Certifiable Robustness

A. M. Sadeghzadeh, Ph.D.

Sharif University of Technology
Computer Engineering Department (CE)
Data and Network Security Lab (DNSL)

April 28, 2023

Today's Agenda

Recap

Obfuscated Gradients

A defense is said to cause gradient masking if it **does not have useful gradients** for generating adversarial examples.

- we refer to the case where defenses are designed in such a way that the constructed defense necessarily causes gradient masking as obfuscated gradients.

Obfuscated Gradients

A defense is said to cause gradient masking if it **does not have useful gradients** for generating adversarial examples.

- we refer to the case where defenses are designed in such a way that the constructed defense necessarily causes gradient masking as obfuscated gradients.

We discover three ways in which defenses obfuscate gradients

Obfuscated Gradients

A defense is said to cause gradient masking if it **does not have useful gradients** for generating adversarial examples.

- we refer to the case where defenses are designed in such a way that the constructed defense necessarily causes gradient masking as obfuscated gradients.

We discover three ways in which defenses obfuscate gradients

1. **Shattered Gradients**
   Shattered Gradients are caused when a defense is nondifferentiable, introduces numeric instability, or otherwise causes a **gradient to be nonexistent or incorrect**.

Obfuscated Gradients

A defense is said to cause gradient masking if it **does not have useful gradients** for generating adversarial examples.

- we refer to the case where defenses are designed in such a way that the constructed defense necessarily causes gradient masking as obfuscated gradients.

We discover three ways in which defenses obfuscate gradients

1. **Shattered Gradients**
   Shattered Gradients are caused when a defense is nondifferentiable, introduces numeric instability, or otherwise causes a **gradient to be nonexistent or incorrect**.
2. **Stochastic Gradients**
   Stochastic Gradients are caused by **randomized defenses**, where either the network itself is randomized or the input is randomly transformed before being fed to the classifier, causing the gradients to become randomized.

Obfuscated Gradients

A defense is said to cause gradient masking if it **does not have useful gradients** for generating adversarial examples.

- we refer to the case where defenses are designed in such a way that the constructed defense necessarily causes gradient masking as obfuscated gradients.

We discover three ways in which defenses obfuscate gradients

1. **Shattered Gradients**
   Shattered Gradients are caused when a defense is nondifferentiable, introduces numeric instability, or otherwise causes a **gradient to be nonexistent or incorrect**.
2. **Stochastic Gradients**
   Stochastic Gradients are caused by **randomized defenses**, where either the network itself is randomized or the input is randomly transformed before being fed to the classifier, causing the gradients to become randomized.
3. **Exploding & Vanishing Gradients**
   Exploding & Vanishing Gradients are often caused by defenses that consist of **multiple iterations of neural network** evaluation, feeding the output of one computation as the input of the next. This type of computation, when unrolled, can be viewed as an **extremely deep neural network evaluation**, which can cause vanishing/exploding gradients.

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

- **Backward Pass Differentiable Approximation (DBPA)** to overcome shattered gradients

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

- **Backward Pass Differentiable Approximation (DBPA)** to overcome shattered gradients
  - Given a pre-trained classifier $f(.)$, construct a preprocessor $g(.)$ and let the secured classifier $\hat{f}(x) = f(g(x))$ where the preprocessor $g(.)$ satisfies $g(x) \approx x$.

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

- **Backward Pass Differentiable Approximation (DBPA)** to overcome shattered gradients
  - Given a pre-trained classifier $f(.)$, construct a preprocessor $g(.)$ and let the secured classifier $\hat{f}(x) = f(g(x))$ where the preprocessor $g(.)$ satisfies $g(x) \approx x$.
  - Recent work has constructed functions $\boldsymbol{g}(.)$ **which are neither smooth nor differentiable**

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

- **Backward Pass Differentiable Approximation (DBPA)** to overcome shattered gradients
  - Given a pre-trained classifier $f(.)$, construct a preprocessor $g(.)$ and let the secured classifier $\hat{f}(x) = f(g(x))$ where the preprocessor $g(.)$ satisfies $g(x) \approx x$.
  - Recent work has constructed functions $\boldsymbol{g}(.)$ **which are neither smooth nor differentiable**
  - We perform **forward propagation** through the neural network **as usual**, but on the **backward pass**, we **replace $g(.)$ with the identity function**.

$$\nabla_x f(g(x))|_{x=\hat{x}} \approx \nabla_x f(x)|_{x=g(\hat{x})}$$

This allows us to compute gradients and therefore mount a white-box attack.

