UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DIPARTIMENTO
DI INFORMATICA

Adversarial Machine Learning : an introduction

Dr. Giuseppina Andresini

Artificial Intelligence

- **AI is the new electricity**
  - AI is the new electricity. It will transform every industry and create huge economic value.
  - Technology like supervised learning is automation on steroids. It is very good at automating tasks and will have an impact on every sector – from healthcare to manufacturing, logistics and retail (cit. Andrew NG)

Deep Learning

Modern AI is:

Numerical Optimization + Data

↓

Deep Learning, Machine Learning

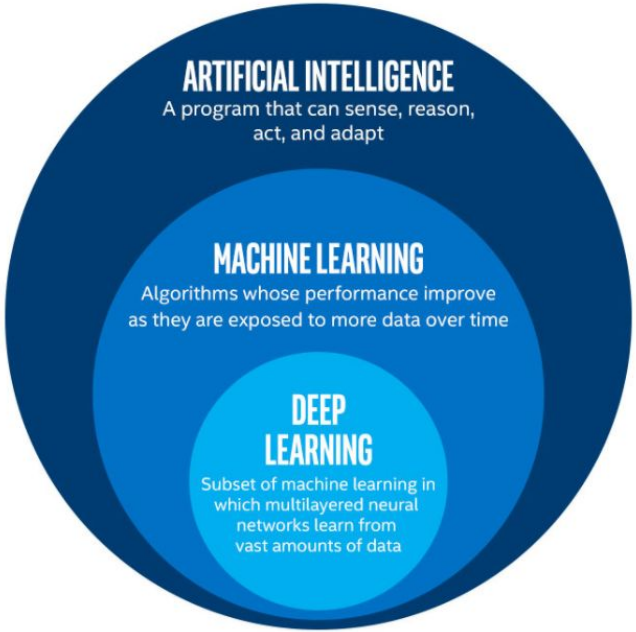
Deep Learning (DL) is:

1. A branch of **Machine Learning (ML)**
2. A set of algorithms that attempt to model **high-level abstractions** representation of data **from raw data**
3. Inspired by the function of the **biological brain**
4. Composed by multiple processing layers of multiple non-linear transformations

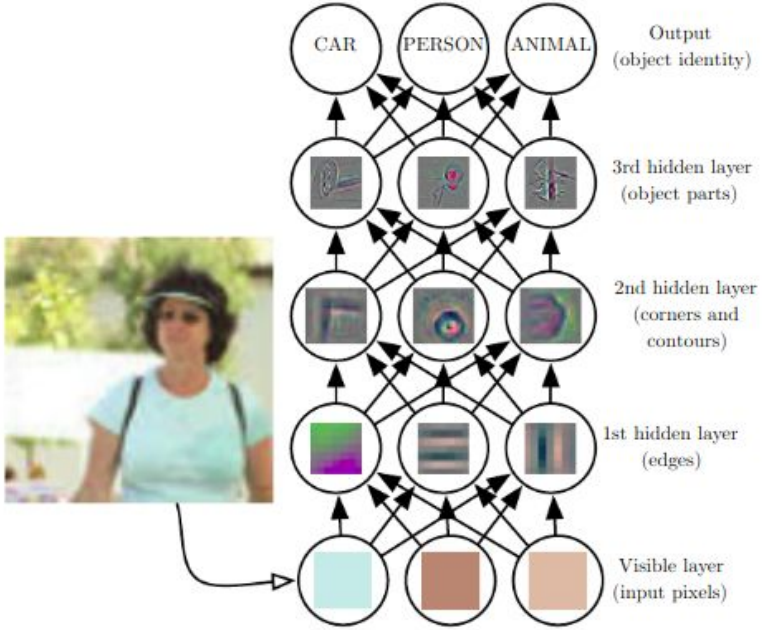
DL is a branch of ML

- Data preparation:
  - ML requires **structured data as input**
    - Human intervention to identify the features based on the data type
      - e.g., for images pixel value, shape, rotation
  - DL extract the high-level abstractions representation of data **from raw data** without human intervention

- Approach
  - ML need to human intervention to provide adjustments to improve the accuracy of the model
  - DL performs an **end-to-end learning** :
    - From raw data to a task (e.g., classification) the model learn how to do it automatically



End-to-end learning



Source: https://www.deeplearningbook.org/

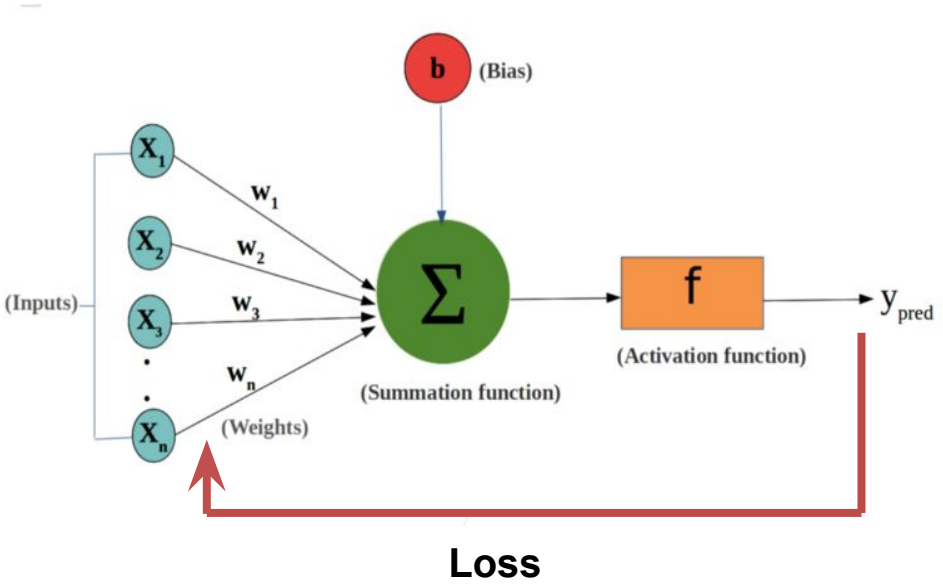
- DL models extract learns representations (features) from the image in automatically manner.
  - For each hidden layer, given the pixels:
    - the **1st hidden layer** identify edges, by comparing the brightness of neighboring pixels.
    - the **2nd hidden layer** search for corners and extended contours (as collections of edges)
    - the **3rd hidden layer** detect parts of specific objects (finding specific collections of contours and corners).
    - Finally, this description of the image in terms of the object parts it contains is used to r**ecognize the objects present in the image.**

Deep learning : a recap

How NNs learn?

Learning by **trial-and-error**

- **Trial:**
  - Compute the prediction of a given input data
- **Evaluate:**
  - Provide a feedback of the correctness of the prediction
  - Comparing the actual output with the expected input
    - loss= y-ŷ
- **Adjust:**
  - The weights are iteratively adjusted in order to reduce the error



The NN is refined over time to find the weights that minimise the error on the training and provide more accurate predictions

Loss function

- The loss function measures how much the predicted output label is different from the **expected output (ground truth label)**.
- The error at each hidden layer is an average of the evaluated error
- The choice of loss function is sensitive to the application task:

Mean squared error (MSE)

- The most commonly used loss function for regression
- Squared differences between true and predicted values

$$L_{\text{mse}}(y, \hat{y}) = ||y - \hat{y}||^2$$

Cross entropy loss

- Used in classification task
- Measures the performance of a classification model whose output is a probability between 0 and 1 by increasing its value when the predicted probability diverges from the actual label.

$$L_{\text{multiclasscrossentropy}}(y, p(\hat{y})) = -\sum_{c} y_c log p(\hat{y})_c$$

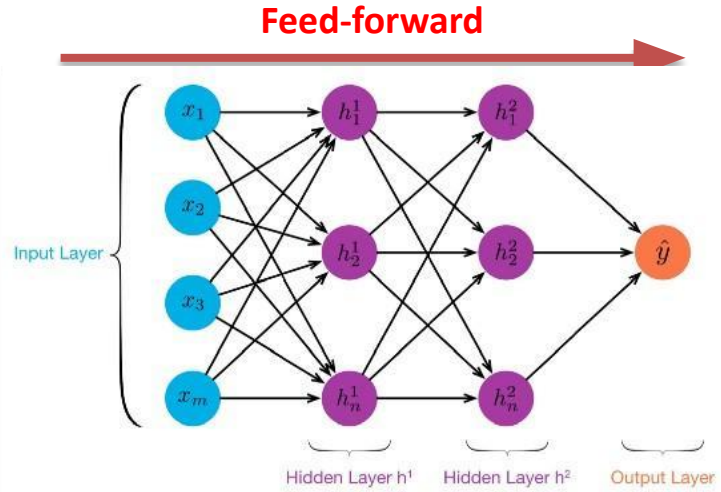
Training a NN with backpropagation

KDDE

- In **single-layer NNs** the error (loss) is computed as a direct function of weights
- In **multi-layer NNs** the loss is a composition function of the weights in earlier layers:
  - The gradient of composition function is performed by using the **backpropagation algorithm:**
    - Two phases: ***forward*** and ***backward***

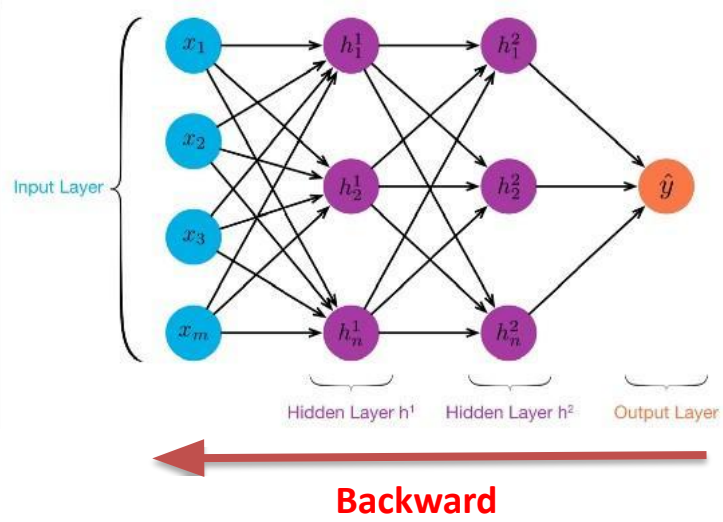
Training: forward phase

- The inputs are fed into NNs
- Computation across layers using the current set of weights
- Final prediction:
  - **Compare** the output with training
  - **Compute** the derivative of the loss with respect to the output



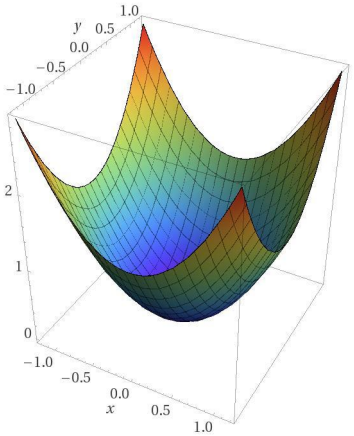
Training: backward phase

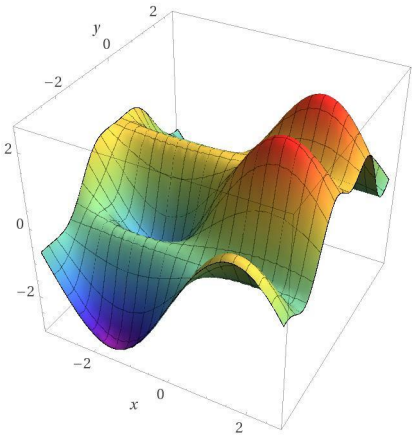
- Learn the **gradient** of a loss function with respect to the weights
  - Using the **chain rule**
  - The gradient is used to update the weights
  - Starting from output layer to input

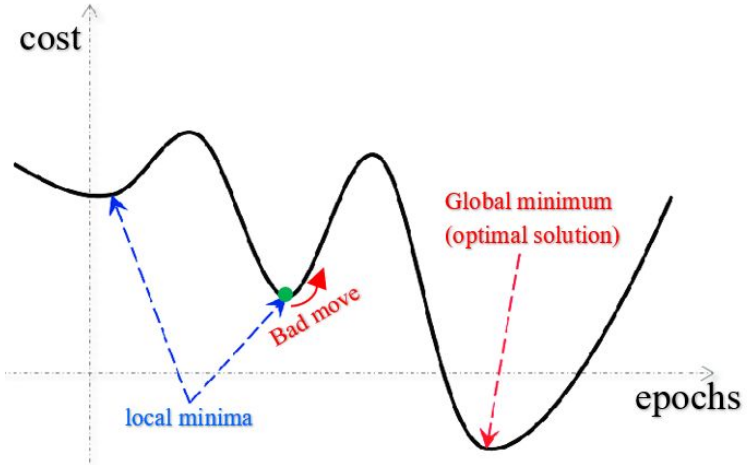


Gradient Descent

- First-order iterative optimization algorithm
  - Goal: find a **local minimum** of a differentiable function (loss function)
  - To train NNs and update the weights
    - To find the weights that minimize the cost function (loss function)
    - Partial derivative with y respect to its inputs x (*dy/dx)*

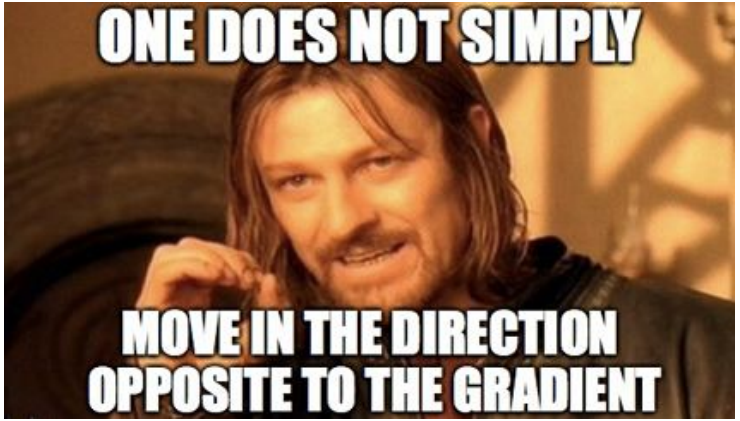






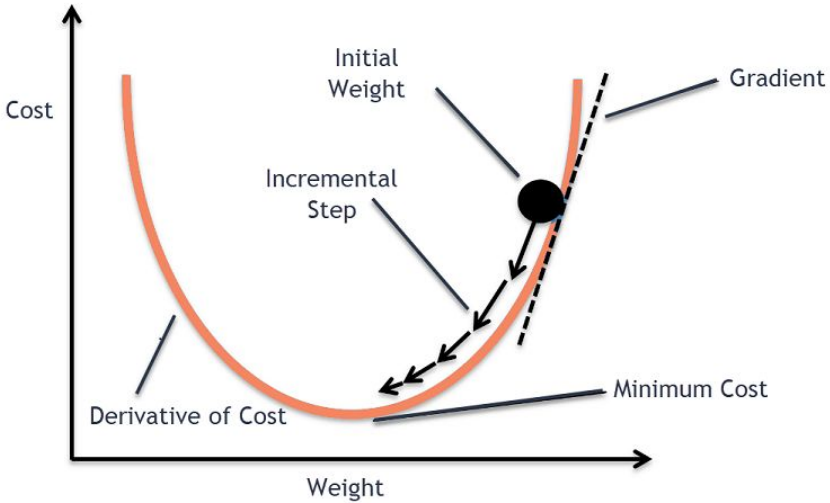
Gradient Descent

- A gradient measure how much the output of a function changes if you change the input
  - Measures the change in all weights with regard to the change in error.
  - Slope of a function
    - The steeper the slope and the faster a model can learn.
    - If the slope is zero, the model stops learning.



