

# Artificial Intelligence III: Advanced Deep Learning Methods

**Ana Neacșu & Vlad Vasilescu**

**Chapter 2: Defense strategies**

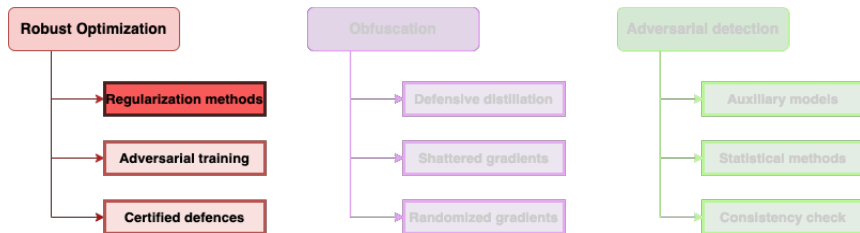
National University of Science and Technology POLITEHNICA Bucharest, Romania  
BIOSINF Master Program

October 2024

# Introduction

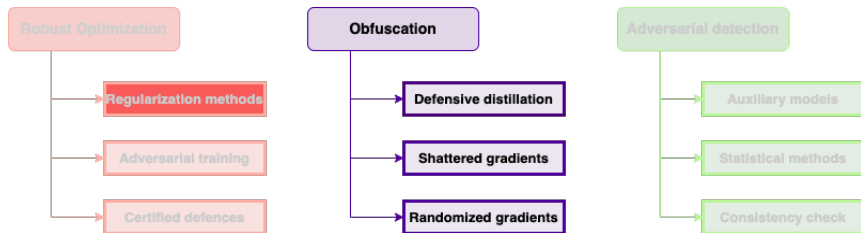
# Defense strategies

*Three* main methodologies to train models that are robust against adversarial perturbations.



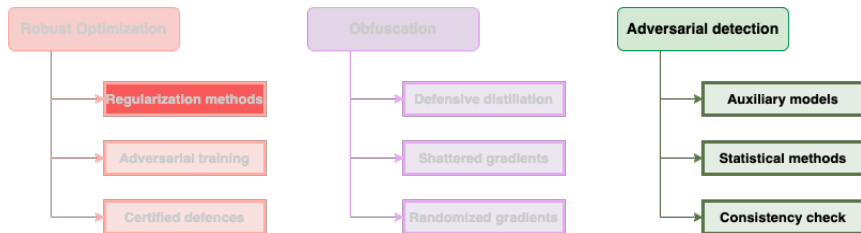
# Defense strategies

*Three main methodologies to train models that are robust against adversarial perturbations.*



# Defense strategies

*Three main methodologies to train models that are robust against adversarial perturbations.*



# Robustness quantification

## Definition

Let  $T$  be the learnt system and  $x$  the clean sample  $\rightarrow T(x) = y_{\text{true}}$   
 $\tilde{x} = (x + z)$  – adversarial sample  $\rightarrow T(\tilde{x}) \neq y_{\text{true}}$

**Lipschitz** constant  $\theta \geq 0$  used to assess the robustness of neural network to adversarial inputs:

$$\|T(x + z) - T(x)\| \leq \theta \|z\|$$

The **Lipschitz constant** provides an upper bound on the output perturbation knowing the magnitude of the input one, for a given metric.

Lipschitz constant of a differentiable function:  $\theta = \sup_x \|\nabla T(x)\|$

where  $\|\nabla T(x)\|$  is the **spectral norm** (maximum singular value) of the Jacobian of  $T$  at  $x$ .

# Robustness quantification

## Definition

Let  $T$  be the learnt system and  $x$  the clean sample  $\rightarrow T(x) = y_{\text{true}}$   
 $\tilde{x} = (x + z)$  – adversarial sample  $\rightarrow T(\tilde{x}) \neq y_{\text{true}}$

**Lipschitz** constant  $\theta \geq 0$  used to assess the robustness of neural network to adversarial inputs:

$$\|T(x + z) - T(x)\| \leq \theta \|z\|$$

The **Lipschitz constant** provides an upper bound on the output perturbation knowing the magnitude of the input one, for a given metric.

Lipschitz constant of a differentiable function:  $\theta = \sup_x \|\nabla T(x)\|$

where  $\|\nabla T(x)\|$  is the **spectral norm** (maximum singular value) of the Jacobian of  $T$  at  $x$ .

# Introduction to Lipschitz Continuity

- Lipschitz continuity is a strong form of continuity. A function is Lipschitz continuous if the rate at which the function's value changes is bounded.
- This means the function's output doesn't change too rapidly relative to changes in its input.
- Formally, it's about a constraint on the slope of the function.

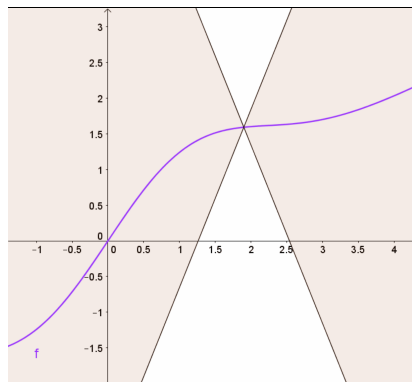


Figure: Example of a Lipschitz continuous function



# Formal Definition

## Definition 1. Lipschitz continuity

A function  $f$  from  $\mathcal{S} \subset \mathbb{R}^n$  into  $\mathbb{R}^m$  is Lipschitz continuous at a point  $x \in \mathcal{S}$  if there is a constant  $\theta$  such that for all  $y \in \mathcal{S}$  near  $x$  the following inequality holds:

$$\|f(x) - f(y)\| \leq \theta \|x - y\| \quad (1)$$

The constant  $\theta$  is called the Lipschitz constant. Geometrically, this means that the slope of any secant line is bounded by  $\theta$ .

# Examples and Implications

## Example 1 (Lipschitz)

$f(x) = 2x + 1$  on  $\mathbb{R}$ . The Lipschitz constant is  $\theta = 2$ .

## Example 2 (Not Lipschitz):

$f(x) = \sqrt{x}$  on  $[0, 1]$ . The slope approaches infinity as  $x$  approaches 0.