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

- **Backward Pass Differentiable Approximation (DBPA)** to overcome shattered gradients
  - Given a pre-trained classifier $f(.)$, construct a preprocessor $g(.)$ and let the secured classifier $\hat{f}(x) = f(g(x))$ where the preprocessor $g(.)$ satisfies $g(x) \approx x$.
  - Recent work has constructed functions $\boldsymbol{g}(.)$ **which are neither smooth nor differentiable**
  - We perform **forward propagation** through the neural network **as usual**, but on the **backward pass**, we **replace** $g(.)$ **with the identity function**.

$$\nabla_x f(g(x))|_{x=\hat{x}} \approx \nabla_x f(x)|_{x=g(\hat{x})}$$

  This allows us to compute gradients and therefore mount a white-box attack.
- **Expectation over Transformation** to overcome stochastic gradients

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

- **Backward Pass Differentiable Approximation (DBPA)** to overcome shattered gradients
  - Given a pre-trained classifier $f(.)$, construct a preprocessor $g(.)$ and let the secured classifier $\hat{f}(x) = f(g(x))$ where the preprocessor $g(.)$ satisfies $g(x) \approx x$.
  - Recent work has constructed functions $\boldsymbol{g}(.)$ **which are neither smooth nor differentiable**
  - We perform **forward propagation** through the neural network **as usual**, but on the **backward pass**, we **replace $g(.)$ with the identity function**.

$$\nabla_x f(g(x))|_{x=\hat{x}} \approx \nabla_x f(x)|_{x=g(\hat{x})}$$

    This allows us to compute gradients and therefore mount a white-box attack.
- **Expectation over Transformation** to overcome stochastic gradients
  - The optimization problem can be solved by gradient descent, noting that $\nabla \mathbb{E}_{t\sim T} f(t(x)) = \mathbb{E}_{t\sim T} \nabla f(t(x))$, differentiating through the classifier and transformation, and **approximating the expectation with samples** at each gradient descent step.

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

- **Backward Pass Differentiable Approximation (DBPA)** to overcome shattered gradients
  - Given a pre-trained classifier $f(.)$, construct a preprocessor $g(.)$ and let the secured classifier $\hat{f}(x) = f(g(x))$ where the preprocessor $g(.)$ satisfies $g(x) \approx x$.
  - Recent work has constructed functions $\boldsymbol{g}(.)$ **which are neither smooth nor differentiable**
  - We perform **forward propagation** through the neural network **as usual**, but on the **backward pass**, we **replace** $g(.)$ **with the identity function**.

$$\nabla_x f(g(x))|_{x=\hat{x}} \approx \nabla_x f(x)|_{x=g(\hat{x})}$$

    This allows us to compute gradients and therefore mount a white-box attack.
- **Expectation over Transformation** to overcome stochastic gradients
  - The optimization problem can be solved by gradient descent, noting that $\nabla \mathbb{E}_{t\sim T} f(t(x)) = \mathbb{E}_{t\sim T} \nabla f(t(x))$, differentiating through the classifier and transformation, and **approximating the expectation with samples** at each gradient descent step.
- **Reparameterization** to overcome exploding & vanishing gradients

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

- **Backward Pass Differentiable Approximation (DBPA)** to overcome shattered gradients
  - Given a pre-trained classifier $f(.)$, construct a preprocessor $g(.)$ and let the secured classifier $\hat{f}(x) = f(g(x))$ where the preprocessor $g(.)$ satisfies $g(x) \approx x$.
  - Recent work has constructed functions $\boldsymbol{g}(.)$ **which are neither smooth nor differentiable**
  - We perform **forward propagation** through the neural network **as usual**, but on the **backward pass**, we **replace $g(.)$ with the identity function**.

$$\nabla_x f(g(x))|_{x=\hat{x}} \approx \nabla_x f(x)|_{x=g(\hat{x})}$$

    This allows us to compute gradients and therefore mount a white-box attack.
- **Expectation over Transformation** to overcome stochastic gradients
  - The optimization problem can be solved by gradient descent, noting that $\nabla \mathbb{E}_{t\sim T} f(t(x)) = \mathbb{E}_{t\sim T} \nabla f(t(x))$, differentiating through the classifier and transformation, and **approximating the expectation with samples** at each gradient descent step.
- **Reparameterization** to overcome exploding & vanishing gradients
  - Assume we are given a classifier $f(g(x))$ where $g(.)$ performs some optimization loop to transform the input x to a new input $\hat{x}$.

