Master Data Science – Advanced ML Topics

Session: Adversarial Machine Learning

Presented by:

Dr. Karim Tit & Mr. Mohamed Djilani

University of Luxembourg

Previous slides from:

Dr. Salah GHAMIZI,

Luxembourg Institute of Science and Technology

Hello!

I am Karim TIT

Postdoctoral researcher @ University of Luxembourg

Email for questions (slides 1-53): karim.tit@uni.lu

Hello!

I am Mohamed Djilani

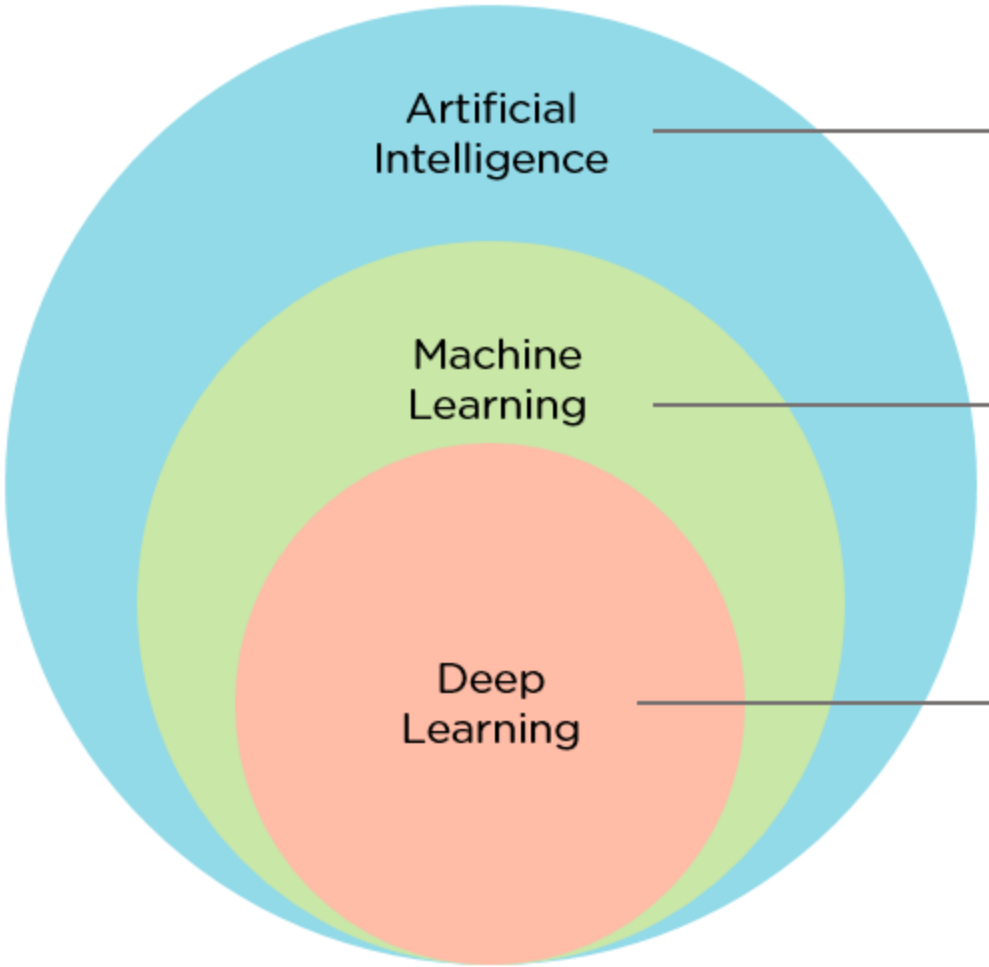
PhD student @ University of Luxembourg

Email for questions (slides 54-100):
mohamed.djilani@uni.lu

Machine Learning - Quick Recap

Part 0

AI / ML / DL ? Kesako?

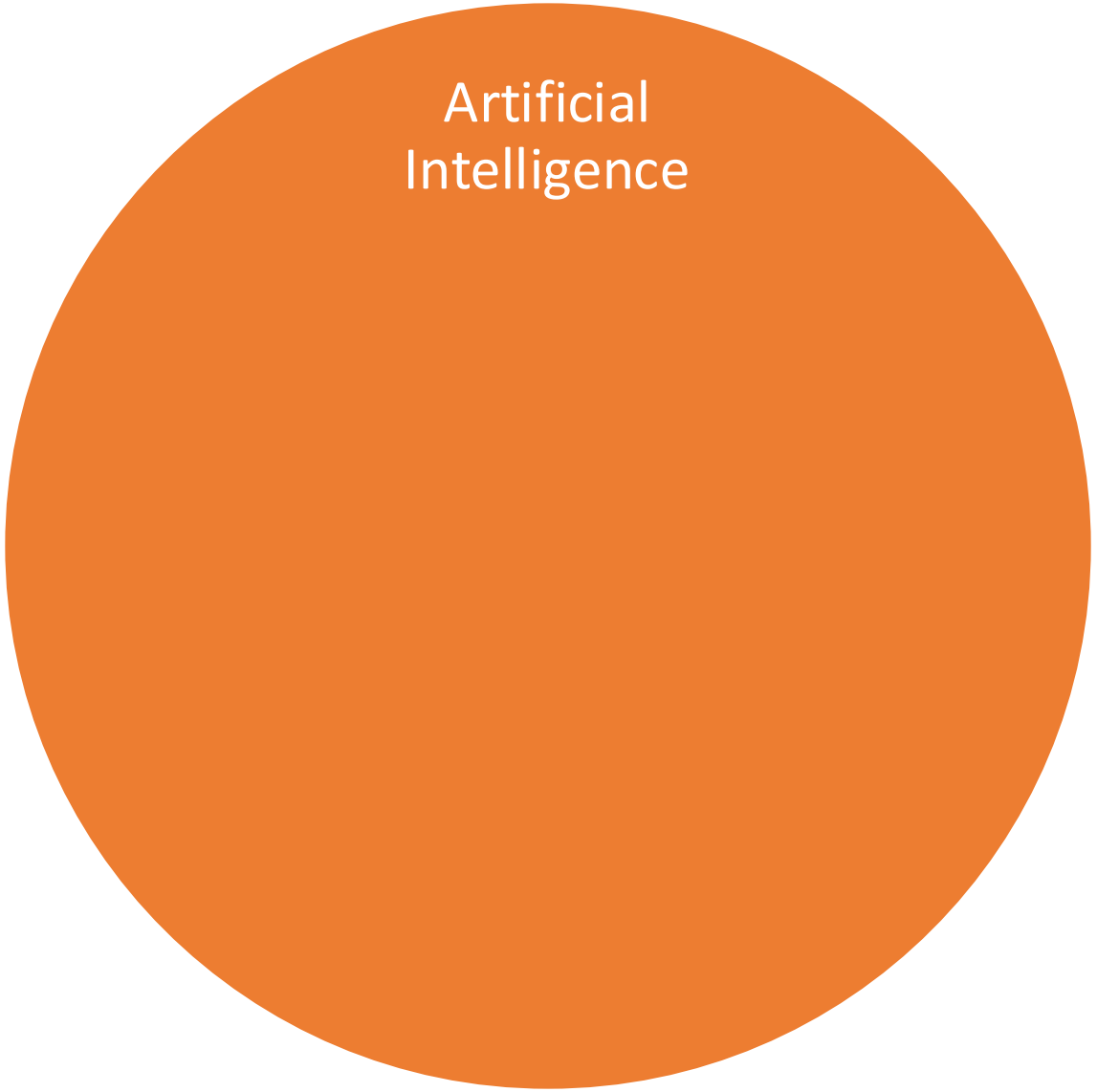


Ability of a machine to imitate intelligent human behavior

Application of AI that allows a system to automatically learn and improve from experience

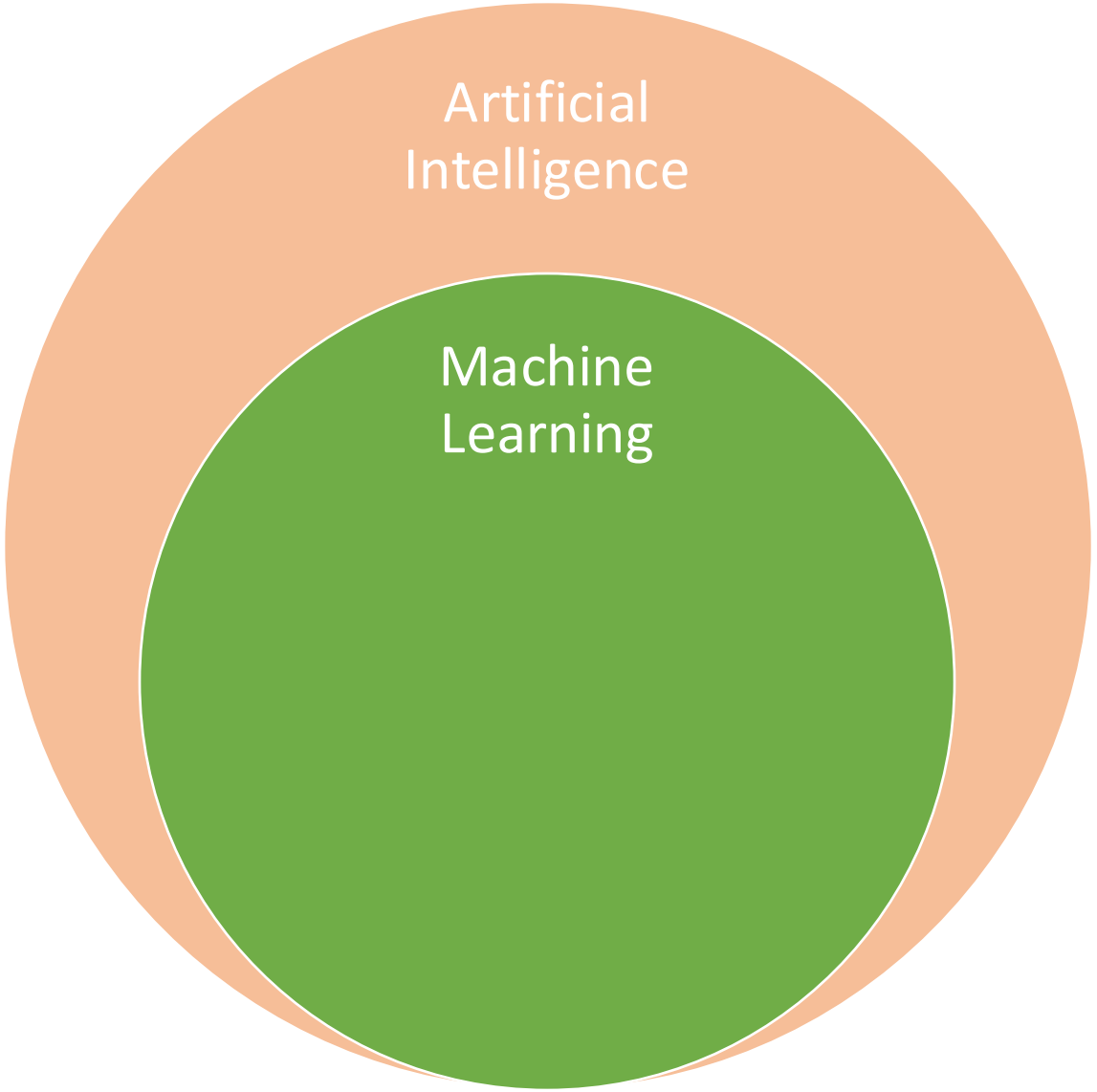
Application of Machine Learning that uses complex algorithms and deep neural nets to train a model

AI / ML / DL ? Kesako?



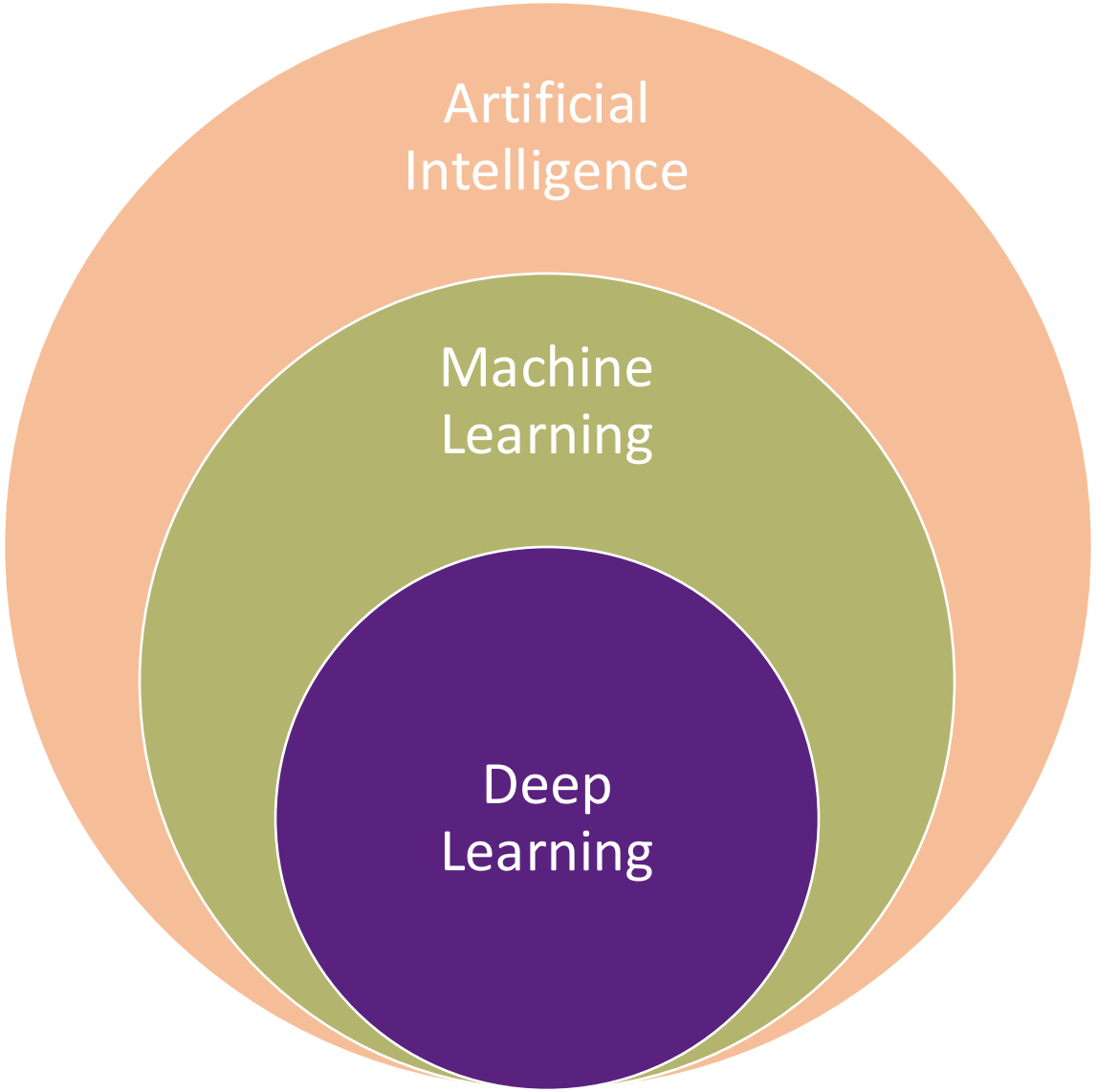
Ability of a machine to **emulate** intelligent human behaviour and solving complex tasks

AI / ML / DL ? Kesako?



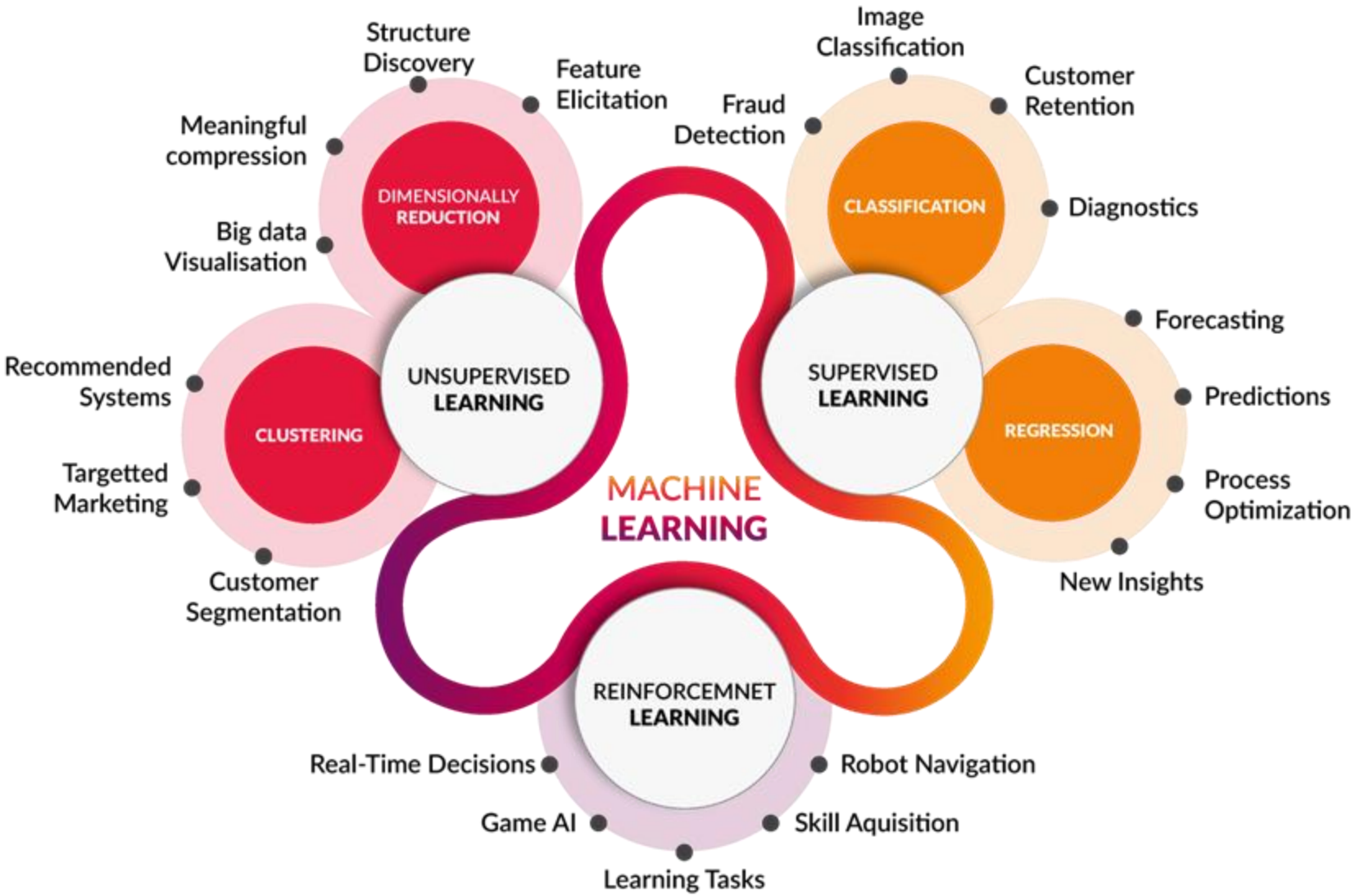
A branch of artificial intelligence where algorithms learn the hidden patterns of data to make predictions on new similar data, without being **explicitly** programmed for each task.

AI / ML / DL ? Kesako?