**Implications:** Lipschitz continuity guarantees the existence and uniqueness of solutions to certain differential equations and is important in various areas of analysis and numerical methods. It's stronger than uniform continuity.

differentiable at  $x \Rightarrow$  Lipschitz continuous at  $x \Rightarrow$  continuous at  $x$

## Naive defenses

# Pre-processing as a defense

**Idea :** Can some pre-processing techniques be used as a defense?

Consider  $x$  the perturbed data, and a classification system  $f$ , that is fooled by the corrupted input. Can we find a pre-processing operator  $g(x)$  to prevent that?

The problem can be viewed from many perspectives

- Denoising problem
- Geometry projection

# Pre-processing as a defense

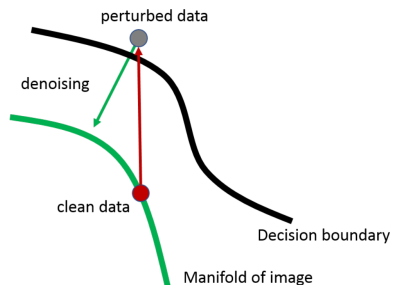
**Idea :** Can some pre-processing techniques be used as a defense?

Consider  $x$  the perturbed data, and a classification system  $f$ , that is fooled by the corrupted input. Can we find a pre-processing operator  $g(x)$  to prevent that?

The problem can be viewed from many perspectives

- Denoising problem
- Geometry projection

# Visualization



# Input transform

- Countering Adversarial Images using Input Transformations use *Total variance minimization*.

## Objective

$$\underset{z}{\text{minimize}} \|(1 - X) \odot (z - x)\|_2 + \lambda \text{TV}(z), \quad (2)$$

where

- TV – Total Variation
- $\text{TV}(z) = \|D_1(z)\|_1 + \|D_2(z)\|_1$
- $\lambda \in ]0, 1[$
- $\odot$  is point-wise multiplication

# Some examples

- Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser

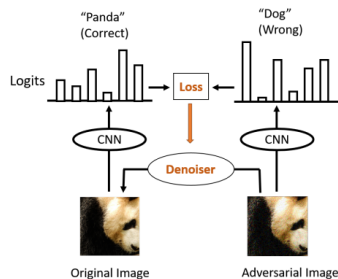


Figure 1: The idea of high-level representation guided denoiser. The difference between the original image and adversarial image is tiny, but the difference is amplified in high-level representation (logits for example) of a CNN. We use the distance over high-level representations to guide the training of an image denoiser to suppress the influence of adversarial perturbation.



# Use random transform

**Idea :** Since attack is so specific along one direction, we could use a random perturbation to change its path.

**Observation:** In a high dimensional space, a small change in *direction* can have a huge impact in the *destination*.

Some examples:

- Use a suite of classifiers, so the attacker cannot know the one that is used.
- Randomly move around pixels so that the attack is distorted
- Randomly mask out some pixels to defend against the attack and then solve a matrix completion problem.

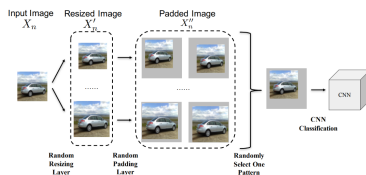


Figure: <https://arxiv.org/pdf/1711.01991>

# Pixel Deflection

## Paper: Deflecting Adversarial Attacks with Pixel Deflection

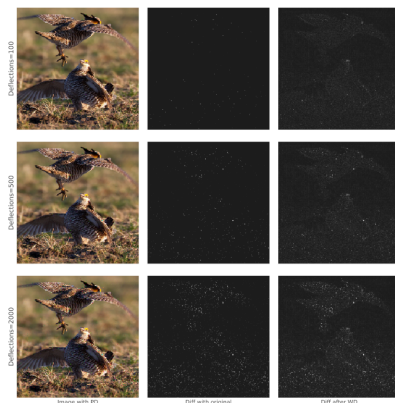


Figure 1: Impact of Pixel Deflection on a natural image and subsequent denoising using wavelet transform. Left: Image with given number of pixels deflected. Middle: Difference between clean image and deflected image. Right: Difference between clean image and deflected image after denoising. Enlarge to see details.

# Ensemble Learning

**Idea :** Create a suite of  $K$  classifiers, each one with a unique key. Combine the features from all the systems using a key-based aggregator function.

**Paper:** Defending against adversarial attacks by randomized diversification

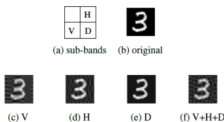


Figure 6: Local randomization in the DCT sub-bands by key-based sign flipping.

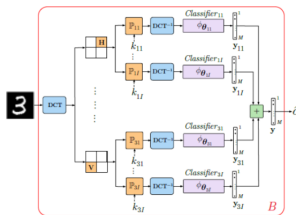


Figure 7: Classification with local DCT sign permutations.

## Adversarial Training

# Adversarial Training (AT)

**Idea :** feed the classifier with adversarial examples during training to boost its robustness against perturbations

**Paper:** [Towards Deep Learning Models Resistant to Adversarial Attacks](#)

Formulated as a min-max problem

$$\underset{\theta}{\text{minimize}} \quad \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \max_{\delta \in \Omega} \mathcal{L}(\theta, x + \delta, y) \right]$$

- $\mathcal{D}$  is the training set
- $x \in \mathbb{R}^d$  and  $y \in \mathbb{R}$  are in input and the associated ground truth.
- $\mathbb{E}$  is the expected value
- $\delta$  is the adversarial perturbation, that must live withing  $\Omega$
- $\theta$  is a vector encompassing all the parameters of the network
- $\mathcal{L}$  is the loss function

# AT General Algorithm

## 1. Generate adversarial examples

At epoch  $t$ , for a subset  $\mathcal{D}' \subseteq \mathcal{D}$  generate adversarial perturbations  $\delta_i^{(t)}$  given the current model parameters  $\theta^{(t)}$  and perturbation budget  $\epsilon$

## 2. Augment training set (max. problem)

Define  $\mathcal{D}_{adv} = \{(x_i + \delta_i^{(t)}, y_i) | x_i \in \mathcal{D}'\}$  and construct a new training set  $\mathcal{D}^{(t)} = \mathcal{D}_{adv} \cup \mathcal{D}$

## 3. Adversarial training (min. problem)

Train on augmented  $\mathcal{D}^{(t)}$  for  $N \geq 1$  epochs, then repeat Step 1.

