Towards Analyzing Semantic Robustness of Deep Neural Networks – Supplementary Materials –

Anonymous ICCV submission

Paper ID 152

1. 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 generater (*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 continous 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 behaviour of a classifier $\mathbf{C}$ of that image across multiple shapes and multiple famous DNNs. Now this function of interest is defined as follows.

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

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. 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 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 :

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

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

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

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).

1.1. Divergence of the Bounds

To develop an algorithm for $\mathbf{\Phi}$, we deploy the idea by [7] 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 1.1. *Let f be a continous 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 $\mathbf{p}^T \nabla L \geq 0$.[2].
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 1.2. *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 1.1 can diverge the bounds if followed in agradient ascent techinque 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 divregence. □

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 .

1.2. Naive Approach

$$L = -\text{Area}_{\text{in}} + \frac{\lambda}{2}\,|b-a|_2^2 \tag{5}$$

using Libeniz rule as in Lemma 1.1, we get the following update steps for the objective L :

$$\begin{aligned}\frac{\partial L}{\partial a} &= f(a) - \lambda(b-a)\\ \frac{\partial L}{\partial b} &= -f(b) + \lambda(b-a)\end{aligned} \tag{6}$$

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.

$$L(a_1, a_2, b_1, b_2) = \\ -\int_{a_2}^{b_2}\int_{a_1}^{b_1} f(u, v)dvdu + \frac{\lambda}{2}\,|b_1 - a_1|_2^2 + \frac{\lambda}{2}\,|b_2 - a_2|_2^2 \tag{7}$$

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

$$\begin{aligned}L(a_1, a_2, b_1, b_2) \;\approx\; &\frac{\lambda}{2}\,|b_1 - a_1|_2^2 + \frac{\lambda}{2}\,|b_2 - a_2|_2^2\\ &- \frac{(b_1 - a_1)(b_2 - a_2)}{4}\big(\, f(a_1, a_2) + f(b_1, a_2) +\\ &\qquad f(a_1, b_2) + f(b_1, b_2)\,\big)\end{aligned} \tag{8}$$

to find the update direction for the first bound a_1 by taking the partial derivative of the function in Eq (7) we get the following update direction for a_1 along with its trapezoidal approximation in order to able to compute it during the optimization:

$$\begin{aligned}\frac{\partial L}{\partial a_1} &= \int_{a_2}^{b_2} f(a_1, v)dv \;-\; \lambda(b_1 - a_1)\\ &\approx \frac{(b_2 - a_2)}{2}\Big(\, f(a_1, a_2) + f(a_1, b_2))\,\Big) \;-\; \lambda(b_1 - a_1)\end{aligned} \tag{9}$$

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)

$$\begin{aligned}\frac{\partial L}{\partial a_1} &\approx \frac{(b_2 - a_2)}{2}\Big(\, f(a_1, a_2) + f(a_1, b_2))\,\Big) \;-\; \lambda(b_1 - a_1)\\ \frac{\partial L}{\partial b_1} &\approx -\frac{(b_2 - a_2)}{2}\Big(\, f(b_1, a_2) + f(b_1, b_2))\,\Big) \;+\; \lambda(b_1 - a_1)\\ \frac{\partial L}{\partial a_2} &\approx \frac{(b_1 - a_1)}{2}\Big(\, f(a_1, a_2) + f(b_1, a_2))\,\Big) \;-\; \lambda(b_2 - a_2)\\ \frac{\partial L}{\partial b_2} &\approx -\frac{(b_1 - a_1)}{2}\Big(\, f(a_1, b_2) + f(b_1, b_2))\,\Big) \;+\; \lambda(b_2 - a_2)\end{aligned} \tag{10}$$

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

$$\text{volume}(\mathbb{D}) = \triangle = \frac{1}{2^n}\prod_{i=1}^{n}\mathbf{r}_i \tag{11}$$

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.

$$\begin{aligned}\text{corners}(\mathbb{D}) = \mathbf{D}_{n\times 2^n} &= \left[\mathbf{d}^1|\mathbf{d}^2|..|\mathbf{d}^{2^n}\right]\\ \mathbf{D} &= \mathbf{1}^T\mathbf{a} \;+\; \mathbf{M}^T \odot (\mathbf{1}^T\mathbf{r})\end{aligned} \tag{12}$$

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 $2n - 1$ deined as follws.

$$\mathbf{M}_{n\times 2^n} = \left[\mathbf{m}^0|\mathbf{m}^1|..|\mathbf{m}^{2^n-1}\right] \;,\; \text{where } \mathbf{m}^i = \text{binary}_n(i) \tag{13}$$

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

$$\mathbf{f}_{\mathbb{D}} = \left[f(\mathbf{d}^1), f(\mathbf{d}^2), ..., f(\mathbf{d}^{2^n})\right]^T \;,\; \mathbf{d}^{\mathbf{i}} = \mathbf{D}_{:,i} \tag{14}$$

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

$$\begin{aligned}L(\mathbf{a}, \mathbf{b}) &= -\int\cdots\int_{\mathbb{D}} f(u_1, \ldots, u_n)\, du_1 \ldots du_n + \frac{\lambda}{2}\,|\mathbf{r}|^2\\ &\approx\; -\triangle\mathbf{1}^T\mathbf{f}_{\mathbb{D}} \;+\; \frac{\lambda}{2}\,|\mathbf{r}|^2\\ \nabla_{\mathbf{a}}L &\approx\; 2\triangle\text{diag}^{-1}(\mathbf{r})\overline{\mathbf{M}}\mathbf{f}_{\mathbb{D}} + \lambda\mathbf{r}\\ \nabla_{\mathbf{b}}L &\approx\; -2\triangle\text{diag}^{-1}(\mathbf{r})\mathbf{M}\mathbf{f}_{\mathbb{D}} - \lambda\mathbf{r}\end{aligned} \tag{15}$$

1.3. 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.

$$A = a - \alpha\frac{b-a}{2}, B = b + \alpha\frac{b-a}{2} \tag{16}$$

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 [5]. Using their formulation to transform L as follows.

$$\begin{aligned} L = \frac{\text{Area}_{\text{out}}}{\text{Area}_{\text{in}}} &= \text{Area}_{\text{out}} - \lambda\,\text{Area}_{\text{in}} \\ &= \int_A^B f(a)du - \int_a^b f(a)du - \lambda\int_a^b f(a)du \end{aligned} \tag{17}$$

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

$$\begin{aligned} L &= \text{Area}_{\text{out}} - \text{Area}_{\text{in}} \\ &= \int_A^a f(u)du + \int_b^B f(u)du - \int_a^b f(u)du \\ &= \int_A^B f(u)du - 2\int_a^b f(u)du \\ &= \int_{a-\alpha\frac{b-a}{2}}^{b+\alpha\frac{b-a}{2}} f(u)du - 2\int_a^b f(u)du \end{aligned} \tag{18}$$

using Libeniz ruloe as in Lemma 1.1, we get the following update steps for the objective L :

$$\begin{aligned} \frac{\partial L}{\partial a} &= -(1+\frac{\alpha}{2})f(A) - \frac{\alpha}{2}f(B) + 2f(a) \\ \frac{\partial L}{\partial b} &= (1+\frac{\alpha}{2})f(B) + \frac{\alpha}{2}f(A) - 2f(b) \end{aligned} \tag{19}$$

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.

$$\begin{aligned} A_1 = a_1 - \frac{b_1 - a_1}{2}, \quad B_1 = b_1 + \frac{b_1 - a_1}{2} \\ A_2 = a_2 - \frac{b_2 - a_2}{2}, \quad B_1 = b_2 + \frac{b_2 - a_2}{2} \end{aligned} \tag{20}$$

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

$$\begin{aligned} &L(a_1, a_2, b_1, b_2) = \\ &\int_{A_2}^{B_2}\int_{A_1}^{B_1} f(u, v)dvdu - 2\int_{a_2}^{b_2}\int_{a_1}^{b_1} f(u, v)dvdu \end{aligned} \tag{21}$$

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

$$\begin{aligned} &L(a_1, a_2, b_1, b_2) \approx \\ &\frac{(B_1 - A_1)(B_2 - A_2)}{4}\big(\, f(A_1, A_2) + f(B_1, A_2) + \\ &\qquad f(A_1, B_2) + f(B_1, B_2)\,\big) \\ &-\frac{(b_1 - a_1)(b_2 - a_2)}{2}\big(\, f(a_1, a_2) + f(b_1, a_2) + \\ &\qquad f(a_1, b_2) + f(b_1, b_2)\,\big) \\ &= \frac{(b_1 - a_1)(b_2 - a_2)}{4}\Big(\\ &(1+\alpha)^2\big(\, f(A_1, A_2) + f(B_1, A_2) + \\ &\qquad f(A_1, B_2) + f(B_1, B_2)\,\big) \\ &-2\big(\, f(a_1, a_2) + f(b_1, a_2) + f(a_1, b_2) + f(b_1, b_2)\,\big)\,\Big) \end{aligned} \tag{22}$$

to find the update direction for the first bound a_1 by taking the partial derivative of the function in Eq (21) we get the following update direction for a_1 along with its trapezoidal approximation in order to able to compute it during the optimization:

$$\begin{aligned} \frac{\partial L}{\partial a_1} &= -(1+\frac{\alpha}{2})\int_{A_2}^{B_2}\Big(f(A_1, v) + \frac{\alpha}{2}f(B_1, v)\Big)dv \\ &+ 2\int_{a_2}^{b_2} f(a_1, v)dv \\ &\approx \frac{(b_2 - a_2)}{2}\Big(\\ &-(1+\alpha)\big(\,(1+\frac{\alpha}{2})(f(A_1, A_2) + f(A_1, B_2)) \\ &\qquad + \frac{\alpha}{2}\,(f(B_1, A_2) + f(B_1, B_2))\,\big) \\ &+2\Big(\, f(a_1, a_2) + f(a_1, b_2))\,\Big) \end{aligned} \tag{23}$$