Attack Techniques

There is a number of techniques to overcome obfuscated gradients

- **Backward Pass Differentiable Approximation (DBPA)** to overcome shattered gradients
  - Given a pre-trained classifier $f(.)$, construct a preprocessor $g(.)$ and let the secured classifier $\hat{f}(x) = f(g(x))$ where the preprocessor $g(.)$ satisfies $g(x) \approx x$.
  - Recent work has constructed functions $\boldsymbol{g}(.)$ **which are neither smooth nor differentiable**
  - We perform **forward propagation** through the neural network **as usual**, but on the **backward pass**, we **replace $g(.)$ with the identity function**.

$$\nabla_x f(g(x))|_{x=\hat{x}} \approx \nabla_x f(x)|_{x=g(\hat{x})}$$

  This allows us to compute gradients and therefore mount a white-box attack.
- **Expectation over Transformation** to overcome stochastic gradients
  - The optimization problem can be solved by gradient descent, noting that $\nabla \mathbb{E}_{t\sim T} f(t(x)) = \mathbb{E}_{t\sim T} \nabla f(t(x))$, differentiating through the classifier and transformation, and **approximating the expectation with samples** at each gradient descent step.
- **Reparameterization** to overcome exploding & vanishing gradients
  - Assume we are given a classifier $f(g(x))$ where $g(.)$ performs some optimization loop to transform the input x to a new input $\hat{x}$.
  - We make a **change-of-variable** $x = h(z)$ for some function $h(.)$ such that $g(h(z)) = h(z)$ for all z, but $h(.)$ is differentiable. This allows us to **compute gradients through** $f(h(z))$ and thereby circumvent the defense.

Case Study: ICLR 2018 Defenses

As a case study for evaluating the prevalence of obfuscated gradients, we study the **ICLR 2018** non-certified defenses that argue robustness in a white-box threat model.

- To show a defense can be bypassed, it is only necessary to demonstrate one way to do so; in contrast, a defender must show no attack can succeed.

Case Study: ICLR 2018 Defenses

As a case study for evaluating the prevalence of obfuscated gradients, we study the **ICLR 2018** non-certified defenses that argue robustness in a white-box threat model.

- To show a defense can be bypassed, it is only necessary to demonstrate one way to do so; in contrast, a defender must show no attack can succeed.
- **Of the 9 accepted papers, 7 rely on obfuscated gradients.**

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%$*$
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.005 (ℓ_2)	0%$*$
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%$*$
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%$*$
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	55%$**$
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%
Na et al. (2018)	CIFAR	0.015 (ℓ_∞)	15%

Table 1. **Summary of Results:** Seven of nine defense techniques accepted at ICLR 2018 cause obfuscated gradients and are vulnerable to our attacks. Defenses denoted with $*$ propose combining adversarial training; we report here the defense alone, see §5 for full numbers. The fundamental principle behind the defense denoted with $**$ has 0% accuracy; in practice, imperfections cause the theoretically optimal attack to fail, see §5.4.2 for details.

Circumventing Defenses to Adversarial Examples

Adversarial Training

Adversarial training does not cause obfuscated gradients and it **passes all tests** for characteristic behaviors of obfuscated gradients that we list. However it has some limitation

- Adversarial retraining has been shown to be **difficult at ImageNet scale**
- Training exclusively on ℓ_∞ adversarial examples provides only **limited robustness** to adversarial examples under **other distortion metrics**

Thermometer Encoding

Defense Details (Buckman et al. (2018))

- Given an image x, for each pixel color $x_{i,j,c}$, the l-level thermometer encoding $\tau(x_{i,j,c})$ is a l-dimensional vector where $\tau(x_{i,j,c})_k = 1$ if $x_{i,j,c} > \frac{k}{l}$, and 0 otherwise.
  - For a 10-level thermometer encoding, $\tau(0.66) = 1111110000$.

Thermometer Encoding

Defense Details (Buckman et al. (2018))

- Given an image x, for each pixel color $x_{i,j,c}$, the l-level thermometer encoding $\tau(x_{i,j,c})$ is a l-dimensional vector where $\tau(x_{i,j,c})_k = 1$ if $x_{i,j,c} > \frac{k}{l}$, and 0 otherwise.
  - For a 10-level thermometer encoding, $\tau(0.66) = 1111110000$.

Discussion

- This defense causes **gradient shattering**
- This can be observed through their **black-box attack evaluation**
  - Adversarial examples generated on a standard adversarially trained model transfer to a thermometer encoded model reducing the accuracy to 67%, well below the 80% robustness to the white-box iterative attack.

Thermometer Encoding

Defense Details (Buckman et al. (2018))

- Given an image x, for each pixel color $x_{i,j,c}$, the l-level thermometer encoding $\tau(x_{i,j,c})$ is a l-dimensional vector where $\tau(x_{i,j,c})_k = 1$ if $x_{i,j,c} > \frac{k}{l}$, and 0 otherwise.
    - For a 10-level thermometer encoding, $\tau(0.66) = 1111110000$.

Discussion

- This defense causes **gradient shattering**
- This can be observed through their **black-box attack evaluation**
    - Adversarial examples generated on a standard adversarially trained model transfer to a thermometer encoded model reducing the accuracy to 67%, well below the 80% robustness to the white-box iterative attack.