# AT General Algorithm

## 1. Generate adversarial examples

At epoch  $t$ , for a subset  $\mathcal{D}' \subseteq \mathcal{D}$  generate adversarial perturbations  $\delta_i^{(t)}$  given the current model parameters  $\theta^{(t)}$  and perturbation budget  $\epsilon$

## 2. Augment training set (max. problem)

Define  $\mathcal{D}_{adv} = \{(x_i + \delta_i^{(t)}, y_i) | x_i \in \mathcal{D}'\}$  and construct a new training set  $\mathcal{D}^{(t)} = \mathcal{D}_{adv} \cup \mathcal{D}$

## 3. Adversarial training (min. problem)

Train on augmented  $\mathcal{D}^{(t)}$  for  $N \geq 1$  epochs, then repeat Step 1.

# AT General Algorithm

## 1. Generate adversarial examples

At epoch  $t$ , for a subset  $\mathcal{D}' \subseteq \mathcal{D}$  generate adversarial perturbations  $\delta_i^{(t)}$  given the current model parameters  $\theta^{(t)}$  and perturbation budget  $\epsilon$

## 2. Augment training set (max. problem)

Define  $\mathcal{D}_{adv} = \{(x_i + \delta_i^{(t)}, y_i) | x_i \in \mathcal{D}'\}$  and construct a new training set  $\mathcal{D}^{(t)} = \mathcal{D}_{adv} \cup \mathcal{D}$

## 3. Adversarial training (min. problem)

Train on augmented  $\mathcal{D}^{(t)}$  for  $N \geq 1$  epochs, then repeat Step 1.



# Problems of conventional AT (1)

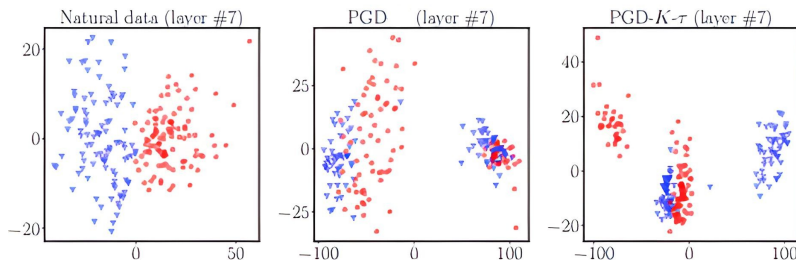
**Problem:** Adversarial perturbations generated by strong attacks significantly cross over the decision boundary and are close to natural data, which hurts the generalization.

**Paper:** Attacks Which Do Not Kill Training Make Adversarial Learning Stronger

**Solution:** *Friendly Adversarial Training* (FAT) – instead of finding the worst-case attack, search for the weakest one to construct  $\mathcal{D}_{adv}$

# Problems of conventional AT (1)

An *Early-stopped PGD* algorithm ( $PGD - K - \tau$ ) is used to find weak adversarial samples without affecting generalization.



Feature projections for two classes ( $\nabla$  and  $\nabla$ ). While standard PGD leads to a significant mixing of the two classes,  $PGD - K - \tau$  maintains separability, i.e. generalization. [\[Source\]](#)

## Problems of conventional AT (2)

**Problem:** Individual data points have different intrinsic robustness – different distances to the decision boundary. However, AT uses the same fixed  $\epsilon$  to generate adversarial data from all of them.

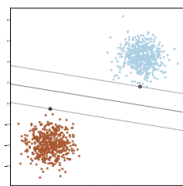
**Paper:** CAT: *Customized Adversarial Training for Improved Robustness*

**Solution:** Adaptively modify the  $\epsilon_i$  for the  $i^{th}$  training sample:

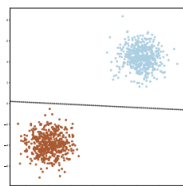
$$\epsilon_i = \operatorname{argmin}_{\epsilon} \left\{ \max_{\mathbf{x}'_i \in \mathcal{B}(\mathbf{x}_i, \epsilon)} f_{\theta}(\mathbf{x}'_i) \neq y_i \right\}$$

The inner maximization uses PGD to find  $\mathbf{x}'_i$ , while the outer minimization modifies  $\epsilon_i$  at each epoch depending on the attack's success.

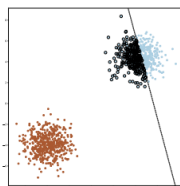
## Problems of conventional AT (2)



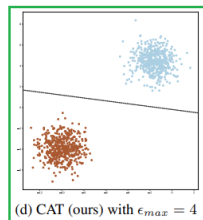
(a) Standard training



(b) Adv-train with  $\epsilon = 1$



(c) Adv-train with  $\epsilon = 4$



(d) CAT (ours) with  $\epsilon_{max} = 4$

Decision boundaries for a linearly separable binary classification problem. [\[Source\]](#)

(b) AT with **low fixed**  $\epsilon$  leads to a decision boundary close to some examples.

(c) AT with **high fixed**  $\epsilon$  leads to loss of generalization, adversarial samples becoming mixed with the other class.

(d) CAT with **bounded flexible**  $\epsilon$  leads to a good trade-off between generalization and robustness

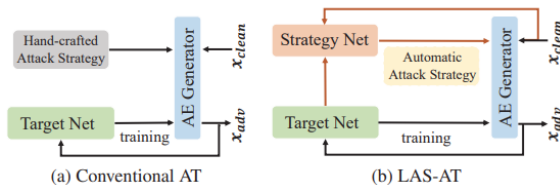
## Problems of conventional AT (3)

### Problems:

- How does one choose an attack to solve the inner maximization?
- Using fixed attack strategies during AT leads to limited robustness.

