IMPERIAL

The Alan Turing Institute

Robustness of Learning Algorithms

Matthew Wicker

Talk for University of Edinburgh MATH11228

24 February 2025

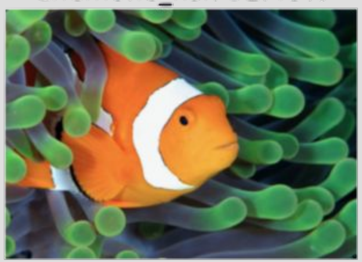
Talk agenda

- Background: Why Robustness?
- Crafting Adversarial Attacks
    - Fast Gradient Sign Method
    - Projected Gradient Descent
- Proving Robustness to Adversarial Attacks
    - Interval bound propagation
- Adversarial Training
- Jupyter Notebook Assignment!

Deep learning "understands" images

ballplayer 69.22%

anemone_fish 92.48%



African_elephant 89.94%

forklift 98.95%

ice_cream 99.60%

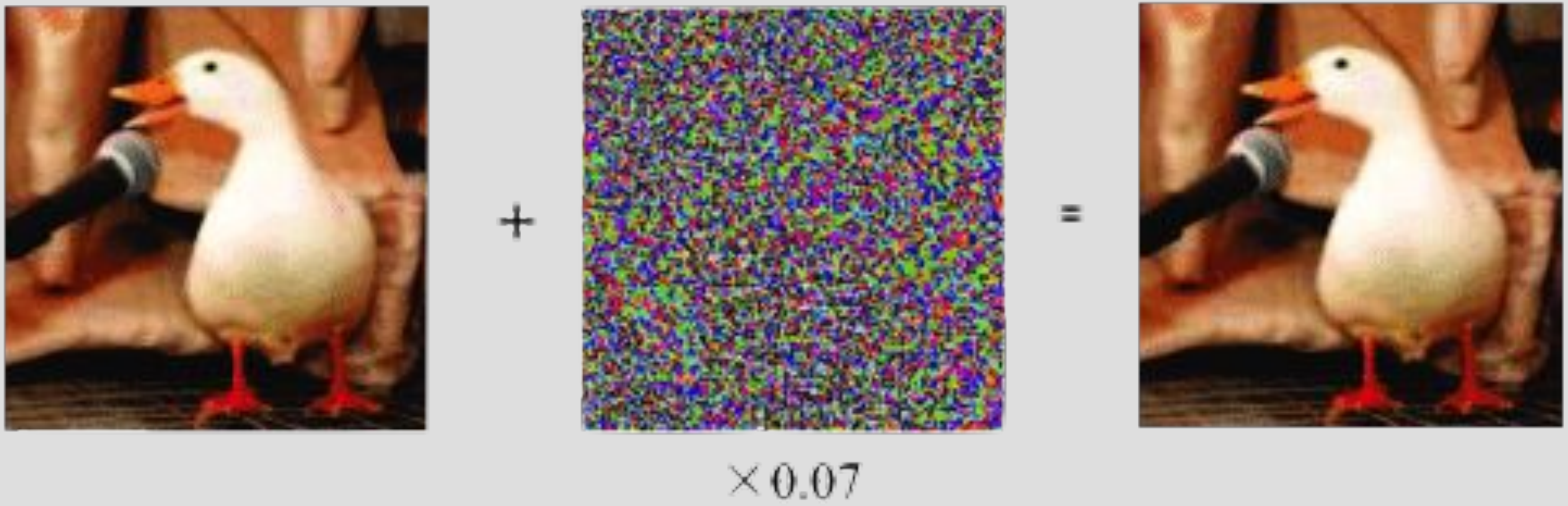
lemon 97.06%

magnetic_compass 97.08%

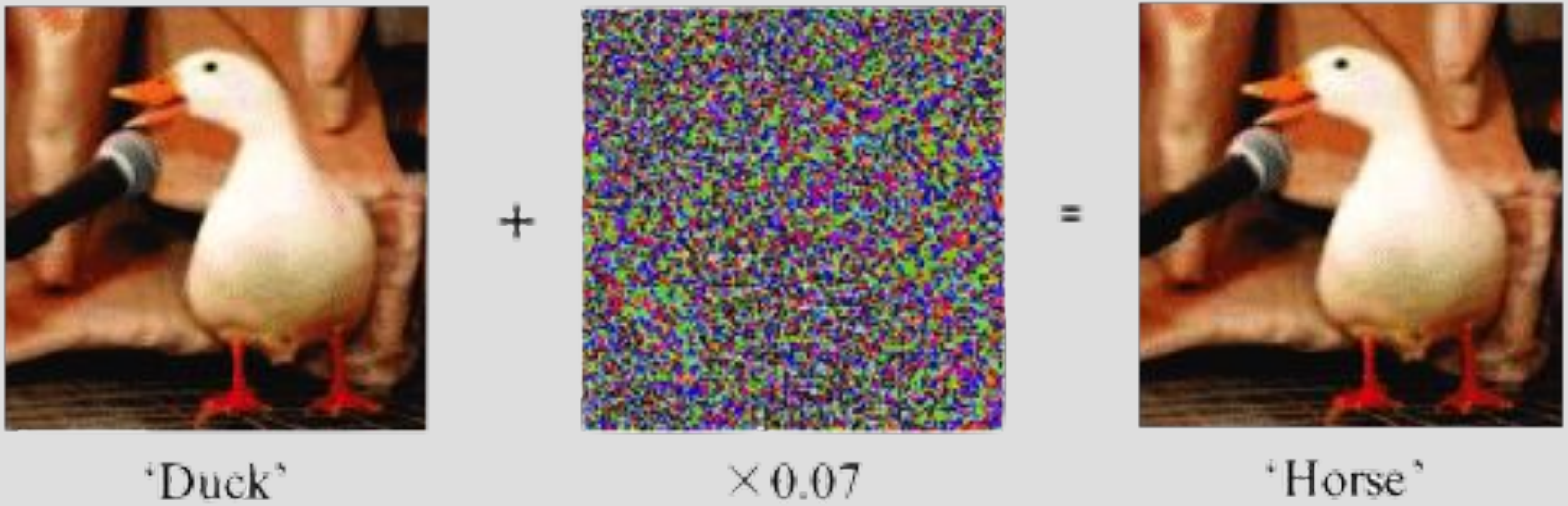
ice_bear 84.80%

What is the difference between these images?

Just a little bit of noise!



Just a little bit of noise! ... can be catastrophic



Adversarial examples for neural networks

original semantic segmentation framework

compromised semantic segmentation framework

[Cisse et. al., NeurIPS 2018]

Adversarial examples for neural networks

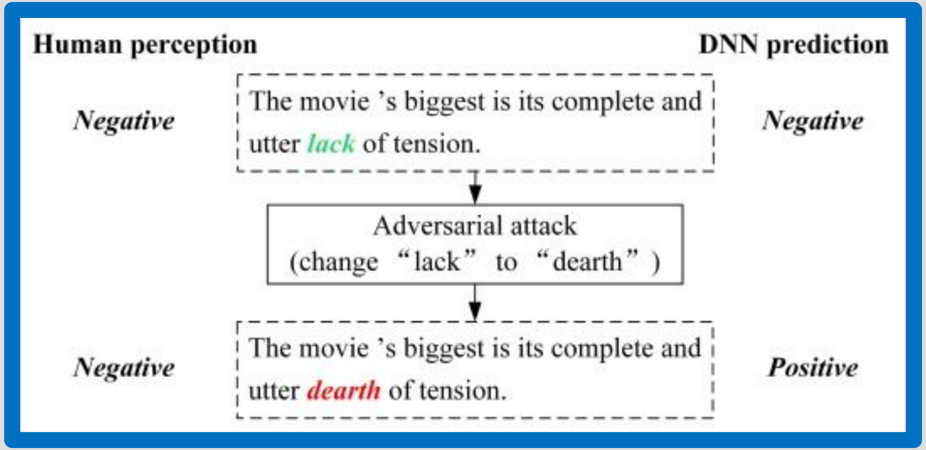
original semantic segmentation framework

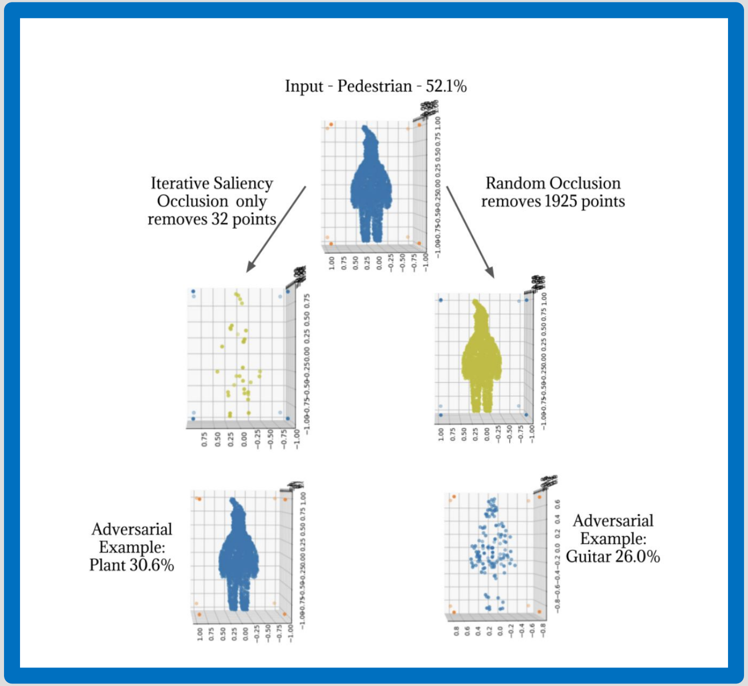
compromised semantic segmentation framework

[Cisse et. al., NeurIPS 2018]