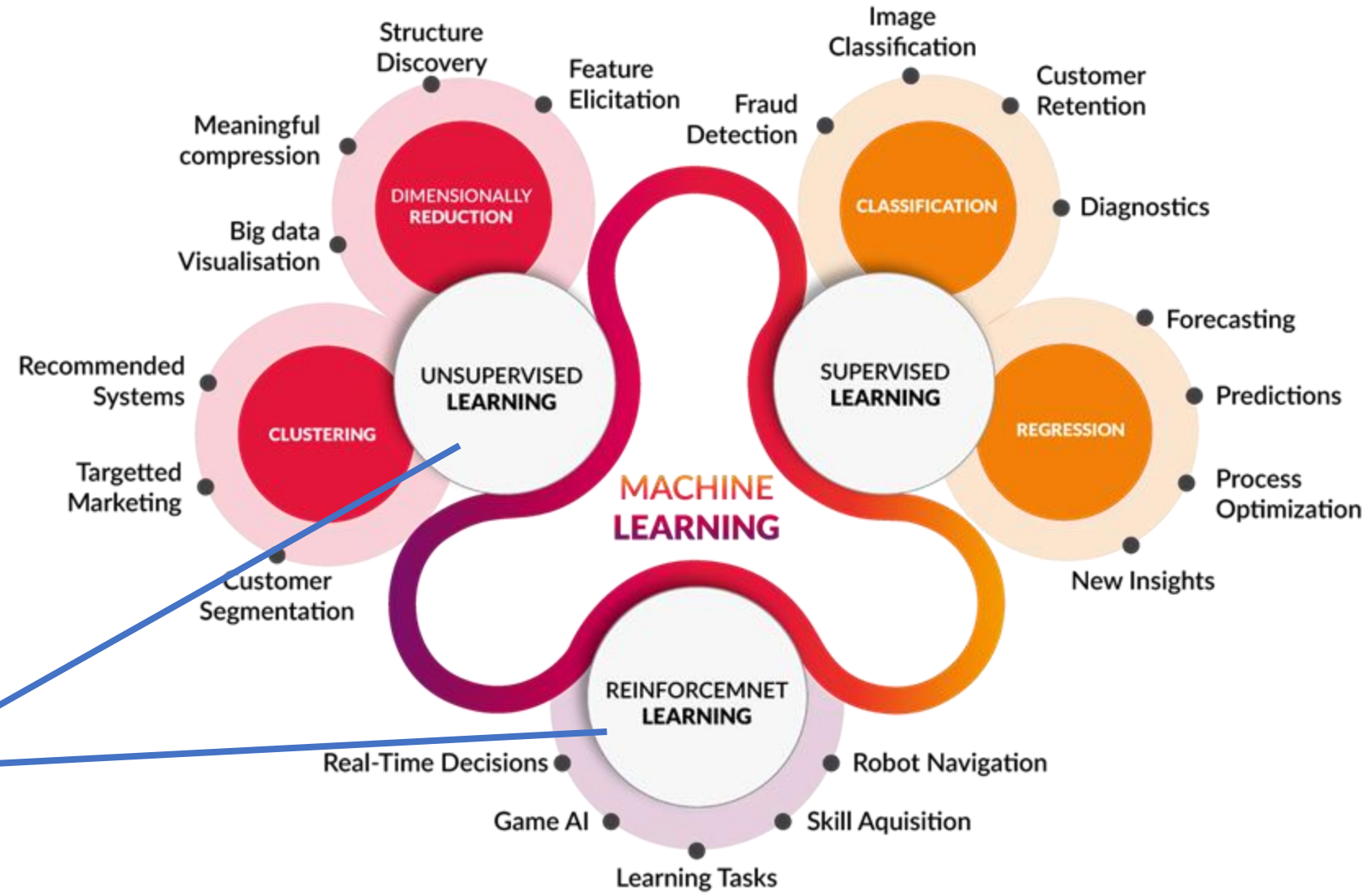


Set of Machine Learning techniques that use intricate algorithms and **deep neural networks** to train a model

ML Taxonomy

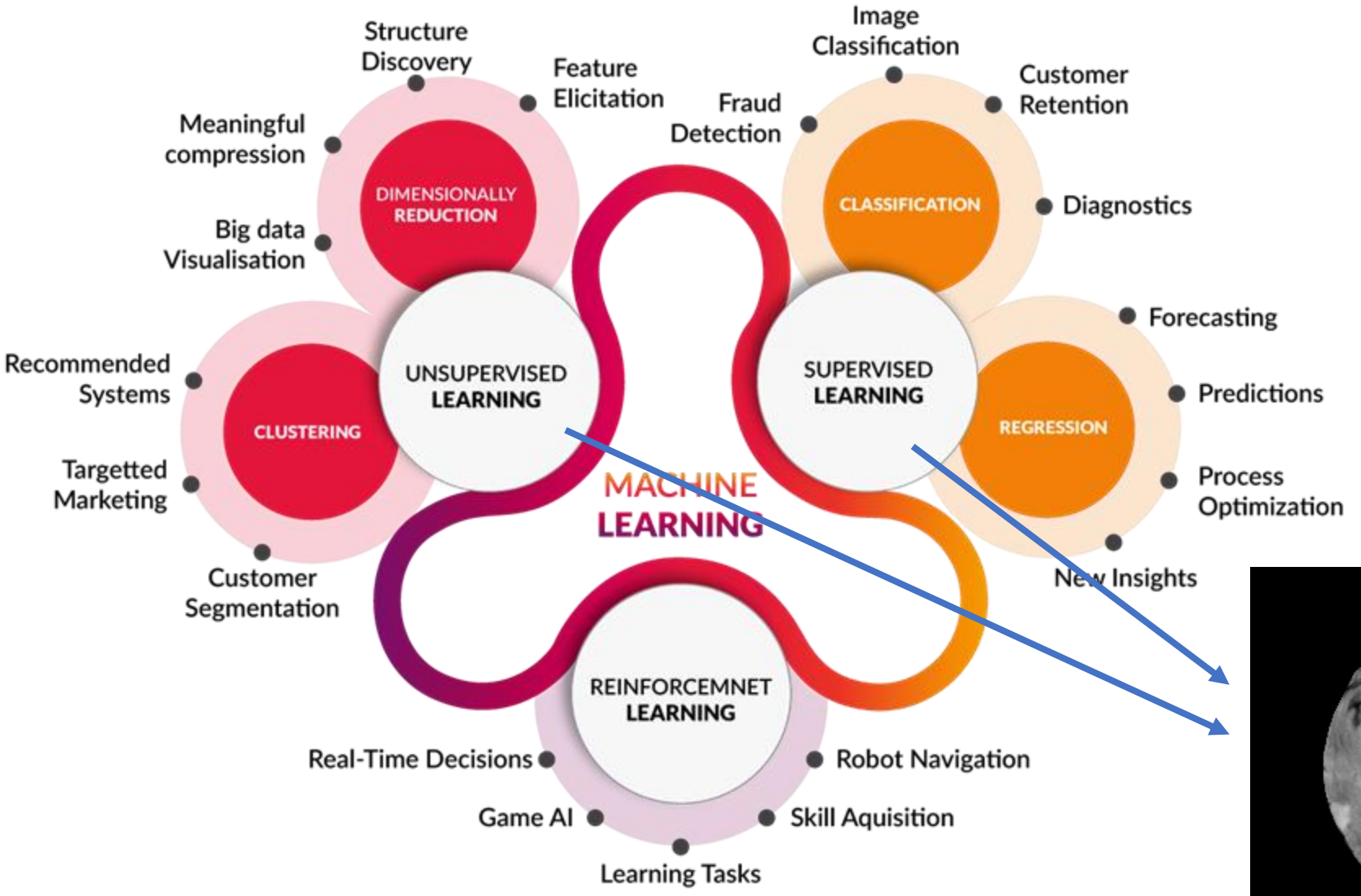

MACHINE LEARNING
UNSUPERVISED LEARNING
SUPERVISED LEARNING
REINFORCEMNET LEARNING
DIMENSIONALLY REDUCTION
CLUSTERING
CLASSIFICATION
REGRESSION
Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
Recommended Systems
Targetted Marketing
Customer Segmentation
Image Classification
Customer Retention
Fraud Detection
Diagnostics
Forecasting
Predictions
Process Optimization
New Insights
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

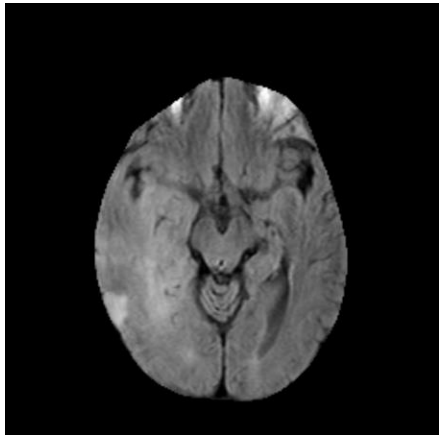
ML Taxonomy



ChatGPT

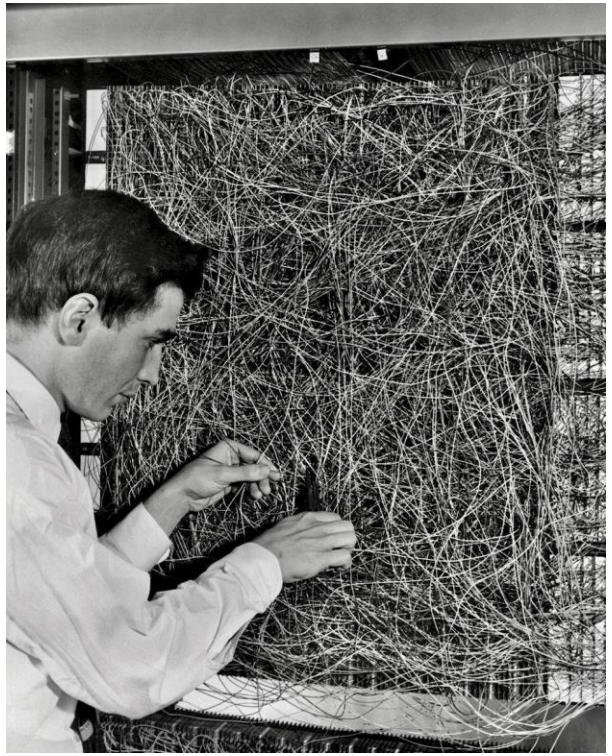
ML Taxonomy


Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
DIMENSIONALLY REDUCTION
Image Classification
Customer Retention
Fraud Detection
CLASSIFICATION
Diagnostics
Recommended Systems
Targetted Marketing
Customer Segmentation
CLUSTERING
UNSUPERVISED LEARNING
SUPERVISED LEARNING
MACHINE LEARNING
Forecasting
Predictions
Process Optimization
New Insights
REGRESSION
REINFORCEMNET LEARNING
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

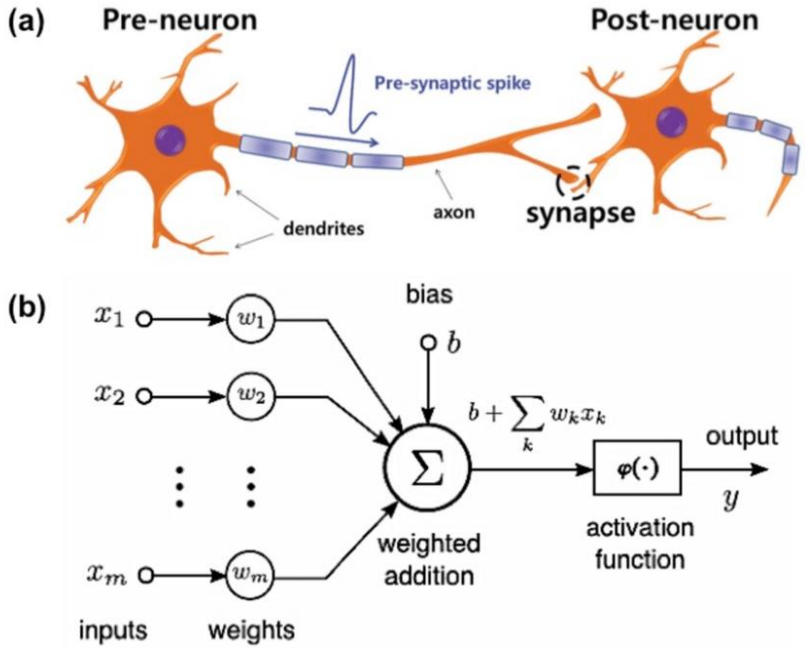


Artificial Neural Networks

Frank Rosenblatt: The Perceptron (1957)



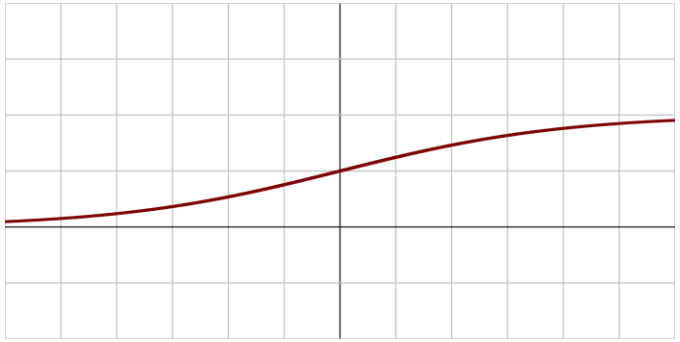
From biological to artificial neurons



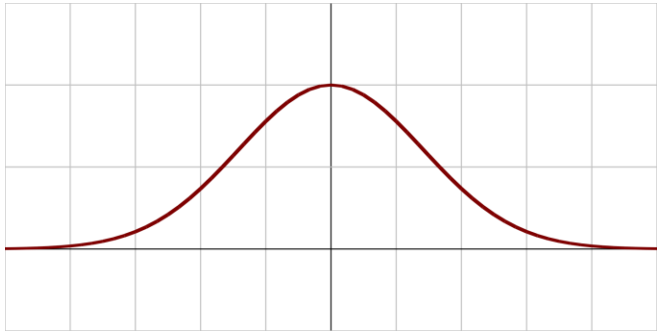
Artificial Neural Networks

How to handle non-linear problems?

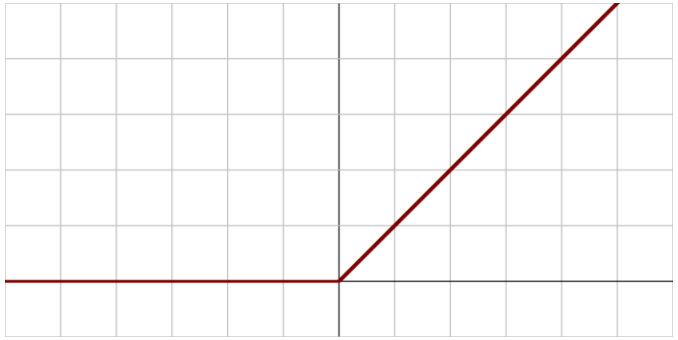
1/ Adding non-linear activation functions →Artificial Neural Networks



Sigmoid



Gaussian

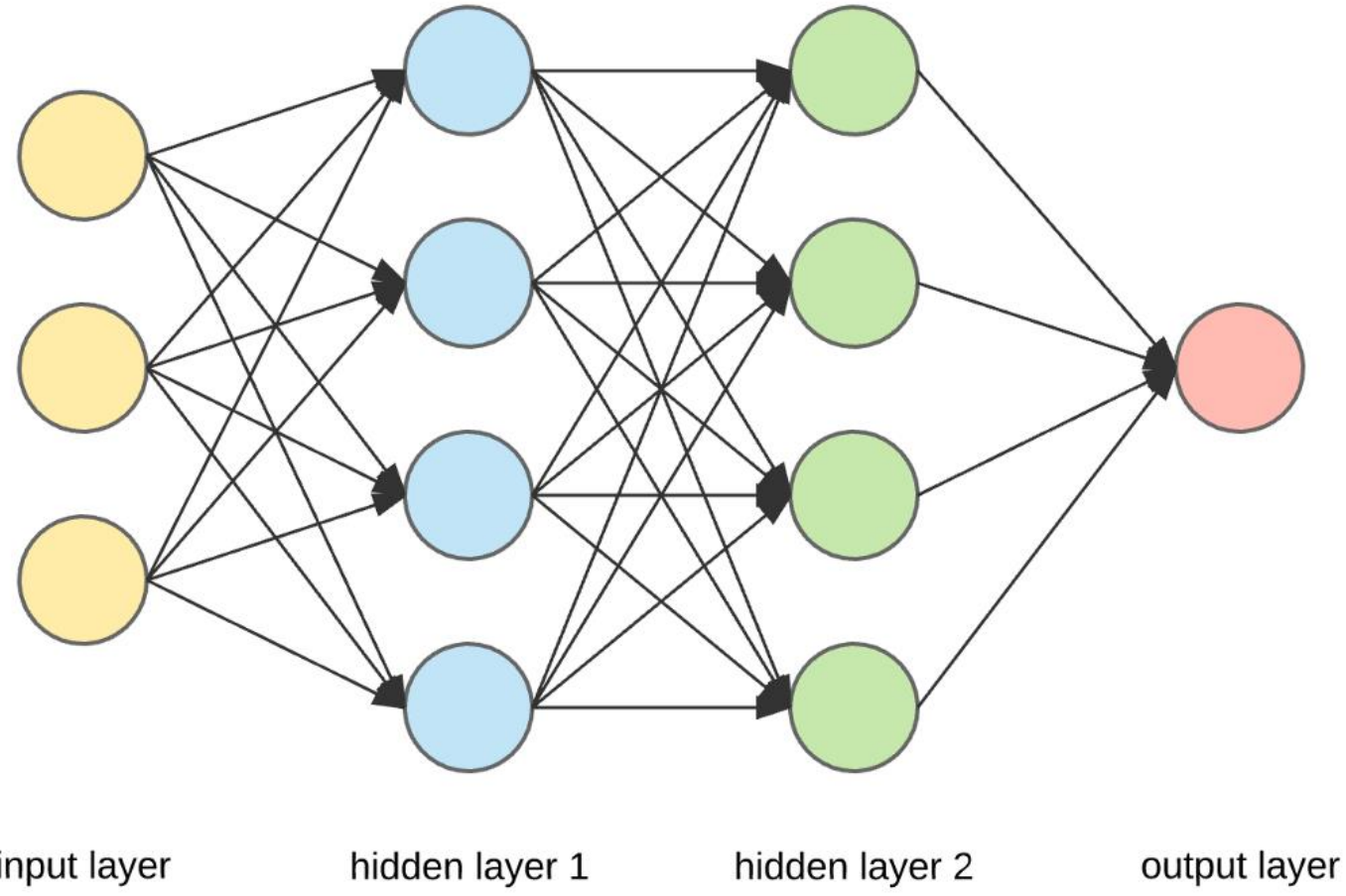


RELU

Artificial Neural Networks

How to handle non-linear problems?

2/ Adding more layers: Fully connected Deep Neural Networks

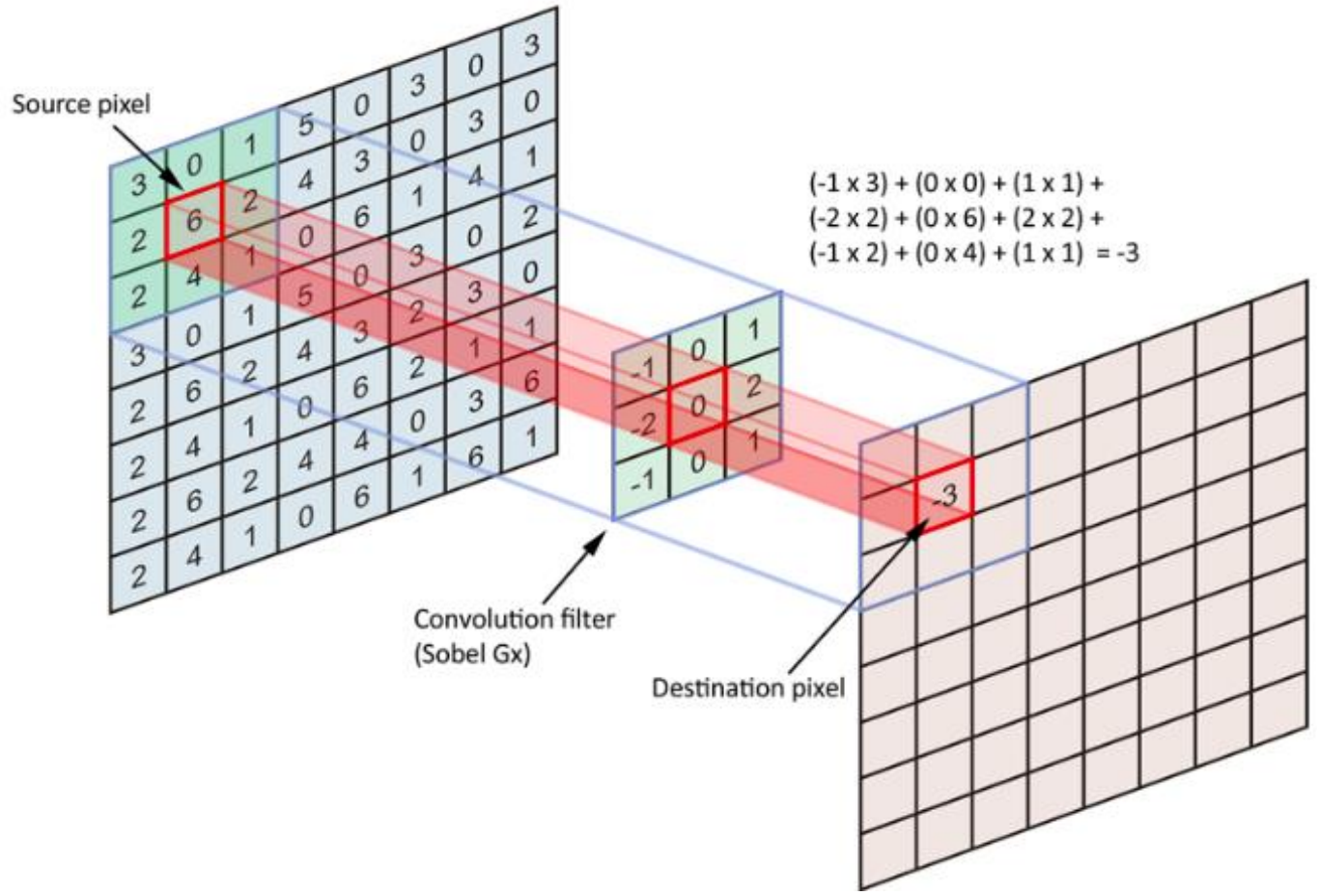


Artificial Neural Networks

How to handle non-linear problems?

3/ Adding convolution layers: Convolutional Neural Networks (CNNs)

Idea:
Passes a filter with a fixed window size and weights to an input image

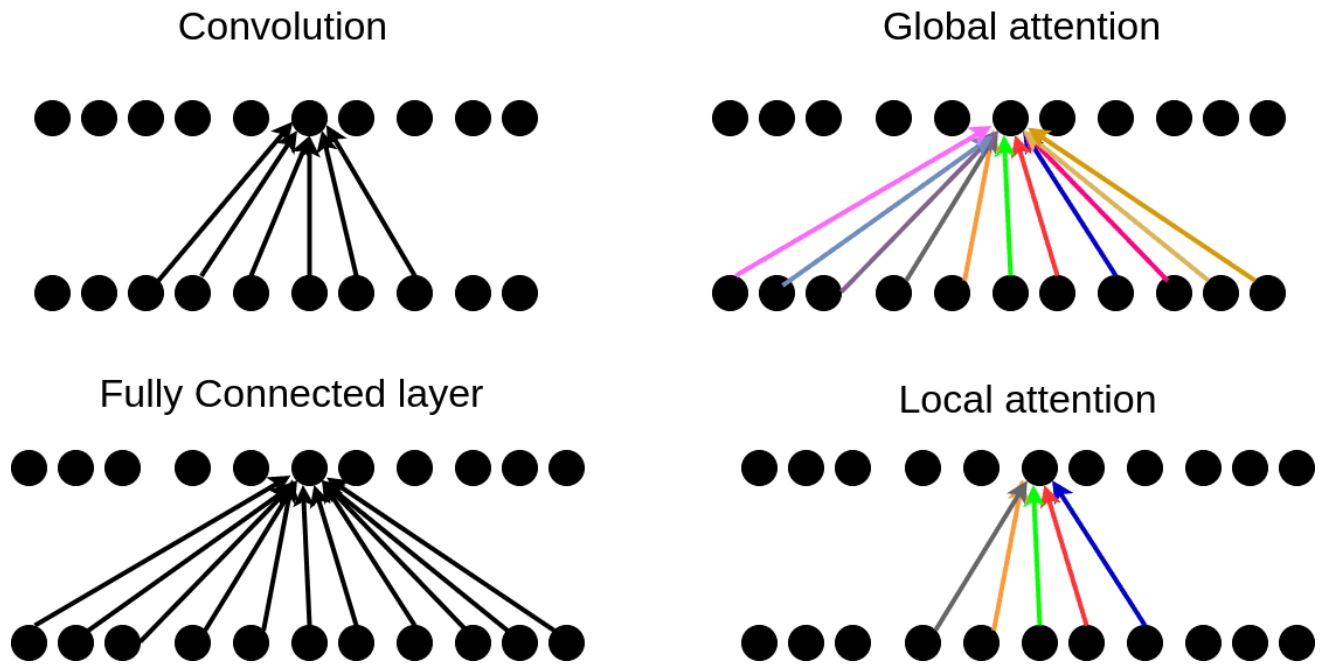


Artificial Neural Networks

How to handle non-linear problems?