Gradient Descent

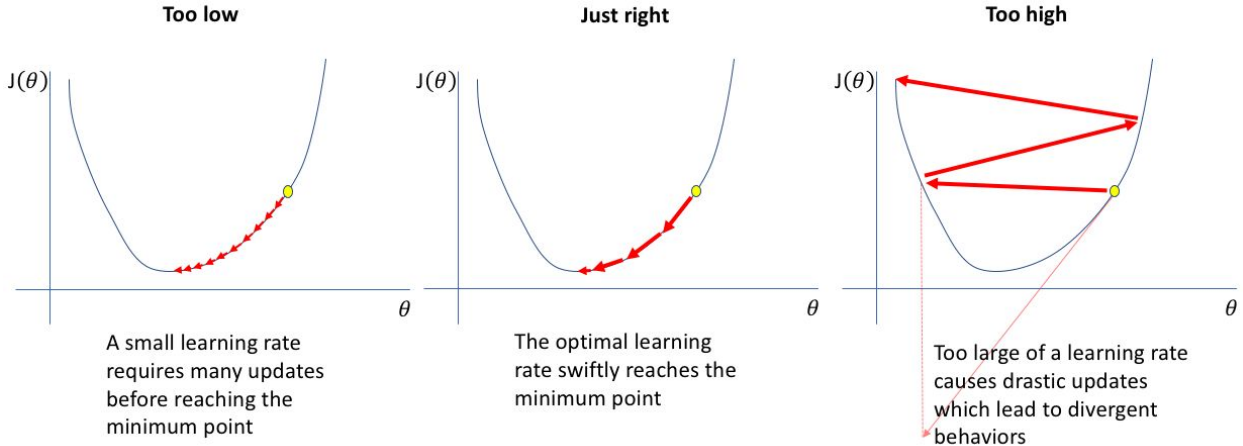
- Let us consider a **cost function** ***y=f(x)=L(x,y)***
- **Goal: minimizing *f(x)***
- On each iteration:
  - Compute the **slope (gradient)** of f(x) at current point
  - Update the parameters in **the opposite direction** of the gradient
    - the parameters where the gradient gives the direction of the steepest ascent
    - by moving x in small steps with opposite sign of the derivative
    - the size of the step on each iteration is determined by the **learning rate *α***
    - Until we reach a local minimum

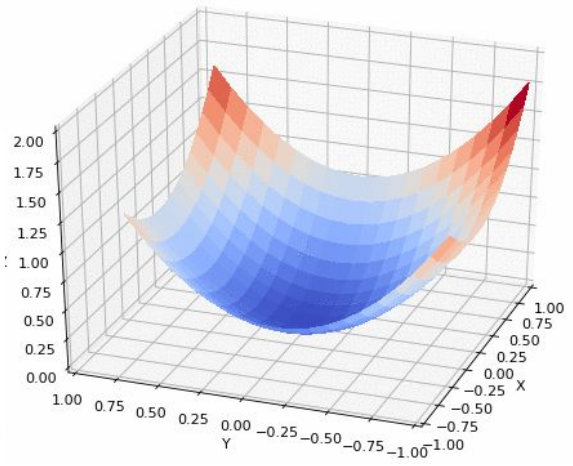


$$W' = W - \alpha \nabla_W L(x, y, W)$$

Gradient Descent

- Learning rate: for gradient descent
  - scale the magnitude of our weight updates in order to minimize the network's loss function





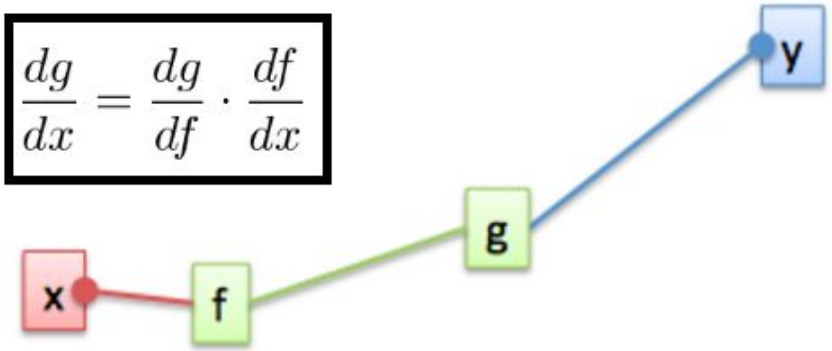
$$W' = W - \alpha \nabla_W L(x, y, W)$$

Chain rule

- In multi-layer NN the loss is a composition function of the weights in earlier layers
- The gradient is computed using backpropagation algorithm that leverages a **chain rules**
  - It calculates the error gradient in terms of summations of local-gradient products

$$y = g(f(x))$$

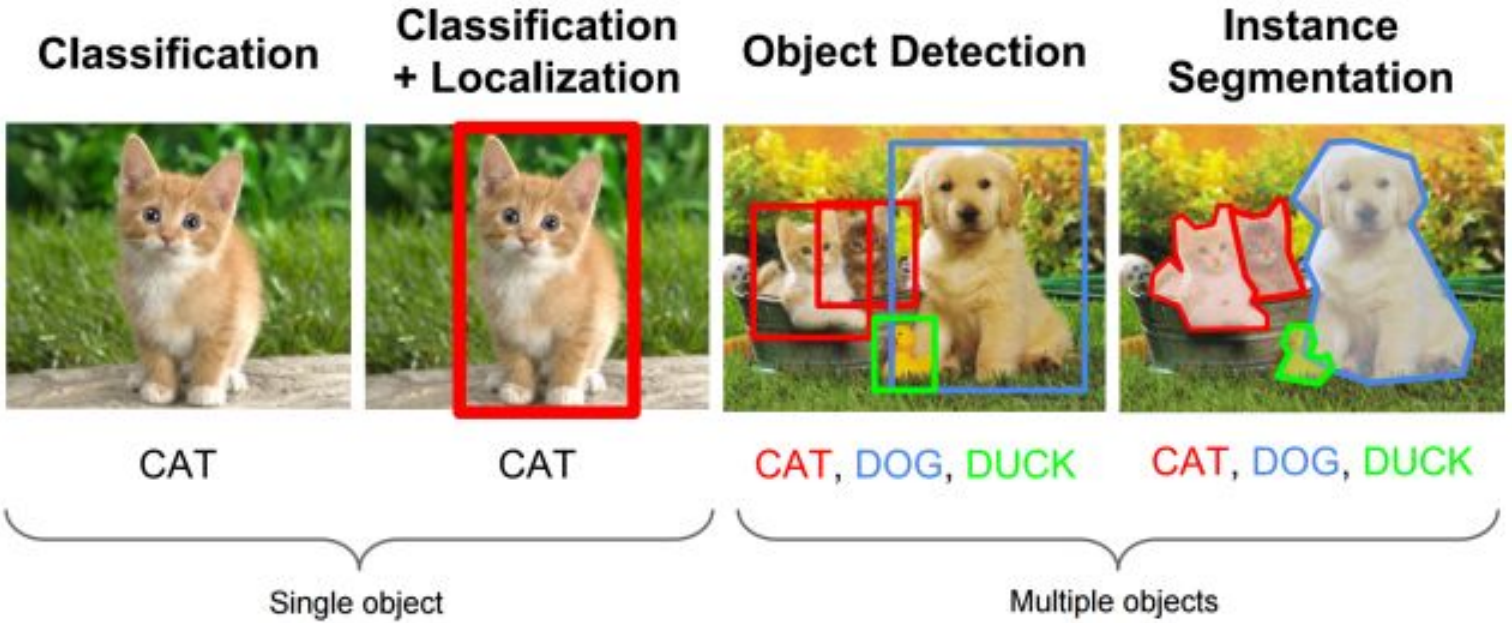
$$\frac{dg}{dx} = \frac{dg}{df} \cdot \frac{df}{dx}$$



source:
https://bit.ly/3Gm5Ap5

Deep Learning : applications

Object classification and object detection



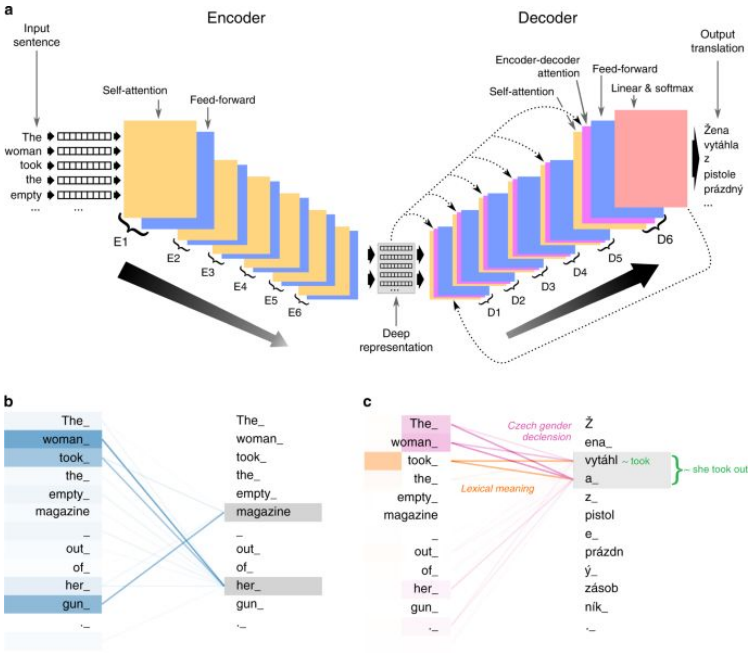
Is this a cat?

What is there in image and where?

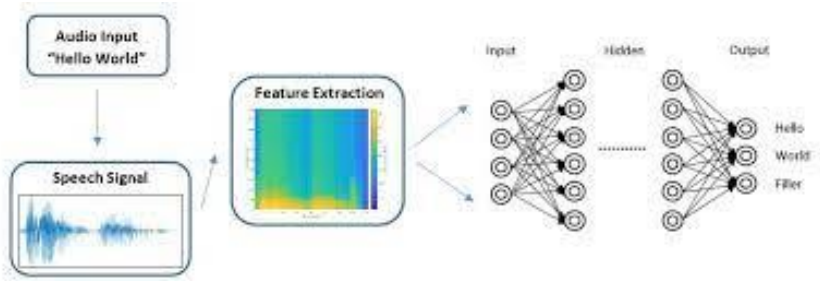
Which pixels belong to which object?

