Towards Analyzing Semantic Robustness of Deep Neural Networks

Abdullah Hamdi, Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
{abdullah.hamdi, Bernard.Ghanem} @kaust.edu.sa

Abstract

*Despite the impressive performance of Deep Neural Networks (DNNs) on various vision tasks, they still exhibit erroneous high sensitivity toward semantic primitives (*e.g. *object pose). We propose a theoretically grounded analysis for DNNs robustness in the semantic space. We qualitatively analyze different DNNs semantic robustness by visualizing the DNN global behavior as semantic maps and observe interesting behavior of some DNNs. Since generating these semantic maps does not scale well with the dimensionality of the semantic space, we develop a bottom-up approach to detect robust regions of DNNs. To achieve this, We formalize the problem of finding robust semantic regions of the network as optimization of integral bounds and develop expressions for update directions of the region bounds. We use our developed formulations to quantitatively evaluate the semantic robustness of different famous network architectures. We show through extensive experimentation that several networks, though trained on the same dataset and while enjoying comparable accuracy, they do not necessarily perform similarly in semantic robustness. For example, InceptionV3 is more accurate despite being less semantically robust than ResNet50. We hope that this tool will serve as the first milestone towards understanding the semantic robustness of DNNs.*

1. Introduction

As a result of recent advances in machine learning and computer vision, deep neural networks (DNNS) has become an essential part of our lives. DNNs are used to suggest articles to read, detect people in surveillance cameras, automate big machines in factories, and even diagnose X-rays for patients in hospitals. So What is the catch here? These DNNs struggle from a detrimental weakness on specific naive scenarios, despite having a strong performance on average. Figure 1 shows how a small perturbation on the view angle of the teapot object results in a drop in InceptionV3 [36] confidence score from 100% to almost 0%. The *softmax* confidence scores are plotted against one semantic parameter (*i.e.*, the azimuth angle around the teapot) and it fails in such a simple task. Similar behaviors are consistently observed across different DNNs (trained on ImageNet [31]). Figure 2 shows that even after averaging over different shapes, we observe a similar behaviour, but with less severity.

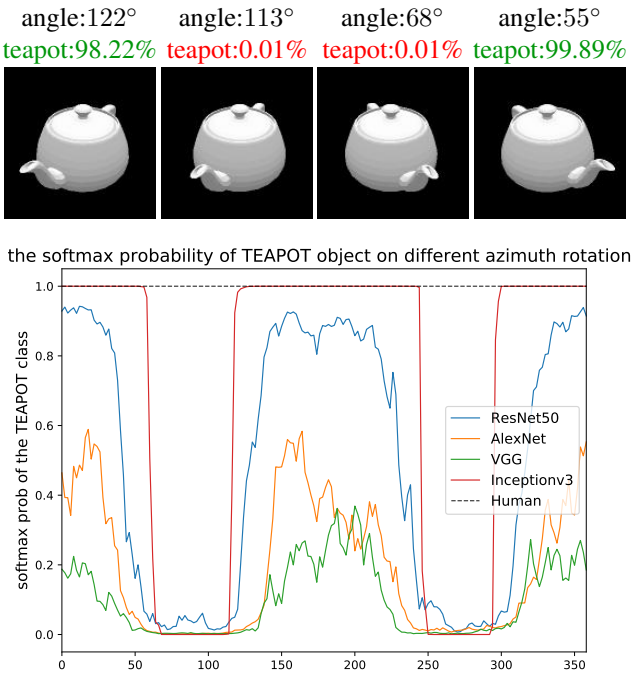

Figure 1: **Semantic Robustness of Deep Networks**. Trained Neural networks can perform poorly for small perturbations in the semantics of the image. (*top*):We show how for a simple teapot object, perturbing the azimuth view angle of the object can dramatically affect the score of InceptionV3 [36] score of the teapot class. (*bottom*):We show a plot of the softmax confidence scores of different DNNs on the the same teapot object viewed from 360 degrees around the object. For comparison, Lab researchers identified the object from all angles (18 equally spaced samples)

Furthermore, because DNNs are not easily interpretable, they work well without a complete understanding of *why* they behave in such manner. A whole direction of research is dedicated to study and analyze DNNs. Examples of such

analysis is activation visualization [9, 40, 25], noise injection [11, 26, 3], and studying effect of image manipulation on DNNs [13, 13, 12]. We provide a new lens of semantic robustness analysis of such DNNs as can be seen in Figure 1 and subsequent figures. These Network Semantic Maps (NSM) demonstrate unexpected behavior of some DNNs in which adversarial regions lies inside a very confident region of the semantic space, which constitutes a "trap" that is hard to detect without such analysis and can lead to catastrophic failure of the DNN.

Recent work in the adversarial attacks explores the DNNs' sensitivity and perform gradient updates to derive targeted perturbations [38, 14, 6, 24]. In practice, such attacks are less likely to naturally occur than semantic attacks, such as changes in camera viewpoint and lighting conditions. The literature on semantic attacks is sparser since they are more subtle and challenging to analyze [41, 18]. This is due to the fact we are not able to distinguish between failure cases that result from the network structure, and learning, or from the data bias [39]. The current methods for adversarial semantic attacks either work on individual examples [1], or try to find distributions but rely on sampling methods which do not scale with dimensionality [18]. We present a novel approach to finds robust/adversarial regions in the n-dimensional semantic space that scale better than sampling-based methods[18], and we use such algorithm to quantify semantic robustness of popular DNNs on a collected dataset.

Contributions. **(1)** We analyze the networks on the semantic lens showing unexpected behavior in the 1,2D semantic space. **(2)** We develop a new method to detect regions in the semantic space that the DNN behaves robustly/poorly that scale well with increasing dimensions (unlike other sampling-based methods). The method is optimization based and follows rigorously from optimizing the bounds around a point of interest. **(3)** We develop a new metric to measure semantic robustness of DNNs that we dub Semantic Robustness Volume Ratio (SRVR), and we use it to benchmark famous DNNs on a collected dataset.

2. Related Work

2.1. Understanding Deep Neural Networks

Deep Learning and DNNs are black-box tools that work well on average on a wide range of tasks. However, due to limited understanding of their behavior, a whole direction of research is dedicated to study and analyze neural networks. There are different lenses to analyze DNNs depending on the purpose of analysis. A famous line of works tries to visualize the network hidden layers by inverting the activations to get a visual image that represents a specific activation [9, 25, 40]. Others observe the behavior of these networks under injected noise [11, 2, 4, 15, 37, 3]. Sax *et al*. analyze the deep networks from the lens of information theory and show how the choice of activation layers affect the learning from entropy perspective [32]. Geirhos *et al*. shows that changing the texture of the object while keeping the borders can hugely deteriorate the recognizability of the object by the DNN [13]. More closely related to our work is the work of Fawzi *et al*., which shows that geometric changes in the image affect the performance of the classifier greatly [12]. They also propose a probabilistic analysis framework to measure the robustness under these nuisance transformations (*e.g*., perspective and illumination) [10], but they are limited to non-semantic 2D image transformation.

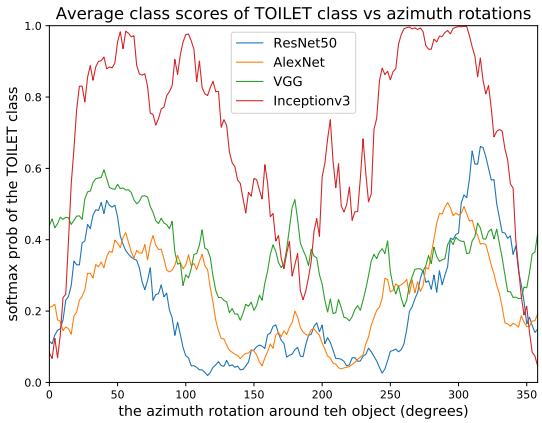

Figure 2: **Class Global Adversarial Regions**: average scores on 10 different shapes from the toilet class. We can see these semantic regions are shared among different shapes of that specific class as well as by different DNNs.

2.2. Adversarial Attacks on Deep Neural Networks

Pixel-based Adversarial Attacks. The way that DNNs fail for some noise added to the image motivated the adversarial attacks literature. Szegedy first introduced a formulation of attacking neural networks as an optimization [38]. The method minimizes the perturbation of the image pixels, while still fooling a trained classifier to predict a wrong class label. Several works followed the same approach but with different formulations [14, 24, 27, 6]. However, all these methods are limited to pixel perturbations and only fool classifiers, while we consider more general cases of attacks, *e.g*. changes in camera viewpoint to fool a DNN by finding adversarial regions. Most of these attacks are white-box attacks, in which the algorithm has access to network gradients. Another direction of adversarial attacks treat the classifiers as a black box, in which the adversary only can quarry points and get a score from the classifier without backpropagating through the DNN [28, 8]. We formulate

the problem of finding the region as an optimization of the corners of hyper-rectangle in the semantic space in both a black box fashion (only function evaluations) as well as the white box formulation which utilizes the gradient of the function.

Semantic Adversarial Attacks. Moving away from pixel perturbation to semantic 3D scene parameters, Zeng *et al*. [41] generate attacks on deep classifiers by perturbing scene parameters like lighting and surface normals. They show that the most common image space adversarial attacks are not authentic and cannot be realized in real 3D scenes. These semantic parameters are the same parameters studied in the Vision as Inverse Graphics paradigm [16, 17]. A completely different approach of VIG is to make the graphics operations differentiable from the beginning, which allows for an easy inverting by taking the gradient of the image to the parameters input directly. The most notable work in differentiable rendering is the Neural Mesh renderer (NMR) by Kato *et al*. [22]. NMR approximates the non differentiable rasterization by a differentiable counterpart, allowing the full rendering pipeline to be differentiable and implementing the technique in Pytorch [29], which lead to broad adoption. We use NMR as our primary rendering method of meshes since we can obtain the gradient of the composite function of the network and the Renderer to the semantic parameters, which is necessary for our Algorithm 2. Recently Hamdi *et al*. proposes generic adversarial attacks that incorporate semantic and pixel attacks, in which they define the adversarial attack as sampling from some latent distribution, and they learn a GAN on filtered semantic samples [18]. However, their work used sampling-based approach to learn these adversarial regions, which does not scale with the dimensionality of the problem.

2.3. Optimizing Integral Bounds

Naive Approach. To develop an algorithm for robust region finding, we adopt an idea of weekly supervised activity detection in videos by Shou *et al*. [33]. The idea works on maximizing the inner average while minimizing the outer average of the function in the region and optimizing the bounds to achieve the objective. This is achieved because optimizing the bounds to maximize the area exclusively can lead to diverging the bounds to $-\infty, \infty$. To solve the issue of diverging bounds, the following naive formulation is simply regularizing the loss by penalty of the norm of the region size. The expressions for the loss of n=1 dimension is $L = -\text{Area}_{\text{in}} + \frac{\lambda}{2}|b-a|_2^2 = \int_a^b f(u)du \ + \frac{\lambda}{2}|b-a|_2^2$, where $f : \mathbb{R}^1 \rightarrow (0,1)$ is the function of interest and a, b are the left and right bound respectively and λ is a hyper-parameter. The update directions to minimize the loss are $\frac{\partial L}{\partial a} = f(a) - \lambda(b-a)$, $\frac{\partial L}{\partial b} = -f(b) + \lambda(b-a)$. The regularizer will prevent the region to grow to ∞ and the best bounds will be found if loss is minimized with gradient descent or any similar approach. To extend the naive approach to n-dimensions, we will face an another integral in the update direction (hard to compute). Therefore, we deploy the following trapezoid approximation for the integral.

Trapezoidal Approximation. The trapezoidal approximation of definite integrals is a first-order approximation from Newton-Cortes formulas for numerical integration [35]. The rule states that $\int_a^b f(u)du \approx (b-a)\frac{f(a)+f(b)}{2}$. An asymptotic error estimate is given by $-\frac{(b-a)^2}{48}[f'(b) - f'(a)] + \mathcal{O}\left(\frac{1}{8}\right)$. So as long the derivatives are bounded by some lipschitz constant $\mathbb{L}$, then the error becomes bounded by the following $|\text{error}| \leq \mathbb{L}(b-a)^2$.

3. Methodology

Typical adversarial pixel attacks involve a neural network agent $\mathbf{C}$ (*e.g*. classifier or detector) that takes an image $\mathbf{x} \in [0,1]^d$ as input and outputs a multinoulli distribution over K class labels with softmax values $[l_1, l_2, ..., l_K]$, where l_j is the softmax value for class j. The adversary (attacker) tries to produce a perturbed image $\mathbf{x}' \in [0,1]^d$ that is as close as possible to $\mathbf{x}$, such that $\mathbf{C}$ changes its class prediction from $\mathbf{x}$ to $\mathbf{x}'$.

In our case we consider a more general case where we are interested in the parameters $\mathbf{u} \in \Omega \subset \mathbb{R}^n$, a hidden latent parameter that generate the image and is passes to scene generator (*e.g*. a renderer function $\mathbf{R}$) that takes the parameter $\mathbf{u}$ and a an object shape $\mathbf{S}$ of a class that is identified by $\mathbf{C}$. Ω is the continuous semantic space for the parameters that we intend to study. The renderer creates the image $\mathbf{x} \in \mathbb{R}^d$, and then we study the behavior of a classifier $\mathbf{C}$ of that image across multiple shapes and multiple famous DNNs. Now, this function of interest is defined as follows

$$f(\mathbf{u}) = \mathbf{C}_z(\mathbf{R}(\mathbf{S}_z, \mathbf{u})) \ , \ \ 0 \leq f(\mathbf{u}) \leq 1 \qquad (1)$$

where z is class label of interest of study and we observe the network score for that class by rendering a shape $\mathbf{S}_z$ of the same class.The shape and class labels are constants and only the parameters varies for f during analysis.

3.1. Region Finding as an Operator

We can visualize the function in Eq (1) for any shape $\mathbf{S}_z$ as long as the DNN can identify the shape at some region in the semantic space Ω of interest, as we did in Figure 1. However , plotting such figure is expensive and the complexity of plotting it increase exponentially with a big base. The complexity of plotting this type of semantic maps (we call Network Semantic Map NSM) is N for $n = 1$, where N is the number of samples needed for that dimension to be fully characterized. The complexity is N^2 for $n = 2$, and we can see that for a general dimension n, the complexity

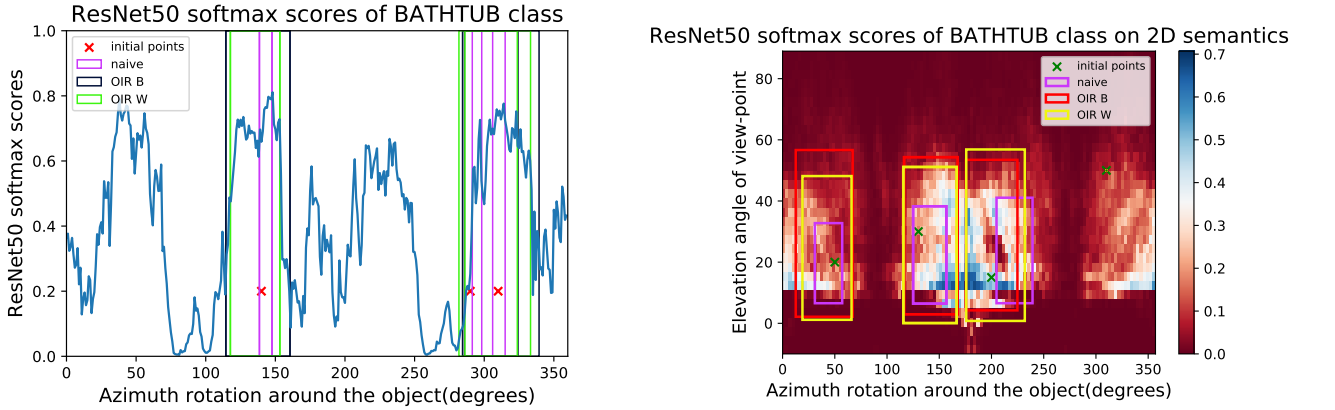

Figure 3: **Semantic Robust Region Finding**: finding robust regions of semantic parameters for Rsnet50 [19] in a bathtub object by the three bottom-up formulations (naive , OIR_W , OIR_B). (*left*): 1D case (azimuth angle of camera) with three initial points. (*right*): 2D case (azimuth angle and elevation angle of camera) with four initial points. We note that the naive approach usually predicts smaller regions, while the OIR formulations finds more comprehensive regions.

