in Layer$^2$ has two neurons. 4) The $\alpha$ in these SDNs are set as 2.

Each baseline DNN has two hidden layers. The number neurons in each layer is the same as corresponding SDNs. Cross entropy loss and Adam [7] are used to train all the networks 1500 epochs with batch size 256. The evaluation result is shown in Table 1. Compared with typical DNNs, the accuracy of SDN only drops 2.72, 2.88, 2.89, 2.83 percent respectively. Besides, the accuracy of SDN and the sat-rate increase as the numbers of groups in each layer increase.

5.2 Generating adversarial examples

Since our verification method is aimed at global verification, i.e., finding all the adversarial regions in the input space, the generation of attacks is only a by-product. As this generation is not based on local or testing way, it is meaningless to compare its efficiency with state-of-the-art attack generation approaches like [13, 2, 10, 15, 6]. Given parameters $(R, r) = (0.04, 0.2)$, we show the adversarial examples in SDN (20,20) and point out the corresponding adversarial regions.

The digits in the first line of Figure 6 classified as “2” are the adversarial examples of the digits below them classified as “0, 6, 8, 4, 9” respectively. Take 4 as an example, the upper 4 is in “[[18,1],[1,15]]”, denoting the activation pattern where $G_{18}^1$ and $G_{1}^2$ are active doors, $G_{1}^1$ and $G_{15}^2$ are inactive doors. By inputting “[[18,1],[1,15]]”, we can find a group of inequations returned by our algorithm, and the conjunction of these inequations represents the corresponding classification region of “[[18,1],[1,15]]”. The lower 4 classified as 4 is in the activation pattern “[[4,10],[1,3]]”. The classification region of “[[18,1],[1,15]]” is the protruding region found by our method which is close to “[[4,10],[1,3]]’s” classification region. Moreover, we have found out that there is only one connected component of class 2 in the input space. Obviously, it is easy to select a large number of adversarial examples in the adversarial regions.

5.3 Feasibility of precise global verification

As our work is the first global robustness verification work, there is no baseline method existing for this case study. To draw an exact conclusion whether the results of the proposed method are correct, we train a SDN on a two-dimensional synthetic dataset shown in Figure 7(a). In Figure 7(a) the big blue region at the top-right belongs to the first class and the small blue region at the lower-left which is the “noise” in this dataset belongs to the first class as well. The points in the white region belong to the second class. The classification results of the trained SDN are visually shown in Figure 7(b) where the points in the blue regions belong to the first class and the points in the white region belong
to the second class. Our method has found the adversarial regions in the orange circles. Obviously, these regions containing adversarial examples are what we do not want. The verification result shows that this SDN is not globally robust.

6 Conclusion

In this paper, we present a novel global verification framework. To the best of our knowledge, this is the first work that provides a complete solution to achieving global robust verification for neural networks. Based on the proposed rule-based back-propagation, we analyze the relationship between activation patterns and classification rules, and thus design a new network SDN. Together with a region-based global robustness analysis approach, the verification can be finished in an acceptable duration, dramatically reducing computational complexity. Our evaluation shows that the SDN can perform comparable with classic DNNs, and favorable, the global verification can be achieved, which is unattainable in the past. We hypothesize that SDN is suitable for safety-critical fields, especially for some classification tasks which are not very complex but eager for strict robustness. Developing verification framework for large-scale networks is our further research direction.

7 Appendix

We assign serial numbers to each activation pattern to store the mapping result in a B+ tree for the convenience of global verification:

$$\Theta.number = \sum_{h}^{n-1} (g_h * group.num_h + g'_h - 1) \text{ if } (g_h < g'_h) \quad \text{else } (g_h * group.num_h + g'_h)$$

$$\Theta.number = \Pi_{j=h+1}^n |\Delta'.j| + (g_n * group.num_n + g'_n)$$

where \(n\) is the number of layers, \(h\) represents Layer\(_h\), \(g_h(g'_h)\) is the index of Layer\(_h\)'s active(inactive) door, and \(group.num_h\) is the number of groups in Layer\(_h\). Two activation patterns \(\Theta_1\) and \(\Theta_2\) satisfy \(\Theta_1.number < \Theta_2.number\) iff

$$\{(\exists k \forall h < k) \: \Theta_1.g_h = \Theta_2.g_h \land \Theta_1.g'_h = \Theta_2.g'_h\} \land \{\Theta_1.g_k < \Theta_2.g_k \lor (\Theta_1.g_k = \Theta_2.g_k \land \Theta_1.g'_k < \Theta_2.g'_k)\}$$

To prove Theorem 1 we need Lemma 1 and Lemma 2.

Lemma 1. Given two vertexes \(v_i\) and \(v_j(j > i)\), they are adjacent iff they share a common boundary.