4/ Adding attention layers : Transformers

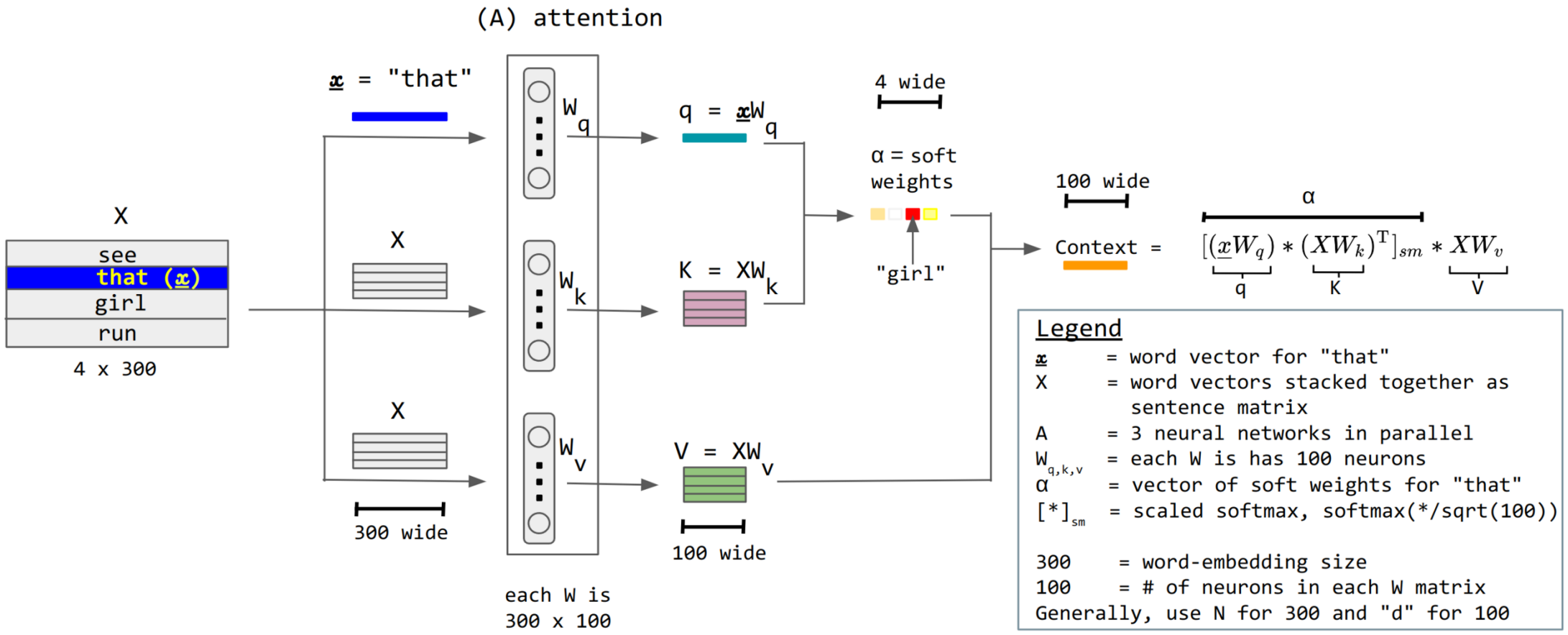
Idea:
Learn 'soft' **data-dependent** connections (attention) between the neurons/features at the same time as learning the 'hard' **usual** connections



Artificial Neural Networks

How to handle non-linear problems?

4/ Adding attention layers : Transformers



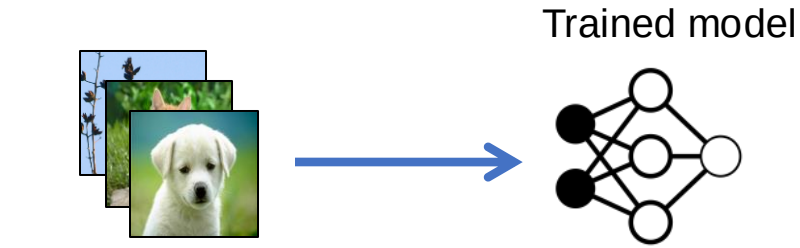
Practical: Part 1

Link to notebook:
https://shorturl.at/JWNu7

Machine Learning Threats & Attacks

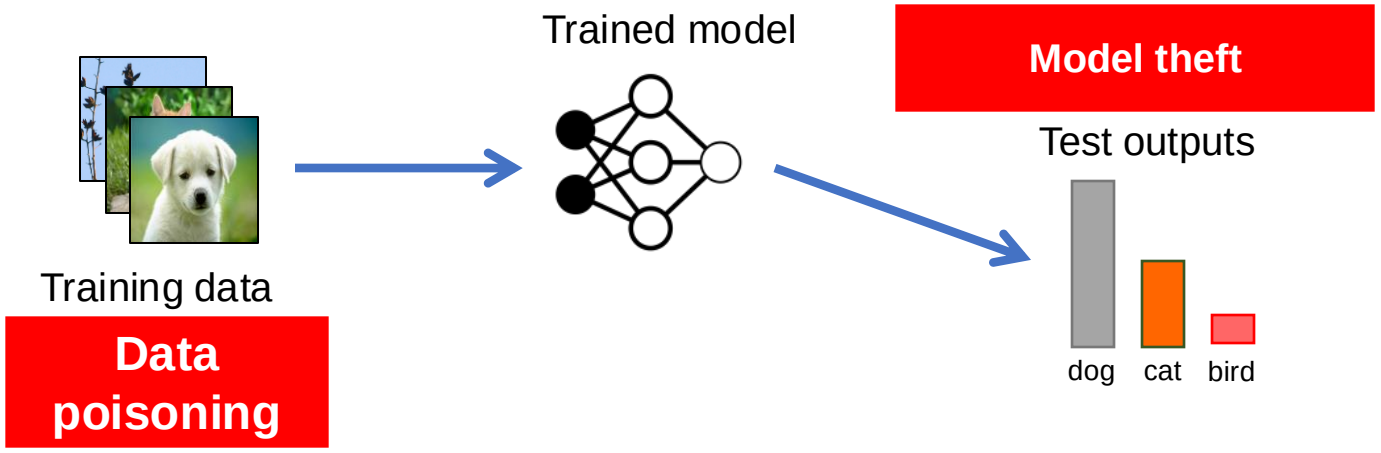
Part I

Main threats of ML components

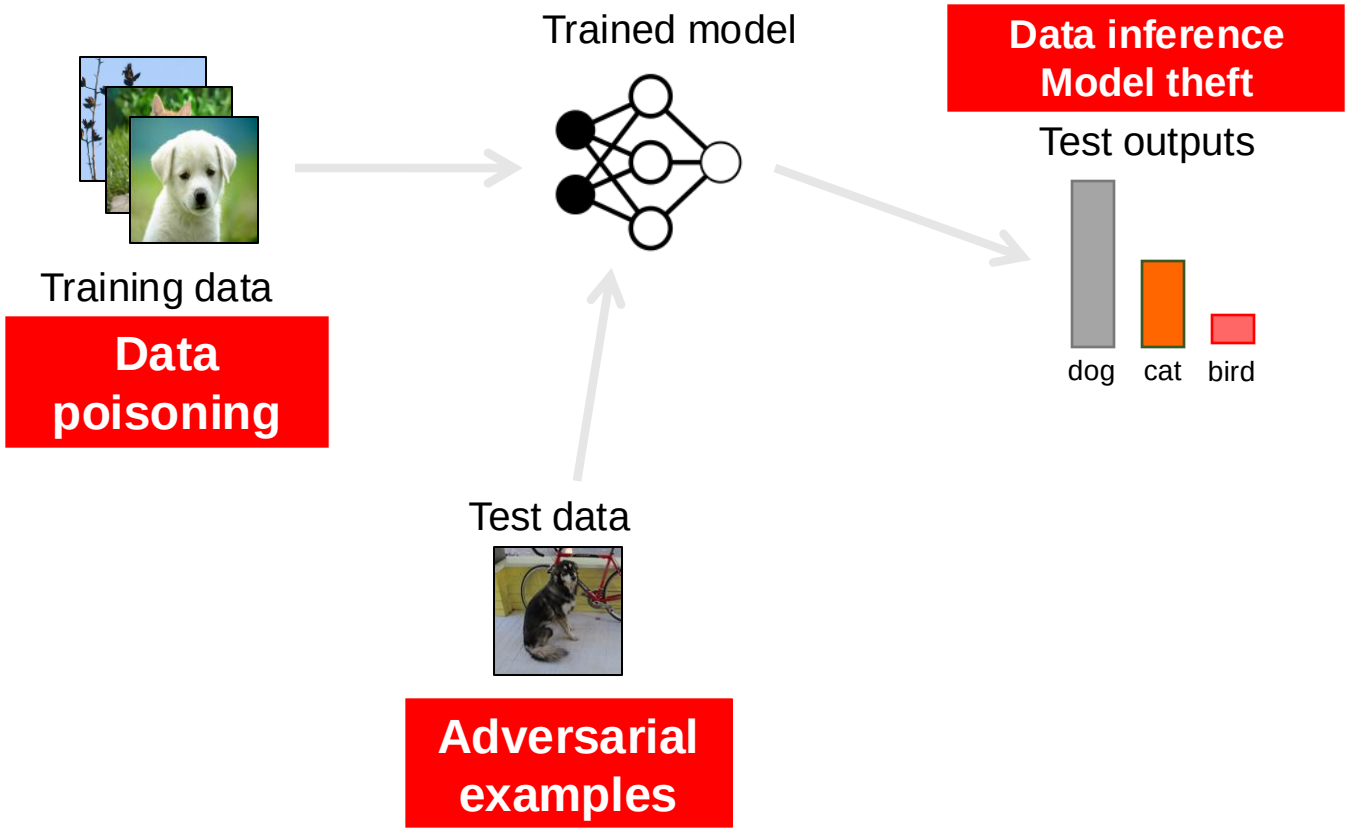


Data poisoning

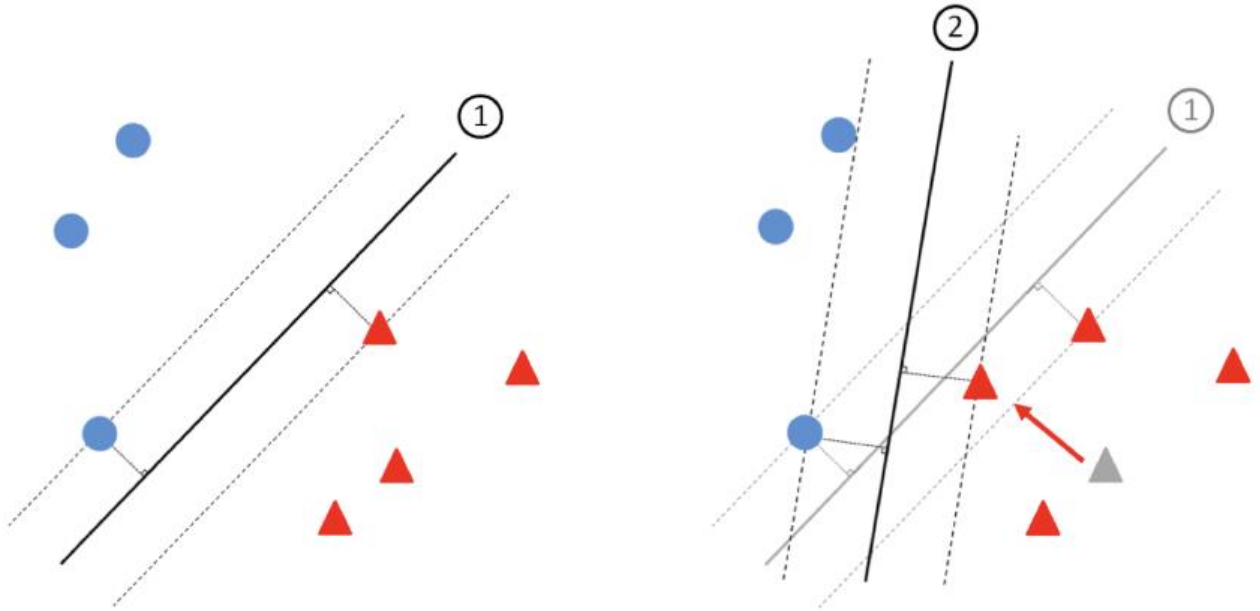
Main threats of ML components



Main threats of ML components



Poisoning attack



Poisoning attack

Availability attack

Poisoning attack

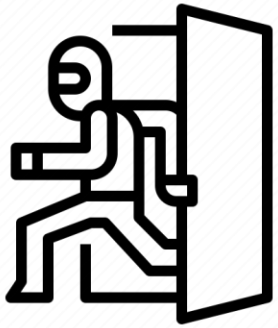
Availability attack

3% poisoning ➔ 11% drop of performance

Poisoning attack

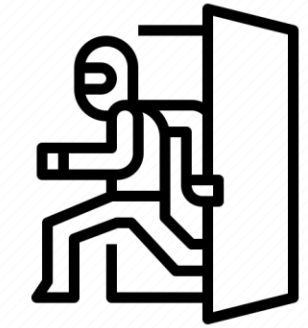
Availability attack

Integrity attack



Poisoning attack

Integrity attack
(Back doors)



Poisoning attack

Poisoning attack

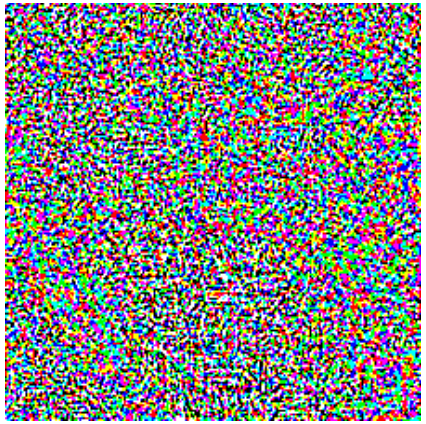
- Tay bot used the interactions with its Twitter users as training data
- By repeatedly interacting with Tay using racist and offensive language, they were able to bias Tay's dataset towards that language as well
- Within 24 hours of its deployment, Tay had to be decommissioned

Evasion attack and adversarial examples

Original example

ML predicts:
“Panda”
(80% confidence)

Small adversarial noise



Adversarial example

What humans still see

Gibbon

What ML predicts: “Gibbon”
(99% confidence)

"Explaining and Harnessing Adversarial Examples", Goodfelow et al., ICLR 2015.

Adversarial examples beyond pixels

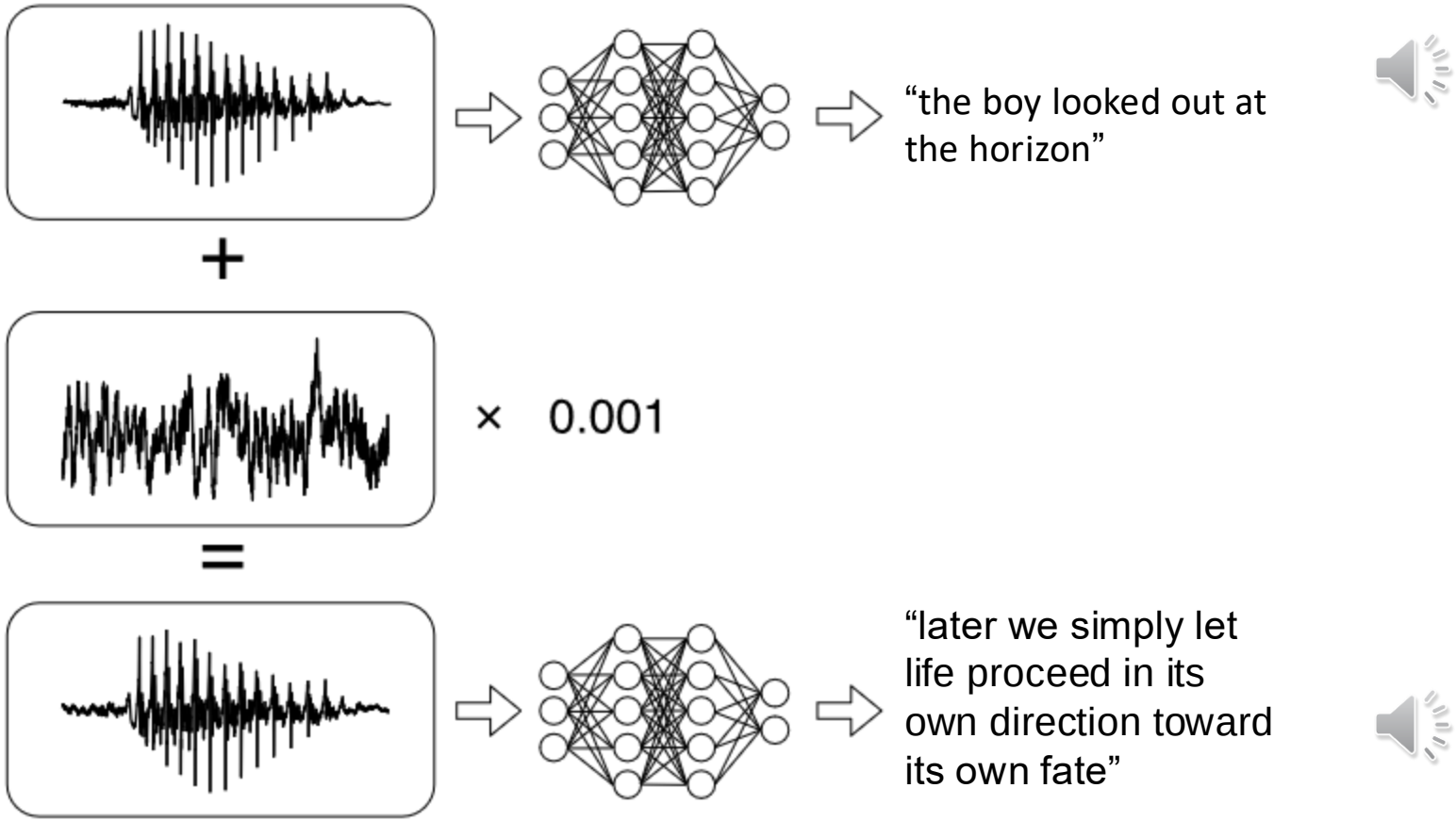


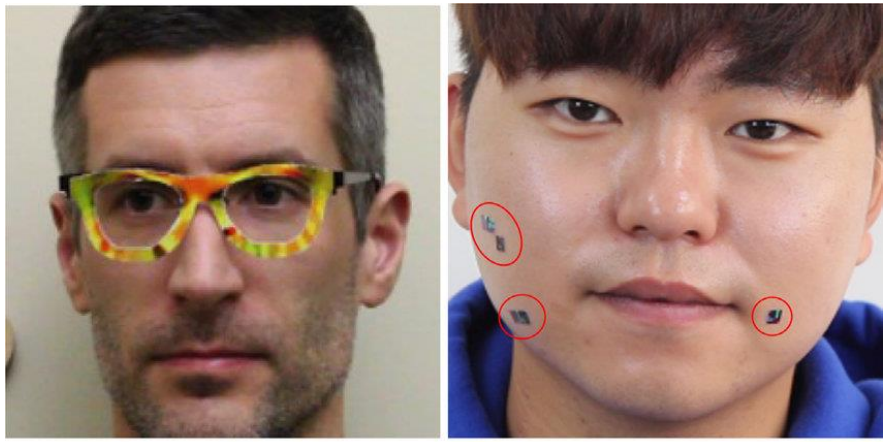
Figure 1. Illustration of our attack: given any waveform, adding a small perturbation makes the result transcribe as any desired target phrase.

Carlini, Nicholas, and David Wagner. "Audio adversarial examples: Targeted attacks on speech-to-text." *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018.

Adversarial examples in the physical world

Adversarial examples in the physical world

Crowd face recognition system



(a) (b)

Journal of Information Security and Applications 60 (2021) 102874

Contents lists available at ScienceDirect

Journal of Information Security and Applications

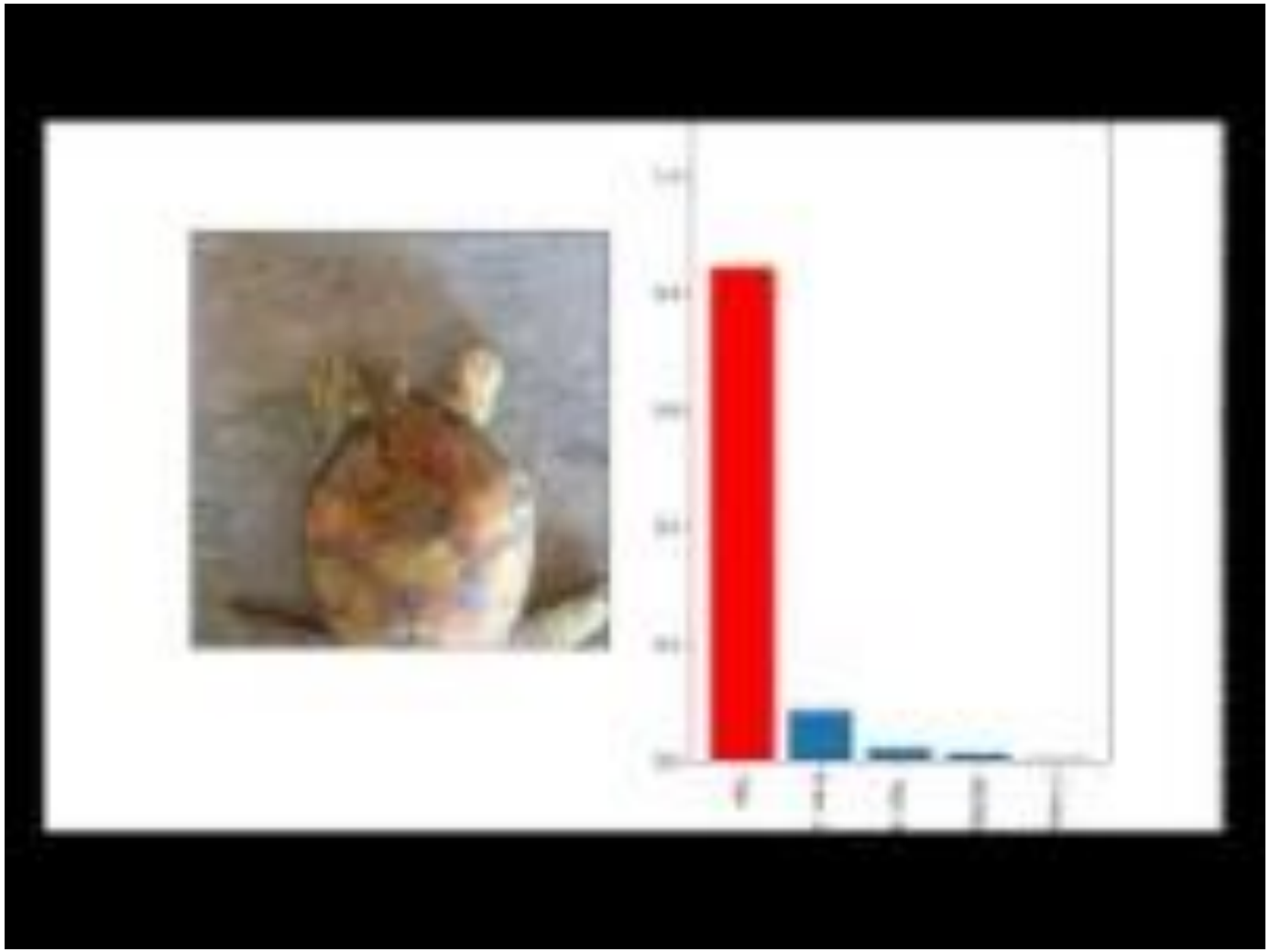
journal homepage: www.elsevier.com/locate/jisa

