Towards Analyzing Semantic Robustness of Deep Neural Networks

Anonymous ICCV submission

Paper ID 152

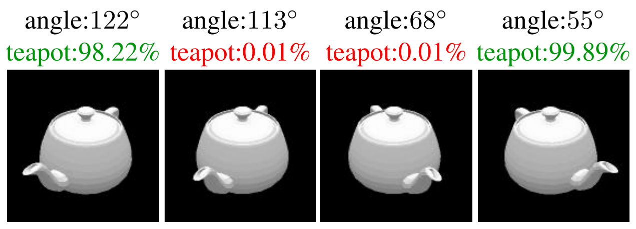
Abstract

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 that scale well with input dimensionality as opposed to naive sampling. We formalize the problem of finding robust semantic regions of the network and use this formulation to 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 InveptionV3's [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]).

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 [8, 40, 25], noise injection [10, 26, 3], or effect of image manipulation [12, 12, 11]. 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) show 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.



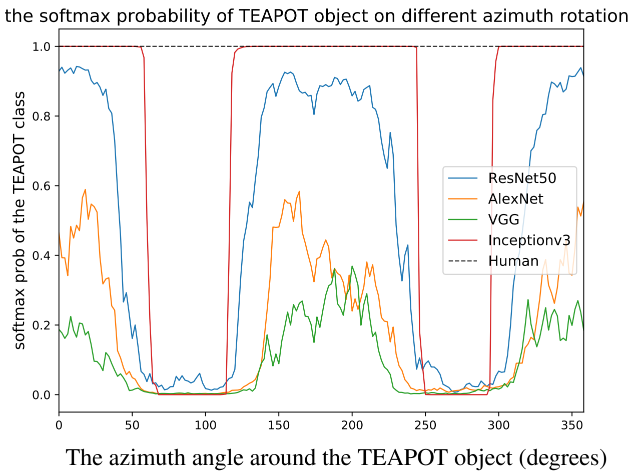


Figure 1: **Semantic Robustness of Deep Networks**. Trained Neural networks can perform poorly for small perturbations in the semantics of the image. We show how for a simple teapot object, perturbing the azimuth view angle of the object can dramatically affect the score of InceptionV [36] score of the teapot class.

Recent work in the adversarial attacks explores the DNNs' sensitivity and perform gradient updates to derive targeted perturbations [38, 13, 5, 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, 17]. 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 [17]. We present a novel approach to finds robust/adversarial regions in the n-dimensional semantic space that scale better than [17], 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 and conduct benchmarking of famous DNNs on a collected dataset by this metric Semantic Robustness Volume Ratio (SRVR).

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 [8, 25, 40]. Others observe the behavior of these networks under injected noise [10, 2, 4, 14, 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 [12]. 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 [11]. They also propose a probabilistic analysis framework to measure the robustness under these nuisance transformations (*e.g.*, perspective and illumination) [9], but they are limited to non-semantic 2D image transformation.

2.2. Adversarial Attacks on Deep Neural Networks

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 [13, 24, 27, 5]. 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, 7]. 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.

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 [15, 16]. 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.* [21]. NMR approximates the nondifferentiable 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 [17]. 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 region finding, we adopt an idea of weekly supervised activity detection in videos by [33] which focus on maximizing the inner average while minimizing the outer average of the function in the region and optimizing the bounds to achieve the ob-