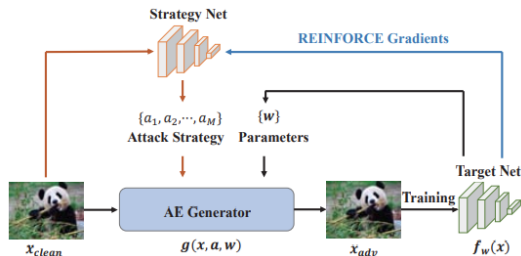
**Paper:** *LAS-AT: Adversarial Training with Learnable Attack Strategy*

**Solution:** Using an additional network to propose the attack strategy:



Difference between AT and LAS-AT. [Source]

## Problems of conventional AT (3)



[Source]

- **Strategy Net** learns to generate attack strategies that maximize the training loss, different for each  $x_{clean}$
- Predictions  $\{a_1, \dots, a_M\}$  correspond to labels for different attack hyperparameters, each  $a_i$  being encoded as a one-hot vector of length  $K_i$
- **REINFORCE**<sup>1</sup> estimates gradients of the adversarial loss w.r.t. parameters of **Strategy Net**, since they cannot be directly computed

# AT + Fake Data

**Problem:** AT suffers from *Robust Overfitting* – **robust test accuracy** starts decreasing during training, while **robust train accuracy** continues to rise

**Paper:** *Overfitting in adversarially robust deep learning*

## Why does it happen?

- Network gets used to the *adversarial noise* generated for training data
- But loses generality when it comes to *adversarial noise* generated on test data (i.e., it learns the noise rather than the pattern)

# AT + Fake Data

**Problem:** AT suffers from *Robust Overfitting* – **robust test accuracy** starts decreasing during training, while **robust train accuracy** continues to rise

**Paper:** *Overfitting in adversarially robust deep learning*

## Why does it happen?

- Network gets used to the *adversarial noise* generated for training data
- But loses generality when it comes to *adversarial noise* generated on test data (i.e., it learns the noise rather than the pattern)



## AT + Fake Data

**Problem:** AT suffers from *Robust Overfitting* – **robust test accuracy** starts decreasing during training, while **robust train accuracy** continues to rise

**Paper:** *Overfitting in adversarially robust deep learning*

### Why does it happen?

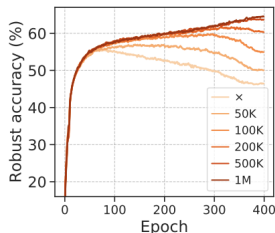
- Network gets used to the *adversarial noise* generated for training data
- But loses generality when it comes to *adversarial noise* generated on test data (i.e., it learns the noise rather than the pattern)

# AT + Fake Data

**Solution:** Replace adversarial training samples with generated data (e.g. DDPM-based diffusion)

## Papers:

- *Fixing Data Augmentation to Improve Adversarial Robustness*
- *Better Diffusion Models Further Improve Adversarial Training*



Robust accuracy evolution when using different amounts of generated data during AT. 'x' corresponds to no additional data. [\[Source\]](#)

## AT + Fake Data

## Augment training set with generated data

Using a set of generators  $\mathcal{G} = \{g_1, \dots, g_m\}$ , define the augmented training set  $\mathcal{D}_{aug} = \mathcal{D}_{fake} \cup \mathcal{D}$  as follows:

- For unconditional generators:

$$\mathcal{D}_{fake} = \left\{ \left( g_i(z), \mathcal{T}(g_i(z)) \right) \middle| g_i \in \mathcal{G}, z \sim \mathcal{N}(0, 1) \right\}$$

where  $\mathcal{T}(\cdot)$  is a pre-trained classifier on  $\mathcal{D}$

- For conditional generators:

$$\mathcal{D}_{fake} = \left\{ \left( g_i(z | y_j), y_j \right) \middle| g_i \in \mathcal{G}, z \sim \mathcal{N}(0, 1), y_j \in \mathcal{Y} \right\}$$

where  $\mathcal{Y}$  is the set of all labels in  $\mathcal{D}$

## Regularization Techniques

# Regularization Training

**Idea :** Use an objective for minimization that leads to robust structures, while maintaining clean accuracy

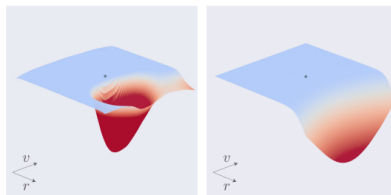
Including additional loss terms

$$\mathcal{L}(x_i, y_i, \theta) \leftarrow \mathcal{L}_{cls}(x_i, y_i, \theta) + \alpha \mathcal{L}_{reg}(x_i, y_i, \theta, \text{*args})$$

- $\theta$  – network parameters
- $\mathcal{L}_{cls}$  – standard loss function
- $\alpha$  – regularization weight
- $\mathcal{L}_{reg}$  – regularization loss, which could take as parameters data, model parameters, or both
- \*args – other optional parameters controlling the regularization

# Regularization Training (1)

**Motivation:** AT leads to a significant decrease in the curvature of the loss surface with respect to inputs, leading to a more “linear” behavior



(a) Original (CIFAR-10)      (b) Fine-tuned (CIFAR-10)

Loss landscape before AT (a) and after AT (b). Blue color corresponds to low loss values, while red corresponds to high loss = adversarial region.

**Paper :** Robustness via Curvature Regularization, and Vice Versa

**Idea :** Penalize large eigenvalues in the Hessian of loss w.r.t. input points.

# Regularization Training (1)

## Curvature Regularization

$$\mathcal{L}_{reg}(x, y, \theta) = \|\nabla \mathcal{L}_{cls}(x + hz, y, \theta) - \nabla \mathcal{L}_{cls}(x, y, \theta)\|_2^2$$

where  $h$  is a small constant, and:

$$z = \frac{\nabla \mathcal{L}_{cls}(x, y, \theta)}{\|\nabla \mathcal{L}_{cls}(x, y, \theta)\|_2}$$

- $\mathcal{L}_{cls}$  is a standard classification loss
- Different from AT which flattens the whole loss landscape around clean samples, this reduces curvature only in some significant directions, where adversarial attacks are more likely to be found
- Imposes regularity of gradients on sufficiently small neighbourhoods ( $h$ ) around clean points

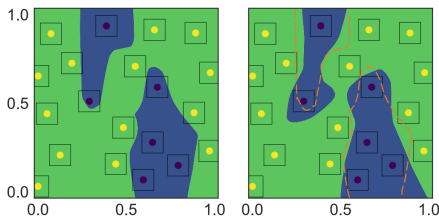
## Regularization Training (2)

**Paper :** TRADES: TRadeoff-inspired Adversarial DEFense via Surrogate-loss minimization

**Idea:** Manipulate the trade-off between natural error  $\mathcal{R}_{nat}$  and robust error  $\mathcal{R}_{rob}$ , where:

$$\mathcal{R}_{rob} = \mathcal{R}_{nat} + \mathcal{R}_{bdy}$$

where  $\mathcal{R}_{bdy}$  is the *boundary error*, measuring the n.o. correctly classified points laying near the decision boundary



Standard training (**left**) and TRADES (**right**) decision boundaries.



