OccRob: Efficient SMT-Based Occlusion Robustness Verification of Deep Neural Networks

Xingwu Guo [1], Ziwei Zhou [1], Yueling Zhang [1], Guy Katz [2], Min Zhang [1](✉)

[1] Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China
zhangmin@sei.ecnu.edu.cn
[2] The Hebrew University of Jerusalem, Jerusalem, Isarel

Abstract. Occlusion is a prevalent and easily realizable semantic perturbation to deep neural networks (DNNs). It can fool a DNN into misclassifying an input image by occluding some segments, possibly resulting in severe errors. Therefore, DNNs planted in safety-critical systems should be verified to be robust against occlusions prior to deployment. However, most existing robustness verification approaches for DNNs are focused on non-semantic perturbations and are not suited to the occlusion case. In this paper, we propose the first efficient, SMT-based approach for formally verifying the occlusion robustness of DNNs. We formulate the occlusion robustness verification problem and prove it is NP-complete. Then, we devise a novel approach for encoding occlusions as a part of neural networks and introduce two acceleration techniques so that the extended neural networks can be efficiently verified using off-the-shelf, SMT-based neural network verification tools. We implement our approach in a prototype called OccRob and extensively evaluate its performance on benchmark datasets with various occlusion variants. The experimental results demonstrate our approach's effectiveness and efficiency in verifying DNNs' robustness against various occlusions, and its ability to generate counterexamples when these DNNs are not robust.

1 Introduction

Deep neural networks (DNNs) are computer-trained *programs* that can implement hard-to-formally-specify tasks. They have repeatedly demonstrated their potential in enabling artificial intelligence in various domains, such as face recognition [6] and autonomous driving [27]. They are increasingly being incorporated into safety-critical applications with interactive environments. To ensure the security and reliability of these applications, DNNs must be highly dependable against adversarial and environmental perturbations. This dependability property is known as *robustness* and is attracting a considerable amount of research efforts from both academia and industry, aimed at ensuring robustness via different technologies such as adversarial training [13,28], testing [40,33], and formal verification [34,10,5].

Occlusion is a prevalent kind of perturbation, which may cause DNNs to misclassify an image by occluding some segment thereof [38,25,8]. For instance, a "turn left" traffic sign may be misclassified as "go straight" after it is occluded by a tape, probably resulting in traffic accidents. A similar situation may occur in face recognition, where many well-trained neural networks fail to recognize faces correctly when they are partially occluded, such as when glasses are worn[37]. A neural network is called *robust against occlusions*

S. Sankaranarayanan and N. Sharygina (Eds.): TACAS 2023, LNCS 13993, pp. 208–226, 2023.
https://doi.org/10.1007/978-3-031-30823-9_11

if small occlusions do not alter its classification results. Generally, we wish a DNN to be robust against occlusions that appear negligible to humans.

It is challenging to verify whether a DNN is robust or not on an input image if the image is occluded. On the one hand, the verification problem is non-convex due to the non-linear activation functions in DNNs. It is NP-complete even when dealing with common, fully connected feed-forward neural networks (FNNs) [20]. On the other hand, unlike existing perturbations, occlusions are challenging to encode using L_p norms. Most existing robustness verification approaches assume that perturbations need to be defined by L_p norms and then apply approximations and abstract interpretation techniques [34,10,5] as part of the verification process. The semantic effect of occlusions partially alters the values of some neighboring pixels from large to small or in the inverse direction, e.g., 255 to 0, when a black occlusion occludes a white pixel. Therefore, existing techniques for perturbations in L_p norms are not suited to occlusion perturbations.

SMT-based approaches have been shown to be an efficient approach to DNN verification [20]. They are both sound and complete, in that they always return definite results and produce counterexamples in non-robust cases. We show that, although it is straightforward to encode the occlusion robustness verification problem into SMT formulas, solving the constraints generated by this naïve encoding is experimentally beyond the reach of state-of-the-art SMT solvers, due to the inclusion of a large number of the piece-wise ReLU activation functions. Consequently, such a straightforward encoding approach cannot scale to large networks.

In this paper, we systematically study the occlusion robustness verification problem of DNNs. We first formalize and prove that the problem is NP-complete for ReLU-based FNNs. Then, we propose a novel approach for encoding various occlusions and neural networks together to generate new equivalent networks that can be efficiently verified using off-the-shelf SMT-based robustness verification tools such as Marabou [21]. In our encoding approach, although additional neurons and layers are introduced for encoding occlusions, the number is reasonably small and independent of the networks to be verified. The efficiency improvement of our approach comes from the fact that our approach significantly reduces the number of constraints introduced while encoding the occlusion and leverages the backend verification tool's optimization against the neural network structure. Furthermore, we introduce two acceleration techniques, namely input-space splitting to reduce the search space of a single verification, which can significantly improve verification efficiency, and label sorting to help verification terminates earlier. We implement a tool called OccRob with Marabou as the backend verification tool. To our knowledge, this is the first work on formally verifying the occlusion robustness of deep neural networks.

To demonstrate the effectiveness and efficiency of OccRob, we evaluate it on six representative FNNs trained on two benchmark datasets. The empirical results show that our approach is effective and efficient in verifying various types of occlusions with respect to the occlusion position, size, and occluding pixel value.

Contributions. We make the following three major contributions: (i) we propose a novel approach for encoding occlusion perturbations, by which we can leverage *off-the-shelf* SMT-based robustness verification tools to verify the robustness of neural networks against various occlusion perturbations; (ii) we prove the verification problem of the occlusion robustness is NP-complete and introduce two acceleration techniques, i.e., label sorting and input space splitting, to improve the efficiency of verification further; and (iii) we implement a tool called OccRob and conduct experiments extensively on a collection of benchmarks to demonstrate its effectiveness and efficiency.

Paper Organization. Sec. 2 introduces preliminaries. Sec. 3 formulates the occlusion robustness verification problem and studies its complexity. Sec. 4 presents our encoding approach and acceleration techniques for the verification. Sec. 5 shows the experimental results. Sec. 6 discusses related work, and Sec. 7 concludes the paper.

We omit the complete proofs and experimental results due to the page limit. Please refer to the technical report [15] for more details.

2 Preliminaries

2.1 Deep Neural Networks and the Robustness

As shown in Fig. 1, a deep neural network consists of multiple layers. The neurons on the input layer take input values, which are computed and propagated through the hidden layers and then output by the output layer. The neurons on each layer are connected to those on the predecessor and successor layers. We only consider fully connected, feedforward networks (FNNs) [11].

Given a λ-layer neural network, let $W^{(i)}$ be the weight matrix between the $(i-1)$-th and i-th layers, and $\mathsf{b}^{(i)}$ be the biases of the corresponding neurons, where $i = 1, 2, \ldots, \lambda$. The network implements a function $F : \mathbb{R}^u \rightarrow \mathbb{R}^r$ that is recursively defined by:

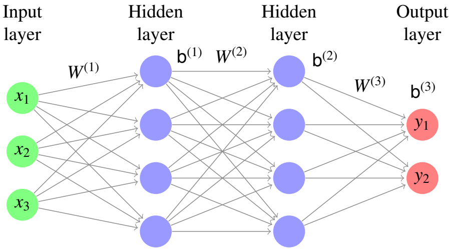


Fig. 1: A fully-connected feed-forward neural network (FNN).

$$\begin{aligned} z^{(0)} &= x \\ z^{(i)} &= \sigma(W^{(i)} \cdot z^{(i-1)} + \mathsf{b}^{(i)}),\ for\ i = 1, \ldots, \lambda - 1 \end{aligned} \qquad \text{(Layer Function)}$$

$$F(x) = W^{(\lambda)} \cdot z^{(\lambda-1)} + \mathsf{b}^{(\lambda)} \qquad \text{(Network Function)}$$

where $\sigma(\cdot)$ is called an *activation function* and $z^{(i)}$ denotes the result of neurons at the i-th layer.

For example, Fig. 1 shows a 3-layer neural network with three input neurons and two output neurons, namely, $\lambda = 3$, $u = 3$ and $r = 2$.