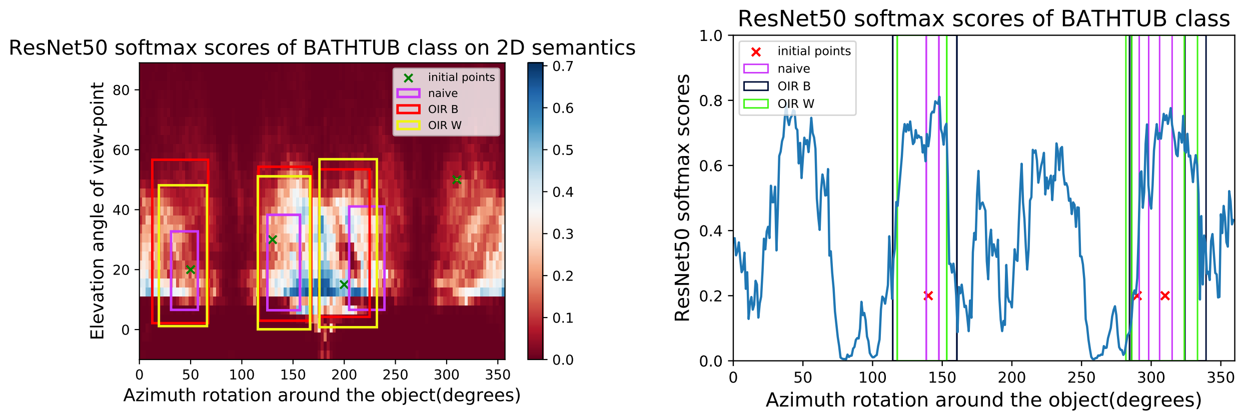


Figure 2: **Semantic Robust Region Finding**: finding robust regions in the of semantic parameters by the three formulations (naive , OIR_W , OIR_B) in (*left*) 2D case with 4 initial points (*right*).1D case with Three initial points points. We note that the naive approach usually predicts smaller regions, while the OIR formulations finds more comprehensive regions.

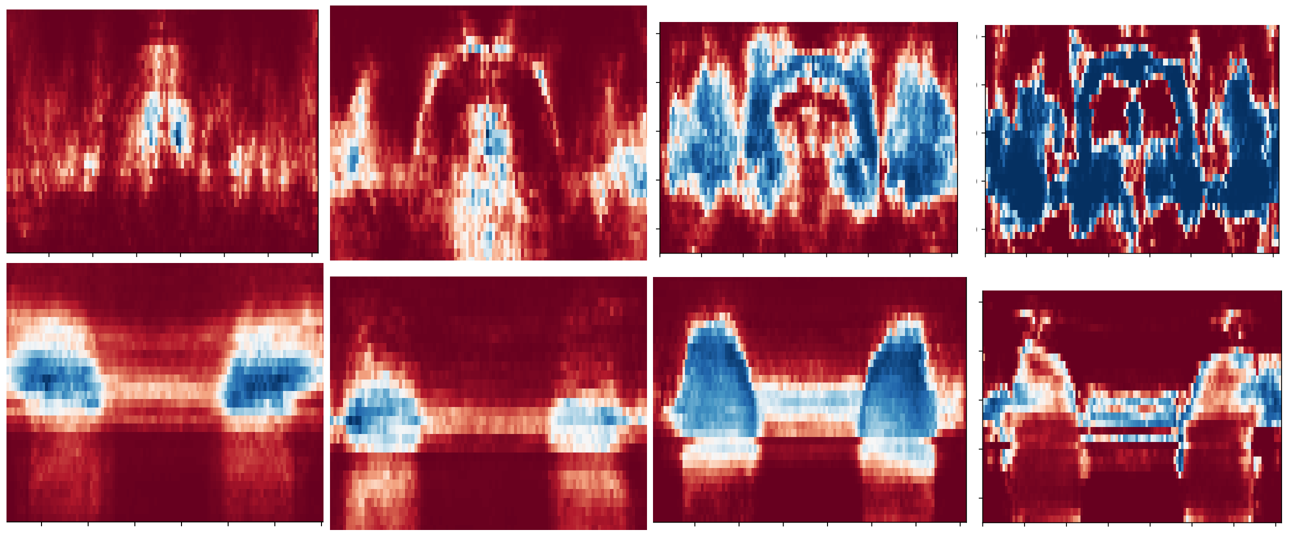


Figure 3: **Network Semantic Maps**: Plotting the 2D semantic maps of 4 different networks on two shapes of a chair class (*top*)and cup class (*bottom*). From left to right the networks are: AlexNet[23],VGG[34],Resnet50[18],and InceptionV3[36]. We note that Inception is very confident about its decision , but the cost is that it creates semantic "traps" where a hugh fall of performance happens in the middle of robust region. This behaviour is more apparent for complex shapes (like the chair)

jective. This is done because optimizing the bounds to only maximize the area can lead to diverging 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 $L = -\text{Area}_{\text{in}} + \frac{\lambda}{2}\left|b-a\right|_2^2 = \int_a^b f(u)du + \frac{\lambda}{2}\left|b-a\right|_2^2$, 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 hyperparameter. 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 regualirizer 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 approch to n-dimensions, we will face an another integral in the update direction (hard to compute). Therefore, we deply the following trapozoid 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 $-\frac{(b-a)^2}{48}\left[f'(b)-f'(a)\right]+\mathcal{O}\left(\frac{1}{8}\right)$. 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 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 $\mathbf{C}$. 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

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

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.