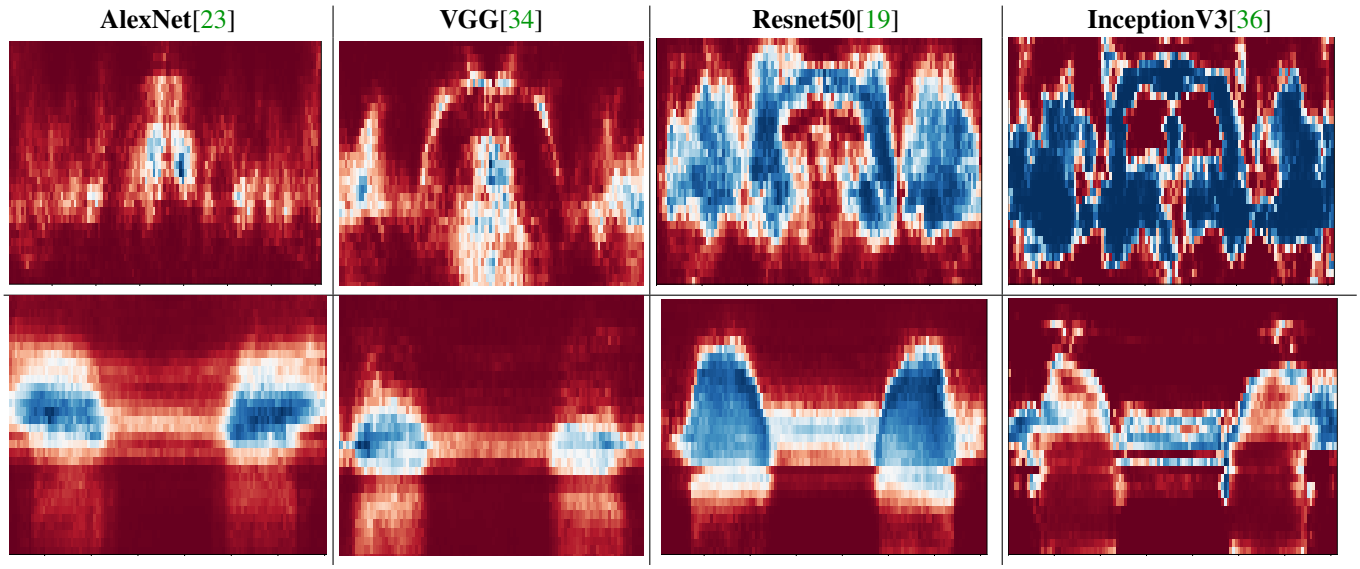

Figure 4: **Network Semantic Maps**: plotting the 2D semantic maps (as in Figure 3 *right*) of four different networks on two shapes of a chair class (*top*) and cup class (*bottom*).We note that InceptionV3 is very confident about its decision , but the cost is that it creates semantic "traps" where a sharp fall of performance happens in the middle of a robust region. This behaviour is more apparent for complex shapes (*e.g.* the chair in *top* row)

of plotting the NMS to fill the semantic space Ω adequately is N^n. This number is huge even if we have only moderate dimensionality. Also note that we don't need to specify the whole Ω before finding the robust region around a point $\mathbf{u}$, which is an advantage of SADA [18]. As we will see in Section 4.4, this approach can be used to characterize the space much more efficiently as explained in Section 5, and we use it to measure robustness as in Section 4.4. Explicitly, we defined the region finding as an operator $\mathbf{\Phi}$ that takes the function of interest in Eq (1) and initial point in the semantic space $\mathbf{u} \in \Omega$, and a shape $\mathbf{S}_z$ of some class z. The operator will return the hyper-rectangle $\mathbb{D} \subset \Omega$ where the DNN is robust in the region and doesn't drop the score of the intended class sharply as well as it keeps identifying the shape with label z as illustrated in Figure 4. The robust-region-finding operator is then defined as follows

$$\begin{aligned} &\mathbf{\Phi}_{\text{robust}}(f(\mathbf{u}), \mathbf{S}_z, \mathbf{u}_0) = \mathbb{D} = \{\mathbf{u} : \mathbf{a} \leq \mathbf{u} \leq \mathbf{b}\} \\ &\text{s.t. } \mathbb{E}_{\mathbf{u}\sim\mathbb{D}}[f(\mathbf{u})] \geq 1 - \epsilon_m \ , \ \ \mathbf{u}_0 \in \mathbb{D} \ , \ \text{VAR}[f(\mathbf{u})] \leq \epsilon_v \end{aligned} \tag{2}$$

where the left and right bounds of $\mathbb{D}$ are $\mathbf{a} = [a_1, a_2, ..., a_n]$ and $\mathbf{b} = [b_1, b_2, ..., b_n]$, respectively. The two samall thresholds ϵ_m, ϵ_v are to insure high performance and low variance of the DNN network in that robust region. We can define the opposite operator which is to find adversarial re-

gions as:

$$\begin{aligned} &\mathbf{\Phi}_{\text{adv}}(f(\mathbf{u}), \mathbf{S}_z, \mathbf{u}_0) = \mathbb{D} = \{\mathbf{u} : \mathbf{a} \leq \mathbf{u} \leq \mathbf{b}\} \\ &\text{s.t. } \mathbb{E}_{\mathbf{u}\sim\mathbb{D}}[f(\mathbf{u})] \leq \epsilon_m \ , \ \ \mathbf{u}_0 \in \mathbb{D} \ , \ \text{VAR}[f(\mathbf{u})] \geq \epsilon_v \end{aligned} \tag{3}$$

We can show clearly that $\mathbf{\Phi}_{\text{adv}}$ and $\mathbf{\Phi}_{\text{robust}}$ are related

$$\mathbf{\Phi}_{\text{adv}}(f(\mathbf{u}), \mathbf{S}_z, \mathbf{u}_0) = \mathbf{\Phi}_{\text{robust}}(1 - f(\mathbf{u}), \mathbf{S}_z, \mathbf{u}_0) \tag{4}$$

So we can just focus our attentions on $\mathbf{\Phi}_{\text{robust}}$ to find robust regions , and the adversarial regions follows directly from Eq (4). In all our methods, the region of interest $\mathbb{D} = \{\mathbf{u} : \mathbf{a} \leq \mathbf{u} \leq \mathbf{b}\}$, Here , we assume the size of the region is positive at every dimension , *i.e.* $\mathbf{r} = \mathbf{b} - \mathbf{a} > \mathbf{0}$. The volume of the region $\mathbb{D}$ normalized by exponent of dimension n is expressed as follows

$$\text{volume}(\mathbb{D}) = \triangle = \frac{1}{2^n} \prod_{i=1}^{n} \mathbf{r}_i \tag{5}$$

The region $\mathbb{D}$ can also be defined in terms of the matrix $\mathbf{D}$ of all the corner points $\{\mathbf{d}^i\}_{i=1}^{2^n}$ as follows.

$$\begin{aligned} \text{corners}(\mathbb{D}) = \mathbf{D}_{n\times 2^n} &= \left[\mathbf{d}^1 | \mathbf{d}^2 | .. | \mathbf{d}^{2^n}\right] \\ \mathbf{D} &= \mathbf{1}^T \mathbf{a} \ + \ \mathbf{M}^T \odot (\mathbf{1}^T \mathbf{r}) \\ \mathbf{M}_{n\times 2^n} = \left[\mathbf{m}^0 | \mathbf{m}^1 | .. | \mathbf{m}^{2^n-1}\right] &\ , \text{ where } \ \mathbf{m}^i = \text{binary}_n(i) \end{aligned} \tag{6}$$

where $\mathbf{1}$ is the all-ones vector of size 2^n, $\odot$ is the Hadamard product of matrices (element-wise) , and $\mathbf{M}$ is a constant masking matrix defined as the matrix of binary numbers of n bits that range from 0 to $2n-1$

3.2. Deriving Update Directions

Extending Naive to n-dimensions. We start by defining the function vector $\mathbf{f}_{\mathbb{D}}$ of all function evaluations at all corner points of $\mathbb{D}$

$$\mathbf{f}_{\mathbb{D}} = \left[f(\mathbf{d}^1), f(\mathbf{d}^2), ..., f(\mathbf{d}^{2^n})\right]^T \ , \ \mathbf{d}^{\mathbf{i}} = \mathbf{D}_{:,i} \tag{7}$$

Then using Trapezoid approximation and Leibniz rule of calculus, the loss expression and the update directions.

$$\begin{aligned} L(\mathbf{a}, \mathbf{b}) &= -\int \cdots \int_{\mathbb{D}} f(u_1, \ldots, u_n)\, du_1 \ldots du_n + \frac{\lambda}{2} |\mathbf{r}|^2 \\ &\approx \ -\triangle \mathbf{1}^T \mathbf{f}_{\mathbb{D}} \ + \ \frac{\lambda}{2} |\mathbf{r}|^2 \\ \nabla_{\mathbf{a}} L &\approx \ 2\triangle \text{diag}^{-1}(\mathbf{r}) \overline{\mathbf{M}} \mathbf{f}_{\mathbb{D}} + \lambda \mathbf{r} \\ \nabla_{\mathbf{b}} L &\approx \ -2\triangle \text{diag}^{-1}(\mathbf{r}) \mathbf{M} \mathbf{f}_{\mathbb{D}} - \lambda \mathbf{r} \end{aligned} \tag{8}$$

We show all the derivations for n=1,n=2, and for general-n expression explicitly in the **Appendix**.

Outer-Inner Ratio Loss (OIR). We introduce an outer region A, B with bigger area that contains the small region (a, b). We follow the following assumption to insure that outer area is always positive $A = a - \alpha\frac{b-a}{2}, B = b + \alpha\frac{b-a}{2}$, where α is the small boundary factor of the outer area to inner area. we formulate the problem as a ratio of outer over inner area and we try to make this ratio as close as possible to 0 . $L = \frac{\text{Area}_{\text{out}}}{\text{Area}_{\text{in}}}$ By using DencklBeck technique for solving non-linear fractional programming problems [30]. Using their formulation to transform L as follows.

$$\begin{aligned} L = \frac{\text{Area}_{\text{out}}}{\text{Area}_{\text{in}}} \ &= \ \text{Area}_{\text{out}} \ - \ \lambda \, \text{Area}_{\text{in}} \\ &= \int_A^B f(a)da \ - \ \int_a^b f(a)da \ - \ \lambda \int_a^b f(a)da \end{aligned} \tag{9}$$

where $\lambda^* = \frac{\text{Area}^*_{\text{out}}}{\text{Area}^*_{\text{in}}}$ is the DencklBeck factor and it is equal to the small objective best achieved.

Black-Box (OIR_B). Here we set $\lambda = 1$ to simplify the problem. This yields the following expression of the loss $L = \text{Area}_{\text{out}} - \text{Area}_{\text{in}} = \int_A^B f(u)du - 2\int_a^b f(u)du$ which is similar to the area contrastive loss in [33]. The update rules would be $\frac{\partial L}{\partial a} = -(1 + \frac{\alpha}{2})f(A) - \frac{\alpha}{2}f(B) + 2f(a)$
$\frac{\partial L}{\partial b} = (1 + \frac{\alpha}{2})f(B) + \frac{\alpha}{2}f(A) - 2f(b)$

To extend to n-dimensions, we define an outer bigger region $\mathbb{Q}$ that include the smaller region $\mathbb{D}$ and defined as : $\mathbb{Q} = \{\mathbf{u} : \mathbf{a} - \frac{\alpha}{2}\mathbf{r} \leq \mathbf{u} \leq \mathbf{b} + \frac{\alpha}{2}\mathbf{r}\}$, where $\mathbf{a}, \mathbf{b}, \mathbf{r}$ are defined as before, while α is defined as the boundary factor of the outer region in all the dimensions equivalently. The inner region $\mathbb{D}$ is defined as in Eq (6) and outer regions can also be defined in terms of the corner points as follows.

$$\begin{aligned} \text{corners}(\mathbb{Q}) = \mathbf{Q}_{n\times 2^n} &= \left[\mathbf{q}^1 | \mathbf{q}^2 | .. | \mathbf{q}^{2^n}\right] \\ \mathbf{Q} &= \mathbf{1}^T (\mathbf{a} - \frac{\alpha}{2}\mathbf{r}) \ + \ (1 + \alpha)\mathbf{M}^T \odot (\mathbf{1}^T \mathbf{r}) \end{aligned} \tag{10}$$

Let $\mathbf{f}_{\mathbb{D}}$ be function vector as in Eq (7) and $\mathbf{f}_{\mathbb{Q}}$ be another function vector evaluated at all possible outer corner points as follows.

$$\mathbf{f}_{\mathbb{Q}} = \left[f(\mathbf{q}^1), f(\mathbf{q}^2), ..., f(\mathbf{q}^{2^n})\right]^T \ , \ \mathbf{q}^{\mathbf{i}} = \mathbf{Q}_{:,i} \tag{11}$$

Now the loss and update directions for the n-dimensional case becomes as follows .

$$\begin{aligned} L(\mathbf{a}, \mathbf{b}) &= \int \cdots \int_{\mathbb{Q}} f(u_1, \ldots, u_n)\, du_1 \ldots du_n \\ &\quad - 2\int \cdots \int_{\mathbb{D}} f(u_1, \ldots, u_n)\, du_1 \ldots du_n \\ &\approx \triangle \left((1 + \alpha)^n \mathbf{1}^T \mathbf{f}_{\mathbb{Q}} \ - \ 2\,\mathbf{1}^T \mathbf{f}_{\mathbb{D}}\right) \\ \nabla_{\mathbf{a}} L \approx &2\triangle \text{diag}^{-1}(\mathbf{r}) \left(2\overline{\mathbf{M}} \mathbf{f}_{\mathbb{D}} \ - \ \overline{\mathbf{M}}_{\mathbb{Q}} \mathbf{f}_{\mathbb{Q}}\right) \\ \nabla_{\mathbf{b}} L \approx &2\triangle \text{diag}^{-1}(\mathbf{r}) \left(-2\mathbf{M} \mathbf{f}_{\mathbb{D}} \ + \ \mathbf{M}_{\mathbb{Q}} \mathbf{f}_{\mathbb{Q}}\right) \end{aligned} \tag{12}$$

Where diag(.) is the diagonal matrix of the vector argument or the diagonal vector of the matrix argument. $\overline{\mathbf{M}}_{\mathbb{Q}}$ is the outer region constant matrix defined as follows.

$$\begin{aligned}\overline{\mathbf{M}}_{\mathbb{Q}} &= (1+\alpha)^{n-1}\left((1+\frac{\alpha}{2})\overline{\mathbf{M}}+\frac{\alpha}{2}\mathbf{M}\right)\\ \mathbf{M}_{\mathbb{Q}} &= (1+\alpha)^{n-1}\left((1+\frac{\alpha}{2})\mathbf{M}+\frac{\alpha}{2}\overline{\mathbf{M}}\right)\end{aligned} \tag{13}$$

White-Box OIR (OIR_W.) The following formulation is white-box in nature (it need the gradient of the function f in order to update the current estimates of the bound) this is useful when the function in hand is differntiable (*e.g.* DNN) , to obtain more intelligent regions ,rather then the regions surrounded by near 0 values of the function f. We set $\lambda = \frac{\alpha}{\beta}$ in Eq (9), where α is the small boundary factor of the outer area, β is the emphasis factor (we will show later how it determines the emphasis on the function vs the gradient). Hence, the objective in Eq (9) becomes :

$$\begin{aligned}&\arg\min_{a,b} L = \arg\min_{a,b} \text{ Area}_{\text{out}} - \lambda \text{ Area}_{\text{in}}\\ &= \arg\min_{a,b} \int_A^a f(u)du + \int_b^B f(u)du - \frac{\alpha}{\beta}\int_a^b f(u)du\\ &= \arg\min_{a,b} \frac{\beta}{\alpha}\int_{a-\alpha\frac{b-a}{2}}^{b+\alpha\frac{b-a}{2}} f(u)du - (1+\frac{\beta}{\alpha})\int_a^b f(u)du\\ \frac{\partial L}{\partial a} &= \frac{\beta}{\alpha}\left(f(a) - f\left(a-\alpha\frac{b-a}{2}\right)\right)\\ &\quad - \frac{\beta}{2}f\left(b+\alpha\frac{b-a}{2}\right) - \frac{\beta}{2}f\left(a-\alpha\frac{b-a}{2}\right) + f(a)\end{aligned} \tag{14}$$

now since λ^* should be small for the optimal objective as $\lambda \to 0$, $\alpha \to 0$ and hence the derivative in Eq (14) becomes the following.

$$\begin{aligned}\lim_{\alpha\to 0}\frac{\partial L}{\partial a} &= \frac{\beta}{2}\left((b-a)f'(a)+f(b)\right) + (1-\frac{\beta}{2})f(a)\\ \lim_{\alpha\to 0}\frac{\partial L}{\partial b} &= \frac{\beta}{2}\left((b-a)f'(b)+f(a)\right) - (1-\frac{\beta}{2})f(b)\end{aligned} \tag{15}$$

we can see that the update rule for a and b depends on the function value **and** the derivative of f at the boundaries a and b respectively, with β controlling the dependence. If $\beta \to 0$, the update directions in Eq (15) collapse to the unregularized naive update. To extend to n-dimensions, we have to define a term that involves the gradient of the function , which is the all-corners gradient matrix $\mathbf{G}_{\mathbb{D}}$.

$$\mathbf{G}_{\mathbb{D}} = \left[\nabla f(\mathbf{d}^1) \mid \nabla f(\mathbf{d}^2) \mid ... \mid \nabla f(\mathbf{d}^{2^n})\right]^T \tag{16}$$

Algorithm 1: Robust n-dimensional Region Finding for Black-Box DNNs by Outer-Inner Ratios

Requires: Semantic Function of a DNN $f(\mathbf{u})$ in Eq (1), initial semantic parameter $\mathbf{u}_0$, number of iterations T , learning rate η , object shape $\mathbf{S}_z$ of class label z, boundary factor α, Small ϵ

Form constant binary matrices $\mathbf{M}, \overline{\mathbf{M}}, \mathbf{M}_{\mathbb{Q}}, \overline{\mathbf{M}_{\mathbb{Q}}}, \mathbf{M}_{\mathbb{D}}, \overline{\mathbf{M}_{\mathbb{D}}}$

Initialize bounds $\mathbf{a}_0 \leftarrow \mathbf{u}_0 - \epsilon\mathbf{1}$, $\mathbf{b}_0 \leftarrow \mathbf{u}_0 + -\epsilon\mathbf{1}$

$\mathbf{r}_0 \leftarrow \mathbf{a}_0 - \mathbf{b}_0$, update region volume $\triangle_0$ as in Eq (5)

for $t \leftarrow 1$ **to** T **do**

  form the all-corners function vectors $f_{\mathbb{D}}, f_{\mathbb{Q}}$ as in Eq (11)

  $\nabla_{\mathbf{a}} L \leftarrow 2\triangle_{t-1}\text{diag}^{-1}(\mathbf{r}_{t-1})\left(2\overline{\mathbf{M}}\mathbf{f}_{\mathbb{D}} - \overline{\mathbf{M}}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\right)$

  $\nabla_{\mathbf{b}} L \leftarrow 2\triangle_{t-1}\text{diag}^{-1}(\mathbf{r}_{t-1})\left(-2\mathbf{M}\mathbf{f}_{\mathbb{D}} + \mathbf{M}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\right)$

  update bounds: $\mathbf{a}_t \leftarrow \mathbf{a}_{t-1} - \eta\nabla_{\mathbf{a}} L$, $\mathbf{b}_t \leftarrow \mathbf{b}_{t-1} - \eta\nabla_{\mathbf{b}} L$

  $\mathbf{r}_t \leftarrow \mathbf{a}_t - \mathbf{b}_t$, update region volume $\triangle_t$ as in Eq (5)

end

Returns: robust region bounds: $\mathbf{a}_T, \mathbf{b}_T$.

Algorithm 2: Robust n-dimensional Region Finding for White-Box DNNs by Outer-Inner Ratios

Requires: Semantic Function of a DNN $f(\mathbf{u})$ in Eq (1), initial semantic parameter $\mathbf{u}_0$, , learning rate η , object shape $\mathbf{S}_z$ of class label z, emphasis factor β, Small ϵ

Form constant binary matrices $\mathbf{M}, \overline{\mathbf{M}}, \mathbf{M}_{\mathbb{D}}, \overline{\mathbf{M}_{\mathbb{D}}}$

Initialize bounds $\mathbf{a}_0 \leftarrow \mathbf{u}_0 - \epsilon\mathbf{1}$, $\mathbf{b}_0 \leftarrow \mathbf{u}_0 + -\epsilon\mathbf{1}$

$\mathbf{r}_0 \leftarrow \mathbf{a}_0 - \mathbf{b}_0$, update region volume $\triangle_0$ as in as in Eq (5)

for $t \leftarrow 1$ **to** T **do**

  form the all-corners function vector $f_{\mathbb{D}}$ as in Eq (11)

  form the all-corners gradients matrix $\mathbf{G}_{\mathbb{D}}$ as in Eq (16)

  form the gradient selection vectors $\mathbf{s}, \bar{\mathbf{s}}$ as in Eq (19)

  $\nabla_{\mathbf{a}} L \leftarrow$

  $\triangle_{t-1}\left(\text{diag}^{-1}(\mathbf{r}_{t-1})\overline{\mathbf{M}}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\overline{\mathbf{M}}\mathbf{G}_{\mathbb{D}} + \beta\bar{\mathbf{s}}\right)$

  $\nabla_{\mathbf{b}} L \leftarrow$

  $\triangle_{t-1}\left(-\text{diag}^{-1}(\mathbf{r}_{t-1})\mathbf{M}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\mathbf{M}\mathbf{G}_{\mathbb{D}}) + \beta\mathbf{s}\right)$

  update bounds: $\mathbf{a}_t \leftarrow \mathbf{a}_{t-1} - \eta\nabla_{\mathbf{a}} L$, $\mathbf{b}_t \leftarrow \mathbf{b}_{t-1} - \eta\nabla_{\mathbf{b}} L$

  $\mathbf{r}_t \leftarrow \mathbf{a}_t - \mathbf{b}_t$, update region volume $\triangle_t$ as in Eq (5)

end

Returns: robust region bounds: $\mathbf{a}_T, \mathbf{b}_T$.