For the sake of simplicity, we use $\Phi_F(x) = arg\ max_{\ell \in L} F(x)$ to denote the label ℓ such that the probability $F_\ell(x)$ of classifying x to ℓ is larger than those to other labels, where L represents the set of labels. The activation function σ usually can be a piece-wise Rectified Linear Unit (ReLU), $\sigma(x) = max(x, 0)$), or S-shape functions like Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$, Tanh $\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, or Arctan $\sigma(x) = tan^{-1}(x)$. In this work, we focus on the networks that only contain ReLU activation functions, which are widely adopted in real-world applications.

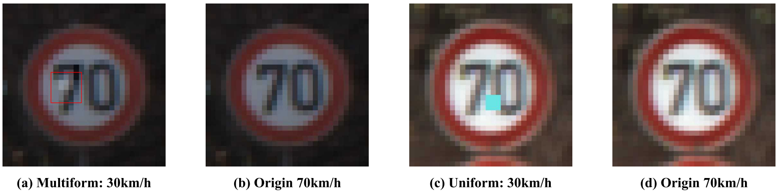

(a) Multiform: 30km/h
(b) Origin 70km/h
(c) Uniform: 30km/h
(d) Origin 70km/h

Fig. 2: Two multiform and uniform occlusions to traffic signs causing mis-classifications.

A neural network is called *robust* if small perturbations to its inputs do not alter the classification result [39]. Specifically, given a network F, an input x_0 and a set Ω of perturbed inputs of x_0, F is called locally robust with respect to x_0 and Ω if F classifies all the perturbed inputs in Ω to the same label as it does x_0.

Definition 1 (Local Robustness [17]). *A neural network $F : \mathbb{R}^u \to \mathbb{R}^r$ is called locally robust with respect to an input x_0 and a set Ω of perturbed inputs of x if $\forall x \in \Omega, \Phi_F(x) = \Phi_F(x_0)$ holds.*

Usually, the set Ω of perturbed inputs is defined by an ℓ_p-norm ball around x_0 with a radius of ϵ, i.e., $\mathbb{B}_p(x_0, \epsilon) := \{x \mid \|x - x_0\|_p \leq \epsilon\}$ [17,2].

2.2 Occlusion Perturbation

In the context of image classification networks, occlusion is a kind of perturbation that blocks the pixels in certain areas before the image is fed into the network. Existing studies showed that the classification accuracy of neural networks could be significantly decreased when the input objects are artificially occluded [23,44].

Occlusions can have various occlusion shapes, sizes, colors, and positions. The shapes can be square, rectangle, triangle, or irregular shape. The size is measured by the number of occluded pixels. The occlusion color specifies the colors occluded pixels can take. The coloring of an occlusion can be either uniform, where all occluded pixels share the same color, or multiform, where these colors can vary in the range of $[-\epsilon, \epsilon]$, where ϵ specifies the threshold between an occluded pixel's value and its original value.

Prior studies [8,3] showed that both the uniform and multiform occlusions could cause misclassification to neural networks. Fig. 2 shows two examples of multiform and uniform occlusions, respectively. The traffic sign for "70km/h speed limit" in Fig. 2(a) is misclassified to "30km/h" by adding a 5 × 5 multiform occlusion. Fig. 2(d) shows another sign, with different light conditions, where a 3 × 3 uniform occlusion (in Fig. 2(c)) causes the sign to be misclassified to "30km/h".

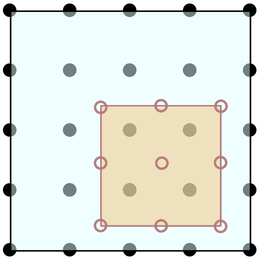


Fig. 3: An example occlusion on a 5 × 5 image at real number position.

The occlusion position is another aspect of defining occlusions. An occlusion can be placed precisely on the pixels of an image, or between a pixel and its neighbors. Fig. 3 shows an example, where the dots represent image pixels and the circles are the occluding pixels that will substitute the occluded ones. We say that an occlusion pixel $\vartheta_{i',j'}$ at location (i', j') surrounds an image pixel $p_{i,j}$ at location (i, j) if and only if $|i - i'| < 1$ and $|j - j'| < 1$. Note that i', j' are real numbers, representing the location where the occlusion pixel o is placed on the image. An image pixel can be occluded by the substitute occlusion pixels if the occlusion pixels surround the image pixel.

There are at most four surrounding occlusion pixels for each image pixel, as shown in Fig. 3. Let $\mathbb{I}_p$ be the set of the locations where the surrounding occlusion pixels of p are placed. After the occlusion, the value of pixel $p_{i,j}$ is altered to the new one denoted by $p'_{i,j}$, which can be computed by interpolation [19,22] such as next neighbour interpolation or Bi-linear interpolation based on occlusion pixels in $\mathbb{I}_p$. Besides that, we use a method based on L_1-distance to calculate how much a pixel is occluded. Since the L_1-distance of two adjacent pixels is 1, a surrounding occlusion pixel should not affect the image pixel if their L_1-distance is greater than 1. The formula $max(0, (|1 - i' + i|) + (1 - j' + j) - 1)$ indicates how much an image pixel at (i, j) is occluded by an occlusion pixel at (i', j'). For instance, occlusion pixel at $(i', j') = (0.9, 0.9)$ has no effect to image pixel $(i, j) = (0, 0)$ since their L_1-distance is larger than 1. Therefore, the occlusion factor $s_{i,j}$ for pixel p at (i, j) can be calculated based on all surrounding occlusion pixels in $\mathbb{I}_p$ as:

$$s_{i,j} = max(0, \textstyle\sum_{i'_0, j' \in \mathbb{I}_p}(|1 - j + j'|) + \sum_{i', j'_0 \in \mathbb{I}_p}(|1 - i' + i|) - 1) \tag{1}$$

where (i'_0, j'_0) is the first element of $\mathbb{I}_p$. Notably, s is 1 for completely occluded pixel and 0 for the pixel that is not occluded, otherwise s has a value between $(0, 1)$. Also, it is a special case for Equation 1 when (i', j') are integers, where s can be reduced to 0 or 1.

3 The Occlusion Robustness Verification Problem

Let $\mathbb{R}^{m\times n}$ be the set of images whose height is m and width is n. We use $\mathbb{N}_{1,m}$ (*resp.* $\mathbb{N}_{1,n}$) to denote the set of all the natural numbers ranging from 1 to m (*resp.* n). A coloring function $\zeta : \mathbb{R}^{m\times n} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a mapping of each pixel of an image to its corresponding color value. Given an image $x \in \mathbb{R}^{m\times n}$, $\zeta(x, i, j)$ defines the value to color the pixel of x at (i, j).

Definition 2 (Occlusion function). *Given a coloring function ζ and an occlusion ϑ of size $w \times h$, the occlusion function is defined as function $\gamma_{\zeta,w\times h} : \mathbb{R}^{m\times n} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{m\times n}$ such that $x' = \gamma_{\zeta,w\times h}(x, a, b)$ if for all $i \in \mathbb{N}_{1,n}$ and $j \in \mathbb{N}_{1,m}$, there is,*

$$x'_{i,j} = x_{i,j} - s_{i,j} \times (x_{i,j} - \zeta(x, i, j)), \tag{2}$$

$$where,\ \zeta(x, i, j) = \frac{\sum_{(i',j')\in\mathbb{I}_{x_{i,j}}} \vartheta_{i',j'} \sqrt{(i - i')^2 + (j - j')^2}}{\sum_{(i',j')\in\mathbb{I}_{x_{i,j}}} \sqrt{(i - i')^2 + (j - j')^2}}. \tag{3}$$

s in Equation 2 is the occlusion factor for pixel at (i, j) as mentioned in Sec. 2.2. Note that when i', j' are integers, Equation 2 can be reduced to $x_{i,j} = \vartheta_{i,j}$, which represents that $x_{i,j}$ is completely occluded by the occlusion. In other words, the integer case is a special case of the real number case. Also, when pixel at (i, j) is not occluded, since $s_{i,j} = 0$. In this case, Equation 2 can be reduced to $x'_{i,j} = x_{i,j}$.