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)

$$\begin{aligned} \frac{\partial L}{\partial a_1} \approx \frac{(b_2 - a_2)}{2}\Big(\\ -(1+\alpha)\big(\,(1+\frac{\alpha}{2})(f(A_1, A_2) + f(A_1, B_2)) \\ \qquad + \frac{\alpha}{2}\,(f(B_1, A_2) + f(B_1, B_2))\,\big) \\ +2\Big(\, f(a_1, a_2) + f(a_1, b_2))\,\Big) \end{aligned} \tag{24}$$

$$\frac{\partial L}{\partial b_1} \approx \frac{(b_2-a_2)}{2}\Big((1+\alpha)\Big((1+\frac{\alpha}{2})(f(B_1,A_2)+f(B_1,B_2)) \Big) + \frac{\alpha}{2}\ (f(A_1,A_2)+f(A_1,B_2)) - 2\Big(f(b_1,a_2)+f(b_1,b_2)) \Big)\Big) \tag{25}$$

$$\frac{\partial L}{\partial a_2} \approx \frac{(b_1-a_1)}{2}\Big(-(1+\alpha)\Big((1+\frac{\alpha}{2})(f(A_1,A_2)+f(B_1,A_2)) + \frac{\alpha}{2}\ (f(A_1,B_2)+f(B_1,B_2)) \Big) + 2\Big(f(a_1,a_2)+f(b_1,a_2)) \Big)\Big) \tag{26}$$

$$\frac{\partial L}{\partial b_2} \approx \frac{(b_1-a_1)}{2}\Big((1+\alpha)\Big((1+\frac{\alpha}{2})(f(A_1,B_2)+f(B_1,B_2)) + \frac{\alpha}{2}\ (f(A_1,A_2)+f(B_1,A_2)) \Big) - 2\Big(f(a_1,b_2)+f(b_1,b_2)) \Big)\Big) \tag{27}$$

Now, for $f:\mathbb{R}^n \to (0,1)$, we define the inner region hyper-rectangle as before $\mathbb{D}=\{\mathbf{x}:\mathbf{a}\le\mathbf{x}\le\mathbf{b}\}$, but now define an outer bigger region $\mathbb{Q}$ that include the smaller region $\mathbb{D}$ and defined as follows : $\mathbb{Q}=\{\mathbf{x}:\mathbf{a}-\frac{\alpha}{2}\mathbf{r}\le\mathbf{x}\le\mathbf{b}+\frac{\alpha}{2}\mathbf{r}\}$, 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.

$$\begin{aligned}\text{corners}(\mathbb{D}) = \mathbf{D}_{n\times 2^n} &= \Big[\mathbf{d}^1|\mathbf{d}^2|..|\mathbf{d}^{2^n}\Big]\\ \mathbf{D} &= \mathbf{1}^T\mathbf{a}\ +\ \mathbf{M}^T\odot(\mathbf{1}^T\mathbf{r})\\ \text{corners}(\mathbb{Q}) = \mathbf{Q}_{n\times 2^n} &= \Big[\mathbf{q}^1|\mathbf{q}^2|..|\mathbf{q}^{2^n}\Big]\\ \mathbf{Q} &= \mathbf{1}^T(\mathbf{a}-\frac{\alpha}{2}\mathbf{r})\ +\ (1+\alpha)\mathbf{M}^T\odot(\mathbf{1}^T\mathbf{r})\end{aligned} \tag{28}$$

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 (60). Now we define two function vectors evaluated at all possible inner and outer corner points respectively.

$$\begin{aligned}\mathbf{f}_{\mathbb{D}} &= \Big[f(\mathbf{d}^1),f(\mathbf{d}^2),...,f(\mathbf{d}^{2^n})\Big]^T,\ \mathbf{d^i}=\mathbf{D}_{:,i}\\ \mathbf{f}_{\mathbb{Q}} &= \Big[f(\mathbf{q}^1),f(\mathbf{q}^2),...,f(\mathbf{q}^{2^n})\Big]^T,\ \mathbf{c^i}=\mathbf{Q}_{:,i}\end{aligned} \tag{29}$$

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

$$\begin{aligned}L(\mathbf{a},\mathbf{b}) &= \int\cdots\int_{\mathbb{Q}} f(u_1,\dots,u_n)\,du_1\dots du_n - 2\int\cdots\int_{\mathbb{D}} f(u_1,\dots,u_n)\,du_1\dots du_n\\ &\approx \triangle\Big((1+\alpha)^n\mathbf{1}^T\mathbf{f}_{\mathbb{Q}}\ -\ 2\,\mathbf{1}^T\mathbf{f}_{\mathbb{D}}\Big)\\ \nabla_{\mathbf{a}}L &\approx 2\triangle\text{diag}^{-1}(\mathbf{r})\Big(2\overline{\mathbf{M}}\mathbf{f}_{\mathbb{D}}\ -\ \overline{\mathbf{M}}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\Big)\\ \nabla_{\mathbf{b}}L &\approx 2\triangle\text{diag}^{-1}(\mathbf{r})\Big(-2\mathbf{M}\mathbf{f}_{\mathbb{D}}\ +\ \mathbf{M}_{\mathbb{Q}}\mathbf{f}_{\mathbb{Q}}\Big)\end{aligned} \tag{30}$$

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

$$\begin{aligned}\overline{\mathbf{M}}_{\mathbb{Q}} &= (1+\alpha)^{n-1}\Big((1+\frac{\alpha}{2})\overline{\mathbf{M}}+\frac{\alpha}{2}\mathbf{M}\Big)\\ \mathbf{M}_{\mathbb{Q}} &= (1+\alpha)^{n-1}\Big((1+\frac{\alpha}{2})\mathbf{M}+\frac{\alpha}{2}\overline{\mathbf{M}}\Big)\end{aligned} \tag{31}$$

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 (17), 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 (17) becomes :

$$\begin{aligned}&\arg\min_{a,b} L = \arg\min_{a,b}\ \text{Area}_{\text{out}} - \lambda\ \text{Area}_{\text{in}}\\ &= \arg\min_{a,b}\ \int_A^a f(u)du + \int_b^B f(u)du - \frac{\alpha}{\beta}\int_a^b f(u)du\\ &= \arg\min_{a,b}\ \frac{\beta}{\alpha}\int_{a-\alpha\frac{b-a}{2}}^{b+\alpha\frac{b-a}{2}} f(u)du - (1+\frac{\beta}{\alpha})\int_a^b f(u)du\end{aligned} \tag{32}$$

using Libeniz ruloe as in Lemma 1.1, we get the following derivatives of the bound a :

$$\frac{\partial L}{\partial a} = \frac{\beta}{\alpha}\Big(f(a)-f\Big(a-\alpha\frac{b-a}{2}\Big)\Big) - \frac{\beta}{2}f\Big(b+\alpha\frac{b-a}{2}\Big)\ -\frac{\beta}{2}f\Big(a-\alpha\frac{b-a}{2}\Big)+f(a) \tag{33}$$

now since λ^* should be small for the optimal objective , then as $\lambda\to 0,\ \alpha\to 0$ and hence the derivative in Eq (33) becomes the following.

$$\lim_{\alpha\to 0}\frac{\partial L}{\partial a} = \lim_{\alpha\to 0}\beta\frac{\Big(f(a)-f\Big(a-\alpha\frac{b-a}{2}\Big)\Big)}{\alpha} - \frac{\beta}{2}f(b)\ -\frac{\beta}{2}f(a)+f(a) \tag{34}$$

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

$$\lim_{\alpha\to 0}\frac{\partial L}{\partial a} = \frac{\beta}{2}\Big((b-a)f'(a)+f(b)\Big) \;+\; (1-\frac{\beta}{2})f(a) \tag{35}$$

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

$$\lim_{\alpha\to 0}\frac{\partial L}{\partial b} = \frac{\beta}{2}\Big((b-a)f'(b)+f(a)\Big) \;-\; (1-\frac{\beta}{2})f(b) \tag{36}$$

If $\beta \to 0$, the update directions in Eq (35,36) collapse to the unregularized naive update in Eq (6) .

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.

$$\begin{aligned} A_1 &= a_1 - \frac{b_1-a_1}{2}, \;\; B_1 = b_1 + \frac{b_1-a_1}{2} \\ A_2 &= a_2 - \frac{b_2-a_2}{2}, \;\; B_1 = b_2 + \frac{b_2-a_2}{2} \end{aligned} \tag{37}$$

we follow similar approach as in Eq (32) to obtain the following expression.

$$\begin{aligned} &L(a_1,a_2,b_1,b_2) = \\ &\frac{\beta}{\alpha}\int_{A_2}^{B_2}\int_{A_1}^{B_1} f(u,v)dvdu \;-\; (1+\frac{\beta}{\alpha})\int_{a_2}^{b_2}\int_{a_1}^{b_1} f(u,v)dvdu \end{aligned} \tag{38}$$

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

$$\begin{aligned} &L(a_1,a_2,b_1,b_2) \approx -1 + (1+\alpha)^2 \\ &\frac{(\; f(A_1,A_2)+f(B_1,A_2)+f(A_1,B_2)+f(B_1,B_2)\;)}{(\; f(a_1,a_2)+f(b_1,a_2)+f(a_1,b_2)+f(b_1,b_2)\;)} \end{aligned} \tag{39}$$

to find the update direction for the first bound a_1 by taking the partial derivative of the function in Eq (21) 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:

$$\begin{aligned} \frac{\partial L}{\partial a_1} &= -\frac{\beta}{\alpha}(1+\frac{\alpha}{2})\int_{A_2}^{B_2}\Big(f(A_1,v)+\frac{\alpha}{2}f(B_1,v)\Big)dv \\ &\quad + \;(1+\frac{\beta}{\alpha})\int_{a_2}^{b_2} f(a_1,v)dv \\ &\approx \frac{(b_2-a_2)}{2}(\\ &\quad -(1+\alpha)(\;(\frac{\beta}{\alpha}+\frac{\beta}{2})(f(A_1,A_2)+f(A_1,B_2)) \\ &\qquad\qquad +\frac{\beta}{2}\;(f(B_1,A_2)+f(B_1,B_2))\;) \\ &\quad +(1+\frac{\beta}{\alpha})\Big(\;f(a_1,a_2)+f(a_1,b_2))\;\Big) \end{aligned} \tag{40}$$

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.

$$\begin{aligned} \lim_{\alpha\to 0}\frac{\partial L}{\partial a_1} &\approx \frac{(b_2-a_2)}{2}(\\ \lim_{\alpha\to 0}\frac{\beta}{\alpha}&(\;(\;f(a_1,a_2)+f(a_1,b_2))-(f(A_1,A_2)+f(A_1,B_2))\;) \\ &-\lim_{\alpha\to 0}\frac{3\beta}{2}\;(f(A_1,A_2)+f(A_1,B_2)\;) \\ &-\lim_{\alpha\to 0}\frac{\beta}{2}(\;f(B_1,A_2)+f(B_1,B_2)\;) \\ &+\Big(\;f(a_1,a_2)+f(a_1,b_2)\;)\;\Big) \end{aligned} \tag{41}$$