Evaluation

- We use the **BPDA** approach where we let $f(x) = \tau(x)$. Observe that if we define

$$\hat{\tau}(x_{i,j,c})_k = min(max(\frac{x_{i,j,c}}{k/l}, 0), 1)$$

then

$$\tau(x_{i,j,c})_k = floor(\hat{\tau}(x_{i,j,c})_k)$$

So we can let $g(x) = \hat{\tau}(x)$ and replace the backwards pass with the function $g(.)$.

Thermometer Encoding Evaluation

Accuracy of various models on adversarial examples.

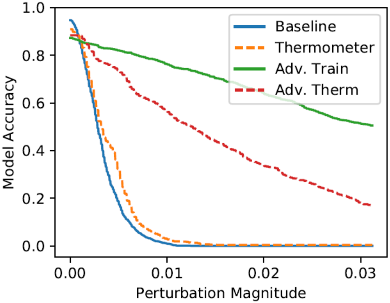


Figure 1. Model accuracy versus distortion (under ℓ_∞). Adversarial training increases robustness to 50% at $\epsilon = 0.031$; thermometer encoding by itself provides limited value, and when coupled with adversarial training performs worse than adversarial training alone.

Input Transformation

Guo et al. (2018) propose five input transformations to counter adversarial examples.

1. Image cropping and rescaling
2. Bit-depth reduction
3. JPEG compression
4. Total variance minimization
5. Image quilting (reconstruct images by replacing small patches with patches from "clean" images)

Input Transformation

Guo et al. (2018) propose five input transformations to counter adversarial examples.

1. Image cropping and rescaling
2. Bit-depth reduction
3. JPEG compression
4. Total variance minimization
5. Image quilting (reconstruct images by replacing small patches with patches from "clean" images)

To overcome transformations

- We circumvent **image cropping and rescaling** with a direct application of **EOT**.
- To circumvent **bit-depth reduction and JPEG compression**, we use BPDA and approximate the **backward pass with the identity function**.
- To circumvent **total variance minimization and image quilting**, which are both non-differentiable and randomized, we apply **EOT** and use **BPDA** to approximate the gradient through the transformation.

Stochastic Gradients

Defenses

- **Stochastic Activation Pruning (SAP)**
  - SAP (Dhillon et al., 2018) **randomly drops some neurons** of each layer f^i to 0 with probability proportional to their absolute value.

Stochastic Gradients

Defenses

- **Stochastic Activation Pruning (SAP)**
  - SAP (Dhillon et al., 2018) **randomly drops some neurons** of each layer f^i to 0 with probability proportional to their absolute value.
- **Mitigating Through Randomization**
  - Xie et al. (2018) propose to defend against adversarial examples by adding a randomization layer before the input to the classifier.
    - It **randomly rescales** the image and add zero-pad to it

Stochastic Gradients

Defenses

- **Stochastic Activation Pruning (SAP)**
  - SAP (Dhillon et al., 2018) **randomly drops some neurons** of each layer f^i to 0 with probability proportional to their absolute value.
- **Mitigating Through Randomization**
  - Xie et al. (2018) propose to defend against adversarial examples by adding a randomization layer before the input to the classifier.
    - It **randomly rescales** the image and add zero-pad to it

To circumventing both defenses, we estimate the gradients by computing the expectation over instantiations of randomness.

- At each iteration of gradient descent, instead of taking a step in the direction of $\nabla_x f(x)$ we move in the direction of $\sum_{i=1}^{k} \nabla_x f(x)$.

PIXELDEFEND (Exploding and Vanishing Gradient)

Song et al. (2018) propose using a PixelCNN generative model to project a potential adversarial example back onto the data manifold before feeding it into a classifier.

- PixelDefend **purifies** adversarially perturbed images prior to classification by using a greedy decoding procedure to approximate finding the highest probability example within an ϵ-ball of the input image.

PIXELDEFEND (Exploding and Vanishing Gradient)

Song et al. (2018) propose using a PixelCNN generative model to project a potential adversarial example back onto the data manifold before feeding it into a classifier.

- PixelDefend **purifies** adversarially perturbed images prior to classification by using a greedy decoding procedure to approximate finding the highest probability example within an ϵ-ball of the input image.

To circumventing PIXELDEFEND

- We sidestep the problem of computing gradients through an unrolled version of PixelDefend by approximating gradients with BPDA (**approximate PixelCNN derivative as the derivative of the identity function**).

Randomized Smoothing

Randomized Smoothing

Certified Adversarial Robustness via Randomized Smoothing

Jeremy Cohen [1] **Elan Rosenfeld** [1] **J. Zico Kolter** [1 2]

Certifiable Robustness

A classifier is **certifiably robust** if for any input x, **there exist a guarantee** that the **classifier's prediction is constant within some set around** x, often an ℓ_2 or ℓ_∞ ball.

Certifiable Robustness

A classifier is **certifiably robust** if for any input x, **there exist a guarantee** that the **classifier's prediction is constant within some set around** x, often an ℓ_2 or ℓ_∞ ball.