Now, the loss and update directions are given as follows.

$$\begin{aligned}L(\mathbf{a},\mathbf{b}) &\approx \frac{(1+\alpha)^n\mathbf{1}^T\mathbf{f}_{\mathbb{Q}}}{\mathbf{1}^T\mathbf{f}_{\mathbb{D}}} - 1\\ \nabla_{\mathbf{a}} L &\approx \triangle\left(\text{diag}^{-1}(\mathbf{r})\overline{\mathbf{M}}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\overline{\mathbf{M}}\mathbf{G}_{\mathbb{D}}) + \beta\bar{\mathbf{s}}\right)\\ \nabla_{\mathbf{b}} L &\approx \triangle\left(-\text{diag}^{-1}(\mathbf{r})\mathbf{M}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\mathbf{M}\mathbf{G}_{\mathbb{D}}) + \beta\mathbf{s}\right)\end{aligned} \tag{17}$$

where the mask is the special mask

$$\begin{aligned}\overline{\mathbf{M}}_{\mathbb{D}} &= \left(\gamma_n\overline{\mathbf{M}} - \beta\mathbf{M}\right)\\ \mathbf{M}_{\mathbb{D}} &= \left(\gamma_n\mathbf{M} - \beta\overline{\mathbf{M}}\right)\\ \gamma_n &= 2 - \beta(2n-1)\end{aligned} \tag{18}$$

$\mathbf{s}$ is a weighted sum of the gradient from other dimensions ($i \neq k$) contributing to the update direction of dimension k, where $k \in \{1, 2, ..., n\}$.

$$\mathbf{s}_k = \frac{1}{\mathbf{r}_k} \sum_{i=1, i\neq k}^{n} \mathbf{r}_i((\overline{\mathbf{M}}_{i,:} - \mathbf{M}_{i,:}) \odot \overline{\mathbf{M}}_{k,:})\mathbf{G}_{:,i}$$
$$\bar{\mathbf{s}}_k = \frac{1}{\mathbf{r}_k} \sum_{i=1, i\neq k}^{n} \mathbf{r}_i((\mathbf{M}_{i,:} - \overline{\mathbf{M}}_{i,:}) \odot \mathbf{M}_{k,:})\mathbf{G}_{:,i} \quad (19)$$

The derivation of the 2-dimensional case and n-dimensional case of the OIR formulation is included in the **Appendix**. We try to use the trapezoid approximation directly on the loss and then differentiate the approximation. We get an expression that involves the derivative of the function, and we obtain an n-dimensional extension for it in the **Appendix**. However, when applying to find the region it diverges (probably due to large approximations error). Algorithms 1, and 2 summarize the techniques.

4. Experiments

4.1. Setup and Data

The semantic parameters $\mathbf{u}$ we pick are the azimuth rotations of the viewpoint and the elevation angle from the horizontal plane where the object is always at the center of the rendering, which is common in the literature [18, 20]. We use 100 shapes from 10 different classes from ShapeNet [7], the largest dataset for 3D models that are normalized in the semantic lens. We filter these 100 shapes from much more shapes to make sure that: (1) the class label is available in ImageNet and that ImageNet classifiers can identify the exact class, (2) the selected shapes are identified by the classifiers at some part of the semantic space. To do this, we measured the average score in the space and accepted the shape only if its average Resnet softmax score is 0.1. To render the images we use differentiable renderer NMR [22] which allows obtaining the gradient to the semantic input parameters.The networks of interest were Resnet50 [19], VGG [34], AlexNet [23], and InceptionV3 [36]. We use the official implementation by Pytorch models [29]. this DNNs

4.2. Mapping the Networks

We map the networks similar to Figure 1, but for all the 100 shapes on the first semantic parameter (the azimuth rotation) as well as the joint (azimuth and elevation) and show the results in Figure 4. The ranges for the two parameters were [0°,360°],[−10°,90°], with 3*3 grid. The total of evaluations is 4K forward passes from every network for every shape (total of 1.6 M forward passes). We show all of the remaining results in the **Appendix** .

Deep Networ	SRVR	Top-1 error	Top-5 Error
AlexNet [23]	8.87%	43.45	20.91
VGG-11 [34]	9.72%	30.98	11.37
ResNet50 [19]	**16.79%**	23.85	7.13
Inceptionv3 [36]	7.92%	**22.55**	**6.44**

Table 1: **Benchmarking famous DNNs in Semantic Robustness vs error rate**. We develop Semantic Robustness Volume Ratio (SRVR) metric to estimate semantic robustness of famous Networks in Section 4.4. We see that semantic robustness doesn't necessarily depends on the accuracy of the DNN, which motivates studying them as an independent metric from the classification accuracy. Errors are reported from Pytorch official implementation, which we use[29].

4.3. Growing Semantic Robust Regions

We implement the three bottom-up approaches in Table 2 and Algorithms 1 and 2 and open-source the code on GitHub [1] and provide a tutorial notebook [2]. The hyperparameters were set to $\eta = 0.1, \alpha = 0.05, \beta = 0.0009\lambda = 0.1, T = 800$. We can observe in Figure 3 that multiple initial points inside the same robust region converge to the same boundary. One key difference to be noted between the naive approach in Eq (8) and the OIR formulations in Eq (12,17) is that naive approach fails to capture robust regions in some scenarios and fall for trivial regions (see Figure 3).

4.4. Applications

Quantifying Semantic Robustness.
Looking at these NSM can lead to insights about the network, but we would like to develop a systemic approach to quantify the robustness of these DNNs. To do this, we develop the Semantic Robustness Volume Ratio (SRVR) metric. The SVR metric of the network is only the ratio between the expected size of the robust region obtained by Algorithms 1,2 over the nominal total volume of the semantic map of interest. Explicitly, the SRVR of network $\mathbf{C}$ for class label z is defined as follows.

$$\text{SRVR}_z = \frac{\mathbb{E}[\text{Vol}(\mathbb{D})]}{\text{Vol}(\Omega)} = \frac{\mathbb{E}_{\mathbf{u}_0 \sim \Omega, \mathbf{S}_z \sim \mathbb{S}_z}[\text{Vol}(\mathbf{\Phi}(f, \mathbf{S}_z, \mathbf{u}_0))]}{\text{Vol}(\Omega)} \quad (20)$$

where f, Φ are defined in Eq (1,2) respectively. We take the average volume of all the adversarial regions found for multiple initilzations and multiple shapes of the same class z and then divide by the nominal volume of the entire space . This gives a percentage of how close is the DNN from the

[1] https://github.com/ajhamdi/semantic-robustness
[2] https://colab.research.google.com/drive/1cZzTPu1uwftnRLqtIIjjqw-YZSKh4QYn

Analysis Approach	Paradigm	Total Sampling complexity	Black -box Functions	Forward pass /step	Backward pass /step	Identification Capabaility	Hyper-parameters
Grid Sampling	top-down	$\mathcal{O}(N^n)$ $N \gg 2$	✓	-	-	Fully identifies the semantic map of DNN	no hyper-parameters
Naive	bottom-up	$\mathcal{O}(2^n)$	✓	2^n	0	finds strong robust regions only around $\mathbf{u}_0$	λ, experimentally determined
OIR_B	bottom-up	$\mathcal{O}(2^{n+1})$	✓	2^{n+1}	0	finds strong and week robust regions around $\mathbf{u}_0$	α, experimentally determined
OIR_W	bottom-up	$\mathcal{O}(2^n)$	✗	2^n	2^n	finds strong and week robust regions around $\mathbf{u}_0$	$0 \leq \beta < \frac{2}{2n-1}$ dependeds on n and Lipschitz constant $\mathbb{L}$

Table 2: **Semantic Analysis Techniques**: comparing different approaches to analyse the semantic robustness of DNN.

ideal behaviour of identifying the object robustly in the entire space. The SRVR metric is not strict in its value since the analyzer define the semantic space of interest and the shapes used. However, SRVR relative score from one DNN to another DNN is of extreme importance as it conveys information about the network that might not be clear by only observing the accuracy of the network. For example, we can see in Table 1 that while InceptionV3 [36] is the best in terms of accuracy, it lags behind Resnet50[19] in terms of semantic robustness. This observation is also consistent with the qualitative NSMs in Figure 4 in which we can see that while Inception is very confident, it can fail completely inside these confident regions. Note that the reported SRVR results are averaged over all the 10 classes over all the 100 shapes, and we use 4 constant initial points for all experiments and the semantic parameters are the azimuth and elevation as in Figure 3,4. As can be seen in Figure 3 different methods predict different regions , so we take the average size of the the three methods used (naive, OIR_W, OIR_B) to give a middle estimate of the volume used in the SRVR results reported in Table 1.

Finding Semantic Bias in the Data.
While looking at the above figures are mesmerizing and can generate a lot of insight about the DNNs and the training data of ImageNet [31], that does not allow to make a conclusion either about the network nor about the data. Therefore, we can average these semantic maps of these networks to factor out the effect of the network structure and training and maintain only the data effect . we show two such maps (we call Data Semantic Map DSM). We can see that the networks have holes in the semantic maps that are shared among DNNs, indicating bias in the data. Identifying this gap in the data can help training a more semantically robust network by adversarial training on these data-based adversarial regions as performed in the adversarial attack literature [14]. Figure shows an example of such semantic map which shows how the data in ImageNet [31] did not have such angles of the easy cup class.

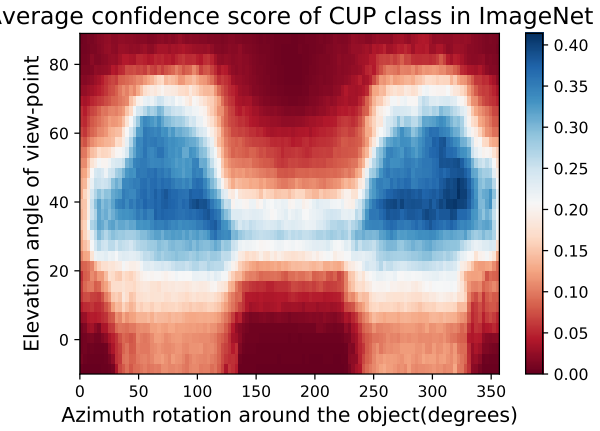

Figure 5: **Semantic Bias in ImageNet**. By taking the average semantic maps over 10 shapes of cup class and over different networks, we get a visualization of the bias of the data. Those angles of low score are probably not well-represented in ImageNet[31].

5. Analysis

First, by observing the figures of the 1D plots (as in Figure 2), we can see that we can have robust regions in which lie adversarial regions. These "traps" are dangerous in ML since it is hard to identify (without NSM) and they can cause failure cases for some models. These traps can be either attributed to the model architecture and training and loss or can be attributed to the bias in the dataset from which the model was trained (*i.e.* ImageNet [31]. In Section 4.4 we study the effect of data bias on these results. Here, we are interested more on the effect of the training, architecture, and any reason that might create a unique signature of the network on the 2D semantic map. One obvious reason is the trade-off between accuracy and robustness, which is well established in the field of pixel perturbation robustness [3, 11].

Note plotting NMS is extremely expensive even for a moderate dimensionality *e.g.* $n = 8$. To see this , for the plot in Figure 1 we use $N = 180$ points in the range of 360

degrees. If all the other dimensionality require the same number $N = 180$ of samples for their individual range , the total joint space requires $180^n = 180^8 = 1.1 * 10^{18}$ samples, which is million times the number of stars in the universe. Evaluating the DNN for that many forward passes is impossible. Thus, we follow a different approach, a bottom-up one, where we start from one point in the semantic space $\mathbf{u}_0$ and we grow an n-dimensional hyper-rectangle around that point to find the robust "neighborhood" of that point for this specific neural network. For example in Figure 1, we have 3 robust regions , so we would expect $3^8 = 6561$ regions if $n = 8$, and we need about $2^8 = 256$ samples to fill in all the regions (assuming the regions fill half the space), and $500 * 2^n = 128K$ samples to find the region around the point. So in total we only need $500 * 4^n * 3^n = 8.*10^8 \ll 1.1*10^{18}$ samples to characterize the space. This what motivates the use of region growing in this work. Table 1 compares different analysis approaches for semantic robustness of DNNs.

We can see that InceptionV3 is the most accurate network (Table 1), but it is less robust (Table 1 and Figure 4) and it jumps sharply between very confident regions to almost zero confidence. This behavior violates our condition for the robust semantic region as in Eq (2), and it is reflected on the qualitative and quantitative results. A possible explanation of this behaviour of disassociation between cognition and abstract geometric inception comes from Neuroscience. Recent research shows that human brain process geometry and other semantic aspects unconsciously, before using the conscious-mind to identify objects[21], which indicates primitive understanding can be disassociate from the identification task (*e.g.* classification).

6. Conclusion

We analyse DNNs from a semantic lens and show how more confident networks tend to create adversarial semantic regions inside highly confident regions. We developed a bottom-up approach to analyse the networks semantically by growing adversarial regions, which scales well with dimensionality and we use it to benchmark the semantic robustness of DNNs. We aim to investigate how to use the insights we gain from this work to develop and train semantically robust networks from the start while maintaining the accuracy. Another direct extension of our work is to develop large scale semantic robustness challenge where we label these robust/adversarial regions in the semantic space and release some of them to allow for training and then we test the trained models on hidden test set to measure robustness while reporting the accuracy on ImageNet validation set to make sure that the features of the model did not get affected by the robust training.

References

[1] M. A. Alcorn, Q. Li, Z. Gong, C. Wang, L. Mai, W. Ku, and A. Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. *CoRR*, abs/1811.11553, 2018. 2, 30

[2] G. An. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996. 2

[3] A. Bibi, M. Alfadly, and B. Ghanem. Analytic expressions for probabilistic moments of pl-dnn with gaussian input. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 8

[4] C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Training*, 7(1), 2008. 2

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 23

[6] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2017. 2

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 7, 30

[8] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 15–26, New York, NY, USA, 2017. ACM. 2

[9] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837, 2016. 2

[10] A. Fawzi and P. Frossard. Measuring the effect of nuisance variables on classifiers. In *BMVC*, 2016. 2

[11] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, 2016. 2, 8

[12] A. Fawzi, S. M. Moosavi Dezfooli, and P. Frossard. The robustness of deep networks - a geometric perspective. *IEEE Signal Processing Magazine*, 2017. 2

[13] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018. 2

[14] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 2, 8

[15] Y. Grandvalet, S. Canu, and S. Boucheron. Noise injection: Theoretical prospects. *Neural Computation*, 9(5):1093–1108, 1997. 2

[16] U. Grenander. *Pattern analysis: lectures in pattern theory, volume I*. Springer, 1978. 3

[17] U. Grenander. *Pattern analysis: lectures in pattern theory, volume II*. Springer, 1978. 3

[18] A. Hamdi, M. Müller, and B. Ghanem. SADA: semantic adversarial diagnostic attacks for autonomous applications. *CoRR*, abs/1812.02132, 2018. 2, 3, 4, 7, 30

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 4, 7, 8

[20] H. Hosseini and R. Poovendran. Semantic adversarial examples. *CoRR*, abs/1804.00499, 2018. 7

[21] D. Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, 2011. 9

[22] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. 3, 7

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 4, 7

[24] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016. 2

[25] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. *CoRR*, abs/1412.0035, 2014. 2

[26] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[27] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[28] A. Nitin Bhagoji, W. He, B. Li, and D. Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[29] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 3, 7

[30] R. G. Ródenas, M. L. López, and D. Verastegui. Extensions of dinkelbach's algorithm for solving non-linear fractional programming problems. *Top*, 7(1):33–70, Jun 1999. 5, 24

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. 1, 8, 11, 21, 22

[32] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018. 2

[33] Z. Shou, H. Gao, L. Zhang, K. Miyazawa, and S.-F. Chang. Autoloc: weakly-supervised temporal action localization in untrimmed videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 154–171, 2018. 3, 5, 23

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4, 7

[35] A. H. Stroud. Methods of numerical integration (philip j. davis and philip rabinowitz). *SIAM Review*, 18(3):528–2, 07 1976. Copyright - Copyright] 1976 Society for Industrial and Applied Mathematics; Last updated - 2012-03-05; CODEN - SIREAD. 3, 28

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 1, 4, 7, 8

[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *ICLR*, 2013. 2

[38] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. 2

[39] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 2

[40] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Hoggles: Visualizing object detection features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, 2013. 2

[41] X. Zeng, C. Liu, Y. Wang, W. Qiu, L. Xie, Y. Tai, C. Tang, and A. L. Yuille. Adversarial attacks beyond the image space. *CoRR*, abs/1711.07183, 2017. 2, 3

A. Analyzing Deep Neural Networks

Here we visualize different Network semantic maps generated during our analysis. In the 1D case we fix the elevation of the camera to a nominal angle of 35° and rotate around the object. In the 2D case, we change both the elevation and azimuth around the object. These maps can be generated to any type of semantic parameters that affect the generation of the image , and not viewing angle.

A.1. Networks Semantic Maps (1D)

In Figure 6,7 we visualize the 1D semantic maps of rotating around the object and recording different DNNs performance and averaging the profile over 10 different shapes per class.

A.2. Networks Semantic Maps (2D)

In Figure 8,9 we visualize the 2D semantic maps of elevation angles and rotating around the object and recording different DNNs performance and averaging the maps over 10 different shapes per class.

A.3. Convergence of the Region Finding Algorithms

Here we show how when we apply the region detection algorithms; the naive detect the smallest region while the OIR formulations detect bigger more general robust region. This result happens even with different initial points; they always converge to the same bounds of that robust region of the semantic maps. Figure 10 show 4 different initializations for 1D case along with predicted regions. In Figure 11,12 shows the bounds evolving during the optimization of the three algorithms (naive , OIR_B and OIR_W) for 500 steps.

A.4. Examples of Found Regions (with Example Renderings)

In Figure 13,14 we provide examples of 2D regions found with the three algorithms along with renderings of the shapes from the robust regions detected.

A.5. Analyzing Semantic Data Bias in ImageNet

In Figure 15,16, we visualizing semantic data bias in the common training dataset (*i.e*. ImageNet [31]) by averaging the Networks Semantic Maps (NSM) of different networks and on different shapes, Different classes have a different semantic bias in ImageNet as clearly shown in the maps above. These places of high confidence probably reveal the areas where an abundance of examples exists in ImageNet, while holes convey scarcity of such examples in ImageNet corresponding class.