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

$$\begin{aligned} \lim_{\alpha\to 0}\frac{\partial L}{\partial a_1} &\approx \frac{(b_2-a_2)}{2}(\\ \beta(\;&\nabla f(a_1,a_2)\cdot\begin{pmatrix}\frac{b_1-a_1}{2}\\ \frac{b_2-a_2}{2}\end{pmatrix}+\nabla f(a_1,b_2)\cdot\begin{pmatrix}\frac{b_1-a_1}{2}\\ -\frac{b_2-a_2}{2}\end{pmatrix}\;) \\ &+\Big(1-\frac{3\beta}{2}\Big)\Big(\;f(a_1,a_2)+f(a_1,b_2)\;)\;\Big) \\ &-\frac{\beta}{2}(\;f(b_1,a_2)+f(b_1,b_2)\;) \end{aligned} \tag{42}$$

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)

$$\begin{aligned} \lim_{\alpha\to 0}\frac{\partial L}{\partial a_1} &\approx \frac{(b_2-a_2)}{2}(\\ \beta(\;&\nabla f(a_1,a_2)\cdot\begin{pmatrix}(b_1-a_1)\\ (b_2-a_2)\end{pmatrix}+\nabla f(a_1,b_2)\cdot\begin{pmatrix}(b_1-a_1)\\ -(b_2-a_2)\end{pmatrix}\;) \\ &+\Big(1-\frac{3\beta}{2}\Big)\Big(\;f(a_1,a_2)+f(a_1,b_2)\;)\;\Big) \\ &-\frac{\beta}{2}(\;f(b_1,a_2)+f(b_1,b_2)\;) \end{aligned} \tag{43}$$

$$\lim_{\alpha\to 0}\frac{\partial L}{\partial b_1} \approx \frac{(b_2-a_2)}{2}(\\ \beta(\ \nabla f(b_1,a_2)\cdot\begin{pmatrix}(b_1-a_1)\\-(b_2-a_2)\end{pmatrix}+\nabla f(b_1,b_2)\cdot\begin{pmatrix}(b_1-a_1)\\(b_2-a_2)\end{pmatrix}\ ) \\ -\Big(1-\frac{3\beta}{2}\Big)\Big(\ f(b_1,a_2)+f(b_1,b_2)\ )\Big) \\ +\frac{\beta}{2}(\ f(a_1,a_2)+f(a_1,b_2)\ ) \tag{44}$$

$$\lim_{\alpha\to 0}\frac{\partial L}{\partial a_2} \approx \frac{(b_2-a_2)}{2}(\\ \beta(\ \nabla f(a_1,a_2)\cdot\begin{pmatrix}(b_1-a_1)\\(b_2-a_2)\end{pmatrix}+\nabla f(b_1,a_2)\cdot\begin{pmatrix}-(b_1-a_1)\\(b_2-a_2)\end{pmatrix}\ ) \\ +\Big(1-\frac{3\beta}{2}\Big)\Big(\ f(a_1,a_2)+f(b_1,a_2)\ )\Big) \\ -\frac{\beta}{2}(\ f(a_1,b_2)+f(b_1,b_2)\ ) \tag{45}$$

$$\lim_{\alpha\to 0}\frac{\partial L}{\partial b_2} \approx \frac{(b_2-a_2)}{2}(\\ \beta(\ \nabla f(a_1,b_2)\cdot\begin{pmatrix}-(b_1-a_1)\\(b_2-a_2)\end{pmatrix}+\nabla f(a_2,b_2)\cdot\begin{pmatrix}(b_1-a_1)\\(b_2-a_2)\end{pmatrix}\ ) \\ -\Big(1-\frac{3\beta}{2}\Big)\Big(\ f(a_1,b_2)+f(b_1,b_2)\ )\Big) \\ +\frac{\beta}{2}(\ f(a_1,a_2)+f(b_1,a_2)\ ) \tag{46}$$

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

$$L(\mathbf{a},\mathbf{b}) \approx \frac{(1+\alpha)^n\mathbf{1}^T\mathbf{f}_{\mathbb{Q}}}{\mathbf{1}^T\mathbf{f}_{\mathbb{D}}}\ -\ 1 \\ \nabla_{\mathbf{a}}L \approx \triangle\Big(\text{diag}^{-1}(\mathbf{r})\overline{\mathbf{M}}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}}\ +\ \beta\text{diag}(\overline{\mathbf{M}}\mathbf{G}_{\mathbb{D}})\ +\beta\overline{\mathbf{s}}\Big) \\ \nabla_{\mathbf{b}}L \approx \triangle\Big(-\text{diag}^{-1}(\mathbf{r})\mathbf{M}_{\mathbb{D}}\mathbf{f}_{\mathbb{D}}\ +\ \beta\text{diag}(\mathbf{M}\mathbf{G}_{\mathbb{D}})\ +\beta\mathbf{s}\Big) \tag{47}$$

where the mask is the special mask

$$\overline{\mathbf{M}}_{\mathbb{D}} = \Big(\gamma_n\overline{\mathbf{M}}\ -\ \beta\mathbf{M}\Big) \\ \mathbf{M}_{\mathbb{D}} = \Big(\gamma_n\mathbf{M}\ -\ \beta\overline{\mathbf{M}}\Big) \\ \gamma_n = 2\ -\ \beta(2n-1) \tag{48}$$

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 $k\in\{1,2,...,n\}$.

$$\mathbf{s}_k = \frac{1}{\mathbf{r}_k}\sum_{i=1,i\neq k}^{n}\mathbf{r}_i((\overline{\mathbf{M}}_{i,:}-\mathbf{M}_{i,:})\odot\overline{\mathbf{M}}_{k,:})\mathbf{G}_{:,i} \\ \bar{\mathbf{s}}_k = \frac{1}{\mathbf{r}_k}\sum_{i=1,i\neq k}^{n}\mathbf{r}_i((\mathbf{M}_{i,:}-\overline{\mathbf{M}}_{i,:})\odot\mathbf{M}_{k,:})\mathbf{G}_{:,i} \\ k\in\{1,2,...,n\} \tag{49}$$

1.4. Trapezoidal Approximation Formulation

Here we use the trapezoidal approximation of the integral, a first-order approximation from Newton-Cortes formulas for numerical integration [8]. The rule states that a definite integral can be approximated as follows:

$$\int_a^b f(u)du \approx (b-a)\frac{f(a)+f(b)}{2} \tag{50}$$

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 (5) becomes the following .

$$L = -\text{Area}_{\text{in}}+\lambda\left|b-a\right|_2^2 \\ \approx -(b-a)\frac{f(a)+f(b)}{2}+\lambda\left|b-a\right|_2^2 \tag{51}$$

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 (6)

$$\frac{\partial L}{\partial a} = -\frac{b-a}{2}f'(a)\ +\ \frac{f(a)+f(b)}{2}\ -\lambda(b-a) \\ \frac{\partial L}{\partial b} = -\frac{b-a}{2}f'(b)\ -\ \frac{f(a)+f(b)}{2}\ +\lambda(b-a) \tag{52}$$

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 $f:\mathbb{R}^2\to(0,1)$, and we define the loss integral as a function of four bounds of a rectangler region and apply the trapezoidal approximation

as follows.

$$
\begin{aligned}
L(a_1,a_2,b_1,b_2) &= -\int_{a_1}^{b_1}\int_{a_2}^{b_2} f(u,v)dvdu \\
&\quad + \frac{\lambda}{2}\left|b_1-a_1\right|_2^2 + \frac{\lambda}{2}\left|b_2-a_2\right|_2^2 \\
&\approx \frac{\lambda}{2}\left|b_1-a_1\right|_2^2 + \frac{\lambda}{2}\left|b_2-a_2\right|_2^2 \\
&\quad - \frac{(b_1-a_1)(b_2-a_2)}{4}\Big(f(a_1,a_2)+f(b_1,a_2)+ \\
&\qquad f(a_1,b_2)+f(b_1,b_2)\Big)
\end{aligned} \tag{53}
$$

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

$$
\begin{aligned}
\frac{\partial L}{\partial a_1} &= (b_1-a_1)(b_2-a_2)\frac{f_1'(a_1,a_2)+f_1'(a_1,b_2)}{4} \\
&\quad - (b_2-a_2)\frac{f(a_1,a_2)+f(a_1,b_2)+f(b_1,a_2)+f(b_1,b_2)}{4} \\
&\quad - \lambda(b_1-a_1)
\end{aligned} \tag{54}
$$

$$
\begin{aligned}
\frac{\partial L}{\partial a_2} &= (b_1-a_1)(b_2-a_2)\frac{f_2'(b_1,a_2)+f_2'(b_1,b_2)}{4} \\
&\quad - (b_2-a_2)\frac{f(a_1,a_2)+f(a_1,b_2)+f(b_1,a_2)+f(b_1,b_2)}{4} \\
&\quad - \lambda(b_2-a_2)
\end{aligned} \tag{55}
$$

$$
\begin{aligned}
\frac{\partial L}{\partial b_1} &= (b_1-a_1)(b_2-a_2)\frac{f_1'(a_1,a_2)+f_1'(b_1,a_2)}{4} \\
&\quad + (b_1-a_1)\frac{f(a_1,a_2)+f(a_1,b_2)+f(b_1,a_2)+f(b_1,b_2)}{4} \\
&\quad + \lambda(b_1-a_1)
\end{aligned} \tag{56}
$$

$$
\begin{aligned}
\frac{\partial L}{\partial b_2} &= (b_1-a_1)(b_2-a_2)\frac{f_2'(a_1,b_2)+f_2'(b_1,b_2)}{4} \\
&\quad + (b_1-a_1)\frac{f(a_1,a_2)+f(a_1,b_2)+f(b_1,a_2)+f(b_1,b_2)}{4} \\
&\quad + \lambda(b_2-a_2)
\end{aligned} \tag{57}
$$

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 $f:\mathbb{R}^n \rightarrow (0,1)$, 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 : $\mathbb{D} = \{\mathbf{x} : \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}$, Herer , we assume the size of the region is positve at ervy 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

$$
\text{volume}(\mathbb{D}) = \triangle = \frac{1}{2^n}\prod_{i=1}^{n}\mathbf{r}_i \tag{58}
$$

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.

$$
\begin{aligned}
\text{corners}(\mathbb{D}) = \mathbf{D}_{n\times 2^n} &= \left[\mathbf{d}^1|\mathbf{d}^2|..|\mathbf{d}^{2^n}\right] \\
\mathbf{D} &= \mathbf{1}^T\mathbf{a} \;+\; \mathbf{M}^T \odot (\mathbf{1}^T\mathbf{r})
\end{aligned} \tag{59}
$$

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 $2n-1$

$$
\mathbf{M}_{n\times 2^n} = \left[\mathbf{m}^0|\mathbf{m}^1|..|\mathbf{m}^{2^n-1}\right] \;,\; \text{where } \mathbf{m}^i = \text{binary}_n(i) \tag{60}
$$