- Classifier $f : \mathbb{R}^d \to [0,1]^K$ is **ϵ-robust** at x, if

$$\forall \ \|\delta\|_p \leq \epsilon, \quad \underset{i \in [K]}{argmax} \ f_i(x+\delta) = \underset{i \in [K]}{argmax} \ f_i(x)$$

where K is the number of classes.

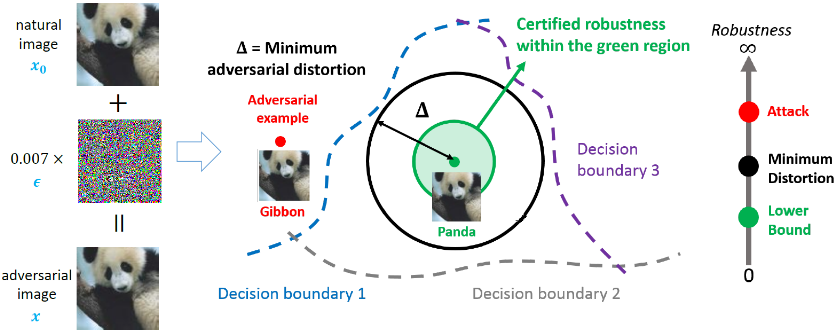
Certifiable Robustness

A classifier is **certifiably robust** if for any input x, **there exist a guarantee** that the **classifier's prediction is constant within some set around** x, often an ℓ_2 or ℓ_∞ ball.

- Classifier $f : \mathbb{R}^d \to [0,1]^K$ is **ϵ-robust** at x, if

$$\forall \ \|\delta\|_p \leq \epsilon, \quad \underset{i\in[K]}{argmax}\ f_i(x+\delta) = \underset{i\in[K]}{argmax}\ f_i(x)$$

where K is the number of classes.



Lipschitz continuity

A function $f : I \rightarrow R$ over some set $I \subseteq \mathbb{R}^d$ is called Lipschitz continuous if there exists a positive real constant L such that, for all $x, y \in I$,

$$|f(y) - f(x)| \leq L\|y - x\|_2$$

or

$$f(x) - L\|y - x\|_2 \leq f(y) \leq f(x) + L\|y - x\|_2$$

We call L the Lipschitz constant of f over I.

Lipschitz continuity

A function $f : I \to R$ over some set $I \subseteq \mathbb{R}^d$ is called Lipschitz continuous if there exists a positive real constant L such that, for all $x, y \in I$,

$$|f(y) - f(x)| \leq L\|y - x\|_2$$

or

$$f(x) - L\|y - x\|_2 \leq f(y) \leq f(x) + L\|y - x\|_2$$

We call L the Lipschitz constant of f over I.

Let functions f_1 and f_2 be both Lipschitz continuous with constants L_1 and L_2, the upper Lipschitz constant of their composition $f_1 \circ f_2$ is $L_1 L_2$.

Lipschitz continuity

A function $f : I \to R$ over some set $I \subseteq \mathbb{R}^d$ is called Lipschitz continuous if there exists a positive real constant L such that, for all $x, y \in I$,

$$|f(y) - f(x)| \leq L\|y - x\|_2$$

or

$$f(x) - L\|y - x\|_2 \leq f(y) \leq f(x) + L\|y - x\|_2$$

We call L the Lipschitz constant of f over I.

Let functions f_1 and f_2 be both Lipschitz continuous with constants L_1 and L_2, the upper Lipschitz constant of their composition $f_1 \circ f_2$ is $L_1 L_2$.

$$|f_1(f_2(y)) - f_1(f_2(x))| \leq L_1|f_2(y) - f_2(x)| \leq L_1 L_2\|y - x\|_2$$

Lipschitz continuity

A function $f : I \rightarrow R$ over some set $I \subseteq \mathbb{R}^d$ is called Lipschitz continuous if there exists a positive real constant L such that, for all $x, y \in I$,

$$|f(y) - f(x)| \leq L\|y - x\|_2$$

or

$$f(x) - L\|y - x\|_2 \leq f(y) \leq f(x) + L\|y - x\|_2$$

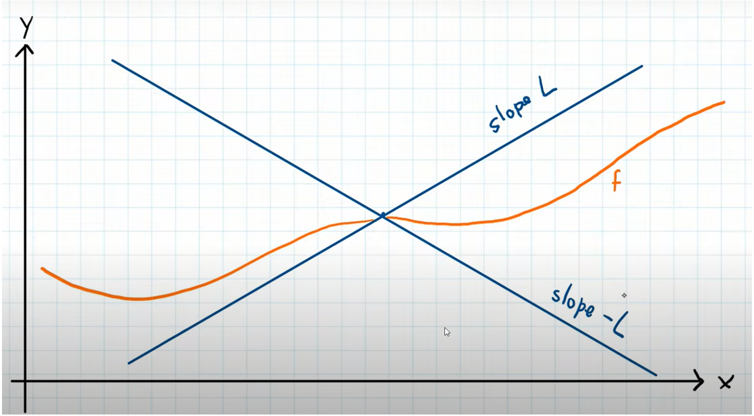
We call L the Lipschitz constant of f over I.