Interpolation is handled by ζ showed in Equation 3. It shows the standard form for the color of the new $x'_{i,j}$. A unique color value is specified for all the pixels in the occluded area for a uniform occlusion. Therefore, ζ in Equation 3 can be reduced to $\zeta(x, i, j) = \mu$ for some $\mu \in [0, 1]$. The coloring function in a multiform occlusion is defined as $\zeta(x, i, j) = x_{i,j} + \Delta_p$ with $\Delta_p \in [-\epsilon, \epsilon]$, where $\epsilon \in \mathbb{R}$ defines the threshold that a pixel can be altered.

Definition 3 (Local occlusion robustness). *Given a DNN $F : \mathbb{R}^{m\times n} \to \mathbb{R}^r$, an occlusion function $\gamma_{\zeta,w\times h} : \mathbb{R}^{m\times n} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{m\times n}$ with respect to coloring function ζ and occlusion size $w \times h$, and an input image x, F is called local occlusion robust on x with $\gamma_{\zeta,w\times h}$ if $\Phi_F(x) = \Phi_F(\gamma_{\zeta,w\times h}(x, a, b))$ holds for all $1 \leq a \leq n$ and $1 \leq b \leq m$.*

Intuitively, Definition 3 means that F is robust on x against the occlusions of $\gamma_{\zeta,w\times h}$, if on any occluded image of x by the occlusion function $\gamma_{\zeta,w\times h}$, F always returns the same classification result as on the original image x. Depending on the coloring function ζ, the definition applies to various occlusions concerning shapes, colors, sizes, and positions. We can also extend the above definition to the global occlusion robustness if F is robust on all images concerning $\gamma_{\zeta,w\times h}$.

We prove that even for the case of uniform occlusion, a special case of the multiform one, the local occlusion robustness verification problem is NP-complete on the ReLU-based neural networks.

4 SMT-Based Occlusion Robustness Verification

4.1 A Naïve SMT Encoding Method

The verification problem of FNNs' local occlusion robustness can be straightforwardly encoded into an SMT problem. In Definition 3, we assume that x is classified by Φ to the label ℓ_q, i.e., $\Phi(x) = \ell_q$, for a label $\ell_q \in L$. To prove F is robust on x after x is occluded by occlusion ϑ with size $w \times h$, it suffices to prove that F classifies every occluded image $x' = \gamma_{\zeta,w\times h}(a, b)$ to ℓ_q for all $1 \leq a \leq n$ and $1 \leq b \leq m$. This is equivalent to proving that the following constraints are not satisfiable:

$$1 \leq a \leq n, 1 \leq b \leq m, \tag{4}$$

$$\begin{aligned}&\textstyle\bigwedge_{i\in\mathbb{N}_{1,n}, j\in\mathbb{N}_{1,m}}\\ &\quad\Big(((a-1 < i < a+w+1) \wedge (b-1 < j < b+h+1) \wedge x'_{i,j} = \gamma_{\zeta,w\times h}(x, a, b)_{i,j}) \vee\\ &\quad((i \geq a+w+1) \vee (i \leq a-1) \vee (j \geq b+h+1) \vee (j \leq b-1)) \wedge x'_{i,j} = x_{i,j})\Big),\end{aligned} \tag{5}$$

$$\textstyle\bigvee_{l\in\mathbb{N}_{1,q-1}\cup\mathbb{N}_{q+1,r}} F(x')_l \geq F(x')_q. \tag{6}$$

The conjuncts in Eq. 5 define that x' is an occluded instance of x, and the disjuncts in Eq. 6 indicate that, when satisfiable, there exists some label ℓ_i which has a higher probability than ℓ_q to be classified to. Namely, the occlusion robustness of F on x is falsified, with x' being a witness of the non-robustness. Note that this naive encoding

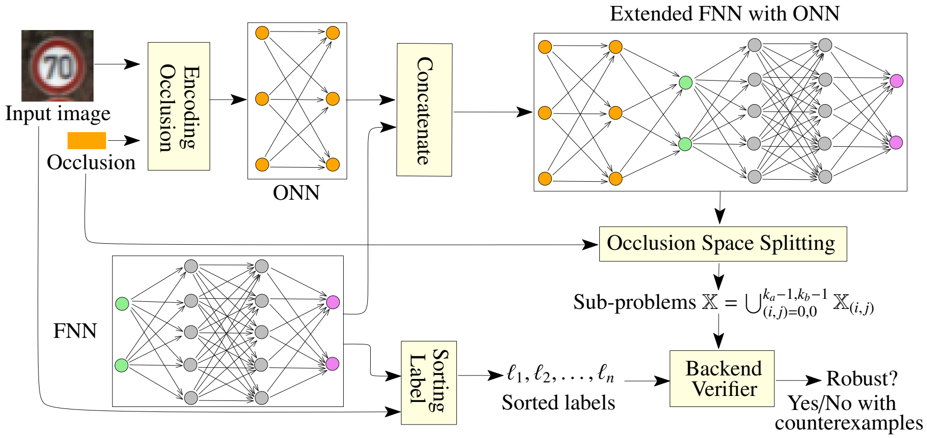

Extended FNN with ONN
Input image
Occlusion
Encoding Occlusion
ONN
Concatenate
Occlusion Space Splitting
Sub-problems $\mathbb{X} = \bigcup_{(i,j)=0,0}^{k_a-1,k_b-1} \mathbb{X}_{(i,j)}$
FNN
Sorting Label
$\ell_1, \ell_2, \ldots, \ell_n$
Sorted labels
Backend Verifier
Robust?
Yes/No with counterexamples

Fig. 4: The workflow of encoding and verifying FNN's robustness against occlusions.

considers the occlusion position's real number cases since function γ implicitly includes the interpolation.

Although the above encoding is straightforward, solving the encoded constraints is experimentally beyond the reach of general-purpose existing SMT solvers due to the piece-wise linear ReLU activation functions in the definition of F in the constraints of Eq. 6, and the large search space $m \times n \times (2\epsilon)^{w \times h}$ (see Experiment II in Sec. 5).

4.2 Our Encoding Approach

An Overview of the Approach. To improve efficiency, we propose a novel approach for encoding occlusion perturbations into four layers of neurons and concatenating the original network to these so-called *occlusion layers*, constituting a new neural network which can be efficiently verified using state-of-the-art, SMT-based verifiers.

Fig. 4 shows the overview of our approach. Given an input image and an occlusion, we first construct a 3-hidden-layer occlusion neural network (ONN) and then concatenate it to the original FNN by connecting the ONN's output layer to the FNN's input layer. The combined network represents all possible occluded inputs and their classification results. The robustness of the constructed network can be verified using the existing SMT-based neural network verifiers.

We introduce two acceleration techniques to speed up the verification further. First, we divide the occlusion space into several smaller, orthogonal spaces, and verify a finite set of sub-problems on the smaller spaces. Second, we employ the eager falsification technique [14] to sort the labels according to their probabilities of being misclassified to. The one with a larger probability is verified earlier by the backend tools. Whenever a counterexample is returned, an occluded image is found such that its classification result differs from the original one. If all sub-problems are verified and no counterexamples are found, the network is verified robust on the input image against the provided occlusion.

Encoding Occlusions as Neural Networks. Given a coloring function ζ, an occlusion size $w \times h$ and an input image x of size $m \times n$, we construct a neural network $O : \mathbb{R}^{4+ct} \to \mathbb{R}^{m \times n}$ to encode all the possible occluded images of x, where $c = 1$ if x is a grey image and $c = 3$ if x is an RGB image, $t = 0$ for the uniform occlusion and $t = w \times h$ for the multiform one.

Fig. 5 shows the neural network architecture for encoding occlusions. We divide it into a fundamental part and an additional part. The former encodes the occlusion position and the uniform occlusion color. The additional part is needed only by the multiform occlusion to encode the coloring function. Without loss of generality, we assume that the input layer takes the vector (a, w, b, h, ζ), where (a, b) is the top-left coordinate of occlusion area in x. The coloring function ζ is admitted by other $c \times t$ neurons in the input layer when the occlusion is multiform.