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

$$
\mathbf{f}_{\mathbb{D}} = \left[f(\mathbf{d}^1), f(\mathbf{d}^2), ..., f(\mathbf{d}^{2^n})\right]^T,\; \mathbf{d}^{\mathbf{i}} = \mathbf{D}_{:,i} \tag{61}
$$

now we can see that the loss integral in Eq (53) becomes as follows .

$$
\begin{aligned}
L(\mathbf{a},\mathbf{b}) &= \int\cdots\int_{\mathbb{D}} f(u_1,\ldots,u_n)\,du_1\ldots du_n + \frac{\lambda}{2}\left|\mathbf{r}\right|^2 \\
&\approx \triangle \mathbf{1}^T\mathbf{f}_{\mathbb{D}} \;+\; \frac{\lambda}{2}\left|\mathbf{r}\right|^2
\end{aligned} \tag{62}
$$

Similarly to Eq (61), we define the gradient matrix $\mathbf{G}_{\mathbb{D}}$ as the matrix of all gradient vectors evaluated at all corner points of $\mathbb{D}$

$$
\mathbf{G}_{\mathbb{D}} = \left[\nabla f(\mathbf{d}^1) \mid \nabla f(\mathbf{d}^2) \mid ... \mid \nabla f(\mathbf{d}^{2^n})\right]^T \tag{63}
$$

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

$$
\begin{aligned}
\nabla_{\mathbf{a}} L &\approx \triangle\Big(\text{diag}(\overline{\mathbf{M}}\mathbf{G}_{\mathbb{D}}) \;+\; \mathbf{1}^T\mathbf{f}_{\mathbb{D}}\,\text{diag}^{-1}(\mathbf{r})\mathbf{1}\Big) + \lambda\mathbf{r} \\
\nabla_{\mathbf{b}} L &\approx \triangle\Big(\text{diag}(\mathbf{M}\mathbf{G}_{\mathbb{D}}) \;-\; \mathbf{1}^T\mathbf{f}_{\mathbb{D}}\,\text{diag}^{-1}(\mathbf{r})\mathbf{1}\Big) - \lambda\mathbf{r}
\end{aligned} \tag{64}
$$

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

2. 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.

2.1. Networks Semantic Maps (1D)

In Figure 1,2 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.

2.2. Networks Semantic Maps (2D)

In Figure 3,4 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.

2.3. 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 comprehensive robust region. This happens even with different initial points, they always converge to the same bounds of that robust region of the semantic maps. Figure 5 show 4 different initializations for 1D case along with predicted regions. In Figure 6,7 shows the bounds evolving during the optimization of the three algorithms (naive , OIR_B and OIR_W) for 500 steps.

2.4. Examples of Found Regions (with Example Renderings)

In Figure 8,9 we provide examples of 2D regions found with the three algorithms along with renderings of the shapes from the robust regions detected.

2.5. Analyzing Semantic Data Bias in ImageNet

In Figure 10,11, we visualizing semantic data bias in the common training dataset (*i.e*. ImageNet [6]) 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. These places of high confidence probably reveal the areas where abundance of examples exists in ImageNet, while holes conveys scarcity of such examples in ImageNet corresponding class.

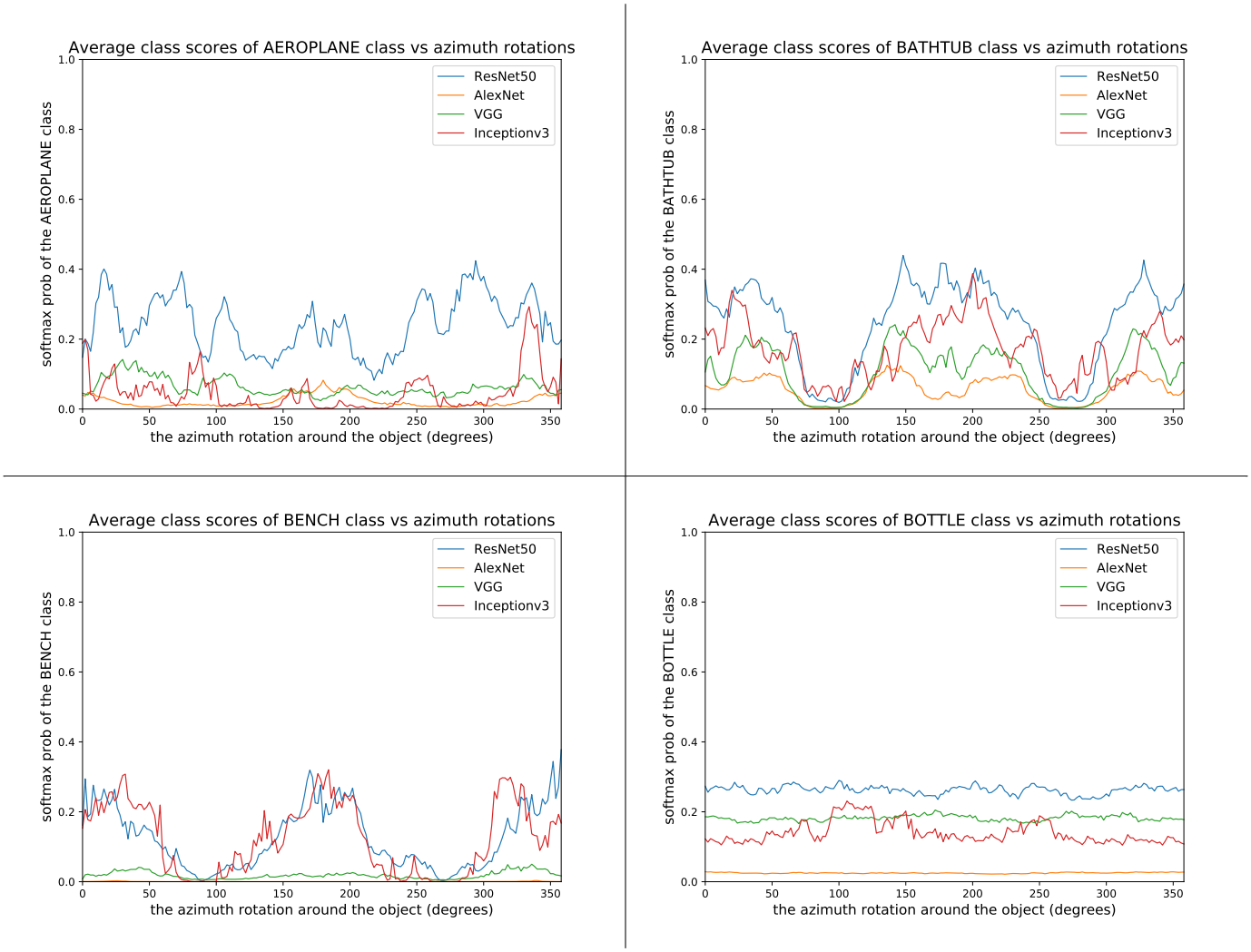


Figure 1: **1D Network Semantic Maps NMS-I**. Visualizing 1D Semantic Robustness profile for different networks averaged over 10 different shapes. Oserve that differnt DNNs profiles differ depending on the training , accuracy , and network architectures that all result in a unique ”signatures” for the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

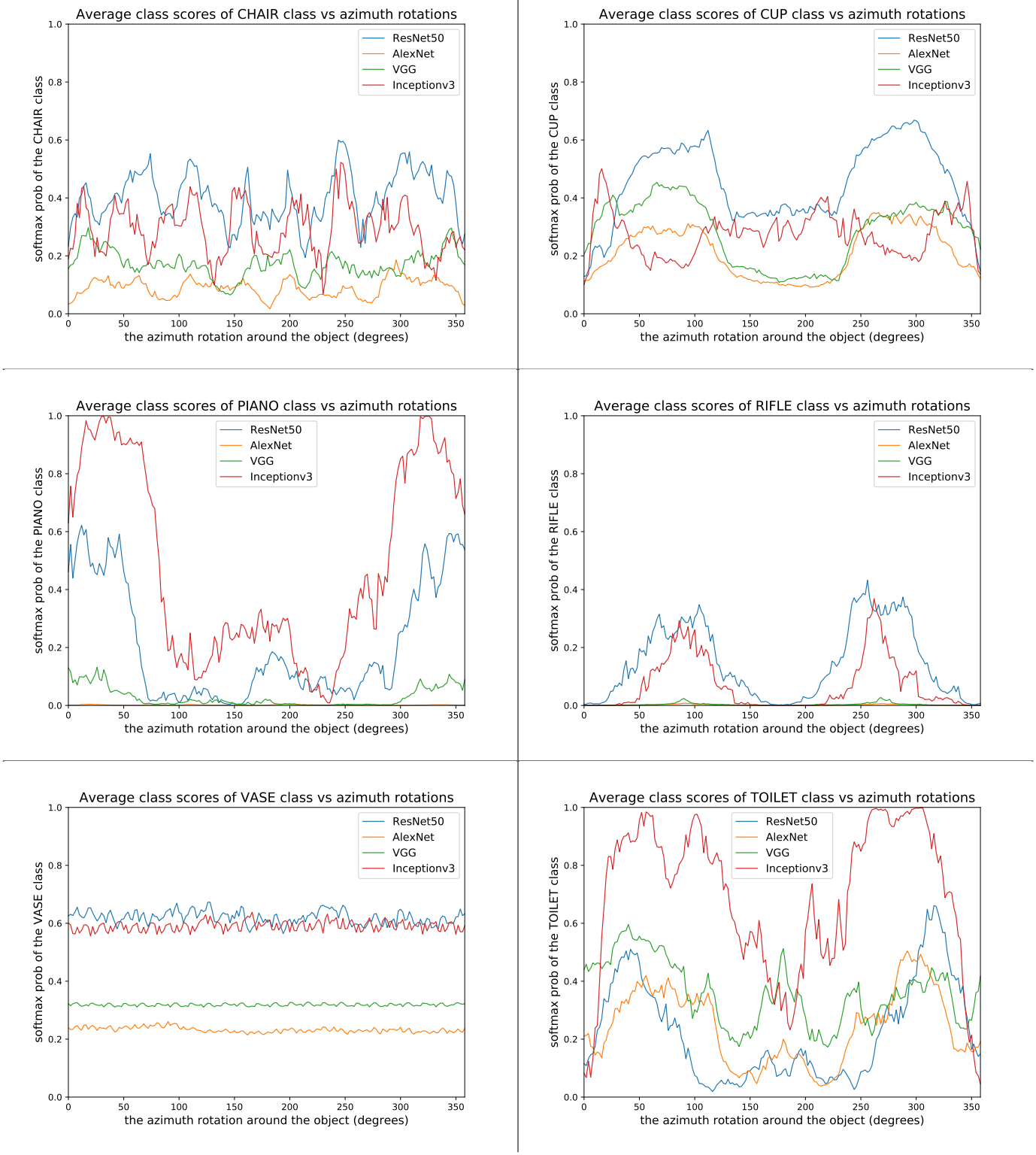


Figure 2: **1D Network Semantic Maps NMS-II**.Visualizing 1D Semantic Robustness profile for different networks averaged over 10 different shapes. Oserve that differnt DNNs profiles differ depending on the training , accuracy , and network architectures that all result in a unique ”signatures” for the DNN on that class. The correlation between the DNN profiles is due to the common data bias in ImageNet.