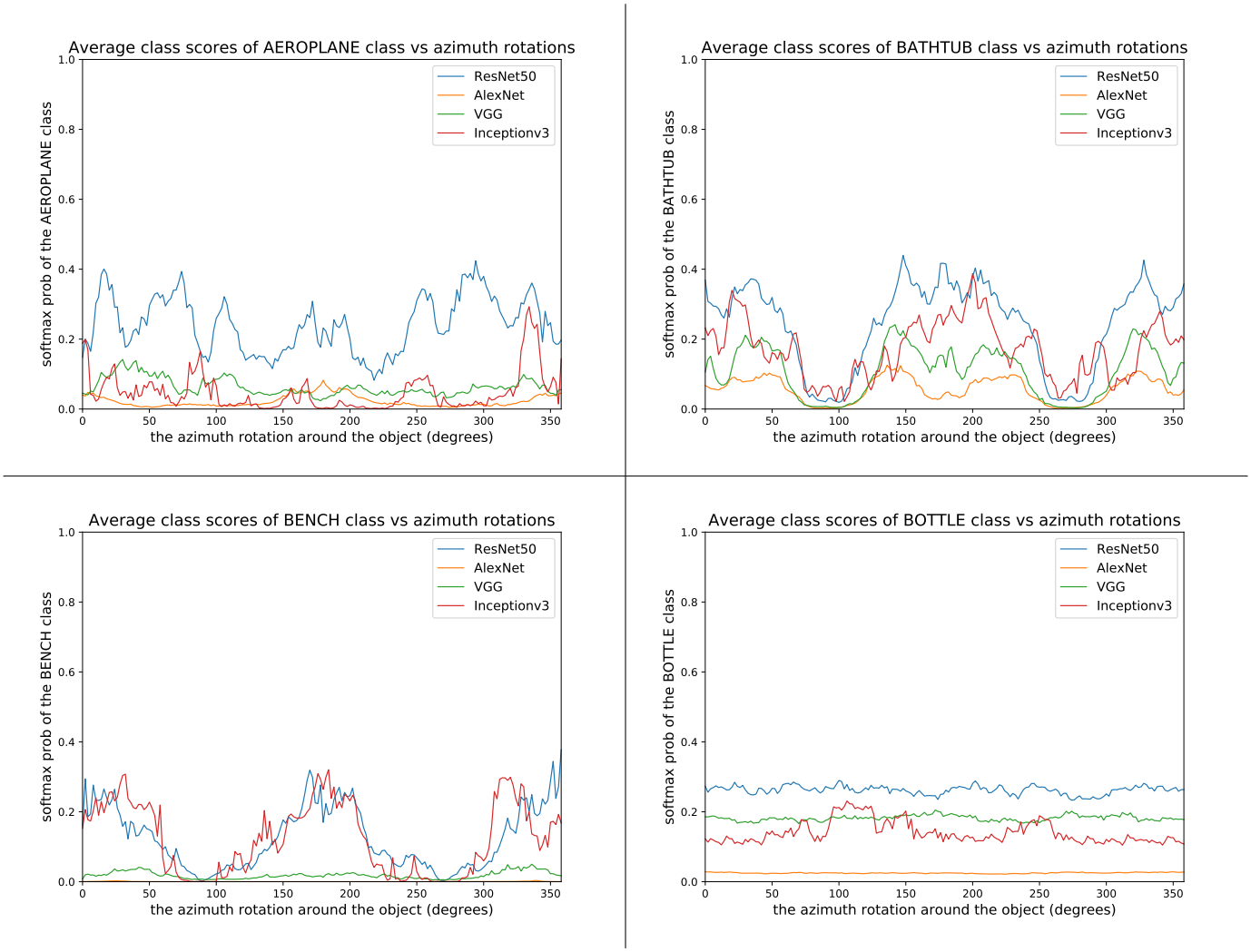

Figure 6: **1D Network Semantic Maps NMS-I**: visualizing 1D Semantic Robustness profile for different networks averaged over 10 different shapes. Observe that different DNNs profiles differ depending on the training , accuracy , and network architectures that all result in a unique "signatures" for the DNN on that class. The correlation between the DNN profiles is due to the common data bias in ImageNet.

Figure 7: **1D Network Semantic Maps NMS-II**: visualizing 1D Semantic Robustness profile for different networks averaged over 10 different shapes. Observe that different DNNs profiles differ depending on the training , accuracy , and network architectures that all result in a unique "signatures" for the DNN on that class. The correlation between the DNN profiles is due to the common data bias in ImageNet.

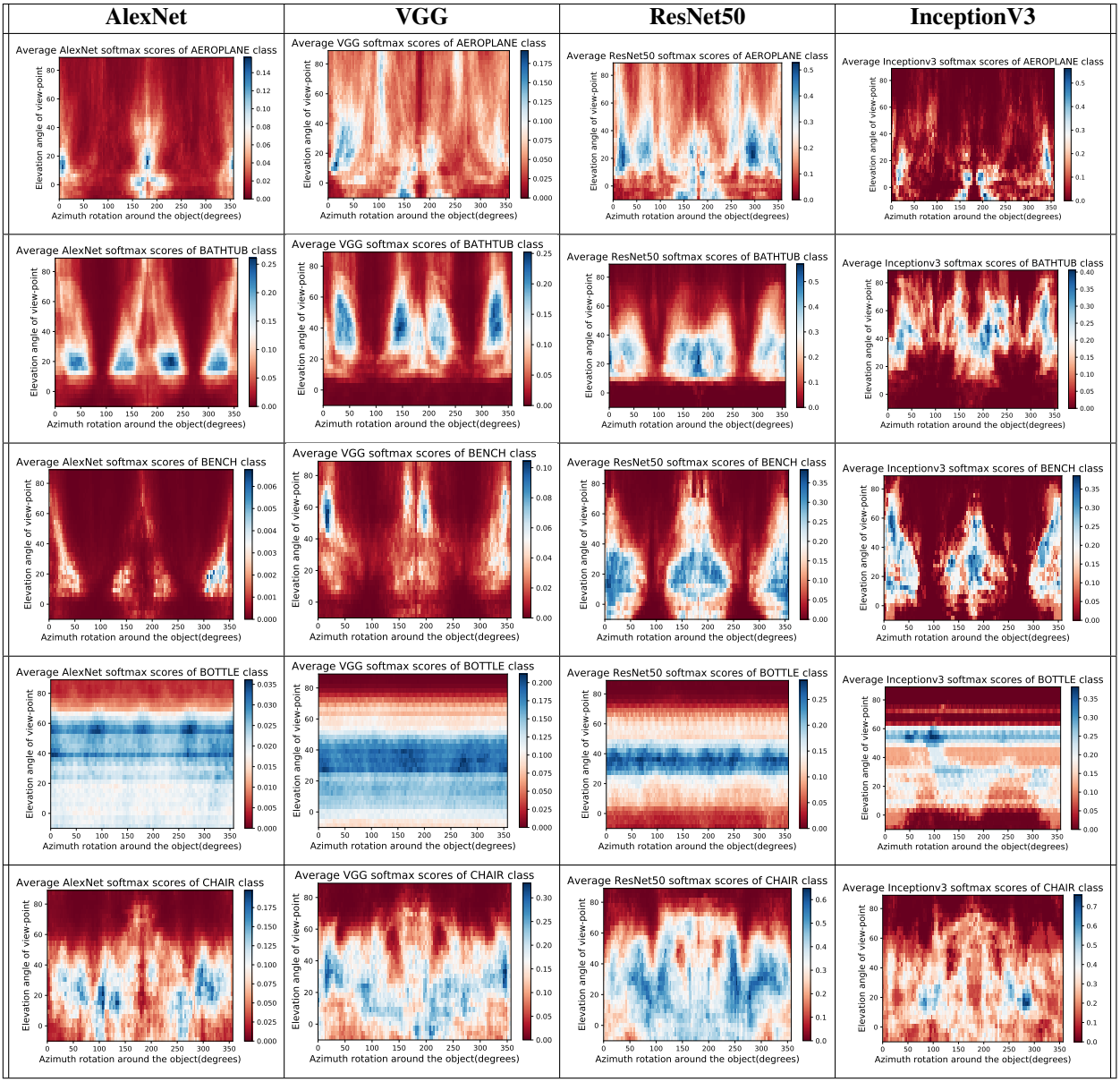

Figure 8: **2D Network Semantic Maps NMS-I**. Visualizing 2D Semantic Robustness profile for different networks averaged over 10 different shapes. Every row is different class. observe that different DNNs profiles differ depending on the training , accuracy , and network architectures that all result in a unique ”signatures” for the DNN on that class.

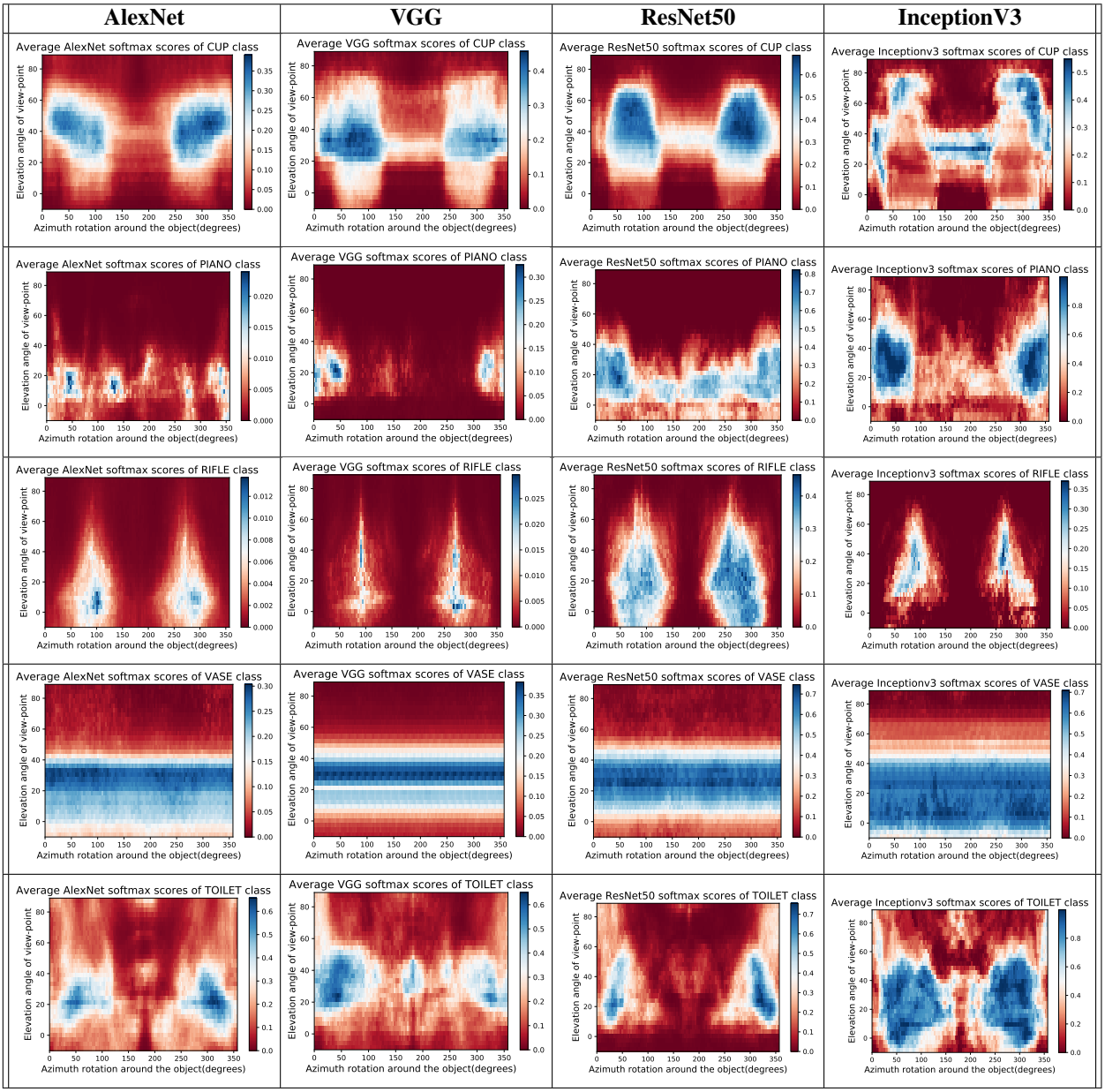

Figure 9: **2D Network Semantic Maps NMS-II**: visualizing 2D Semantic Robustness profile for different networks averaged over 10 different shapes. Every row is different class. observe that different DNNs profiles differ depending on the training , accuracy , and network architectures that all result in a unique ”signatures” for the DNN on that class.

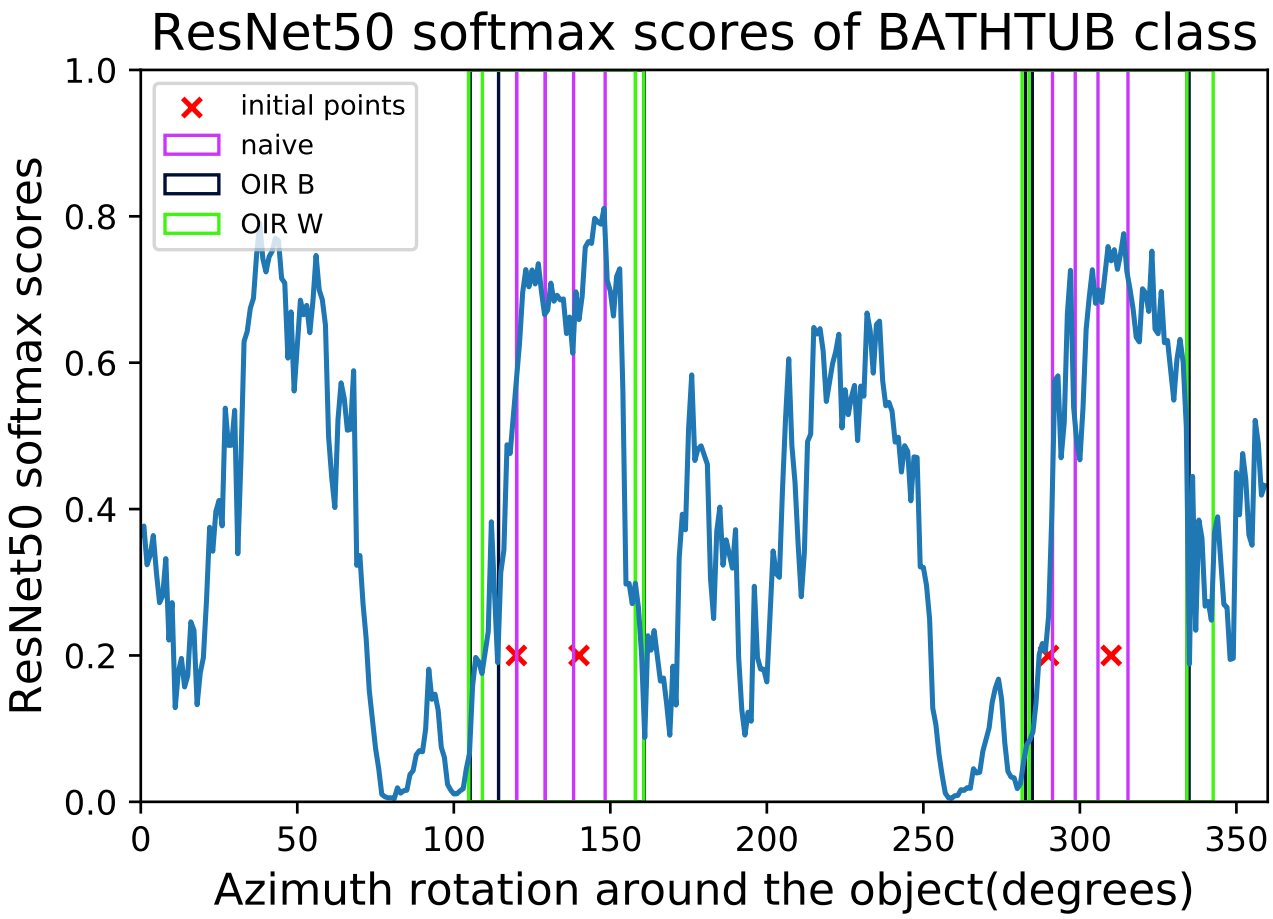

Figure 10: **Robust Region Detection with different initializations**: visualizing the Robust Regions Bounds found by the three algorithms for four initial points. We can see that the naive produce different bounds of the same region for different initializations while OIR detect the same region regardless of initialization.

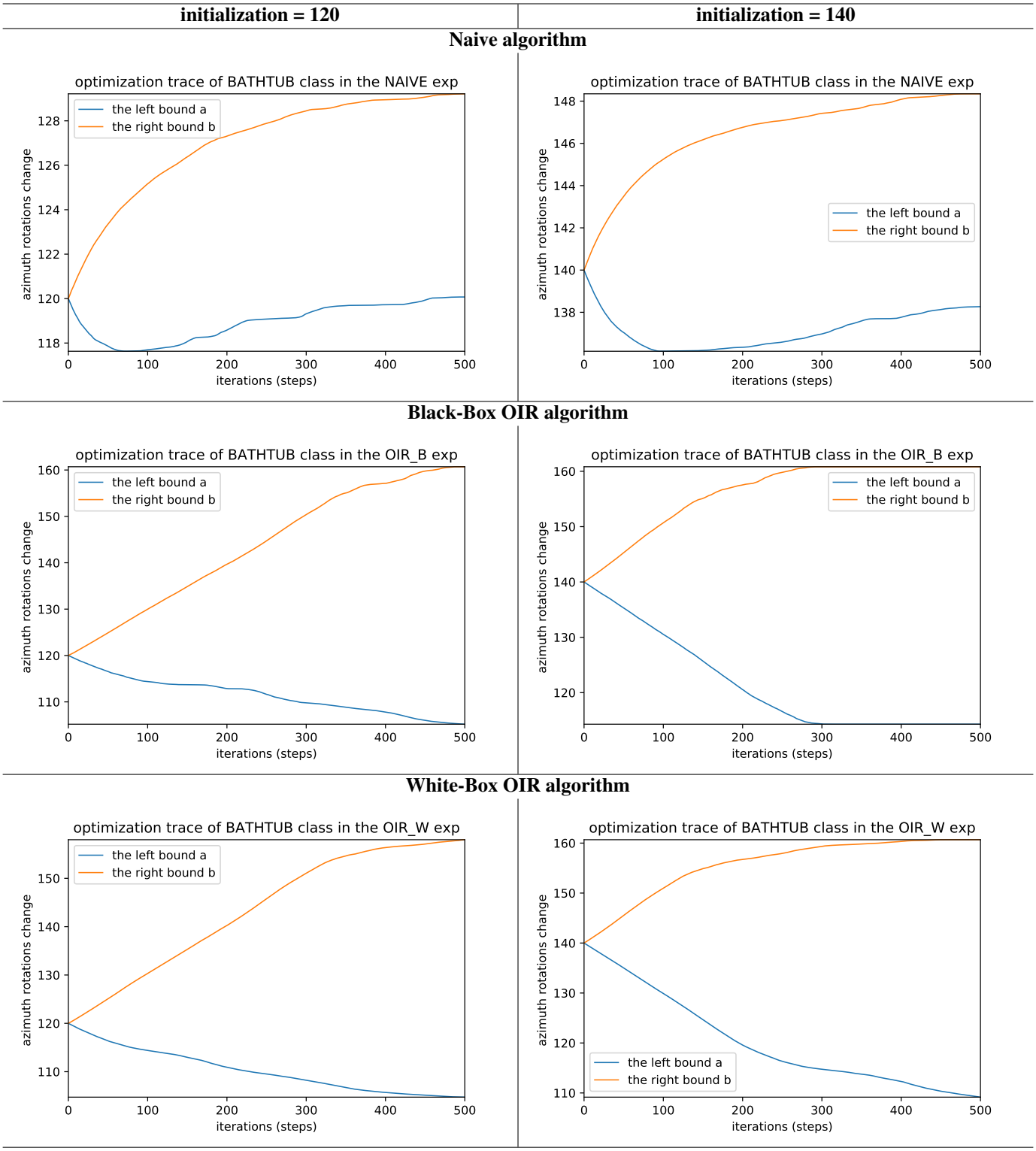

Figure 11: **Robust Region Bounds Growing I**: visualizing the bounds growing Using different algorithms from two different initializations (120 and 140) in Figure 10. We can see that OIR formulations converge to the same bounds of the robust region regardless of the initialization, which indicates effectiveness in detecting these regions, unlike the naive approach which can stop in a local optimum.