Roborace v1.1 on October 2020

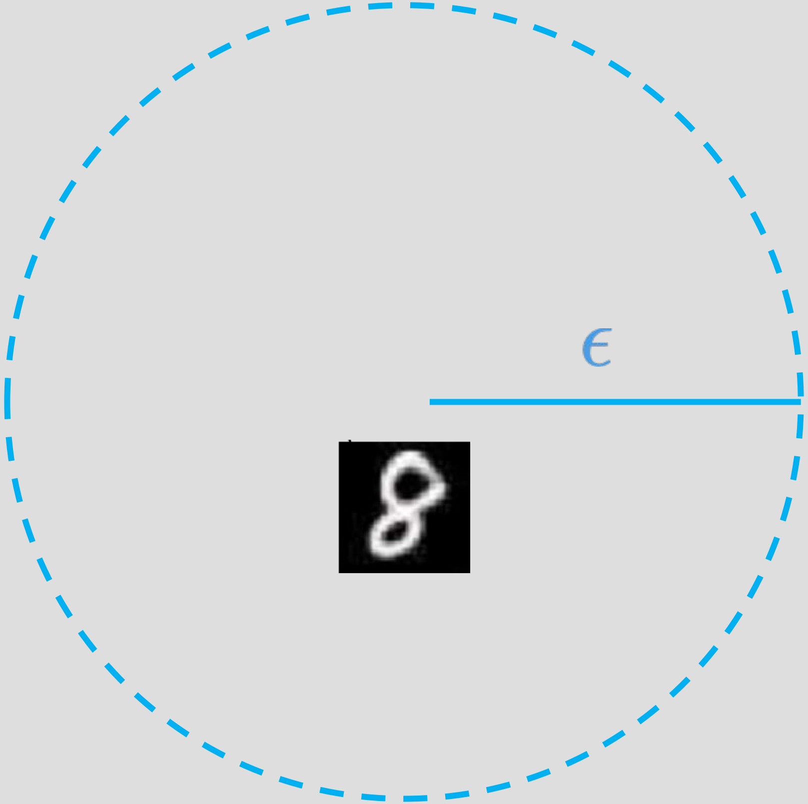
Adversarial Attacks Across Modalities





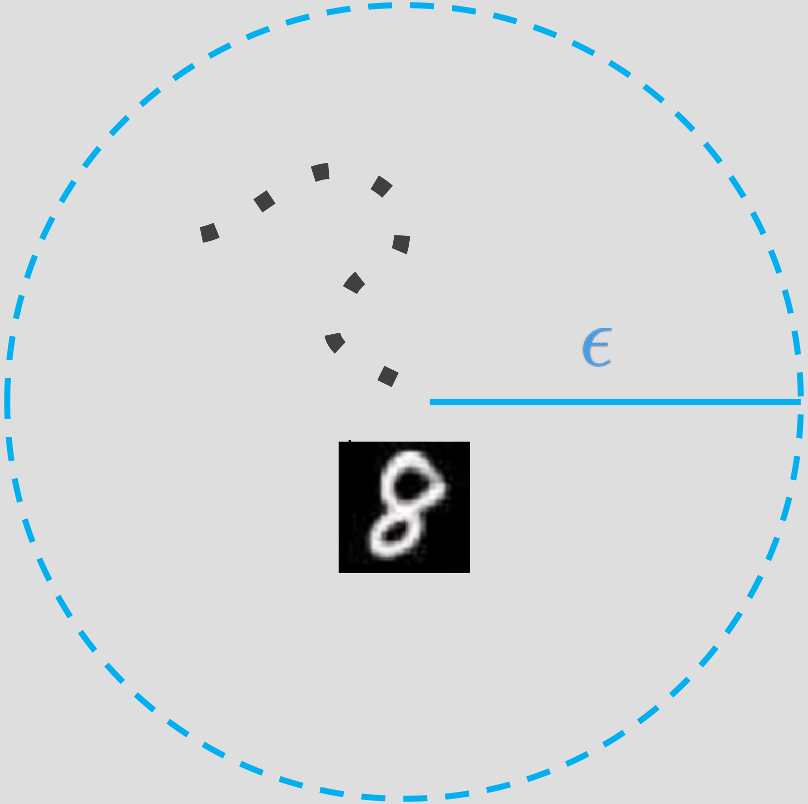
Adversarial examples and their impact transcend data modality

Adversarial attacks



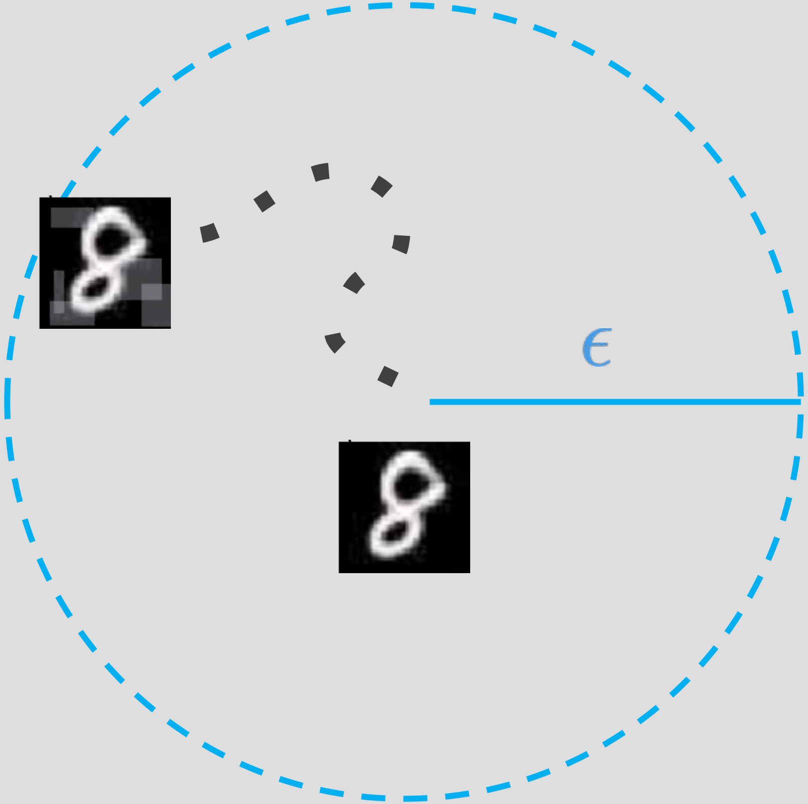
- Start with a natural image
- Given a specification of some epsilon ball

Adversarial attacks



- Start with a natural image
- Given a specification of some epsilon ball
- Search the ball (FGSM/PGD)

Adversarial attacks



- Start with a natural image
- Given a specification of some epsilon ball
- Search the ball
- Return the *worst* example you have found

Crafting Adversarial Attacks as Optimization

$$x^\star = \arg\max_{x' \in \mathcal{B}_\epsilon(x)} \boxed{\mathcal{L}(f^\theta(x'))}$$

- Loss function - Measure of fit for our model

Crafting Adversarial Attacks as Optimization

$$x^\star = \arg\max_{\boxed{x' \in \mathcal{B}_\epsilon(x)}} \boxed{\mathcal{L}(f^\theta(x'))}$$

- Loss function - Measure of fit for our model
- Epsilon Ball - The set of "similar" inputs

Crafting Adversarial Attacks as Optimization

$$x^\star = \boxed{\arg\max}_{\boxed{x' \in \mathcal{B}_\epsilon(x)}} \boxed{\mathcal{L}(f^\theta(x'))}$$

- Loss function - Measure of fit for our model
- Epsilon Ball - The set of "similar" inputs
- Argmax - the input from the ball that max. the loss

Crafting Adversarial Attacks as Optimization

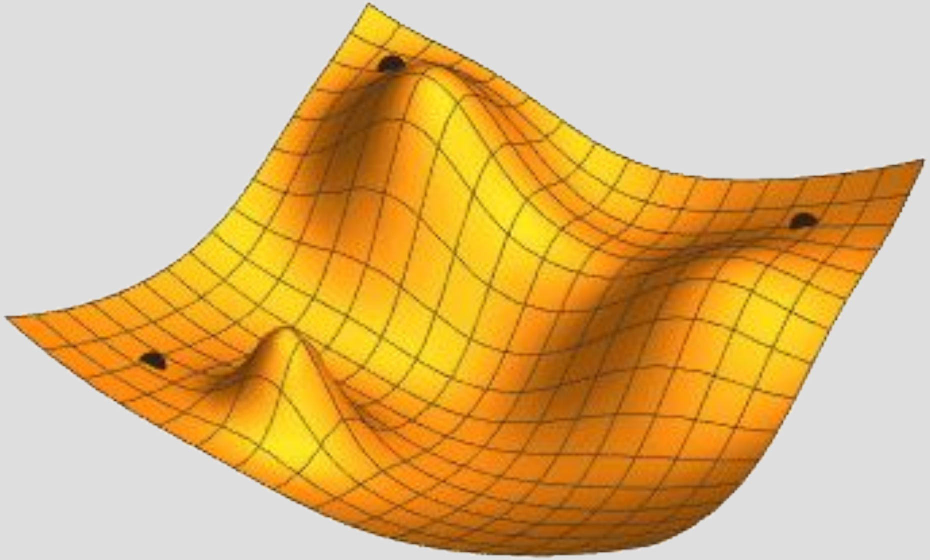
$$\boxed{x^\star} = \boxed{\arg\max}_{\boxed{x' \in \mathcal{B}_\epsilon(x)}} \boxed{\mathcal{L}(f^\theta(x'))}$$

- Loss function - Measure of fit for our model
- Epsilon Ball - The set of "similar" inputs
- Argmax - the input from the ball that max. the loss
- An approximate worst-case example

Solving this optimization problem

The workhorse of modern machine learning approaches is first-order optimization.

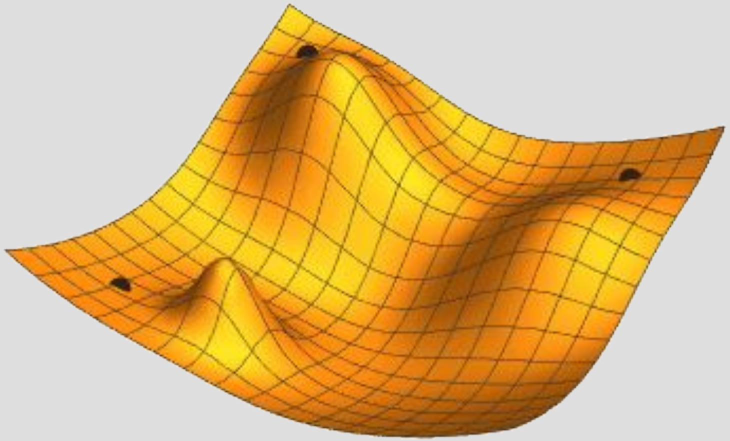
Let's look at a few popular approaches



Gradient Descent (general formulation)

$$x^{(t+1)} = x^{(t)} - \alpha \nabla_x \mathcal{L}(f^\theta(x^{(t)}))$$

As a recap: a good physical interpretation of gradient descent is to consider the loss function as a landscape that we will roll a ball down the hill.
In this picture alpha is the speed of the ball, and t is the time



Gradient Descent (general formulation)

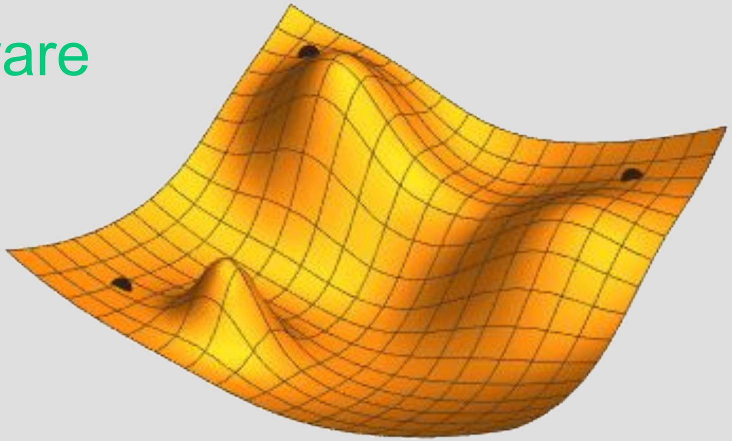
$$x^{(t+1)} = x^{(t)} - \alpha \nabla_x \mathcal{L}(f^\theta(x^{(t)}))$$

Pros:

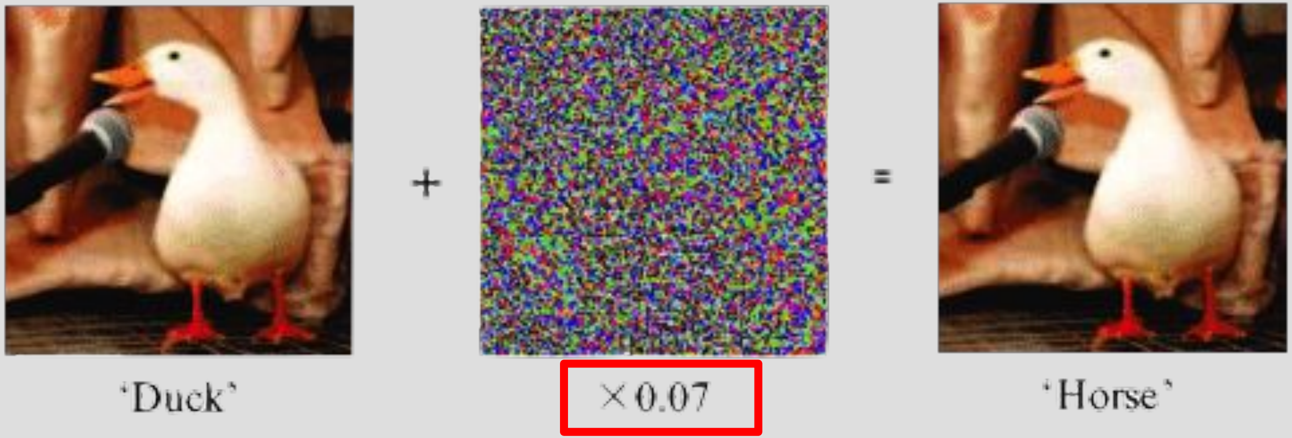
- Easy to implement w/ modern software
- Conceptually simple

Cons:

- No convergence (non-convex + hig-dimensional)
- Expensive as t gets large!