Text

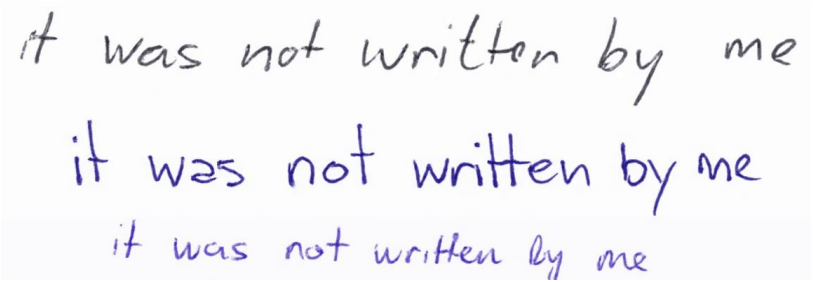
Language Translation



Object recognition: speech



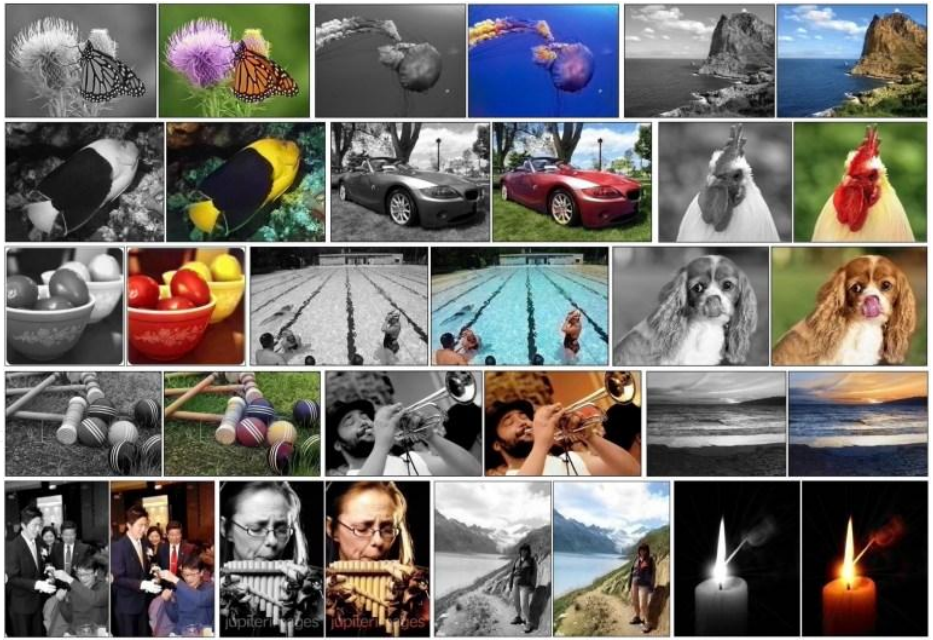
Automatic handwriting generation



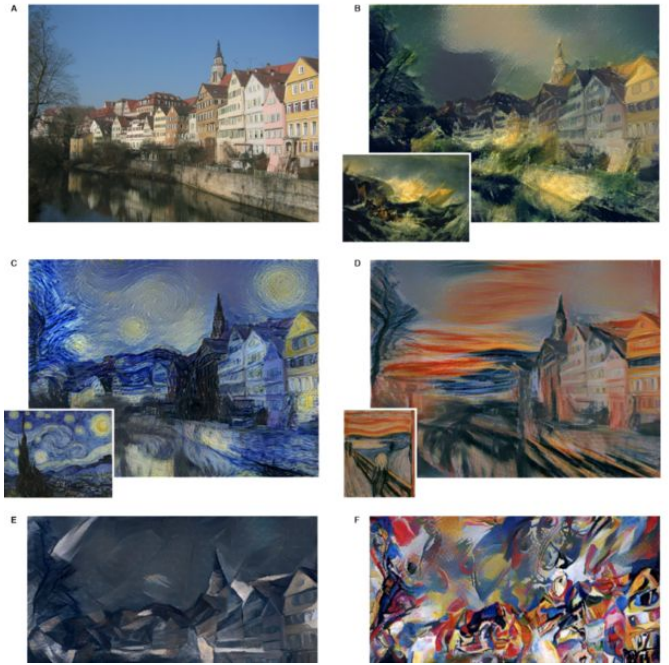
Art

KDDE

Colorize Black & White Images

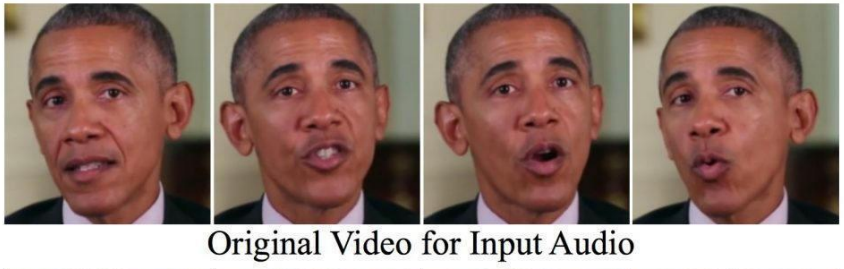


Style transfer

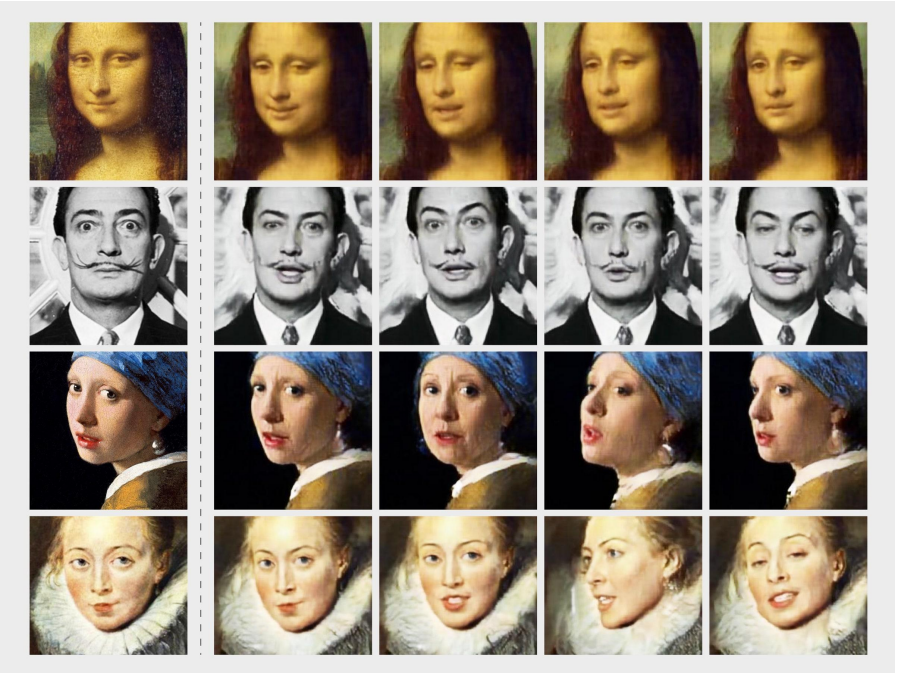


Images

Faking faces and video



Synthesizing Obama: Learning Lip Sync from Audio
https://www.youtube.com/watch?v=9Yq67CjDqvw



Images

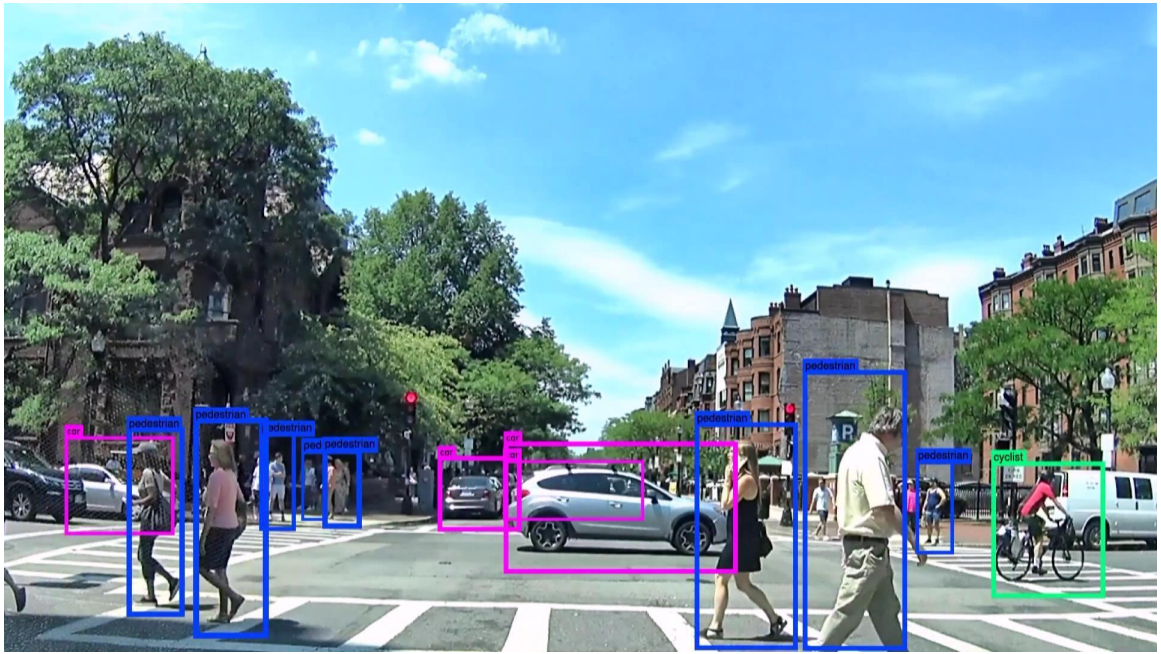
Image reconstruction

Images

Face creation

Self driving

Self driving cars



Cybersecurity

Cybersecurity: intrusion detection, malware detection, fraud detection…

Dark side of AI: adversarial learning

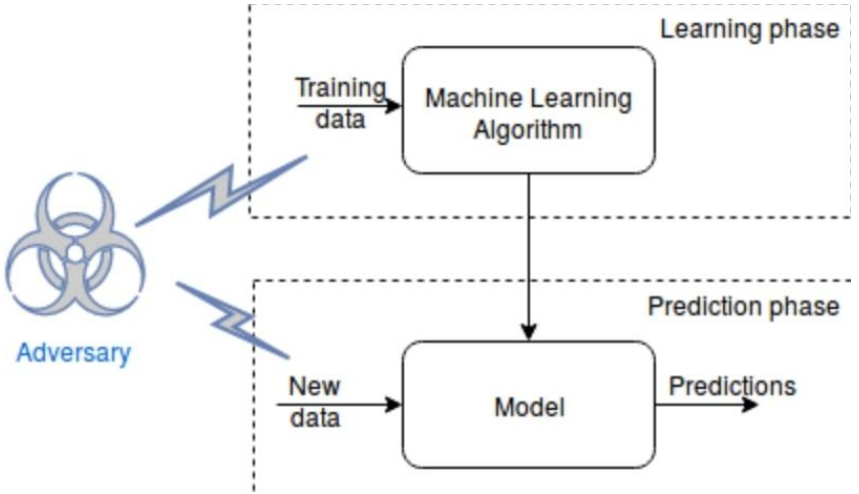
AI in Cybersecurity

KDDE

- Cybersecurity products are increasingly incorporating **Artificial Intelligence (AI)** technologies (e.g., Machine Learning, Deep Learning)
  - **PRO:**
    - Increase the **ability** to detect attacks
    - Reduce the **time** spent for threat detection and incident responses
  - **CONS**:
    - Attackers use AI to better understand their targets and design **new attacks**
    - AI systems are vulnerable to **adversarial attacks**

Adversarial machine learning

- Study of **attacks** on machine learning (and deep learning) algorithms
  - Collection of techniques to train neural networks on how to spot intentionally misleading data or behaviors
  - **Defenses** against such attacks
    - preemptively locate vulnerabilities and craft more flexible learning algorithms



https://deepai.org/machine-learning-glossary-and-terms/adversarial-machine-learning

Adversarial biometric

Chaos Computer Club (CCC), demonstrated that a fingerprint of the phone user, photographed from a glass surface, was enough to create a fake finger that could unlock an iPhone 5s secured with TouchID.

iPhone 5s fingerprint sensor 'hacked' within days of launch

A group of German hackers has found a way to bypass Apple's TouchID, and claims that fingerprint biometrics is an unsuitable method of access control.

Police are forced to destroy the fingerprints and DNA samples of terror suspects who are not charged Photo: ALAMY

By Sophie Curtis
9:33AM BST 23 Sep 2013
Follow 11.7K followers

iPhone
Technology » Apple »
Technology News »
Sophie Curtis »

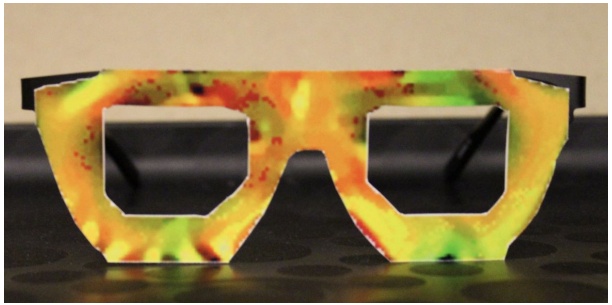
The group, known as the Chaos Computer Club (CCC), demonstrated that a fingerprint of the phone user, photographed from a glass surface, was enough to create a fake finger that could unlock an iPhone 5s secured with TouchID.

Fingerprint images coming from a live (left side) and a fake finger (right side), almost indistinguishable.

Fake fingers provided by PRA Lab for the ICB 2013 – Spoofing Challenge

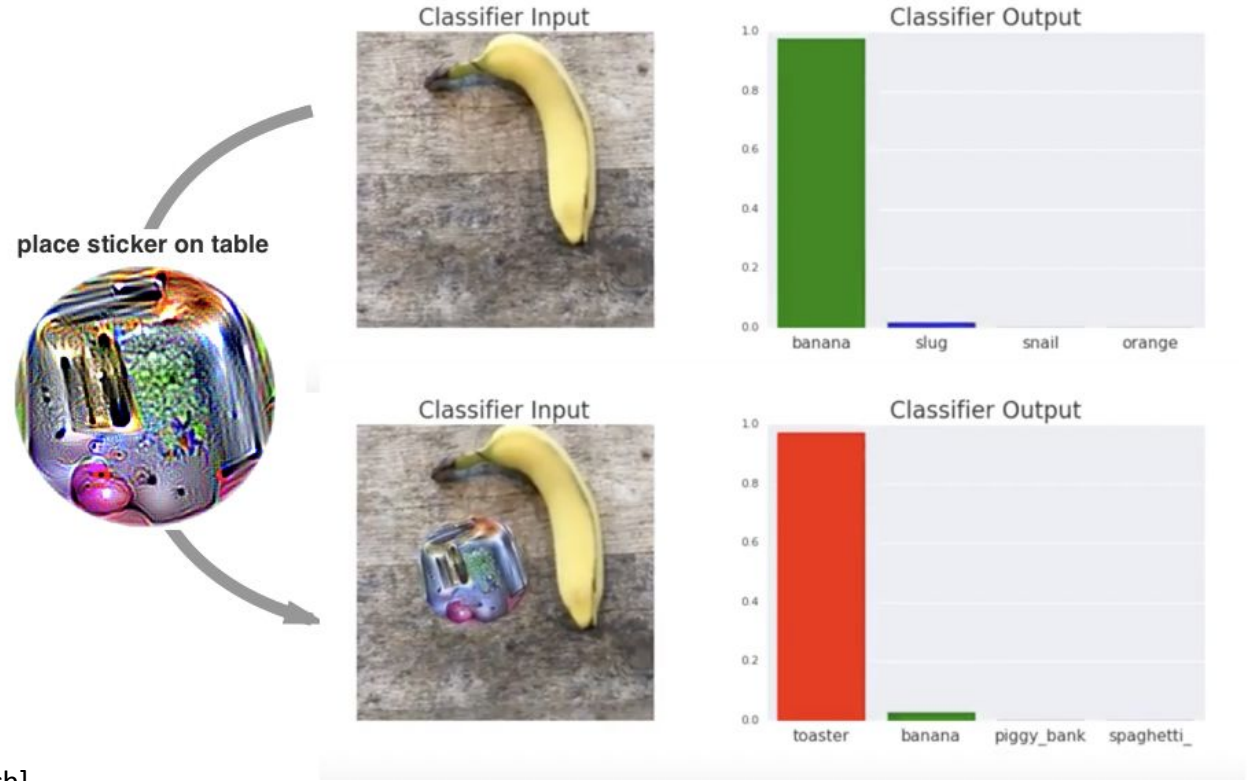
https://www.euronews.com/2014/03/17/behind-the-mask-of-biometric-security?jwsource=cl

Adversarial Image



[Sharif et al (ACM CCS 2016) -Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition]

Adversarial Image



[Brown et al 2017 -Adversarial Patch]

Adversarial audio

https://nicholas.carlini.com/code/audio_adversarial_examples/

Mozilla implementation of DeepSpeech.