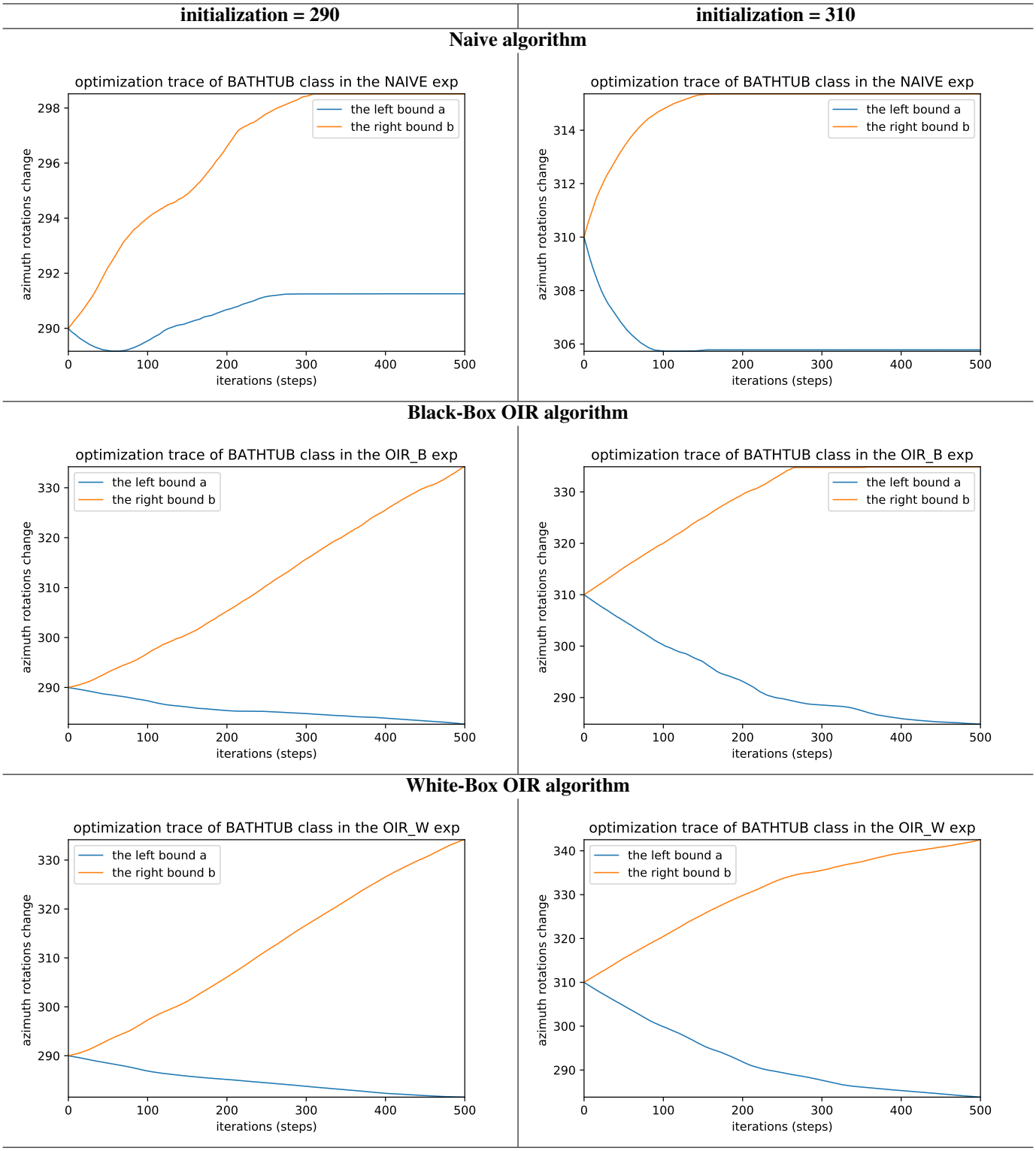

Figure 12: **Robust Region Bounds Growing I**: visualizing the bounds growing Using different algorithms from two different initializations (290, 310) in Figure 10. We can see that OIR formulations converge to the same bounds of the robust region regardless of the initialization, which indicates effectiveness in detecting these regions, unlike the naive approach which can stop in a local optimum.

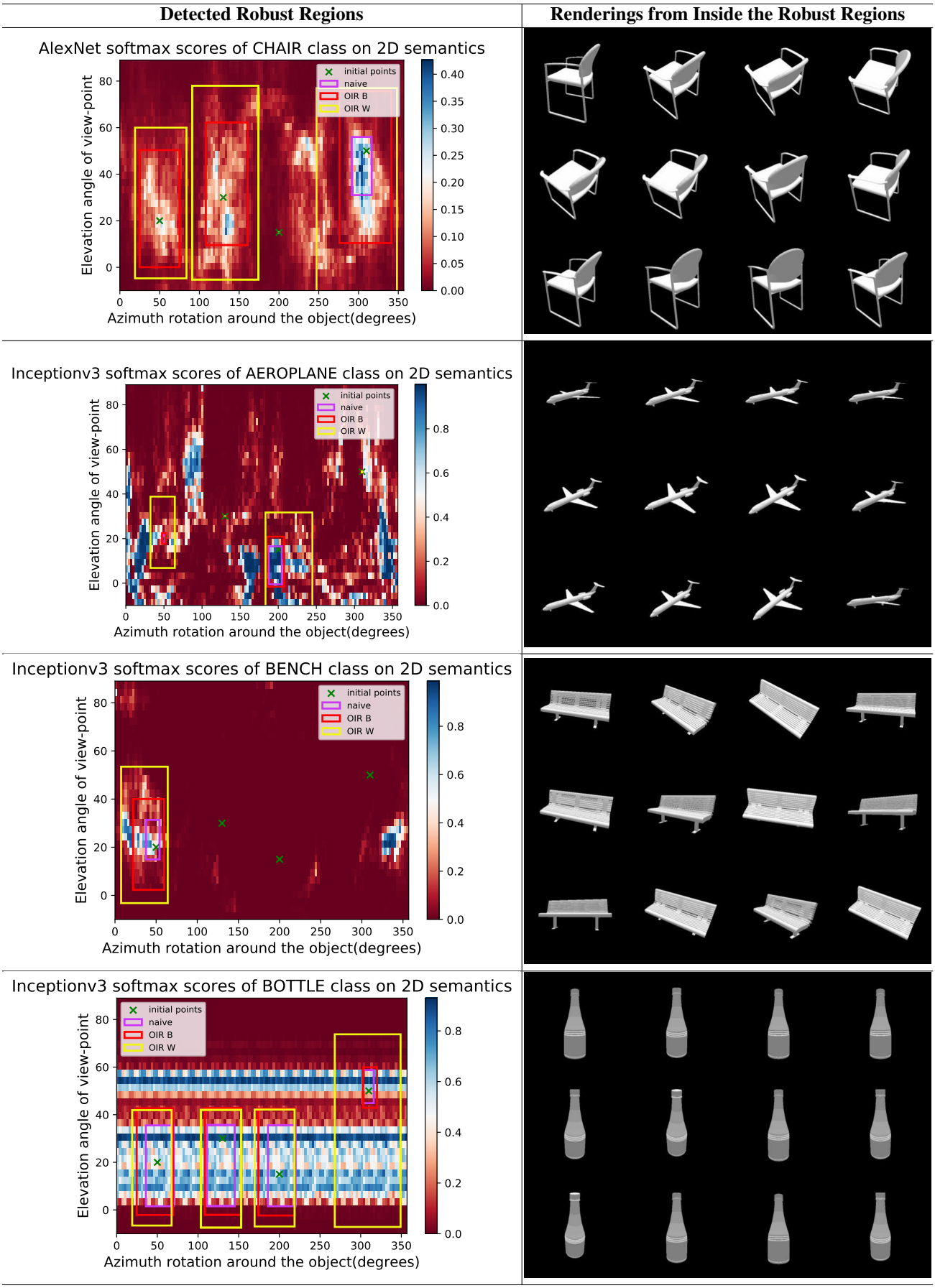

Figure 13: **Qualitative Examples of Robust Regions I**: visualizing different runs of the algorithm to find robust regions along with different renderings from inside these regions for those specific shapes used in the experiments.

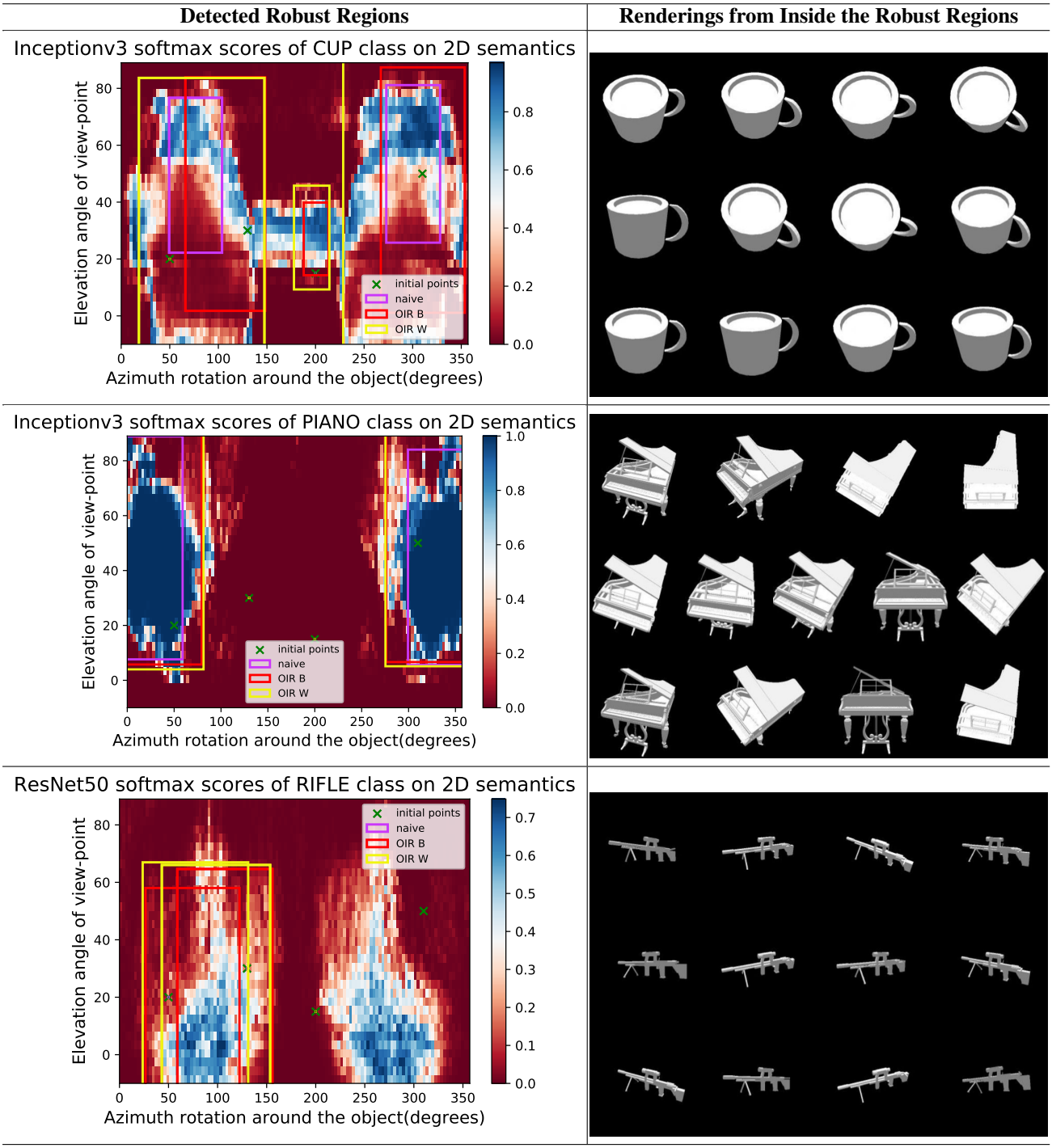

Figure 14: **Qualitative Examples of Robust Regions II**: visualizing different runs of the algorithm to find robust regions along with different renderings from inside these regions for those specific shapes used in the experiments.

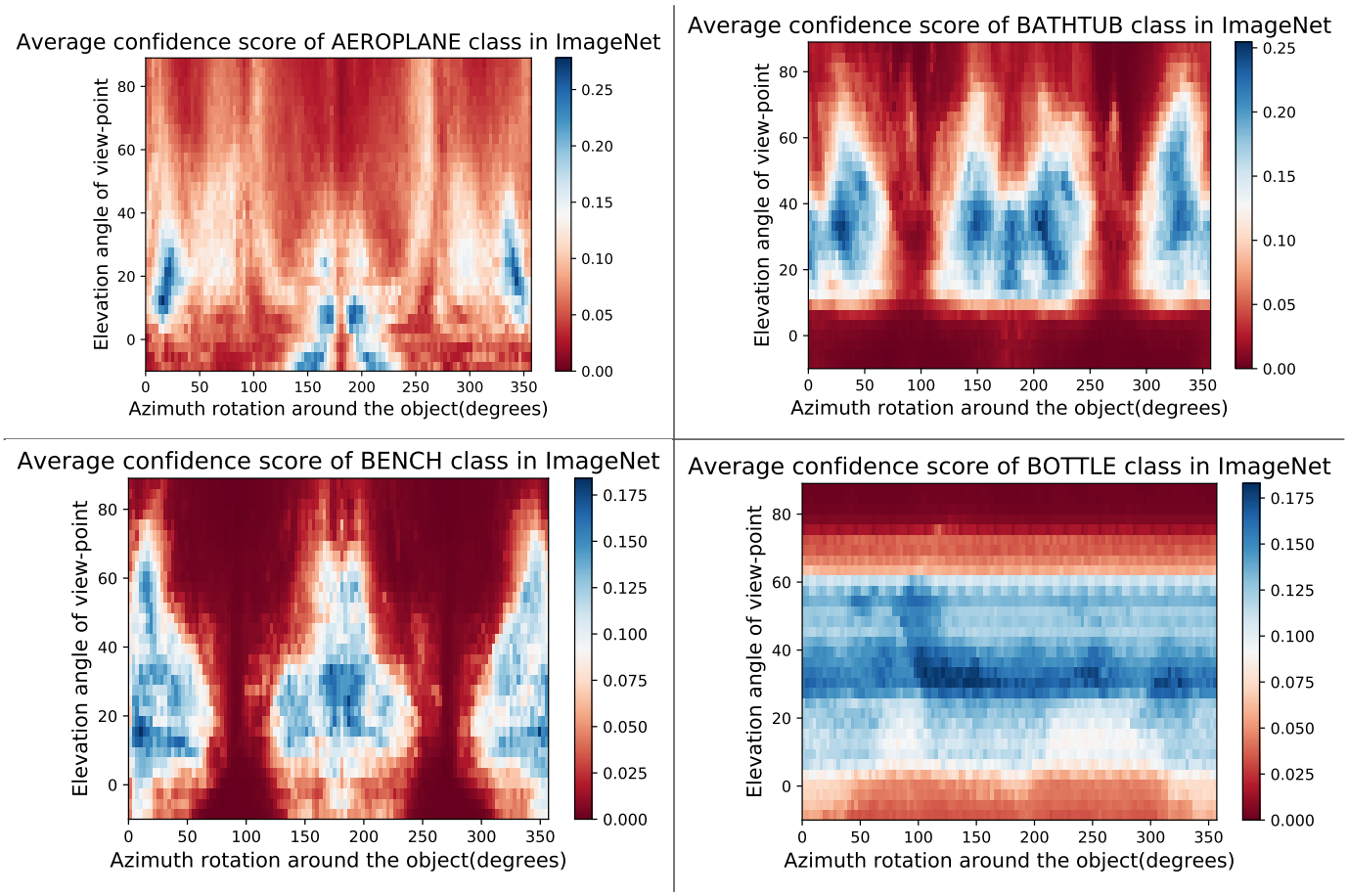

Figure 15: **Data Semantic Maps DSM-I**: visualizing Semantic Data Bias in the common training dataset (*i.e.* ImageNet [31]) by averaging the Networks Semantic Maps (NSM) of different networks and on different shapes, Different classes have different semantic bias in ImageNet as clearly shown in the maps above. The symmetry in the maps are attributed to the 3D symmetry of the objects.