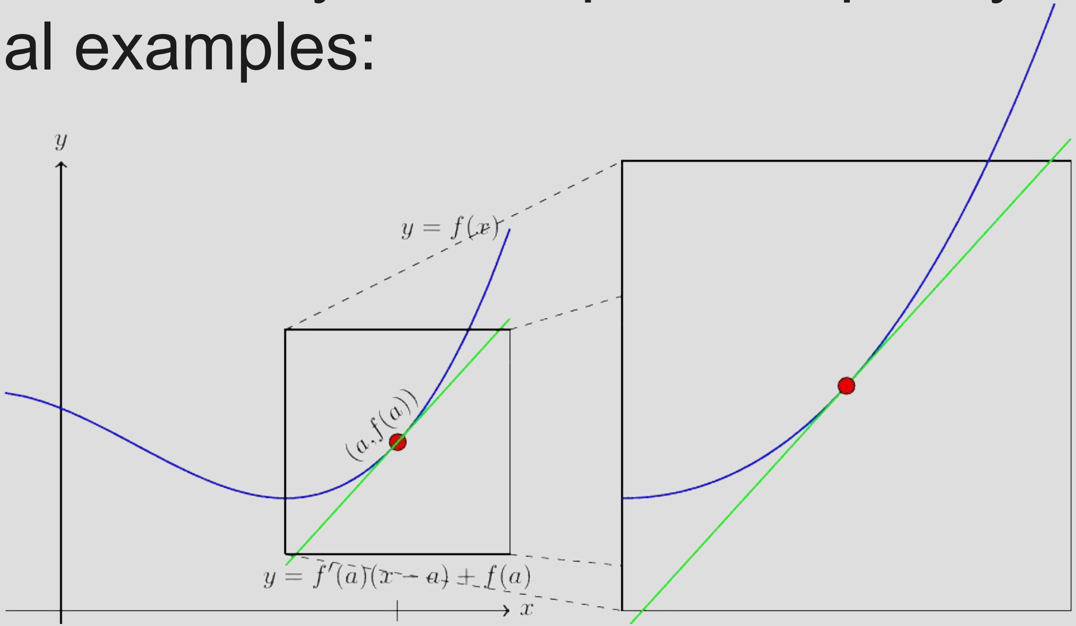
Fast Gradient Sign Method



Insight: sure the function is high-dim and the optimization non-convex, but when epsilon is (really) small, smooth functions all look the same!

Fast Gradient Sign Method

If we are in this small epsilon regime, then how can we leverage the linearity of the space to quickly compute adversarial examples:



Fast Gradient Sign Method

If we are in this small epsilon regime, then how can we leverage the linearity of the space to quickly compute adversarial examples:

$$x^{\text{fgsm}} = x^{(0)} - \epsilon \nabla_x \text{sign}\left(\mathcal{L}(f^{\theta}(x^{(0)}))\right)$$

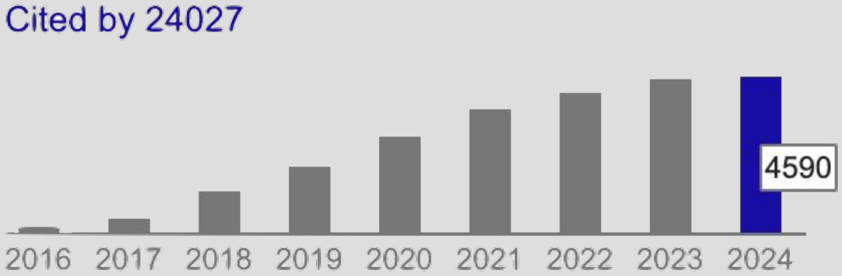
Fast Gradient Sign Method

If we are in this small epsilon regime, then how can we leverage the linearity of the space to quickly compute adversarial examples:

$$x^{\text{fgsm}} = x^{(0)} - \epsilon \nabla_x \text{sign}\left(\mathcal{L}(f^{\theta}(x^{(0)}))\right)$$

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES

Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy
Google Inc., Mountain View, CA
{goodfellow,shlens,szegedy}@google.com



Fast Gradient Sign Method

Contrasting this with the general formulation of gradient descent allows us to appreciate the simplicity of the approach:

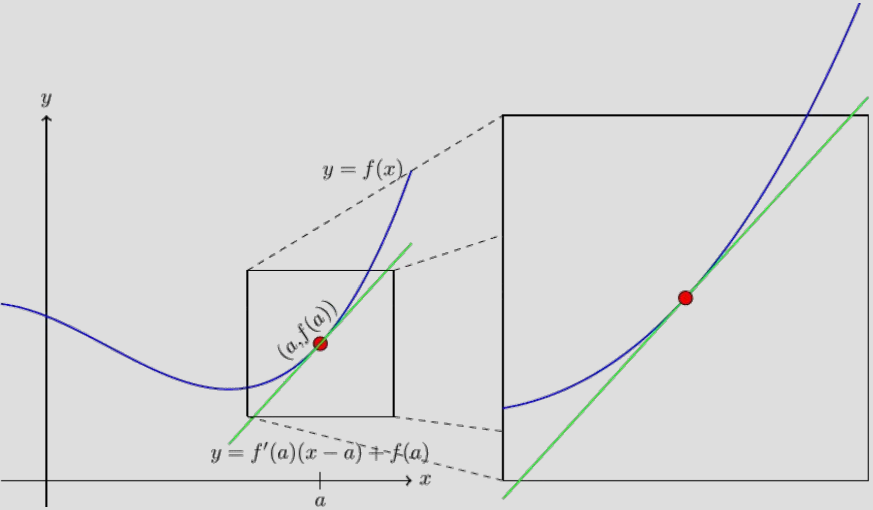
$$x^{(t+1)} = x^{(t)} - \alpha \nabla_x \mathcal{L}(f^\theta(x^{(t)}))$$

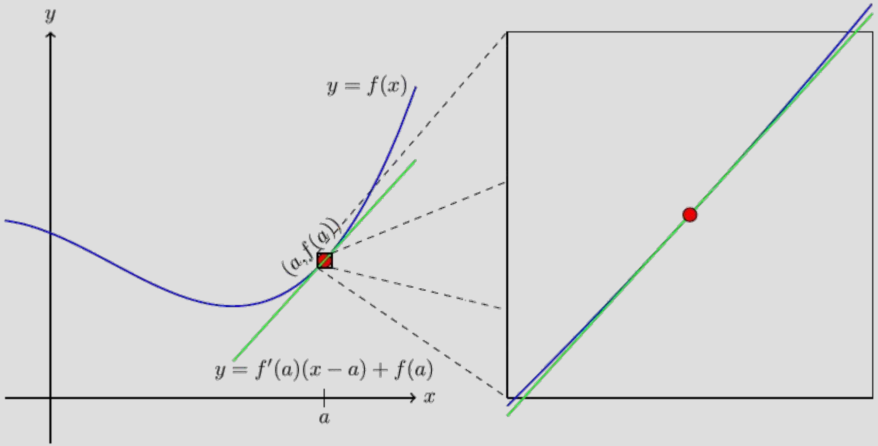
$$x^{\text{fgsm}} = x^{(0)} - \epsilon \nabla_x \text{sign}\left(\mathcal{L}(f^\theta(x^{(0)}))\right)$$

Fast Gradient Sign Method

	Untargeted		Average Case		Least Likely	
	mean	prob	mean	prob	mean	prob
Our L_∞	0.004	100%	0.006	100%	0.01	100%
FGS	0.004	100%	0.064	2%	-	0%
IGS	0.004	100%	0.01	99%	0.03	98%

When the assumption breaks?





Projected Gradient Descent

We bring back many of the assumptions of vanilla gradient descent but we introduce a couple of important modifications

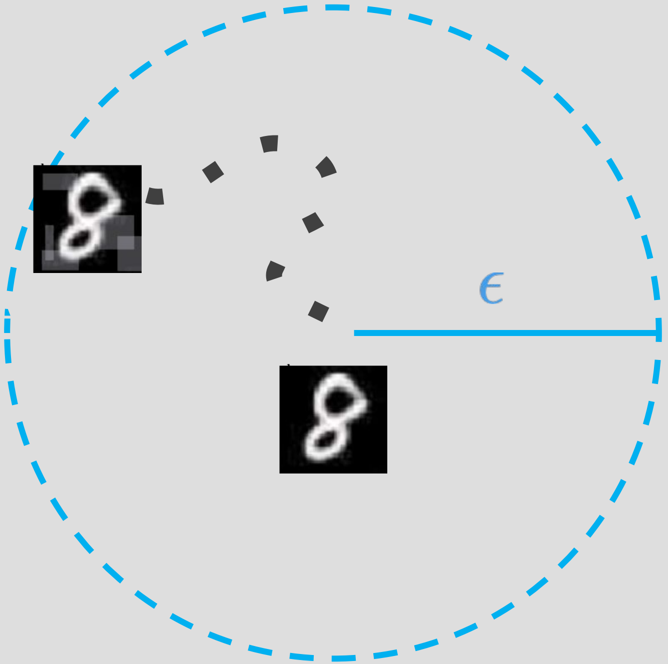
$$x^{(0)} = x + \mathcal{N}(0, \epsilon)$$

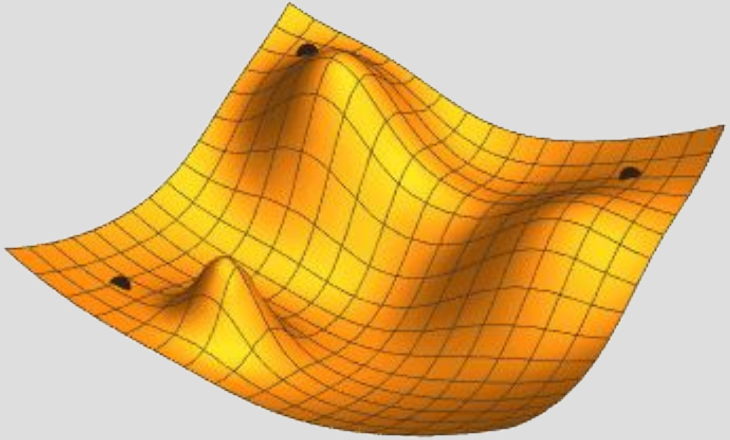
$$\hat{x}^{(t+1)} = x^{(t)} + \alpha \nabla_x \text{sign}\left(\mathcal{L}(f^\theta(x^{(t)}))\right)$$

$$x^{(t+1)} = \text{Proj}_\epsilon(\hat{x}^{(t+1)})$$

What if we do not find an adversarial example?