[Reveal Transcription] “okay google browse to evil dot com”

[Reveal Transcription] “without the dataset the article is useless”

[Carlini et al -Audio Adversarial Examples: Targeted Attacks on Speech-to-Text]

What are adversarial examples?

Puppy or Muffin?

SheepDog or Mop?

Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake; they're like optical illusions for machines.

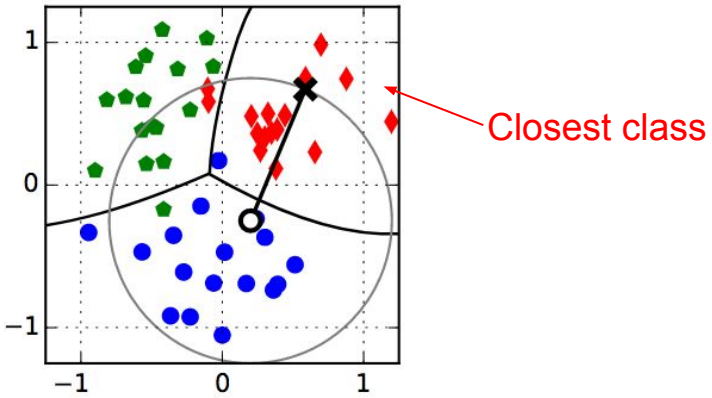
https://openai.com/blog/adversarial-example-research/

Categorization of adversarial attacks

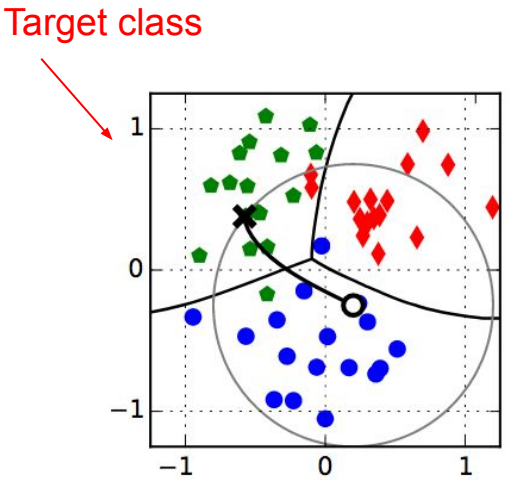
- **Goal:**
  - Error generic -> untarget attacks
  - Error specific -> target attacks
- **Capability**
  - Poison
  - Evasion
- **Knowledge**
  - White box
  - Black box

Attacker's goal

- **Error Specificity**:
  - sample misclassified as a specific class or misclassified as any of the classes different from the true class



Error-generic

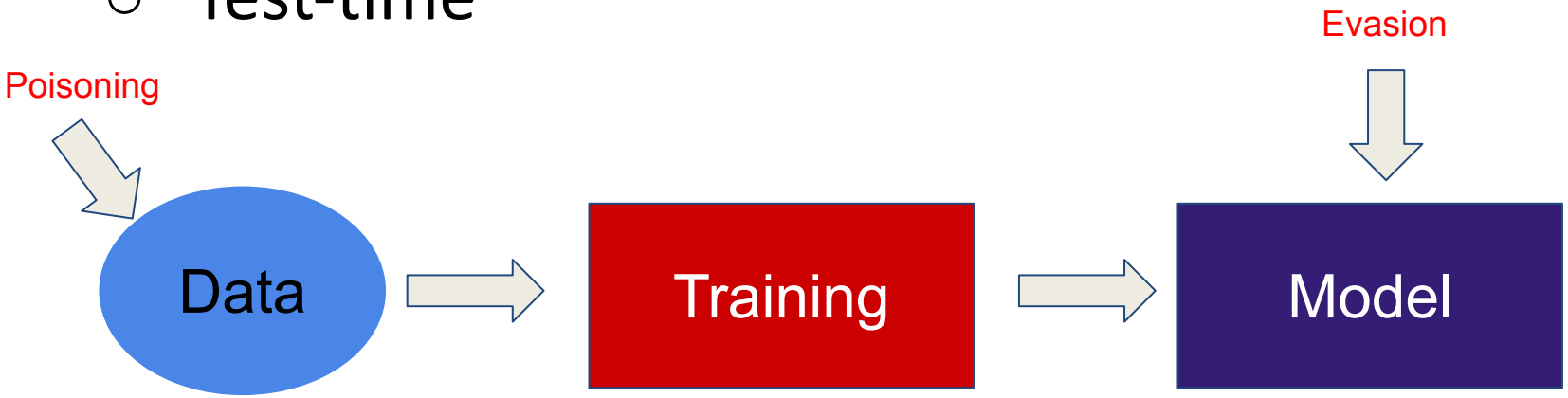


Error-specific

Melis et al. 2017. Is Deep Learning Safe for Robot Vision?

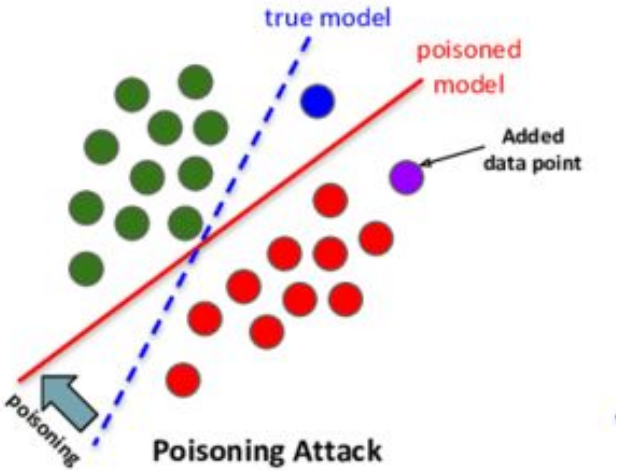
Attacker's capability

- **Poisoning**:
  - Training-time
- **Evasion**:
  - Test-time

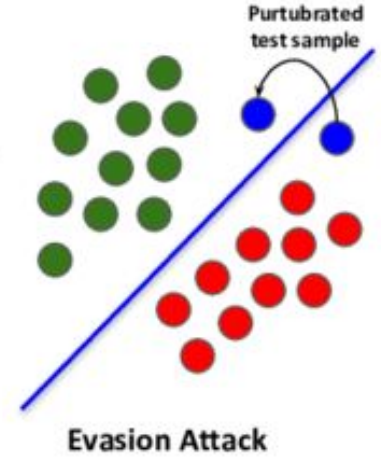


Attacker’s capability

- Injection of bad data into the model’s training



- A perturbed example is create to be misclassified by the model
  - e.g., spam emails -try a number of different email contents against the model and discover a way to get their spam email classified as innocuous.



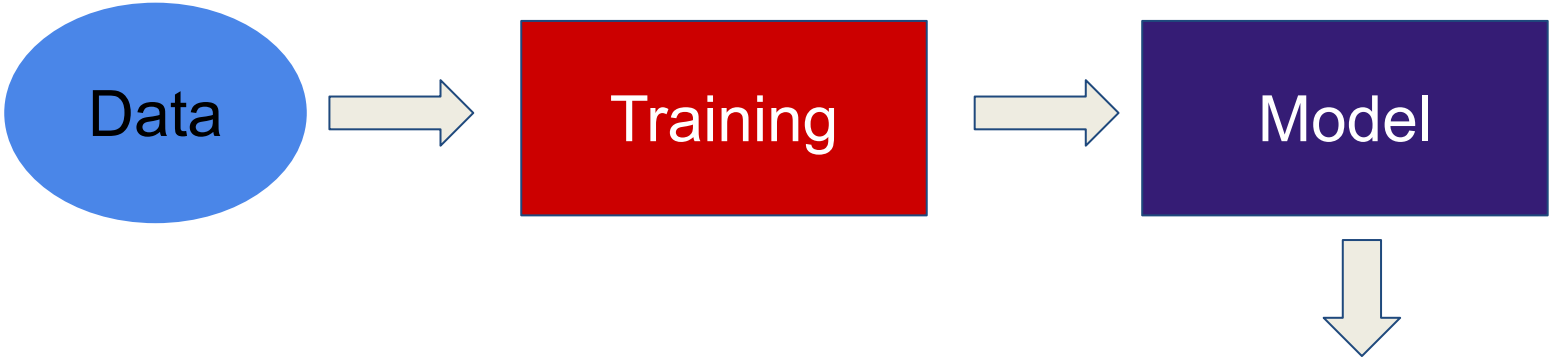
Poisoning

KDDE

Strategies:

- **Data modification:**
  - Label modification of training data
    - [Biggio et al 2011 -Support Vector Machines Under Adversarial Label Noise]
- **Data injection:**
  - Augment data with adversarial examples
    - [Barreno et al 2001 - Can machine learning be secure?]
- **Logic corruption:**
  - The adversary has to ability to meddle with the learning algorithm
    - [Kloft and Laskov 2010 - Online Anomaly Detection under Adversarial Impact]

Poisoning



“Giuseppina Andresini”

Poisoning: data injection

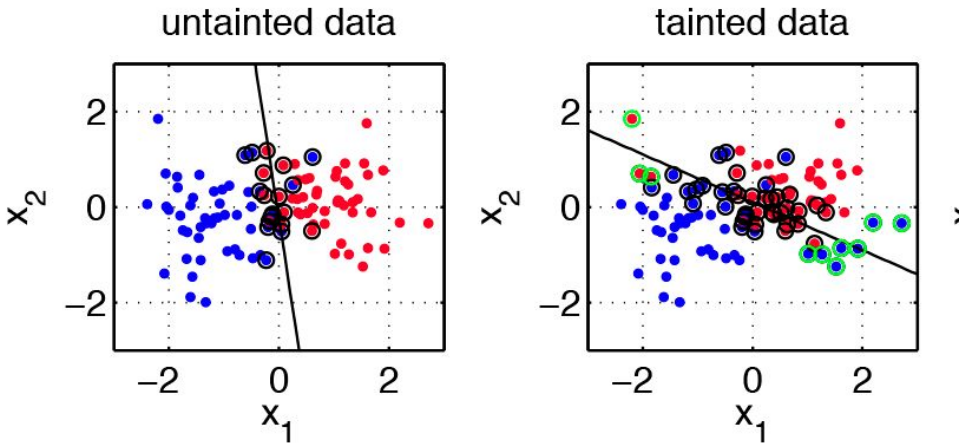
✖ 30

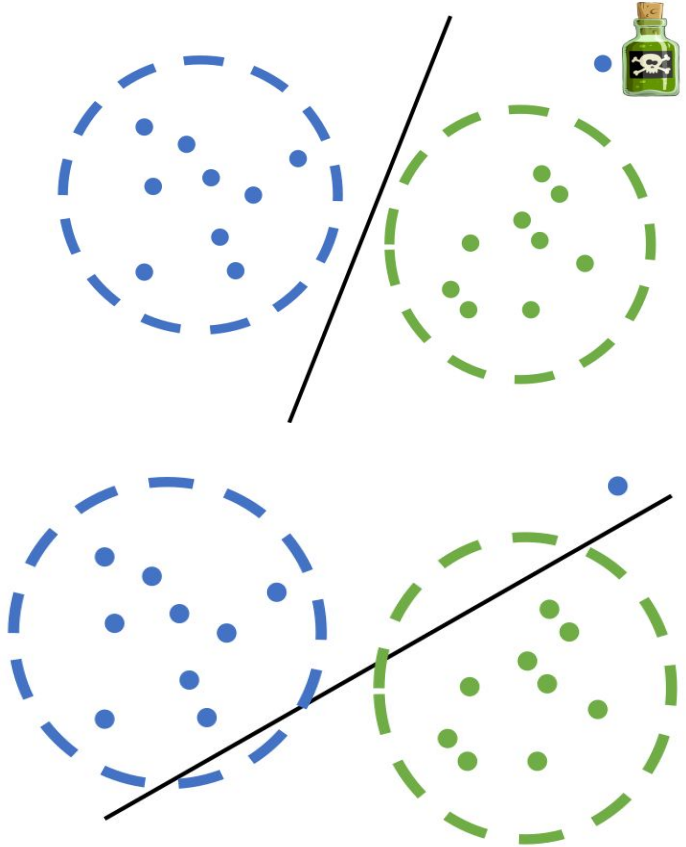
Jennifer Lawrence

Data → Training → Model

“Jennifer Lawrence”

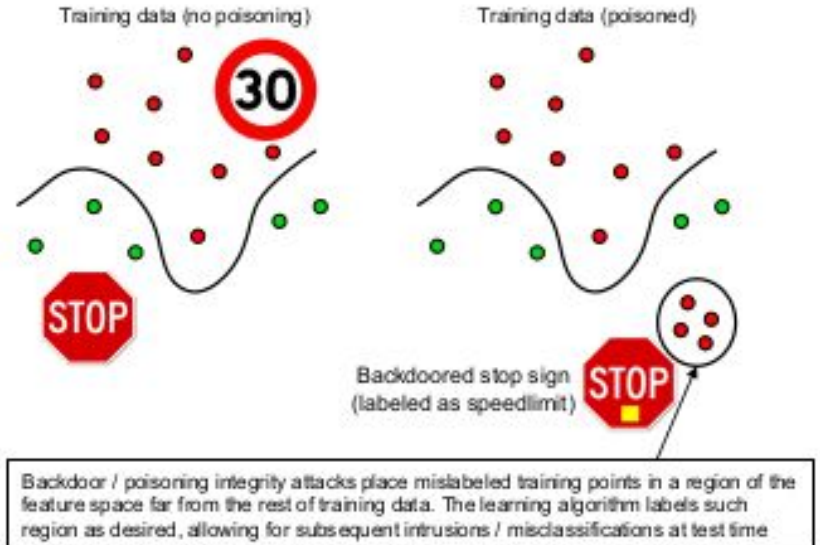
Poisoning: data modification





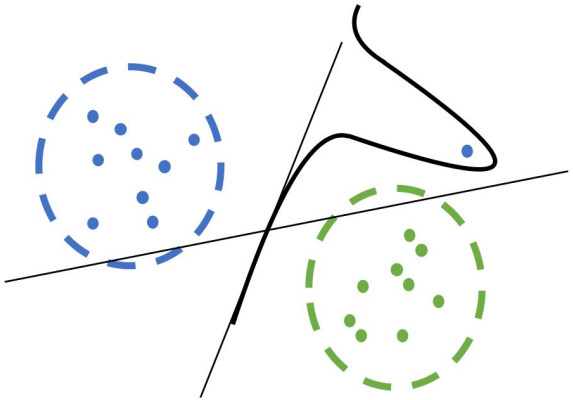
[Biggio et al (2011) -Support Vector Machines Under Adversarial Label Noise]