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 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 $n = 1$, and the complexity is N^2 for $n = 2$. We can see that for a general dimension n, the complexity 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 dont need to specify the whole Ω before finding the robust region around a point $\mathbf{u}$, which is an advantage of SADA [17]. As we will se in Section 4.4, this approach can be used to characterize the space much more efficiently as in Table 1 and used 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 , and 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 3. The robust-region-finding operator is then defined as follows

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

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 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 as:

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

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

$$\begin{aligned}\text{corners}(\mathbb{D}) = \mathbf{D}_{n\times 2^n} &= \left[\mathbf{d}^1 | \mathbf{d}^2 | .. | \mathbf{d}^{2^n}\right] \\ \mathbf{D} &= \mathbf{1}^T\mathbf{a}\ +\ \mathbf{M}^T \odot (\mathbf{1}^T\mathbf{r}) \\ \mathbf{M}_{n\times 2^n} &= \left[\mathbf{m}^0 | \mathbf{m}^1 | .. | \mathbf{m}^{2^n - 1}\right]\ , \text{ where } \ \mathbf{m}^i = \text{binary}_n(i)\end{aligned} \tag{5}$$

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 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 $\mathbb{D}$

$$\mathbf{f}_{\mathbb{D}} = \left[f(\mathbf{d}^1), f(\mathbf{d}^2), ..., f(\mathbf{d}^{2^n})\right]^T\ , \ \mathbf{d}^{\mathbf{i}} = \mathbf{D}_{:,i} \tag{6}$$

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

$$\begin{aligned}L(\mathbf{a}, \mathbf{b}) &= -\int \cdots \int_{\mathbb{D}} f(u_1, \dots, u_n)\, du_1 \dots du_n + \frac{\lambda}{2}\left|\mathbf{r}\right|^2 \\ &\approx\ -\triangle\mathbf{1}^T\mathbf{f}_{\mathbb{D}}\ +\ \frac{\lambda}{2}\left|\mathbf{r}\right|^2 \\ \nabla_{\mathbf{a}}L &\approx\ 2\triangle\text{diag}^{-1}(\mathbf{r})\overline{\mathbf{M}}\mathbf{f}_{\mathbb{D}} + \lambda\mathbf{r} \\ \nabla_{\mathbf{b}}L &\approx\ -2\triangle\text{diag}^{-1}(\mathbf{r})\mathbf{M}\mathbf{f}_{\mathbb{D}} - \lambda\mathbf{r}\end{aligned} \tag{7}$$

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

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.

$$\begin{aligned} L = \frac{\text{Area}_{\text{out}}}{\text{Area}_{\text{in}}} &= \text{Area}_{\text{out}} - \lambda\ \text{Area}_{\text{in}} \\ = \int_A^B f(a)du &- \int_a^b f(a)du - \lambda \int_a^b f(a)du \end{aligned} \tag{8}$$

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 yeilds 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 equivilantly. The inner region $\mathbb{D}$ is defined as in Eq (5) and outer regions can also be defined in terms of the corner points as follows.

$$\begin{aligned} \text{corners}(\mathbb{Q}) &= \mathbf{Q}_{n\times 2^n} = \left[\mathbf{q}^1|\mathbf{q}^2|..|\mathbf{q}^{2^n}\right] \\ \mathbf{Q} &= \mathbf{1}^T(\mathbf{a} - \frac{\alpha}{2}\mathbf{r}) + (1+\alpha)\mathbf{M}^T \odot (\mathbf{1}^T\mathbf{r}) \end{aligned} \tag{9}$$

Let $\mathbf{f}_{\mathbb{D}}$ be function vector as in Eq (6) and $\mathbf{f}_{\mathbb{Q}}$ be another function vector evaluated at all possible outer corner points as follows.

$$\mathbf{f}_{\mathbb{Q}} = \left[f(\mathbf{q}^1), f(\mathbf{q}^2), ..., f(\mathbf{q}^{2^n})\right]^T, \ \mathbf{q}^{\mathbf{i}} = \mathbf{Q}_{:,i} \tag{10}$$