Adversarial attacks by attaching noise markers on the face against deep face recognition

Gwonsang Ryu [a], Hosung Park [b], Daeseon Choi [c,*]

[a] Department of Software Convergence, Graduate School of Soongsil University, Seoul, 07027, South Korea
[b] Department of Cyber Security and Police, Busan University of Foreign Studies, Busan, 46234, South Korea
[c] Department of Software, Soongsil University, Seoul, 07027, South Korea

Adversarial examples in the physical world



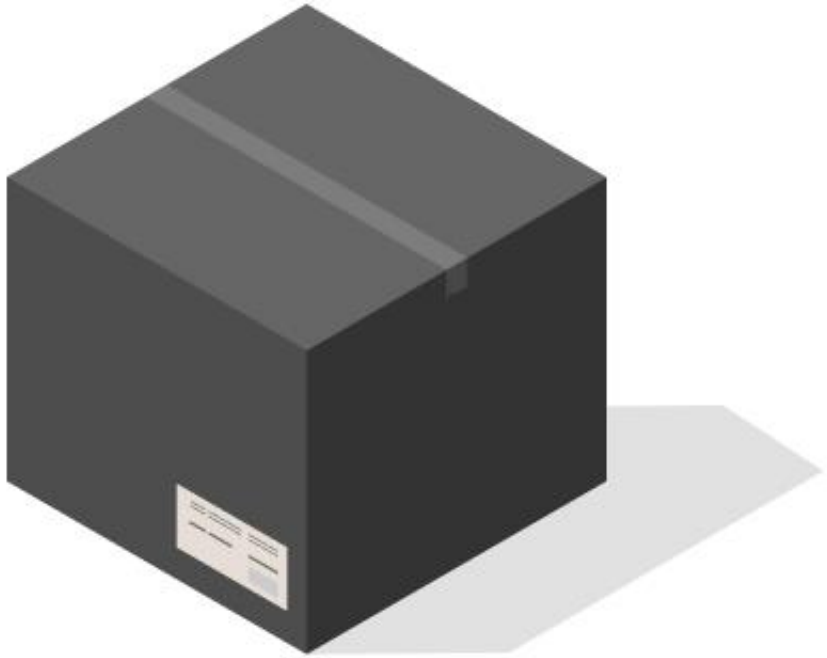
Taxonomy > Threat model

Access to:

- Parameters
- Architecture
- Hyper-parameters
- Gradients
- Training set

White box - we know everything

Taxonomy > Threat model



Black box - we do not know anything

No direct access to anything

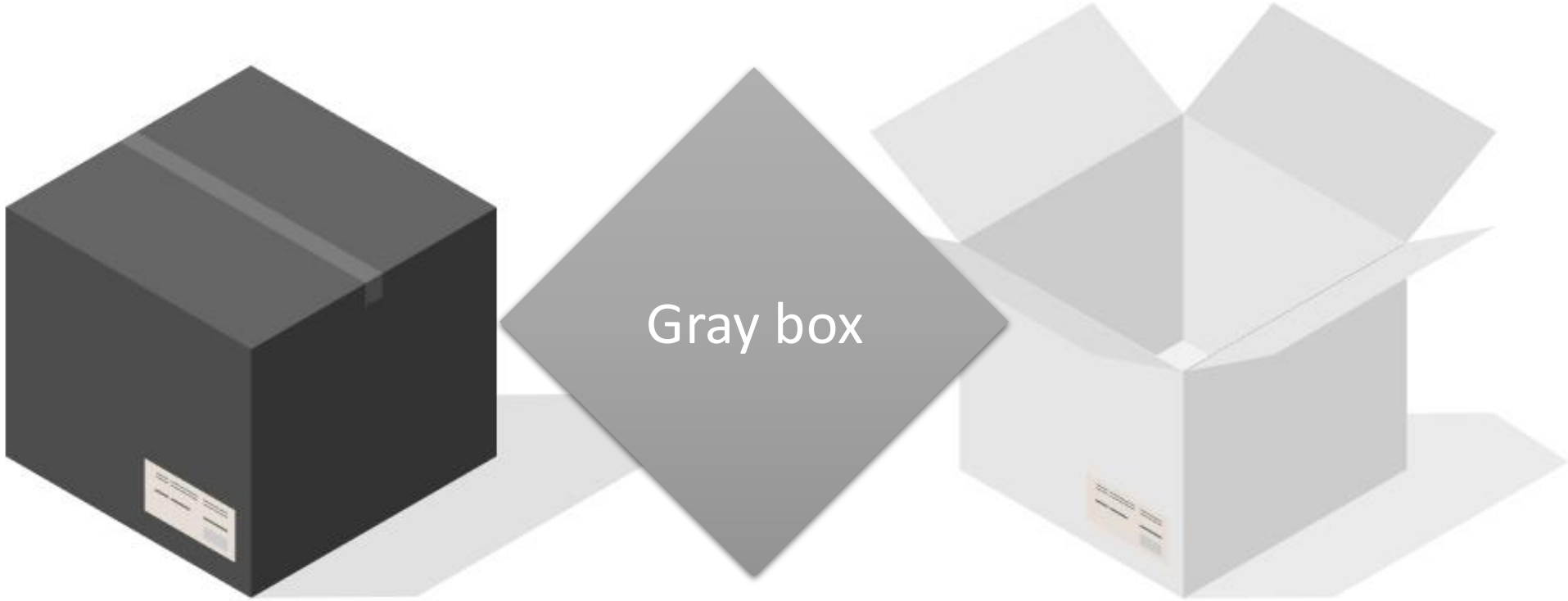
Intuitions about:

- Architecture
- Training distributions

Access through **API queries**

- Prediction logits
- OR **just** predictions

Taxonomy > Threat model



Black box - we do not know anything

White box - we know everything

Taxonomy > Threat model

Non-targeted

Any wrong result of the ML model is satisfactory

Targeted

We expect a specific wrong result of the ML model

Harder

Whitebox Evasion attacks
In computer vision

Part II

Formal definition

- Given an image x, which is labeled by the classifier as class q, i.e., $C(x)=q$
- Create an adversarial image x_{adv} by adding small perturbations δ to the original image, i.e., $x_{adv} = x+\delta$,

  such that the distance $D(x, x_{adv})= D(x,x+\delta)$ is minimal

  and that the classifier assigns a label to the adversarial image that is different than q, i.e., $C(x_{adv})=C(x+\delta)=t\neq q$

Formal definition > Minimization problem

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x+\delta \in [0,1]^n \end{aligned}$$

distance between x and $x+\delta$

$x+\delta$ is classified as target class t

each element of $x+\delta$ is in [0,1] (to be a valid image)

Formal definition > Distance metrics

Distance metrics between x and x_{adv}: $D(x, x_{adv})$

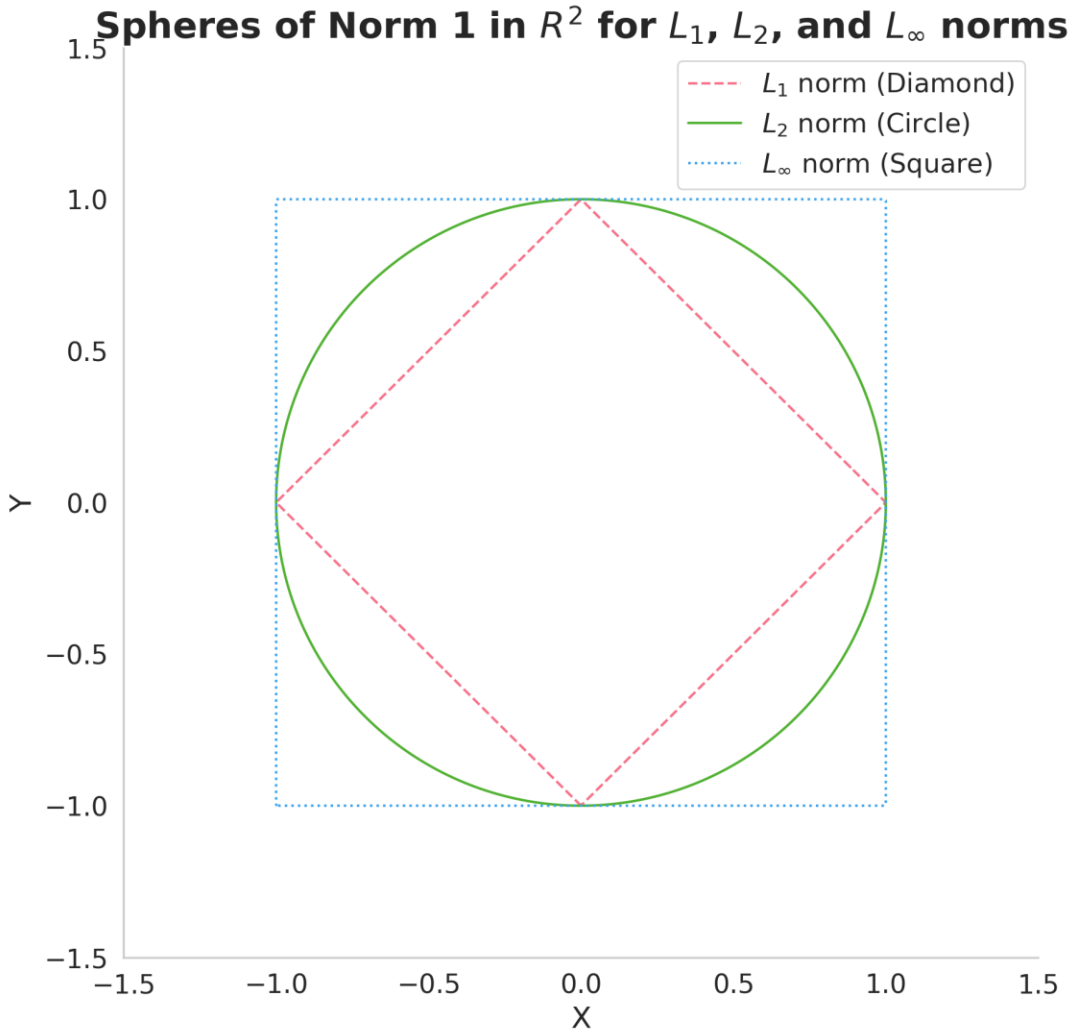
- ℓ_0 norm: the number of elements in x_{adv} such that $x^i \neq x^i_{adv}$
  Corresponds to the number of pixels that have been changed in the image x_{adv}

- ℓ_1 norm: city-block distance, or Manhattan distance
  $$\ell_1 = \left|x^1 - x^1_{adv}\right| + \left|x^2 - x^2_{adv}\right| + \cdots + |x^n - x^n_{adv}|$$

- ℓ_2 norm: Euclidean distance, or mean-squared error
  $$\ell_2 = \sqrt{(x^1 - x^1_{adv})^2 + (x^2 - x^2_{adv})^2 + \cdots + (x^n - x^n_{adv})^2}$$

- ℓ_∞ norm: measures the maximum change to any of the pixels in the x_{adv} image
  $$\ell_\infty = max\left(\left|x^1 - x^1_{adv}\right|, \left|x^2 - x^2_{adv}\right|, \ldots, \left|x^n - x^n_{adv}\right|\right)$$

Illustration > Distance metrics



Fast Gradient Sign Method (FGSM)

An adversarial image x_{adv} is created by adding perturbation noise to an image x

$$x_{adv} = x + \epsilon \cdot \text{sign}\left(\nabla_x \mathcal{L}(h(x, w), y)\right)$$

- Notation: input image x, cost function $\mathcal{L}$, NN model h, NN weights (parameters) w, gradient ∇, noise magnitude ϵ
- Perturbation noise is calculated as the gradient of the loss function $\mathcal{L}$ with respect to the input image x for the true class label y
- This increases the loss for the true class $y \rightarrow$ the model misclassifies the image x_{adv}

$$\text{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

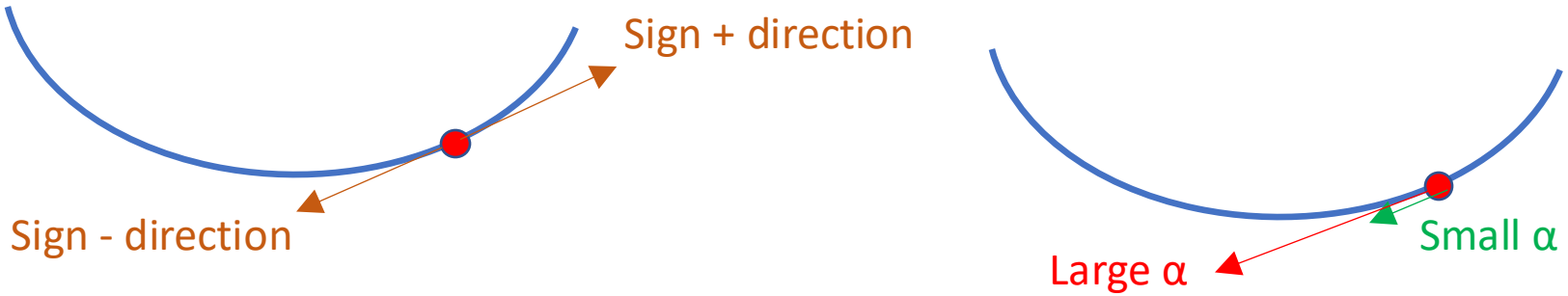
"Explaining and Harnessing Adversarial Examples", Goodfelow et al., ICLR 2015.

FGSM > Why does it work?

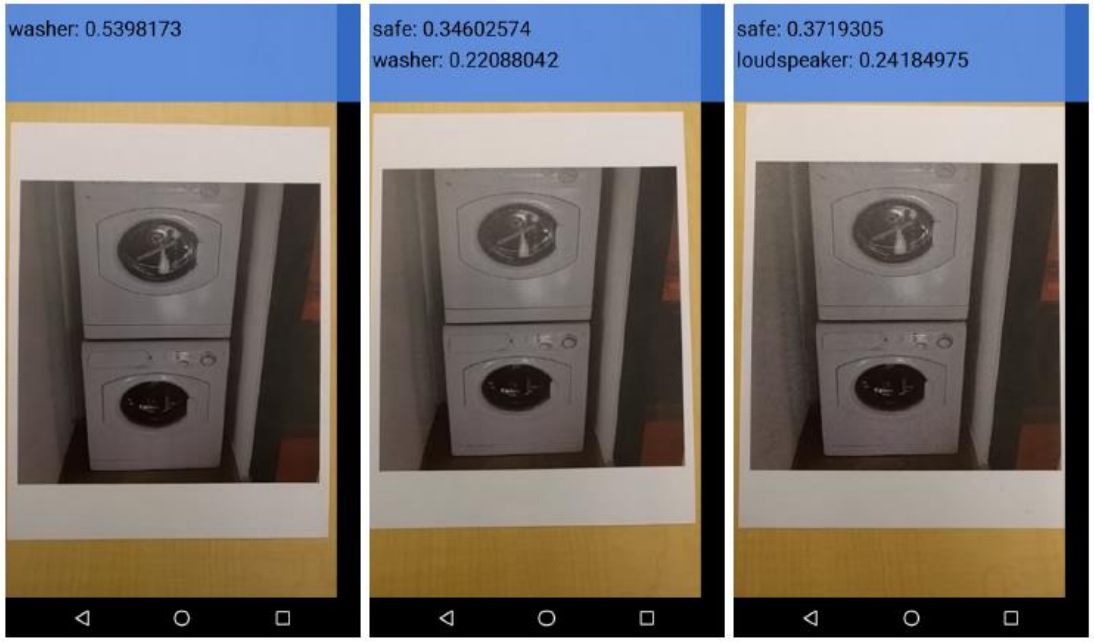
The sign and magnitude of the gradient give the direction and the slope of the steepest descent

➔ To minimize the loss function, the weights w are changed in the opposite direction of the gradient, i.e., $w = w - \alpha \frac{\partial loss}{\partial w}$

➔ FGSM operates on the inputs, they are changed in the direction of the gradient: $x = x + \alpha \frac{\partial loss}{\partial w}$



Basic iterative method (BIM; Kurakin, 2017)



(b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

What if we compute FGSM **iteratively** with a small step?

$$x_{t+1} = x_t + \alpha \ \text{sign}\left(\nabla_x \mathcal{L}(h(x, w), y)\right)$$

Projected Gradient Descent (PGD, Madry 2017)

Extension of BIM with projection and random starts

PGD is regarded as the strongest first-order attack

➔ *First-order attack means that the adversary uses only the gradients of the loss function with respect to the input*

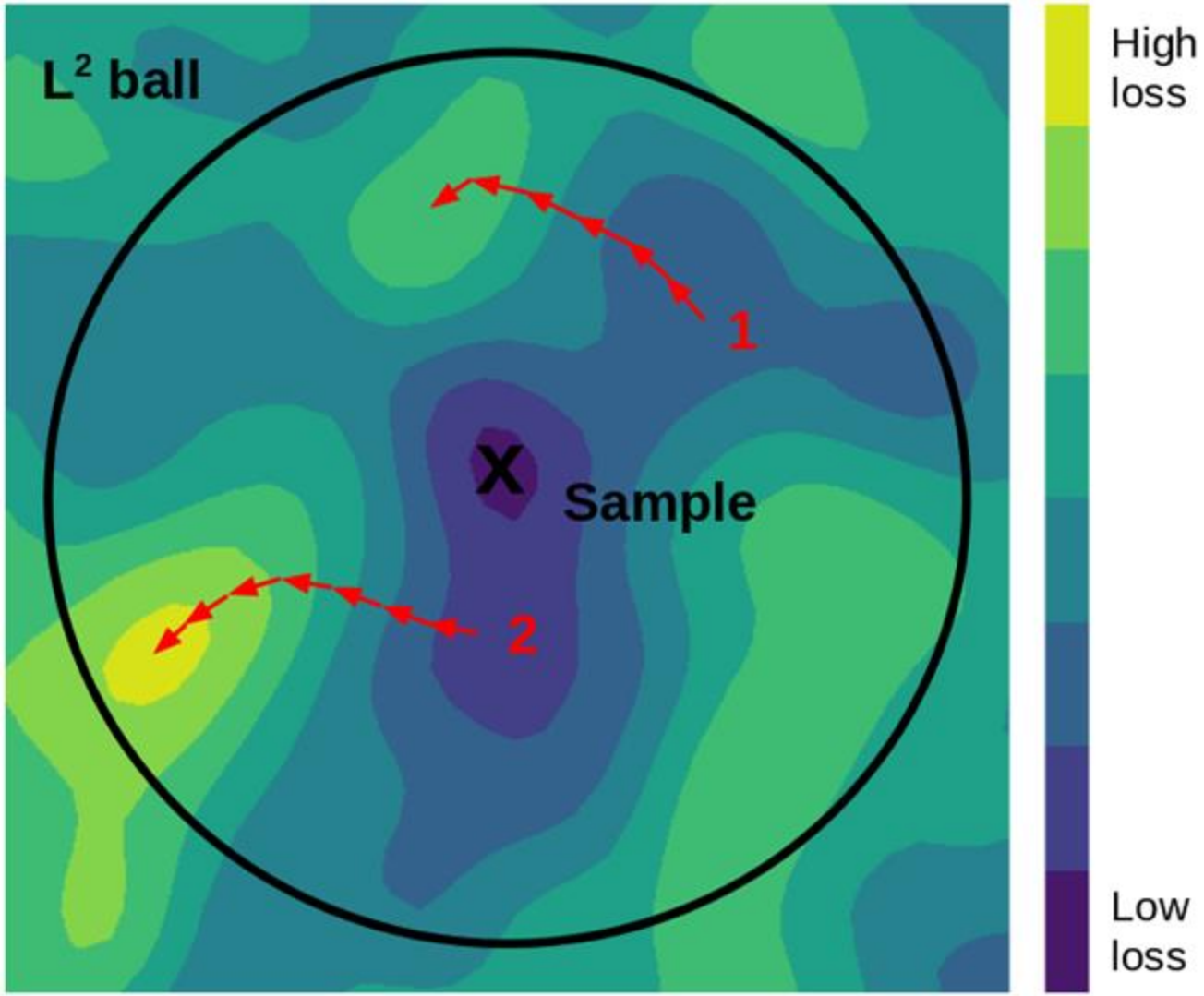


Image modified from https://towardsdatascience.com/know-your-enemy-7f7c5038bdf3

Projected Gradient Descent (PGD)