Let functions f_1 and f_2 be both Lipschitz continuous with constants L_1 and L_2, the upper Lipschitz constant of their composition $f_1 \circ f_2$ is $L_1 L_2$.

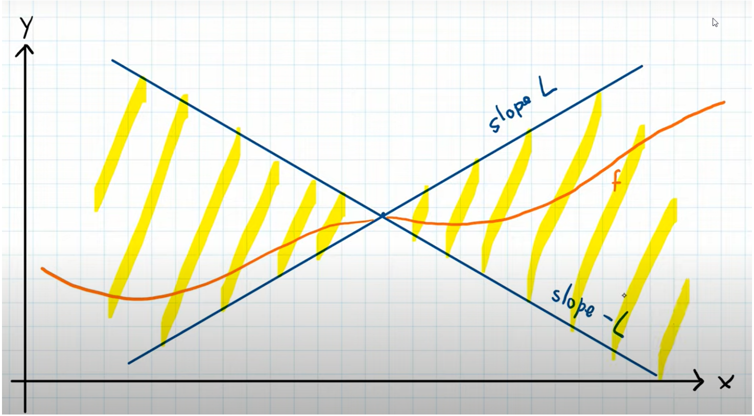
$$|f_1(f_2(y)) - f_1(f_2(x))| \leq L_1|f_2(y) - f_2(x)| \leq L_1 L_2\|y - x\|_2$$

Generally, Let $f = f_1 \circ f_2 \circ .... \circ f_K$ and the Lipschitz constant of f_i be L_i for all $i \in \{1, 2, ..., K\}$, then the Lipschitz constant of f is $L \leq \prod_{k=1}^{K} L_k$.

Lipschitz continuity



Lipschitz continuity



Lipschitz continuity

Let $f : I \rightarrow R$ be a continuous and differntiable function over some set $I \subseteq \mathbb{R}^d$, if we have $\|f'(x)\|_2 \leq m$ for all $x \in I$, then m is the upper Lipschitz constant of f ($L \leq m$).

Lipschitz continuity

Let $f : I \to R$ be a continuous and differntiable function over some set $I \subseteq \mathbb{R}^d$, if we have $\|f'(x)\|_2 \leq m$ for all $x \in I$, then m is the upper Lipschitz constant of f $(L \leq m)$.

Proof sketch:
Mean value theorem: Let $f : I \to R$ be a continuous and differntiable function over some set $I \subseteq \mathbb{R}^d$, For all $a, b \in I$ $(b > a)$, there exists some $c \in (a, b)$ such that:

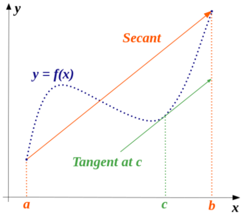
$$f'(c) = \frac{f(b) - f(a)}{b - a}$$

For all $a, b \in I$, there exist $c \in (a, b)$, such that:

$$|f(b) - f(a)| = \|f'(c).b - a\|_2 \leq \|f'(c)\|_2 \|b - a\|_2.$$

Since we know that $\|f'(c)\|_2 \leq m$, we have

$$|f(b) - f(a)| \leq m\|b - a\|_2.$$



Certifiable Robustness for L-Lipschitz classifier

Theorem 0: If $f : \mathbb{R}^d \to [0,1]^K$ is L-lipschitz, then f is ϵ-robust at x with $\epsilon = \frac{1}{2L}(P_A - P_B)$, where $P_A = \underset{i}{max}\ f_i(x)$, $P_B = \underset{j \neq i}{max}\ f_j(x)$, and $f_k(x)$ is the k-th element of the probability vector $f(x)$.

Certifiable Robustness for L-Lipschitz classifier

Theorem 0: If $f : \mathbb{R}^d \to [0,1]^K$ is L-lipschitz, then f is ϵ-robust at x with $\epsilon = \frac{1}{2L}(P_A - P_B)$, where $P_A = \max_i f_i(x)$, $P_B = \max_{j \neq i} f_j(x)$, and $f_k(x)$ is the k-th element of the probability vector $f(x)$.

Proof.