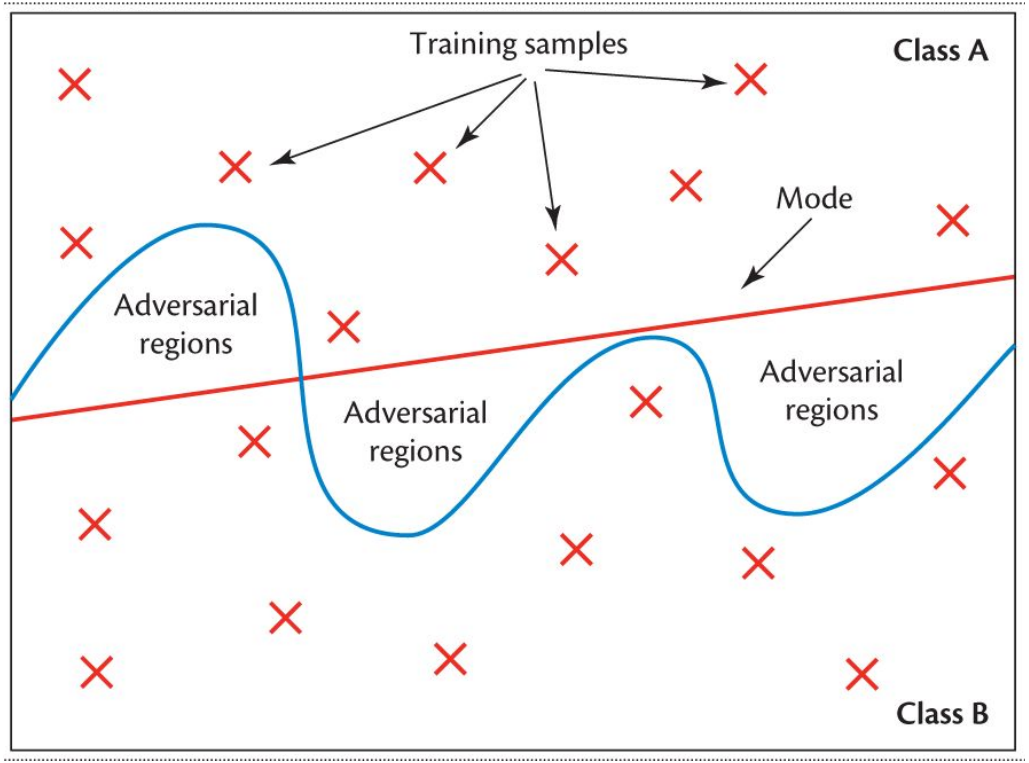
Poisoning: data modification



[Biggio et al (2011) -Support Vector Machines Under Adversarial Label Noise]

Poisoning: deep neural network

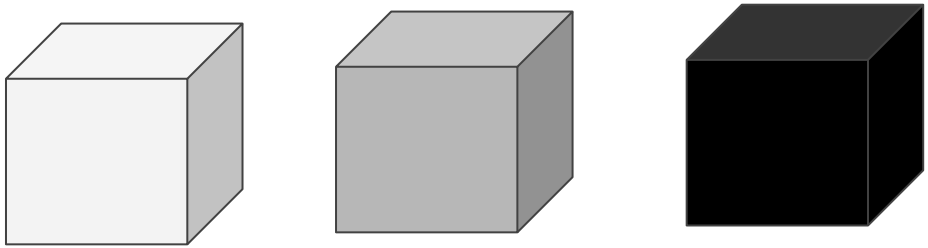




Evasion: **attacker's knowledge**

KDDE

- **White box** attacks:
  - Knowledge about structure and parameters of the target model
- **Gray box** attacks:
  - Knowledge about feature representation or learning algorithm
- **Black box** attacks:
  - No knowledge about feature representation and parameters of the model

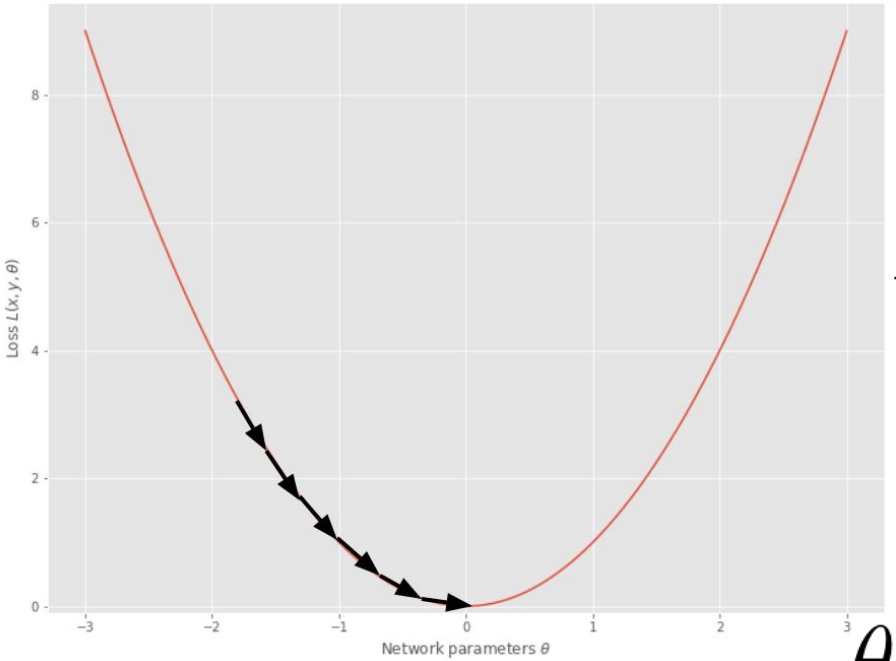


White box attacks

KDDE

- Fast Gradient Sign Method (FGSM) - 2015
- Jacobian Based Saliency Map (JSMA) -2016
- DeepFool - 2016
- Carlini & Wagner - 2017
- …….

Training neural network



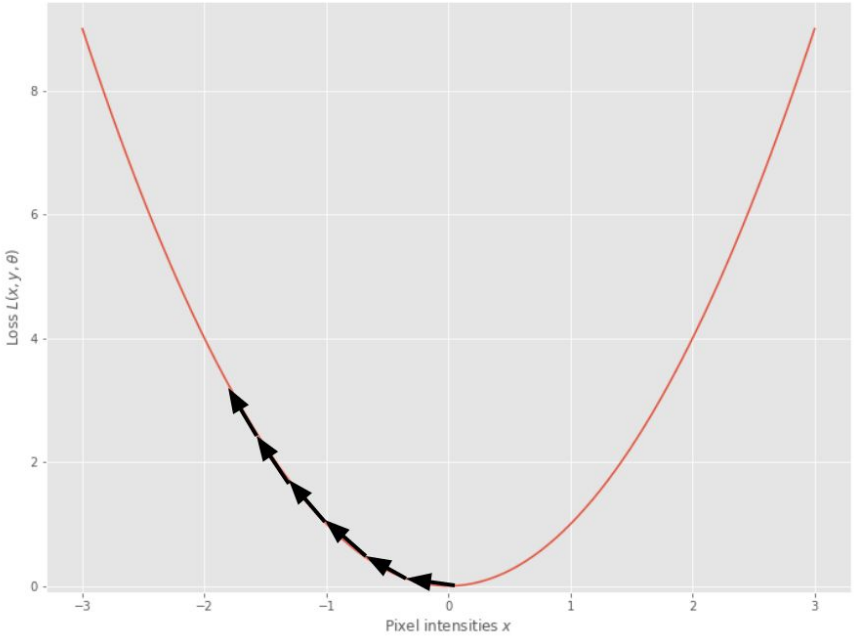
Goal: finding the **minimum** of this loss function is equivalent to finding a good set of network parameters.

$$L(\boldsymbol{x}, y, \theta) = (f_\theta(\boldsymbol{x}) - y)^2$$

How: differentiate this loss with respect to the parameters (theta) . Update the parameters such that the loss on that sample will decrease.

$$\theta' = \theta - \alpha \nabla_\theta L(\boldsymbol{x}, y, \theta)$$

Adversarial sample



Goal: **increase** the loss of the model on the sample x

$$\boldsymbol{x}' = \boldsymbol{x} + \alpha \nabla_{\boldsymbol{x}} L(\boldsymbol{x}, y, \theta)$$

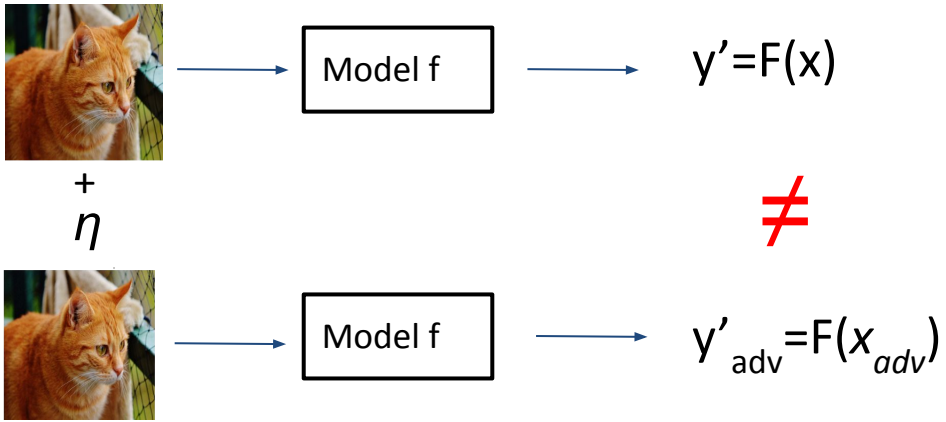
Adversarial sample

Let us consider a sample ***(x, y)*** and a model ***F*** that maps x to a predicted class label ***y'***.

An adversarial sample $\mathbf{x_{adv}}$ for an original sample **x** is defined as:

$x_{adv} = x+\eta$ where η is a **small** perturbation added to the input such that $f(x_{adv}) \neq y$

Attack goal:target or untarget



Adversarial attacks: norm

Given an original sample **x** an adversarial attack aims to find an adversarial example $\mathbf{x_{adv}}$

$x_{adv} = x+\eta$ where η is a **small** perturbation added to the input

with $||x - x_{adv}|| = ||\eta|| < \varepsilon$ ← perturbation bound

- The constraint has the objective of disallowing perturbations which could make *x* unrecognisable (imperceptible perturbation)
  - without altering much the original sample, attacks can perturb a few pixels strongly (L_0), all pixels slightly (L_∞) or a mix of both (L_1 and L_2)

[Kotyan et al 2020 Adversarial Robustness Assessment: Why both L_0 and L_∞ Attacks Are Necessary]

Adversarial attacks: norm

- ε (pertubation bound) is measured with a mathematical norm
  - L_0 : minimize the number of elements (pixels) modified in x_{adv} such that $x \neq x_{adv}$
    - e.g., in sticker added to stop-sign all the background is preserved only a tiny fraction of the environment is modified
  - L_1: minimize the Manhattan distance (sum of the total perturbation values) for each pixel to create the adversarial sample.
    - $L_1=|x^1-x^1_{adv}|+|x^2-x^2_{adv}|+.....+|x^n-x^n_{adv}|$
    - Attack quite uncommon
  - L_2 : minimize Euclidean distance (MSE) for each pixel as upper bound to create adv. sample
    - $L_2=\sqrt{(x^1-x^1_{adv})^2+(x^2-x^2_{adv})^2+.....+(x^n-x^n_{adv})^2}$
    - Commonly attack
  - L_∞ : what is the maximum value/change to *any* of the pixels in the x_{adv} image
    - $l_\infty=max(|x^1 \neq x^1_{adv}|+|x^2 \neq x^2_{adv}|+.....+|x^n \neq x^n_{adv}|)$

Evasion attacks: norm

- Different meaning:
  - L_0: a pixel that changes by 0.0001 is as influential as one that changes by 100, since the metric only account the number of pixels to change
  - L_∞ : the pixel with the maximum change is taken into account.
    - This norm is the commonly used in adversarial attacks

Adversarial attack: norm

Norm: way to measuring the small change in original samples to create adversarial samples (distance from original sample)

[Shafahi et al 2020 Are adversarial examples inevitable?]