Iteratively compute:

$$x^{t+1} = \text{Clip}_{x,\epsilon}(x^t + \alpha \,.\, sign(\nabla_{x^t}(L(\theta_f, x^t, y))))$$

model parameters

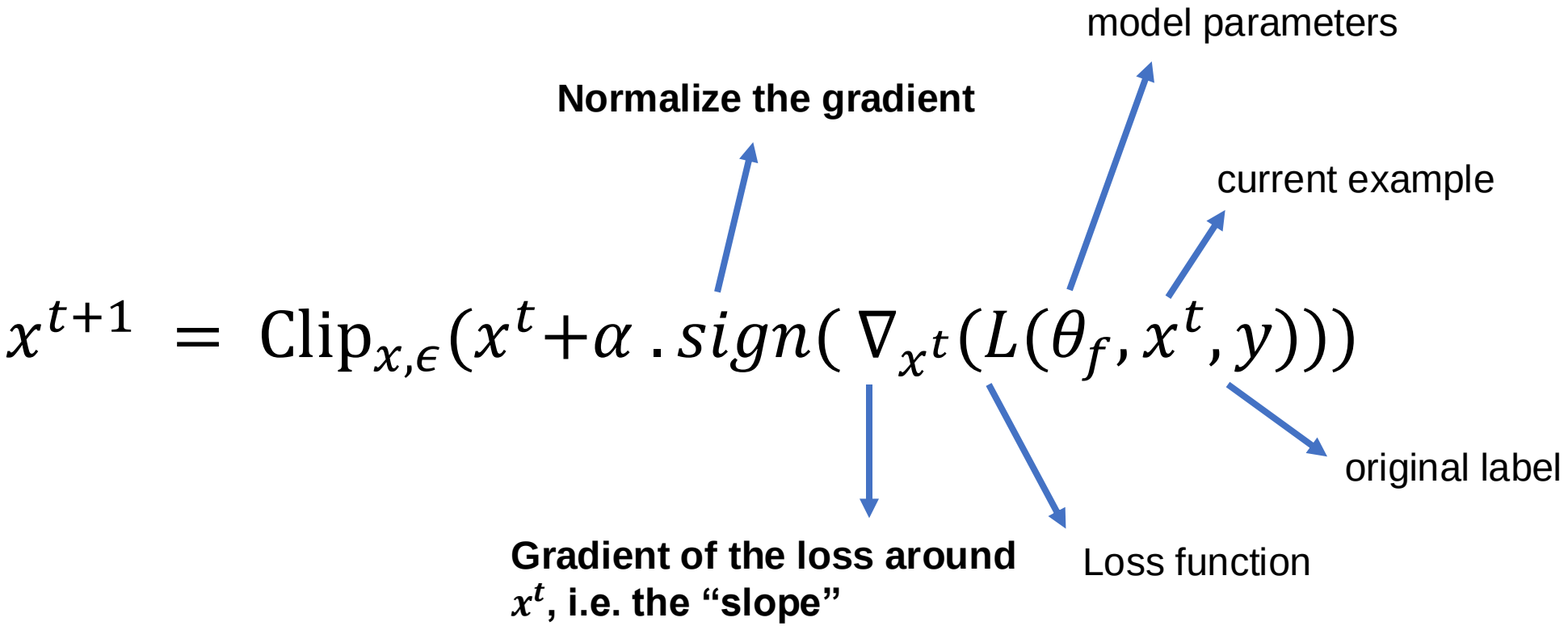
current example

original label

Loss function

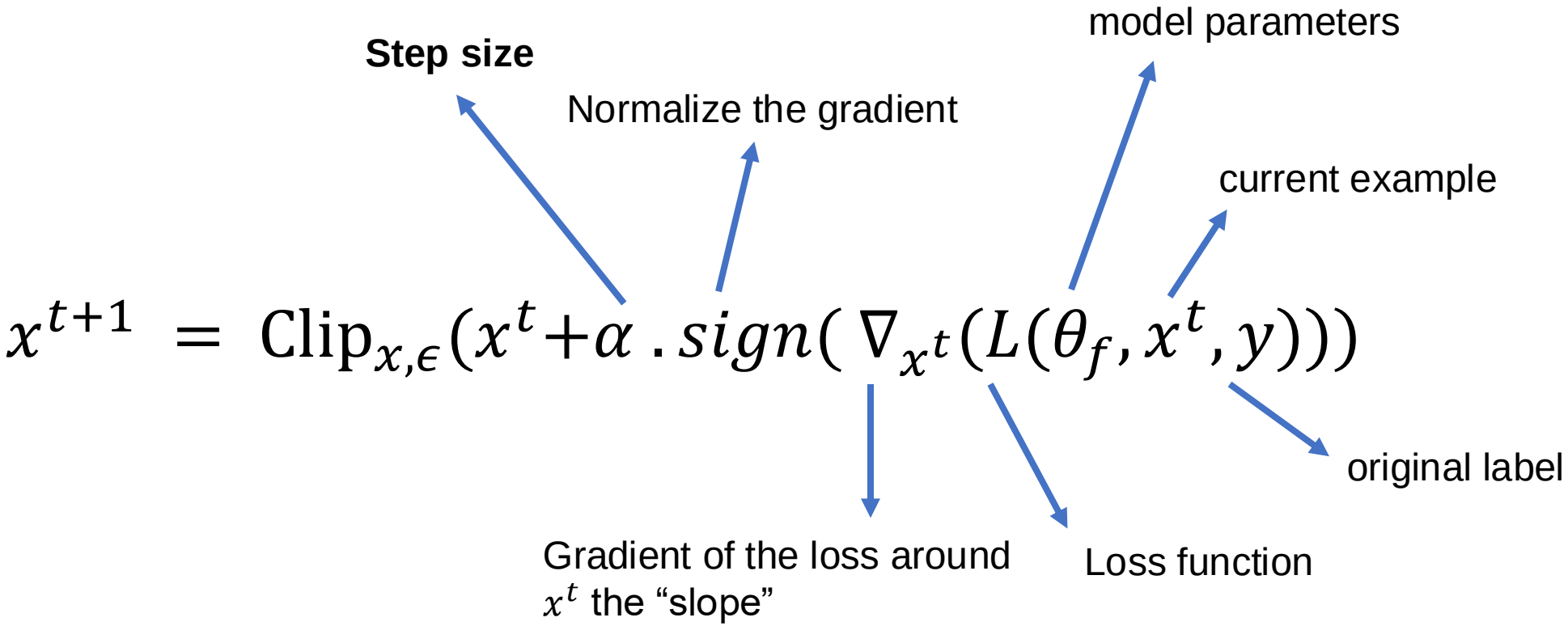
Projected Gradient Descent (PGD)

Iteratively compute:



Projected Gradient Descent (PGD)

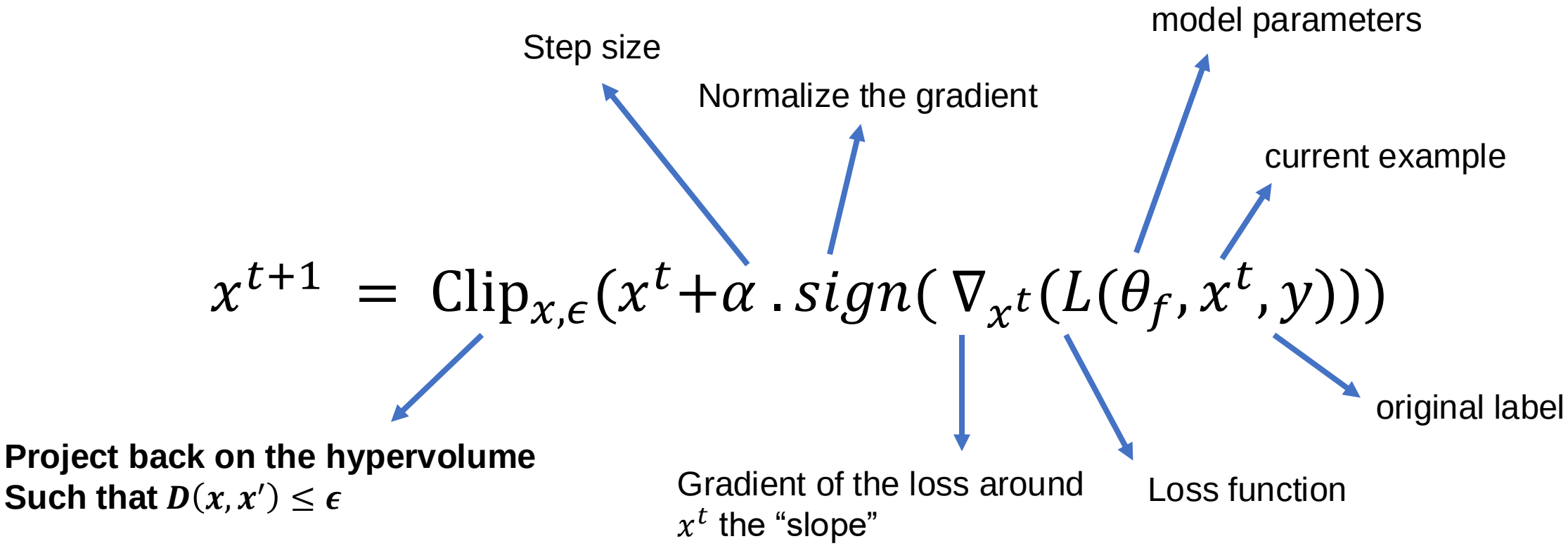
Iteratively compute:



$$x^{t+1} = \text{Clip}_{x,\epsilon}(x^t + \alpha \,.\, sign(\nabla_{x^t}(L(\theta_f, x^t, y))))$$

Projected Gradient Descent (PGD)

Iteratively compute:



Targeted PGD

For a targeted attack, if the target class label is denoted t, adversarial examples are created by using

$$x_{adv}^{t} = \Pi_{\epsilon}\left(x^{t-1} - \alpha \cdot \mathrm{sign}\left(\nabla_x \mathcal{L}(h(x^{t-1}), t)\right)\right)$$

- I.e., it is based on minimizing the loss function with respect to the target class t
- This is opposite to non-targeted attacks, which maximize the loss function with respect to the true class label

Practical: Part 2

Other gradient attacks

MI-FGSM (Dong, 2018)

Algorithm 1 MI-FGSM

Input: A classifier f with loss function J; a real example $\boldsymbol{x}$ and ground-truth label y;

Input: The size of perturbation ϵ; iterations T and decay factor μ.

Output: An adversarial example $\boldsymbol{x}^*$ with $\|\boldsymbol{x}^* - \boldsymbol{x}\|_\infty \leq \epsilon$.

$\alpha = \epsilon / T$;

$\boldsymbol{g}_0 = 0$; $\boldsymbol{x}_0^* = \boldsymbol{x}$;

for $t = 0$ to $T - 1$ **do**

  Input $\boldsymbol{x}_t^*$ to f and obtain the gradient $\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t^*, y)$;

  Update $\boldsymbol{g}_{t+1}$ by accumulating the velocity vector in the gradient direction as

$$\boldsymbol{g}_{t+1} = \mu \cdot \boldsymbol{g}_t + \frac{\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t^*, y)}{\|\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t^*, y)\|_1}; \tag{6}$$

  Update $\boldsymbol{x}_{t+1}^*$ by applying the sign gradient as

$$\boldsymbol{x}_{t+1}^* = \boldsymbol{x}_t^* + \alpha \cdot \text{sign}(\boldsymbol{g}_{t+1}); \tag{7}$$

end for

return $\boldsymbol{x}^* = \boldsymbol{x}_T^*$.

Other gradient attacks

Carlini-Wagner (CW; Carlini, 2017)

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & f(x+\delta) \leq 0 \\ & x+\delta \in [0,1]^n \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) + c \cdot f(x+\delta) \\ \text{such that} \quad & x+\delta \in [0,1]^n \end{aligned}$$

$$f_2(x') = (\max_{i \neq t}(F(x')_i) - F(x')_t)^+$$

Stronger attacks

PGD has limitations!

1) Fixed step size

Unaware of iteration budget
Agnostic of previous iterations

Stronger attacks

PGD has limitations!

1) Fixed step size

Unaware of iteration budget
Agnostic of previous iterations

APGD adds a momentum

Algorithm 1 APGD

Input: f, S, $x^{(0)}$, η, N_{iter}, $W = \{w_0, \ldots, w_n\}$
Output: $x_{\max}$, $f_{\max}$
$x^{(1)} \leftarrow P_S\left(x^{(0)} + \eta \nabla f(x^{(0)})\right)$
$f_{\max} \leftarrow \max\{f(x^{(0)}), f(x^{(1)})\}$
$x_{\max} \leftarrow x^{(0)}$ **if** $f_{\max} \equiv f(x^{(0)})$ **else** $x_{\max} \leftarrow x^{(1)}$
for $k = 1$ **to** $N_{\text{iter}} - 1$ **do**
  $z^{(k+1)} \leftarrow P_S\left(x^{(k)} + \eta \nabla f(x^{(k)})\right)$
  $x^{(k+1)} \leftarrow P_S\Big(x^{(k)} + \alpha(z^{(k+1)} - x^{(k)}) + (1-\alpha)(x^{(k)} - x^{(k-1)})\Big)$
  if $f(x^{(k+1)}) > f_{\max}$ **then**
    $x_{\max} \leftarrow x^{(k+1)}$ and $f_{\max} \leftarrow f(x^{(k+1)})$
  end if
  if $k \in W$ **then**
    if Condition 1 **or** Condition 2 **then**
      $\eta \leftarrow \eta/2$ and $x^{(k+1)} \leftarrow x_{\max}$
    end if
  end if
end for

Stronger attacks

PGD has limitations!

1) Fixed step size

Unaware of iteration budget
Agnostic of previous iterations

APGD adds a momentum

APGD proposes adaptive steps in two phases:
a/ Large step η for exploration
b/ Small step η for exploitation

Algorithm 1 APGD

Input: f, S, $x^{(0)}$, η, N_{iter}, $W = \{w_0, \ldots, w_n\}$
Output: $x_{\max}$, $f_{\max}$
$x^{(1)} \leftarrow P_S\left(x^{(0)} + \eta \nabla f(x^{(0)})\right)$
$f_{\max} \leftarrow \max\{f(x^{(0)}), f(x^{(1)})\}$
$x_{\max} \leftarrow x^{(0)}$ **if** $f_{\max} \equiv f(x^{(0)})$ **else** $x_{\max} \leftarrow x^{(1)}$
for $k = 1$ **to** $N_{\text{iter}} - 1$ **do**
  $z^{(k+1)} \leftarrow P_S\left(x^{(k)} + \eta \nabla f(x^{(k)})\right)$
  $x^{(k+1)} \leftarrow P_S\Big(x^{(k)} + \alpha(z^{(k+1)} - x^{(k)}) + (1-\alpha)(x^{(k)} - x^{(k-1)})\Big)$
  if $f(x^{(k+1)}) > f_{\max}$ **then**
    $x_{\max} \leftarrow x^{(k+1)}$ and $f_{\max} \leftarrow f(x^{(k+1)})$
  end if
  if $k \in W$ **then**
    if Condition 1 **or** Condition 2 **then**
      $\eta \leftarrow \eta/2$ and $x^{(k+1)} \leftarrow x_{\max}$
    end if
  end if
end for

Stronger attacks

PGD has limitations!

2) For converging models, CE leads to vanishing gradients

Unaware of iteration budget
Agnostic of previous iterations

APGD adds a momentum

APGD proposes adaptive steps in two phases:
a/ Large step η for exploration
b/ Small step η for exploitation

APGD replaces its loss function

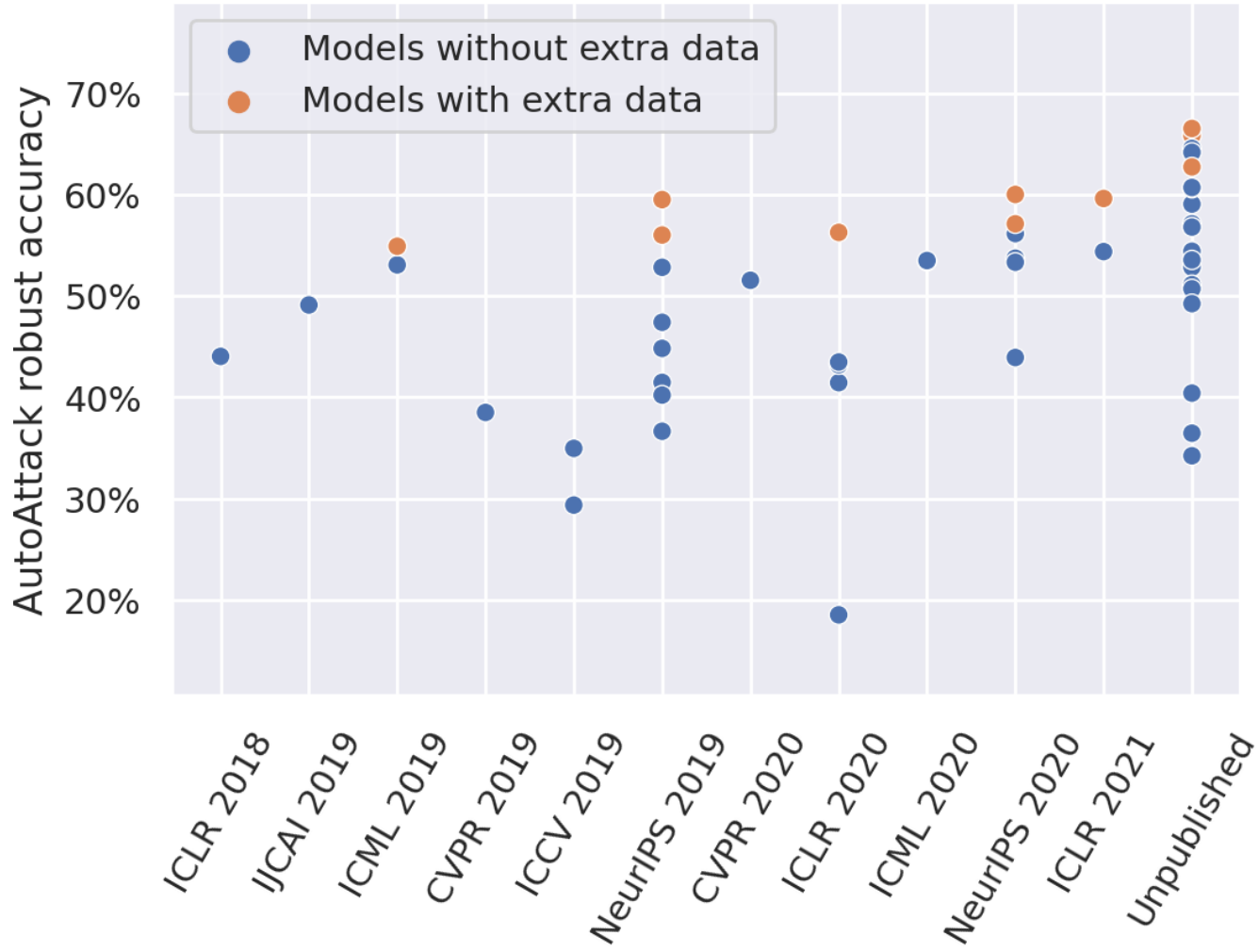
Algorithm 1 APGD

Input: f, S, $x^{(0)}$, η, N_{iter}, $W = \{w_0, \ldots, w_n\}$
Output: $x_{\max}$, $f_{\max}$
$x^{(1)} \leftarrow P_{\mathcal{S}}\left(x^{(0)} + \eta \nabla f(x^{(0)})\right)$
$f_{\max} \leftarrow \max\{f(x^{(0)}), f(x^{(1)})\}$
$x_{\max} \leftarrow x^{(0)}$ **if** $f_{\max} \equiv f(x^{(0)})$ **else** $x_{\max} \leftarrow x^{(1)}$
for $k = 1$ **to** $N_{\text{iter}} - 1$ **do**
  $z^{(k+1)} \leftarrow P_{\mathcal{S}}\left(x^{(k)} + \eta \nabla f(x^{(k)})\right)$
  $x^{(k+1)} \leftarrow P_{\mathcal{S}}\Big(x^{(k)} + \alpha(z^{(k+1)} - x^{(k)}) + (1-\alpha)(x^{(k)} - x^{(k-1)})\Big)$
  if $f(x^{(k+1)}) > f_{\max}$ **then**
    $x_{\max} \leftarrow x^{(k+1)}$ and $f_{\max} \leftarrow f(x^{(k+1)})$
  end if
  end if
end for

$$DLR(x, y) = -\frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}}$$, where π is decreasing ordering of z

Stronger attacks > Whitebox Benchmarks

Robustbench



Practical Part 3

Blackbox Evasion attacks

In computer vision

Part III

Blackbox evasion attacks

Transfer attacks

Craft whitebox adversarial examples on a surrogate model transferable to target

Query attacks

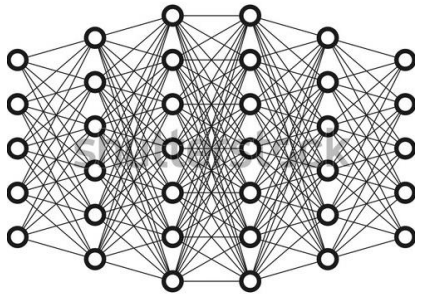
Craft adversarial examples by accessing the output of target model with queries

Transfer attacks

Vanilla transfer attacks

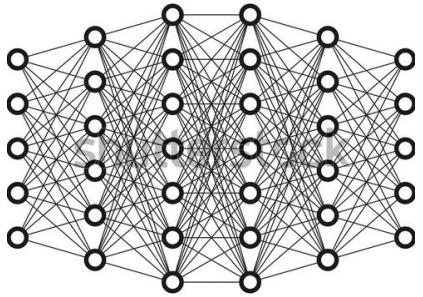
Hope that the surrogate and target have similar decision boundaries

Adversarial image



Surrogate model

Dog!



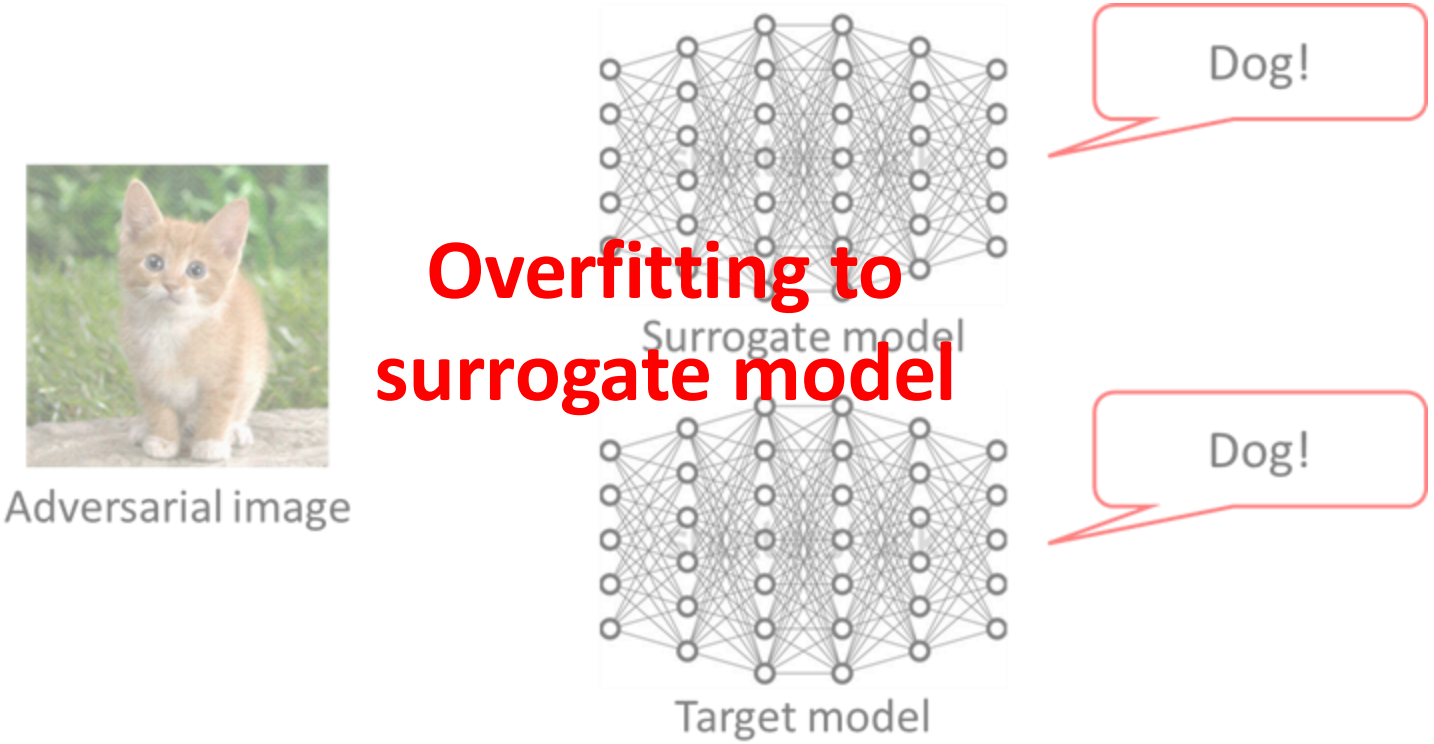
Target model

Dog!