Since f is L-lipschitz, we have

$$\forall x, y \in \mathbb{R}^d, \|f(y) - f(x)\|_2 \leq L\|y - x\|_2$$

Denote $x' = x + \delta$ and assume that $\|\delta\| \leq \epsilon$. we get

$$\|f(x') - f(x)\|_2 \leq L\|x' - x\|_2 \rightarrow \|f(x') - f(x)\|_2 \leq L\|\delta\|_2 \leq L\epsilon$$

Hence, P_A can be reduced at most by $L\epsilon$ and P_B can be increased at most by $L\epsilon$. We have ($P_i' = f_i(x')$)

$$P_A' \geq P_A - L\epsilon \quad \text{and} \quad P_B' \leq P_B + L\epsilon$$

Since we want that the label of x' be the same as x, P_A' must be greater than P_B' ($P_A' \geq P_B'$). We have

$$P_A - L\epsilon \geq P_B + L\epsilon \ \rightarrow \ 2L\epsilon \leq P_A - P_B \ \rightarrow \ \epsilon \leq \frac{1}{2L}(P_A - P_B)$$

$\square$

Smoothed Classifier

If we compute the (upper bound of) Lipschitz constant of the classifier, we can determine the radius (ϵ) of the robustness for each sample.

- However, **the lipschitz constant of deep neural networks** is very large and it **grows exponentially** with the number of layers. Hence, the certification is useless ($\epsilon \rightarrow 0$).

Smoothed Classifier

If we compute the (upper bound of) Lipschitz constant of the classifier, we can determine the radius (ϵ) of the robustness for each sample.

- However, **the lipschitz constant of deep neural networks** is very large and it **grows exponentially** with the number of layers. Hence, the certification is useless ($\epsilon \longrightarrow 0$).

We can **transform classifier f to a smoothed version** in order to bound the Lipschitz constant of the classifier.

Smoothed Classifier

If we compute the (upper bound of) Lipschitz constant of the classifier, we can determine the radius (ϵ) of the robustness for each sample.

- However, **the lipschitz constant of deep neural networks** is very large and it **grows exponentially** with the number of layers. Hence, the certification is useless ($\epsilon \longrightarrow 0$).

We can **transform classifier f to a smoothed version** in order to bound the Lipschitz constant of the classifier.

- To smooth classifier f, we convolve it with a **Gaussian kernel**.

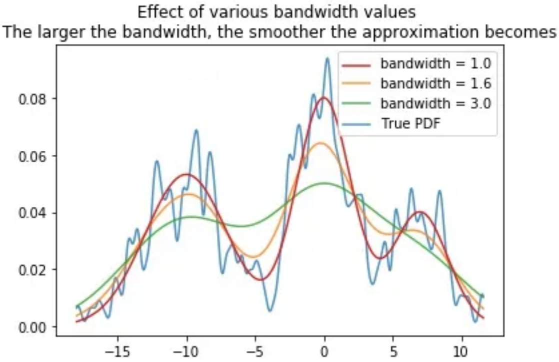
Smoothed Classifier

If we compute the (upper bound of) Lipschitz constant of the classifier, we can determine the radius (ϵ) of the robustness for each sample.

- However, **the lipschitz constant of deep neural networks** is very large and it **grows exponentially** with the number of layers. Hence, the certification is useless ($\epsilon \to 0$).

We can **transform classifier f to a smoothed version** in order to bound the Lipschitz constant of the classifier.

- To smooth classifier f, we convolve it with a **Gaussian kernel**.



[source]

Randomized Smoothing

Consider a classification problem from $\mathbb{R}^d$ to classes $\mathcal{Y}$. Randomized smoothing is a method for constructing a new, **smoothed classifier** $\hat{f}$ from an arbitrary base classifier f.

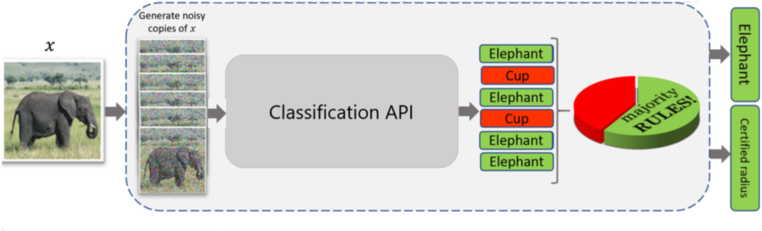
Randomized Smoothing

Consider a classification problem from $\mathbb{R}^d$ to classes $\mathcal{Y}$. Randomized smoothing is a method for constructing a new, **smoothed classifier** $\hat{f}$ from an arbitrary base classifier f.

- When queried at x, the smoothed classifier $\hat{f}$ returns whichever class the base classifier f is most likely to return when x is perturbed by isotropic Gaussian noise:

$$\begin{aligned} \hat{f}(x) &= \underset{c \in \mathcal{Y}}{argmax}\ \mathbb{P}(f(x+\epsilon) = c) \\ \text{where} \quad & \epsilon \sim \mathcal{N}(0, \sigma^2 I) \end{aligned} \tag{1}$$

The noise level σ is a hyperparameter of the smoothed classifier $\hat{f}$ which controls a robustness/accuracy tradeoff.