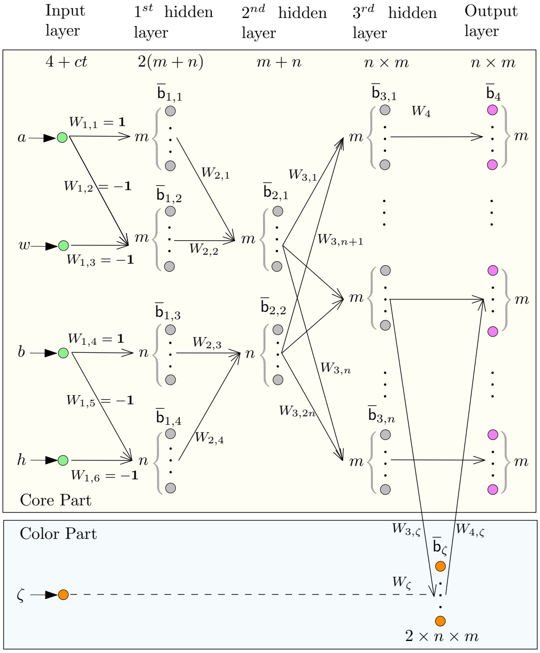


Fig. 5: An occlusion neural network for the occlusions on an image x with ζ and $w \times h$.

(1) Encoding occlusion positions. We explain the weights and biases that are defined in the neural network to encode the occlusion position. On the connections between the input layer and the first hidden layer, the weights in matrices $W_{1,1}$, $W_{1,2}$ and $W_{1,3}$ are 1, -1 and -1, respectively. Note that we hide all the edges whose weights are 0 in the figure for clarity. The biases in $\bar{b}_{1,1}$ are $(-1, -2, \ldots, -m)$ for the first m neurons on the first hidden layer. Those in $\bar{b}_{1,2}$ are $(2, 3, \ldots, m + 1)$. The weights in $W_{1,4}$, $W_{1,5}$, $W_{1,6}$ and the biases in $\bar{b}_{1,3}$ and $\bar{b}_{1,4}$ are defined in the same way. We omit the details due to the page limitation.

For the second layer, the diagonals of weight matrices $W_{2,1}$ to $W_{2,4}$ are set to -1, and the rest of their entries are 0. The biases in $\bar{b}_{2,1}$ and $\bar{b}_{2,2}$ are 1. After the propagation to the second hidden layer, a pixel at position (i, j) in the image x is occluded if and only if both the outputs of the i^{th} neuron in the first m neurons and the j^{th} neuron in the remaining n neurons on the second hidden layer are 1.

The third hidden layer represents the occlusion status of each pixel in the original image x. $2n$ weight matrices connect the second layer and the $n \times m$ neurons of the third layer. For example, we consider the weights in $W_{3,i}$ and $W_{3,n+i}$ which connect the i^{th} group of m neurons in the third layer to the second layer. The size of $W_{3,i}$ is $m \times m$, and the weights in the i^{th} row are 1 while the rest is 0. The size of $W_{3,n+i}$ is $m \times n$. The weights on its diagonal are set to 1, while the rest are set to 0. All the biases in $\bar{b}_{3,1}$ to $\bar{b}_{3,n}$ are -1. The output of the third layer indicates the occlusion status of all the pixels. If a pixel at (i, j) is occluded, then the output of the $(i \times m + j)^{th}$ neuron in the third layer is 1, and otherwise, 0.

(2) Encoding Coloring Functions. We consider the uniform and multiform coloring functions separately for verification efficiency, although the former is a special case of the latter. We first consider the general multiform case. In the multiform case, we introduce $2 \times n \times m$ extra neurons in the third hidden layer, as shown in the bottom part of Fig. 5. These neurons can be combined with the third layer, but it would be more clear to separate them. The weight matrix $W_{3,\zeta}$ connects the third layer to these neurons, with its first half of diagonal set to 1, and the second half set to -1. This helps retain the sign of the input ζ during propagation. The weight matrix W_{ζ} connects the input ζ to these neurons, whose diagonal are 1, and the biases $\bar{\mathsf{b}}_{\zeta}$ are -1. These neurons work just like the third layer, except that they not only represent the occlusion status of pixels, but also preserve the input ζ. If a pixel at (i, j) is occluded and ζ has a positive value, then the $(i \times m + j)^{th}$ output in the first half of them is ζ. The $(i \times m + j)^{th}$ output in the second half is ζ when ζ has a negative value. Otherwise, the output is 0. In the uniform case, it can be encoded together with input images, and we thus explain it in the following paragraph.

(3) Encoding Input Images. In the fourth layer, we use W_4 to denote the weight matrix connecting the third layer. W_4 is used to encode pixel values of the input image x and the coloring function ζ of occlusions. In the uniform case, the weight $\mathsf{w}(i, i)$ in the diagonal of W_4 is $\mathsf{w}(i, i) = \zeta_i - x_i$ and the biases $\bar{\mathsf{b}}_4 = \mathbf{x}$ where $\mathbf{x}$ is the flattened vector of the original input image. In the multiform case, the weight matrix $W_{4,\zeta}$ connects the neurons in the bottom part that preserves information of input ζ to the fourth layer. The first half of $W_{4,\zeta}$ is identical to W_4, and the second half of $W_{4,\zeta}$ has its diagonal set to -1. It provides the value of the coloring function ζ with any sign for each occluded pixel. The output of the j^{th} neuron in the i^{th} group of the fourth layer is the raw pixel value plus ζ if the pixel at (i, j) is occluded; otherwise, it is the raw pixel value of p.

An Illustrative Example. We show an example of constructing the occlusion network on a 2×2, single-channel image in Fig. 6. In this example, we assume that the input image is $x = [0.4, 0.6, 0.55, 0.72]$ and the occlusion applied to x has a size of 1×1, which means $w = 1$ and $h = 1$. For uniform occlusion, the coloring function ζ has a fixed value of 0, and for multiform case, the threshold ϵ that a pixel can be altered is 0.1.

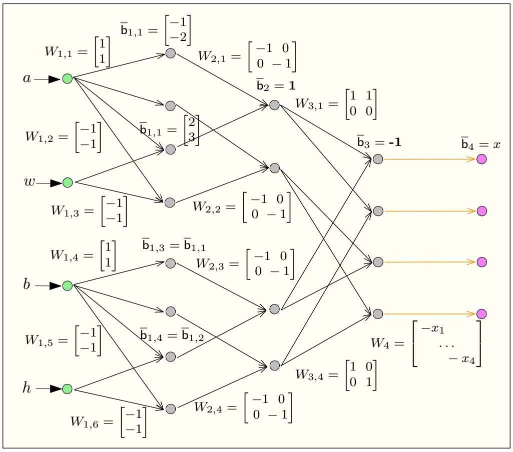

$\bar{\mathsf{b}}_{1,1} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$
$W_{1,1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
$W_{2,1} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
a
$\bar{\mathsf{b}}_2 = \mathbf{1}$
$W_{3,1} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$
$W_{1,2} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$
$\bar{\mathsf{b}}_{1,1} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$
$\bar{\mathsf{b}}_3 = \textbf{-1}$
$\bar{\mathsf{b}}_4 = x$
w
$W_{1,3} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$
$W_{2,2} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
$W_{1,4} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
$\bar{\mathsf{b}}_{1,3} = \bar{\mathsf{b}}_{1,1}$
$W_{2,3} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
b
$W_{1,5} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$
$\bar{\mathsf{b}}_{1,4} = \bar{\mathsf{b}}_{1,2}$
$W_4 = \begin{bmatrix} -x_1 & & \\ & \cdots & \\ & & -x_4 \end{bmatrix}$
$W_{3,4} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
h
$W_{2,4} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
$W_{1,6} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$

Fig. 6: An example of encoding a one-pixel uniform occlusion as a neural network.