Does that mean one does not exist?





What would be the goal?

We would like to know that for all inputs inside the ball, a misclassification does not exist. We can convert our optimization problem to this logical formula:

$$\max_{x^\star \in \mathcal{B}_\epsilon(x)} \mathcal{L}(f^\theta(x^\star)) \leq \tau \implies \forall x' \in \mathcal{B}_\epsilon(x), \mathcal{L}(f^\theta(x')) \leq \tau$$

What would be the goal?

We would like to know that for all inputs inside the ball, a misclassification does not exist. We can convert our optimization problem to this logical formula:

$$\max_{x^\star \in \mathcal{B}_\epsilon(x)} \mathcal{L}(f^\theta(x^\star)) \leq \tau \implies \forall x' \in \mathcal{B}_\epsilon(x), \mathcal{L}(f^\theta(x')) \leq \tau$$

Proving such a property is also known as "certification"

Certification of adversarial robustness

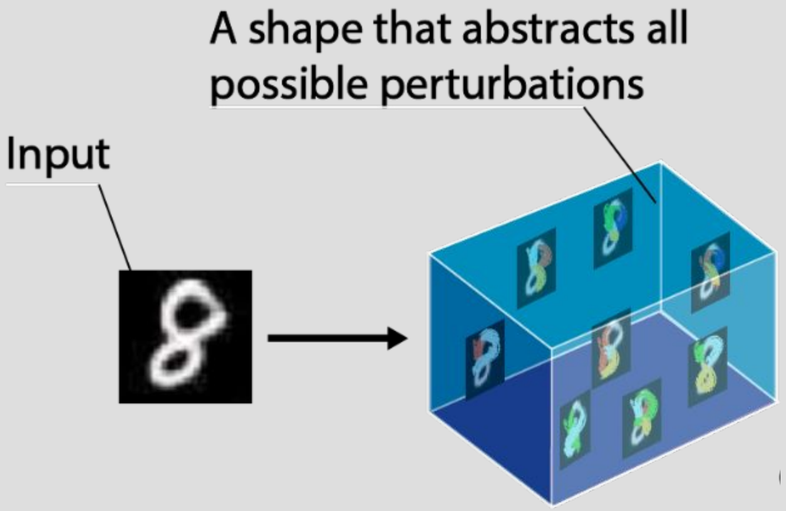


Diagram from https://github.com/eth-sri/eran

Certification of adversarial robustness

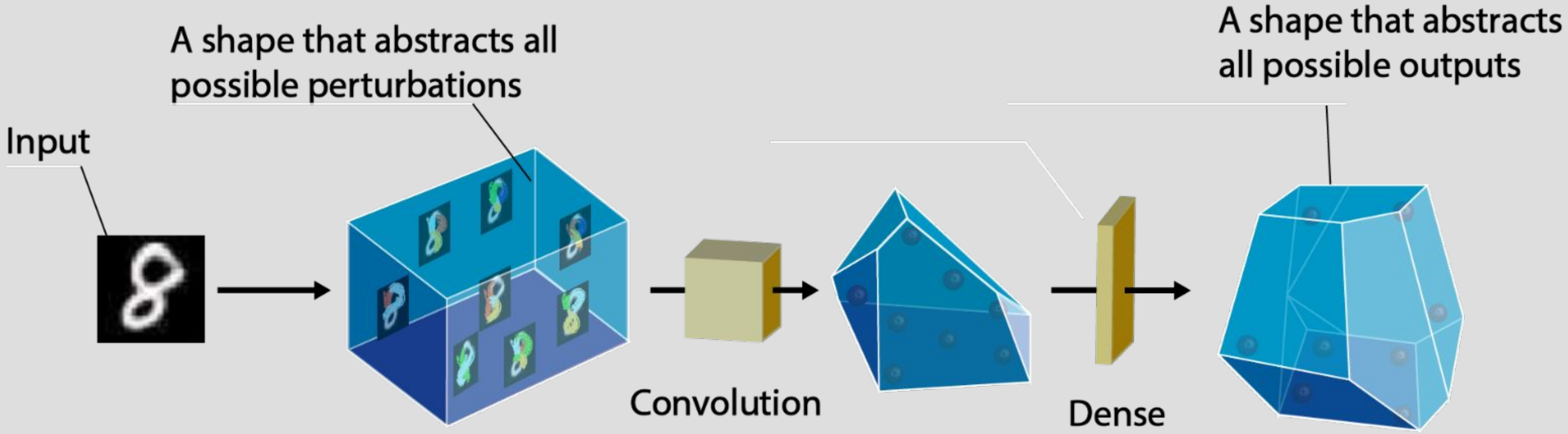


Diagram from https://github.com/eth-sri/eran

Certification of adversarial robustness

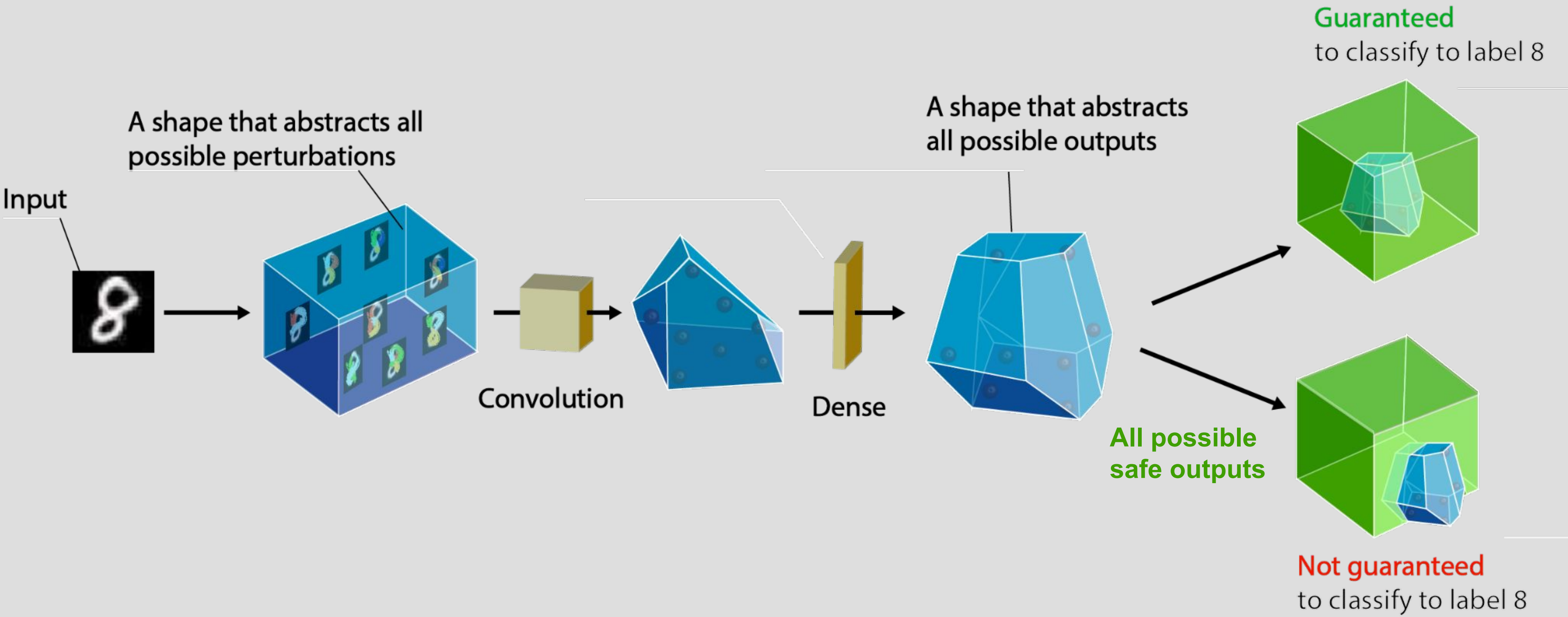
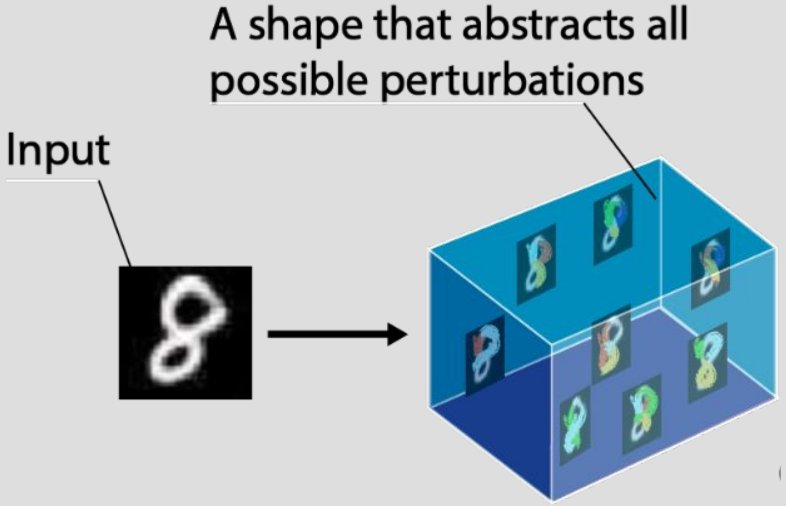


Diagram from https://github.com/eth-sri/eran

Certification of adversarial robustness



$$\boldsymbol{x} \in [\boldsymbol{x} - \epsilon, \boldsymbol{x} + \epsilon]$$

Diagram from https://github.com/eth-sri/eran

Certification of robustness for NNs

Feed-forward NN formulation:

$$\boldsymbol{\zeta}^{(i)} = \boldsymbol{W}^{(i)}\boldsymbol{z} + \boldsymbol{b}^{(i)}$$

$$\boldsymbol{z}^{(i+1)} = \sigma(\boldsymbol{\zeta}^{(i)})$$

Certification of robustness for NNs

Feed-forward NN formulation:

$$\zeta^{(i)} = \boldsymbol{W}^{(i)} \boldsymbol{z} + \boldsymbol{b}^{(i)}$$

Affine transformation

$$\boldsymbol{z}^{(i+1)} = \sigma(\zeta^{(i)})$$

Activation function

Bounding the affine transformation:

$$z^{(i-1)} \in [z_L^{(i-1)}, z_U^{(i-1)}]$$

$$z^{(i-1)} \in [2, 4] \quad W^{(i)} = 5$$

$$\implies W^{(i)} z^{(i-1)} \in \quad [?, ?]$$

Bounding the affine transformation:

$$z^{(i-1)} \in [z_L^{(i-1)}, z_U^{(i-1)}]$$

$$z^{(i-1)} \in [2, 4] \quad W^{(i)} = 5$$

$$\implies W^{(i)} z^{(i-1)} \in [10, 20]$$

Bounding the affine transformation (generalized result):

$$z^{(i-1)} \in [2, 4] \quad W^{(i)} = 5$$

$$z_{\mu}^{(i-1)} = 3 \quad z_{r}^{(i-1)} = 1$$

Bounding the affine transformation (generalized result):

$$z^{(i-1)} \in [2,4] \quad W^{(i)} = 5$$

$$z_{\mu}^{(i-1)} = 3 \quad z_{r}^{(i-1)} = 1$$

$$W^{(i)} z_{\mu}^{(i-1)} - |W^{(i)}| z_{r}^{(i-1)}$$

Bounding the affine transformation (generalized result):