[source]

Notation

Suppose that when the base classifier f classifies $\mathcal{N}(x, \sigma^2 I)$, the most probable class c_A is returned with probability P_A, and the "runner-up" class is returned with probability P_B.

- $\underline{P_A}$ is a lower bound for P_A and $\overline{P_B}$ is a lower bound for P_B.

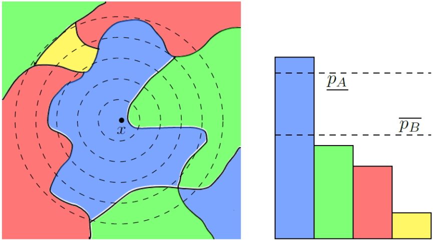


Figure 1. Evaluating the smoothed classifier at an input x. **Left**: the decision regions of the base classifier f are drawn in different colors. The dotted lines are the level sets of the distribution $\mathcal{N}(x, \sigma^2 I)$. **Right**: the distribution $f(\mathcal{N}(x, \sigma^2 I))$. As discussed below, $\underline{p_A}$ is a lower bound on the probability of the top class and $\overline{p_B}$ is an upper bound on the probability of each other class. Here, $g(x)$ is "blue."

Robustness guarantee

Theorem 1. Let $f : \mathbb{R}^d \to \mathcal{Y}$ be any deterministic or random function, and let $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Let $\hat{f}$ be defined as in (1). Suppose $C_A \in \mathcal{Y}$ and $\underline{P_A}, \overline{P_B} \in [0, 1]$ satisfy:

$$\mathbb{P}(f(x+\epsilon) = C_A) \geq \underline{P_A} \geq \overline{P_B} \geq \max_{C \neq C_A} \mathbb{P}(f(x+\epsilon) = C)$$

Then $\hat{f}(x+\delta) = C_A$ for all $\|\delta\|_2 \leq R$, where

$$R = \frac{\sigma}{2}(\Phi^{-1}(\underline{P_A}) - \Phi^{-1}(\overline{P_B}))$$

where Φ^{-1} is the inverse of the standard Gaussian CDF.

Robustness guarantee

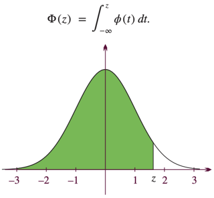
Theorem 1. Let $f : \mathbb{R}^d \to \mathcal{Y}$ be any deterministic or random function, and let $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Let $\hat{f}$ be defined as in (1). Suppose $C_A \in \mathcal{Y}$ and $\underline{P_A}, \overline{P_B} \in [0, 1]$ satisfy:

$$\mathbb{P}(f(x+\epsilon) = C_A) \geq \underline{P_A} \geq \overline{P_B} \geq \max_{C \neq C_A} \mathbb{P}(f(x+\epsilon) = C)$$

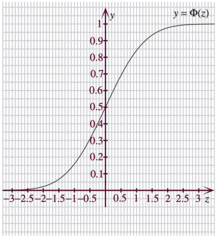
Then $\hat{f}(x+\delta) = C_A$ for all $\|\delta\|_2 \leq R$, where

$$R = \frac{\sigma}{2}(\Phi^{-1}(\underline{P_A}) - \Phi^{-1}(\overline{P_B}))$$

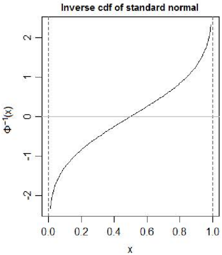
where Φ^{-1} is the inverse of the standard Gaussian CDF.



This is the graph of the standard normal probability density function $\phi(z)$.



This is the graph of the standard normal cumulative distribution function $\Phi(z)$.



Several Observations About Theorem 1

Several observations about theorem 1

- **Theorem 1 assumes nothing about f**. This is crucial since it is unclear which well-behavedness assumptions, if any, are satisfied by modern deep architectures.

Several Observations About Theorem 1

Several observations about theorem 1

- **Theorem 1 assumes nothing about f**. This is crucial since it is unclear which well-behavedness assumptions, if any, are satisfied by modern deep architectures.
- The certified radius R is large when: (1) the noise level σ is high, (2) the probability of the top class C_A is high, and (3) the probability of each other class is low.

Several Observations About Theorem 1

Several observations about theorem 1

- **Theorem 1 assumes nothing about f**. This is crucial since it is unclear which well-behavedness assumptions, if any, are satisfied by modern deep architectures.
- The certified radius R is large when: (1) the noise level σ is high, (2) the probability of the top class C_A is high, and (3) the probability of each other class is low.
- **The certified radius R goes to ∞ as $\underline{P_A} \to 1$ and $\overline{P_B} \to 0$.** This should sound reasonable: the Gaussian distributi is supported on all of $\mathbb{R}^d$, so the only way that $f(x + \epsilon) = C_A$ with probability 1 is if $f = C_A$ almost everywhere.