# Regularization Training (2)

## TRADES

$$\mathcal{L}_{reg}(x, y, \theta) = \max_{x', |x' - x| \leq \epsilon} \mathcal{L}_{CE}(f_{\theta}(x'), f_{\theta}(x))$$

where  $\mathcal{L}_{CE}$  is the standard cross-entropy loss

- Encourages the output to be smooth in the vicinity of  $x$ , pushing the decision boundary away
- Similar with AT: forces adversarial samples  $x'$  to be seen as clean samples  $x$  by the network  $f_{\theta}$
- Different from AT: doesn't explicitly force the same class for  $x'$ , but pushes towards very similar outputs of the network

# Regularization Training (3)

**Paper :** Metric Learning for Adversarial Robustness

**Idea :** Introduces *Triplet Loss Adversarial* (TLA) learning, pushing towards:

- 1 Similarity between adversarial examples  $x_a$  and samples they've been generated from  $x_p$  (*positive*)
- 2 Dissimilarity between adversarial examples  $x'$  and clean samples from a different class  $x_n$  (*negative*)



## Regularization Training (3)

### TLA – Triplet loss regularizer

$$\mathcal{L}_{reg}(x_p, x_a, x_n, \theta) = \max \left( D(f_{\theta}^{(k)}(x_p), f_{\theta}^{(k)}(x_a)) - D(f_{\theta}^{(k)}(x_n), f_{\theta}^{(k)}(x_a)) + \alpha, 0 \right)$$

where  $x_a = x_p + \delta$ ,  $f_{\theta}^{(k)}$  outputs the feature vector from  $k^{th}$  layer, and:

$$D(x, y) = 1 - \frac{|\langle x, y \rangle|}{\|x\|_2 \|y\|_2}$$

- Hyperparameter  $\alpha$  is called *margin*, and it forces the two distances to be at least  $\alpha$ -distanced
- $x_a$  is called *anchor*, since all distances are related to it
- Negative samples  $x_n$  for computing  $\mathcal{L}_{reg}$  are selected based on the closest negatives to  $x_a$ , measured in the embedding space

# Regularization Training (3)

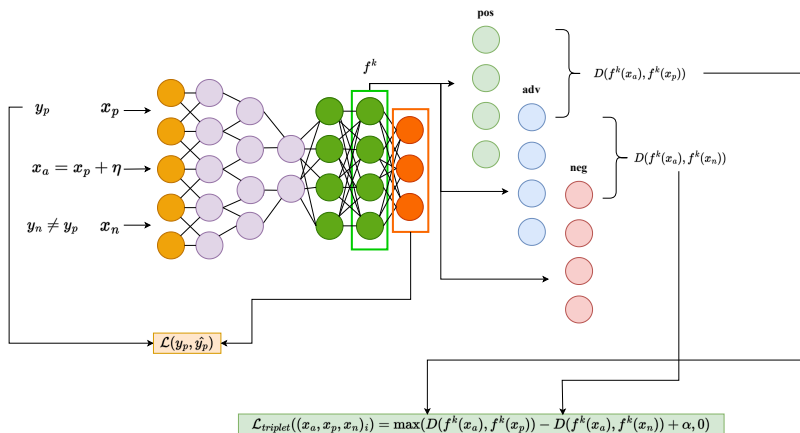
**Problem :** Features  $f_{\theta}^{(k)}$  might have very different ranges between  $x_a, x_p, x_n$

## TLA with Norm Regularization

$$\mathcal{L}_{reg}(x_p, x_a, x_n, \theta) + \lambda \left( \|f_{\theta}^{(k)}(x_a)\|_2 + \|f_{\theta}^{(k)}(x_p)\|_2 + \|f_{\theta}^{(k)}(x_n)\|_2 \right)$$

- Further reduces the influence of very high / low feature norms over the regularization term
- Forces similarity / dissimilarity in the *angular space*, rather than between  $\ell_2$  magnitudes

# Regularization Training (3)

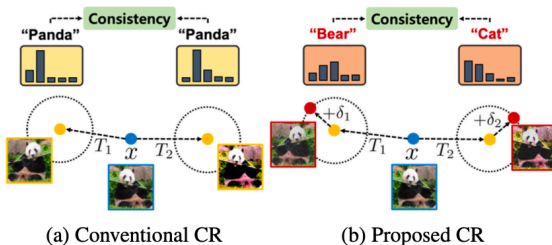


Triplet Loss Adversarial training workflow

## Regularization Training (4)

**Paper :** Consistency Regularization for Adversarial Robustness

**Idea :** Enforce consistency between adversarial examples generated from different transformations of the same image



Usual consistency (a) and *attacked augmentation* consistency (b).  $T_1$  and  $T_2$  are two different image transformations (e.g. rotation, random crop, flip).

# Regularization Training (4)

## Consistency Regularization

$$\mathcal{L}_{reg} = \text{JS}\left(\hat{f}_{\theta}(T_1(x) + \delta_1, \tau) \parallel \hat{f}_{\theta}(T_1(x) + \delta_1, \tau)\right)$$

where JS is the Jensen-Shannon divergence (i.e. symmetric variant of KL) and

$$\hat{f}_{\theta}(T(x) + \delta, \tau) = \text{Softmax}\left(\frac{f_{\theta}(T(x) + \delta)}{\tau}\right)$$

is a temperature-scaled ( $\tau$ ) variant of classifier  $f_{\theta}$

- Augmentations are randomly sampled in every training step  $\rightarrow$  prediction on adversarial data becomes consistent
- Temperature-scaling enforces a sharper distribution, since confidence (i.e. max probability) for AT remains low

## Certified Robust Methods



# Why "certified"?

## Problems :

- AT and Regularization training are empirical defenses, which aim to improve robustness against empirical attacks
- We need mechanisms for providing / imposing a lower bound on robust accuracy, against **any attack** with maximum budget  $\epsilon$

**Paper :** Certified Defenses Against Adversarial Examples

## Solution : Certified Defenses:

- 1 Probabilistic
- 2 Deterministic

# Why "certified"?

## Certified Defense formulation

For a given input  $x$  and it's associated class  $y$ , a certification assessment over a robust classifier  $f_\theta$  w.r.t. to an attack budget  $\epsilon$  can be stated as follows:

$$\arg \max_j f_\theta(x + \delta)_j = y, \quad \forall \delta \text{ with } \|\delta\|_p \leq \epsilon$$

Note that the above statement doesn't refer to any attack mechanism. More concretely, the following relation between logits holds:

$$f_\theta(x + \delta)_y - f_\theta(x + \delta)_j \geq m(\epsilon, \theta)$$

where  $m(\epsilon, \theta)$  is called the *certified margin*. If  $m(\epsilon, \theta) > 0$ , classifier  $f_\theta$  is robust within  $\mathbb{B}(x, \epsilon)$ .