Adversarial attack: norm

Way to measuring the small change in original samples to create adversarial samples

	Norm bound?			
Access to gradient?	L_0	L_1	L_2	L_∞
Untarget			Carlini Wagner DeepFool	PGD FGSM BIM
Target	JSMA			

Fast Gradient Sign Method (FGSM)

$+ .007 \times$

$=$

"panda"

57.7% confidence

noise

"gibbon"

99.3% confidence

[Goodfellow et al 2015 Explaining and Harnessing Adversarial Examples]

Fast Gradient Sign Method (FGSM)

Goal: maximize **the loss** for the constructed adversarial sample

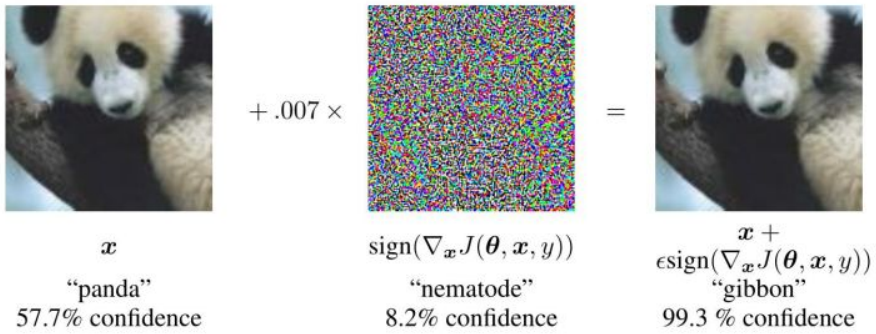
$X_{adv} = X + \eta$ with respect to the true label y

$$max\ L\ (F(x+\eta),\ y)$$

instead of optimizing the parameters to decrease loss (with images constant) we optimize the image pixels to increase loss (constant parameters)

with $||\eta||_\infty < \varepsilon$

- Single step: perturbation is computed once by following the gradient of the function F
- Constraint optimization: the attack applies a bound equal to L_∞



[Goodfellow et al 2015 Explaining and Harnessing Adversarial Examples]

Fast Gradient Sign Method (FGSM)

Adversarial example: $X_{adv} = X + \eta$

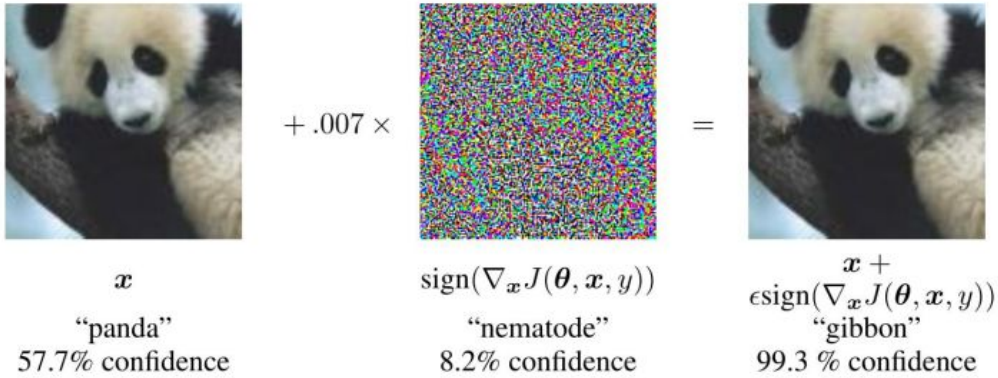
Perturbation

$\eta = \varepsilon \, \text{sign}(\nabla_x L(x, y, W))$

small scalar (e.g., 0.1, 0.007)

Gradient loss function with respect to the image x for the true label y

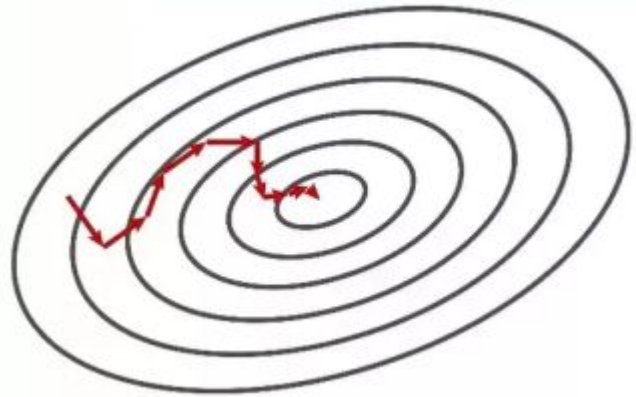
distort pixels in the opposite direction of the loss with respect to the target class y



[Goodfellow et al 2015 Explaining and Harnessing Adversarial Examples]

Iterative multistep algorithms

- FGSM is a **single-step gradient update** : limits the power of attack
- More powerful: iterative object optimization running steepest descent for multiple iterations
- Two extension of FGSM:
  - **BIM**
  - **PGD**



Basic Iterative Method (BIM)

- BIM is an extension of FGSM
  - Add noise in multiple iterations with a step size α
  - After each iteration the result is clipped to ensure that the perturbation is within ε-neighbourhood of original sample x (maximum perturbation for each pixel)

$$x^{0}_{adv} = x$$

$$x^{t}_{adv} = Clip_{x,\varepsilon}\{ x^{t-1}_{adv} + \alpha \cdot sign(\nabla_x L(f(x^{t-1}_{adv}), y))\}$$

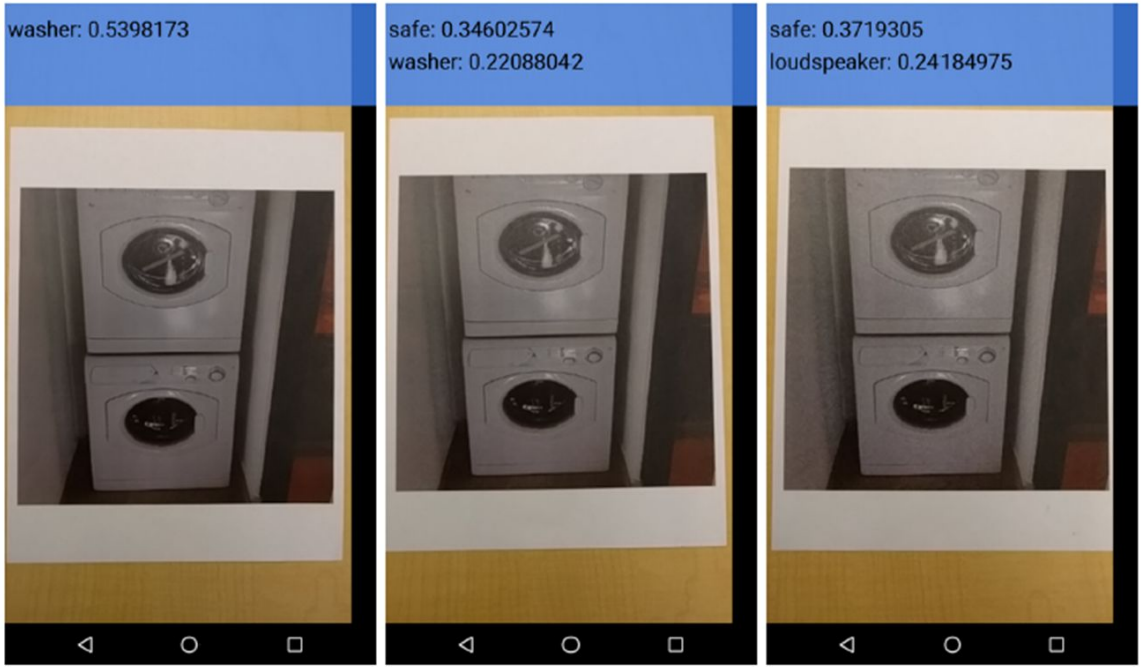
a = step size of iteration

t = number of iterations

$$Clip_{x,\varepsilon}\{x_{adv}\} = min\{ 255, x+\varepsilon, max\{0, x-\varepsilon, x_{adv}\}\}$$

[Kurakin et al 2017 Adversarial Sample in Physical World]

Basic Iterative Method (BIM)



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

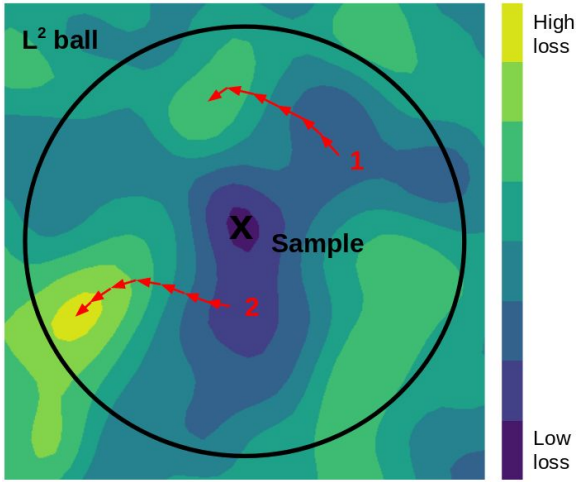
[Kurakin et al 2017 Adversarial Sample in Physical World]

Project Gradient Descent (PGD)

PGD is an extension of BIM , where after each step of perturbation, the adversarial example is projected back onto the ϵ-ball of x (decided by norm) using a projection function Π

$$x^{t}_{adv} = \Pi_{\epsilon}\left(x^{t-1} + \alpha \cdot \text{sign}\left(\nabla_{x} L(f(x^{t-1}), y)\right)\right)$$

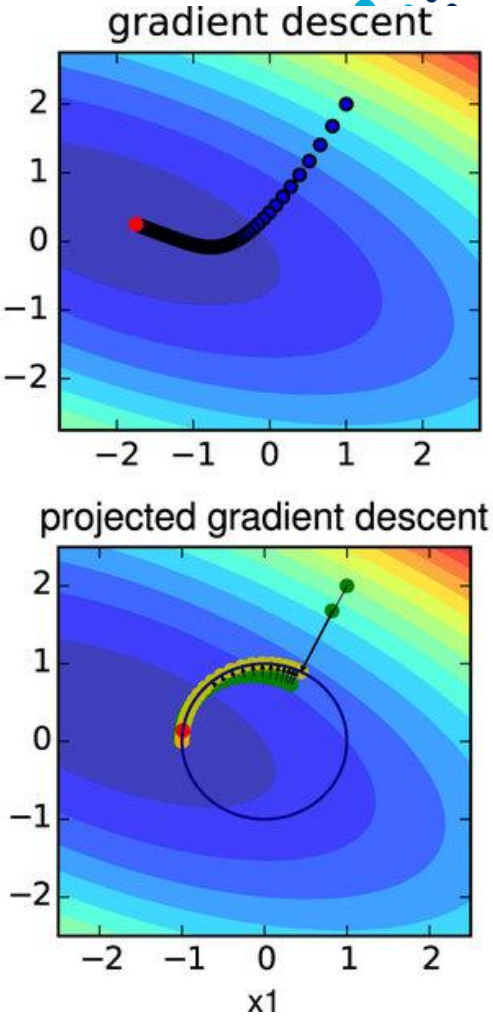
α=gradient step size



[Madry et al 2019 Towards Deep Learning Models Resistant to Adversarial Attacks]

Project Gradient Descent (PGD)

- **Gradient Descent**:is a standard way to solve unconstrained optimization problem
  - $\min_{x \in R} f(x) \rightarrow$ any x in R can be a solution
- **Project Gradient Descent:** is a standard way to solve constrained optimization problem.
  - $\min_{x \in Q} f(x) \rightarrow$ not any x in R can be a solution but inside the set Q
  - given a point x, PGD try to find a point in Q which is closest to x



[Madry et al 2019 Towards Deep Learning Models Resistant to Adversarial Attacks]

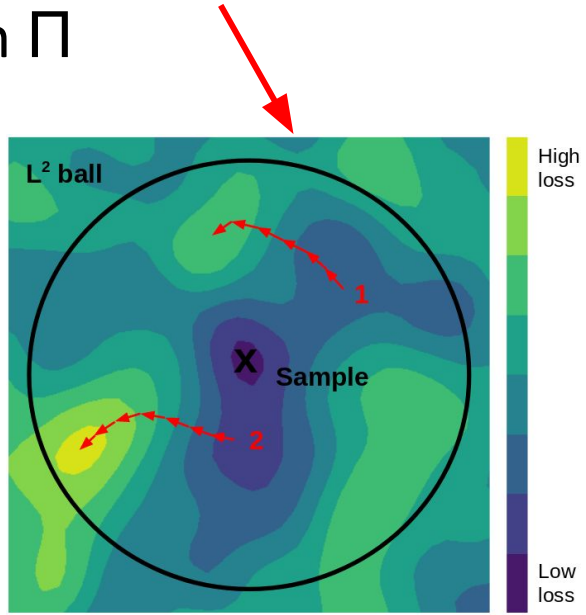
Project Gradient Descent (PGD)

PGD is an extension of BIM , where after each step of perturbation, the adversarial example is **projected back onto the ϵ-ball of x** (decided by L norm) using a projection function Π

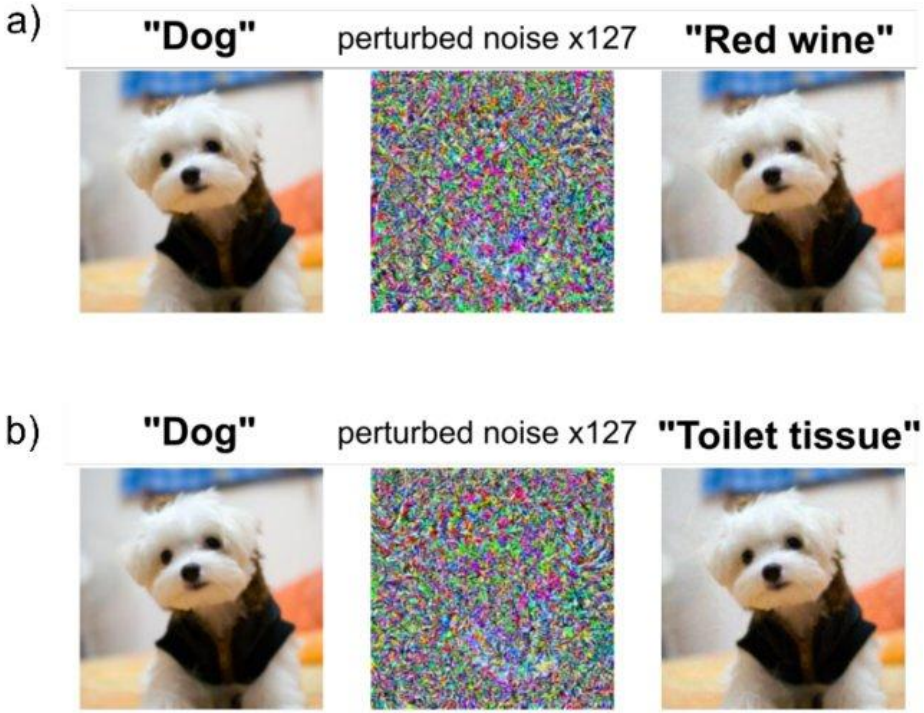
$$x^{t}_{adv} = \Pi_{\epsilon}\left(x^{t-1} + \alpha \cdot sign(\nabla_x L(h(x^{t-1}), y))\right)$$

Different from BIM, PGD uses random initialization for *x* for each iteration by adding random noise from a uniform distribution with values in the range $(-\epsilon, \epsilon)$

