Verifying Attention Robustness of Deep Neural Networks Against Semantic Perturbations

Satoshi Munakata, Caterina Urban, Haruki Yokoyama, Koji Yamamoto, Kazuki Munakata

▶ **To cite this version:**

Satoshi Munakata, Caterina Urban, Haruki Yokoyama, Koji Yamamoto, Kazuki Munakata. Verifying Attention Robustness of Deep Neural Networks Against Semantic Perturbations. 15th International Symposium on NASA Formal Methods (NFM 2023), May 2023, Houston (TX), United States. pp.37-61, 10.1007/978-3-031-33170-1_3 . hal-04249934

HAL Id: hal-04249934
https://inria.hal.science/hal-04249934v1

Submitted on 19 Oct 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Verifying Attention Robustness of Deep Neural Networks against Semantic Perturbations

Satoshi Munakata[1], Caterina Urban[2], Haruki Yokoyama[1], Koji Yamamoto[1], and Kazuki Munakata[1]

[1] Fujitsu, Kanagawa, Japan
[2] Inria & ENS | PSL & CNRS, Paris, France

Abstract. It is known that deep neural networks (DNNs) classify an input image by paying particular attention to certain specific pixels; a graphical representation of the magnitude of attention to each pixel is called a *saliency-map*. Saliency-maps are used to check the validity of the classification decision basis, e.g., it is not a valid basis for classification if a DNN pays more attention to the background rather than the subject of an image. Semantic perturbations can significantly change the saliency-map. In this work, we propose the first verification method for *attention robustness*, i.e., the local robustness of the changes in the saliency-map against combinations of semantic perturbations. Specifically, our method determines the range of the perturbation parameters (e.g., the brightness change) that maintains the difference between the actual saliency-map change and the expected saliency-map change below a given threshold value. Our method is based on activation region traversals, focusing on the outermost robust boundary for scalability on larger DNNs. We empirically evaluate the effectiveness and performance of our method on DNNs trained on popular image classification datasets.

1 Introduction

Classification Robustness. Deep neural networks (DNN) are now dominant solutions in computer vision, notably for image classification [20]. However, quality assurance is essential when DNNs are used in safety-critical systems [2]. From an assurance point of view, one key property that has been extensively studied is the robustness of the classification against input perturbations [16]. In particular, a long line of work has focused on robustness to adversarial input perturbations [35]. However, DNNs have been shown to also be vulnerable to input perturbations likely to naturally occur in practice, such as small brightness changes, translations, rotations, and other spacial transformations [8, 19, 40, 7, 11]. In this paper we focus on such *semantic perturbations*. A number of approaches have been proposed to determine the range of perturbation parameters (e.g., the amount of brightness change and translation) that do not change the classification [3, 23]. However, we argue that classification robustness is not a sufficient quality assurance criterion in safety-critical scenarios.

Classification Validity. It is known that DNNs classify an input image by paying particular attention to certain specific pixels in the image; a graphical

representation of the magnitude of attention to each pixel, like a heatmap, is called *saliency-map* [28, 34]. A saliency-map can be obtained from the gradients of DNN outputs with respect to an input image, and it is used to check the validity of the classification decision basis. For instance, if a DNN classifies the subject type by paying attention to a background rather than the subject to be classified in an input image (as in the case of "Husky vs. Wolf [26]"), it is not a valid basis for classification. We believe that such low validity classification should not be accepted in safety-critical situations, even if the classification labels are correct. Semantic perturbations can significantly change the saliency-maps [24, 12, 13]. However, existing robustness verification methods only target changes in the classification labels and not the saliency-maps.

Our Approach: Verifying Attention Robustness. In this work, we propose the first verification method for *attention robustness*[3], i.e., the local robustness of the changes in the saliency-map against combinations of semantic perturbations. Specifically, our method determines the range of the perturbation parameters (e.g., the brightness change) that maintains the difference between (a) the actual saliency-map change and (b) the expected saliency-map change below a given threshold value. Regarding the latter (b), brightness change keeps the saliency-map unchanged, whereas translation moves one along with the image. Although the concept of such difference is the same as *saliency-map consistency* used in semi-supervised learning [12, 13], for the sake of verification, it is necessary to calculate the minimum and maximum values of the difference within each perturbation parameter sub-space. Therefore, we specialize in the gradient-based saliency-maps for the image classification DNNs [28] and focus on the fact that DNN output is linear with respect to DNN input within an activation region [14]. That is, the actual saliency-map calculated from the gradient only is constant within each region; thus, we can compute the range of the difference by sampling a single point within each region if the saliency-map is expected to keep, while by convex optimization if the saliency map is expected to move. Our method is based on traversing activation regions on a DNN with layers for classification and semantic perturbations; it is also possible to traverse (i.e., verify) all activation regions in a small DNN or traverse only activation regions near the outermost robust boundary in a larger DNN. Experimental results demonstrate that our method can show the extent to which DNNs can classify with the same basis regardless of semantic perturbations and report on performance and performance factors of activation region traversals.

Contributions. Our main contributions are:

- We formulate the problem of attention robustness verification; we then propose a method for verifying attention robustness for the first time. Using our method, it is also possible to traverse and verify all activation regions or only ones near the outermost decision boundary.
- We implement our method in a python tool and evaluate it on DNNs trained with popular datasets; we then show the specific performance and factors of

[3] In this paper, the term "attention" refers to the focus of certain specific pixels in the image, and not to the "attention mechanism" used in transformer models[39].

verifying attention robustness. In the context of traversal verification methods, we use the largest DNNs for performance evaluation.

2 Overview

Situation. Suppose a situation where we have to evaluate the weaknesses of a DNN for image classification against combinations of semantic perturbations caused by differences in shooting conditions, such as lighting and subject position. For example, as shown in Figure 1, the original label of the handwritten text image is *"0"*; however, the DNN often misclassifies it as the other labels, with changes in brightness, patch, and translations. Therefore, we want to know in advance the ranges of semantic perturbation parameters that are likely to cause such misclassification as a weakness of the DNN for each typical image. However, classification robustness is not sufficient for capturing such weaknesses in the following cases.

Case 1. Even if the brightness changes so much that the image is not visible to humans, the classification label of the perturbed image may happen to match the original label. Then vast ranges of the perturbation parameters are evaluated as robust for classification; however, such overestimated ranges are naturally invalid and unsafe. For instance, Figure 2 shows the changes in MNIST image "8" and the actual saliency-map when the brightness is gradually changed; although the classification seems robust because the labels of each image are the same, the collapsed saliency-maps indicate that the DNN does not pay proper attention to text "8" in each image. Therefore, our approach uses the metric *attention inconsistency*, which quantifies the degree of collapse of a saliency-map, to further evaluate the range of the perturbation parameter as satisfying the property *attention robustness*; i.e., the DNN is paying proper attention as well as the original image. Attention inconsistency is a kind of distance (cf. Figure 4) between an actual saliency-map (second row) and an expected one (third row); e.g., the saliency-map of *DNN-1* for translation perturbation (column (T)) is expected to follow image translation; however, if it is not, then attention inconsistency is high. In addition, Figure 2 shows an example of determining that attention robustness is satisfied if each attention inconsistency value (third row) is less than or equal to threshold value δ.

Case 2. The classification label often changes by combining semantic perturbations, such as brightness change and patch, even for the perturbation parameter ranges that each perturbation alone could be robust. It is important to understand what combinations are weak for the DNN; however, it is difficult to verify all combinations as there are many semantic perturbations assumed in an operational environment. In our observations, a perturbation that significantly collapses the saliency-map is more likely to cause misclassification when combined with another perturbation because another perturbation can change the intensity of pixels to which the DNN should not pay attention. Therefore, to understand the weakness of combining perturbations, our approach visualizes the *outermost boundary* at which the sufficiency of robustness switches on the

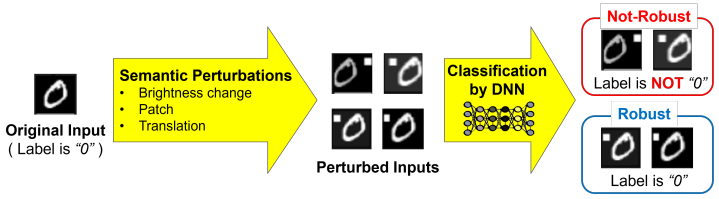


Fig. 1: Misclassifications caused by combinations of semantic perturbations.

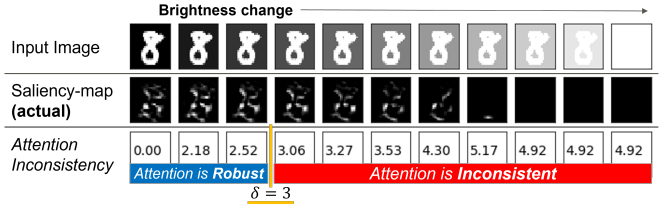


Fig. 2: Perturbation-induced changes in images (first row), saliency-maps (second row) and the metric quantified the degree of collapse of each saliency-map (third row); where δ denotes the threshold to judge a saliency-map is valid or not.

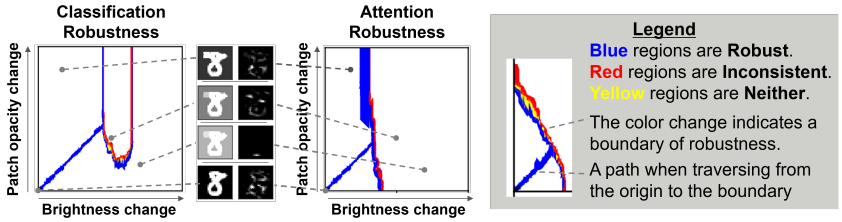


Fig. 3: The outermost boundaries of classification robustness (left) and attention robustness (right); the origin at the bottom-left corresponds to the input image without perturbation, and each plotted point denotes the perturbed input image (middle). The shapes of the boundaries indicate the existence of regions that the DNN successfully classifies without sufficient evidence.