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

$$\begin{aligned} L(\mathbf{a},\mathbf{b}) = &\int \cdots \int_{\mathbb{Q}} f(u_1,\dots,u_n)\, du_1 \dots du_n \\ &- 2\int \cdots \int_{\mathbb{D}} f(u_1,\dots,u_n)\, du_1 \dots du_n \\ &\approx \triangle\left((1+\alpha)^n \mathbf{1}^T\mathbf{f}_{\mathbb{Q}} - 2\,\mathbf{1}^T\mathbf{f}_{\mathbb{D}}\right) \\ \nabla_{\mathbf{a}}L \approx &2\triangle\text{diag}^{-1}(\mathbf{r})\left(2\overline{\mathbf{M}}\mathbf{f}_{\mathbb{D}} - \overline{\mathbf{M}}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\right) \\ \nabla_{\mathbf{b}}L \approx &2\triangle\text{diag}^{-1}(\mathbf{r})\left(-2\mathbf{M}\mathbf{f}_{\mathbb{D}} + \mathbf{M}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\right) \end{aligned} \tag{11}$$

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.

$$\begin{aligned} \overline{\mathbf{M}}_{\mathbb{Q}} &= (1+\alpha)^{n-1}\left((1+\frac{\alpha}{2})\overline{\mathbf{M}} + \frac{\alpha}{2}\mathbf{M}\right) \\ \mathbf{M}_{\mathbb{Q}} &= (1+\alpha)^{n-1}\left((1+\frac{\alpha}{2})\mathbf{M} + \frac{\alpha}{2}\overline{\mathbf{M}}\right) \end{aligned} \tag{12}$$

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 (8), 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 (8) becomes :

$$\begin{aligned} &\arg\min_{a,b} L = \arg\min_{a,b}\ \text{Area}_{\text{out}} - \lambda\ \text{Area}_{\text{in}} \\ &= \arg\min_{a,b} \int_A^a f(u)du + \int_b^B f(u)du - \frac{\alpha}{\beta}\int_a^b f(u)du \\ &= \arg\min_{a,b} \frac{\beta}{\alpha}\int_{a-\alpha\frac{b-a}{2}}^{b+\alpha\frac{b-a}{2}} f(u)du - (1+\frac{\beta}{\alpha})\int_a^b f(u)du \\ &\frac{\partial L}{\partial a} = \frac{\beta}{\alpha}\left(f(a) - f\left(a - \alpha\frac{b-a}{2}\right)\right) \\ &\quad - \frac{\beta}{2}f\left(b + \alpha\frac{b-a}{2}\right) - \frac{\beta}{2}f\left(a - \alpha\frac{b-a}{2}\right) + f(a) \end{aligned} \tag{13}$$

now since λ^* should be small for the optimal objective as $\lambda \rightarrow 0,\ \alpha \rightarrow 0$ and hence the derivative in Eq (13) becomes the following.

$$\begin{aligned} \lim_{\alpha\rightarrow 0}\frac{\partial L}{\partial a} &= \frac{\beta}{2}\left((b-a)f'(a) + f(b)\right) + (1 - \frac{\beta}{2})f(a) \\ \lim_{\alpha\rightarrow 0}\frac{\partial L}{\partial b} &= \frac{\beta}{2}\left((b-a)f'(b) + f(a)\right) - (1 - \frac{\beta}{2})f(b) \end{aligned} \tag{14}$$

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 $\beta \rightarrow 0$, the update directions in Eq (14) 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 $\mathbf{G}_{\mathbb{D}}$.

$$\mathbf{G}_{\mathbb{D}} = \left[\nabla f(\mathbf{d}^1)\ |\ \nabla f(\mathbf{d}^2)\ |\ ...\ |\ \nabla f(\mathbf{d}^{2^n})\right]^T \tag{15}$$

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

$$\begin{aligned} L(\mathbf{a},\mathbf{b}) &\approx \frac{(1+\alpha)^n\mathbf{1}^T\mathbf{f}_{\mathbb{Q}} - \mathbf{1}^T\mathbf{f}_{\mathbb{D}}}{\mathbf{1}^T\mathbf{f}_{\mathbb{D}}} - 1 \\ \nabla_{\mathbf{a}}L &\approx \triangle\left(\text{diag}^{-1}(\mathbf{r})\overline{\mathbf{M}}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\overline{\mathbf{M}}\mathbf{G}_{\mathbb{D}}) + \beta\overline{\mathbf{s}}\right) \\ \nabla_{\mathbf{b}}L &\approx \triangle\left(-\text{diag}^{-1}(\mathbf{r})\mathbf{M}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\mathbf{M}\mathbf{G}_{\mathbb{D}}) + \beta\mathbf{s}\right) \end{aligned} \tag{16}$$