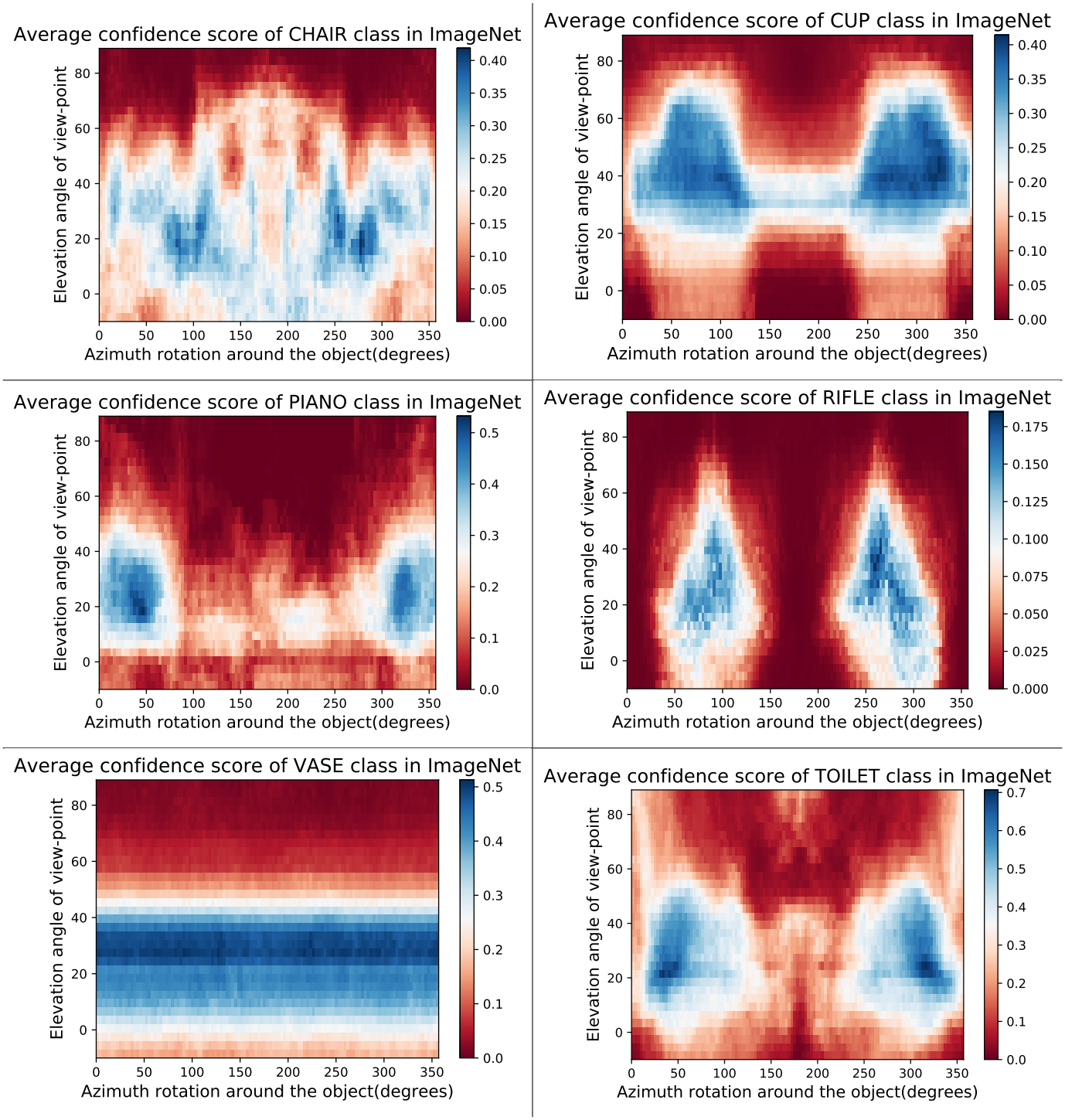

Figure 16: **Data Semantic Maps DSM-II**: visualizing Semantic Data Bias in the common training dataset (*i.e.* ImageNet [31]) By averaging the Networks Semantic Maps (NSM) of different networks and on different shapes, Different classes have different semantic bias in ImageNet as clearly shown in the maps above. The symmetry in the maps are attributed to the 3D symmetry of the objects.

B. Detailed Derivations of the Update Directions of the Bounds

In our case we consider a more general case where er are interested in the $\mathbf{u} \in \Omega \subset \mathbb{R}^n$, a hidden latent parameter that generate the image and is passes to scene generator (*e.g.* a renderer function $\mathbf{R}$) that takes the parameter $\mathbf{u}$ and a an object shape $\mathbf{S}$ of a class that is identified by classifier $\mathbf{C}$. Ω is the continuous semantic space for the parameters that we intend to study. The renderer creates the image $\mathbf{x} \in \mathbb{R}^d$, and then we study the behavior of a classifier $\mathbf{C}$ of that image across multiple shapes and multiple famous DNNs. Now, this function of interest is defined as follows.

$$f(\mathbf{u}) = \mathbf{C}_z(\mathbf{R}(\mathbf{S}_z, \mathbf{u}))\ ,\ \ 0 \leq f(\mathbf{u}) \leq 1 \tag{21}$$

Where z is a class label of interest of study, and we observe the network score for that class by rendering a shape $\mathbf{S}_z$ of the same class. The shape and class labels are constants, and only the parameters vary for f. The robust-region-finding operator is then defined as follows

$$\begin{aligned} &\mathbf{\Phi}_{\text{robust}}(f(\mathbf{u}), \mathbf{S}_z, \mathbf{u}_0) = \mathbb{D} = \{\mathbf{u} : \mathbf{a} \leq \mathbf{u} \leq \mathbf{b}\} \\ &\text{s.t. } \mathbb{E}_{\mathbf{u}\sim\mathbb{D}}[f(\mathbf{u})] \geq 1 - \epsilon_m\ ,\ \ \mathbf{u}_0 \in \mathbb{D}\ ,\ \text{VAR}[f(\mathbf{u})] \leq \epsilon_v \end{aligned} \tag{22}$$

where the left and right bounds of $\mathbb{D}$ are $\mathbf{a} = [a_1, a_2, ..., a_n]$ and $\mathbf{b} = [b_1, b_2, ..., b_n]$ respectively. The two samll thresholds ϵ_m, ϵ_v are to insure high performance and low variance of the DNN network in that robust region. We can define the opposite operator which is to find adversarial regions like follows :

$$\begin{aligned} &\mathbf{\Phi}_{\text{adv}}(f(\mathbf{u}), \mathbf{S}_z, \mathbf{u}_0) = \mathbb{D} = \{\mathbf{u} : \mathbf{a} \leq \mathbf{u} \leq \mathbf{b}\} \\ &\text{s.t. } \mathbb{E}_{\mathbf{u}\sim\mathbb{D}}[f(\mathbf{u})] \leq \epsilon_m\ ,\ \ \mathbf{u}_0 \in \mathbb{D}\ ,\ \text{VAR}[f(\mathbf{u})] \geq \epsilon_v \end{aligned} \tag{23}$$

We can show clearly that $\mathbf{\Phi}_{\text{adv}}$ and $\mathbf{\Phi}_{\text{robust}}$ are related as follows

$$\mathbf{\Phi}_{\text{adv}}(f(\mathbf{u}), \mathbf{S}_z, \mathbf{u}_0) = \mathbf{\Phi}_{\text{robust}}(1 - f(\mathbf{u}), \mathbf{S}_z, \mathbf{u}_0) \tag{24}$$

So we can just focus our attentions on $\mathbf{\Phi}_{\text{robust}}$ to find robust regions , and the adversarial regions follows directly from Eq (24).

B.1. Divergence of the Bounds

To develop an algorithm for $\mathbf{\Phi}$, we deploy the idea by [33] which focus on maximizing the inner area of the function in the region and fitting the bounds to grow the region bounds. As we will show , maximizing the region by maximizing the integral can lead to divergence , as follows :

Lemma B.1. *Let f be a continuous scalar function $f : \mathbb{R}^1 \to \mathbb{R}^1$, and let L be the function defining the definite integral of f in terms of the two integral bounds ,* i.e. $L(a, b) = \int_a^b f(u)du$. *Then, to maximize L, $-f(a)$ and $f(b)$ are valid ascent directions for the two bounds a, b respectively.*

Proof. a direction $\mathbf{p}$ is an ascent direction of objective L if it satisfies the inequality $\mathbf{p}^T \nabla L \geq 0$.[5].
To find $\frac{\partial l}{\partial a} = \frac{\partial}{\partial a}\int_a^b f(u)du$, we use Leibniz rule from the fundamental theorem of calculus which states that $\frac{d}{dx}\int_{a(x)}^{b(x)} f(u)du = f(b(x))\frac{d}{dx}b(x) - f(a(x))\frac{d}{dx}a(x)$
Therefore, $\frac{\partial}{\partial a}\int_a^b f(u)du = f(b) \times 0 - f(a) \times 1 = -f(a)$. Similarly, $\frac{\partial}{\partial b}\int_a^b f(u)du = f(b)$. By picking $\mathbf{p} = [-f(a), f(b)]^T$, then $\mathbf{p}^T\nabla L = f(a)^2 + f(b)^2 \geq 0$. This proves that $\mathbf{p}$ is a valid ascent direction for objective L. □

Theorem B.2. *Let f be a positive continuous scalar function $f : \mathbb{R}^1 \to (0, 1)$, and let L be the function defining the definite integral of f in terms of the two integral bounds ,* i.e. $L(a, b) = \int_a^b f(u)du$. *Then, following the ascent direction in Lemma B.1 can diverge the bounds if followed in a gradient ascent technique with fixed learning rate .*

Proof. If we follow the direction $= [-f(a), f(b)]^T$, with a fixed learning rate η, then the update rules for a, b will be as follws. $a_k = a_{k-1} - \eta f(a)$, $b_k = b_{k-1} + \eta f(b)$. for initial points a_0, b_0, then $a_k = a_0 - \eta\sum_{i=0}^k f(\text{Area}_{\text{in}})$, and $b_k = b_0 + \eta\sum_{i=0}^k f(b_i)$. We can see now if $f(u) = c, 0 < c < 1$, then as $k \to \infty$, the bounds $a_k \to -\infty$, $b_k \to \infty$. This leads to the claimed divergence. □

To solve the issue of bounds diverging we propose the following formulations for one dimensional bounds , and then we extend them to n- dimensions , which allows for finding the n-dimensional semantic robust/adversarial regions that are similar to figure 2 . some of the formulations are black-box in nature (they dint need the gradient of the function f in order to update the current estimates of the bound) while others .

B.2. Naive Approach

$$L = -\text{Area}_{\text{in}} + \frac{\lambda}{2}|b - a|_2^2 \tag{25}$$

using Libeniz rule as in Lemma B.1, we get the following update steps for the objective L :

$$\begin{aligned} \frac{\partial L}{\partial a} &= f(a) - \lambda(b - a) \\ \frac{\partial L}{\partial b} &= -f(b) + \lambda(b - a) \end{aligned} \tag{26}$$

where λ is regularizing the boundaries not to extend too much in the case the function evaluation wa positive all the time.

Extension to n-dimension:. Lets start by $n = 2$. Now, we have $f : \mathbb{R}^2 \to (0, 1)$, then we define the loss integral as a

function of four bounds of a rectangular region as follows.

$$L(a_1,a_2,b_1,b_2) = \\ -\int_{a_2}^{b_2}\int_{a_1}^{b_1} f(u,v)dvdu + \frac{\lambda}{2}|b_1-a_1|_2^2 + \frac{\lambda}{2}|b_2-a_2|_2^2 \tag{27}$$

We apply the trapezoidal approximation on the loss to obtain the following expression.

$$L(a_1,a_2,b_1,b_2) \approx \frac{\lambda}{2}|b_1-a_1|_2^2 + \frac{\lambda}{2}|b_2-a_2|_2^2 \\ -\frac{(b_1-a_1)(b_2-a_2)}{4}(\, f(a_1,a_2)+f(b_1,a_2)+ \\ f(a_1,b_2)+f(b_1,b_2)\,) \tag{28}$$

to find the update direction for the first bound a_1 by taking the partial derivative of the function in Eq (27) we get the following update direction for a_1 along with its trapezoidal approximation in order to able to compute it during the optimization:

$$\frac{\partial L}{\partial a_1} = \int_{a_2}^{b_2} f(a_1,v)dv \;-\; \lambda(b_1-a_1) \\ \approx \frac{(b_2-a_2)}{2}\Big(\, f(a_1,a_2)+f(a_1,b_2))\,\Big) \;-\; \lambda(b_1-a_1) \tag{29}$$

Doing similar steps to the first bound for the other three bounds we obtain the full update directions for (a_1,a_2,b_1,b_2)

$$\frac{\partial L}{\partial a_1} \approx \frac{(b_2-a_2)}{2}\Big(\, f(a_1,a_2)+f(a_1,b_2))\,\Big) \;-\; \lambda(b_1-a_1) \\ \frac{\partial L}{\partial b_1} \approx -\frac{(b_2-a_2)}{2}\Big(\, f(b_1,a_2)+f(b_1,b_2))\,\Big) \;+\; \lambda(b_1-a_1) \\ \frac{\partial L}{\partial a_2} \approx \frac{(b_1-a_1)}{2}\Big(\, f(a_1,a_2)+f(b_1,a_2))\,\Big) \;-\; \lambda(b_2-a_2) \\ \frac{\partial L}{\partial b_2} \approx -\frac{(b_1-a_1)}{2}\Big(\, f(a_1,b_2)+f(b_1,b_2))\,\Big) \;+\; \lambda(b_2-a_2) \tag{30}$$

Now, for $f:\mathbb{R}^n \to (0,1)$, we define the inner region hyper-rectangle as before $\mathbb{D} = \{\mathbf{x} : \mathbf{a} \le \mathbf{x} \le \mathbf{b}\}$.Here , we assume the size of the region is positive at every dimension , *i.e.* $\mathbf{r} = \mathbf{b}-\mathbf{a} > \mathbf{0}$. The volume of the region $\mathbb{D}$ normalized by exponent of dimension n is expressed as follows

$$\text{volume}(\mathbb{D}) = \triangle = \frac{1}{2^n}\prod_{i=1}^{n}\mathbf{r}_i \tag{31}$$

The region $\mathbb{D}$ can also be defined in terms of the matrix $\mathbf{D}$ of all the corner points $\{\mathbf{d}^i\}_{i=1}^{2^n}$ as follows.

$$\text{corners}(\mathbb{D}) = \mathbf{D}_{n\times 2^n} = \Big[\mathbf{d}^1|\mathbf{d}^2|..|\mathbf{d}^{2^n}\Big] \\ \mathbf{D} = \mathbf{1}^T\mathbf{a} \;+\; \mathbf{M}^T \odot (\mathbf{1}^T\mathbf{r}) \tag{32}$$

where $\mathbf{1}$ is the all-ones vector of size 2^n, $\odot$ is the Hadamard product of matrices (element-wise) , and $\mathbf{M}$ is a constant masking matrix defined as the matrix of binary numbers of n bits that range from 0 to $2n-1$ defined as follows.

$$\mathbf{M}_{n\times 2^n} = \Big[\mathbf{m}^0|\mathbf{m}^1|..|\mathbf{m}^{2^n-1}\Big] \;,\; \text{where } \mathbf{m}^i = \text{binary}_n(i) \tag{33}$$

We define the function vector as the vector $\mathbf{f}_{\mathbb{D}}$ of all function evaluations at all corner points of $\mathbb{D}$

$$\mathbf{f}_{\mathbb{D}} = \Big[f(\mathbf{d}^1), f(\mathbf{d}^2), ..., f(\mathbf{d}^{2^n})\Big]^T,\; \mathbf{d}^{\mathbf{i}} = \mathbf{D}_{:,i} \tag{34}$$

We follow similar steps as in $n=2$ and obtain the following loss expressions and update directions :

$$L(\mathbf{a},\mathbf{b}) = -\int\cdots\int_{\mathbb{D}} f(u_1,\ldots,u_n)\,du_1\ldots du_n + \frac{\lambda}{2}|\mathbf{r}|^2 \\ \approx \; -\triangle\mathbf{1}^T\mathbf{f}_{\mathbb{D}} \;+\; \frac{\lambda}{2}|\mathbf{r}|^2 \\ \nabla_{\mathbf{a}}L \approx \; 2\triangle\text{diag}^{-1}(\mathbf{r})\overline{\mathbf{M}}\mathbf{f}_{\mathbb{D}} + \lambda\mathbf{r} \\ \nabla_{\mathbf{b}}L \approx \; -2\triangle\text{diag}^{-1}(\mathbf{r})\mathbf{M}\mathbf{f}_{\mathbb{D}} - \lambda\mathbf{r} \tag{35}$$

B.3. Outer-Inner Ratio Loss (OIR)

We introduce an outer region A,B with bigger area that contains the small region (a,b). We follow the following assumption to insure that outer area is always positive.

$$A = a - \alpha\frac{b-a}{2}, B = b + \alpha\frac{b-a}{2} \tag{36}$$

where α is the small boundary factor of the outer area to inner area. We formulate the problem as a ratio of outer over inner area and we try to make this ratio as close as possible to 0 . $L = \frac{\text{Area}_{\text{out}}}{\text{Area}_{\text{in}}}$ By using DencklBeck technique for solving non-linear fractional programming problems [30]. Using their formulation to transform L as follows.

$$L = \frac{\text{Area}_{\text{out}}}{\text{Area}_{\text{in}}} \;=\; \text{Area}_{\text{out}} \;-\; \lambda\,\text{Area}_{\text{in}} \\ = \int_A^B f(a)du \;-\; \int_a^b f(a)du \;-\; \lambda\int_a^b f(a)du \tag{37}$$

where $\lambda^* = \frac{\text{Area}^*_{\text{out}}}{\text{Area}^*_{\text{in}}}$ is the DencklBeck factor and it is equal to the small objective best achieved.

Black-Box (OIR_B).