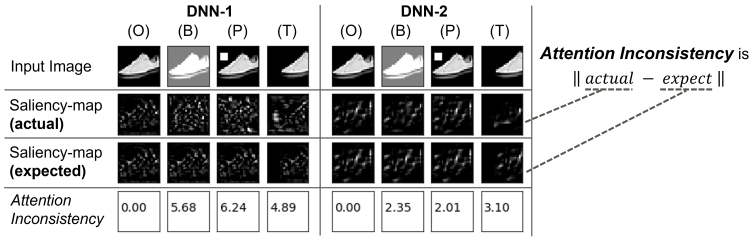


Fig. 4: Differences in changes in saliency-maps for two DNNs. Each saliency-map of DNN-1 above is more collapsed than DNN-2's: where columns (O), (B), (P), and (T) denote original (i.e., without perturbations), brightness change, patch, and translation, respectively.

perturbation parameter space. For instance, Figure 3 shows connected regions that contain the outermost boundary for classification robustness (left side) or attention robustness (right side). The classification boundary indicates that the DNN can misclassify the image with a thin patch and middle brightness. In contrast, the attention boundary further indicates that the brightness change can collapse the saliency-map more than patching, so we can see that any combinations with the brightness change pose a greater risk. Even when the same perturbations are given, the values of attention inconsistency for different DNNs are usually different (cf. Figure 4); thus, it is better to evaluate what semantic perturbation poses a greater risk for each DNN.

3 Problem Formulation

Our method targets feed-forward ReLU-activated neural networks (ReLU-FNNs) for image classification. A ReLU-FNN *image classifier* is a function $f\colon X \to Y$ mapping an N^f-dimensional (pixels $\times$ color-depth) image $x \in X \subseteq \mathbb{R}^{N^f}$ to a classification label $argmax_{j \in Y} f_j(x)$ in the K^f-class label space $Y = \{1, \ldots, K^f\}$, where $f_j : X \to \mathbb{R}$ is the confidence function for the j-th class. ReLU-FNNs include fully-connected neural networks and convolutional neural networks (CNNs).

The ReLU activation function occurs in between the linear maps performed by the ReLU-FNN layers and applies the function $max(0, x_{l,n})$ to each neuron $x_{l,n}$ in a layer $l \in L^f$ (where L^f is the number of layers of ReLU-FNN f). When $x_{l,n} > 0$, we say that $x_{l,n}$ is *active*; otherwise, we say that $x_{l,n}$ is *inactive*. We write $ap^f(x)$ for the *activation pattern* of an image x given as input to a ReLU-FNN f, i.e., the sequence of neuron activation statuses in f when x is taken as input. We write AP^f for the entire set of activation patterns of a ReLU-FNN f.

Given an activation pattern $p \in AP^f$, we write $ar^f(p)$ for the corresponding *activation region*, i.e., the subset of the input space containing all images that share the same activation pattern: $x \in ar^f(p) \Leftrightarrow ap^f(x) = p$. Note that, neuron activation statuses in an activation pattern p yield half-space constraints in the input space [14, 18]. Thus, an activation region $ar^f(p)$ can equivalently be represented as a convex polytope described by the conjunction of the half-space constraints resulting from the activation pattern p.

Classification Robustness. A *semantic perturbation* is a function $g : \Theta \times X \to X$ applying a perturbation with N^g parameters $\theta \in \Theta \subseteq \mathbb{R}^{N^g}$ to an image $x \in X$ to yield a perturbed image $g(\theta, x) \stackrel{\text{def}}{=} g_{N^g}(\theta_{N^g}, \cdot) \circ \cdots \circ g_1(\theta_1, x) = g_{N^g}(\theta_{N^g}, \ldots g_1(\theta_1, x), \ldots) \in X$, where $g_i : \mathbb{R} \times X \to X$ performs the i-th atomic semantic perturbation with parameter θ_i (with $g_i(0, x) = x$ for any image $x \in X$). For instance, a brightness decrease perturbation g_b is a(n atomic) semantic perturbation function with a single brightness adjustment parameter $\beta \geq 0$: $g_b(\beta, x) \stackrel{\text{def}}{=} ReLU(x - \vec{1}\beta)$.

Definition 1 (Classification Robustness). *A perturbation region $\eta \subset \Theta$ satisfies* classification robustness *— written $CR(x; \eta)$ — if and only if the classifi-*

cation label $f(g(\theta, x))$ is the same as $f(x)$ when the perturbation parameter θ is within η: $CR(x;\eta) \stackrel{\text{def}}{=} \forall\theta \in \eta.\ f(x) = f(g(\theta, x))$.

Vice versa, we define *misclassification robustness* when $f(g(\theta, x))$ is always different from $f(x)$ when θ is within η: $MR(x;\eta) \stackrel{\text{def}}{=} \forall\theta \in \eta.\ f(x) \neq f(g(\theta, x))$.

The *classification robustness verification problem* $Prob^{CR} \stackrel{\text{def}}{=} (f, g, x0, \Theta)$ consists in enumerating, for a given input image $x0$, the perturbation parameter regions $\eta^{CR}, \eta^{MR} \subset \Theta$ respectively satisfying $CR(x0;\eta^{CR})$ and $MR(x0;\eta^{MR})$.

Attention Robustness. We generalize the definition of saliency-map from [28] to that of an *attention-map*, which is a function $map_j : X \to X$ from an image $x \in X$ to the heatmap image $m_j \in X$ plotting the magnitude of the contribution to the j-th class confidence $f_j(x)$ for each pixel of x. Specifically, $map_j(x) \stackrel{\text{def}}{=} filter\Big(\frac{\partial f_j(x)}{\partial x_1}, \ldots, \frac{\partial f_j(x)}{\partial x_{Nf}}\Big)$, where $filter(\cdot)$ is an arbitrary image processing function (such as normalization and smoothing) and, following [28, 36], the magnitude of the contribution of each pixel $x_1, \ldots, x_{Nf}$ is given by the gradient with respect to the j-th class confidence. When $filter(x) \stackrel{\text{def}}{=} |x|$, our definition of map_j matches that of saliency-map in [28]. Note that, within an activation region $ar^f(p)$, f_j is linear [14] and thus the gradient $\frac{\partial f_j(x)}{\partial x_i}$ is a constant value.

We expect attention-maps to change consistently with respect to a semantic image perturbation. For instance, for a brightness change perturbation, we expect the attention-map to remain the same. Instead, for a translation perturbation, we expect the attention-map to be subject to the same translation. In the following, we write $\tilde{g}(\cdot)$ for the attention-map perturbation corresponding to a given semantic perturbation $g(\cdot)$. We define *attention inconsistency* as the difference between the actual and expected attention-map after a semantic perturbation: $ai(x;\theta) \stackrel{\text{def}}{=} \sum_{j\in Y} dist\Big(map_j\big(g(\theta, x)\big), \tilde{g}\big(\theta, map_j(x)\big)\Big)$, where $dist\colon X \times X \to \mathbb{R}$ is an arbitrary distance function such as Lp-norm ($||x - x'||_p$). Note that, when $dist(\cdot)$ is L2-norm, our definition of attention inconsistency coincides with the definition of saliency-map consistency given by [12].

Definition 2 (Attention Robustness). *A perturbation region $\eta \subset \Theta$ satisfies* attention robustness *— written $AR(x;\eta,\delta)$ — if and only if the attention inconsistency is always less than or equal to δ when the perturbation parameter θ is within η: $AR(x;\eta,\delta) \stackrel{\text{def}}{=} \forall\theta \in \eta.\ ai(x;\theta) \leq \delta$.*

When the attention inconsistency is always greater than δ, we have *inconsistency robustness*: $IR(x;\eta,\delta) \stackrel{\text{def}}{=} \forall\theta \in \eta.\ ai(x;\theta) > \delta$.

The *attention robustness verification problem* $Prob^{AR} \stackrel{\text{def}}{=} (f, g, x0, \Theta, \delta)$ consists in enumerating, for a given input image $x0$, the perturbation parameter regions $\eta^{AR}, \eta^{IR} \subset \Theta$ respectively satisfying $AR(x0;\eta^{AR},\delta)$ and $IR(x0;\eta^{IR},\delta)$.

Outermost Boundary Verification. In practice, to represent the trend of the weakness of a ReLU-FNN image classifier to a semantic perturbation, we argue

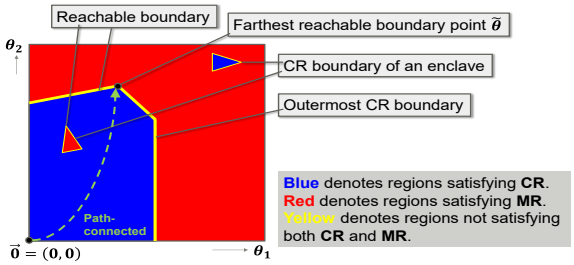


Fig. 5: Illustration of outermost CR boundary on a 2-dimensional perturbation parameter space. The origin $\vec{0}$ is the original image without perturbation.

that it is not necessary to enumerate all perturbation parameter regions within a perturbation parameter space Θ. Instead, we search the *outermost CR/AR boundary*, that is, the perturbation parameter regions η that lay on the CR/AR boundary farthest away from the original image.