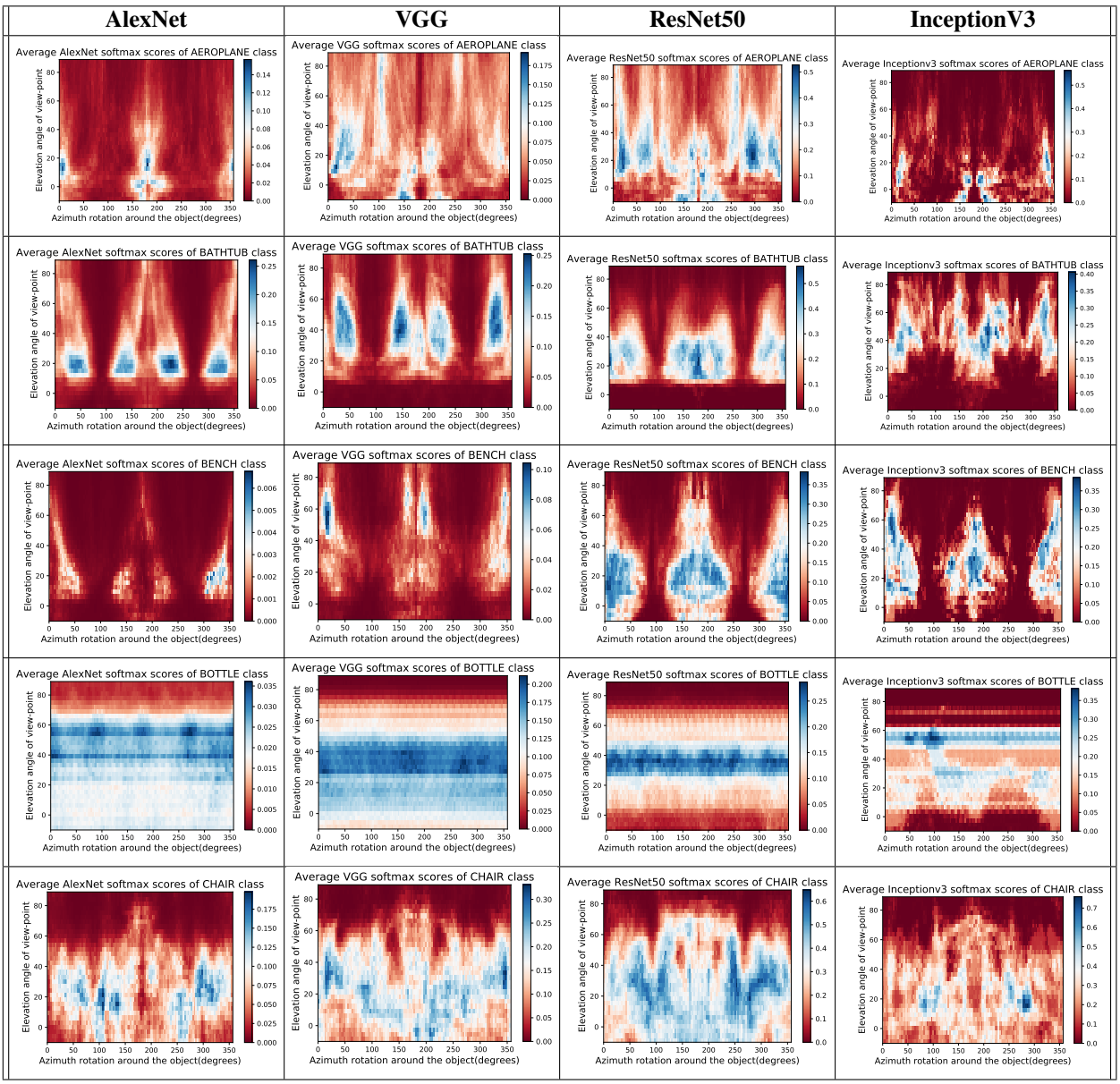


Figure 3: **2D Network Semantic Maps NMS-I**. Visualizing 2D Semantic Robustness profile for different networks averaged over 10 different shapes. Every row is different class. observe that differnt DNNs profiles differ depending on the training , accuracy , and network architectures that all result in a unique ”signatures” for the DNN on that class. Also observe InceptionV3 [9] creates traps of very low score inside a region of high confidence

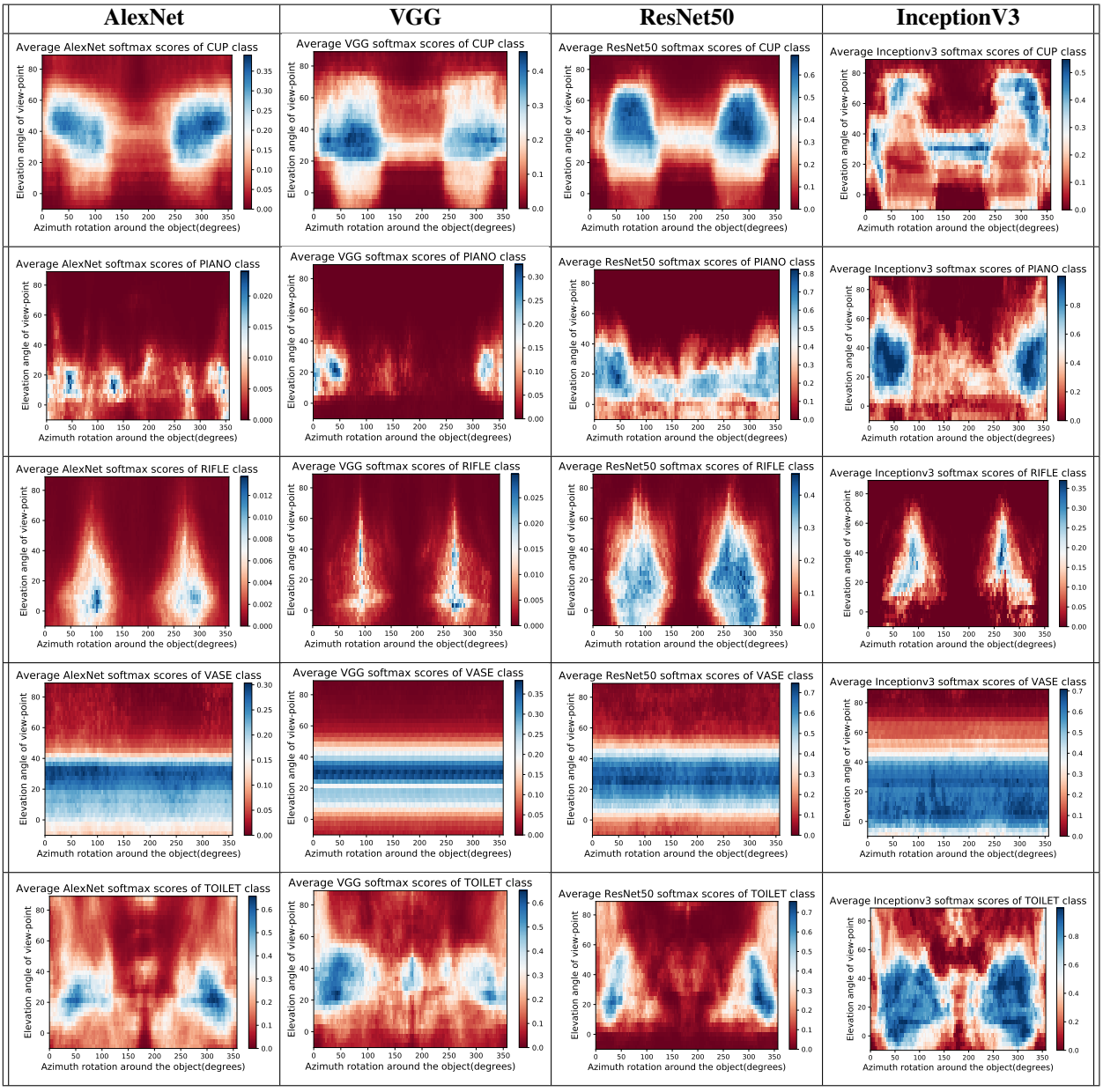


Figure 4: **2D Network Semantic Maps NMS-II**. Visualizing 2D Semantic Robustness profile for different networks averaged over 10 different shapes. Every row is different class. observe that differnt DNNs profiles differ depending on the training , accuracy , and network architectures that all result in a unique ”signatures” for the DNN on that class. Also observe InceptionV3 [9] creates traps of very low score inside a region of high confidence