We suppose the occlusion is applied at position $(1, 2)$, which means $a = 1$ and $b = 2$ for the input of occlusion network. In the forward propagation, we calculate the output of the first layer by $a \times W_{1,1} + \bar{\mathsf{b}}_{1,1}$ and $a \times W_{1,2} + b \times W_{1,3} + \bar{\mathsf{b}}_{1,2}$ and can get $(0, 0, 0, 1)$ for the first four neurons. Following the same process, we get the output of the second 4 neurons, $(1, 0, 0, 0)$. After propagation to the second layer, it outputs $(1, 0), (0, 1)$ based on $W_{2,1}$, $W_{2,2}$ and $\bar{\mathsf{b}}_2$, representing the second column and the first row of x is under occlusion. Likely, the third layer outputs $(0, 1, 0, 0)$ based on its weight matrices and biases, representing that the second pixel in the first row is occluded. After propagation to the fourth layer, the occlusion network outputs an occluded image $x' = [0.4, 0, 0.55, 0.72]$ based on W_4 and $\overline{\mathsf{b}}_4$. It is identical to the expected occluded image, where the second pixel is occluded, and other pixels stay unchanged. Suppose we change a to some real number, for instance, 1.5. After the same propagation, we will get an output of $(0, 0.5, 0, 0.5)$ in the third layer, representing that the neurons in the second column are affected by the occlusion by a factor of 0.5. The fourth layer then outputs $[0.4, 0.3, 0.55, 0.36]$, which is the corresponding occluded image x'.

In the multiform case, as mentioned at the first, we suppose the threshold $\epsilon = 0.1$, and keep all other settings. Then after the same propagation to the third layer, the third layer would output $(0, 1, 0, 0)$, representing that the second pixel is occluded. Those extra neurons then output $(0, 0.1, 0, 0, 0, 0, 0, 0)$ where the second neuron in the first half is 0.1 and 0 for the remaining. This indicates both that the second pixel in the first row is occluded, and has an epsilon of 0.1. After propagation to the fourth layer, the occlusion network outputs $x' = [0.4, 0.7, 0.55, 0.72]$ based on its W_4 and $\overline{\mathsf{b}}_4$. As expected, the second pixel is occluded and increases by 0.1, and other pixels stay unchanged. For the case of a negative ϵ of -0.1, the extra neurons output $(0, 0, 0, 0, 0, 0.1, 0, 0)$. Note that the second neuron in the second half is 0.1 and the remaining are 0, which helps retain the sign of -0.1. The fourth layer then outputs $[0.4, 0.5, 0.55, 0.72]$, which is the expected occluded image where the second pixel decreases by 0.1.

4.3 The Correctness of the Encoding

Given an input image x, a rectangle occlusion of size $w \times h$, and a coloring function ζ, let O be the corresponding occlusion neural network constructed in the approach above. Let F be the FNN to verify. We concatenate O to F by connecting O's output layer to F's input layer. The combined network implements the composed function $F \circ O$. The problem of verifying the occlusion robustness of F on the input image x is reduced to a regular robustness verification problem of $F \circ O$.

Theorem 1 (Correctness). *An FNN F is robust on the input image x with respect to a rectangle occlusion in the size of $w \times h$ and a coloring function ζ if and only if $\Phi_{F\circ O}((a, w, b, h, \zeta)) = \Phi_F(x)$ for all $1 \leq a \leq n$ and $1 \leq b \leq m$.*

Theorem 1 means that all the occluded images from x are classified by F to the same label as x, which implies the correctness of our proposed encoding approach. To prove Theorem 1, it suffices to show that the encoded occlusion neural network represents all the possible occluded images. In other words, when being perceived as a function, the network outputs the same occluded image as the occlusion function for the same occlusion coordinate (a, b), as formalized in the following lemma.

Lemma 1. *Given an occlusion function $\gamma_{\zeta,w\times h} : \mathbb{R}^{m\times n} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{m\times n}$ and an input image x, let $O_{\gamma,x} : \mathbb{R}^{4+ct} \to \mathbb{R}^{m\times n}$ be the corresponding occlusion neural network. There is $\gamma_{\zeta,w\times h}(x, a, b) = O_{\gamma,x}(a, w, b, h, \zeta)$ for all $1 \leq a \leq n$ and $1 \leq b \leq m$.*

Proof (Sketch). It suffices to prove $\gamma_{\zeta,w\times h}(x, a, b)_{i,j} = O_{\gamma,x}(a, w, b, h, \zeta)_{i,j}$ for all $i \in \mathbb{N}_{1,n}$ and $j \in \mathbb{N}_{1,m}$. By Definition 2, we consider the following two cases:

Case 1: When a pixel p at position (i, j) is fully occluded, we have $\gamma_{\zeta,w\times h}(x, a, b)_{i,j} = \zeta(x, i, j)$. We need to prove that $O_{\gamma,x}(a, w, b, h, \zeta)_{i,j} = \zeta(x, i, j)$.

Suppose p is covered by an arbitrary uniform occlusion with size of $w_0 \times h_0$ at position (a_0, b_0). We can observe that for that pixel p, $i > a_0 \wedge i < a_0 + w_0 - 1$ and $j > b_0 \wedge j < b_0 + h_0 - 1$ hold since p is covered by the occlusion.

We show the output of $O_{\gamma,x}(a, w, b, h, \zeta)_{i,j}$ by inspecting the $(i * n + j)^{th}$ output of the occlusion network after propagation, starting from inspecting the output of the i^{th} and $(i + m)^{th}$ neurons of the first layer. According to the network structure discussed in Sec. 4.2, we can tell that the i^{th} neuron in the first layer is 0 only when $i > a_0$, the same property holds for the $(i + m)^{th}$ neuron when $i < a_0 + w_0 - 1$. Therefore, the output for the i^{th} and $(i + m)^{th}$ neurons of the first layer is 0, which leads to the i^{th} neuron in the first part of the second layer has output of value 1. Through the similar process, we can get that the value of $z^{(2)}_j$ in the second part of the second layer is also 1.

The $(i \times n + j)^{th}$ neuron in the third layer is based on the i^{th} neuron and j^{th} neuron of the second layer that we just discussed. Therefore, the output of that neuron, $z^{(3)}_{i\times n+j}$, is 1. For uniform occlusion, suppose the coloring function ζ has a fixed value μ_0. By propagating the output $z^{(3)}_{i\times n+j}$ to the fourth layer, which is calculated as $W_4 \times z^{(3)} + \overline{\mathsf{b}}_4$, the $(i \times n + j)^{th}$ output of the fourth layer is $1 \times (\mu_0 - p_{i,j}) + p_{i,j} = \mu_0$. Likely, for multiform occlusion, ζ indicates the threshold ϵ_0 that a pixel can change. The $(i \times n + j)^{th}$ extra neuron outputs ϵ_0 , then the corresponding neuron in the fourth layer outputs $p_{i,j} + \epsilon_0$.

This output of $O_{\gamma,x}(a, w, b, h, \zeta)_{i,j}$ is identical to $\gamma_{\zeta,w\times h}(x, a, b)_{i,j}$, the expected pixel value at position (i, j), which also indicates that the color is correctly encoded.

Case 2: When a pixel p at position (i, j) is not occluded, we have $\gamma_{\zeta,w\times h}(x, a, b)_{i,j} = x_{i,j}$. Then, we need to prove that $O_{\gamma,x}(a, w, b, h, \zeta)_{i,j} = x_{i,j}$.

In this case, we can observe that $i < a_0 \vee i \geq a_0 + w_0$ and $j < b_0 \vee j \geq b_0 + h_0$ hold for pixel p. Then We can tell that the corresponding neuron in the third layer outputs 0 and the output of the $(i * n + j)^{th}$ neuron in the fourth layer is the origin pixel value of p following the similar process discussed in case 1.

For the occlusion with real number position, some more cases need to be discussed, but the proof has a very similar sketch as the normal occlusion with integer position. We leverage the equality of $a \times b = exp(log(a) + log(b))$ and add it to the propagation between the third layer and those extra neurons only when the occlusion is at real number positions in the multiform case. And we use $ReLU(a + b - 1)$ as an alternative to logarithms and exponents in implementation since Marabou does not support such operations. Due to the page limit, please refer to [15] for the details of the full proof.

Theorem 1 can be directly derived from Lemma 1 and Definition 3 by substituting $\gamma_{\zeta,w\times h}(x, a, b)$ for $O_{\gamma,x}(a, w, b, h, \zeta)$ in the definition.

4.4 Verification Acceleration Techniques

Existing SMT-based neural network verification tools can directly verify the composed neural network. The number of ReLU activation functions in the network is the primary factor in determining the verification time cost by the backend tools. In the occlusion part, the number of ReLU nodes is independent of the scale of the original networks to be verified. Therefore, our approach's scalability relies only on the underlying tools.