An illustration of the outermost CR boundary is given in Figure 5. More formally, we define the outermost CR boundary as follows:

Definition 3 (Outermost CR Boundary). *The outermost CR boundary of a classification robustness verification problem, $ob^{CR}(Prob^{CR})$, is a set of perturbation parameter regions $HS \subset \mathcal{P}(\Theta)$ such that:*

1. *for all perturbation regions $\eta \in HS$, there exists a path connected-space from the original image $x0$ (i.e., $\vec{0} \in \Theta$) to η that consists of regions satisfying CR (written $Reachable(\eta; x0)$);*
2. *all perturbation regions $\eta \in HS$ lay on the classification boundary, i.e., $\exists \theta, \theta' \in \eta.\ f(g(\theta, x0)) = f(x0) \wedge f(g(\theta', x0)) \neq f(x0)$;*
3. *there exists a region $\eta \in HS$ that contains the farthest reachable perturbation parameter point $\tilde{\theta}$ from the original image, i.e., $\tilde{\theta} = max_{\theta \in \Theta} ||\theta||_2$ such that $Reachable(\{\theta\}; x0)$.*

The definition of the outermost AR boundary is analogous. Note that not all perturbation regions inside the outermost CR/AR boundary satisfy the CR/AR property (cf. the enclaves in Figure 5).

The *outermost CR boundary verification problem* and *outermost AR boundary verification problem* $Prob^{CR}_{ob} = (f, g, x0, \Theta)$ and $Prob^{AR}_{ob} = (f, g, x0, \Theta, \delta)$ consist in enumerating, for a given input image $x0$, the perturbation parameter regions η^{CR}_{ob} and η^{AR}_{ob}) that belong to the outermost CR and AR boundary $ob^{CR}(Prob^{CR})$ and $ob^{AR}(Prob^{AR})$.

4 Geometric Boundary Search (*GBS*)

In the following, we describe our Geometric Boundary Search (GBS) method for solving $Prob^{CR}_{ob}$, and $Prob^{AR}_{ob}$ shown in Algorithm 1 and 2. In Appendix H.8, we describe a baseline Breadth-First Search (BFS) method for solving $Prob^{CR}$, and $Prob^{AR}$ (enumerating all perturbation parameter regions).

4.1 Encoding Semantic Perturbations

After some variables initialization (cf. Line 1 in Algorithm 1), the semantic perturbation g is encoded into a ReLU-FNN $g^{x0} : \Theta \to X$ (cf. Line 2).

In this paper, we focus on combinations of atomic perturbations such as brightness change (B), patch placement (P), and translation (T). Nonetheless, our method is applicable to any semantic perturbation as long as it can be represented or approximated with sufficient accuracy.

For the encoding, we follow [23] and represent (combinations of) semantic perturbations as a piecewise linear function by using affine transformations and ReLUs. For instance, a brightness decrease perturbation $g_b(\beta, x0) \stackrel{\text{def}}{=} ReLU(x0 - \vec{1}\beta)$ (cf. Section 3) can be encoded as a ReLU-FNN as follows:

$$g_b(\beta, x0) \stackrel{\text{encode}}{\longrightarrow} \begin{bmatrix} 1\ 0 \dots\ 0 \\ 0\ 1 \dots\ 0 \\ \dots \\ 0\ 0 \dots\ 1 \end{bmatrix} ReLU \left(\begin{bmatrix} -1\ 1\ 0 \dots\ 0 \\ -1\ 0\ 1 \dots\ 0 \\ \dots \\ -1\ 0\ 0 \dots\ 1 \end{bmatrix} \begin{bmatrix} \beta \\ x0_1 \\ \dots \\ x0_{N^f} \end{bmatrix} \right) + \vec{0}$$

which we can combine with the given ReLU-FNN f to obtain the compound ReLU-FNN $f \circ g_b^{x0}$ to verify. The full encoding for all considered (brightness, patch, translation) perturbations is shown in Appendix H.5.

4.2 Traversing Activation Regions.

GBS then performs a traversal of activation regions of the compound ReLU-FNN $f \circ g^{x0}$ near the outermost CR/AR boundary for $Prob_{ob}^{CR}/Prob_{ob}^{AR}$. Specifically, it initializes a queue Q with the activation pattern $ap^{f \circ g^{x0}}(\vec{0})$ of the original input image $x0$ with no semantic perturbation, i.e., $\theta = \vec{0}$ (cf. Line 3 in Algorithm 1, we explain the other queue initialization parameters shortly). Given a queue element $q \in Q$, the functions $p(q)$, $isFollowing(q)$, and $lineDistance(q)$ respectively return the 1st, 2nd, and 3rd element of q.

Then, for each activation pattern p in Q (cf. Line 6), *GBS* reconstructs the corresponding perturbation parameter region η (subroutine *constructActivationRegion*, Line 7) as the convex polytope resulting from p (cf. Section 3 and η in Figure 6-(1a)).

Next, for each neuron $x_{l,n}$ in $f \circ g^{x0}$ (cf. Line 11), it checks whether its activation status cannot flip within the perturbation parameter space Θ, i.e., the resulting half-space would have no feasible points within Θ (subroutine *isStable*, Line 12, cf. half-space $h_{1,5}$ in Figure 6-(1a)). Otherwise, a new activation pattern p' is constructed by flipping the activation status of $x_{l,n}$ (subroutine *flipped*, Line 13) and added to a local queue Q' (cf. Line 9, and 23, 25) if p' has not been observed already (cf. Line 14) and it is feasible (subroutine *calcInteriorPointOnFace*, Lines 15-16, cf. point θ^F and half-space $h_{1,2}$ in Figure 6-(1a)).

The perturbation parameter region η is then simplified to $\tilde{\eta}$ (subroutine *simplified*, Line 2 in Algorithm 2; e.g., reducing the half-spaces used to represent η to just $h_{1,2}$ and $h_{1,3}$ in Figure 6-(1a)). $\tilde{\eta}$ is used to efficiently calculate the

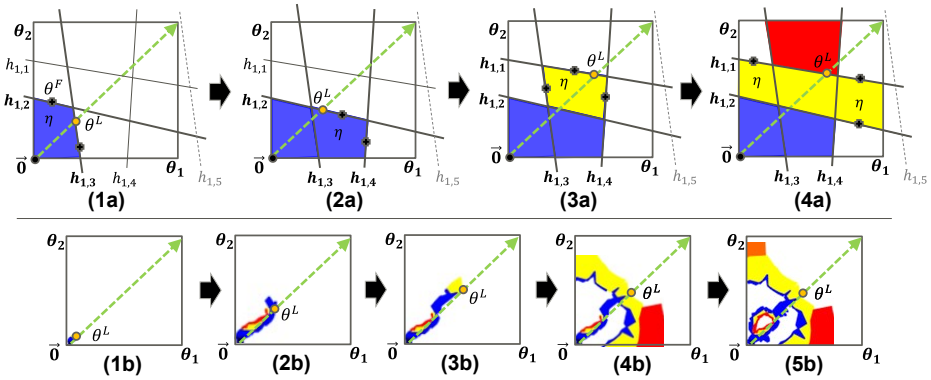


Fig. 6: A running example of *GBS*. The upper row shows the basic traversing flow, while the lower row shows the flow of avoiding enclaves. $h_{l,n}$ denotes a half-space corresponding to neuron activity $p_{l,n}$.

range of attention inconsistency within η (subroutine *calcRange*, Line 3 in Algorithm 2, cf. Section 4.4), and then attention/inconsistency robustness can be verified based on the range (Line 5 and 8 in Algorithm 2). Furthermore, classification/misclassification robustness can be verified in the same way if subroutine *calcRange* returns the range of confidence $f_{f(x0)}(g^{x0}(\theta)) - f_j(g^{x0}(\theta))$ within $\tilde{\eta}$ (cf. Section 4.4) and $\delta = 0 \wedge w^\delta = 0$. At last, the local queue Q' is pushed onto Q (cf. Line 29 in Algorithm 1).

To avoid getting stuck around enclaves inside the outermost CR/AR boundary (cf. Figure 5) during the traversal of activation regions, *GBS* switches status when needed between "searching for a decision boundary" and "following a found decision boundary". The initial status is set to "searching for a decision boundary", i.e., $\neg isFollowing$ when initializing the queue Q (cf. Line 3). The switch to $isFollowing$ happens when region η is on the boundary (i.e., $lo \leq \delta \leq up$) or near the boundary (i.e., $\delta - w^\delta \leq lo \leq \delta + w^\delta$, cf. Line 15 in Algorithm 2 and Figure 6-(3a,1b,3b)), where w^δ is a hyperparameter to determine whether the region is close to the boundary or not. The hyperparameter w^δ should be greater than 0 to verify attention/inconsistency robustness because attention inconsistency changes discretely for ReLU-FNNs (cf. Section 4.3). *GBS* can revert back to searching for a decision boundary if, when following a found boundary, it finds a reachable perturbation parameter region that is farther from $\vec{0}$ (cf. Lines 19-20 in Algorithm 1 and Figure 6-(2b)).