$$z^{(i-1)} \in [2, 4] \quad W^{(i)} = 5$$

$$z_{\mu}^{(i-1)} = 3 \quad z_{r}^{(i-1)} = 1$$

$$W^{(i)} z_{\mu}^{(i-1)} - |W^{(i)}| z_{r}^{(i-1)}$$

$$5(3) - 5(1) = 10$$

Bounding the affine transformation (generalized result):

$$z^{(i-1)} \in [2, 4] \quad W^{(i)} = 5$$

$$z_{\mu}^{(i-1)} = 3 \quad z_{r}^{(i-1)} = 1$$

$$W^{(i)} z_{\mu}^{(i-1)} - |W^{(i)}| z_{r}^{(i-1)} \qquad W^{(i)} z_{\mu}^{(i-1)} + |W^{(i)}| z_{r}^{(i-1)}$$

$$5(3) - 5(1) = 10 \qquad 5(3) + 5(1) = 20$$

Bounding the affine transformation (generalized result):

$$z^{(i-1)} \in [2, 4] \quad W^{(i)} = 5$$

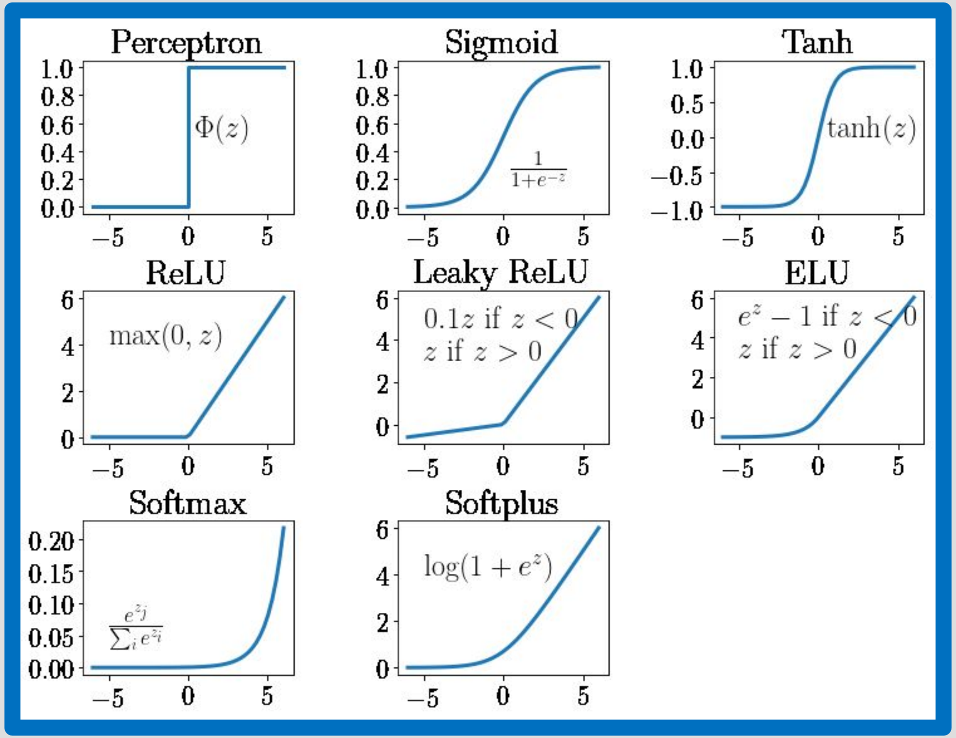
$$z_{\mu}^{(i-1)} = 3 \quad z_{r}^{(i-1)} = 1$$

$$W^{(i)} z_{\mu}^{(i-1)} - |W^{(i)}| z_{r}^{(i-1)} \qquad W^{(i)} z_{\mu}^{(i-1)} + |W^{(i)}| z_{r}^{(i-1)}$$

$$5(3) - 5(1) = 10 \qquad 5(3) + 5(1) = 20$$

Mini-exercise: use summation notation to prove this bound generalizes to matrix multiplication!

Bounding the activation function:



Bounding the activation function:

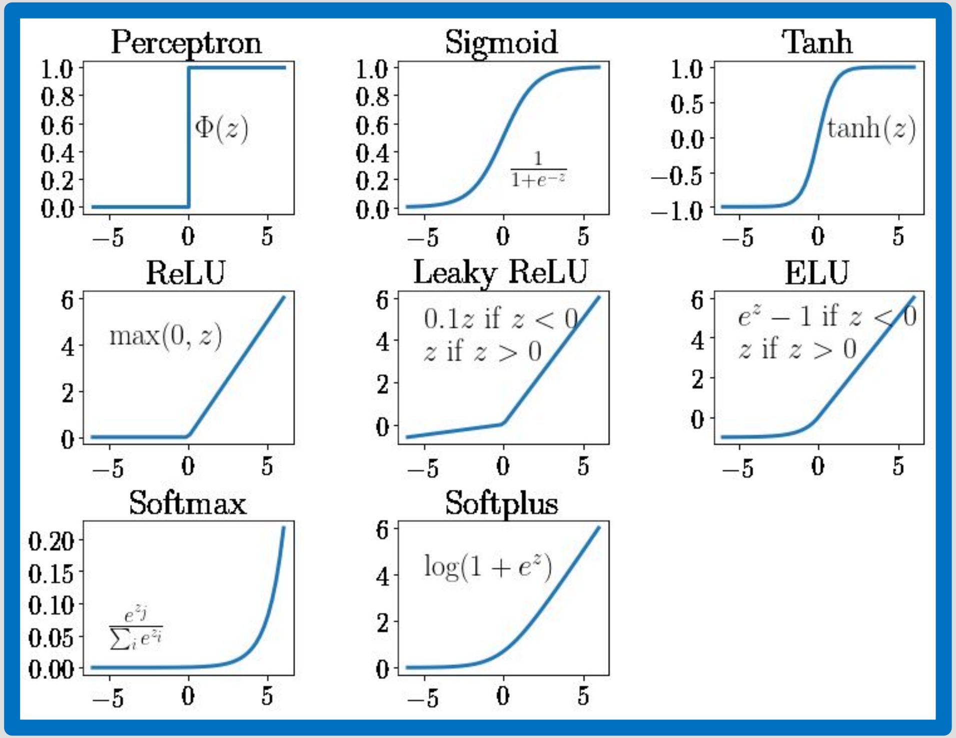
Each of the common activation functions is monotonic:

$$x \leq y \implies \sigma(x) \leq \sigma(y).$$

This means that:

$$\zeta \leq \zeta^U \implies \sigma(\zeta) \leq \sigma(\zeta^U)$$

$$\zeta^L \leq \zeta \implies \sigma(\zeta^L) \leq \sigma(\zeta)$$



Interval Bound Propagation

$$z_c^{(i)} = \frac{\boxed{z_l^{(i)}} + \boxed{z_u^{(i)}}}{2}, \quad z_r^{(i)} = \frac{\boxed{z_u^{(i)}} - \boxed{z_l^{(i)}}}{2}$$

$$\boxed{\zeta_u^{(i)} = W^{(i)}\, z_c^{(i)} + b^{(i)} + \left|W^{(i)}\right| z_r^{(i)}}$$

$$\boxed{\zeta_l^{(i)} = W^{(i)}\, z_c^{(i)} + b^{(i)} - \left|W^{(i)}\right| z_r^{(i)}}$$

$$\boxed{z_l^{(i+1)} = \sigma\left(\zeta_l^{(i)}\right)}, \quad \boxed{z_u^{(i+1)} = \sigma\left(\zeta_u^{(i)}\right)}$$

Certification of adversarial robustness

Dataset	Model (Same settings as [35, 37, 26])	β-CROWN FSB Ver.%	β-CROWN FSB Time(s)	Upper bound
MNIST	MLP $5 \times 100^{\ddagger}$	**69.9**	102	84.2
	MLP 8×100	**62.0**	103	82.0
	MLP 5×200	**77.4**	86	90.1
	MLP 8×200	**73.5**	95	91.1
	ConvSmall	**72.7**	7.0	73.2
	ConvBig	**79.3**	3.1	80.4
CIFAR	ConvSmall	**46.3**	6.8	48.1
	ConvBig	**51.6**	15.3	55.0
	ResNet	**24.8**	1.6	24.8

Harnessing Attacks and Certification

$$x^{\text{adv}} = \text{Attack}(x, \epsilon)$$

$$y^{\text{worst}} = f^{\theta}(x^{\text{adv}})$$

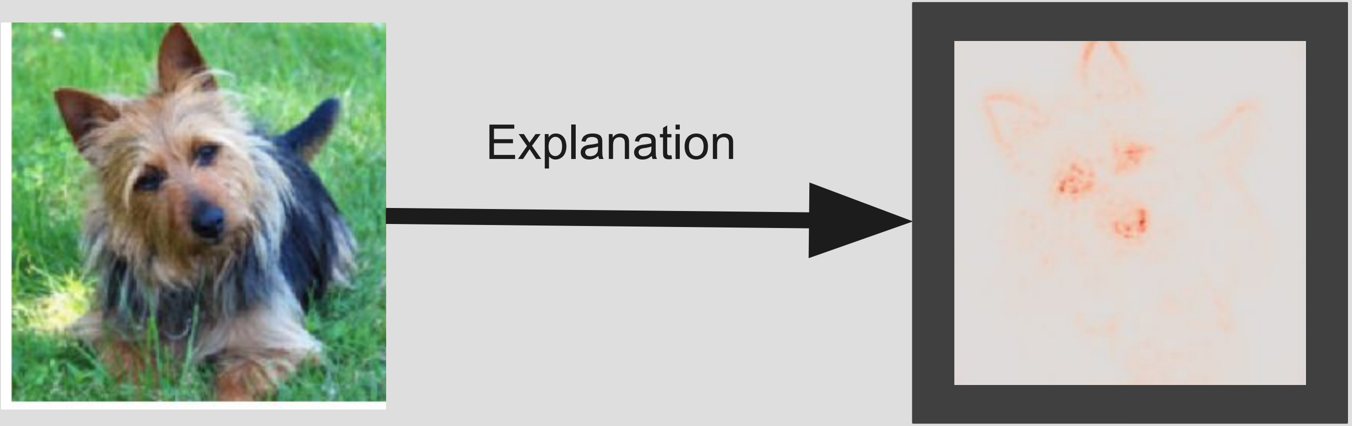
$$y^{\text{worst}} = \text{Certify}(x, \epsilon)$$

Adversarially Robust Loss:

$$\min_{\theta \in \Theta} \gamma \mathcal{L}(y, \hat{y}) + (1 - \gamma)\mathcal{L}(y, y^{\text{worst}})$$