To further improve the verification efficiency, we integrate two algorithmic acceleration techniques by dividing the verification problem into small independent sub-problems that can be solved separately.

Occlusion Space Splitting. We observed that verifying the composed neural network with a large input space can significantly degrade the efficiency of backend verifiers. Even for small FNNs with only tens of ReLUs, the verifiers may run out of time due to the large occlusion space for searching. For instance, the complexity of Reluplex [20] can be derived from the original SMT method of Simplex [32]. It has a complexity of $\mathit{\Omega}(v \times m \times n)$, where m and n represent the number of constraints and variables, and v represents the number of pivots operated in the Simplex method. In the worst case, v can grow exponentially. Reduction in the search space can reduce the number of pivot operations, therefore significantly improving verification efficiency.

Based on the above observation, we can divide $[1, m]$ (*resp.* $[1, n]$) into $k_m \in \mathbb{Z}^+$ (*resp.* $k_n \in \mathbb{Z}^+$) intervals $[m_0, m_1], \ldots, [m_{k_m-1}, m_{k_m}]$ (*resp.* $[n_0, n_1], \ldots, [n_{k_n-1}, n_{k_n}]$) and verify the problem on the Cartesian product of the two sets of intervals.

$$\begin{aligned}&\forall x' \in \mathbb{X}.\mathit{\Phi}(x') = \mathit{\Phi}(x) \equiv \textstyle\bigwedge_{(i,j)=(0,0)}^{(k_m-1,k_n-1)} \forall x' \in \mathbb{X}_{(i,j)}.\mathit{\Phi}(x') = \mathit{\Phi}(x), \text{ where}\\ &\mathbb{X} = \textstyle\bigcup_{(i,j)=(0,0)}^{(k_m-1,k_n-1)} \mathbb{X}_{(i,j)} = \textstyle\bigcup_{(i,j)=(0,0)}^{(k_m-1,k_n-1)} \{\gamma_{\zeta,w\times h}(x, a, b) | m_i \leq a \leq m_{i+1}, n_j \leq b \leq n_{j+1}\}.\end{aligned} \tag{7}$$

In this way, we split the occlusion space into $k_m \times k_n$ sub-spaces. It is equivalent to prove $\forall x' \in \mathbb{X}.\mathit{\Phi}(x')$ for all $\mathbb{X}_{(i,j)}$ with $0 \leq i < k_m$ and $0 \leq j < k_n$, without losing the soundness and completeness. We call each verification instance a *query*, which can be solved more efficiently than the one on the whole occlusion space by backend verifiers. Furthermore, another advantage of occlusion space splitting is that these divided queries can be solved in parallel by leveraging multi-threaded computing.

Eager Falsification by Label Sorting. Another *Divide & Conquer* approach for acceleration is to divide the verification problem into independent sub-problems by the classification labels in L, as defined below:

$$\forall x' \in \mathbb{X}.\mathit{\Phi}(x') = \mathit{\Phi}(x) \equiv \forall x' \in \mathbb{X}. \textstyle\bigwedge_{\ell' \in L} \mathit{\Phi}(x) = \ell' \vee \mathit{\Phi}(x') \neq \ell'. \tag{8}$$

The dual problem to disprove the robustness can be solved to find some label ℓ' such that $\mathit{\Phi}(x) \neq \ell' \wedge \mathit{\Phi}(x') = \ell'$. We can first solve those that have higher probabilities of being non-robust. Once a sub-problem is proved non-robust, the verification terminates, with no need to solve the remainder. Such approach is called *eager falsification* [14]. Based on this methodology, we sort the sub-problems in a descent order according to the probabilities at which the original image is classified to the corresponding labels by the neural network. A higher probability implies that the image is more likely to be classified to the corresponding label. Heuristically, there is a higher probability of finding

Table 1: Occlusion verification results on two medium FNNs trained on MNIST and GTSRB in different occlusion sizes 2 × 2 and 5 × 5 and occlusion radius ϵ.

		Medium FNN (600 ReLUs) on MNIST					Medium FNN (343 ReLUs) on GTSRB				
Size	ϵ	- / +	T_+	T_-	T_{build}	TO(%)	- / +	T_+	T_-	T_{build}	TO(%)
2 × 2	0.05	**2** / 28	120.01	11.98	0.068	0.00	**8** / 13	103.64	24.18	0.089	0.00
	0.10	**3** / 27	121.37	19.18	0.067	0.00	**8** / 13	108.62	22.57	0.088	0.00
	0.20	**4** / 26	122.12	39.57	0.067	0.00	**10** / 11	113.7	23.17	0.084	0.00
	0.30	**6** / 24	165.98	45.6	0.086	0.00	**11** / 10	117.97	26.41	0.089	0.00
	0.40	**7** / 23	183.65	47.32	0.098	4.75	**14** / 7	115.49	31.53	0.096	0.14
5 × 5	0.05	**5** / 25	123.45	49.04	0.065	0.00	**9** / 12	123.99	26.02	0.101	0.00
	0.10	**6** / 24	124.13	44.09	0.073	0.00	**12** / 9	127.65	26.96	0.01	0.00
	0.20	**10** / 20	179.89	52.51	0.073	3.26	**16** / 5	126.98	27.22	0.102	0.00
	0.30	**14** / 16	284.67	65.98	0.076	5.45	**18** / 3	146.68	29.11	0.100	0.04
	0.40	**22** / 8	339.78	97.28	0.074	7.33	**19** / 2	169.17	26.52	0.103	0.09

[*] - / +: the numbers of non-robust and robust cases; T_+ (*resp.* T_-): average verification time in robust (*resp.* non-robust) cases; T_{build}: the building time of occlusion neural networks; TO (%): the percentage of runtime-out cases among all the queries.

an occlusion such that the occluded image is misclassified to that label. We sequence the queries into backend verifiers until all are verified, or a non-robust case is reported. Our experimental results will show that this approach can achieve up to 8 and 24 times speedup in the robust and non-robust cases, respectively.

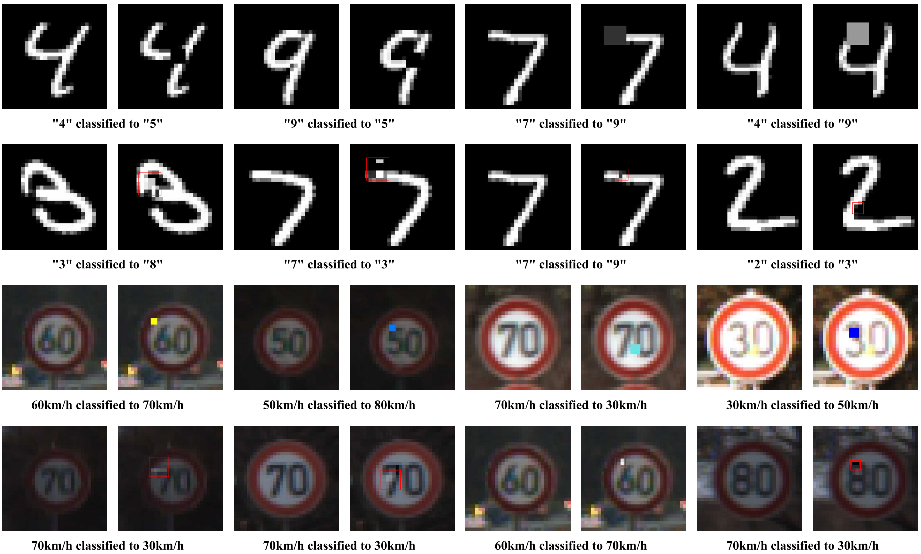
5 Implementation and Evaluation

We implemented our approach in a Python tool called OccRob, using the PyTorch framework. As a backend tool, we chose the Marabou [21] state-of-the-art, SMT-based DNN verifier. We evaluated our proposed approach extensively on a suite of benchmark datasets, including MNIST [24] and GTSRB [16]. The size of the networks trained on the datasets for verification is measured by the number of ReLUs, ranging from 70 to 1300. All the experiments are conducted on a workstation equipped with a 32-core AMD Ryzen Threadripper CPU @ 3.7GHz and 128 GB RAM and Ubuntu 18.04. We set a timeout threshold of 60 seconds for a single verification task. All code and experimental data, including the models and verification scripts can be accessed at https://github.com/MakiseGuo/OccRob.