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

```
Requires: Senatic Function of a DNN $f(\mathbf{u})$ in Eq (1), initial semantic parameter $\mathbf{u}_0$, number of iterations T, learning rate $\eta$, object shape $\mathbf{S}_z$ of class label $z$, boundary factor $\alpha$, smalll $\epsilon$
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 Section 3.1
for $t \leftarrow 1$ to $T$ do
    form the all-corners function vectors $f_{\mathbb{D}}, f_{\mathbb{Q}}$ as in Eq (10)
    $\nabla_{\mathbf{a}} L \leftarrow 2\triangle_{t-1}\text{diag}^{-1}(\mathbf{r}_{t-1})\left(2\overline{\mathbf{M}}\mathbf{f}_{\mathbb{D}} - \overline{\mathbf{M}_{\mathbb{Q}}}\mathbf{f}_{\mathbb{Q}}\right)$
    $\nabla_{\mathbf{b}} L \leftarrow 2\triangle_{t-1}\text{diag}^{-1}(\mathbf{r}_{t-1})\left(-2\mathbf{M}\mathbf{f}_{\mathbb{D}} + \mathbf{M}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\right)$
    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 as in Section 3.1
end
Returns: robust region bounds: $\mathbf{a}_T, \mathbf{b}_T$ .
```

where the mask is the special mask

$$\begin{aligned}\overline{\mathbf{M}}_{\mathbb{D}} &= \left((2-\beta n)\overline{\mathbf{M}} - \beta\mathbf{M}\right)\\ \mathbf{M}_{\mathbb{D}} &= \left((2-\beta n)\mathbf{M} - \beta\overline{\mathbf{M}}\right)\end{aligned} \tag{17}$$

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

$$\begin{aligned}\mathbf{s}_k &= \frac{1}{\mathbf{r}_k}\sum_{i=1,i\neq k}^{n}\mathbf{r}_i((\overline{\mathbf{M}}_{i,:} - \mathbf{M}_{i,:}) \odot \overline{\mathbf{M}}_{k,:})\mathbf{G}_{:,i}\\ \bar{\mathbf{s}}_k &= \frac{1}{\mathbf{r}_k}\sum_{i=1,i\neq k}^{n}\mathbf{r}_i((\mathbf{M}_{i,:} - \overline{\mathbf{M}}_{i,:}) \odot \mathbf{M}_{k,:})\mathbf{G}_{:,i}\end{aligned} \tag{18}$$

The derivation of the 2-dimensional case and n-dimensional case of the OIR formulation is included in the **supplementary material**. We try to use the trapozoid approximation directly on the loss and then differntiate the approximation. We get an expression that involves the deravative of the function , and we obtain an n-dimensional extension for it in the **supplementary material**. However, when applying to find the region it diverges (probably due to large approximations error). Algorithms 21 summarizes the techniques.

4. Experiments

4.1. Setup and Data

The semantic parameters $\mathbf{u}$ we pick are the azimuth rotations of the view point and the elevation angle from horizontal plane where the object is always at the center of the rendering, which is common in the literature [17, 19]. We

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

```
Requires: Senatic Function of a DNN $f(\mathbf{u})$ in Eq (1), initial semantic parameter $\mathbf{u}_0$, , learning rate $\eta$ , object shape $\mathbf{S}_z$ of class label $z$, emphasis factor $\beta$, smalll $\epsilon$
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 Section 3.1
for $t \leftarrow 1$ to $T$ do
    form the all-corners function vector $f_{\mathbb{D}}$ as in Eq (10)
    form the all-corners gradients matrix $\mathbf{G}_{\mathbb{D}}$ as in Eq (15)
    form the gradient selection vectors $\mathbf{s}, \bar{\mathbf{s}}$ as in Eq (18)
      $\nabla_{\mathbf{a}} L \leftarrow$
      $\triangle_{t-1}\left(\text{diag}^{-1}(\mathbf{r}_{t-1})\overline{\mathbf{M}}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\overline{\mathbf{M}}\mathbf{G}_{\mathbb{D}} + \beta\bar{\mathbf{s}}\right)$
      $\nabla_{\mathbf{b}} L \leftarrow$
      $\triangle_{t-1}\left(-\text{diag}^{-1}(\mathbf{r}_{t-1})\mathbf{M}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}} + \beta\text{diag}(\mathbf{M}\mathbf{G}_{\mathbb{D}}) + \beta\mathbf{s}\right)$
    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 as in Section 3.1
end
Returns: robust region bounds: $\mathbf{a}_T, \mathbf{b}_T$ .
```