Advanced Material: Moving Beyond Robustness

Certifying explainability for NNs

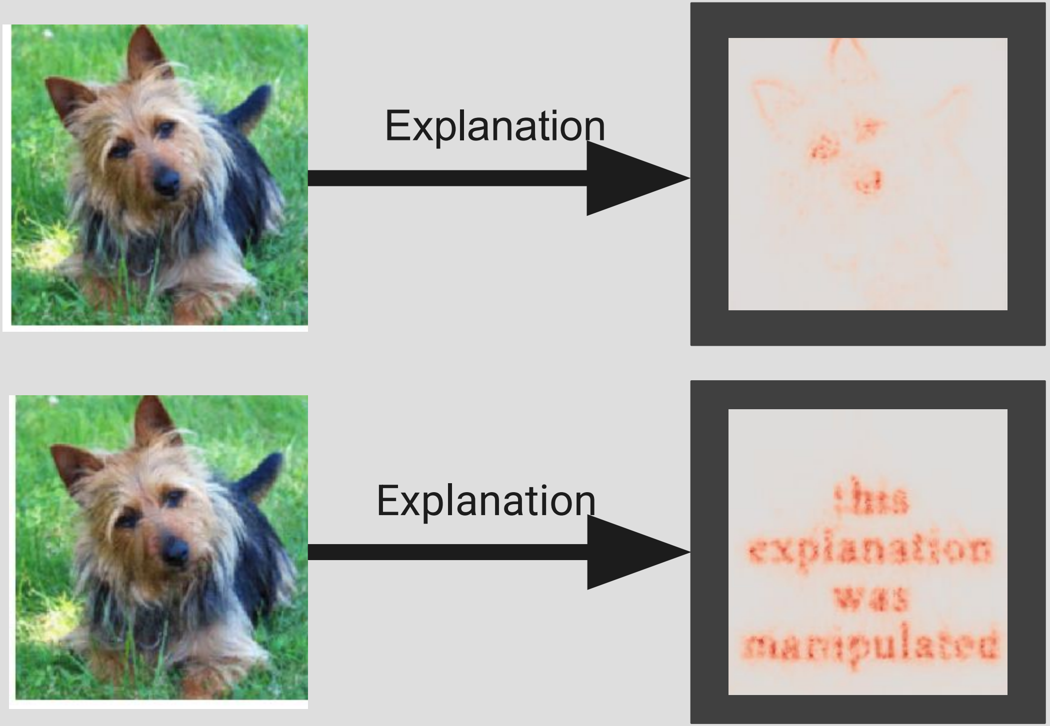


Explainability methods enable us to understand *why* a neural network makes its predictions

A common source of explanations is the *input gradient*

[Dombrowski, et. al. 2020]

Certifying explainability for NNs



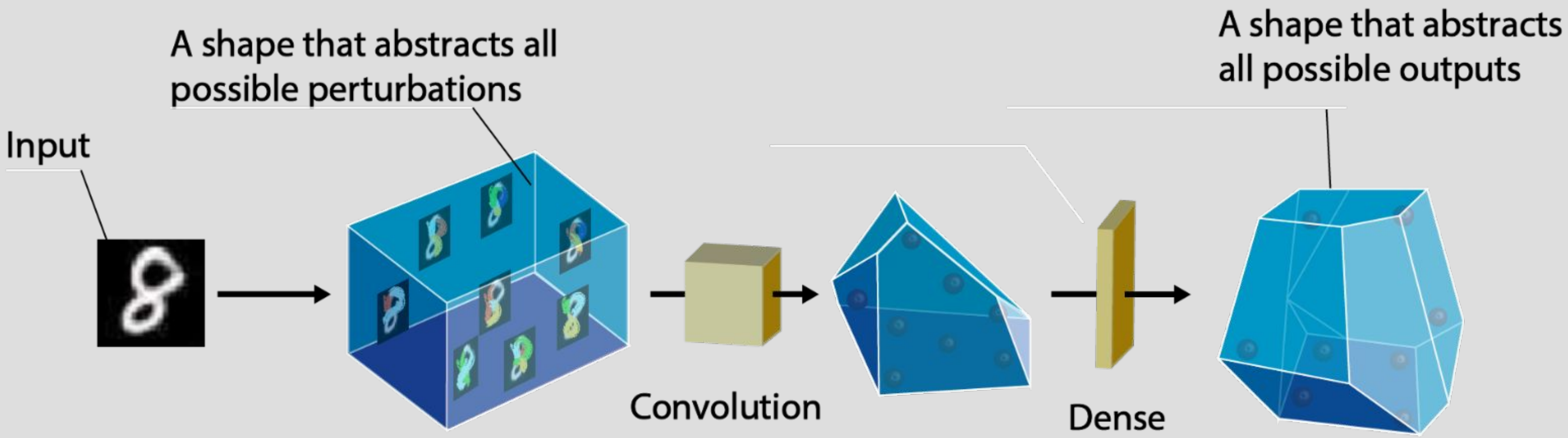
For highly non-linear models, this can be uninformative/misleading

[Dombrowski, et. al. 2020]

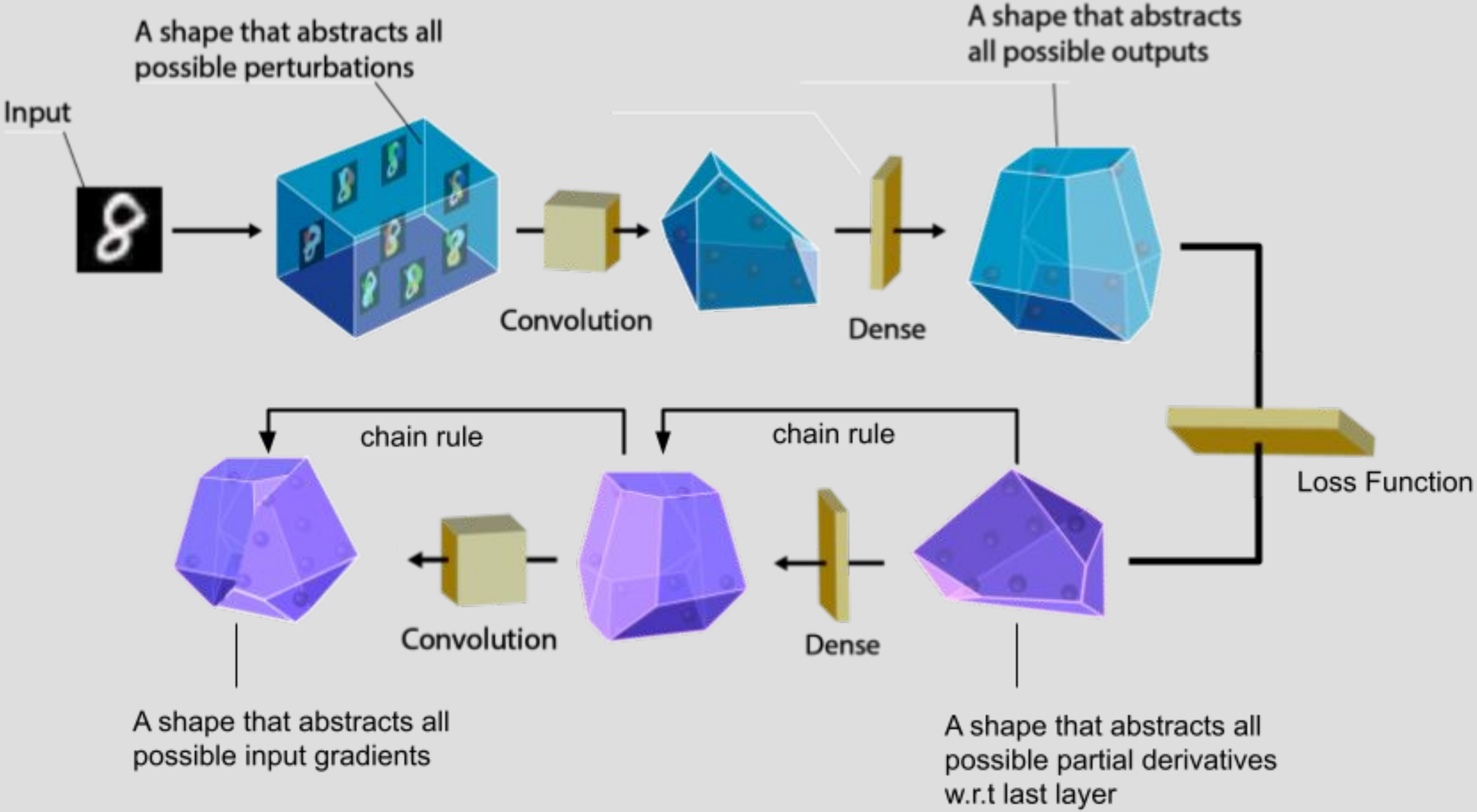
Defining robustness of explanations

$$\left\| \min_{x \in T} \frac{\partial \mathcal{L}}{\partial x_i} - \max_{x \in T} \frac{\partial \mathcal{L}}{\partial x_i} \right\| \leq \delta_i$$

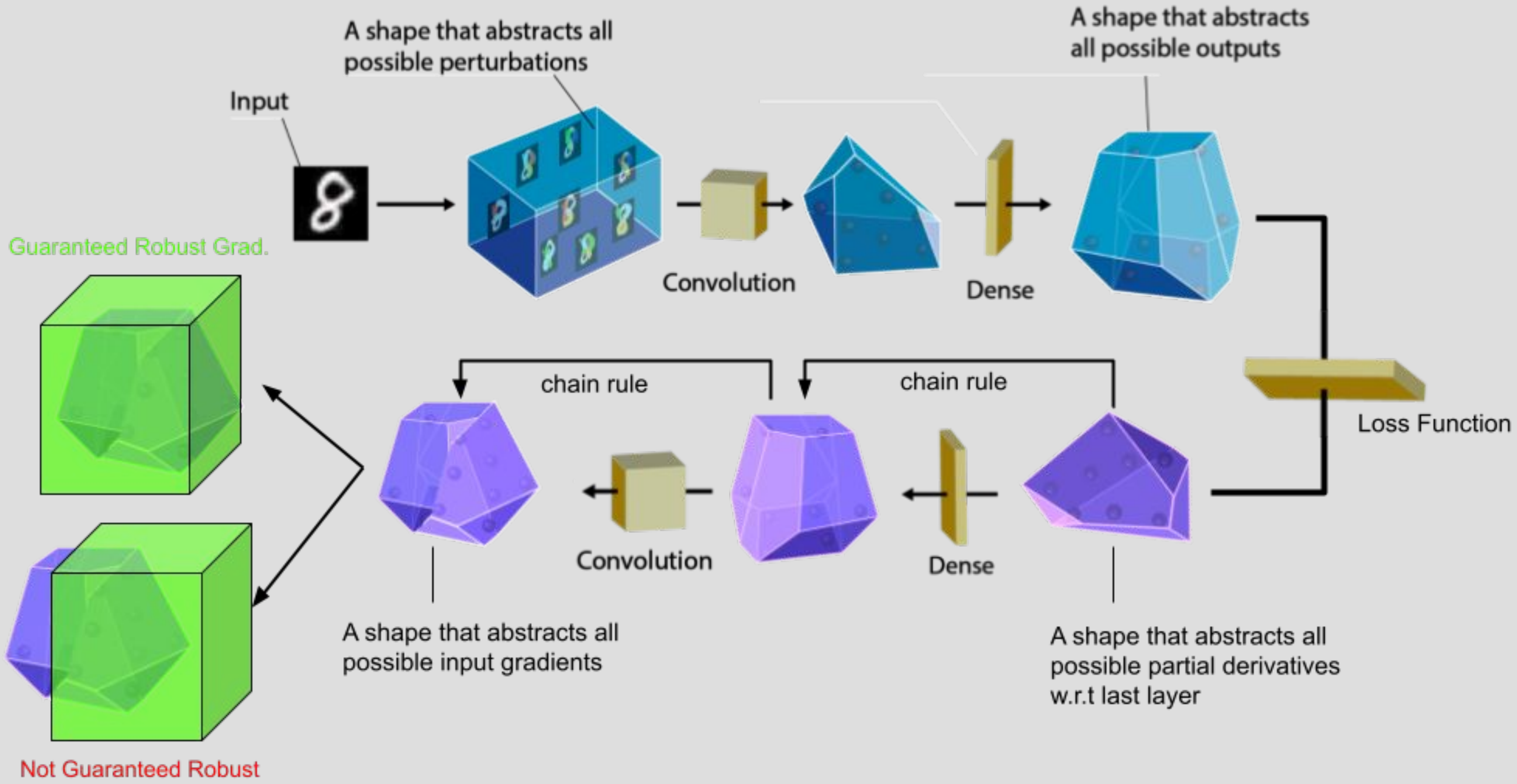
We say the input gradient is robust if the difference between the maximum and minimum values are bounded below a pre-specified value



base figure from: https://github.com/eth-sri/eran, https://ggndpsngh.github.io/files/DeepPoly.pdf