4.3 Calculating Attention Inconsistency

Gradients within an Activation Region. Let $p \in AP^{f \circ g^{x0}}$ be an activation pattern of the compound ReLU-FNN $f \circ g^{x0}$. The gradient $\frac{\partial f_j(g^{x0}(\theta))}{\partial \theta_s}$ is constant within $ar^{f \circ g^{x0}}(p)$ (cf. Section 3). We write $g_i^{x0}(\theta)$ for the i-th pixel x_i of a perturbed image in $\{g^{x0}(\theta) \mid \theta \in ar^{f \circ g^{x0}}(p)\} \subset X$. The gradient $\frac{\partial g_i^{x0}}{\partial \theta} = \frac{\partial x_i}{\partial \theta}$ is

Algorithm 1 $gbs(f, g, x0, \Theta; \delta, w) \to (H^{CR}, H^{MR}, H^{CB}, H^{AR}, H^{IR}, H^{AB})$

Input: $f, g, x0, \Theta, \delta$
Output: $H^{CR}, H^{MR}, H^{CB}, H^{AR}, H^{IR}, H^{AB} \subset \mathcal{P}(\Theta)$

$H^{CR}, H^{MR}, H^{CB}, H^{AR}, H^{IR}, H^{AB} \leftarrow \{\}, \{\}, \{\}, \{\}, \{\}, \{\}$
$g^{x0} \leftarrow g(\cdot, x0)$ *// partially applying $x0$ to g; i.e, $g^{x0}(\theta) = g(\theta, x0)$.*
$Q \subset AP^{f \circ g^{x0}} \times \mathbb{B} \times \mathbb{R} \leftarrow \{(ap^{f \circ g^{x0}}(\vec{0}), \bot, 0)\}$ *// queue for boundary search.*
$OBS \subset AP^{f \circ g^{x0}} \leftarrow \{\}$ *// observed activation patterns.*
while $\#|Q| > 0$ *// loop for geometrical-boundary search.* **do**
  $q \leftarrow popMaxLineDistance(Q)$; $p \leftarrow p(q)$; $OBS \leftarrow OBS \cup \{p\}$
  $\eta \leftarrow constructActivationRegion(f \circ g^{x0}, p)$
  $FS \subset \mathbb{Z} \times \mathbb{Z} \leftarrow \{\}$ *// (l, n) means the n-th neuron in l-th layer is a face of η*
  $Q' \subset AP^{f \circ g^{x0}} \times \mathbb{B} \times \mathbb{R} \leftarrow \{\}$ *// local queue for an iteration.*
  // Push each activation region connected to η.
  for $l = 1$ **to** #layers of $f \circ g^{x0}$; $n = 1$ **to** #neurons of the l-th layer **do**
    continue **if** $isStable(p, l, n, \Theta)$ *// skip if activation of $x_{l,n}$ cannot flip in Θ.*
    $p' \leftarrow flipped(p, l, n)$ *// flip activation status for neuron $x_{l,n}$.*
    continue **if** $p' \in OBS$ **else** $OBS \leftarrow OBS \cup \{p'\}$ *// skip if p' was observed.*
    $\theta^F \leftarrow calcInteriorPointOnFace(\eta, l, n)$
    continue **if** $\theta^F = null$ *// skip if p' is infeasible.*
    $FS \leftarrow FS \cup \{(l, n)\}$ *// (l,n) is a face of η.*
    $\theta^L \leftarrow calcInteriorPointOnLine(\eta, l, n)$
    if $isFollowing(q) \wedge \theta^L \neq null \wedge ||\theta^L||_2 > lineDistance(q)$ **then**
      $q \leftarrow (p, \bot, lineDistance(q))$ *// Re-found the line in boundary-following.*
    end if
    if $\neg isFollowing(q) \wedge \theta^L \neq null$ **then**
      $Q' \leftarrow Q' \cup \{(p', \bot, ||\theta^L||_2)\}$ *// continue line-search.*
    else
      $Q' \leftarrow Q' \cup \{(p', isFollowing(q), lineDistance(q))\}$ *// continue current.*
    end if
  end for
  (...Verify η...) *// See Algorithm 2 for AR/IR (analogous for CR/MR)*
  $Q \leftarrow Q \cup Q'$ *// Push*
end while

also a constant value. By the chain rule, we have $\frac{\partial f_j(x)}{\partial x_i} = \frac{\partial f_j(g^{x0}(\theta))/\partial \theta_s}{\partial x_i / \partial \theta_s}$. Thus $\frac{\partial f_j(x)}{\partial x_i}$ is also constant. This fact is formalized by the following lemma:

Lemma 1. $\frac{\partial f_j(x)}{\partial x_i} = C \;\; (x \in \{g^{x0}(\theta) \mid \theta \in ar^{f \circ g^{x0}}(p)\})$

(cf. the small example in Appendix H.7). Therefore, the gradient $\frac{\partial f_j(x)}{\partial x_i}$ can be computed as the weights of the j-th class output for ReLU-FNN f about activation pattern $ap^f(\dot{x})$; where, $\dot{x} = g^{x0}(\dot{\theta})$ and $\dot{\theta}$ is an arbitrary sample within $ar^{f \circ g^{x0}}(p)$ (cf. Appendix H.1). For the perturbed gradient $\tilde{g}(\theta, \frac{\partial f_j(x)}{\partial x_i})$, let $\tilde{g}(\theta)$ be the ReLU-FNN $g^{\frac{\partial f_j(x)}{\partial x_i}}(\theta')$. Thus, the same consideration as above applies.

Attention Inconsistency (AI). We assume both $filter(\cdot)$ and $dist(\cdot)$ are convex downward functions for calculating the maximum/minimum value by convex

Algorithm 2 (Expanding from Algorithm 1 for $AR(x0;\eta,\delta)/IR(x0;\eta,\delta)$)

(...Verify** η**...)
$\tilde{\eta} \leftarrow simplified(\eta, FS)$ *// limit the constraints on* η *to* FS.
$(lo, up) \leftarrow calcRange(x0;\tilde{\eta})$ *// the range ([lower and upper) of ai within* $\tilde{\eta}$.
$nearBoundary \leftarrow (lo \leq \delta \leq up) \vee (\delta - w^{\delta} \leq lo \leq \delta + w^{\delta}) \vee (\delta - w^{\delta} \leq up \leq \delta + w^{\delta})$
if $lo \leq up \leq \delta$ */* satisfying AR */* **then**
  $H^{AR} \leftarrow H^{AR} \cup \{\tilde{\eta}\}$
  $Q' \leftarrow \{\}$ **if** $isFollowing(q) \wedge \neg nearBoundary$ *// no traversing connected regions.*
else if $\delta < lo \leq up$ */* satisfying IR */* **then**
  $H^{IR} \leftarrow H^{IR} \cup \{\tilde{\eta}\}$
  $Q' \leftarrow \{\}$ **if** $\neg nearBoundary$ *// no traversing connected regions.*
else
  $H^{AB} \leftarrow H^{AB} \cup \{\tilde{\eta}\}$
end if
if $\neg isFollowing(q) \wedge nearBoundary$ **then**
  (...Update** Q' ***such that $\forall q' \in Q.\ isFollowing(q')$**...)** *// switch to boundary-following.*
end if

optimization. Specifically, $filter(\cdot)$ is one of the identity function (I), the absolute function (A), and the 3×3 mean filter (M). $dist(\cdot)$ is one of the L_1-norm (L_1) and the L_2-norm (L_2): where, w is the width of image $x \in X$.

4.4 Verifying CR/MR and AR/IR

Our method leverages the fact that the gradient of a ReLU-FNN output with respect to the input is constant within an activation region (cf. Section 3); thus, CR/MR can be resolved by linear programming, and AR/IR can be resolved by just only one sampling if the saliency-map is expected to keep or convex optimization if the saliency-map is expected to move.

Verifying CR/MR. When $x0$ is fixed, each activation region of the ReLU-FNN $f(g(\theta, x0)) : \Theta \rightarrow Y$ is a region in the perturbation parameter space Θ. Within an activation region $\eta \subset \Theta$ of the ReLU-FNN $f(g(\theta, x0))$, $CR(f, g, x0, \eta)$ is satisfied if and only if the ReLU-FNN output corresponding to the label of the original image $x0$ cannot be less than the ReLU-FNN outputs of all other labels, i.e., $min_{j \in Y \setminus \{f(x0)\}, \theta \in \eta} f_{f(x0)}(x) - f_j(g(\theta, x0)) > 0 \Leftrightarrow CR(f, g, x0, \eta)$ Each DNN output $f_j(g(\theta, x0))$ is linear within η, and thus, the left-hand side of the above equation can be determined *soundly* and *completely* by using an LP solver (Eq. 3-(c) in Appendix H.1). Similarly, $MR(f, g, x0, \eta)$ is satisfied if and only if the ReLU-FNN output corresponding to the label of the original image $x0$ cannot be greater than the ReLU-FNN outputs of any other labels.

Verifying AR/IR. Within an activation region $\eta \subset \Theta$ of the ReLU-FNN $f(g(\theta, x0))$, $AR(f, g, x0, \eta, \delta)$ is satisfied if and only if the following equation holds: $max_{\theta \in \eta} ai(\theta, x0) \leq \delta \Leftrightarrow AR(f, g, x0, \eta, \delta)$ If $filter(\cdot)$ and $dist(\cdot)$ are both convex downward functions (CDFs), as the sum of CDFs is also a CDF, the left-hand side of the above equation can be determined by comparing the values

Table 1: ReLU-FNNs used in our experiments. Networks with name prefix "M-" ("F-") is trained on MNIST (Fashion-MNIST). In column Layers, "FC' denotes fully connected layers while "Conv" denotes convolutional layers.

Name	#Neurons	Layers
M-FNN-100	100	FC×2
M-FNN-200	200	FC×4
M-FNN-400	400	FC×8
M-FNN-800	800	FC×16
M-CNN-S	2,028	Conv×2,FC×1
M-CNN-M	14,824	Conv×2,FC×1

Name	#Neurons	Layers
F-FNN-100	100	FC×2
F-FNN-200	200	FC×4
F-FNN-400	400	FC×8
F-FNN-800	800	FC×16
F-CNN-S	2,028	Conv×2,FC×1
F-CNN-M	14,824	Conv×2,FC×1

at both ends. On the other hand, $IR(f, g, x0, \eta, \delta)$ is satisfied if and only if the following equation holds: $min_{\theta \in \eta} ai(\theta, x0) > \delta \Leftrightarrow IR(f, g, x0, \eta, \delta)$ The left-hand side of the above equation can be determined by using a convex optimizer. Note that if the saliency-map is expected to keep against perturbations, the above optimization is unnecessary because $ai(\theta \in eta, x0)$ is constant.