We evaluate our proposed method concerning efficiency and scalability in the occlusion robustness verification of ReLU-based FNNs. Our goals are threefold:

1. To demonstrate the effectiveness of the proposed approach for the robustness verification against various types of occlusion perturbations.
2. To evaluate the efficiency improvement of the proposed approach, compared with the naive SMT-based method.
3. To demonstrate the effectiveness of the acceleration techniques in efficiency improvement.

Experiment I: Effectiveness. We first evaluate the effectiveness of OccRob in robustness verification against various types of occlusions of different sizes and color ranges. Table 1



"4" classified to "5"

"9" classified to "5"

"7" classified to "9"

"4" classified to "9"

"3" classified to "8"

"7" classified to "3"

"7" classified to "9"

"2" classified to "3"

60km/h classified to 70km/h

50km/h classified to 80km/h

70km/h classified to 30km/h

30km/h classified to 50km/h

70km/h classified to 30km/h

70km/h classified to 30km/h

60km/h classified to 70km/h

70km/h classified to 30km/h

Fig. 7: Occlusive adversarial examples automatically generated for non-robust images.

shows the verification results and time costs against multiform occlusions on two medium FNNs trained on MNIST and GTSRB. We consider two occlusion sizes, 2×2 and 5×5, respectively. The occluding color range is from 0.05 to 0.40. In each verification task, we selected the first 30 images from each of the two datasets and verified the network's robustness around them, under corresponding occlusion settings. As expected, larger occlusion sizes and occluding color ranges imply more non-robust cases. One can see that OccRob can almost always verify and falsify each input image, except for a few time-outs. The robust cases cost more time than the non-robust cases, but all can be finished in a few minutes. Note that the time overhead for building occlusion neural networks is almost negligible, compared with the verification time. The effectiveness against uniform occlusions is shown in the following experiment.

Fig. 7 shows several occlusive adversarial examples that are generated by OccRob under different occlusion settings. These occlusions do not alter the semantics of the original images and should be classified to the same results as those non-occluded ones. However, they are misclassified to other results.

Experiment II: Efficiency improvement over the naive encoding method. We compare the efficiency of OccRob with that of a naive SMT encoding approach on verifying uniform occlusions since the naive encoding approach cannot handle verification against multiform occlusions. We apply the same acceleration techniques, such as parallelization and a variant of input space splitting, to the naive approach, which otherwise times out for almost all verification tasks even on the smallest model.

Table 2 shows the average verification time on six FNNs of different sizes against uniform occlusions. We can observe that OccRob affords a significant improvement in efficiency, up to 30 times higher than the naive approach. It can always finish before the preset time threshold, while the naive method fails to verify the two large networks under the same time threshold. The timeout proportion of two medium networks is over 70%. While the small network on MNIST only has an 8% of timeout proportion with the naive method, OccRob barely timeouts on every network.

Table 2: Performance comparison between OccRob (OR) and the naive (NAI) methods on MNIST and GTSRB under different occlusion sizes.

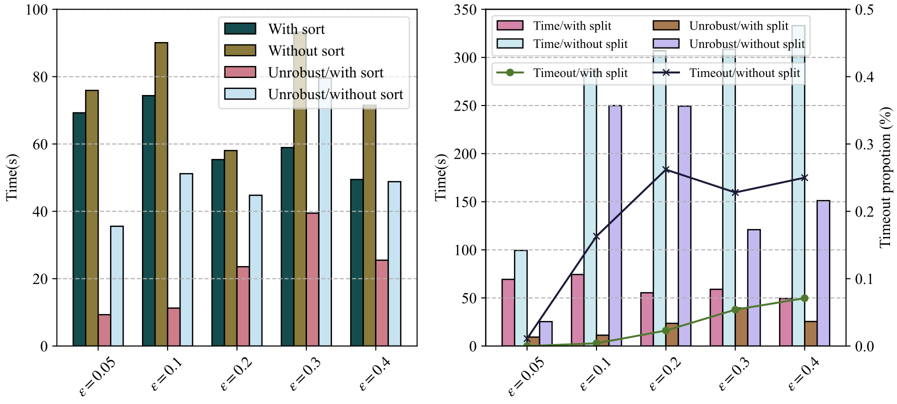
	MNIST						GTSRB					
FNNs	Small FNN		Medium FNN		Large FNN		Small FNN		Medium FNN		Large FNN	
Size	OR	NAI	OR	NAI	OR	NAI	OR	NAI	OR	NAI	OR	NAI
1×1	46.44	63.12	110.18	759.93	206.50	TO	29.76	472.23	69.28	989.08	173.62	TO
2×2	49.62	165.53	98.60	832.98	199.17	TO	21.04	340.89	42.16	680.81	103.42	TO
3×3	51.23	298.59	111.14	863.74	205.67	TO	11.93	169.35	32.00	499.31	81.17	TO
4×4	44.78	256.22	115.99	886.73	225.02	TO	8.90	141.85	31.24	419.62	106.41	TO
5×5	48.96	270.23	113.01	803.40	264.79	TO	6.11	190.81	27.97	418.56	118.99	TO
6×6	47.81	318.28	127.98	642.01	288.18	TO	7.49	213.35	21.70	282.04	60.02	TO
7×7	34.99	357.78	124.47	589.41	222.65	TO	6.02	153.81	31.96	404.18	62.60	TO
8×8	36.05	324.34	129.27	469.24	215.53	TO	5.99	123.07	28.44	250.97	54.37	TO
9×9	34.58	224.01	141.54	375.97	219.61	TO	6.42	102.39	31.30	160.84	59.87	TO
10×10	28.98	178.44	78.89	398.01	182.36	TO	6.61	127.20	28.59	153.96	40.69	TO

Experiment III: Effectiveness of the integrated acceleration techniques. We finally evaluate the effectiveness of the two acceleration techniques integrated with the tool. We evaluate each technique separately by excluding it from OccRob and comparing the verification time of OccRob and the corresponding excluded versions. Fig. 8 shows the experimental results of verifying the medium FNN trained on GTSRB against multiform occlusions by the tools. Fig. 8 (a) shows that label sorting can improve efficiency in both robust and non-robust cases. In particular, the improvement is more significant in the non-robust case, with up to 5 times speedup in the experiment. That is because solving each query is faster than solving all simultaneously, and further OccRob immediately stops dispatching queries once a counterexample is found in the non-robust case. Fig. 8 (b) shows that occlusion space splitting can also significantly improve the efficiency, with up to 8 and 24 times speedups in the robust and non-robust cases, respectively. In addition, Fig. 8 (b) also shows a significant reduction in the number of time-outs.

6 Related Work

Robustness verification of neural networks has been extensively studied recently, aiming at devising efficient methods for verifying neural networks' robustness against various types of perturbations and adversarial attacks. We classify those methods into two categories according to the type of perturbations, which can be semantic or non-semantic. Semantic perturbation has an interpretable meaning, such as occlusions and geometric transformations like rotation, while non-semantic perturbation means that noises perturb inputs with no particular meanings.

Non-semantic perturbations are usually represented as L_p norms, which define the ranges in which an input can be altered. Some robustness verification approaches for



(a) Comparison of label sorting

(b) Comparison of input splitting

Fig. 8: Efficiency evaluation results of the two acceleration techniques.

non-semantic perturbations are both sound and complete by leveraging SMT [20,1] and MILP (mixed integer linear programming) [36] techniques, while some sacrifice the completeness for better scalability by over-approximation [29,2,7], abstract interpretation [34,10,5], interval analysis by symbolic propagation [43,42,26], etc.

In contrast to a large number of works on non-semantic robustness verification, there are only a few studies on the semantic case. Because semantic perturbations are beyond the range of L_p norms [9], those abstraction-based approaches cannot be directly applied to verifying semantic perturbations. Mohapatra et al. [30] proposed to verify neural networks against semantic perturbations by encoding them into neural networks. Their encoding approach is general to a family of semantic perturbations such as brightness and contrast changes and rotations. Their approach for verifying occlusions is restricted to uniform occlusions at integer locations. Sallami et al.[31] proposed an interval-based method to verify the robustness against the occlusion perturbation problem under the same restriction. Singh et al. [35] proposed a new abstract domain to encode both non-semantic and semantic perturbations such as rotations. Chiang et al. [4] called occlusions *adversarial patches* and proposed a certifiable defense by extending interval bound propagation (IBP) [12]. Compared with these existing verification approaches for semantic perturbations, our SMT-based approach is both sound and complete, and meanwhile, it supports a larger class of occlusion perturbations.