[Madry et al 2019 Towards Deep Learning Models Resistant to Adversarial Attacks]



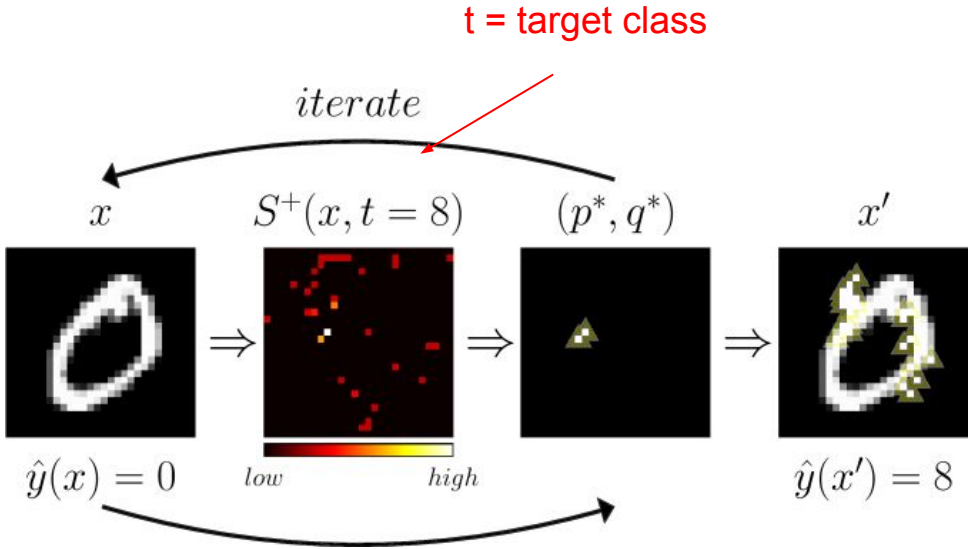
Project Gradient Descent (PGD)



[Machiraju et al 2021. Bio-inspired Robustness: A Review. 🆓]

Jacobian-based Saliency Map Attack (JSMA)

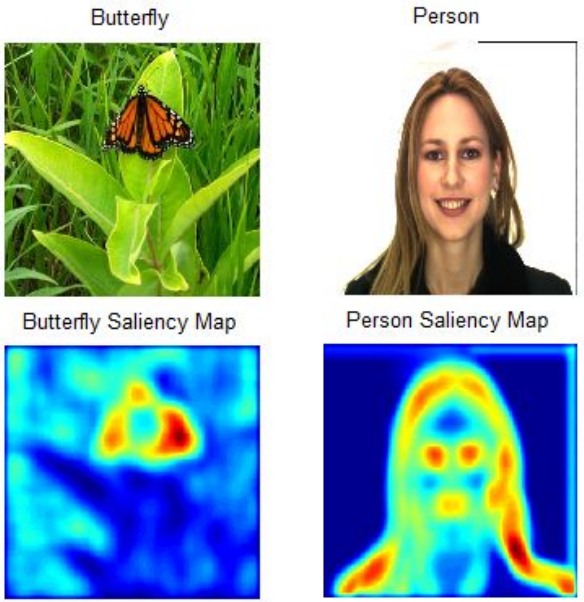
- JSMA is a target attack -> goal to create an adversarial sample that is misclassified in a specified class
- JSMA different from FGSM that alter all pixel uses L_0 norm that greedily modifies pairs of pixels at a time.
  - using a saliency map, which shows an impact each pixel has on the classification result.
  - A large value means, that changing this pixel will have a significant impact on the outcome of the classification.



[Wiyatno *et al 2018* Maximal Jacobian-based Saliency Map Attack]

JSMA: saliency map

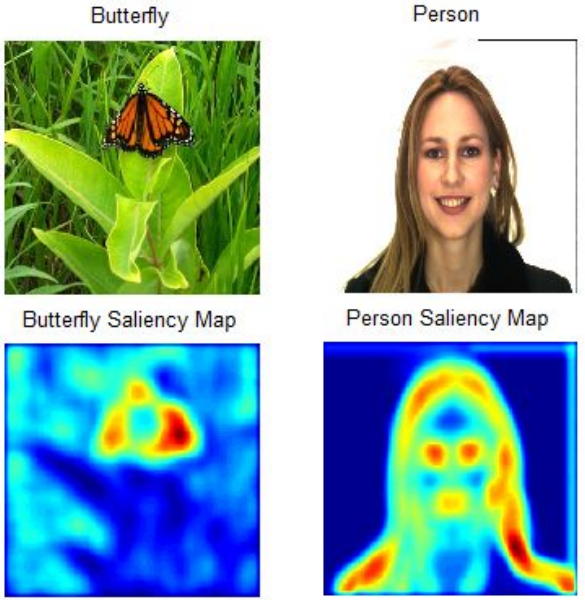
- Saliency Map: visual explanation of the predictions of a classifier
  - gradient of each input feature x^i (e.g., each pixel) to the class score (i.e., how influential to predict a particular class c)
  - $y'(x) = \arg\max_c f(x)(c)$) where f(x) is the softmax probabilities vector predicted by the model.



[Christopher Kanan (2022). Image Descriptors / Features and Saliency Maps]

JSMA: adversarial saliency map

- Adversarial saliency map: visual explanation of which input features should perturb in order to predict the adversarial sample as a target class.
- Adversary want to misclassify a sample x that is assigned to a target *t*
  - $f_t(x)$ must be increased while the confidence for $f_j(x)$ for all other classes $j \neq t$ decrease until $t=\arg\max f(x)$



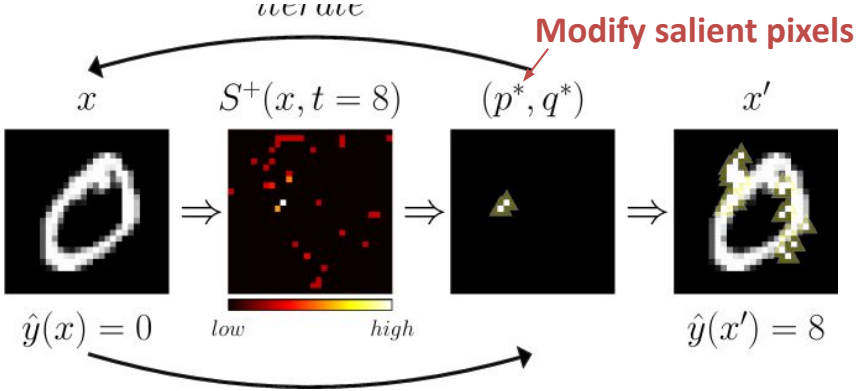
[Christopher Kanan (2022). Image Descriptors / Features and Saliency Maps]

JSMA

S (x,t) = measure how a feature(x^i) is positively correlates with t, while also negatively correlates with all other classes j ≠ t

rejects input with a negative derivative or an overall positive derivative on other classes.

$$S(x,t)[i] = \begin{cases} 0, & \text{if } \frac{\partial F_t}{\partial x_i}(x) < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j}{\partial x_i}(x) > 0 \\ \frac{\partial F_t}{\partial x_i}(x) \left| \sum_{j \neq t} \frac{\partial F_j}{\partial x_i}(x) \right|, & \text{otherwise} \end{cases}$$



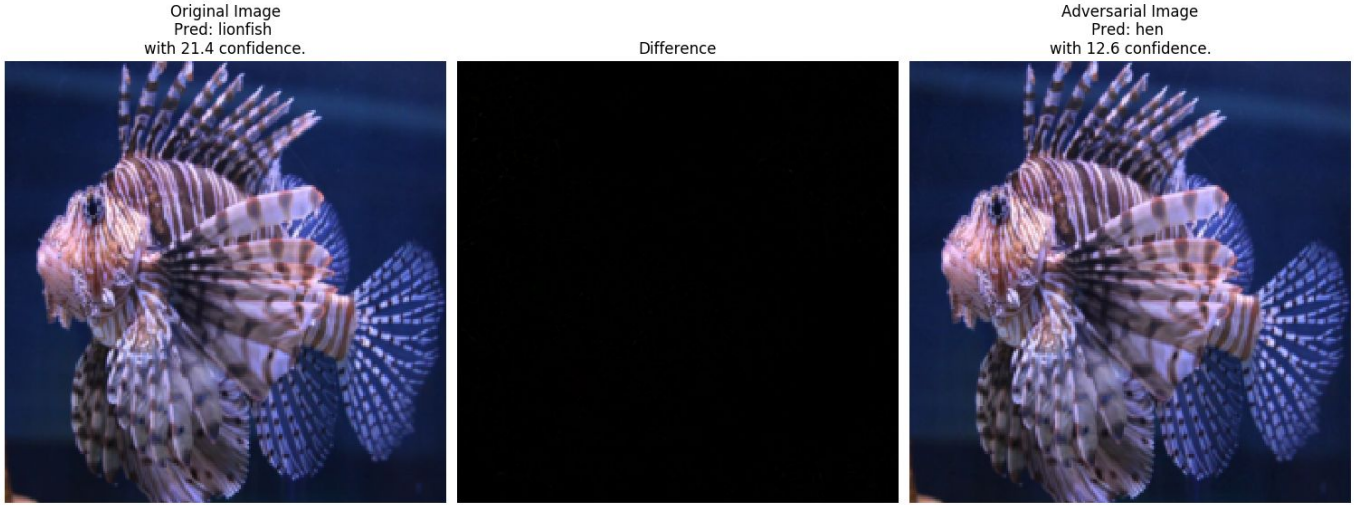
How much $F_t(x)$ will increase and $F_j(x)$ will decrease given a modification of the input feature i

DeepFool

Compute a minimal norm adversarial perturbation for a given image in iterative manner, to find the decision boundary closest to the normal sample

Perform steps by linearly approximate the decision function, according to the l_2 norm

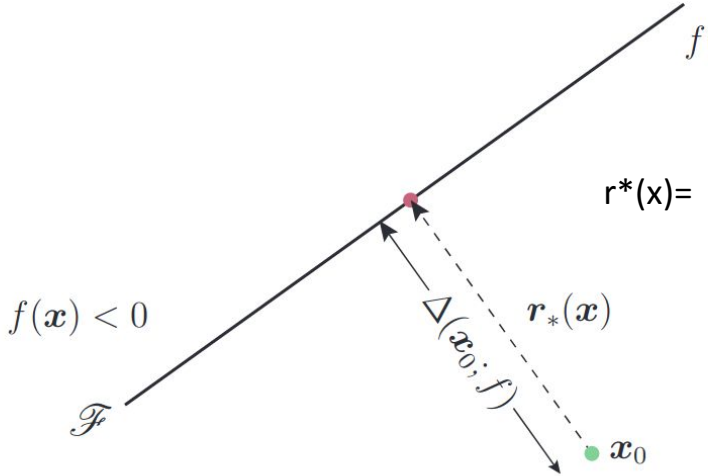
Perturbation is small since gradient is orthogonal to the boundary



[Moosavi-Dezfooli *et al 2016* DeepFool: a simple and accurate method to fool deep neural networks]

DeepFool for binary classifier

- Using a linear binary classifier, that the **robustness** of a model (f) for an input x_0 is equal to the distance of x_0 to the hyperparameter plane (which separates the 2 classes)
- The minimal perturbation r*(x) to change the classifier's decision must project the input image of x_0 orthogonal to the hyperplane of classification



r*(x)=

$$r_*(x) = -\frac{f(x_0)}{||w||_2^2} * w$$

Output

sign inverted so the loss of the classifier *f* is increased

L_2 norm

Gradient

[Moosavi-Dezfooli *et al 2016* DeepFool: a simple and accurate method to fool deep neural networks]

DeepFool

- **!!!Misclassification is not guarantee since models are not linear in nature**
  - To alleviate this issue, the algorithm works iteratively and adds the previous perturbation to the next perturbation, which is performed until the label changes or max iterations are reached

Algorithm 1 DeepFool for binary classifiers

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
  $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$,
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
  $i \leftarrow i + 1$.
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.

1. Start and continue loop while the true label and the label of the adversarially perturbed image is the same.

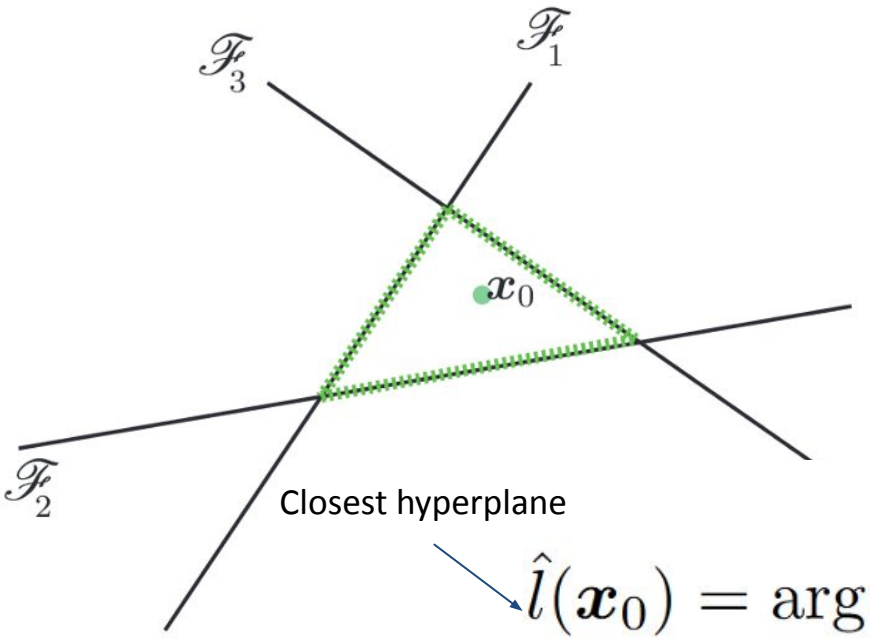
2. Calculate the projection of the input onto the closest hyperplane (minimal perturbation)

3. Add that perturbation to the image and test

[Moosavi-Dezfooli *et al 2016* DeepFool: a simple and accurate method to fool deep neural networks]

DeepFool for multiclass

Multiple binary classifiers



- The minimum perturbation needed would be from the closest hyperplane to x_0.
- Given there are multiple classes, the loss and backpropagation would need to be computed for each class label allowed by the function.
- When finding the minimum perturbation, the difference must be taken between the computed values for each label and the computed values for the label of the original prediction.