Thus, it is straightforward to conclude that our GBS method is sound and complete for verifying CR/MR and AR/IR over the explored activation regions:

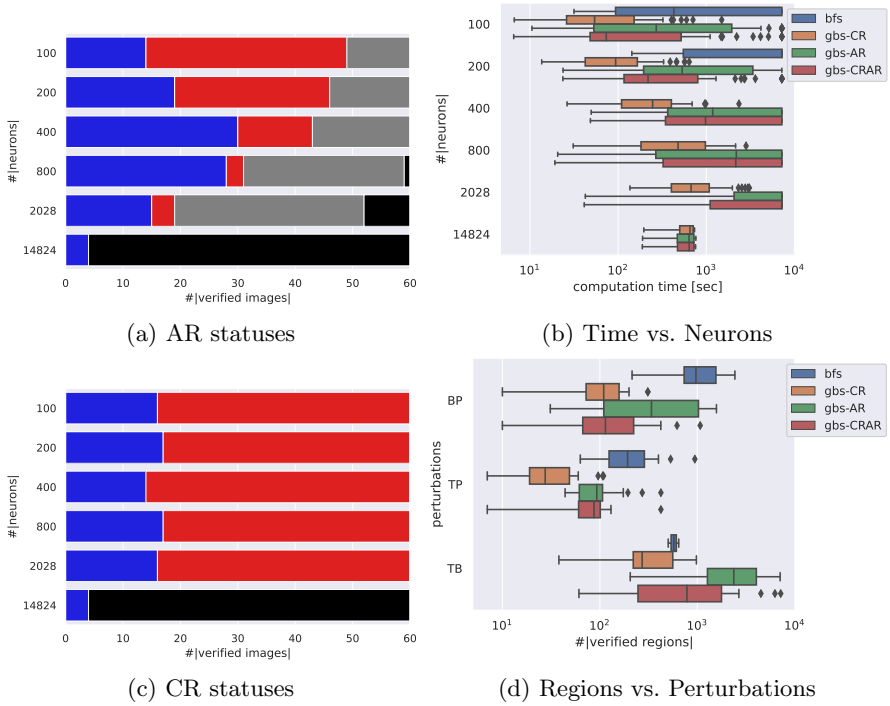
Theorem 1. *Our GBS method shown in Algorithm 1 and 2 is sound and conditionally complete for solving $Prob_{ob}^{CR}$, and $Prob_{ob}^{AR}$.*

- *If the outermost CR/AR boundary truly exists in Θ and the boundary bisects Θ into two parts, one with the origin $\vec{0}$ and the other with its diagonal point (cf. Figure 5), then our GBS method always explores the boundary.*
- *Otherwise, our GBS method does not always find CR/AR boundaries (including enclaves) that should be considered the outermost CR/AR boundary.*

5 Experimental Evaluation

Our *GBS* method is implemented as an open-source Python tool. It is available at `https://zenodo.org/record/6544905`. We evaluated *GBS* on ReLU-FNNs trained on the MNIST [6] and Fashion-MNIST [41] datasets. Table 1 shows the different sizes and architectures used in our evaluation. During each experiment, we inserted semantic perturbation layers (cf. Section 4.1) with a total of 1,568 neurons in front of each ReLU-FNN. All experiments were performed on virtual computational resource "rt_C.small" (with CPU 4 Threads and Memory 30 GiB) of physical compute node "V" (with *2 CPU; Intel Xeon Gold 6148 Processor 2.4 GHz 20 Cores (40 Threads)*, and *12 Memory; 32 GiB DDR4 2666 MHz RDIMM (ECC)*) in the *AI Bridging Cloud Infrastructure (ABCI)* [1].

In our evaluation, we considered three variants of GBS: *gbs-CR*, which searches the CR boundary, *gbs-AR*, which searches the AR boundary, and *gbs-CRAR*, which searches the boundary of the regions satisfying both CR and AR. For *gbs-AR* and *gbs-CRAR* we used the definitions $filter(x) \stackrel{\text{def}}{=} x$, $dist(x, x') \stackrel{\text{def}}{=} ||x - x'||_2$, $\delta \stackrel{\text{def}}{=} 3.0$, and $w^{\delta} = 0.2$ (for *gbs-CR* we used $\delta = 0$ and $w^{\delta} = 0$, cf. Section 4.2).



(a) AR statuses

(b) Time vs. Neurons

(c) CR statuses

(d) Regions vs. Perturbations

Fig. 7: Experimental Results.

Figure 7a and 7c show the breakdown of the verification result for *gbs-AR* and *gbs-CR* over 10 images (not used for training) from each dataset considering three combinations of semantic perturbations (brightness+patch (BP), translation+patch (TP), translation+brightness (TB)). The blue, red, gray, and black bars denote robust (i.e., all explored activation regions were verified to be robust), not-robust (i.e., all activation regions on the outermost CR/AR boundary were explored and at least one of them was found to violate robustness), timed out, and out-of-memory, respectively. We used a time out of two hours for each robustness verification. The figures show that *gbs-AR* timed out at a higher rate for smaller size DNNs than *gbs-CR*. This is due to the fact that *gbs-AR* verified robustness for more images and generally explored more activation regions than *gbs-CR* as also shown in Figure 7d, which compares the number of verified (explored) activation regions for each combination of perturbations. Looking further into this difference, it turns out that the choice of the hyperparameter $w^\theta = 0.2$ also caused *gbs-AR* to explore the AR boundary more extensively than necessary. It remains for future work to study how to choose a good value for w^θ to narrow this search. Overall, however, Figure 7d shows that *GBS* (which searches the outermost CR/AR boundary) reduces as intended the number of explored activation regions with respect to the baseline *BFS* (which enumerates

all perturbation parameter regions). This, in turn, directly affects the performance of the algorithms as demonstrated in Figure 7b, which shows the trend of increasing computation time with increasing the number of neurons for each algorithm as a box plot on a log scale. The figure also shows that the median computation time increases exponentially with the number of neurons for all algorithms. This result is expected [18, 9, 42] and suggests that incorporating abstractions and approximate verification methods [30, 43, 25] is needed in order to scale to very large DNNs such as *VGG16* [29].

6 Related Work

Robustness Verification. To the best of our knowledge, we are the first to formulate and propose an approach for the attention robustness verification problem (see, e.g., recent surveys in the area [16, 38, 2]). [30] first verified classification robustness against image rotation, and [3] verified classification robustness against other semantic perturbations such as image translation, scaling, shearing, brightness change, and contrast change. However, in this paper, we have argued that attention robustness more accurately captures trends in weakness for the combinations of semantic perturbations than existing classification robustness in some cases (cf. Section 2). In addition, approximate verification methods such as *DeepPoly* [30] fail to verify near the boundary [32] while our GBS method enables verification near the boundary by exploratory and exact verification.

[23] proposed that any Lp-norm-based verification tools can be used to verify the classification robustness against semantic perturbations by inserting special DNN layers that induce semantic perturbations in the front of DNN layers for classification. In order to transform the verification problem on the inherently high-dimensional input image space X into one on the low-dimensional perturbation parameter space Θ, we adopted their idea, i.e., inserting DNN layers for semantic perturbations ($\Theta \to X$) in front of DNN layers for classification ($X \to Y$). However, it is our original idea to calculate the value range of the gradient for DNN output $(\partial f_j(g(\theta, x_i))/\partial x_i)$ within an activation region on the perturbation parameter space (cf. Sections 4.3-4.4).

Traversing activation regions. Since [18] first proposed the method to traverse activation regions, several improvements and extensions have been proposed [21, 9]. All of them use all breadth-first searches with a priority queue to compute the maximum safety radius or the maxima of the given objective function in fewer iterations. In contrast, our algorithm GBS uses a breadth-first search with a priority queue to reach the outermost CR/AR boundary in fewer iterations while avoiding enclaves.

[9] responded to the paper reviewer that traversing time would increase exponentially with the size of a DNN [10]. Our experiment also showed that larger DNNs increase traversing time due to the denser activation regions. The rapid increase in the number of activation regions will be one of the biggest barriers to the scalability of traversing methods, including our method. Although the upper bound theoretical estimation for the number of activation regions increases

exponentially with the number of layers in a DNN [15], [14] reported that actual DNNs have surprisingly few activation regions because of the myriad of infeasible activation patterns. Therefore, it will need to understand the number of activation regions of DNNs operating in the real world. To improve scalability, there are several methods of targeting only low-dimensional subspaces in the high-dimensional input space for verification [31, 33, 22]. We have similarly taken advantage of low-dimensionality, e.g., using low-dimensional perturbation parameters to represent high-dimensional input image pixels as mediator variables (i.e., partially applied perturbation function $g^{x^0}(\theta) = x'$) to reduce the elapsed time of LP solvers, determining the stability of neuron activity from few vertices of perturbation parameter space Θ. Another possibility to improve scalability is the method of partitioning the input space and verifying each partition in a perfectly parallel fashion [37]. Our implementation has not been fully parallelized yet but it should be relatively straightforward to do as part of our future work.

Saliency-Map. Since [28] first proposed the method to obtain a saliency-map from the gradients of DNN outputs with respect to an input image, many improvements and extensions have been proposed [5, 34, 27]. We formulated an attention-map primarily using the saliency-map definition by [28]. However, it remains for future work to formulate attention robustness corresponding to improvements, such as gradient-smoothing [5] and line-integrals [34].