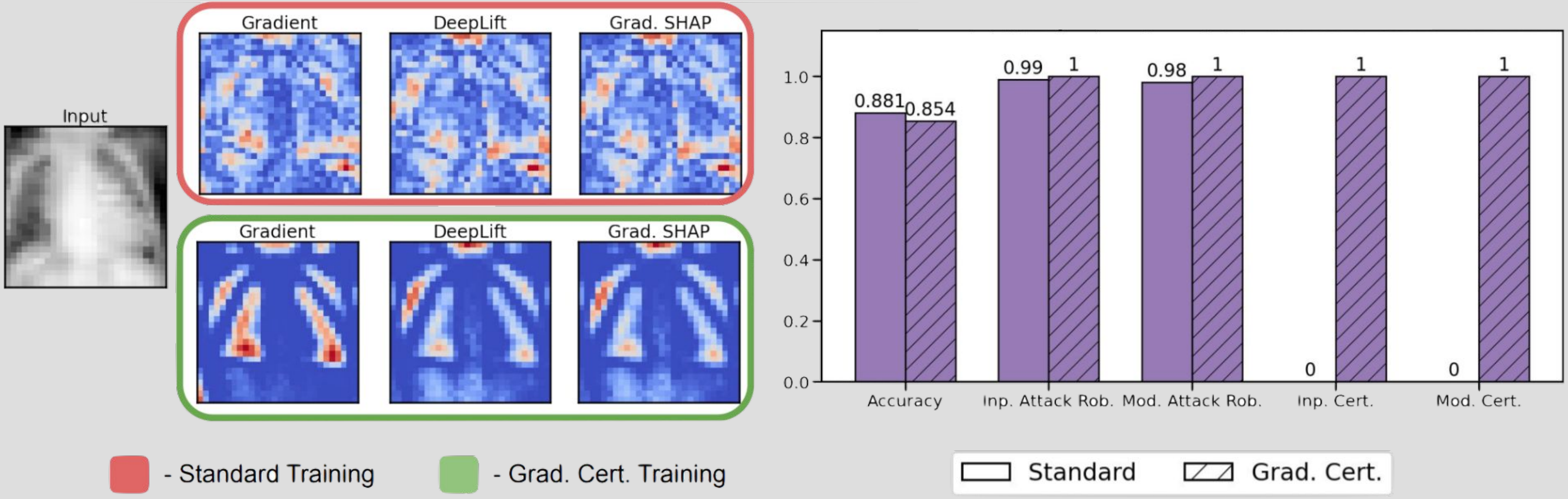


Chain rule leads to quadratic optimization, which needs new propagation techniques [Wicker, et. al., ICLR 2023]

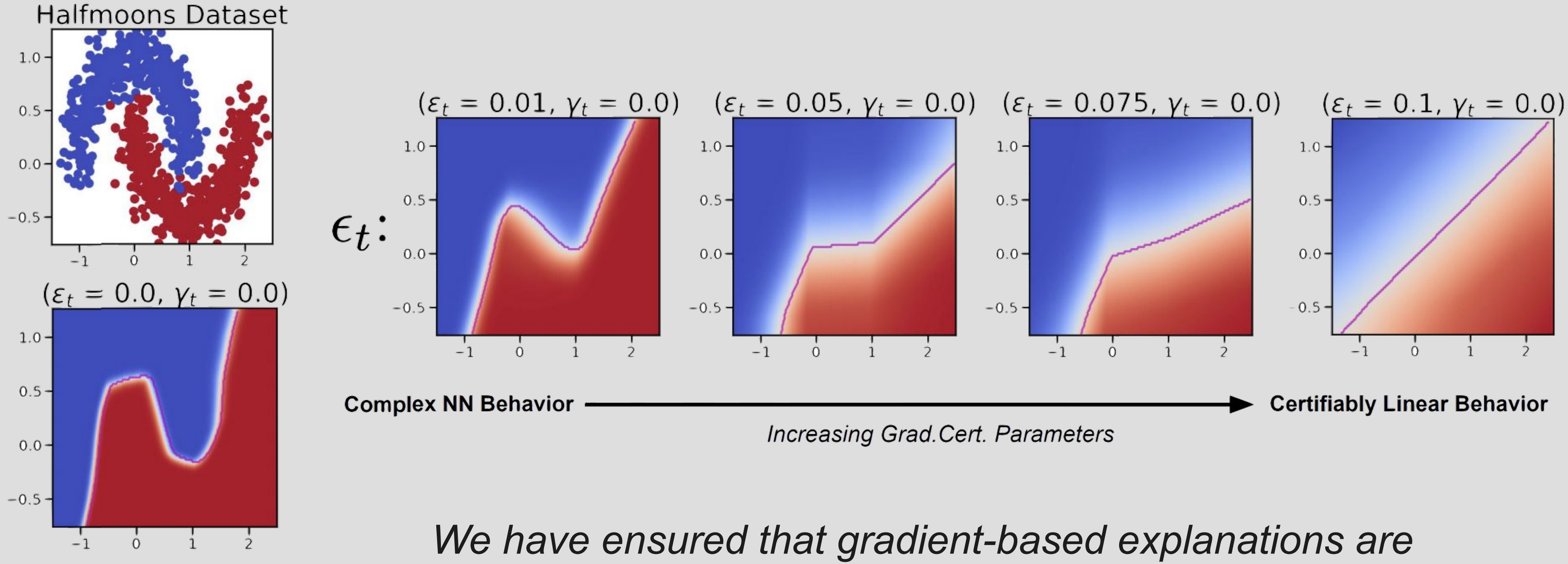


Chain rule leads to quadratic optimization, which needs new propagation techniques [Wicker, et. al., ICLR 2023]

Result of certified robust gradient training

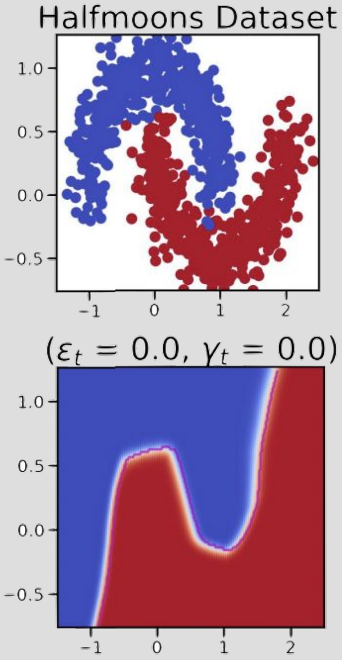


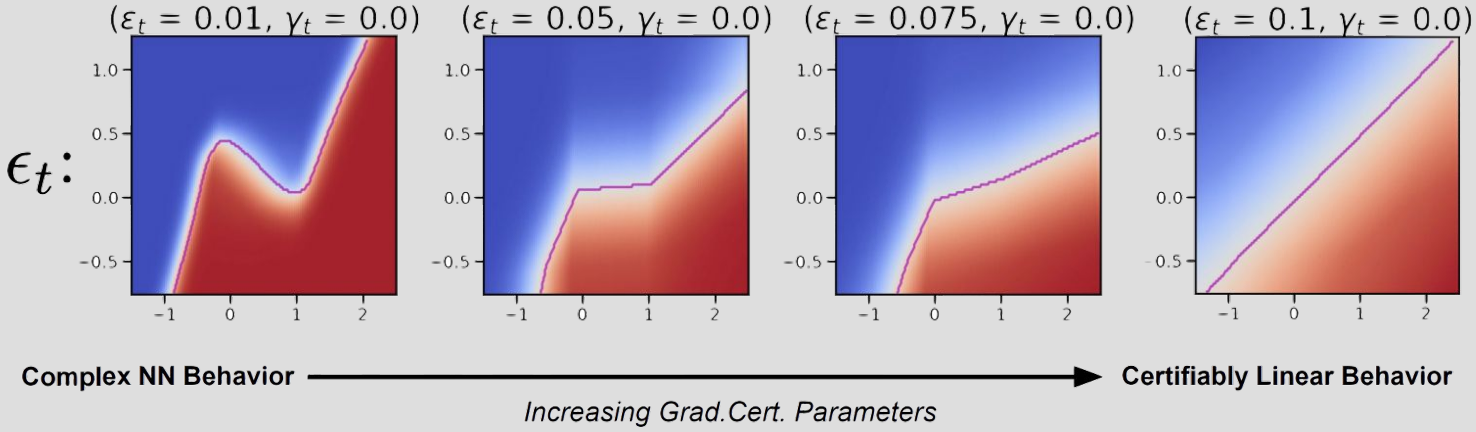
Why certified robust gradient training works



We have ensured that gradient-based explanations are provably (locally) calibrated

Why certified robust gradient training works

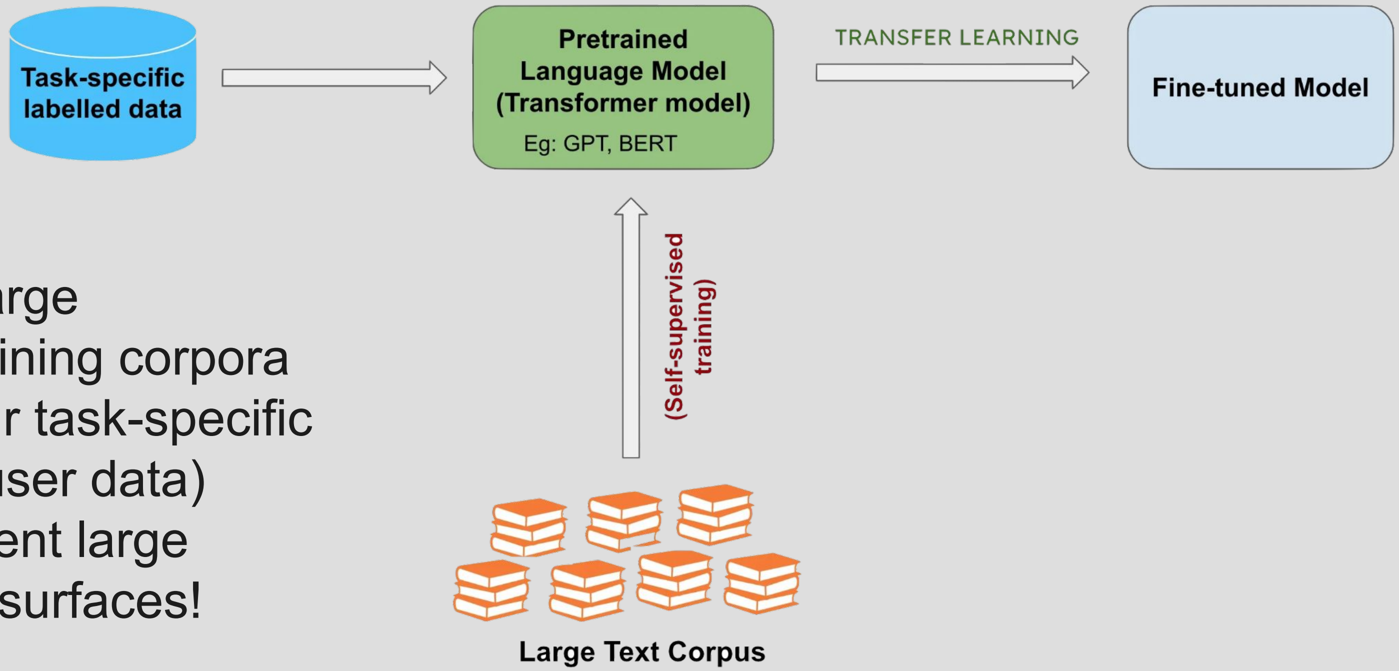




This work has direct links to the AAAI 2023 work by Junqi Jiang, Francesco Leofante, Antonio Rago, Francesca Toni and the CLARG group at Imperial DoC

Advanced Material: Abstract Gradient Training

Modern Training Pipelines



Both large pre-training corpora and our task-specific data (user data) represent large attack surfaces!