Transfer attacks

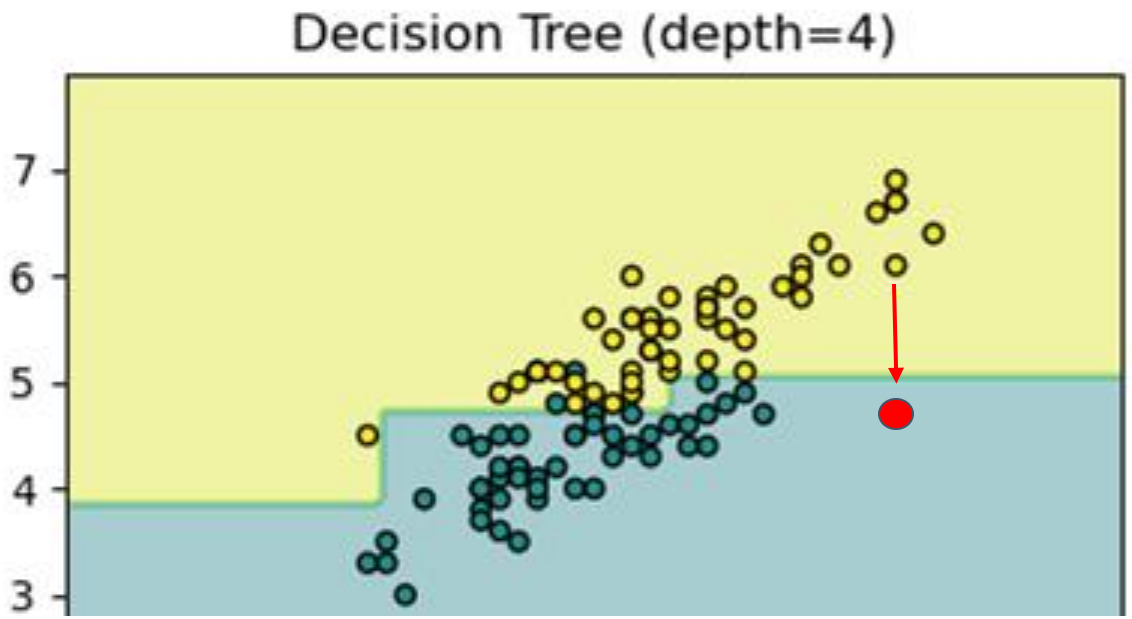
Vanilla transfer attacks

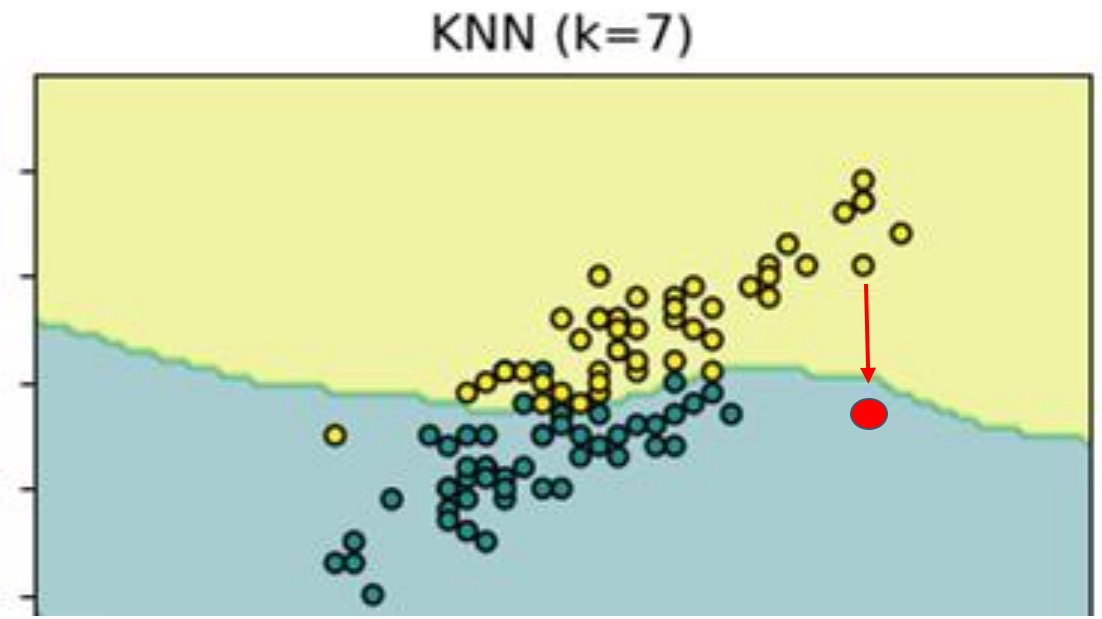
Hope that the surrogate and target have similar decision boundaries



Transfer attacks

Generally succeed…

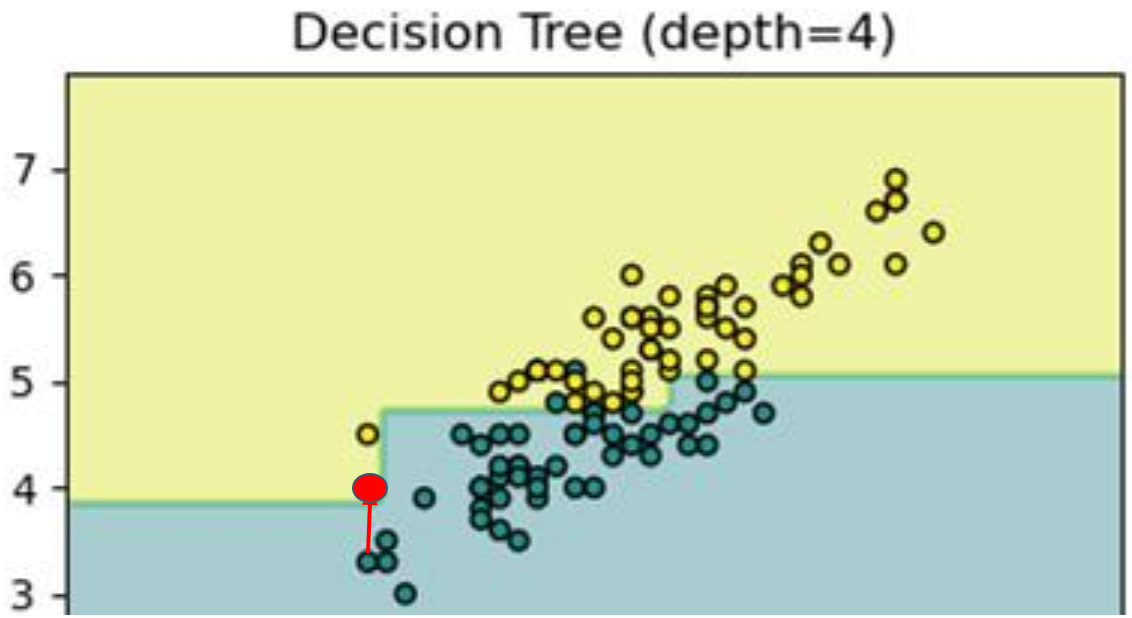


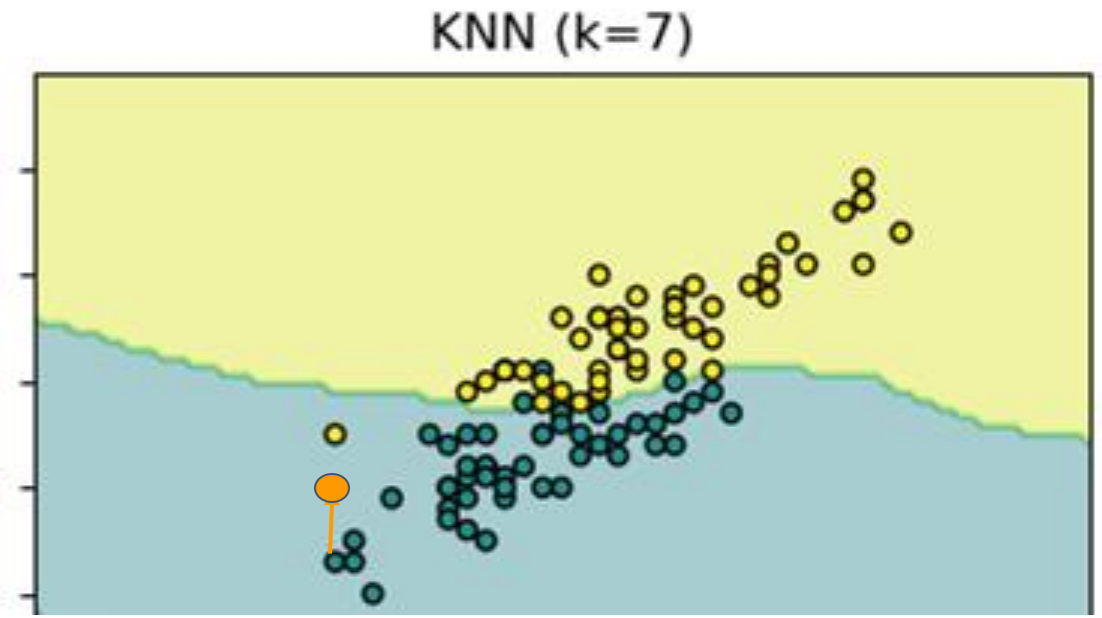


Example from https://scikit-learn.org/stable/auto_examples/ensemble/plot_voting_decision_regions.html

Transfer attacks

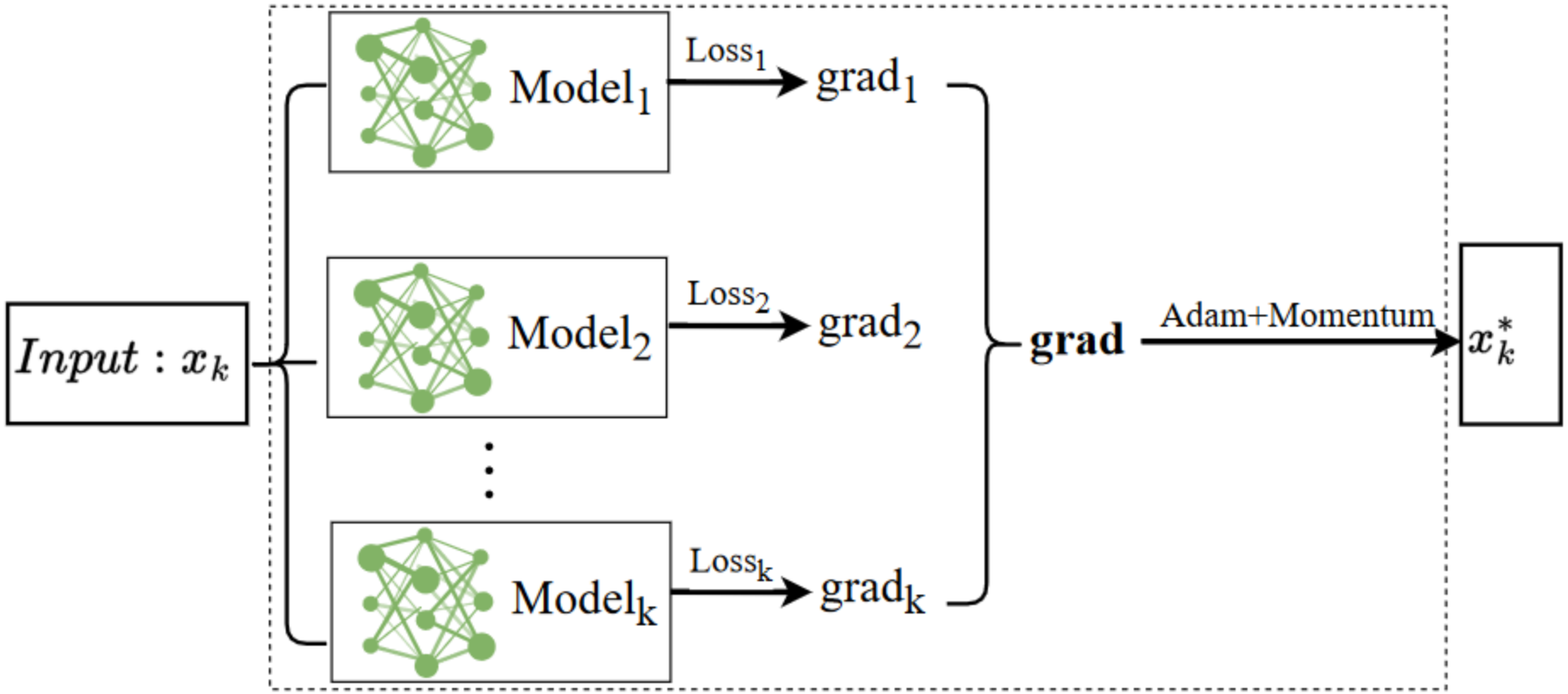
but sometime fail!



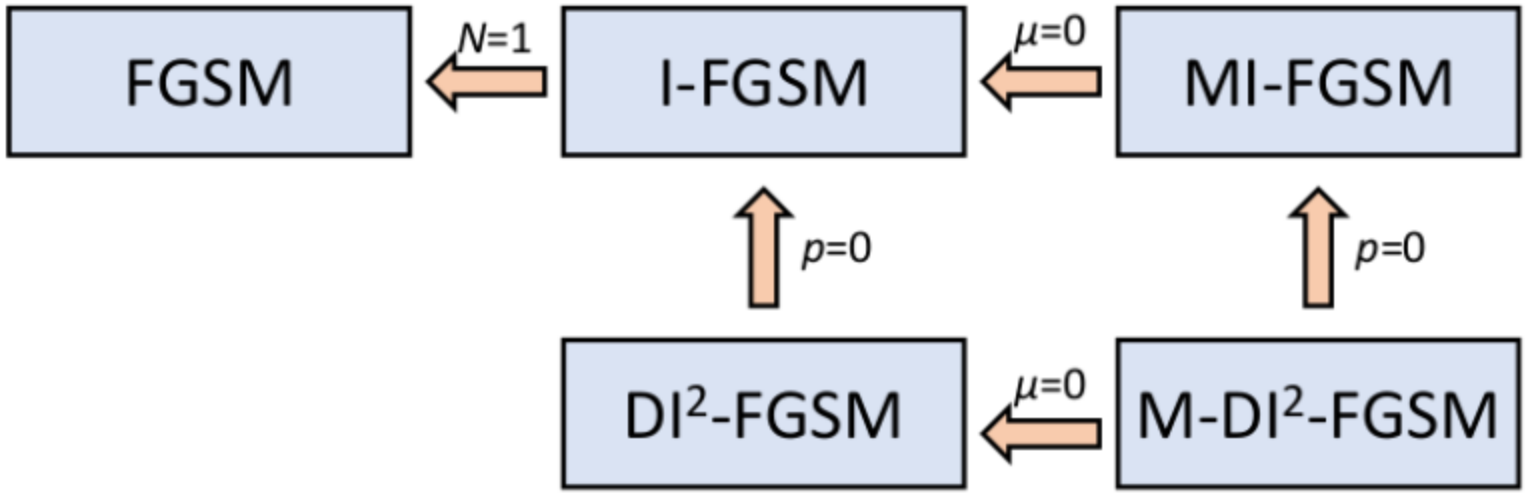


Example from https://scikit-learn.org/stable/auto_examples/ensemble/plot_voting_decision_regions.html