It is known that semantic perturbations can significantly change the saliency-maps [24, 12, 13]. [12] first claimed the saliency-map should consistently follow image translation and proposed the method to quantify saliency-map consistency. We formulated attention inconsistency ac primarily using the saliency-map consistency by [12]. While there have been works on the empirical studies of the attribution robustness[4, 17], the verification of it has not been studied.

7 Conclusion and Future Work

We have presented a verification method for attention robustness based on traversing activation regions on the DNN that contains layers for semantic perturbations and layers for classification. Attention robustness is the property that the saliency-map consistency is less than a threshold value. We have provided a few cases that attention robustness more accurately captures trends in weakness for the combinations of semantic perturbations than existing classification robustness. Although the performance evaluation presented in this study is not yet on a practical scale, such as VGG16 [29], we believe that the attention robustness verification problem we have formulated opens a new door to quality assurance for DNNs. We plan to increase the number of semantic perturbation types that can be verified and improve scalability by using abstract interpretation in future work.

References

1. AIRC, A.: Abci system overview (2022), https://docs.abci.ai/en/system-overview/
2. Ashmore, R., Calinescu, R., Paterson, C.: Assuring the machine learning lifecycle: Desiderata, methods, and challenges. ACM Computing Surveys (CSUR) **54**(5), 1–39 (2021)
3. Balunovic, M., Baader, M., Singh, G., Gehr, T., Vechev, M.: Certifying geometric robustness of neural networks. In: NeurIPS. vol. 32 (2019)
4. Chen, J., Wu, X., Rastogi, V., Liang, Y., Jha, S.: Robust attribution regularization. Advances in Neural Information Processing Systems **32** (2019)
5. Daniel, S., Nikhil, T., Been, K., Fernanda, V., Wattenberg, M.: Smoothgrad: removing noise by adding noise. In: ICMLVIZ. PMLR (2017)
6. Deng, L.: The MNIST Database of Handwritten Digit Images for Machine Learning Research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)
7. Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., Madry, A.: Exploring the landscape of spatial robustness. In: ICML. pp. 1802–1811. PMLR (2019)
8. Fawzi, A., Frossard, P.: Manitest: Are Classifiers Really Invariant? In: BMVC. pp. 106.1–106.13 (2015)
9. Fromherz, A., Leino, K., Fredrikson, M., Parno, B., Pasareanu, C.: Fast geometric projections for local robustness certification. In: ICLR. ICLR (2021)
10. Fromherz, A., Leino, K., Fredrikson, M., Parno, B., Pasareanu, C.: Fast geometric projections for local robustness certification — openreview (2021), https://openreview.net/forum?id=zWy1uxjDdZJ
11. Gao, X., Saha, R.K., Prasad, M.R., Roychoudhury, A.: Fuzz testing based data augmentation to improve robustness of deep neural networks. In: ICSE. pp. 1147–1158. IEEE,ACM (2020)
12. Guo, H., Zheng, K., Fan, X., Yu, H., Wang, S.: Visual attention consistency under image transforms for multi-label image classification. In: CVPR. pp. 729–739. IEEE,CVF (2019)
13. Han, T., Tu, W.W., Li, Y.F.: Explanation consistency training: Facilitating consistency-based semi-supervised learning with interpretability. In: AAAI. vol. 35, pp. 7639–7646. AAAI (2021)
14. Hanin, B., Rolnick, D.: Deep relu networks have surprisingly few activation patterns. In: NeurIPS. vol. 32 (2019)
15. Hinz, P.: An analysis of the piece-wise affine structure of ReLU feed-forward neural networks. Ph.D. thesis, ETH Zurich (2021)
16. Huang, X., Kroening, D., Ruan, W., Sharp, J., Sun, Y., Thamo, E., Wu, M., Yi, X.: A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Computer Science Review **37**, 100270 (2020)
17. Jha, S.K., Ewetz, R., Velasquez, A., Ramanathan, A., Jha, S.: Shaping noise for robust attributions in neural stochastic differential equations. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 9567–9574 (2022)
18. Jordan, M., Lewis, J., Dimakis, A.G.: Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. In: NeurIPS. vol. 32 (2019)
19. Kanbak, C., Moosavi-Dezfooli, S., Frossard, P.: Geometric Robustness of Deep Networks: Analysis and Improvement. In: CVPR. pp. 4441–4449 (2018)
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS. vol. 25 (2012)

21. Lim, C.H., Urtasun, R., Yumer, E.: Hierarchical verification for adversarial robustness. In: ICML. vol. 119, pp. 6072–6082. PMLR (2020)
22. Mirman, M., Hägele, A., Bielik, P., Gehr, T., Vechev, M.: Robustness certification with generative models. In: PLDI. pp. 1141–1154. ACM SIGPLAN (2021)
23. 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: CVPR. pp. 244–252. IEEE,CVF (2020)
24. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digital Signal Processing **73**, 1–15 (2018)
25. Müller, M.N., Makarchuk, G., Singh, G., Püschel, M., Vechev, M.T.: PRIMA: General and Precise Neural Network Certification via Scalable Convex Hull Approximations. Proceedings of the ACM on Programming Languages **6**(POPL), 1–33 (2022)
26. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?" explaining the predictions of any classifier. In: KDD. pp. 1135–1144. ACM SIGKDD (2016)
27. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: ICCV. IEEE (Oct 2017)
28. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: ICLR (2014)
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
30. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. In: POPL. pp. 1–30. ACM New York, NY, USA (2019)
31. Sotoudeh, M., Thakur, A.V.: Computing linear restrictions of neural networks. In: NeurIPS. vol. 32 (2019)
32. Sotoudeh, M., Thakur, A.V.: Provable repair of deep neural networks. In: PLDI. pp. 588–603. ACM SIGPLAN (2021)
33. Sotoudeh, M., Thakur, A.V.: Syrenn: A tool for analyzing deep neural networks. In: TACAS. pp. 281–302 (2021)
34. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: ICML. pp. 3319–3328. PMLR (2017)
35. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing Properties of Neural Networks. In: ICLR (2014)
36. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. In: ICLR (2019)
37. Urban, C., Christakis, M., Wüstholz, V., Zhang, F.: Perfectly parallel fairness certification of neural networks. Proceedings of the ACM on Programming Languages **4**(OOPSLA), 1–30 (2020)
38. Urban, C., Miné, A.: A review of formal methods applied to machine learning. CoRR **abs/2104.02466** (2021), `https://arxiv.org/abs/2104.02466`
39. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
40. Xiao, C., Zhu, J., Li, B., He, W., Liu, M., Song, D.: Spatially Transformed Adversarial Examples. In: ICLR (2018)
41. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms (2017), `http://arxiv.org/abs/1708.07747`

42. Xu, S., Vaughan, J., Chen, J., Zhang, A., Sudjianto, A.: Traversing the local polytopes of relu neural networks: A unified approach for network verification. In: AdvML. AAAI (2022)
43. Yang, P., Li, J., Liu, J., Huang, C., Li, R., Chen, L., Huang, X., Zhang, L.: Enhancing Robustness Verification for Deep Neural Networks via Symbolic Propagation. Formal Aspects of Computing **33**(3), 407–435 (2021)

H Appendix

H.1 Linearity of Activation Regions

Given activation pattern $p \in AP^f$ as constant, within activation region $ar^f(p)$ each output of ReLU-FNN $f_j(x \in ar^f(p))$ is linear for x (cf. Figure 8) because all ReLU operators have already resolved to 0 or x [14]. i.e., $f_j(x \in ar^f(p)) = A'_j x + b'_j$: where, A'_j and b'_j denote simplified weights and bias about activation pattern p and class j. That is, the gradient of each ReLU-FNN output $f_j(x)$

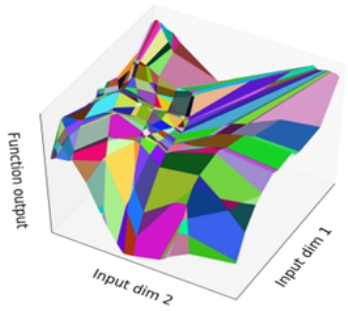


Fig. 8: An example of activation regions [14].
ReLU-FNN output is linear on each activation region, i.e., each output plane painted for each activation region is flat.

within activation region $ar^f(p)$ is constant, i.e., the following equation holds: where $C \in \mathbb{R}$ is a constant value.

$$Feasible^f(p \in AP^f) \Rightarrow \frac{\partial f_j(x)}{\partial x_i} = C \ \ (x \in ar^f(p)) \tag{1}$$

An activation region can be interpreted as the H-representation of a convex polytope on input space $\mathbb{R}^{N^f}$. Specifically, neuron activity $p_{l,n}$ and p have a one-to-one correspondence with a half-space and convex polytope defined by the intersection (conjunction) of all half-spaces, because $f_n^{(l)}(x)$ is also linear when $p \in AP^f$ is constant. Therefore, we interpret activation region $ar^f(p)$ and the following H-representation of convex polytope $HConvex^f(x; p)$ each other as needed: where, A'' and b'' denote simplified weights and bias about activation

pattern p, and $A''_{l,n}x \leq b''_{l,n}$ is the half-space corresponding to the n-th neuron activity $p_{l,n}$ in the l-th layer.

$$HConvex^f(x;p) \stackrel{\text{def}}{=} A''x \leq b'' \equiv \bigwedge_{l,n} A''_{l,n}x \leq b''_{l,n} \tag{2}$$

H.2 Connectivity of Activation Regions

When feasible activation patterns $p, p' \in AP^f$ are in a relationship with each other that flips single neuron activity $p_{l,n} \in \{0,1\}$, they are connected regions because they share single face $HFace^f_{l,n}(x;p) \stackrel{\text{def}}{=} A''_{l,n}x = b''_{l,n}$ corresponding to flipped $p_{l,n}$ [18]. It is possible to flexibly traverse activation regions while ensuring connectivity by selecting a neuron activity to be flipped according to a prioritization; several traversing methods have been proposed [18, 21, 9]. However, there are generally rather many neuron activities that become infeasible when flipped [18]. For instance, half-spaces $h_{1,3}$ is a face of activation region η in Figure 6-(1a); thus, flipping neuron activity $p_{1,3}$, *GBS* can traverse connected region η in Figure 6-(1b). In contrast, half-space $h_{1,1}$ is not a face of activation region η in Figure 6-(1a); thus, flipping neuron activity $p_{1,1}$, the corresponded activation region is infeasible (i.e., the intersection of flipped half-spaces has no area).