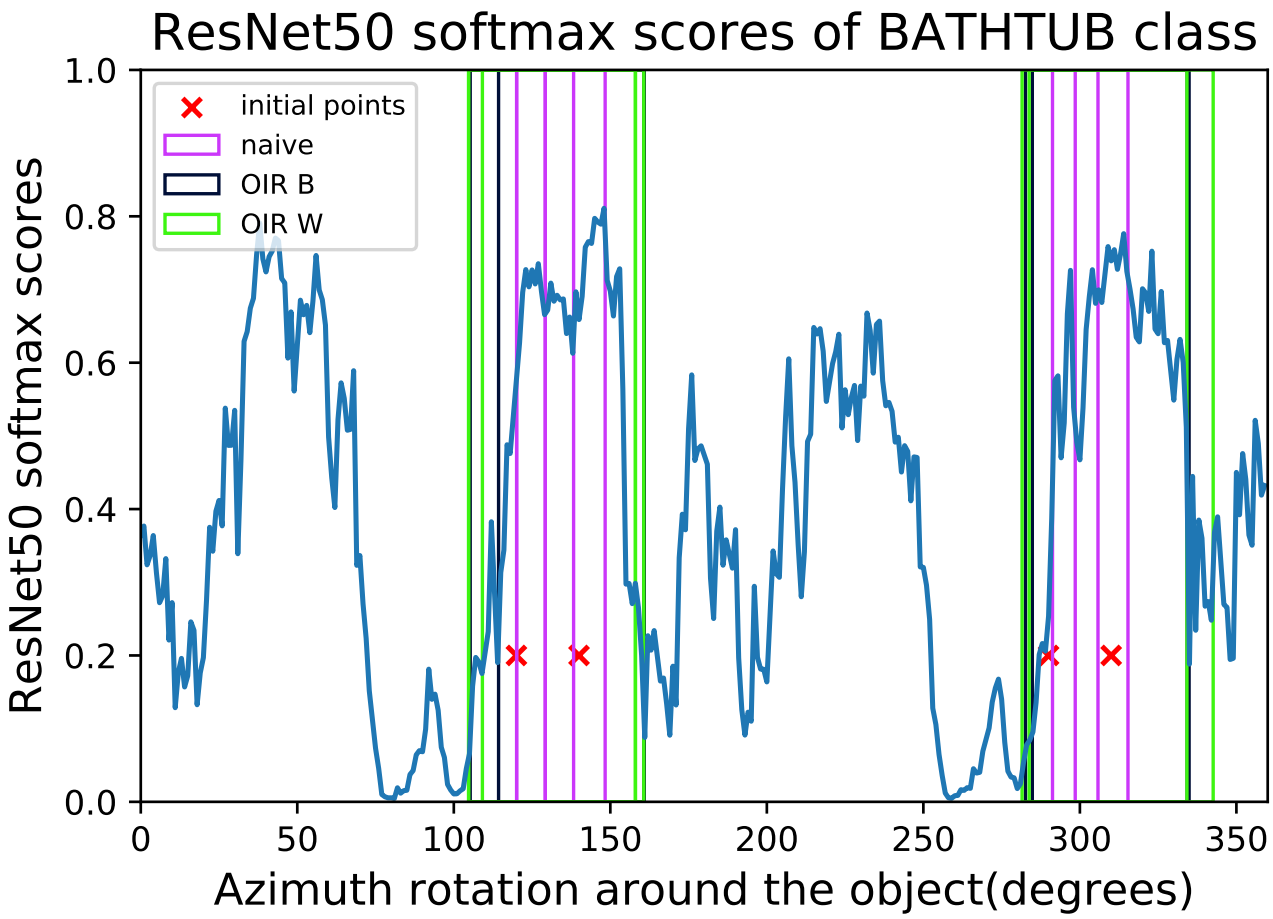


Figure 5: **Robust Region Detection with different initializations**Visualizing the Robust Regions Bounds found by the three algorithms for 4 initial points. we can see that the naive produce different bounds of the same region for different initializations while OIR detect the same region regardless of initialization.

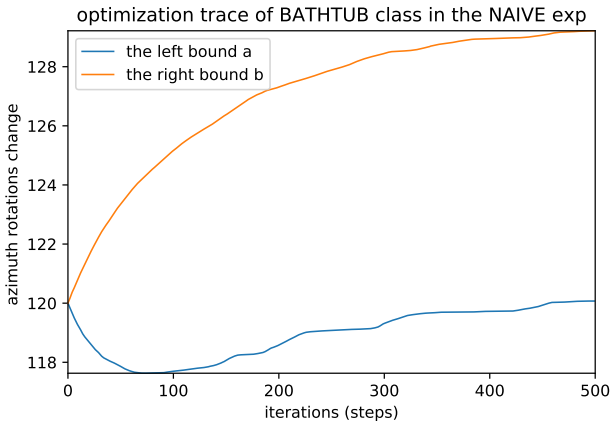
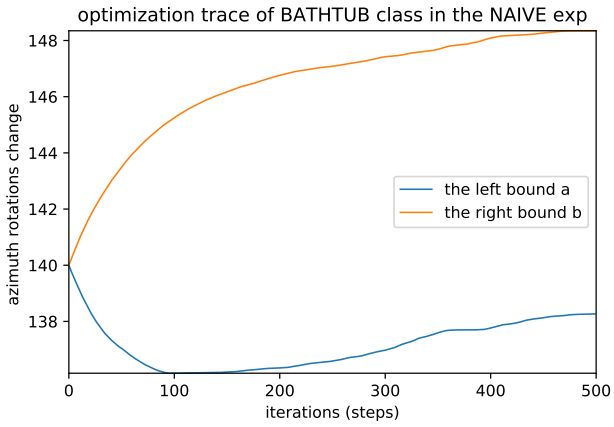
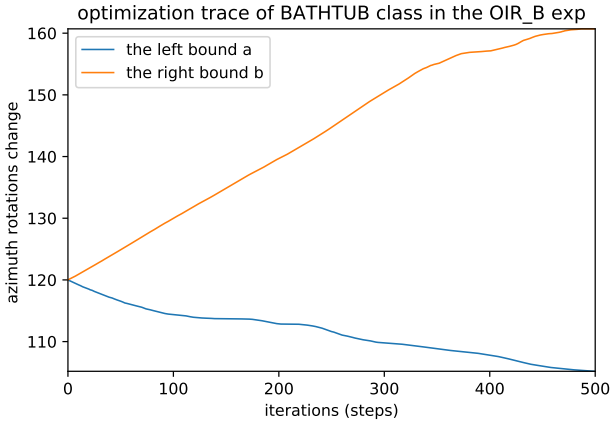
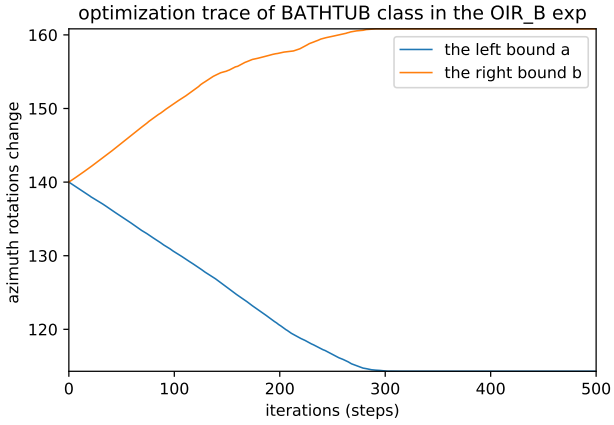
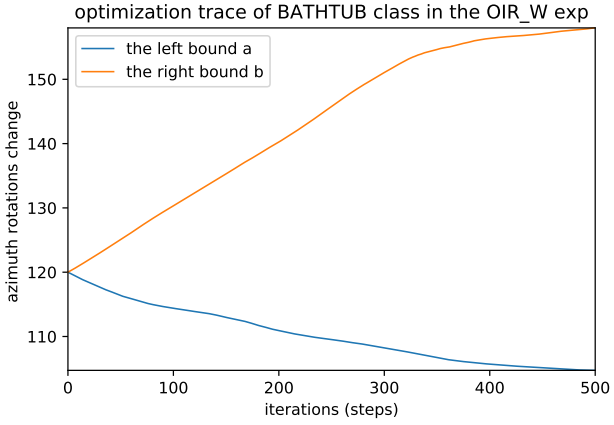
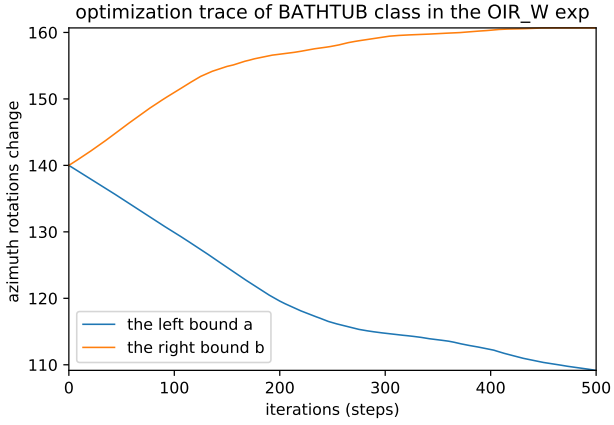
initialization = 120	initialization = 140
Naive algorithm	
	
Black-Box OIR algorithm	
	
White-Box OIR algorithm	
	

Figure 6: **Robust Region Bounds Growing I** Visualizing the bounds growing Using differnt algorithms from two different initializations (120 and 140) in Figure 5. 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.

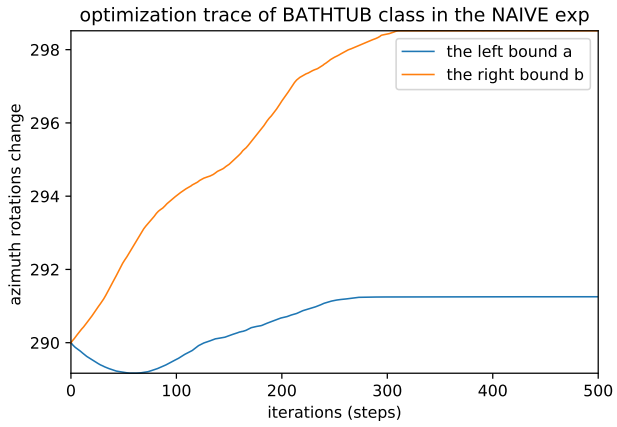
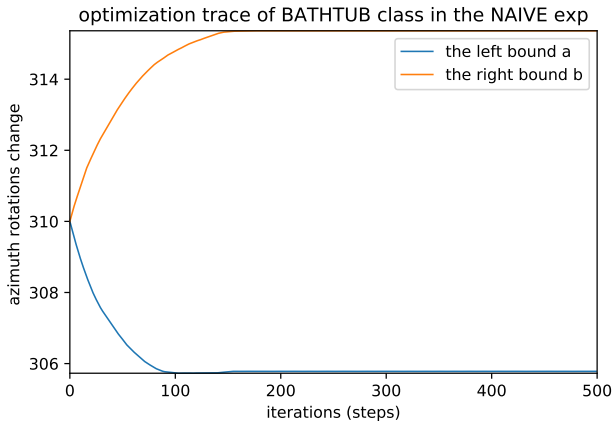
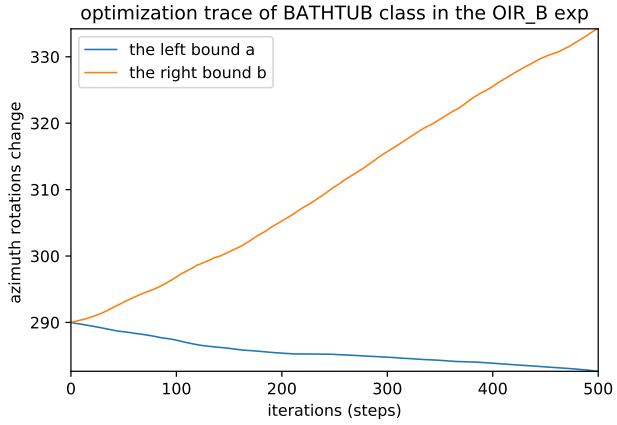
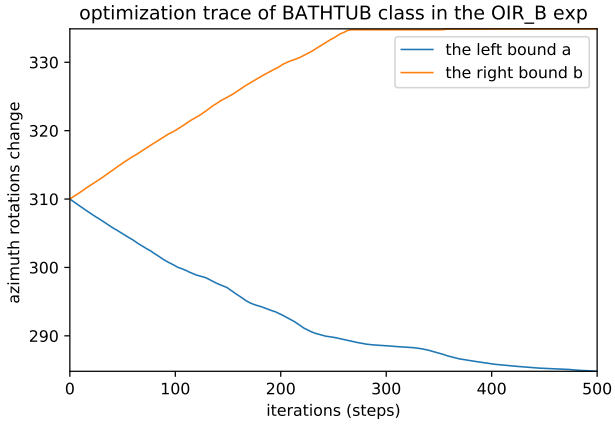
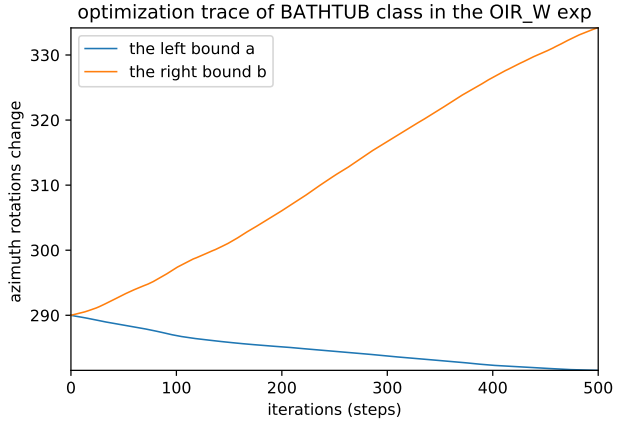
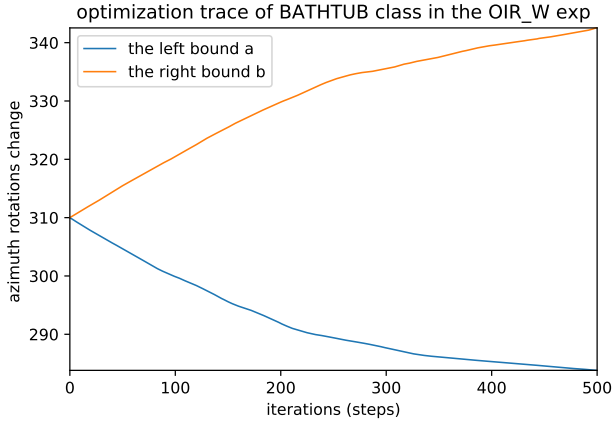
initialization = 290	initialization = 310
Naive algorithm	
	