Transfer attacks > Ensemble



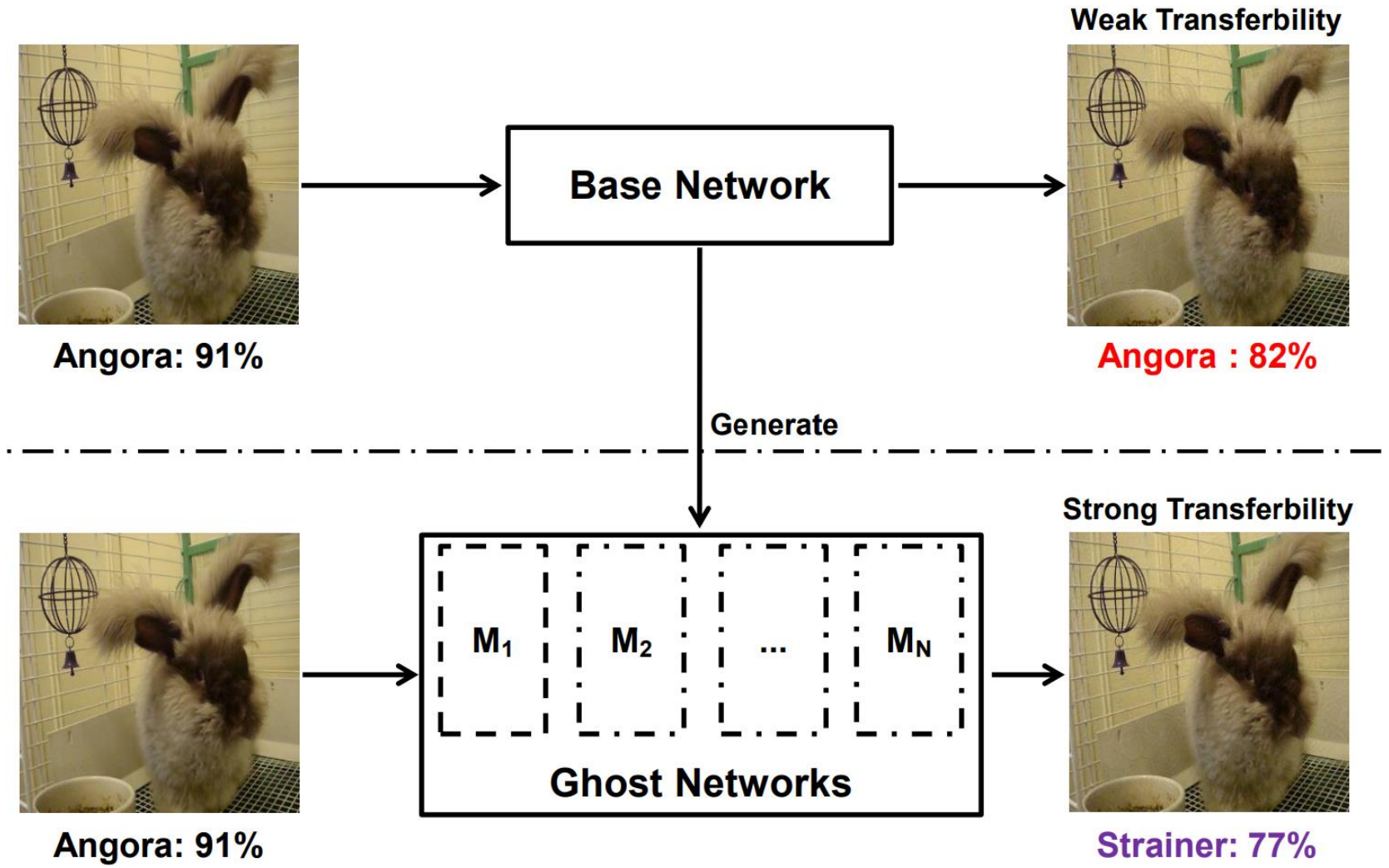
Transfer attacks > MI/DI FGSM



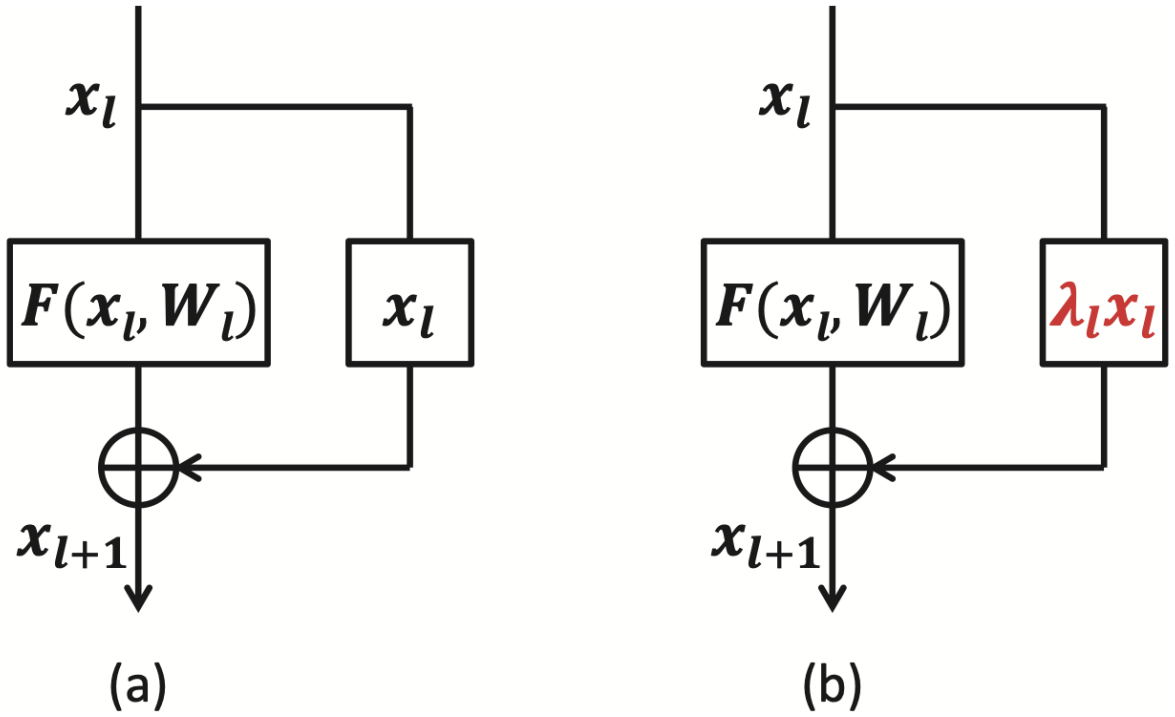
DI: Input Diversity

MI: Iterative Momentum

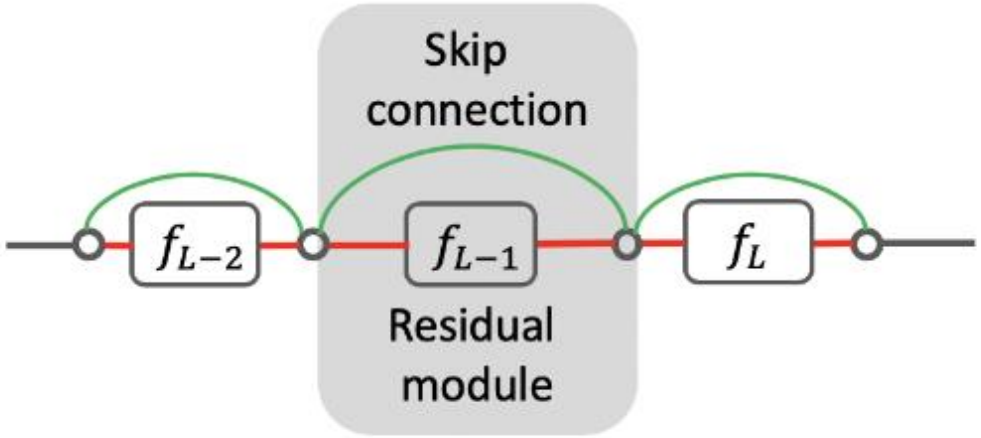
Transfer attacks > Ghost Networks



Transfer attacks > Ghost Networks



Transfer attacks > SGM

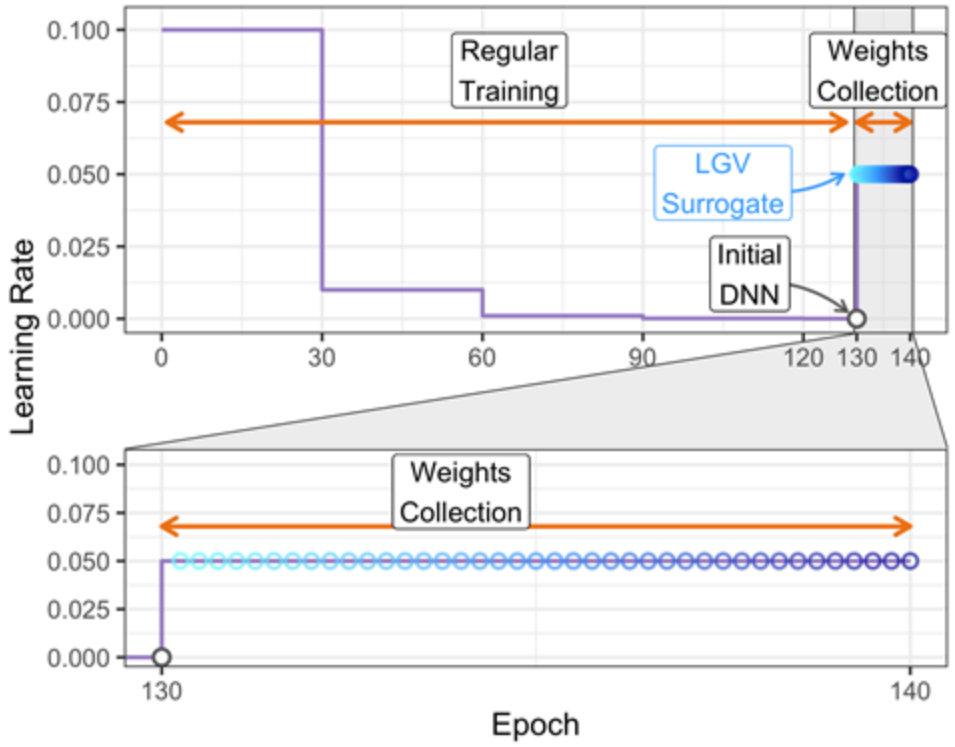


"Using more gradients from the skip connections rather than the residual modules, allows one to craft adversarial examples with high transferability"

	RN18	RN34	RN50	RN101	RN152	DN121	DN169	DN201
PGD	23.23±0.69	24.38±0.41	22.80±0.55	22.98±0.83	26.56±0.75	30.71±0.60	30.90±0.31	36.01±0.59
SGM	**28.92±0.45**	**43.43±0.32**	**36.71±0.55**	**38.38±0.53**	**44.84±0.14**	**57.38±0.14**	**60.45±0.42**	**65.48±0.23**

Transfer attacks > LGV

Phase 1
Collect models during a few additional epochs with a **high learning rate**



Algorithm 1 LGV Weights Collection

Input: n_{epochs} number of epochs, K number of weights, η learning rate, γ momentum, w_0 pretrained weights, $\mathcal{D}$ training dataset

Output: $(w_1, \ldots, w_K)$ LGV weights

1: $w \leftarrow w_0$ ▷ Start from a regularly trained DNN
2: **for** $i \leftarrow 1$ **to** K **do**
3: $w \leftarrow \text{SGD}(w, \eta, \gamma, \mathcal{D}, \frac{n_{\text{epochs}}}{K})$ ▷ Perform $\frac{n_{\text{epochs}}}{K}$ of an epoch of SGD with η learning rate and γ momentum on $\mathcal{D}$
4: $w_i \leftarrow w$
5: **end for**

Transfer attacks > LGV

Phase 2

Attack one random collected model per iteration

Algorithm 2 I-FGSM Attack on LGV

Input: (x, y) natural example, $(w_1, \ldots, w_K)$ LGV weights, n_{iter} number of iterations, ε p-norm perturbation, α step-size

Output: x_{adv} adversarial example

Shuffle $(w_1, \ldots, w_K)$ ▷ Shuffle weights

$x_{\text{adv}} \leftarrow x$

for $i \leftarrow 1$ **to** n_{iter} **do**

  $x_{\text{adv}} \leftarrow x_{\text{adv}} + \alpha \nabla_x \mathcal{L}(x_{\text{adv}}; y, w_{i \bmod K})$ ▷ Compute the input gradient of the loss of a randomly picked LGV model

  $x_{\text{adv}} \leftarrow \text{project}(x_{\text{adv}}, B_\varepsilon[x])$ ▷ Project in the p-norm ball centred on x of ε radius

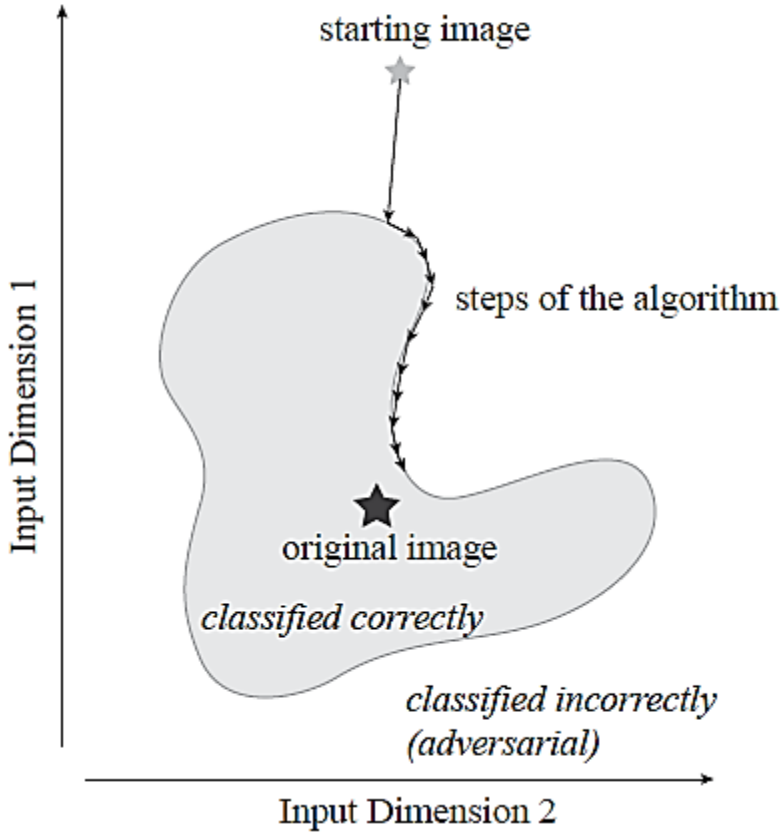
  $x_{\text{adv}} \leftarrow \text{clip}(x_{\text{adv}}, 0, 1)$ ▷ Clip to pixel range values

end for

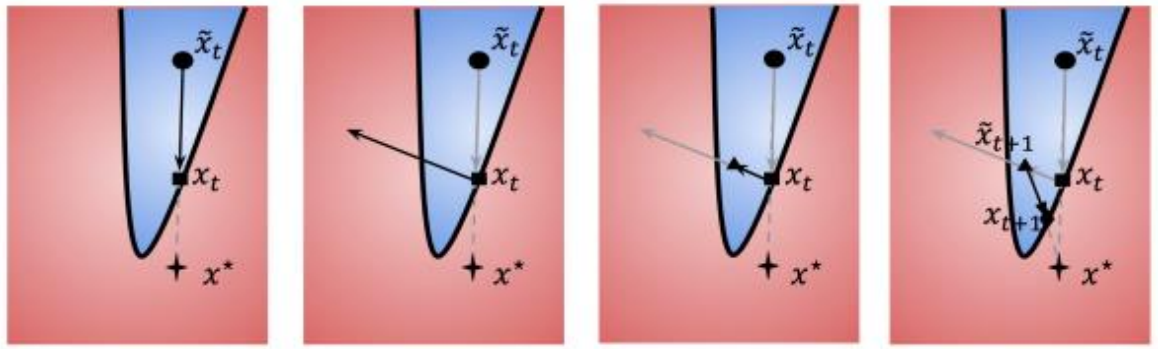
Transfer attacks

	Target							
Surrogate	RN50	RN152	RNX50	WRN50	DN201	VGG19	IncV1	IncV3
Baselines (1 DNN)								
1 DNN	45.3±2.4	29.6±0.9	28.8±0.2	31.5±1.6	17.5±0.6	16.6±0.9	10.4±0.5	5.3±1.0
MI	53.0±2.2	36.3±1.5	34.7±0.4	38.1±2.0	22.0±0.1	21.1±0.3	13.9±0.4	7.3±0.8
GN	63.9±2.4	43.8±2.4	43.3±1.3	47.4±0.9	24.8±0.3	24.1±1.0	14.6±0.3	6.8±1.2
GN+MI	68.4±2.3	49.3±2.5	47.9±1.2	52.1±1.7	28.4±0.8	28.0±0.7	17.5±0.5	8.7±0.5
DI	75.0±0.2	56.4±1.9	59.6±1.5	61.6±2.4	41.6±1.1	39.7±0.9	27.7±1.0	15.2±1.0
DI+MI	81.2±0.3	63.8±1.9	67.6±0.9	68.9±1.5	49.3±0.7	46.7±0.4	33.0±1.0	19.4±0.9
SGM	64.4±0.8	49.1±3.1	48.9±0.6	51.7±2.8	30.7±0.9	33.6±1.3	22.5±1.5	10.7±0.9
SGM+MI	66.0±0.6	51.3±3.5	50.9±0.9	54.3±2.3	32.5±1.3	35.8±0.7	24.1±1.0	12.1±1.2
SGM+DI	76.8±0.5	62.3±2.7	63.6±1.7	65.3±1.4	45.5±0.9	49.9±0.8	36.0±0.7	19.2±1.7
SGM+DI+MI	80.9±0.7	66.9±2.5	68.7±1.2	70.0±1.7	50.9±0.6	56.0±1.4	42.1±1.4	23.6±1.6
Our techniques								
RD	60.6±1.5	40.5±3.0	39.9±0.2	44.4±3.2	22.9±0.8	22.7±0.5	13.9±0.2	6.6±0.7
LGV-SWA	84.9±1.2	63.9±3.7	62.1±0.4	61.1±2.9	44.2±0.4	42.4±1.3	31.5±0.8	12.2±0.8
LGV-SWA+RD	90.2±0.5	71.7±3.4	69.9±1.2	69.1±3.3	49.9±1.0	47.4±2.0	34.9±0.3	13.5±0.9
LGV (ours)	95.4±0.1	85.5±2.3	83.7±1.2	82.1±2.4	69.3±1.0	67.8±1.2	58.1±0.8	25.3±1.9

Query attacks



Boundary attack

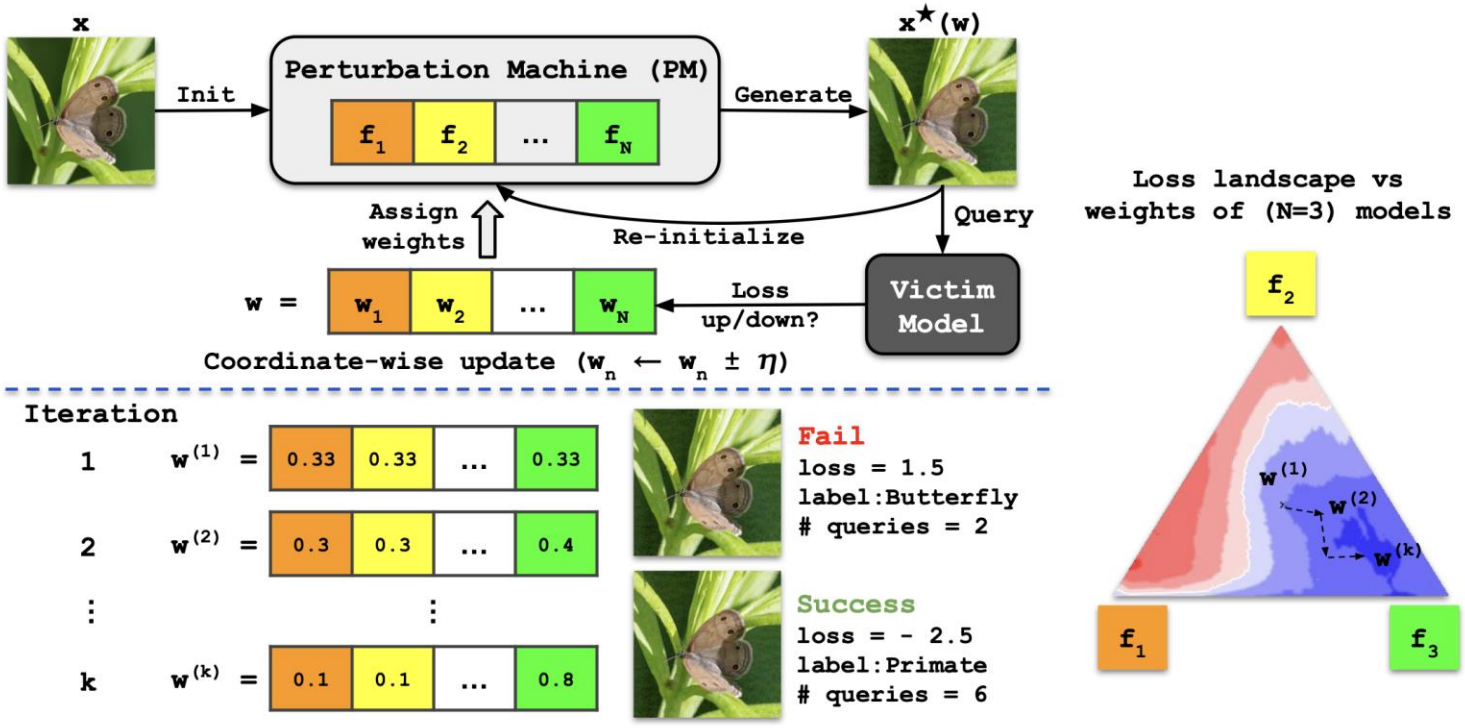


HopSkipJump attack

Query attacks

One pixel attack

Query attacks > BASES



Cai (2023) Blackbox Attacks via Surrogate Ensemble Search

Query attacks > Square

Algorithm 1: The Square Attack via random search

Input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in \{1, \dots, K\}$, number of iterations N

Output: approximate minimizer $\hat{x} \in \mathbb{R}^d$ of the problem stated in Eq. (1)

$\hat{x} \leftarrow init(x)$, $l^* \leftarrow L(f(x), y)$, $i \leftarrow 1$

while $i < N$ ***and*** $\hat{x}$ *is not adversarial* **do**

  $h^{(i)} \leftarrow$ side length of the square to modify (according to some schedule)

  $\delta \sim P(\epsilon, h^{(i)}, w, c, \hat{x}, x)$ (see Alg. 2 and 3 for the sampling distributions)

  $\hat{x}_{\text{new}} \leftarrow$ Project $\hat{x} + \delta$ onto $\{z \in \mathbb{R}^d : \|z - x\|_p \leq \epsilon\} \cap [0, 1]^d$

  $l_{\text{new}} \leftarrow L(f(\hat{x}_{\text{new}}), y)$

  if $l_{new} < l^*$ **then** $\hat{x} \leftarrow \hat{x}_{\text{new}}$, $l^* \leftarrow l_{\text{new}}$;

  $i \leftarrow i + 1$

end

Query attacks > Square

random search procedure

Algorithm 1: The Square Attack via random search

Input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in \{1, \dots, K\}$, number of iterations N

Output: approximate minimizer $\hat{x} \in \mathbb{R}^d$ of the problem stated in Eq. (1)

$\hat{x} \leftarrow init(x)$, $l^* \leftarrow L(f(x), y)$, $i \leftarrow 1$

while $i < N$ ***and*** $\hat{x}$ *is not adversarial* **do**

  $h^{(i)} \leftarrow$ side length of the square to modify (according to some schedule)

  $\delta \sim P(\epsilon, h^{(i)}, w, c, \hat{x}, x)$ (see Alg. 2 and 3 for the sampling distributions)

  $\hat{x}_{\text{new}} \leftarrow$ Project $\hat{x} + \delta$ onto $\{z \in \mathbb{R}^d : \|z - x\|_p \leq \epsilon\} \cap [0, 1]^d$

  $l_{\text{new}} \leftarrow L(f(\hat{x}_{\text{new}}), y)$

  if $l_{new} < l^*$ **then** $\hat{x} \leftarrow \hat{x}_{\text{new}}$, $l^* \leftarrow l_{\text{new}}$;

  $i \leftarrow i + 1$

end

Query attacks > Square

Sample δ from a specific distribution

l_∞ and l_2 perturbations use different distributions

Algorithm 1: The Square Attack via random search

Input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in \{1, \dots, K\}$, number of iterations N

Output: approximate minimizer $\hat{x} \in \mathbb{R}^d$ of the problem stated in Eq. (1)

$\hat{x} \leftarrow init(x)$, $l^* \leftarrow L(f(x), y)$, $i \leftarrow 1$

while $i < N$ ***and*** $\hat{x}$ *is not adversarial* **do**

  $h^{(i)} \leftarrow$ side length of the square to modify (according to some schedule)

  $\delta \sim P(\epsilon, h^{(i)}, w, c, \hat{x}, x)$ (see Alg. 2 and 3 for the sampling distributions)

  $\hat{x}_{\text{new}} \leftarrow \text{Project } \hat{x} + \delta \text{ onto } \{z \in \mathbb{R}^d : \|z - x\|_p \leq \epsilon\} \cap [0, 1]^d$

  $l_{\text{new}} \leftarrow L(f(\hat{x}_{\text{new}}), y)$

  if $l_{new} < l^*$ **then** $\hat{x} \leftarrow \hat{x}_{\text{new}}$, $l^* \leftarrow l_{\text{new}}$;

  $i \leftarrow i + 1$

end

Query attacks > Square

Initial p is a hyperparameter.
The value of p is halved after a certain iterations.

Algorithm 1: The Square Attack via random search

Input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in \{1, \ldots, K\}$, number of iterations N

Output: approximate minimizer $\hat{x} \in \mathbb{R}^d$ of the problem stated in Eq. (1)

$\hat{x} \leftarrow init(x)$, $l^* \leftarrow L(f(x), y)$, $i \leftarrow 1$

while $i < N$ ***and*** $\hat{x}$ *is not adversarial* **do**

  $h^{(i)} \leftarrow$ side length of the square to modify (according to some schedule)

  $\delta \sim P(\epsilon, h^{(i)}, w, c, \hat{x}, x)$ (see Alg. 2 and 3 for the sampling distributions)

  $\hat{x}_{\text{new}} \leftarrow$ Project $\hat{x} + \delta$ onto $\{z \in \mathbb{R}^d : \|z - x\|_p \leq \epsilon\} \cap [0, 1]^d$

  $l_{\text{new}} \leftarrow L(f(\hat{x}_{\text{new}}), y)$

  if $l_{new} < l^*$ **then** $\hat{x} \leftarrow \hat{x}_{\text{new}}$, $l^* \leftarrow l_{\text{new}}$;

  $i \leftarrow i + 1$

end

$h =$ closest interger to $\sqrt{p.w^2}$

Query attacks > Square

Making sure the new adversarial example satisfy the constraints.