H.3 Hierarchy of Activation Regions

When feasible activation patterns $p, p' \in AP^f$ are in a relationship with each other that matches all of L'^f-th upstream activation pattern $p_{\leq L'^f} \stackrel{\text{def}}{=} [p_{l,n} \mid 1 \leq l \leq L'^f, 1 \leq n \leq N^f_l]$ $(1 \leq L'^f \leq L^f)$, they are included parent activation region $ar^f_{\leq L'^f}(p)$ corresponding to convex polytope $HConvex^f_{\leq L'^f}(x;p) \stackrel{\text{def}}{=} \bigwedge_{l \leq L'^f, n} A''_{l,n}x \leq b''_{l,n}$ [21]. That is, $\forall x \in ar^f(p).\ x \in ar^f_{\leq L'^f}(p)$ and $\forall x \in \mathbb{R}^{N^f}.\ HConvex^f(x;p) \Rightarrow HConvex^f_{\leq L'^f}(x;p)$.

Similarly, we define L'^f-th downstream activation pattern as $p_{\geq L'^f} \stackrel{\text{def}}{=} [p_{l,n} \mid L'^f \leq l \leq L^f, 1 \leq n \leq N^f_l]$ $(1 \leq L'^f \leq L^f)$.

H.4 Linear Programming on an Activation Region

Based on the linearity of activation regions and ReLU-FNN outputs, we can use *Linear Programming (LP)* to compute **(a)** the feasibility of an activation region, **(b)** the flippability of a neuron activity, and **(c)** the minimum (maximum) of a ReLU-FNN output within an activation region. We show each LP encoding of the problems (a,b,c) in the *SciPy* LP form [4]: where, $p \in AP^f$ is a given activation

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html

pattern of ReLU-FNN f, and $p_{l,n}$ is a give neuron activity to be flipped.

$$
\begin{aligned}
&(\mathbf{a})\ \ \exists x \in \mathbb{R}^{N^f}.\ HConvex^f(x;p) \overset{\text{encode}}{\longrightarrow} \min_x \vec{0}x\ \ \mathbf{s.t.},\ A''x \leq b'' \\
&(\mathbf{b})\ \ \exists x \in \mathbb{R}^{N^f}.\ HConvex^f(x;p) \wedge HFace^f_{l,n}(x;p) \\
&\qquad\qquad \overset{\text{encode}}{\longrightarrow} \min_x \vec{0}x\ \ \mathbf{s.t.},\ A''x \leq b'',\ A''_{l,n}x = b''_{l,n} \\
&(\mathbf{c})\ \ \min_x f_j(x)\ \ \mathbf{s.t.},\ HConvex^f(x;p) \overset{\text{encode}}{\longrightarrow} \left(\min_x A'_j x\ \ \mathbf{s.t.},\ A''x \leq b''\right) + b'_j
\end{aligned}
\tag{3}
$$

H.5 Full Encoding Semantic Perturbations

We focus here on the perturbations of brightness change (B), patch (P), and translation (T), and then describe how to encode the combination of them into ReLU-FNN $g^{x0} : \Theta \to X$: where, $|\theta^{(l)}| = \dim \theta^{(l)}$, w is the width of image $x0$, px, py, pw, ph are the patch x-position, y-position, width, height, and tx is the amount of movement in x-axis direction. Here, perturbation parameter $\theta \in \Theta$ consists of the amount of brightness change for (B), the density of the patch for (P), and the amount of translation for (T). In contrast, perturbation parameters not included in the dimensions of Θ, such as w, px, py, pw, ph, tx, are assumed to be given as constants before verification.

$$
\begin{aligned}
g(\theta, x0) &\overset{\text{encode}}{\longrightarrow} g^{x0}(\theta)\ \ \textit{// partially applying given constant } x0 \textit{ to } g. \\
g^{x0}(\theta) &= g^{(5)}(\theta \circ x0)\ \ \textit{// concat } x0 \\
g^{(1)}(\mu) &= A^{(T)}\mu\ \ \textit{// translate} \\
g^{(2)}(\mu) &= A^{(P)}g^{(1)}(\mu)\ \ \textit{// patch} \\
g^{(3)}(\mu) &= A^{(B)}g^{(2)}(\mu)\ \ \textit{// brightness change} \\
g^{(4)}(\mu) &= -ReLU(g^{(3)}(\mu)) + \vec{1}\ \ \textit{// clip } max(0, x_i) \\
g^{(5)}(\mu) &= -ReLU(g^{(4)}(\mu)) + \vec{1}\ \ \textit{// clip } min(1, x_i)
\end{aligned}
$$

$$
A^{(B)} = \left[a^{(B)}_{r,c}\right], A^{(P)} = \left[a^{(P)}_{r,c}\right], A^{(T)} = \left[a^{(T)}_{r,c}\right]
$$

$$
a^{(B)}_{r,c} = \begin{cases} 1\ (c = 1 \wedge r \geq |\theta^{(l+1)}|) & \textit{// add } \theta^{(l)}_1 \\ 1\ (c = r + 1) & \textit{// copy } \theta^{(l)}_{\geq 2} \textit{ and } x_i \\ 0\ (\textit{otherwise}) & \end{cases}
$$

$$
a^{(P)}_{r,c} = \begin{cases} 1\ (c = 1 \wedge On(r)) & \textit{// add } \theta^{(l)}_1 \\ 1\ (c = r + 1) & \textit{// copy } \theta^{(l)}_{\geq 2} \textit{ and } x_i \\ 0\ (\textit{otherwise}) & \end{cases}
$$

$$
On(r) \overset{\text{def}}{\equiv} \textbf{let } i \coloneqq r - |\theta^{(l+1)}|.\ (px \leq \lfloor i/w \rfloor \leq px + pw) \wedge (py \leq i \bmod w \leq py + ph)
$$

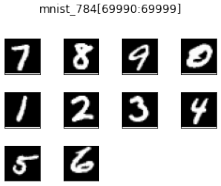


Fig. 9: MNIST images used for experiments.

Fig. 10: Fashion-MNIST images used for experiments.

$$a^{(T)}_{r,c} = \begin{cases} 1 & (c = r+1 \land r \leq |\theta^{(l+1)}|) \quad // \textit{ copy } \theta^{(l)}_{\geq 2} \\ 0 & (c = 1 \land \neg(1 \leq t(r) \leq s(r) \leq N)) \quad // \textit{ zero padding} \\ x0_{tgt(r)} - x0_{src(r)} & (c = 1 \land r \geq |\theta^{(l+1)}|) \quad // \textit{ add } \theta^{(l)}_1 \Delta x0_i \\ 1 & (c = s(r) + |\theta^{(l)}| \land r > |\theta^{(l+1)}|) \quad // \textit{ copy } x0_i \\ 0 & (\textit{otherwise}) \end{cases}$$

$$s(r) \stackrel{\text{def}}{=} \textbf{let } i \coloneqq r - |\theta^{(l+1)}|.\ (\lfloor i/w \rfloor + tx - 1)w + (i \bmod w)$$

$$t(r) \stackrel{\text{def}}{=} \textbf{let } i \coloneqq r - |\theta^{(l+1)}|.\ (\lfloor i/w \rfloor + tx - 2)w + (i \bmod w)$$

H.6 Images used for our experiments

We used 10 images (i.e., Indexes 69990-69999) selected from the end of the MNIST dataset (cf. Figure 9) and the Fashion-MNIST dataset (cf. Figure 10), respectively. We did not use these images in the training of any ReLU-FNNs.

H.7 An example of Lemma 1

Lemma 1 is reprinted below.

$$\frac{\partial f_j(x)}{\partial x_i} = C \quad (x \in \{g^{x0}(\theta) \mid \theta \in ar^{f \circ g}(p)\})$$

A small example of Lemma 1 (cf. Figure 11)

Let $X = [0,1]^3$, $Y = \mathbb{R}^2$, $\Theta = [0,1]^1$, $x0 \in X = (1, 0.5, 0.1)$, $g^{x0}(\theta \in \Theta) \in X = ReLU(-\theta\vec{1} + x0)$, and $f(x \in X) \in Y = ReLU(x_1 + x_2, x_1 + x_3)$.
Because $g^{x0}(0.6) = ReLU(0.4, -0.1, -0.5)$ and $f(g^{x0}(0.6)) = ReLU(0.4, 0.4)$, $p = ap^{f\circ g}(0.6) = [1, 0, 0|1, 1] \in AP^{f\circ g}$.
Then, $p_{\geq 2} = [1, 1] = ap^f(g^{x0}(0.6)) \in AP^f$.
Here, $ar^{f\circ g}(p)$ corresponding to $HConvex^{f\circ g}(\theta; p) \equiv -\theta+1 \geq 0 \wedge -\theta+0.5 \leq 0 \wedge -\theta+0.1 \leq 0 \wedge -\theta+1 \geq 0 \wedge -\theta+1 \geq 0 \equiv 0.5 \leq \theta \leq 1$, on the other hand, $ar^f(p_{\geq 2})$ corresponding to $HConvex^f(x; p_{\geq 2}) \equiv x_1 + x_2 \geq 0 \wedge x_1 + x_3 \geq 0$.
Because $0 \leq x_1 + x_2 = x_1 + x_3 = 1 - \theta \leq 0.5$ $(\theta \in ar^{f\circ g}(p))$, $\forall\theta \in ar^{f\circ g}(p)$. $g^{x0}(\theta) \in ar^f(p_{\geq 2})$.