use 100 shapes from 10 different classes from ShapeNet [6], 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 accept the shape only if its average Resnet softmax score is 0.1. To render the images we use differntible renderer NMR [21] which allows to obtain the gradient to the input semantic parameters .The networks of interest were Resnet50 [18], VGG [34], AlexNet [23], and InceptionV3 [36]. We use the official implementation by Pytorch models [29]. thos DNNs

4.2. Mapping the Networks

We map the networks similar to Figure 1m 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 3. 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 **supplementary material** .

4.3. Growing Semantic Robust Regions

We implement the three approaches in Table 1 from Section 3 and Algorithms 2,1, and we produce a toturial in jupyter Notebook [22] in the **supplementary material**implementing part of our project. The hyper-parameters

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 $\mathbf{u}_0$	$0 \leq \beta \leq \frac{1}{2n}$ dependeds on n and Lipschitz constant $\mathbb{L}$

Table 1: **Semantic Analysis Techniques**: Comparing different approaches to analyse the semantic robustness of DNN.

were set to $\eta = 0.1, \alpha = 0.05, \beta = 0.0009 \lambda = 0.1, T = 800$. We can observe in Figure 2 that multiple intial points inside the same robust region converge to the same boundary. One key difference to be noted between the naive approach in Eq (7) and the OIR formulations in Eq (11,16) is that naive approach fails to capture robust regions if the network score did not reach zero in the bound, so it will continue growing the region until it counter a very small score at the corner points (see fig [](a)).

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 metric (SRVR) . The SVR metric of the network is simply the ratio between the expected size of 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.

$$\text{SRVR}_z = \frac{\mathbb{E}[\text{Vol}(\mathbb{D})]}{\text{Vol}(\Omega)} = \frac{\mathbb{E}_{\mathbf{u}_0 \sim \Omega, \mathbf{S}_z \sim \mathbb{S}_z}[\text{Vol}(\mathbf{\Phi}(f, \mathbf{S}_z, \mathbf{u}_0))]}{\text{Vol}(\Omega)} \tag{19}$$

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 ideal behaviour of identifying the object robustly in the entire space. The SRVR metic is not strict in its value since we define the semantic space of interest and the shapes used , however its relative score to other networks is of extreme importance as it conveys information about the network that might not be clear from only the accuracy of the network. For example, ew can see in Table 2 that while Inceptionv3 [36] are is the best in terms of accuracy, it lags way behind

Deep Networ	SRVR	Top-1 error	Top-5 Error
AlexNet [23]	8.92%	43.45	20.91
VGG-11 [34]	9.88%	30.98	11.37
ResNet50 [18]	**16.88%**	23.85	7.13
Inceptionv3 [36]	7.92%	**22.55**	**6.44**

Table 2: **Benchmarking famous DNNs in Semantic Robustness vs Accuracy:** SRVR metric to estimate semantic robustness of famous Networks. We see that semantic robustness doesn't necessarily depends on the accuracy of the of the DNN, which motivates studying them as independent metric and learn algorithms to increase them while maintaining the accuracy of the network. Errors are reported from the [29] official implementation which we used.

Renet50[18] in terms of semantic robustness. Ths observation is also consistent with the qualitative NSMs in Figure 3in which we can see that while Inception is very confident. 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. As can be seen in Figure 2 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.