Here we set $\lambda = 1$ to simplify the problem. This yields

the following expression of the loss

$$\begin{aligned} L &= \text{Area}_{\text{out}} - \text{Area}_{\text{in}} \\ &= \int_A^a f(u)du + \int_b^B f(u)du - \int_a^b f(u)du \\ &= \int_A^B f(u)du - 2\int_a^b f(u)du \\ &= \int_{a-\alpha\frac{b-a}{2}}^{b+\alpha\frac{b-a}{2}} f(u)du - 2\int_a^b f(u)du \end{aligned} \tag{38}$$

using Libeniz ruloe as in Lemma B.1, we get the following update steps for the objective L :

$$\begin{aligned} \frac{\partial L}{\partial a} &= -(1+\frac{\alpha}{2})f(A) - \frac{\alpha}{2}f(B) + 2f(a) \\ \frac{\partial L}{\partial b} &= (1+\frac{\alpha}{2})f(B) + \frac{\alpha}{2}f(A) - 2f(b) \end{aligned} \tag{39}$$

Extension to n-dimension:. Lets start by $n = 2$. Now, we have $f : \mathbb{R}^2 \to (0, 1)$, and with the following constrains on the outer region.

$$\begin{aligned} A_1 &= a_1 - \frac{b_1 - a_1}{2}, \quad B_1 = b_1 + \frac{b_1 - a_1}{2} \\ A_2 &= a_2 - \frac{b_2 - a_2}{2}, \quad B_1 = b_2 + \frac{b_2 - a_2}{2} \end{aligned} \tag{40}$$

we define the loss integral as a function of four bounds of a rectangular region as follows.

$$\begin{aligned} &L(a_1, a_2, b_1, b_2) = \\ &\int_{A_2}^{B_2}\int_{A_1}^{B_1} f(u, v)dvdu \; - \; 2\int_{a_2}^{b_2}\int_{a_1}^{b_1} f(u, v)dvdu \end{aligned} \tag{41}$$

We apply the trapezoidal approximation on the loss to obtain the following expression.

$$\begin{aligned} &L(a_1, a_2, b_1, b_2) \approx \\ &\frac{(B_1 - A_1)(B_2 - A_2)}{4}\big(f(A_1, A_2) + f(B_1, A_2) + \\ &\qquad f(A_1, B_2) + f(B_1, B_2) \big) \\ &- \frac{(b_1 - a_1)(b_2 - a_2)}{2}\big(f(a_1, a_2) + f(b_1, a_2) + \\ &\qquad f(a_1, b_2) + f(b_1, b_2) \big) \\ &= \frac{(b_1 - a_1)(b_2 - a_2)}{4}\Big(\\ &(1+\alpha)^2\big(f(A_1, A_2) + f(B_1, A_2) + \\ &\qquad f(A_1, B_2) + f(B_1, B_2) \big) \\ &- 2\big(f(a_1, a_2) + f(b_1, a_2) + f(a_1, b_2) + f(b_1, b_2) \big) \Big) \end{aligned} \tag{42}$$

to find the update direction for the first bound a_1 by taking the partial derivative of the function in Eq (41) we get the following update direction for a_1 along with its trapezoidal approximation in order to able to compute it during the optimization:

$$\begin{aligned} \frac{\partial L}{\partial a_1} &= -(1+\frac{\alpha}{2})\int_{A_2}^{B_2}\Big(f(A_1, v) + \frac{\alpha}{2}f(B_1, v)\Big)dv \\ &+ 2\int_{a_2}^{b_2} f(a_1, v)dv \\ &\approx \frac{(b_2 - a_2)}{2}\Big(\\ &- (1+\alpha)\big((1+\frac{\alpha}{2})(f(A_1, A_2) + f(A_1, B_2)) \\ &\qquad + \frac{\alpha}{2}\;(f(B_1, A_2) + f(B_1, B_2)) \big) \\ &+ 2\Big(f(a_1, a_2) + f(a_1, b_2)) \Big) \Big) \end{aligned} \tag{43}$$

Doing similar steps to the first bound for the other three bounds we obtain the full update directions for (a_1, a_2, b_1, b_2)

$$\begin{aligned} \frac{\partial L}{\partial a_1} &\approx \frac{(b_2 - a_2)}{2}\Big(\\ &- (1+\alpha)\big((1+\frac{\alpha}{2})(f(A_1, A_2) + f(A_1, B_2)) \\ &\qquad + \frac{\alpha}{2}\;(f(B_1, A_2) + f(B_1, B_2)) \big) \\ &+ 2\Big(f(a_1, a_2) + f(a_1, b_2)) \Big) \Big) \end{aligned} \tag{44}$$

$$\begin{aligned} \frac{\partial L}{\partial b_1} &\approx \frac{(b_2 - a_2)}{2}\Big(\\ &(1+\alpha)\big((1+\frac{\alpha}{2})(f(B_1, A_2) + f(B_1, B_2)) \big) \\ &\qquad + \frac{\alpha}{2}\;(f(A_1, A_2) + f(A_1, B_2)) \\ &- 2\Big(f(b_1, a_2) + f(b_1, b_2)) \Big) \Big) \end{aligned} \tag{45}$$

$$\begin{aligned} \frac{\partial L}{\partial a_2} &\approx \frac{(b_1 - a_1)}{2}\Big(\\ &- (1+\alpha)\big((1+\frac{\alpha}{2})(f(A_1, A_2) + f(A_1, B_2)) \\ &\qquad + \frac{\alpha}{2}\;(f(A_1, B_2) + f(B_1, B_2)) \big) \\ &+ 2\Big(f(a_1, a_2) + f(b_1, a_2)) \Big) \Big) \end{aligned} \tag{46}$$

$$\begin{aligned} \frac{\partial L}{\partial b_2} &\approx \frac{(b_1 - a_1)}{2}\Big(\\ &(1+\alpha)\big((1+\frac{\alpha}{2})(f(A_1, B_2) + f(B_1, B_2)) \\ &\qquad + \frac{\alpha}{2}\;(f(A_1, A_2) + f(B_1, A_2)) \big) \\ &- 2\Big(f(a_1, b_2) + f(b_1, b_2)) \Big) \Big) \end{aligned} \tag{47}$$

Now, for $f : \mathbb{R}^n \to (0, 1)$, we define the inner region hyper-rectangle as before $\mathbb{D} = \{\mathbf{x} : \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}$, but now define

an outer bigger region $\mathbb{Q}$ that include the smaller region $\mathbb{D}$ and defined as follows : $\mathbb{Q} = \{\mathbf{x} : \mathbf{a} - \frac{\alpha}{2}\mathbf{r} \leq \mathbf{x} \leq \mathbf{b} + \frac{\alpha}{2}\mathbf{r}\}$, where $\mathbf{a}, \mathbf{b}, \mathbf{r}$ are defined as before, while α is defined as the boundary factor of the outer region in all the dimensions equivilantly. The inner and outer regions can also be defined in terms of the corner points as follows.

$$\begin{aligned}
\text{corners}(\mathbb{D}) = \mathbf{D}_{n\times 2^n} &= \Big[\mathbf{d}^1|\mathbf{d}^2|..|\mathbf{d}^{2^n}\Big] \\
\mathbf{D} &= \mathbf{1}^T\mathbf{a} \ + \ \mathbf{M}^T \odot (\mathbf{1}^T\mathbf{r}) \\
\text{corners}(\mathbb{Q}) = \mathbf{Q}_{n\times 2^n} &= \Big[\mathbf{q}^1|\mathbf{q}^2|..|\mathbf{q}^{2^n}\Big] \\
\mathbf{Q} &= \mathbf{1}^T(\mathbf{a} - \frac{\alpha}{2}\mathbf{r}) \ + \ (1+\alpha)\mathbf{M}^T \odot (\mathbf{1}^T\mathbf{r})
\end{aligned} \tag{48}$$

where $\mathbf{1}$ is the all-ones vector of size 2^n, $\odot$ is the Hadamard product of matrices (elemnt-wise) , and $\mathbf{M}$ is a constant masking matrix defined in Eq (33). Now we define two function vectors evaluated at all possible inner and outer corner points respectively.

$$\begin{aligned}
\mathbf{f}_{\mathbb{D}} &= \Big[f(\mathbf{d}^1), f(\mathbf{d}^2), ..., f(\mathbf{d}^{2^n})\Big]^T, \ \mathbf{d^i} = \mathbf{D}_{:,i} \\
\mathbf{f}_{\mathbb{Q}} &= \Big[f(\mathbf{q}^1), f(\mathbf{q}^2), ..., f(\mathbf{q}^{2^n})\Big]^T, \ \mathbf{c^i} = \mathbf{Q}_{:,i}
\end{aligned} \tag{49}$$

Now the loss and update directions for the n-dimensional case becomes as follows .

$$\begin{aligned}
L(\mathbf{a},\mathbf{b}) &= \int \cdots \int_{\mathbb{Q}} f(u_1,\ldots,u_n)\, du_1 \ldots du_n \\
&\quad - 2\int \cdots \int_{\mathbb{D}} f(u_1,\ldots,u_n)\, du_1 \ldots du_n \\
&\approx \triangle\Big((1+\alpha)^n \mathbf{1}^T\mathbf{f}_{\mathbb{Q}} \ - \ 2\,\mathbf{1}^T\mathbf{f}_{\mathbb{D}}\Big) \\
\nabla_{\mathbf{a}} L \approx & 2\triangle \text{diag}^{-1}(\mathbf{r})\Big(2\overline{\mathbf{M}}\mathbf{f}_{\mathbb{D}} \ - \ \overline{\mathbf{M}}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\Big) \\
\nabla_{\mathbf{b}} L \approx & 2\triangle \text{diag}^{-1}(\mathbf{r})\Big(-2\mathbf{M}\mathbf{f}_{\mathbb{D}} \ + \ \mathbf{M}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\Big)
\end{aligned} \tag{50}$$

wheere $\overline{\mathbf{M}}_{\mathbb{Q}}$ is the outer region mask defined as follows.

$$\begin{aligned}
\overline{\mathbf{M}}_{\mathbb{Q}} &= (1+\alpha)^{n-1}\Big((1+\frac{\alpha}{2})\overline{\mathbf{M}} + \frac{\alpha}{2}\mathbf{M}\Big) \\
\mathbf{M}_{\mathbb{Q}} &= (1+\alpha)^{n-1}\Big((1+\frac{\alpha}{2})\mathbf{M} + \frac{\alpha}{2}\overline{\mathbf{M}}\Big)
\end{aligned} \tag{51}$$

White-Box OIR (OIR_W.) The following formulation is white-box in nature (it need the gradient of the function f in order to update the current estimates of the bound) this is useful when the function in hand is differntiable (*e.g.* DNN) , to obtain more intelligent regions, rather then the regions surrounded by near 0 values of the function f. We set $\lambda = \frac{\alpha}{\beta}$ in Eq (37), where α is the small boundary factor of the outer area, β is the emphasis factor (we will show later how it determines the emphasis on the function vs the gradient). Hence, the objective in Eq (37) becomes :

$$\begin{aligned}
&\arg\min_{a,b} L = \arg\min_{a,b} \ \text{Area}_{\text{out}} - \lambda \ \text{Area}_{\text{in}} \\
&= \arg\min_{a,b} \int_A^a f(u)du + \int_b^B f(u)du - \frac{\alpha}{\beta}\int_a^b f(u)du \\
&= \arg\min_{a,b} \ \frac{\beta}{\alpha}\int_{a-\alpha\frac{b-a}{2}}^{b+\alpha\frac{b-a}{2}} f(u)du - (1+\frac{\beta}{\alpha})\int_a^b f(u)du
\end{aligned} \tag{52}$$

using Libeniz ruloe as in Lemma B.1, we get the following derivatives of the bound a :

$$\begin{aligned}
\frac{\partial L}{\partial a} &= \frac{\beta}{\alpha}\Big(f(a) - f\Big(a - \alpha\frac{b-a}{2}\Big)\Big) \\
&\quad - \frac{\beta}{2}f\Big(b + \alpha\frac{b-a}{2}\Big) \ - \frac{\beta}{2}f\Big(a - \alpha\frac{b-a}{2}\Big) + f(a)
\end{aligned} \tag{53}$$

now since λ^* should be small for the optimal objective , then as $\lambda \to 0$, $\alpha \to 0$ and hence the derivative in Eq (53) becomes the following.

$$\begin{aligned}
\lim_{\alpha\to 0}\frac{\partial L}{\partial a} &= \lim_{\alpha\to 0}\beta\frac{\Big(f(a) - f\Big(a - \alpha\frac{b-a}{2}\Big)\Big)}{\alpha} \\
&\quad - \frac{\beta}{2}f(b) \ - \frac{\beta}{2}f(a) + f(a)
\end{aligned} \tag{54}$$

We can see that the first term is proportional to the derivative of f at a, and hence the expression becomes :

$$\lim_{\alpha\to 0}\frac{\partial L}{\partial a} = \frac{\beta}{2}\Big((b-a)f'(a) + f(b)\Big) \ + \ (1 - \frac{\beta}{2})f(a) \tag{55}$$

we can see that the update rule depends on the function value and the derivative of f at the boundary a with β controlling the dependence. Similarly for the boundary b we can see the following direction

$$\lim_{\alpha\to 0}\frac{\partial L}{\partial b} = \frac{\beta}{2}\Big((b-a)f'(b) + f(a)\Big) \ - \ (1 - \frac{\beta}{2})f(b) \tag{56}$$

If $\beta \to 0$, the update directions in Eq (55,56) collapse to the unregularized naive update in Eq (26) .

Extension to n-dimension.. Lets start by $n = 2$. Now, we have $f : \mathbb{R}^2 \to (0, 1)$, and with the following constrains on the outer region.

$$\begin{aligned}
A_1 &= a_1 - \frac{b_1 - a_1}{2}, \ \ B_1 = b_1 + \frac{b_1 - a_1}{2} \\
A_2 &= a_2 - \frac{b_2 - a_2}{2}, \ \ B_1 = b_2 + \frac{b_2 - a_2}{2}
\end{aligned} \tag{57}$$

we follow similar approach as in Eq (52) to obtain the fol-

lowing expression.

$$L(a_1, a_2, b_1, b_2) = \frac{\beta}{\alpha}\int_{A_2}^{B_2}\int_{A_1}^{B_1} f(u,v)dvdu \; - \; (1+\frac{\beta}{\alpha})\int_{a_2}^{b_2}\int_{a_1}^{b_1} f(u,v)dvdu \tag{58}$$

We apply the trapezoidal approximation on the loss to obtain the following approximation.

$$L(a_1, a_2, b_1, b_2) \approx -1 + (1+\alpha)^2 \frac{(\; f(A_1, A_2) + f(B_1, A_2) + f(A_1, B_2) + f(B_1, B_2)\;)}{(\; f(a_1, a_2) + f(b_1, a_2) + f(a_1, b_2) + f(b_1, b_2)\;)} \tag{59}$$

to find the update direction for the first bound a_1 by taking the partial derivative of the function in Eq (41) we get the fol owing update direction for a_1 along with its trapezoidal approximation in order to able to compute it during the optimization:

$$\begin{aligned}\frac{\partial L}{\partial a_1} &= -\frac{\beta}{\alpha}(1+\frac{\alpha}{2})\int_{A_2}^{B_2}\Big(f(A_1,v) + \frac{\alpha}{2}f(B_1,v)\Big)dv \\ &\quad + \; (1+\frac{\beta}{\alpha})\int_{a_2}^{b_2} f(a_1,v)dv \\ &\approx \frac{(b_2-a_2)}{2}(\\ &\quad -(1+\alpha)(\;(\frac{\beta}{\alpha}+\frac{\beta}{2})(f(A_1,A_2)+f(A_1,B_2)) \\ &\qquad\qquad + \frac{\beta}{2}\;(f(B_1,A_2)+f(B_1,B_2))\;) \\ &\quad + (1+\frac{\beta}{\alpha})\Big(\; f(a_1,a_2)+f(a_1,b_2))\;\Big)\end{aligned} \tag{60}$$

grouping the terms which are divided by α together and then taking the limit of $\alpha \to \infty$ (as explained in the 1-d case), we get the following expressions.

$$\begin{aligned}&\lim_{\alpha\to 0}\frac{\partial L}{\partial a_1} \approx \frac{(b_2-a_2)}{2}(\\ &\lim_{\alpha\to 0}\frac{\beta}{\alpha}(\;(\;f(a_1,a_2)+f(a_1,b_2)) - (f(A_1,A_2)+f(A_1,B_2))\;) \\ &\quad - \lim_{\alpha\to 0}\frac{3\beta}{2}\;(f(A_1,A_2)+f(A_1,B_2)\;) \\ &\quad - \lim_{\alpha\to 0}\frac{\beta}{2}(\;f(B_1,A_2)+f(B_1,B_2)\;) \\ &\quad + \Big(\;f(a_1,a_2)+f(a_1,b_2)\;)\;\Big)\end{aligned} \tag{61}$$

Noting that the first term is related to the directional derivatives of f, we get the following limit expression

$$\begin{aligned}&\lim_{\alpha\to 0}\frac{\partial L}{\partial a_1} \approx \frac{(b_2-a_2)}{2}(\\ &\beta(\;\nabla f(a_1,a_2)\cdot\begin{pmatrix}\frac{b_1-a_1}{2}\\ \frac{b_2-a_2}{2}\end{pmatrix} + \nabla f(a_1,b_2)\cdot\begin{pmatrix}\frac{b_1-a_1}{2}\\ -\frac{b_2-a_2}{2}\end{pmatrix}\;) \\ &\quad + \Big(1-\frac{3\beta}{2}\Big)\Big(\;f(a_1,a_2)+f(a_1,b_2)\;)\;\Big) \\ &\quad - \frac{\beta}{2}(\;f(b_1,a_2)+f(b_1,b_2)\;)\end{aligned} \tag{62}$$

Doing similar steps to the first bound for the other three bounds we obtain the full update directions for (a_1, a_2, b_1, b_2)

$$\begin{aligned}&\lim_{\alpha\to 0}\frac{\partial L}{\partial a_1} \approx \frac{(b_2-a_2)}{2}(\\ &\beta(\;\nabla f(a_1,a_2)\cdot\begin{pmatrix}(b_1-a_1)\\ (b_2-a_2)\end{pmatrix} + \nabla f(a_1,b_2)\cdot\begin{pmatrix}(b_1-a_1)\\ -(b_2-a_2)\end{pmatrix}\;) \\ &\quad + \Big(1-\frac{3\beta}{2}\Big)\Big(\;f(a_1,a_2)+f(a_1,b_2)\;)\;\Big) \\ &\quad - \frac{\beta}{2}(\;f(b_1,a_2)+f(b_1,b_2)\;)\end{aligned} \tag{63}$$

$$\begin{aligned}&\lim_{\alpha\to 0}\frac{\partial L}{\partial b_1} \approx \frac{(b_2-a_2)}{2}(\\ &\beta(\;\nabla f(b_1,a_2)\cdot\begin{pmatrix}(b_1-a_1)\\ -(b_2-a_2)\end{pmatrix} + \nabla f(b_1,b_2)\cdot\begin{pmatrix}(b_1-a_1)\\ (b_2-a_2)\end{pmatrix}\;) \\ &\quad - \Big(1-\frac{3\beta}{2}\Big)\Big(\;f(b_1,a_2)+f(b_1,b_2)\;)\;\Big) \\ &\quad + \frac{\beta}{2}(\;f(a_1,a_2)+f(a_1,b_2)\;)\end{aligned} \tag{64}$$

$$\begin{aligned}&\lim_{\alpha\to 0}\frac{\partial L}{\partial a_2} \approx \frac{(b_2-a_2)}{2}(\\ &\beta(\;\nabla f(a_1,a_2)\cdot\begin{pmatrix}(b_1-a_1)\\ (b_2-a_2)\end{pmatrix} + \nabla f(b_1,a_2)\cdot\begin{pmatrix}-(b_1-a_1)\\ (b_2-a_2)\end{pmatrix}\;) \\ &\quad + \Big(1-\frac{3\beta}{2}\Big)\Big(\;f(a_1,a_2)+f(b_1,a_2)\;)\;\Big) \\ &\quad - \frac{\beta}{2}(\;f(a_1,b_2)+f(b_1,b_2)\;)\end{aligned} \tag{65}$$

$$\begin{aligned}&\lim_{\alpha\to 0}\frac{\partial L}{\partial b_2} \approx \frac{(b_2-a_2)}{2}(\\ &\beta(\;\nabla f(a_1,b_2)\cdot\begin{pmatrix}-(b_1-a_1)\\ (b_2-a_2)\end{pmatrix} + \nabla f(a_2,b_2)\cdot\begin{pmatrix}(b_1-a_1)\\ (b_2-a_2)\end{pmatrix}\;) \\ &\quad - \Big(1-\frac{3\beta}{2}\Big)\Big(\;f(a_1,b_2)+f(b_1,b_2)\;)\;\Big) \\ &\quad + \frac{\beta}{2}(\;f(a_1,a_2)+f(b_1,a_2)\;)\end{aligned} \tag{66}$$