### Core tasks :

- Construct effective and accurate margins  $m(\epsilon)$
- Design algorithms for controlling these margins

# Certified Probabilistic Defenses (1)

## Papers :

- Certified Adversarial Robustness via Randomized Smoothing (RS)
- Randomized Smoothing of All Shapes and Sizes

**Idea :** Construct a *smoothed* classifier  $g_\theta$  from a base classifier  $f_\theta$ , s.t.:

$$g_\theta(x, \sigma) = \arg \max_c \mathbb{P}(\arg \max_j f_\theta(x + \epsilon)_j = c), \epsilon \sim \mathcal{N}(0, \sigma^2 I)$$

which corresponds to the most likely class for a Gaussian perturbed  $x$ , and  $\sigma$  is a hyperparameter controlling the robustness / accuracy trade-off.

**How do we assess the  $\ell_2$  robustness radius around  $x$ ?**

# Certified Probabilistic Defenses (1)

## Robustness Guarantee of RS

Suppose that for input data sampled from  $\mathcal{N}(x, \sigma^2 I)$ , the most probable class predicted by  $f_\theta$  is  $c_A$  with probability  $p_A$ , and the second most likely class is  $c_B$  with probability  $p_B$ . Then, the smoothed classifier  $g_\theta$  is robust around  $x$  with  $\ell_2$  radius:

$$R(x, g_\theta) = \frac{\sigma}{2} (\Phi^{-1}(p_A) - \Phi^{-1}(p_B))$$

which corresponds to:

$$\arg \max f_\theta(x + z) = \arg \max f_\theta(x), \forall \|z\|_2 \leq R$$

where

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^0 e^{-t^2/2} dt$$

is the CDF of a standard Gaussian.

- This provides a link between input Gaussian noise power and certified  $\ell_2$  distance

# Certified Probabilistic Defenses (1)

## Prediction & Certification

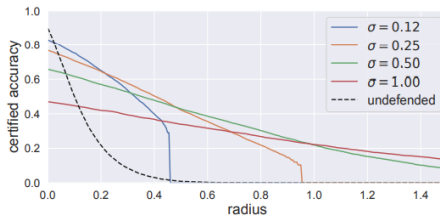
- 1 Sample  $N$  points from  $\mathcal{N}(x, \sigma^2 I)$  and compute predictions of  $f_\theta$
- 2 Compute the two most frequent classes  $c_A$  and  $c_B$ , and their corresponding counts  $n_A$  and  $n_B$
- 3 Compute the p-value of binomial test (for  $p = 0.5$  and significance level  $\alpha$ ):

$$\text{p-val} = \sum_{k=n_A}^{n_A+n_B} \binom{n_A+n_B}{k} p^k (1-p)^{n_A+n_B-k}$$

- 1 if  $\text{p-val} > \alpha$ : **ABSTAIN** (accept null hypothesis  $p = 0.5$  and deem the result statistically insignificant)
- 2 else:
  - **RETURN**  $c_A$  as the robust prediction
  - Estimate a lower bound  $\underline{p}_A$  and upper bound  $\overline{p}_B$  and **RETURN**  $R$

# Certified Probabilistic Defenses (1)

- **Training:** Gaussian data augmentation at variance  $\sigma^2$



Certified accuracy of RS on CIFAR-10 udner different train settings. [\[Source\]](#)

- **Drawbacks :**
  - Requires many samples to estimate certified radius (e.g.  $N = 10^5$ )
  - Can't impose a specific certified radius, since everything is randomly constructed

## Certified Probabilistic Defenses (2)

**Problem :** How to maximize the  $\ell_2$  robustness without explicitly adding  $\ell_2$  adversarial perturbations to data (i.e. AT) ?

**Paper :** MACER: Attack-free and scalable robust training via maximizing certified radius

**Solution :**

- Attack-free training
- Maximizes  $\ell_2$  certified radius during training, using Gaussian data augmentation and an additional loss term (regularization)

# Certified Probabilistic Defenses (2)

## MACER – MAXimizing the CERTified Radius

$$\mathcal{L}(x, y, \theta) = \mathcal{L}_{cls}(x + \eta, y, \theta) + \underbrace{\frac{\lambda\sigma}{2} \max(\gamma - \hat{R}(x, g_\theta), 0)}_{\ell_2 \text{ radius maximization}} \Big|_{g_\theta(x)=y}$$

where  $\hat{R}$  is a **soft randomized smoothing radius** version of  $R$ , which can be differentiated w.r.t.  $\theta$

- The first term corresponds to RS training
- The second term pushes certified radius to be close to  $\gamma$ , for points correctly classified by smoothed classifier  $g_\theta$



# Certified Deterministic Defenses (1)

**Problem :** How to directly control the output perturbation given the input one?

## Papers :

- Achieving robustness in classification using optimal transport with hinge regularization
- Pay attention to your loss: understanding misconceptions about 1-Lipschitz neural networks

**Solution :** Structurally constrain neural networks to impose a desired Lipschitz bound  $L_{p,q}$  over  $f_\theta$ :

$$L_{p,q} = \max_{x \neq y} \frac{\|f_\theta(x) - f_\theta(y)\|_p}{\|x - y\|_q}, \quad p, q \in \mathbb{N}$$