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 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 amound DNNs , indicating bias in the data. Identifying this gap in the data can help training a more semantically robust networks by adversarial training on these data-based

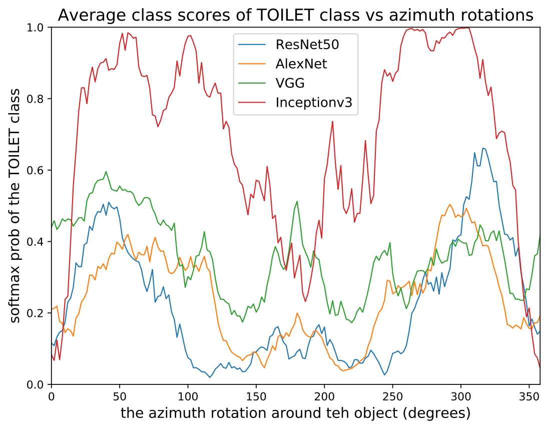


Figure 4: **Class Global Adversarial Regions** : average scores on 10 different shapes from the toilet class. We can see these semantic regions are shared among different shapes of that specific class as well as by different DNNs.

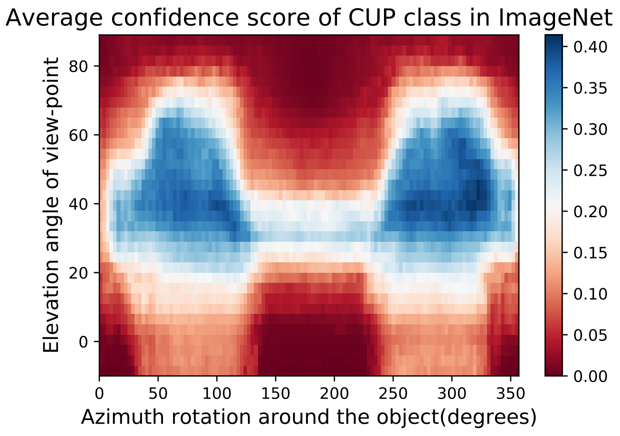


Figure 5: **Semantic Bias in ImageNet**. by taking the average semantic maps of different networks on the CUP class , over 10 shapes, we get a visualization of the bias of the data. Those angles of low score were not probably present in ImageNet[31].

adversrial regions as it is done in the adversarial attack literature [13]. 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.

5. Analysis

First, by observing the figures of the 1D plots we can see that that we can have robust regions in which lie adversarial regions . These "traps" are very dangerous in ML since you can identify with ese 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 tradeoff between accuracy and robustness , which is well established in the field of pixel perturbation robustness [3, 10].

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 $180^n = 180^8 = 1.1 * 10^{18}$, 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 next.

We can see that Inceptionv3 is the most accurate network (Table 2), but it is less robust (Table 2 and Figure 3) and it jumps sharply between very confident regions to almost zero confidence. This behaviour clearly violates our condition for semantic robust region as in Eq (2), and it is reflected on the qualitative and quantitative results. A possible explanation of this behaviour comes from Neuroscience which concludes that human brain process geometry and other semantic aspects unconsciously, before using the conscious-mind to identify objects for example [20], 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.

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

[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. 1, 2, 8

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

[5] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2017. 1, 2

[6] 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. 6

[7] 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

[8] 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. 1, 2

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

[10] 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. 1, 2, 8

[11] A. Fawzi, S. M. Moosavi Dezfooli, and P. Frossard. The robustness of deep networks - a geometric perspective. *IEEE Signal Processing Magazine*, 2017. 1, 2

[12] 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. 1, 2

[13] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 1, 2, 8

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

[15] U. Grenander. *Pattern analysis: lectures in pattern theory, volume I*. Springer, 1978. 2

[16] U. Grenander. *Pattern analysis: lectures in pattern theory, volume II*. Springer, 1978. 2

[17] A. Hamdi, M. Müller, and B. Ghanem. SADA: semantic adversarial diagnostic attacks for autonomous applications. *CoRR*, abs/1812.02132, 2018. 2, 4, 6

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3, 6, 7

[19] H. Hosseini and R. Poovendran. Semantic adversarial examples. *CoRR*, abs/1804.00499, 2018. 6

[20] D. Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, 2011. 8

[21] 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. 2, 6

[22] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016. 6

[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. 3, 6, 7

[24] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016. 1, 2

[25] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. *CoRR*, abs/1412.0035, 2014. 1, 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. 1

[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. 2, 6, 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

[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, 7, 8

[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. 2, 5

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 6, 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

[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, 3, 6, 7

[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. 1, 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. 1, 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