7 Conclusion and Future Work

We introduced an SMT-based approach for verifying the robustness of deep neural networks against various types of occlusions. An efficient encoding method was proposed to represent occlusions using neural networks, by which we reduced the occlusion robustness verification problem to a regular robustness verification problem of neural networks and leveraged *off-the-shelf* SMT-based verifiers for the verification. We implemented a resulting prototype OccRob and intensively evaluated its effectiveness and efficiency on a series of neural networks trained on the public benchmarks, including MNIST and GTSRB. Moreover, as the scalability of DNN verification engines continues to improve, our approach, which uses them as blackbox backends, will also become more scalable.

As our occlusion encoding approach is independent of target neural networks, we believe it can be easily extended to other complex network structures, such as convolutional and recurrent ones, which only depend on the backend verifiers. It would also be interesting to investigate how the generated adversarial examples could be used for neural network repairing [41,18] to train more robust networks.

Acknowledgments

This work has been supported by National Key Research Program (2020AAA0107800), NSFC-ISF Joint Program (62161146001, 3420/21) and NSFC projects (61872146, 61872144), Shanghai Science and Technology Commission (20DZ1100300), Shanghai Trusted Industry Internet Software Collaborative Innovation Center and “Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software (Grant No. 22510750100).

References

1. Amir, G., Wu, H., Barrett, C., Katz, G.: An smt-based approach for verifying binarized neural networks. In: TACAS’21. pp. 203–222. Springer (2021)
2. Boopathy, A., Weng, T.W., Chen, P.Y., Liu, S., Daniel, L.: Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In: AAAI’19. vol. 33, pp. 3240–3247 (2019)
3. Brown, T.B., Mané, D., Roy, A., Abadi, M., Gilmer, J.: Adversarial patch. arXiv preprint arXiv:1712.09665 (2017)
4. Chiang, P.y., Ni, R., Abdelkader, A., Zhu, C., Studer, C., Goldstein, T.: Certified defenses for adversarial patches. arXiv preprint arXiv:2003.06693 (2020)
5. Cohen, J., Rosenfeld, E., Kolter, Z.: Certified adversarial robustness via randomized smoothing. In: ICML’19. pp. 1310–1320. PMLR (2019)
6. Coşkun, M., Uçar, A., Yildirim, Ö., Demir, Y.: Face recognition based on convolutional neural network. In: MEES’17. pp. 376–379. IEEE (2017)
7. Elboher, Y.Y., Gottschlich, J., Katz, G.: An abstraction-based framework for neural network verification. In: CAV’20. pp. 43–65. Springer (2020)
8. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: CVPR’18. pp. 1625–1634 (2018)
9. Fischer, M., Baader, M., Vechev, M.: Certified defense to image transformations via randomized smoothing. NeurIPS’20 **33**, 8404–8417 (2020)
10. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: Ai2: Safety and robustness certification of neural networks with abstract interpretation. In: S&P’18. pp. 3–18. IEEE (2018)
11. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, pp. 168–196. MIT Press (2016), http://www.deeplearningbook.org
12. Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., Kohli, P.: On the effectiveness of interval bound propagation for training verifiably robust models. arXiv preprint arXiv:1810.12715 (2018)
13. Gowal, S., Dvijotham, K.D., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., Kohli, P.: Scalable verified training for provably robust image classification. In: ICCV’19. pp. 4842–4851 (2019)

14. Guo, X., Wan, W., Zhang, Z., Zhang, M., Song, F., Wen, X.: Eager falsification for accelerating robustness verification of deep neural networks. In: ISSRE'21. pp. 345–356. IEEE (2021)
15. Guo, X., Zhou, Z., Zhang, Y., Katz, G., Zhang, M.: OccRob: Efficient smt-based occlusion robustness verification of deep neural networks. arXiv preprint (2023)
16. Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., Igel, C.: Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In: IJCNN'13 (2013)
17. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: CAV'17. pp. 3–29. Springer (2017)
18. Islam, M.J., Pan, R., Nguyen, G., Rajan, H.: Repairing deep neural networks: Fix patterns and challenges. In: ICSE'20. pp. 1135–1146. IEEE (2020)
19. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in neural information processing systems. pp. 2017–2025. PMLR (2015)
20. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient smt solver for verifying deep neural networks. In: CAV'17. pp. 97–117. Springer (2017)
21. 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: CAV'19. pp. 443–452. Springer (2019)
22. Kirkland, E.J.: Bilinear Interpolation, pp. 261–263. Springer US, Boston, MA (2010)
23. Kortylewski, A., Liu, Q., Wang, A., Sun, Y., Yuille, A.: Compositional convolutional neural networks: A robust and interpretable model for object recognition under occlusion. International Journal of Computer Vision **129**(3), 736–760 (2021)
24. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), http://yann.lecun.com/exdb/mnist/
25. Lengyel, H., Remeli, V., Szalay, Z.: Easily deployed stickers could disrupt traffic sign recognition. Perner's Contacts **19**(Special Issue 2), 156–163 (2019)
26. Li, J., Liu, J., Yang, P., Chen, L., Huang, X., Zhang, L.: Analyzing deep neural networks with symbolic propagation: Towards higher precision and faster verification. In: SAS'19. pp. 296–319. Springer (2019)
27. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
28. Lyu, Z., Guo, M., Wu, T., Xu, G., Zhang, K., Lin, D.: Towards evaluating and training verifiably robust neural networks. In: CVPR'21. pp. 4308–4317 (2021)
29. Lyu, Z., Ko, C.Y., Kong, Z., Wong, N., Lin, D., Daniel, L.: Fastened crown: Tightened neural network robustness certificates. In: AAAI'20. vol. 34, pp. 5037–5044 (2020)
30. Mohapatra, J., Weng, T.W., Chen, P.Y., Liu, S., Daniel, L.: Towards verifying robustness of neural networks against a family of semantic perturbations. In: ICCV'20. pp. 244–252 (2020)
31. Mziou Sallami, M., Ibn Khedher, M., Trabelsi, A., Kerboua-Benlarbi, S., Bettebghor, D.: Safety and robustness of deep neural networks object recognition under generic attacks. In: ICONIP'19. pp. 274–286. Springer (2019)
32. Nelder, J.A., Mead, R.: A simplex method for function minimization. The computer journal **7**(4), 308–313 (1965)
33. Pei, K., Cao, Y., Yang, J., Jana, S.: Deepxplore: Automated whitebox testing of deep learning systems. In: SOSP'17. pp. 1–18 (2017)
34. Raghunathan, A., Steinhardt, J., Liang, P.: Certified defenses against adversarial examples. arXiv preprint arXiv:1801.09344 (2018)
35. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages **3**(POPL), 1–30 (2019)
36. Singh, G., Gehr, T., Püschel, M., Vechev, M.: Robustness certification with refinement. In: ICLR'19 (2019)
37. Song, L., Gong, D., Li, Z., Liu, C., Liu, W.: Occlusion robust face recognition based on mask learning with pairwise differential siamese network. In: ICCV'19. pp. 773–782 (2019)

38. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation **23**(5), 828–841 (2019)
39. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
40. Tian, Y., Pei, K., Jana, S., Ray, B.: Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: ICSE'18. pp. 303–314 (2018)
41. Usman, M., Gopinath, D., Sun, Y., Noller, Y., Păsăreanu, C.S.: Nn repair: Constraint-based repair of neural network classifiers. In: CAV'21. pp. 3–25. Springer (2021)
42. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Efficient formal safety analysis of neural networks. In: NeurIPS'18. vol. 31. Curran Associates, Inc. (2018)
43. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: USENIX Security'18. pp. 1599–1614 (2018)
44. Zhu, H., Tang, P., Park, J., Park, S., Yuille, A.: Robustness of object recognition under extreme occlusion in humans and computational models. arXiv preprint arXiv:1905.04598 (2019)