Black-Box OIR algorithm	
	
White-Box OIR algorithm	
	

Figure 7: **Robust Region Bounds Growing I** Visualizing the bounds growing Using differnt algorithms from two different initializations (290, 310) in Figure 5. 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.

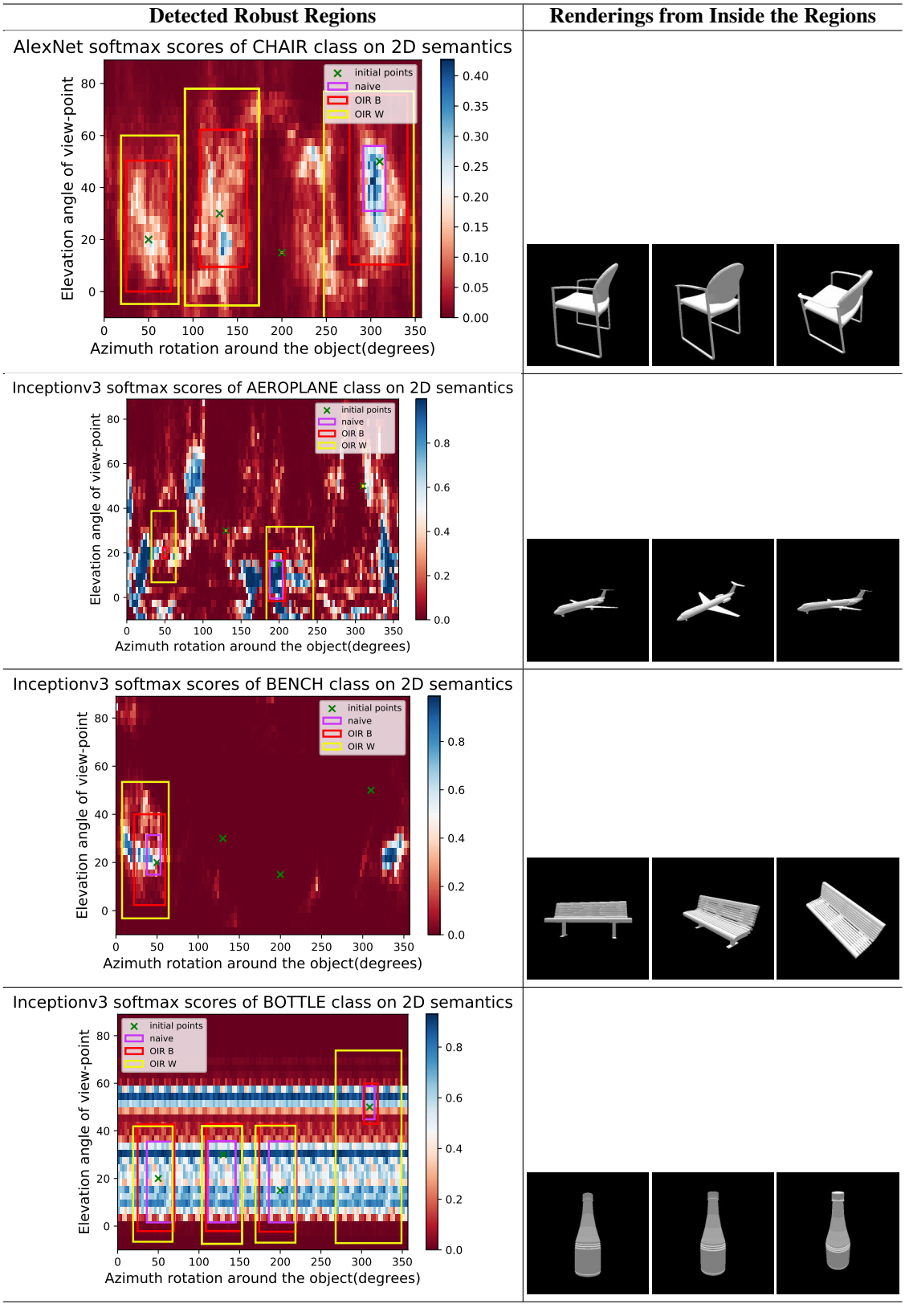


Figure 8: **Qualitative Examples of Robust Regions I**. Different runs of the algorithm to find robust regions along with different renderings from inside these regions for those specific shapes used in the experiments.

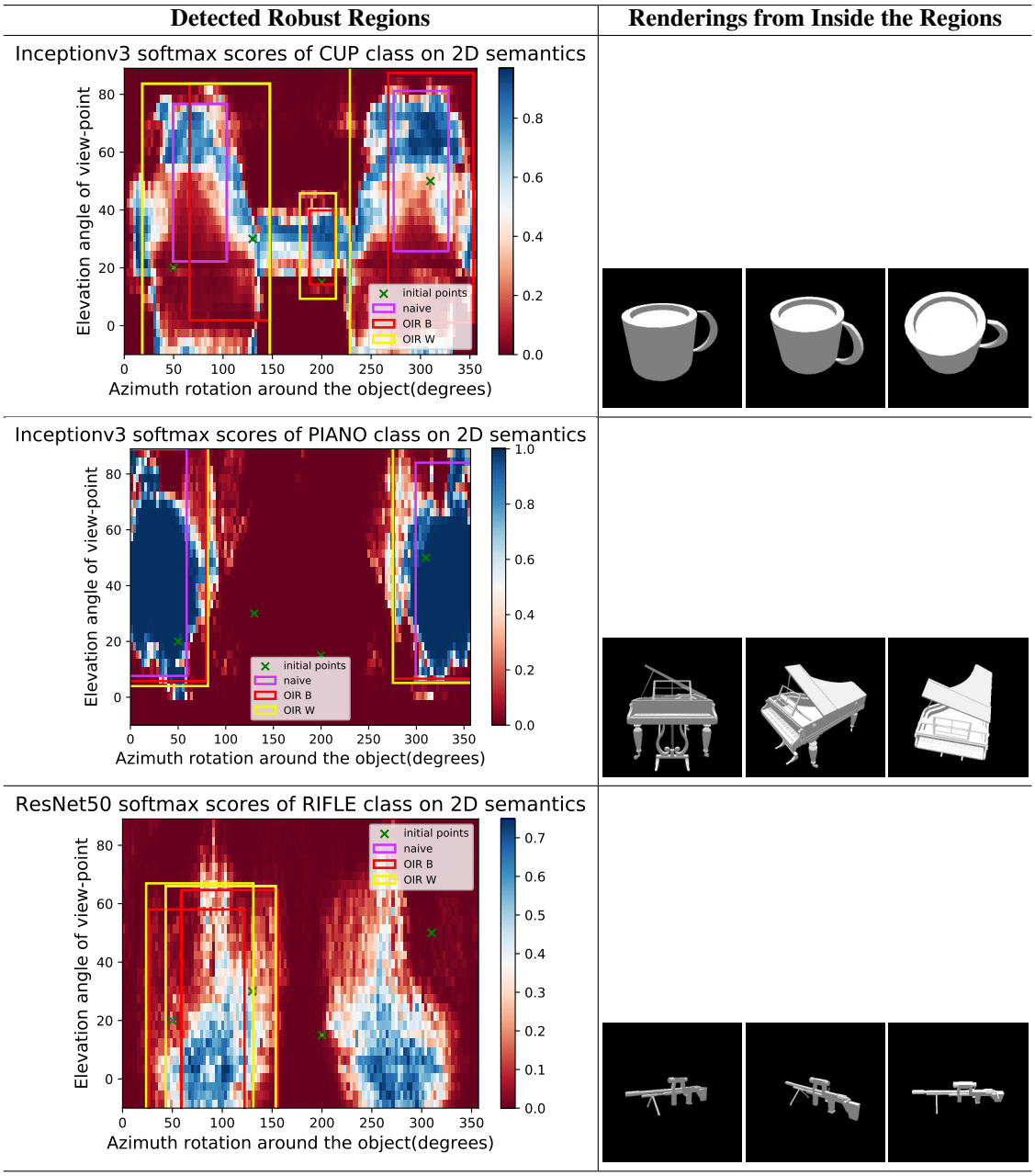


Figure 9: **Qualitative Examples of Robust Regions II** Different runs of the algorithm to find robust regions along with different renderings from inside these regions for those specific shapes used in the experiments.