Now, for $f : \mathbb{R}^n \to (0,1)$Following previous notations we have the following expressions for the loss and update di-

rections for the bound

$$
\begin{aligned}
L(\mathbf{a},\mathbf{b}) &\approx \frac{(1+\alpha)^n \mathbf{1}^T \mathbf{f}_{\mathbb{Q}}}{\mathbf{1}^T \mathbf{f}_{\mathbb{D}}} \; - \; 1 \\
\nabla_{\mathbf{a}} L &\approx \triangle\Big(\text{diag}^{-1}(\mathbf{r})\overline{\mathbf{M}}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} \; + \; \beta\text{diag}(\overline{\mathbf{M}}\mathbf{G}_{\mathbb{D}}) \; + \beta\overline{\mathbf{s}}\Big) \\
\nabla_{\mathbf{b}} L &\approx \triangle\Big(- \text{diag}^{-1}(\mathbf{r})\mathbf{M}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} \; + \; \beta\text{diag}(\mathbf{M}\mathbf{G}_{\mathbb{D}}) \; + \beta\mathbf{s}\Big)
\end{aligned} \tag{67}
$$

where the mask is the special mask

$$
\begin{aligned}
\overline{\mathbf{M}}_{\mathbb{D}} &= \Big(\gamma_n \overline{\mathbf{M}} \; - \; \beta\mathbf{M}\Big) \\
\mathbf{M}_{\mathbb{D}} &= \Big(\gamma_n \mathbf{M} \; - \; \beta\overline{\mathbf{M}}\Big) \\
\gamma_n &= 2 \; - \; \beta(2n-1)
\end{aligned} \tag{68}
$$

Where diag(.) is the diagonal matrix of the vector argument or the diagonal vector of the matrix argument. $\mathbf{s}$ is a weighted sum of the gradient from other dominions ($i \neq k$) contributing to the update direction of dimension k, where $k \in \{1, 2, ..., n\}$.

$$
\begin{aligned}
\mathbf{s}_k &= \frac{1}{\mathbf{r}_k} \sum_{i=1, i\neq k}^{n} \mathbf{r}_i((\overline{\mathbf{M}}_{i,:} - \mathbf{M}_{i,:}) \odot \overline{\mathbf{M}}_{k,:})\mathbf{G}_{:,i} \\
\overline{\mathbf{s}}_k &= \frac{1}{\mathbf{r}_k} \sum_{i=1, i\neq k}^{n} \mathbf{r}_i((\mathbf{M}_{i,:} - \overline{\mathbf{M}}_{i,:}) \odot \mathbf{M}_{k,:})\mathbf{G}_{:,i} \\
& k \in \{1, 2, ..., n\}
\end{aligned} \tag{69}
$$

B.4. Trapezoidal Approximation Formulation

Here we use the trapezoidal approximation of the integral, a first-order approximation from Newton-Cortes formulas for numerical integration [35]. The rule states that a definite integral can be approximated as follows:

$$
\int_a^b f(u)du \approx (b-a)\frac{f(a)+f(b)}{2} \tag{70}
$$

asymptotic error estimate is given by $-\frac{(b-a)^2}{48}\left[f'(b) - f'(a)\right] \; + \; \mathcal{O}\Big(0.125\Big)$. So as long the derivatives are bounded by some lipschitz constant $\mathbb{L}$, then the error becomes bounded by the following $|\text{error}| \leq \mathbb{L}(b-a)^2$. The regularized loss of interest in Eq (25) becomes the following .

$$
\begin{aligned}
L &= -\text{Area}_{\text{in}} + \lambda \left|b-a\right|_2^2 \\
&\approx -(b-a)\frac{f(a)+f(b)}{2} + \lambda \left|b-a\right|_2^2
\end{aligned} \tag{71}
$$

taking the derivative of L approximation directly with respect to these bounds , yields the following update directions which are different from the expressions in Eq (26)

$$
\begin{aligned}
\frac{\partial L}{\partial a} &= -\frac{b-a}{2}f'(a) \; + \; \frac{f(a)+f(b)}{2} \; - \lambda(b-a) \\
\frac{\partial L}{\partial b} &= -\frac{b-a}{2}f'(b) \; - \; \frac{f(a)+f(b)}{2} \; + \lambda(b-a)
\end{aligned} \tag{72}
$$

note that it needs the first derivative $f'(.)$ of the function f evaluated at the bound to update that bound.

Extension to n-dimensions.
Lets start by $n = 2$. Now, we have $f : \mathbb{R}^2 \rightarrow (0, 1)$, and we define the loss integral as a function of four bounds of a rectangler region and apply the trapezoidal approximation as follows.

$$
\begin{aligned}
L(a_1, a_2, b_1, b_2) &= -\int_{a_1}^{b_1}\int_{a_2}^{b_2} f(u,v)dvdu \\
&\quad + \frac{\lambda}{2}\left|b_1 - a_1\right|_2^2 + \frac{\lambda}{2}\left|b_2 - a_2\right|_2^2 \\
&\approx \frac{\lambda}{2}\left|b_1 - a_1\right|_2^2 + \frac{\lambda}{2}\left|b_2 - a_2\right|_2^2 \\
&\quad - \frac{(b_1-a_1)(b_2-a_2)}{4}\Big(f(a_1,a_2) + f(b_1,a_2) + \\
&\qquad f(a_1,b_2) + f(b_1,b_2)\Big)
\end{aligned} \tag{73}
$$

Then following similar steps as in the one-dimensional case we can obtain the following update directions for the four bounds

$$
\begin{aligned}
\frac{\partial L}{\partial a_1} &= (b_1-a_1)(b_2-a_2)\frac{f_1'(a_1,a_2) + f_1'(a_1,b_2)}{4} \\
&- (b_2-a_2)\frac{f(a_1,a_2) + f(a_1,b_2) + f(b_1,a_2) + f(b_1,b_2)}{4} \\
&- \lambda(b_1-a_1)
\end{aligned} \tag{74}
$$

$$
\begin{aligned}
\frac{\partial L}{\partial a_2} &= (b_1-a_1)(b_2-a_2)\frac{f_2'(b_1,a_2) + f_2'(b_1,b_2)}{4} \\
&- (b_2-a_2)\frac{f(a_1,a_2) + f(a_1,b_2) + f(b_1,a_2) + f(b_1,b_2)}{4} \\
&- \lambda(b_2-a_2)
\end{aligned} \tag{75}
$$

$$
\begin{aligned}
\frac{\partial L}{\partial b_1} &= (b_1-a_1)(b_2-a_2)\frac{f_1'(a_1,a_2) + f_1'(b_1,a_2)}{4} \\
&+ (b_1-a_1)\frac{f(a_1,a_2) + f(a_1,b_2) + f(b_1,a_2) + f(b_1,b_2)}{4} \\
&+ \lambda(b_1-a_1)
\end{aligned} \tag{76}
$$

$$
\begin{aligned}
\frac{\partial L}{\partial b_2} &= (b_1-a_1)(b_2-a_2)\frac{f_2'(a_1,b_2) + f_2'(b_1,b_2)}{4} \\
&+ (b_1-a_1)\frac{f(a_1,a_2) + f(a_1,b_2) + f(b_1,a_2) + f(b_1,b_2)}{4} \\
&+ \lambda(b_2-a_2)
\end{aligned} \tag{77}
$$

Where $f_1'(.) = \frac{\partial f(u,v)}{\partial u}, f_2'(.) = \frac{\partial f(u,v)}{\partial v}$. extending the 2-dimensional to general n-dimensions is straight forward. For $f : \mathbb{R}^n \to (0,1)$, we define the following. Let the left bound vector be $\mathbf{a} = [a_1, a_2, ..., a_n]$ and the right bound vector $\mathbf{b} = [b_1, b_2, ..., b_n]$ define the n-dimensional hyper-rectangle region of interest. The region is then definesd as follows : $\mathbb{D} = \{\mathbf{x} : \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}$, Here, we assume the size of the region is positive at every dimension , *i.e.* $\mathbf{r} = \mathbf{b} - \mathbf{a} > \mathbf{0}$. The volume of the region $\mathbb{D}$ normalized by exponent of dimension n is expressed as in Eq (31), the region $\mathbb{D}$ is defined as in Eq (32), $\mathbf{M}$ is defined as in Eq (33), and $\mathbf{f}_{\mathbb{D}}$ is defined as in Eq (34). now we can see that the loss integral in Eq (73) becomes as follows .

$$\begin{aligned} L(\mathbf{a}, \mathbf{b}) &= \int \cdots \int_{\mathbb{D}} f(u_1, \ldots, u_n)\, du_1 \ldots du_n + \frac{\lambda}{2} |\mathbf{r}|^2 \\ &\approx \triangle \mathbf{1}^T \mathbf{f}_{\mathbb{D}} \; + \; \frac{\lambda}{2} |\mathbf{r}|^2 \end{aligned} \tag{78}$$

Similarly to Eq (34), we define the gradient matrix $\mathbf{G}_{\mathbb{D}}$ as the matrix of all gradient vectors evaluated at all corner points of $\mathbb{D}$

$$\mathbf{G}_{\mathbb{D}} = \Big[\nabla f(\mathbf{d}^1) \mid \nabla f(\mathbf{d}^2) \mid ... \mid \nabla f(\mathbf{d}^{2^n})\Big]^T \tag{79}$$

The update directions for the left bound $\mathbf{a}$ and the right bound $\mathbf{b}$ becomes as follows by the trapezoid approximation

$$\begin{aligned} \nabla_{\mathbf{a}} L &\approx \triangle \Big(\text{diag}(\overline{\mathbf{M}} \mathbf{G}_{\mathbb{D}}) \; + \; \mathbf{1}^T \mathbf{f}_{\mathbb{D}} \; \text{diag}^{-1}(\mathbf{r}) \mathbf{1} \Big) + \lambda \mathbf{r} \\ \nabla_{\mathbf{b}} L &\approx \triangle \Big(\text{diag}(\mathbf{M} \mathbf{G}_{\mathbb{D}}) \; - \; \mathbf{1}^T \mathbf{f}_{\mathbb{D}} \; \text{diag}^{-1}(\mathbf{r}) \mathbf{1} \Big) - \lambda \mathbf{r} \end{aligned} \tag{80}$$

Where $\overline{\mathbf{M}}$ is the complement of the binary mask matrix $\mathbf{M}$.

C. Analysis

C.1. Detected Robust Regions

We apply the algorithms on two semantic parameters (the azimuth angle of the camera around the object, and the elevation angle around the object) that are regularly used in the literature [18, 1]. When we use one parameter (the azimuth), we fix the elevation to $35°$. We used two instead of larger numbers because it is easier to verify and visualize 2D, unlike higher dimensions. Also, the complexity of sampling increase exponentially with dimensionality for those algorithms (albeit being much better than grid sampling, see Table 3). The regions in Figure 13,14 look vertical rectangles most of the time. This is because the scale of the horizontal-axis (0,360) is much smaller than the vertical axis (-10,90), so most regions are squares but looks rectangles because of figure scales.

C.2. hyper-parameters

How to select the hyper parameters in all the above algorithms? The answer is not straight forward. For the λ in the naive approach, it is set experimentally by trying different values and using the one which detects some regions that known to behave robustly. The values we found for this are $\lambda = 0.07 \sim 0.1$. The learning rate η is easier to detect with observing the convergence and depends on the full range of study. A rule of thumb is to make 0.001of the full range. For the OIR formulations, we have the boundary factor α which we set to 0.05. A rule of thumb is to set it to be between $0.5 \sim 1/N$, where N is the number of samples needed for that dimensions to adequately characterize the space. In our case $N = 180$, so $1/180 \approx 0.005$. The only hyperparameter with mathematically established bound is the emphasis factor of the OIR_W formulation β. The bound shown in Table 3 which is $0 \leq \beta \leq \frac{2}{2n-1}$ can be shown as follows. We start from Eq (68). This expression is the actual expression for the special masks (we apologize in the typos in the main paper). As we can see, the most important term is γ_n. IT dictates how the function at the boundaries determine the next move of the bounds. Here γ_n should always be positive to ensure the correct direction of the movement for positive function evaluation.

$$\begin{aligned} \gamma_n &> 0 \\ 2-(2n-1)\beta &> 0 \\ \beta &< \frac{2}{2n-2} \end{aligned} \tag{81}$$

C.3. Detest

The data set used is collected from ShapeNet [7] and consists of 10 classes and 10 shapes eaach that are all identified by at least ResNet50 trained on ImagNet. This criteria is important to obtain valid NSM and DSM. The classes are ['aeroplane',"bathtub",'bench', 'bottle','chair',"cup","piano",'rifle','vase',"toilet"]. Part of the dataset is shown in Figure 11,12. 4 shapes faced difficulty of rendering during the SRVR experiments , therefore , they were replaced by another shapes from the same class.

C.4. Possible Future Directions

We analyze DNNs from a semantic lens and show how more confident networks tend to create adversarial semantic regions inside highly confident regions. We developed a bottom-up approach to analyzing the networks semantically by growing adversarial regions, which scales well with dimensionality and we use it to benchmark the semantic robustness of DNNs. We aim to investigate how to use the insights we gain from this work to develop and train semantically robust networks from the start while maintaining accuracy. Another direct extension of our work is to develop large scale semantic robustness challenge where we label these robust/adversarial regions in the semantic space and release some of them to allow for training. Then, we test the trained models on the hidden test set to measure robustness while reporting the accuracy on ImageNet validation set to make sure that the features of the model did not get affected by the robust training.

Analysis Approach	Paradigm	Total Sampling complexity	Black -box Functions	Forward pass /step	Backward pass /step	Identification Capabaility	Hyper-parameters
Grid Sampling	top-down	$\mathcal{O}(N^n)$ $N \gg 2$	✓	-	-	Fully identifies the semantic map of DNN	no hyper-parameters
Naive	bottom-up	$\mathcal{O}(2^n)$	✓	2^n	0	finds strong robust regions only around $\mathbf{u}_0$	λ, experimentally determined
OIR_B	bottom-up	$\mathcal{O}(2^{n+1})$	✓	2^{n+1}	0	finds strong and week robust regions around $\mathbf{u}_0$	α, experimentally determined
OIR_W	bottom-up	$\mathcal{O}(2^n)$	✗	2^n	2^n	finds strong and week robust regions around $\mathbf{u}_0$	$0 \leq \beta < \frac{2}{2n-1}$ dependeds on n and Lipschitz constant $\mathbb{L}$

Table 3: **Semantic Analysis Techniques**: comparing different approaches to analyse the semantic robustness of DNN.

Algorithm 3: Robust n-dimensional Region Finding for Black-Box DNNs by Outer-Inner Ratios

Requires: Senatic Function of a DNN $f(\mathbf{u})$ in Eq (21), initial semantic parameter $\mathbf{u}_0$, number of iterations T , learning rate η , object shape $\mathbf{S}_z$ of class label z, boundary factor α, smalll ϵ

Form constant binary matrices $\mathbf{M}, \overline{\mathbf{M}}, \mathbf{M}_{\mathbb{Q}}, \overline{\mathbf{M}_{\mathbb{Q}}}, \mathbf{M}_{\mathbb{D}}, \overline{\mathbf{M}_{\mathbb{D}}}$

Initialize bounds $\mathbf{a}_0 \leftarrow \mathbf{u}_0 - \epsilon\mathbf{1}$, $\mathbf{b}_0 \leftarrow \mathbf{u}_0 + -\epsilon\mathbf{1}$

$\mathbf{r}_0 \leftarrow \mathbf{a}_0 - \mathbf{b}_0$, update region volume $\triangle_0$ as in Eq (31)

for $t \leftarrow 1$ **to** T **do**

form the all-corners function vectors $f_{\mathbb{D}}, f_{\mathbb{Q}}$ as in Eq (49)

$\nabla_{\mathbf{a}} L \leftarrow 2\triangle_{t-1}\text{diag}^{-1}(\mathbf{r}_{t-1})\Big(2\overline{\mathbf{M}}\mathbf{f}_{\mathbb{D}} - \overline{\mathbf{M}}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\Big)$

$\nabla_{\mathbf{b}} L \leftarrow 2\triangle_{t-1}\text{diag}^{-1}(\mathbf{r}_{t-1})\Big(-2\mathbf{M}\mathbf{f}_{\mathbb{D}} + \mathbf{M}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\Big)$

update bounds: $\mathbf{a}_t \leftarrow \mathbf{a}_{t-1} - \eta\nabla_{\mathbf{a}} L$, $\mathbf{b}_t \leftarrow \mathbf{b}_{t-1} - \eta\nabla_{\mathbf{b}} L$

$\mathbf{r}_t \leftarrow \mathbf{a}_t - \mathbf{b}_t$, update region volume $\triangle_t$ as in Eq (31)

end

Returns: robust region bounds: $\mathbf{a}_T, \mathbf{b}_T$.

Algorithm 4: Robust n-dimensional Region Finding for White-Box DNNs by Outer-Inner Ratios

Requires: Senatic Function of a DNN $f(\mathbf{u})$ in Eq (21), initial semantic parameter $\mathbf{u}_0$, , learning rate η , object shape $\mathbf{S}_z$ of class label z, emphasis factor β, smalll ϵ

Form constant binary matrices $\mathbf{M}, \overline{\mathbf{M}}, \mathbf{M}_{\mathbb{D}}, \overline{\mathbf{M}_{\mathbb{D}}}$

Initialize bounds $\mathbf{a}_0 \leftarrow \mathbf{u}_0 - \epsilon\mathbf{1}$, $\mathbf{b}_0 \leftarrow \mathbf{u}_0 + -\epsilon\mathbf{1}$

$\mathbf{r}_0 \leftarrow \mathbf{a}_0 - \mathbf{b}_0$, update region volume $\triangle_0$ as in Eq (31)

for $t \leftarrow 1$ **to** T **do**

form the all-corners function vector $f_{\mathbb{D}}$ as in Eq (49)

form the all-corners gradients matrix $\mathbf{G}_{\mathbb{D}}$ as in Eq (79)

form the gradient selection vectors $\mathbf{s}, \bar{\mathbf{s}}$ as in Eq (69)

$\nabla_{\mathbf{a}} L \leftarrow$ $\triangle_{t-1}\Big(\text{diag}^{-1}(\mathbf{r}_{t-1})\overline{\mathbf{M}}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\overline{\mathbf{M}}\mathbf{G}_{\mathbb{D}} + \beta\bar{\mathbf{s}}\Big)$

$\nabla_{\mathbf{b}} L \leftarrow$ $\triangle_{t-1}\Big(-\text{diag}^{-1}(\mathbf{r}_{t-1})\mathbf{M}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\mathbf{M}\mathbf{G}_{\mathbb{D}}) + \beta\mathbf{s}\Big)$

update bounds: $\mathbf{a}_t \leftarrow \mathbf{a}_{t-1} - \eta\nabla_{\mathbf{a}} L$, $\mathbf{b}_t \leftarrow \mathbf{b}_{t-1} - \eta\nabla_{\mathbf{b}} L$

$\mathbf{r}_t \leftarrow \mathbf{a}_t - \mathbf{b}_t$, update region volume $\triangle_t$ as in Eq (31)

end

Returns: robust region bounds: $\mathbf{a}_T, \mathbf{b}_T$.