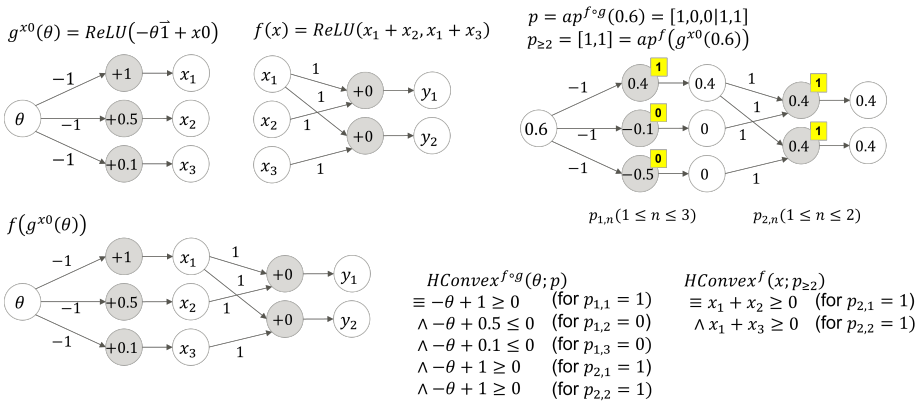


Fig. 11: An image for a small example of Lemma 1.

H.8 Algorithm BFS

Algorithm BFS traverses entire activation regions in perturbation parameter space Θ, as shown in Figure 12.

Algorithm BFS initializes Q with $ap^{f\circ g^{x0}}(\vec{0})$ (Line 3). Then, for each activation pattern p in Q (Lines 5-6), it reconstructs the corresponding activation region η (subroutine constructActivationRegion, Line 8) as the H-representation of p (cf. Equation 2). Next, for each neuron in $f \circ g^{x0}$ (Line 12), it checks whether the neuron activity $p_{l,n}$ cannot flip within the perturbation parameter space Θ, i.e., one of the half-spaces has no feasible points within Θ (subroutine isStable, Line 13). Otherwise, a new activation pattern p' is constructed by flipping $p_{l,n}$ (subroutine flipped, Line 14) and added to the queue (Line 20) if p' is feasible (subroutine calcInteriorPointOnFace, Lines 17-18). Finally, the activation

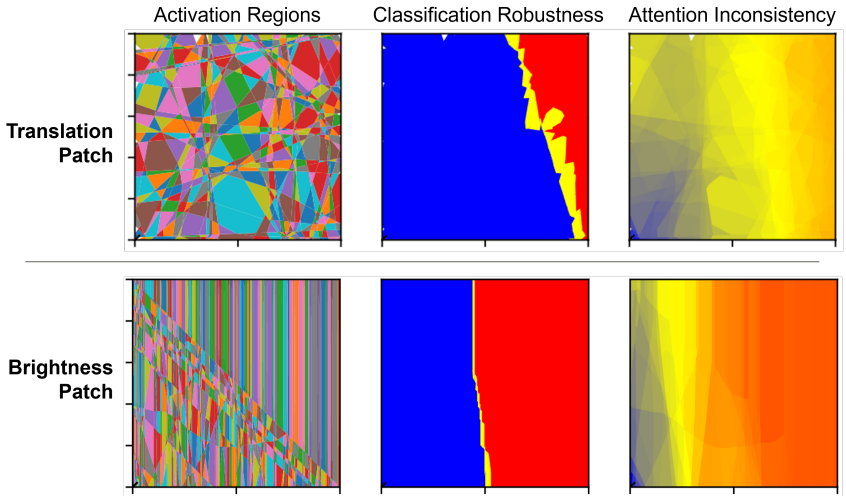


Fig. 12: Examples of *BFS* results. (Near the edges, polygons may fail to render, resulting in blank regions.)

region η is simplified (Line 24) and used to verify CR (subroutine solveCR and solveVR, Lines 25-27, cf. Section 4.4) and VR (subroutine solveAR and solveIR, Lines 32-34, cf. Section 4.4).

H.9 Details of experimental results

Table 2 shows the breakdown of verification statuses in experimental results for each algorithm and each DNN size (cf. Section 5). In particular, for traversing AR boundaries, we can see the problem that the ratio of "Timeout" and "Failed (out-of-memory)" increases as the size of the DNN increases. This problem is because gbs-AR traverses more activation regions by the width of the hyperparameter w^{δ} than gbs-CR. It would be desirable in the future, for example, to traverse only the small number of activation regions near the AR boundary.

Table 2: Breakdown of verification statuses. “Robust” and “NotRobust” mean algorithm found “only robust regions” and “at least one not-robust region”, respectively. “Timeout” and “Failed” mean algorithm did not finish “within 2 hours” and “due to out-of-memory”, respectively.

algorithm	#neurons	Robust	NotRobust	Timeout	Failed
bfs	100	13	22	25	0
bfs	200	11	15	34	0
gbs-CR	100	16	44	0	0
gbs-CR	200	17	43	0	0
gbs-CR	400	14	46	0	0
gbs-CR	800	17	43	0	0
gbs-CR	2028	16	44	0	0
gbs-CR	14824	4	0	0	56
gbs-AR	100	14	35	11	0
gbs-AR	200	19	27	14	0
gbs-AR	400	30	13	17	0
gbs-AR	800	28	3	28	1
gbs-AR	2028	15	4	33	8
gbs-AR	14824	4	0	0	56
gbs-CRAR	100	14	41	5	0
gbs-CRAR	200	19	33	8	0
gbs-CRAR	400	30	14	16	0
gbs-CRAR	800	28	6	26	0
gbs-CRAR	2028	15	8	32	5
gbs-CRAR	14824	4	0	0	56

Algorithm 3 $bfs(f, g, x0, \Theta, \delta) \to (H^{CR}, H^{MR}, H^{CB}, H^{AR}, H^{IR}, H^{AB})$

Input: $f, g, x0, \Theta, \delta$
Output: $H^{CR}, H^{MR}, H^{CB}, H^{AR}, H^{IR}, H^{AB} \subset \mathcal{P}(\Theta)$

$H^{CR}, H^{MR}, H^{CB}, H^{AR}, H^{IR}, H^{AB} \leftarrow \{\}, \{\}, \{\}, \{\}, \{\}, \{\}$
$g^{x0} \leftarrow g(\cdot, x0)$ *// partially applying $x0$ to g; i.e, $g^{x0}(\theta) = g(\theta, x0)$.*
$Q \subset AP^{f \circ g^{x0}} \leftarrow \{ap^{f \circ g^{x0}}(\vec{0})\}$ *// queue for breadth-first search.*
$OBS \subset AP^{f \circ g^{x0}} \leftarrow \{\}$ *// observed activation patterns.*
while $\#|Q| > 0$ *// loop for breadth-first search.* **do**
  $p \leftarrow pop(Q)$
  $OBS \leftarrow OBS \cup \{p\}$
  $\eta \leftarrow constructActivationRegion(f \circ g^{x0}, p)$

  // Push the connected activation regions of η.
  $FS \subset \mathbb{Z} \times \mathbb{Z} \leftarrow \{\}$ *// (l, n) means the n-th neuron in l-th layer is a face of η*
  for $l = 1$ to the layer size of DNN $f \circ g^{x0}$, $n = 1$ to the neuron size of the l-th layer **do**
    continue **if** $isStable(p, l, n, \Theta)$ *// skip if neuron activity $p_{l,n}$ cannot flip within Θ.*
    $p' \leftarrow flipped(p, l, n)$ *// flip neuron activity $p_{l,n}$.*
    continue **if** $p' \in OBS$ *// skip if p' has already observed.*
    $OBS \leftarrow OBS \cup \{p'\}$
    $\theta^F \leftarrow calcInteriorPointOnFace(\eta, l, n)$
    continue **if** $\theta^F = null$ *// skip if p' is infeasible.*
    $FS \leftarrow FS \cup \{(l, n)\}$
    $Q \leftarrow Q \cup \{p'\}$ *// push.*
  end for

  // Verify activation region η.
  $\tilde{\eta} \leftarrow simplified(\eta, FS)$ *// limit the constraints on η to FS.*
  if $solveCR(x0; \tilde{\eta})$ **then**
    $H^{CR} \leftarrow H^{CR} \cup \{\tilde{\eta}\}$
  else if $solveMR(x0; \tilde{\eta})$ **then**
    $H^{MR} \leftarrow H^{MR} \cup \{\tilde{\eta}\}$
  else
    $H^{CB} \leftarrow H^{CB} \cup \{\tilde{\eta}\}$
  end if
  if $solveAR(x0; \tilde{\eta})$ **then**
    $H^{AR} \leftarrow H^{AR} \cup \{\tilde{\eta}\}$
  else if $solveIR(x0; \tilde{\eta})$ **then**
    $H^{IR} \leftarrow H^{IR} \cup \{\tilde{\eta}\}$
  else
    $H^{AB} \leftarrow H^{AB} \cup \{\tilde{\eta}^{AB}\}$
  end if
end while
return $H^{CR}, H^{MR}, H^{CB}, H^{AR}, H^{IR}, H^{AB}$