This ensures that:

$$\|f_\theta(x + \delta) - f_\theta(x)\|_p \leq L_{p,q} \|\delta\|_q$$

# Certified Deterministic Defenses (1)

## Lipschitz bound of feed-foward networks

Consider  $f_\theta$  to be a feed-forward neural network with no skip connections, 1-Lipschitz activation functions, and  $m$  layers defined by their weight tensors  $\{W_i\}_{i=1}^m$ . Then, for the usual case  $p = q = 2$  the following holds:

$$L_{2,2} \leq \prod_{i=1}^m \|W_i\|_{2,2}$$

which is an upper bound of the Lipschitz constant of  $f_\theta$ , and

$$\|W_i\|_{p,q} = \max_{x \neq 0} \frac{\|W_i x\|_p}{\|x\|_q}$$

- For 2D convolutional layers  $W_i \in \mathbb{R}^{C_{in} \times C_{out} \times h \times w}$  one has the following options:
  - 1 Reshape to  $\widetilde{W}_i \in \mathbb{R}^{hw C_{in} \times C_{out}}$  and compute spectral norm  $\|\widetilde{W}_i\|_2$
  - 2 Compute 2D FFT  $\mathbf{W}_i \in \mathbb{R}^{C_{in} \times C_{out} \times n_{fft} \times n_{fft}}$  and take the maximum magnitude over frequency dimensions (roughly)
  - 3 Other techniques ...

# Certified Deterministic Defenses (1)

## Imposing Lipschitz bounds – Spectral Normalization

For a network  $f_\theta$  as previously defined, one can impose a Lipschitz bound  $L_{2,2} = \hat{L}$  using the following normalization scheme:

$$W'_i \leftarrow \frac{W_i}{\|W_i\|_2} (\hat{L})^{\frac{1}{m}}$$

which holds  $\prod_{i=1}^m \|W'_i\|_2 = \hat{L}$

- Spectral Normalization can be applied as an additional step after the usual gradient update
- Faster techniques such as *Power Iteration* and *Gram Iteration* are employed to efficiently estimate  $\|W_i\|_2$  during training

# Certified Deterministic Defenses (1)

**Problem :** In a classification setting, we're interested in the  $\ell_2$  norm of each output neuron, not the entire logit vector

## Lipschitz bounds of individual logits

For a network  $f_\theta$  as previously defined, the Lipschitz bound of logit  $f_\theta^j$  is defined as follows:

$$L_{2,2}^j \leq \left( \prod_{i=1}^{m-1} \|W_i\|_2 \right) \|W_m[:, j]\|_2$$

where  $W_m[:, j]$  corresponds to the  $j^{th}$  column of  $W_m$ , connected to the  $j^{th}$  logit

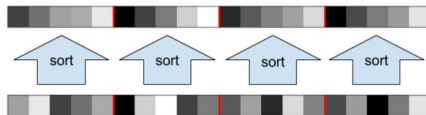
- $\|W_m[:, j]\|_2$  can be imposed similarly to the other layers
- We can quantify how much the correct logit  $y$  will change when input is perturbed:

$$\|f_\theta^y(x + \delta) - f_\theta^y(x)\|_2 \leq L_{2,2}^y \|\delta\|_2$$

# Certified Deterministic Defenses (1)

## Requirements for training 1-Lipschitz networks (under the previous settings)

- **Norm-Preserving activation functions**, e.g. GroupSort / FullSort:



Source: [Sorting Out Lipschitz Function Approximation](#)

- **Gradient Norm Preservation** during backpropagation:  
Bjorck orthonormalization over constrained  $W_i$ 's  $\rightarrow$   
iteratively finds the closest orthonormal matrix  $\rightarrow$   
imposes all singular values to be close to 1, to help gradient flow

# Certified Deterministic Defenses (2)

## Papers :

- Globally-Robust Neural Networks (GloRoNets)
- Relaxing Local Robustness

**Idea :** Introduce an additional class (denoted  $\perp$ ) to signal that a point cannot be certified as  $\epsilon$ -**globally-robust**

### Local vs Global $\epsilon$ -robustness

**Local (at  $x$ ):**  $\|x - x'\| \leq \epsilon \implies \arg \max_j f_\theta(x)_j = \arg \max_j f_\theta(x')_j, \forall x'$

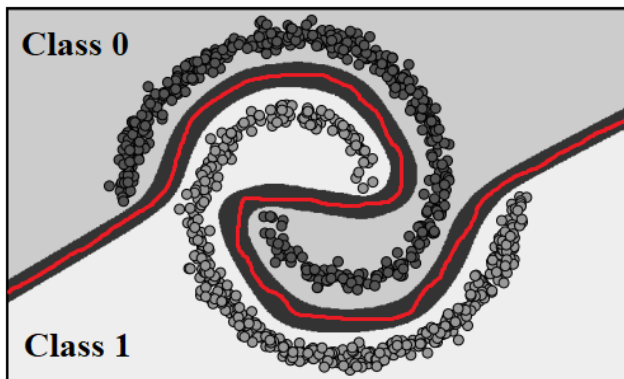
**Global:**  $\|x_1 - x_2\| \leq \epsilon \implies \arg \max_j f_\theta(x_1)_j \stackrel{\perp}{=} \arg \max_j f_\theta(x_2)_j, \forall x_1, x_2$   
 ,where  $\stackrel{\perp}{=}$  holds True if:

$$\arg \max_j f_\theta(x_1)_j = \perp \quad \text{OR}$$

$$\arg \max_j f_\theta(x_2)_j = \perp \quad \text{OR}$$

$$\arg \max_j f_\theta(x_1)_j = \arg \max_j f_\theta(x_2)_j$$

## Certified Deterministic Defenses (2)



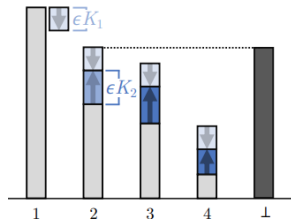
Global robustness between two classes. Red line corresponds to the decision boundary, with surrounding dark gray area corresponding to points labeled as  $\perp$ , due to closeness to the boundary. [\[Source\]](#)

# Certified Deterministic Defenses (2)

## Constructing logit $\perp$

Denote  $K_i = L_{2,2}^i$  the Lipschitz constant of the  $i^{th}$  logit. Then, for an example  $x$  correctly classified as  $y = \arg \max_j f_\theta(x)_j$ , the logit  $y_\perp$  is constructed as follows:

$$y_\perp = \max_{i \neq y} \{f_\theta(x)_i + (K_i + K_y)\epsilon\}$$



$\perp$  logit accounts for two worst-case scenarios:

- ➊ Predicted class decreases by maximum amount  $\epsilon K_1$
- ➋ Second-most likely class increases by maximum amount  $\epsilon K_2$

The model is pushed to predict logits for the correct class that surpass logit  $\perp \implies$  global  $\epsilon$ -robustness



# Certified Deterministic Defenses (2)

## Training GLoRoNets

- 1 For each element  $x_i$  in batch  $B$  construct logit value  $y_{\perp}$ , with an initially set  $\epsilon$
- 2 Concatenate  $y_{\perp}$  with  $f_{\theta}(x_i)$  and optimize network over this augmented logit vector

- For a GLoRoNet trained with a specific  $\epsilon$ , it's final accuracy corresponds to the fraction of points which are globally  $\epsilon$ -robust:

$\forall \delta$  with  $\|\delta\| \leq \epsilon$  and  $\forall x$  correctly classified,  $x + \delta$  is correctly classified

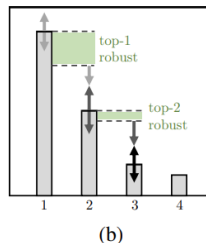
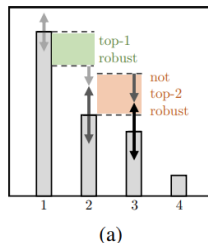
- Alternatively, for any new point  $x$  we can say what's the highest  $\epsilon$  for which it's certified that any attack with  $\|\delta\| \leq \epsilon$  won't work

**Observation:** GLoRoNets are shown to impose strong implicit regularization on global Lipschitz.

**Note:** Lipschitz bound needs to be computed each time  $y_{\perp}$  is computed. Can it be made more efficient?

# Certified Deterministic Defenses (2)

## Generalisation : Certified top-k robustness



Top-2 robust (b) vs. non-top-2 robust (a) predictions. In (a), the third logit is sufficiently large to overcome the second, if sufficiently perturbed. [\[Source\]](#)

- We're interested in keeping the top-k predictions unchanged
- Logit  $\perp$  is constructed s.t. the minimum logit change that can change the top-k predictions is pushed away

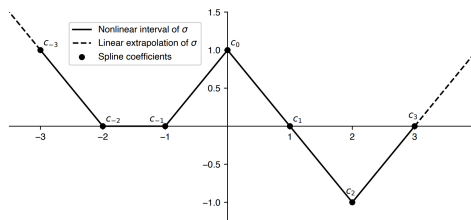
# Certified Deterministic Defenses (3)

**Problem :** Common activation functions limit the expressivity of Lipschitz-constrained networks (e.g. ReLU, LeakyReLU)

**Paper :** Improving Lipschitz-Constrained Neural Networks by Learning Activation Functions

**Solution :** Define a controllable structure for activation functions, ensure universal approximation theory, and make them learnable:

$$\sigma_{\ell,n}(x) = b_{1,\ell,n} + b_{2,\ell,n}x + \sum_{k=1}^{K_{\ell,n}} a_{k,\ell,n} \text{ReLU}(x - \tau_{k,\ell,n})$$



Example of Learnable Linear Spline (LLS) activation. [\[Source\]](#)

# Certified Defenses – Extending to Multi-modal inputs

**Problem :** Hard to adapt previous methods for complex networks / tasks, in a certified manner

**Paper :** MMCert: Provable Defense against Adversarial Attacks to Multi-modal Models

- Input of  $T$  modalities:  $M = (m_1, m_2, \dots, m_T)$ ,  $|m_i| = n_i$
- Attack types: modification, addition, deletion
- N.o. maximum elements attacked in each modality:  $R = (r_1, r_2, \dots, r_T)$
- Sub-sampled input:  $Z = (z_1, z_2, \dots, z_T)$ ,  $|z_i| = k_i$

**Idea :** Use Monte Carlo to estimate a lower and an upper bound for the probability of correct class over multiple sub-sampled inputs:

$$\underline{p}_y = \text{LB}(f_\theta(Z)); \quad \overline{p}_y = \text{UB}(f_\theta(Z))$$

# Certified Defenses – Extending to Multi-modal inputs

## Certified Multi-Modal Classification

Consider  $N$  sub-samplings  $Z_1, \dots, Z_N$  and a multi-modal classifier  $f_\theta$ .

- ① Compute predictions over all  $Z_i$
- ② Compute top-2 most frequent classes  $A$  and  $B$  and their lower and upper bounds  $\underline{p}_A$  and  $\overline{p}_B$ , using RS techniques

- ③ Compute:  $\delta_l = \underline{p}_A - \frac{\lfloor \underline{p}_A \prod_{i=1}^T \binom{n_i}{k_i} \rfloor}{\prod_{i=1}^T \binom{n_i}{k_i}}$  and  $\delta_u = \frac{\lceil \overline{p}_B \prod_{i=1}^T \binom{n_i}{k_i} \rceil}{\prod_{i=1}^T \binom{n_i}{k_i}} - \overline{p}_B$

- ④ Then,  $f_\theta$  is robust w.r.t.  $R = (r_1, r_2, \dots, r_T)$  if:

$$\frac{\prod_{i=1}^T \binom{n_i}{k_i}}{\prod_{i=1}^T \binom{n'_i}{k_i}} \left( \underline{p}_A - \delta_l - 1 + \frac{\prod_{i=1}^T \binom{e_i}{k_i}}{\prod_{i=1}^T \binom{n_i}{k_i}} \right) \geq \frac{\prod_{i=1}^T \binom{n_i}{k_i}}{\prod_{i=1}^T \binom{n'_i}{k_i}} (\overline{p}_B + \delta_u) + 1 - \frac{\prod_{i=1}^T \binom{e_i}{k_i}}{\prod_{i=1}^T \binom{n'_i}{k_i}}$$

where  $e_i = n_i - r_i$

**Note:** For segmentation (i.e. multi-classification) things become a lot more complicated.

## Certified Deterministic Defenses – Limitations

- Reduction in clean accuracy:

Imposing hard constraints affects prediction on points close to the decision boundary.

- Bounds for complex structures:

Developing tight Lipschitz bounds for complex networks (e.g. Transformers) is not straightforward.

- Computational overhead

Certified methods require additional computational steps in addition to standard training.