Proof. Given two adjacent vertexes \(v_i\) and \(v_j(j > i)\), the corresponding classifications are \(S_i\) and \(S_j\). According to the definition of adjacent in Definition 2 there is a \(x_0\) satisfies

$$\exists x_0 \forall x' \in I \{x' \in S_i \lor x' \in S_j\} \land (I \cap S_i \neq \emptyset) \land (I \cap S_j \neq \emptyset)$$

where \(I = \{x' \mid ||x' - x_0||_\infty < \varepsilon\}\)

There must be a \(x\) in \(I\) on a boundary \(F\) which satisfies \(\exists x' \forall x_0 \neq x' C \cap \mathbb{H}_i = \emptyset\) where \(C = \{x' \mid ||x' - x||_\infty < \varepsilon_1\}\). \(\mathbb{H}\) and \(\mathbb{H}_i\) are the point set of boundary \(F\) and \(F_i\). Otherwise we can find a series of points \(u_i(i \geq 0)\) and corresponding balls \(I_i = \{x \mid ||x - u_i||_\infty < \sigma_i\}\) satisfying:

- \(u_i\) is on the boundary \(F_i\) and \(\forall x_0 \mathbb{H}_i + 1 \cap I_i \neq \emptyset\)
- \(u_{i+1} \in I_i, u_{i+1}\) on \(F_{i+1}, I_{i+1} \subseteq I_i\) and \(I_{i+1} \cap \mathbb{H}_i = \emptyset\)
Since we have only finite number of boundaries, here comes the contradiction. Hence we can find a $u_k$ in $F$ which is a boundary of both $v_i$ and $v_j$ and a corresponding $I_k$ satisfies the above condition. As $u_k$ is on the boundary, it satisfies $\forall x \in I_k \ (x \in S_i \lor x \in S_j) \land I_k \cap S_i \neq \emptyset \land I_k \cap S_j \neq \emptyset$ \\
Thus $u_k$ satisfies \\
$\exists \epsilon > 0 \ [y \in \mathbb{B} - H_k I_k \cup H' = \emptyset \land \forall x \in I_k \ (x \in S_i \lor x \in S_j) \land I_k \cap S_i \neq \emptyset \land I_k \cap S_j \neq \emptyset]$ \\
and $F$ is a common boundary shared by $v_i$ and $v_j$.

The sufficient condition is obvious.  

**Lemma 2.** Given two adjacent vertexes $v_i$ and $v_j (j > i)$, the shared common boundary of them comes from the explicit rules of $v_i$.

**Proof.** The activation pattern $\Theta_i$ and $\Theta_j$ of $v_i$ and $v_j$ satisfy $\Theta_i.number = i > \Theta_j.number = j$ thus \\
$\forall t (t < k) \Theta_i.g_t = \Theta_j.g_t \land \Theta_i.g'_t = \Theta_j.g'_t \checkmark$ \\
$\{\Theta_i.g_t < \Theta_j.g_k \lor (\Theta_i.g_k = \Theta_j.g_k \land \Theta_i.g'_k < \Theta_i.g'_k)\}$

This indicates the first change of door happens on $L_i^{t_k}$. The change of activation pattern leads to the “mutation” of neurons in $L_i^{t} (t > k)$. However the change of neurons in $L_i^{t} (t \leq k)$ is continuous. According to the proof of Lemma[1] we can find a path in $D_{\alpha}$ which crosses and only crosses the $F$. Thus When the point on this path approaches the boundary, there is at most one inactive neuron corresponding to $F$ approaches $0$ in $L_i^{t} (t \leq k)$. Here we consider where the $F$ comes from:

- $F$ comes from $Layer_i(t > k)$. The change of sign of $x_i^{t_k}$ will not influence the activation pattern in $L_i^{t_k}$ which contradicts to “the first change of door happens on $L_i^{t_k}$,”

- $F$ comes from $Layer_i(t < k)$. If it changes the activation pattern, it contradicts to “the first change of door happens on $L_i^{t_k}$”. Otherwise, there must be another inactive neuron in $L_i^{k}$ changes the sign at the same time which contradicts to “there is at most one inactive neuron corresponding to $F$ approaches $0$ in $L_i^{t} (t \leq k)$”

- $F$ comes from implicit rules in $Layer_k$. If the activation pattern does not change, here comes the contradiction to “the first change of door happens on $L_i^{t_k}$”. If the activation pattern changes, according to the definition of implicit rules, either the index of active door $g_k$ or inactive door $g'_k$ will become smaller, that is to say, the result is a activation pattern $v_u(u < i)$ instead of $v_j$. Thus we have a contradiction

We eventually find out that $F$ comes from explicit rules of $v_i$ by a process of elimination.

Based on Lemma[1] and Lemma[2] the correctness of Theorem[3] is obvious.

**References**

[1] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *CoRR*, abs/1709.00609, 2017.

[2] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57, 2017.

[3] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. A12: safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 3–18, 2018.
[4] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[5] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I, pages 3–29, 2017.

[6] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I, pages 97–117, 2017.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[8] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/. Accessed January 4, 2020.

[9] Matthew Mirman, Timon Gehr, and Martin T. Vechev. Differentiable abstract interpretation for provably robust neural networks. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, pages 3575–3583, 2018.

[10] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016, pages 372–387, 2016.

[11] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, pages 2651–2659, 2018.

[12] Wenjie Ruan, Min Wu, Youcheng Sun, Xiaowei Huang, Daniel Kroening, and Marta Kwiatkowska. Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, pages 5944–5952, 2019.

[13] Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. CoRR, abs/1908.07000, 2019.

[14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014.

[15] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. Feature-guided black-box safety testing of deep neural networks. In Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I, pages 408–426, 2018.

[16] Min Wu, Matthew Wicker, Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. A game-based approximate verification of deep neural networks with provable guarantees. Theor. Comput. Sci., 807:298–329, 2020.