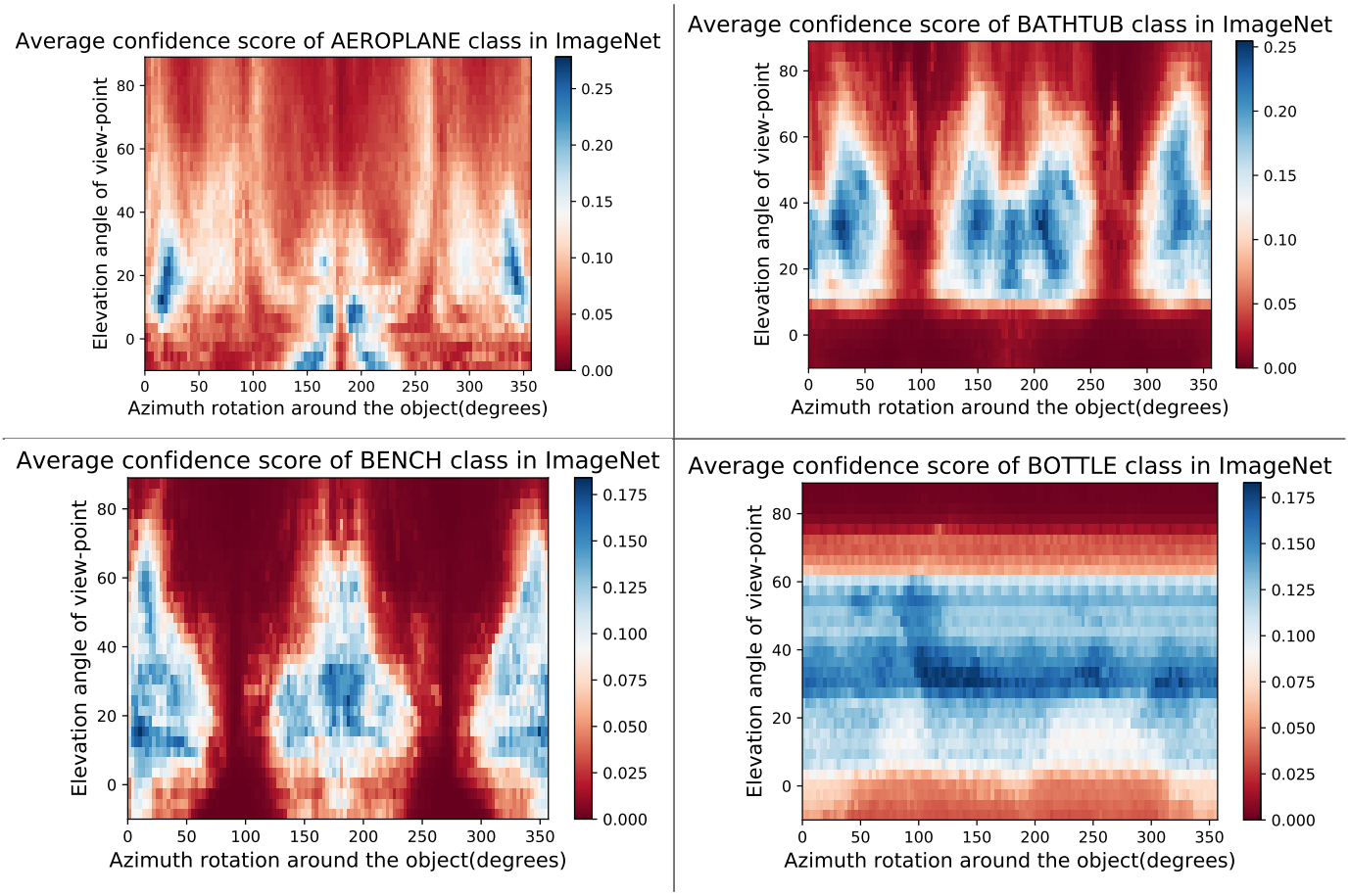


Figure 10: **Data Semantic Maps DSM-I**. Visualizing Semantic Data Bias in the common training dataset (*i.e.* ImageNet [6]) by averaging the Networks Semantic Maps (NSM) of different networks and on different shapes, Different classes have different semantic bias in ImageNet as clearly shown in the maps above. The symmetry in the maps are attributed to the 3D symmetry of the objects.

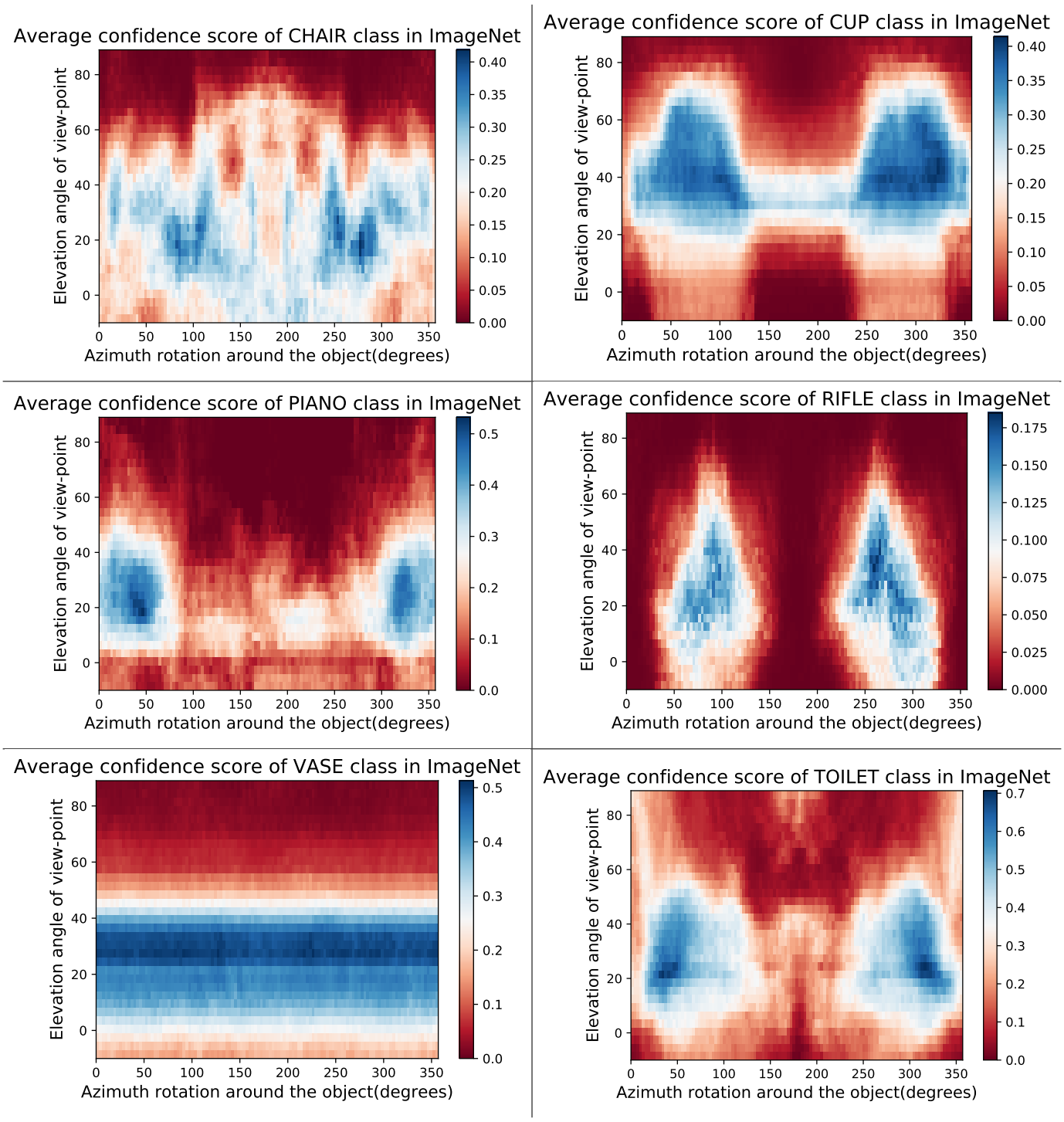


Figure 11: **Data Semantic Maps DSM-II**. Visualizing Semantic Data Bias in the common training dataset (*i.e.* ImageNet [6]) By averaging the Networks Semantic Maps (NSM) of different networks and on different shapes, Different classes have different semantic bias in ImageNet as clearly shown in the maps above. The symmetry in the maps are attributed to the 3D symmetry of the objects.

3. Analysis

3.1. Detected Robust Regions

We applies the algorithms on 2 semantic parameters (the azimuth angle of the camera around the object , and the elevation angle around the object) that are commonly used in the literature [4, 1]. When we use 1 parameter (the azimuth) , we fix the elevation to 35°. We used 2 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 1). The regions in Figure 8,9 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.

3.2. 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 detect 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 definitely depends on the full range of study . A rule of thump 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 thump 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 hyperparamter with mathematically established bound is the emphasis factor of the OIR_W formulation β. The bound shown in Table 1 which is $0 \leq \beta \leq \frac{2}{2n-1}$ can be shown as follows. We start from Eq (48). This 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 insure the correct direction of the move for a positive function evaluation.

$$\begin{aligned} \gamma_n &> 0 \\ 2-(2n-1)\beta &> 0 \\ \beta &< \frac{2}{2n-2} \end{aligned} \tag{65}$$

3.3. Detest

The data set used is collected from ShapeNet [3] 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 6,7. 4 shapes faced difficulty of rendering during the SRVR experiments , therefore , they were replaced by another shapes from the same class.

3.4. Possible Future Directions

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.

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 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 (58)

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

  form the all-corners function vectors $f_{\mathbb{D}}, f_{\mathbb{Q}}$ as in Eq (29)

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

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

  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 (58)

end

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

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

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 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 (58)

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

  form the all-corners function vector $f_{\mathbb{D}}$ as in Eq (29)

  form the all-corners gradients matrix $\mathbf{G}_{\mathbb{D}}$ as in Eq (63)

  form the gradient selection vectors $\mathbf{s}, \overline{\mathbf{s}}$ as in Eq (49)

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

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

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

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

  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 (58)

end

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

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. 20

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 1

[3] 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. 20

[4] A. Hamdi, M. Müller, and B. Ghanem. SADA: semantic adversarial diagnostic attacks for autonomous applications. *CoRR*, abs/1812.02132, 2018. 20

[5] 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. 3

[6] 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. 8, 18, 19

[7] 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. 1

[8] 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. 6

[9] 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. 11, 12