Actually, the sampling distribution ensured that before clipping to $[0,1]^d$

$$||z - x||_p = \epsilon$$

so that the perturbation budget is used almost maximally

Algorithm 1: The Square Attack via random search

Input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in \{1, \dots, K\}$, number of iterations N

Output: approximate minimizer $\hat{x} \in \mathbb{R}^d$ of the problem stated in Eq. (1)

$\hat{x} \leftarrow init(x)$, $l^* \leftarrow L(f(x), y)$, $i \leftarrow 1$

while $i < N$ ***and*** $\hat{x}$ *is not adversarial* **do**

  $h^{(i)} \leftarrow$ side length of the square to modify (according to some schedule)

  $\delta \sim P(\epsilon, h^{(i)}, w, c, \hat{x}, x)$ (see Alg. 2 and 3 for the sampling distributions)

  $\hat{x}_{\text{new}} \leftarrow$ Project $\hat{x} + \delta$ onto $\{z \in \mathbb{R}^d : \|z - x\|_p \leq \epsilon\} \cap [0,1]^d$

  $l_{\text{new}} \leftarrow L(f(\hat{x}_{\text{new}}), y)$

  if $l_{new} < l^*$ **then** $\hat{x} \leftarrow \hat{x}_{\text{new}}$, $l^* \leftarrow l_{\text{new}}$;

  $i \leftarrow i + 1$

end

Query attacks > Square

Algorithm 1: The Square Attack via random search

Input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in \{1, \dots, K\}$, number of iterations N

Output: approximate minimizer $\hat{x} \in \mathbb{R}^d$ of the problem stated in Eq. (1)

$\hat{x} \leftarrow init(x)$, $l^* \leftarrow L(f(x), y)$, $i \leftarrow 1$

while $i < N$ ***and*** $\hat{x}$ *is not adversarial* **do**

  $h^{(i)} \leftarrow$ side length of the square to modify (according to some schedule)

  $\delta \sim P(\epsilon, h^{(i)}, w, c$

  $\hat{x}_{\text{new}} \leftarrow$ Project

  $l_{\text{new}} \leftarrow L(f(\hat{x}_{\text{ne}}$

  if $l_{new} < l^*$ **the**

  $i \leftarrow i + 1$

end

Algorithm 2: Sampling distribution P for l_∞-norm

Input: maximal norm ϵ, window size h, image size w, color channels c

Output: New update δ

$\delta \leftarrow$ array of zeros of size $w \times w \times c$

sample uniformly $r, s \in \{0, \dots, w - h\} \subset \mathbb{N}$

for $i = 1, \dots, c$ **do**

  $\rho \leftarrow Uniform(\{-2\epsilon, 2\epsilon\})$

  $\delta_{r+1:r+h,\, s+1:s+h,\, i} \leftarrow \rho \cdot \mathbb{1}_{h \times h}$

end

Sample the position of the square update

Query attacks > Square

Algorithm 1: The Square Attack via random search

Input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in \{1, \dots, K\}$, number of iterations N

Output: approximate minimizer $\hat{x} \in \mathbb{R}^d$ of the problem stated in Eq. (1)

$\hat{x} \leftarrow init(x)$, $l^* \leftarrow L(f(x), y)$, $i \leftarrow 1$

while $i < N$ ***and*** $\hat{x}$ *is not adversarial* **do**

  $h^{(i)} \leftarrow$ side length of the square to modify (according to some schedule)

  $\delta \sim P(\epsilon, h^{(i)}, w, c$

  $\hat{x}_{\text{new}} \leftarrow$ Project

  $l_{\text{new}} \leftarrow L(f(\hat{x}_{\text{ne}}$

  if $l_{new} < l^*$ **the**

  $i \leftarrow i + 1$

end

Use the same update for each channel:
To visualize it in a RGB image, there are only 2^3 possible colors

Algorithm 2: Sampling distribution P for l_∞-norm

Input: maximal norm ϵ, window size h, image size w, color channels c

Output: New update δ

$\delta \leftarrow$ array of zeros of size $w \times w \times c$

sample uniformly $r, s \in \{0, \dots, w - h\} \subset \mathbb{N}$

for $i = 1, \dots, c$ **do**

  $\rho \leftarrow Uniform(\{-2\epsilon, 2\epsilon\})$

  $\delta_{r+1:r+h,\ s+1:s+h,\ i} \leftarrow \rho \cdot \mathbb{1}_{h \times h}$

end

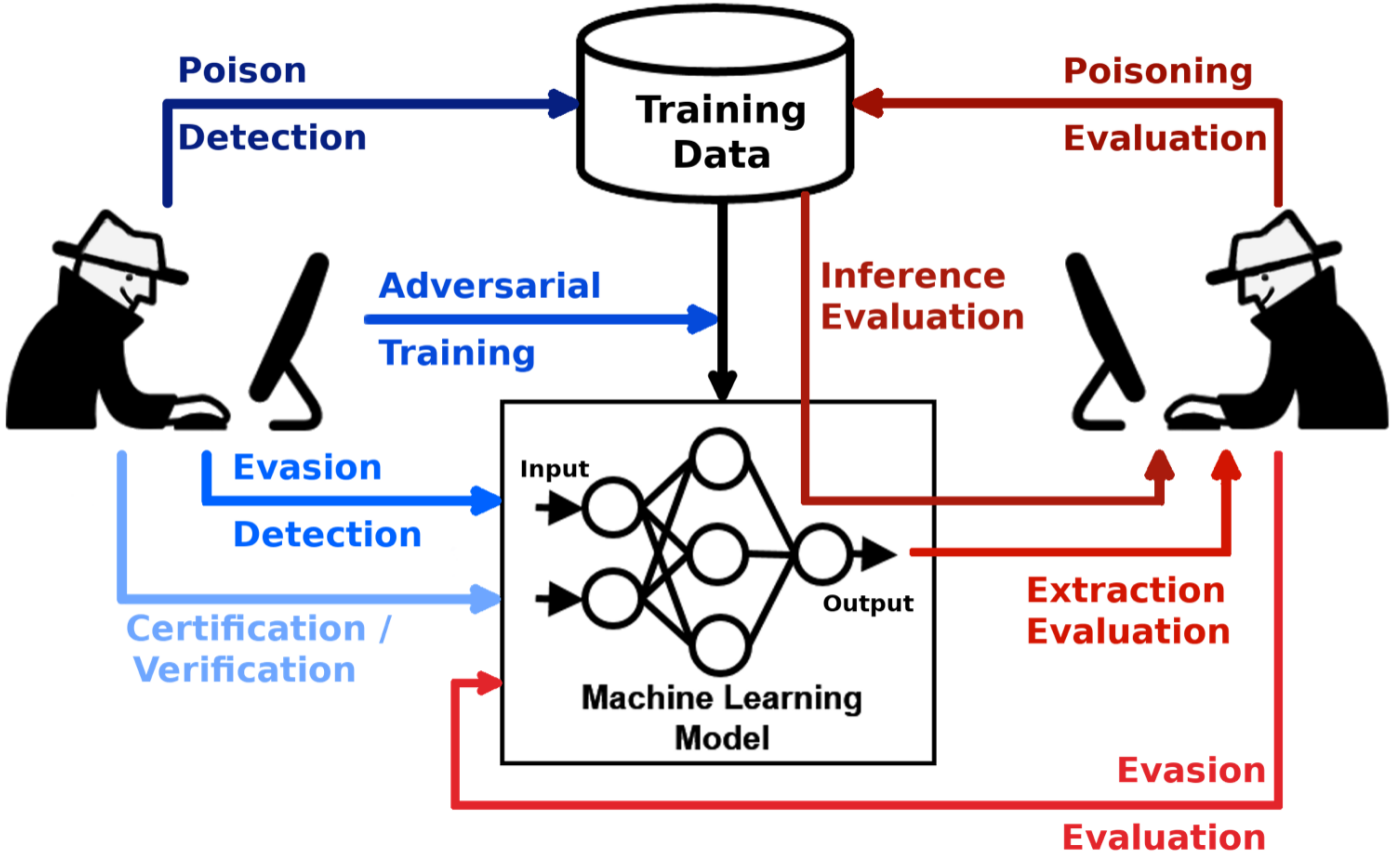
Practical Part 4

Defense against Evasion attacks

Part IV

A two-player game

- Attackers aim to produce strong adversarial examples that evade a model with high confidence while requiring only a small perturbation.



- Defenders aim to produce models that are robust to adversarial examples (i.e., the models don't have adversarial examples, or the adversaries cannot find them easily).

Main Defense strategies

- Adversarial training
- Randomized discretization
- Adversarial Robust Distillation
- Robust optimization (certified defenses)

Adversarial (Re)Training

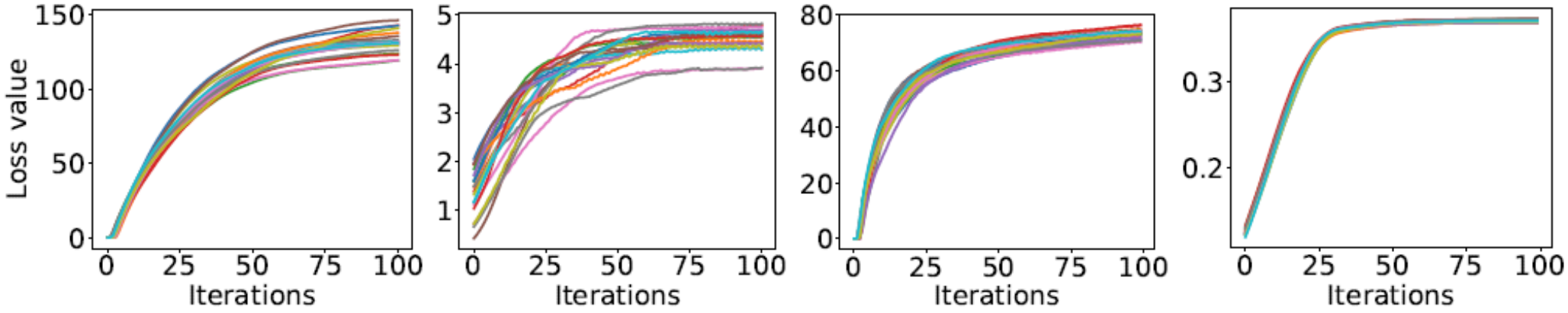
Dynamic augmentation of the triaining with adversarial examples.
Two possible strategies

Train from scratch with both original and adversarial examples at each batch → Adversarial Training

Train with original then fine-tune with adversarial examples → Adversarial Fine-tuning

Adversarial Training

When Adversarial examples are generated with PGD, we have « Madry Adversarial Training »



(a) MNIST Standard training

(b) MNIST Adversarial training

(c) CIFAR10 Natural training

(d) CIFAR10 Adversarial training

Madry (2018) Towards Deep Learning Models Resistant to Adversarial Attacks

Adversarial Training

Adversarial training is assumed to be more expensive than traditional training --->
"FGSM" when combined with random initialization is as effective as PGD-based training

Algorithm 3 FGSM adversarial training for T epochs, given some radius ϵ, N PGD steps, step size α, and a dataset of size M for a network f_θ

for $t = 1 \dots T$ **do**
  for $i = 1 \dots M$ **do**
    // Perform FGSM adversarial attack
    $\delta = \text{Uniform}(-\epsilon, \epsilon)$
    $\delta = \delta + \alpha \cdot \text{sign}(\nabla_\delta \ell(f_\theta(x_i + \delta), y_i))$
    $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
    $\theta = \theta - \nabla_\theta \ell(f_\theta(x_i + \delta), y_i)$ *// Update model weights with some optimizer, e.g. SGD*
  end for
end for

Wong (2020) Fast is better than free: Revisiting adversarial training

Adversarial Training Tricks

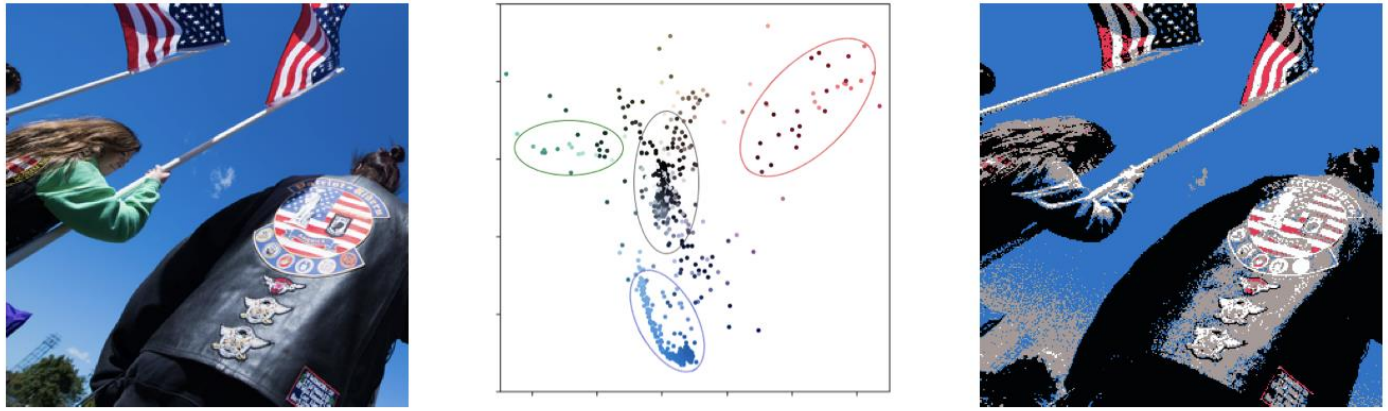
Adversarial training tricks play an important role in adversarial training

Category	Method
Data Augmentation	Mixup, RandAugment
Regularization	Weight Decay, Label Smoothing
Weight Averaging	EMA
Pre-training	21K-pre-training, SimMIM, CLIP

Liu (2023) A Comprehensive Study on Robustness of Image Classification Models: Benchmarking and Rethinking

Randomized discretization

Inject random Gaussian noise, discretize each pixel, and then feed the result into any pre-trained classifier



Zhang (2019) Defending against whitebox adversarial attacks via randomized discretization

Randomized discretization

Inject random Gaussian noise, discretize each pixel, and then feed the result into any pre-trained classifier

Gaussian randomization

$$\widetilde{x}_i = x_i + w_i \quad \text{where } w_i \sim \mathcal{N}(0, \sigma^2 I).$$

Randomized discretization: first adds Gaussian noise to each pixel and then discretizes each based on a finite number of cluster centers

$$\widetilde{x}_i = r(x_i|\mathbf{c}) := \underset{c \in \{c_1, \ldots, c_k\}}{\operatorname{argmin}} \|x_i + w_i - c\|_2$$

Zhang (2019) Defending against whitebox adversarial attacks via randomized discretization

Adversarial Robust Distillation

Knowledge Distillation: involves training a smaller neural network (student) to replicate the outputs of a larger network (teacher) to transfers the teacher's knowledge to the student

$$\mathrm{CKD}(\boldsymbol{x}, y) = (1-\lambda)\mathrm{CE}(f_S(\boldsymbol{x}), y) + \lambda \mathrm{KL}(f_S(\boldsymbol{x}), f_T(\boldsymbol{x}))$$

Adversarial Robust Distillation: The previous loss is effective in transferring clean performance, it does not seem to be able to transfer robustness against adversarial examples.

$$\mathrm{ARD}(\boldsymbol{x}, y) = (1-\lambda)\mathrm{CE}(f_S(\boldsymbol{x}), y) + \lambda \mathrm{KL}(f_S(\boldsymbol{x}'), f_T(\boldsymbol{x})).$$

x': adversarial example, CE: cross entropy, KL: Kullback–Leibler divergence

Maroto (2022) On the benefits of knowledge distillation for adversarial robustness

Adversarial Robust Distillation

Robust Soft Label Adversarial Distillation: it avoids the use of clean labels and focuses only on matching the models' outputs

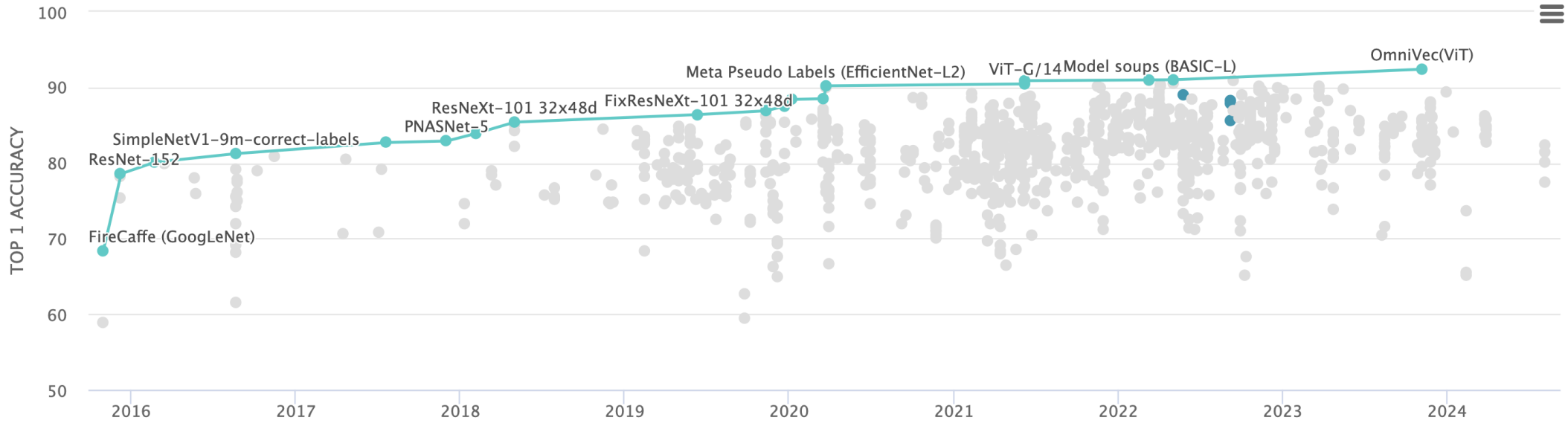
$$\mathrm{ARD}(\boldsymbol{x}, y) = (1-\lambda)\mathrm{CE}(f_S(\boldsymbol{x}), y) + \lambda \mathrm{KL}(f_S(\boldsymbol{x}'), f_T(\boldsymbol{x})).$$

$$\mathrm{RSLAD}(\boldsymbol{x}) = (1-\lambda)\mathrm{KL}(f_S(\boldsymbol{x}), f_T(\boldsymbol{x})) + \lambda \mathrm{KL}(f_S(\boldsymbol{x}'), f_T(\boldsymbol{x})).$$

x': adversarial example

Maroto (2022) On the benefits of knowledge distillation for adversarial robustness

Clean accuracy vs Robust accuracy

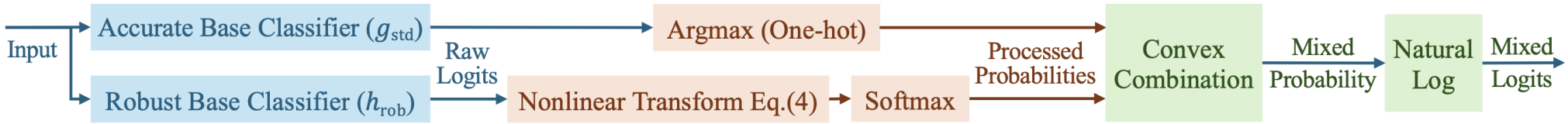


#	Model ID	Paper	Clean accuracy	Robust accuracy	Architecture	Venue
1	Amini2024MeanSparse	*MeanSparse: Post-Training Robustness Enhancement Through Mean-Centered Feature Sparsification*	77.96%	59.64%	MeanSparse ConvNeXt-L	arXiv, Jun 2024
2	Liu2023Comprehensive_Swin-L	*A Comprehensive Study on Robustness of Image Classification Models: Benchmarking and Rethinking*	78.92%	59.56%	Swin-L	arXiv, Feb 2023
3	Bai2024MixedNUTS	*MixedNUTS: Training-Free Accuracy-Robustness Balance via Nonlinearly Mixed Classifiers*	81.48%	58.50%	ConvNeXtV2-L + Swin-L	arXiv, Feb 2024

https://paperswithcode.com/sota/image-classification-on-imagenet
croce (2021) RobustBench: a standardized adversarial robustness benchmark

Clean accuracy vs Robust accuracy

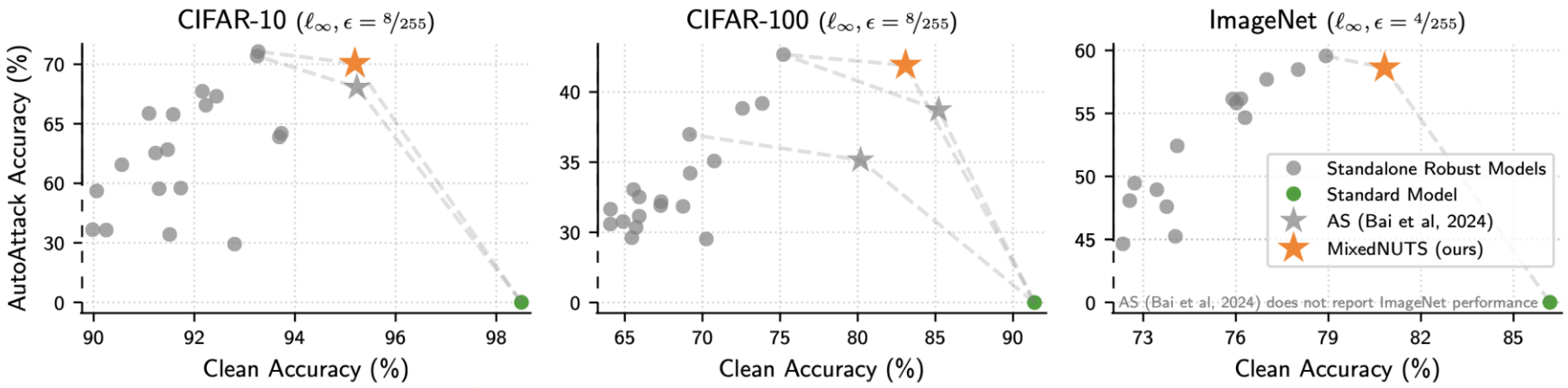
Adversarial robustness often comes at the cost of degraded accuracy



Bai (2024) MixedNUTS: Training-Free Accuracy-Robustness Balance via Nonlinearly Mixed Classifiers

Clean accuracy vs Robust accuracy

Adversarial robustness often comes at the cost of degraded accuracy



Bai (2024) MixedNUTS: Training-Free Accuracy-Robustness Balance via Nonlinearly Mixed Classifiers

Model certification

Existing robustness metrics are « Empirical »

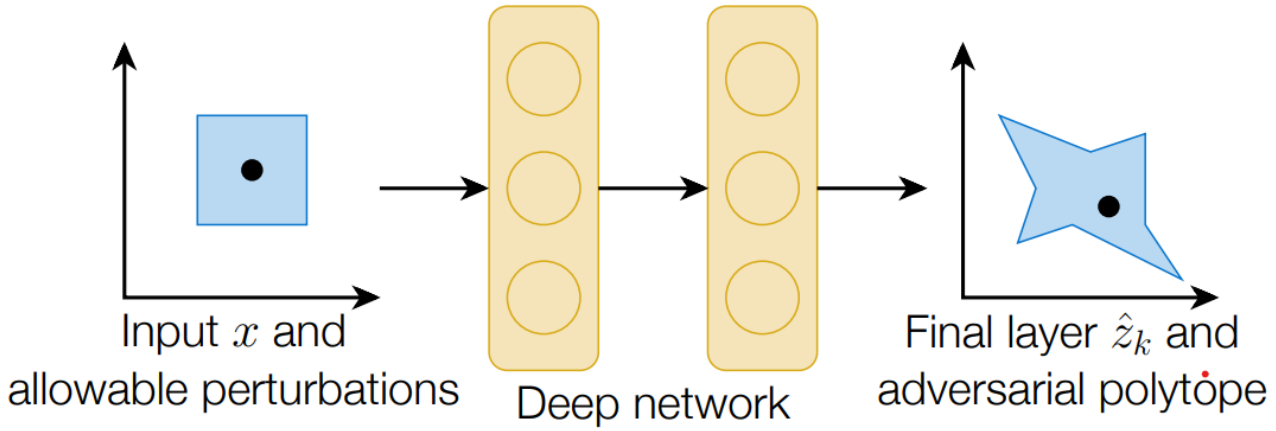
They can « show » where a model is weak
But never where a model is robust

Can we provide a bound of maximum loss after a bounded attack ?

Model certification

Challenge

A regular (convex) adversarial region may be highly complex and concave after layers of neural network



Model certification

Many solutions, 2 will be covered in an upcoming (11/11/2024)

Randomized Smoothing

Train a base classifier trained with Gaussian corruption then smooth it

Activation Bound Relaxation

Relax the bounds of the reachable region so that it is convex and easier to work with

Practical Part 5