Attacks on Model Training

Injecting training data allows us to easily fool models w backdoors:

(a) Static backdoor

(b) Dynamic backdoor

Let's model this as before

$$\mathcal{D}^{\star} = \underset{\mathcal{D}' \in \mathcal{B}_{\epsilon,k}(\mathcal{D})}{\arg\max} \boxed{\mathcal{L}\left(M(\mathcal{D}')\right)}$$

- Loss function - The result of training algorithm M

Let's model this as before

$$\mathcal{D}^{\star} = \underset{\boxed{\mathcal{D}' \in \mathcal{B}_{\epsilon,k}(\mathcal{D})}}{\arg\max} \boxed{\mathcal{L}\left(M(\mathcal{D}')\right)}$$

- Loss function - The result of training algorithm M
- Epsilon Ball - The set of "similar" datasets

Let's model this as before

$$\mathcal{D}^\star = \boxed{\arg\max}_{\boxed{\mathcal{D}' \in \mathcal{B}_{\epsilon,k}(\mathcal{D})}} \boxed{\mathcal{L}\left(M(\mathcal{D}')\right)}$$

- Loss function - The result of training algorithm M
- Epsilon Ball - The set of "similar" datasets
- Argmax - the dataset from the ball - max. the loss

Let's model this as before

$$\boxed{\mathcal{D}^\star} = \boxed{\arg\max}_{\boxed{\mathcal{D}' \in \mathcal{B}_{\epsilon,k}(\mathcal{D})}} \boxed{\mathcal{L}\left(M(\mathcal{D}')\right)}$$

- Loss function - The result of training algorithm M
- Epsilon Ball - The set of "similar" datasets
- Argmax - the dataset from the ball - max. the loss
- An approximate worst-case example

Defining the learning algorithm

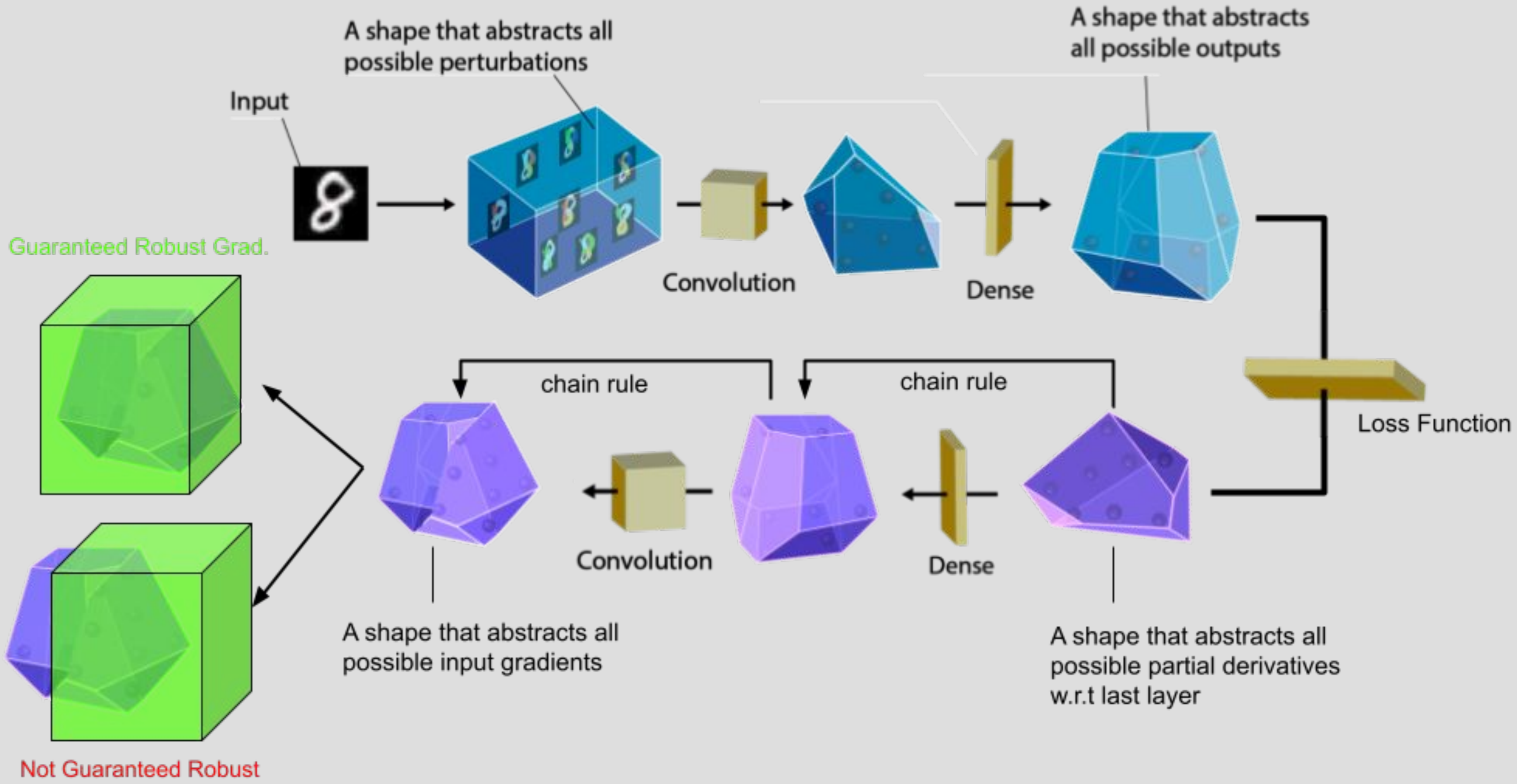
$$M(\mathcal{D}) := \theta^{(T)}$$

$$\theta^{(t+1)} = \theta^{(t)} - \nabla_\theta \mathcal{L}(f^{\theta^{(t)}}(\mathcal{D}))$$

Defining the learning algorithm

$$M(\mathcal{D}) := \theta^{(T)}$$

$$\theta^{(t+1)} = \theta^{(t)} - \underbrace{\nabla_\theta \mathcal{L}(f^{\theta^{(t)}}(\mathcal{D}))}_{\text{We need to figure out the maximum and minimum of this term}}$$



Chain rule leads to quadratic optimization, which needs new propagation techniques [Wicker, et. al., ICLR 2023]

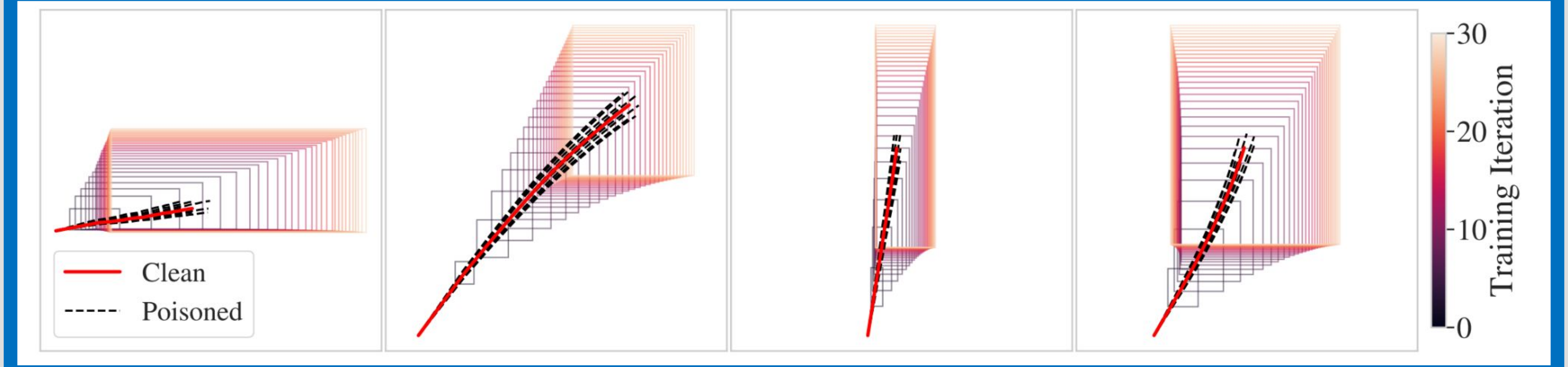
Bounding the result of the learning algo.

$$M(\mathcal{D}) := \theta^{(T)}$$

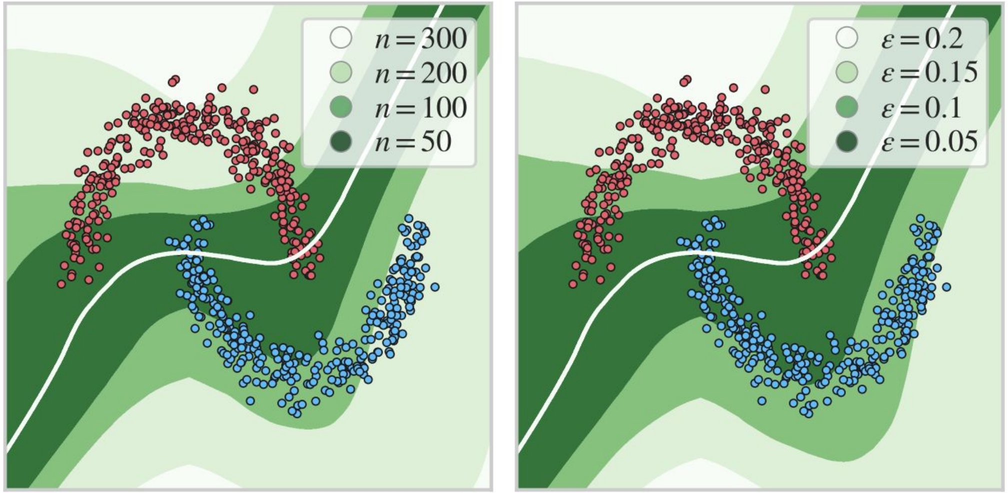
$$\theta^{(t+1),U} = \theta^{(t),U} - \nabla_\theta^L \mathcal{L}(f^{\theta^{(t)}}(\mathcal{D}))$$

$$\theta^{(t+1),L} = \theta^{(t),L} - \nabla_\theta^U \mathcal{L}(f^{\theta^{(t)}}(\mathcal{D}))$$

Bounding the result of the learning algo.



Bounding the result of the learning algo.



(a) Feature poisoning ($\varepsilon=0.01, m=0$)

(b) Feature poisoning ($n=10, m=0$)

Bounding the result of the learning algo.