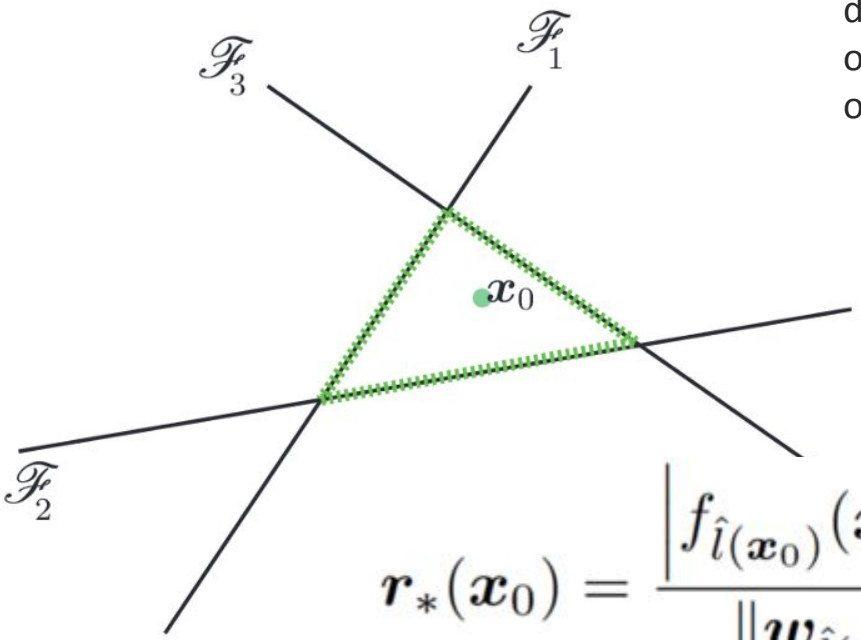
$$\hat{l}(\boldsymbol{x}_0) = \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{\left| f_k(\boldsymbol{x}_0) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_0) \right|}{\left\| \boldsymbol{w}_k - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)} \right\|_2}$$

most probability after the true class

true class

DeepFool for multiclass

Multiple binary classifiers



- The minimum perturbation $r_*(x_0)$ is the vector that project differences of the classifier output and the gradients of the outputs, as well as taking the absolute value of the model output.

$$r_*(\boldsymbol{x}_0) = \frac{\left| f_{\hat{l}(\boldsymbol{x}_0)}(\boldsymbol{x}_0) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_0) \right|}{\|\boldsymbol{w}_{\hat{l}(\boldsymbol{x}_0)} - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)}\|_2^2} (\boldsymbol{w}_{\hat{l}(\boldsymbol{x}_0)} - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)})$$

[Moosavi-Dezfooli *et al 2016* DeepFool: a simple and accurate method to fool deep neural networks]

DeepFool for multiclass

Algorithm 2 DeepFool: multi-class case

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
  for $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
    $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  end for
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

1. Store the difference between the original gradients and the gradients of each of classes (w_k) and the difference between labels (f_k)

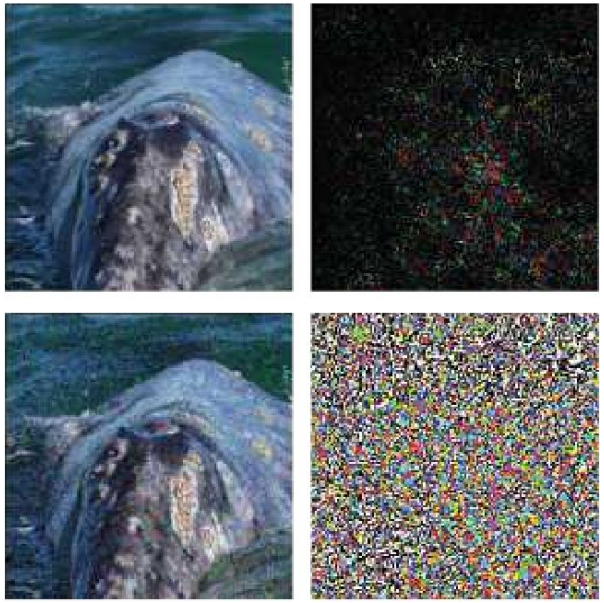
2. Calculate the minimal vector that projects c on the closes hyperplane

3. Total perturbation is the sum over all calculated perturbation

[Moosavi-Dezfooli *et al 2016* DeepFool: a simple and accurate method to fool deep neural networks]

DeepFool vs FGSM

True label: whale



Predict: turtle
DeepFool

Predict: turtle
FGSM

Carlini Wagner (CW) attack

- Goal: find a small change η that make an image x misclassified but so that the result is still a valid image/example
  - All the other attacks were using constraints on perturbation (ϵ), here the score is used as a penalizer, modulated by C.
- How close we are getting to being classified as t

$$f=[1-C(x+\eta)_t]$$

$C(x+\eta)_t$ =t is satisfied if $f(x+\eta) \leq 0$ is satisfied

Probability of x+η to be classified as t.
If the probability is low value of f is closer to 1 value of f is closer to 1 whereas when it is classified as t, f is equal to 0.

[Carlini et al 2017 Towards Evaluating the Robustness of Neural Networks]

Carlini Wagner (CW) attack

- How close we are getting to being classified as t

$$f(x') = \max(\max\{Z(x')i\} - Z(x')t, -k)$$
$$\text{with } i \neq t$$

lower limit of loss, at last the value in -k will always hold

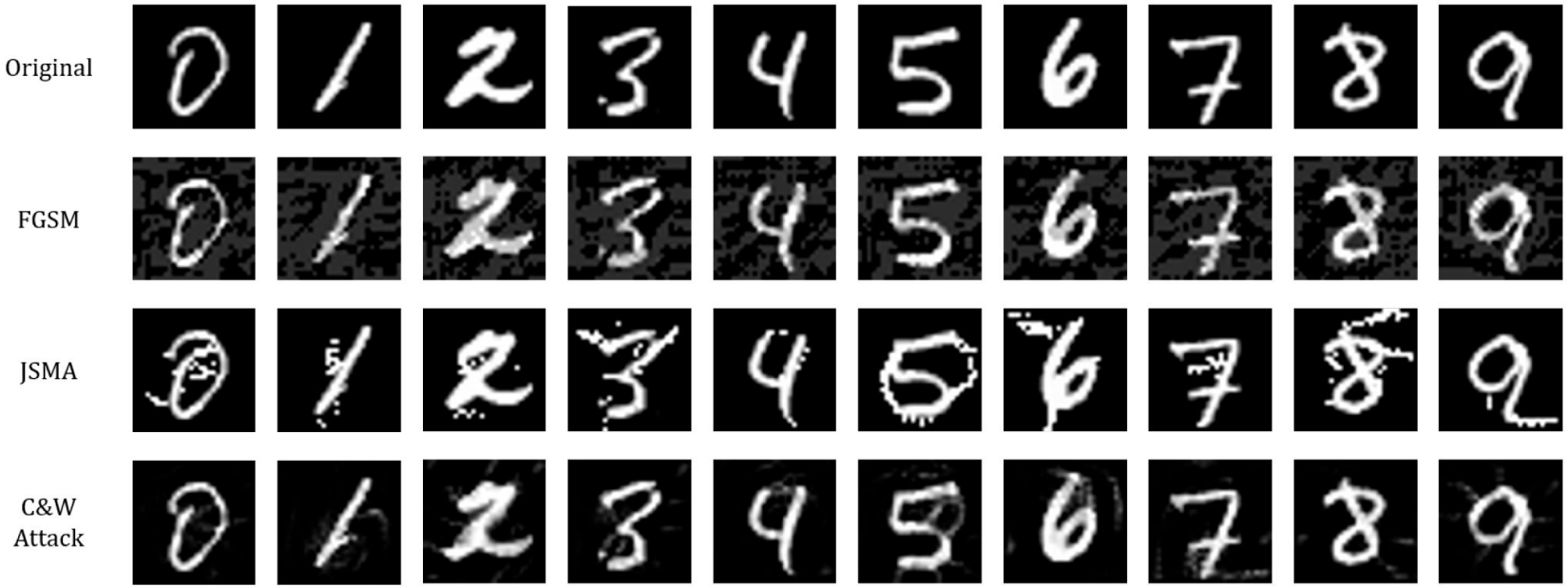
highest probabilities for non target classes

vector of probabilities for target class

difference between “what the model thinks the current image most probably is” and “what we want it to think/ misclassified target”. So when the model thinks that this image is what we want it to think, this value is negative (the probability of target class is higher than any of the non target classes)

[Carlini et al 2017 Towards Evaluating the Robustness of Neural Networks]

Comparison



[Sanglee et al 2020 On the Effectiveness of Adversarial Training in Defending against Adversarial Example Attacks for Image Classification]

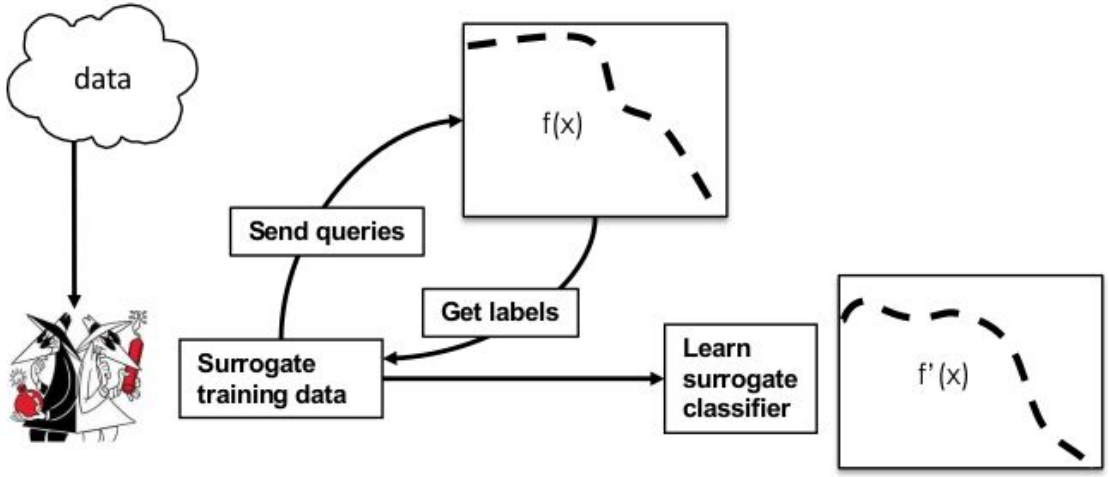
Evasion: black box

KDDE

- **Transferability** based black-box attack :
  - train a substitute model, and craft adversarial examples against the substitute, and transfer them to a victim model
- It is very likely that an adversarial example of one network can fool another network
- Transferability depends on the type of attack
  - e.g. examples built with FGSM are highly transferable

Evasion: Black box

Training a local model to substitute for the target DNN, using inputs synthetically generated by an adversary and labeled by the target DNN.



[Papernot et al. 2017 Practical Black-Box Attacks against Machine Learning]

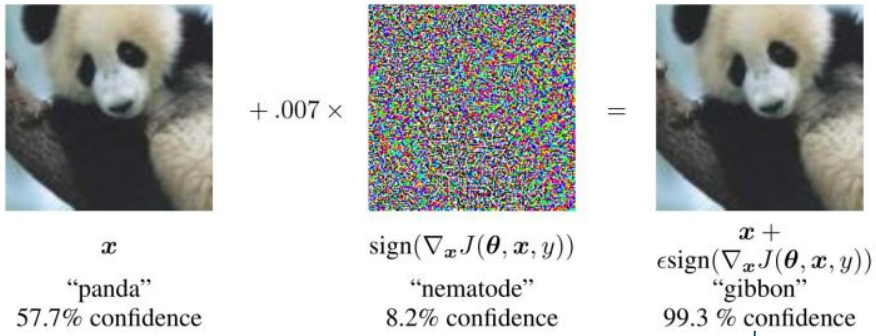
Evasion: black box

KDDE

- train a substitute network based on the input/output
- pairs of the target network
- build adversarial examples for the substitute network
- attack the target network with the examples built for the substitute network
- due to transferability the attack is very likely to succeed

Adversarial transferability

White box scenario:
access to the model



Transferability captures the ability of an attack against a machine-learning model to be effective against a different, potentially unknown, model that was
learned for the same problem (this was observed in the context of deep learning, as well as for
other learning paradigms)

Adversarial sample

Target model

gibbon

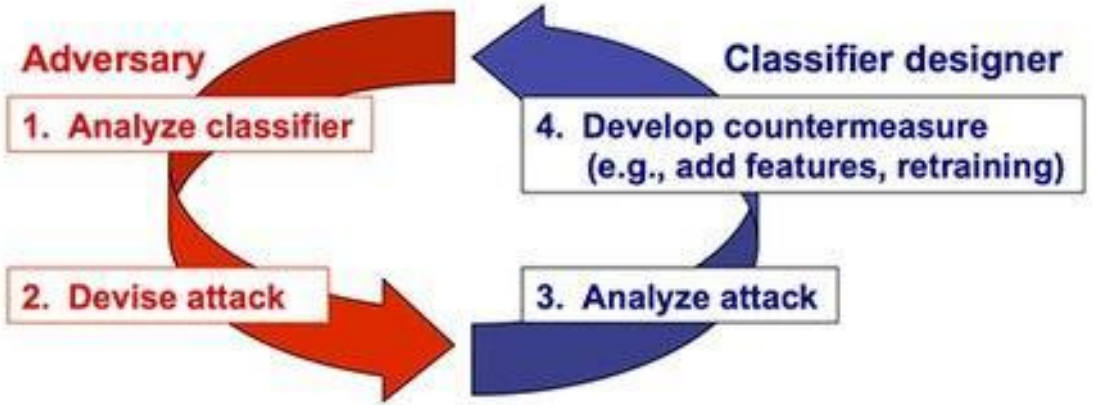
[Demontis et al 2019 Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks]
[Papernot et al 2016 Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples]

Defense Strategy

- Reactive defense
- Proactive defense

Reactive defense

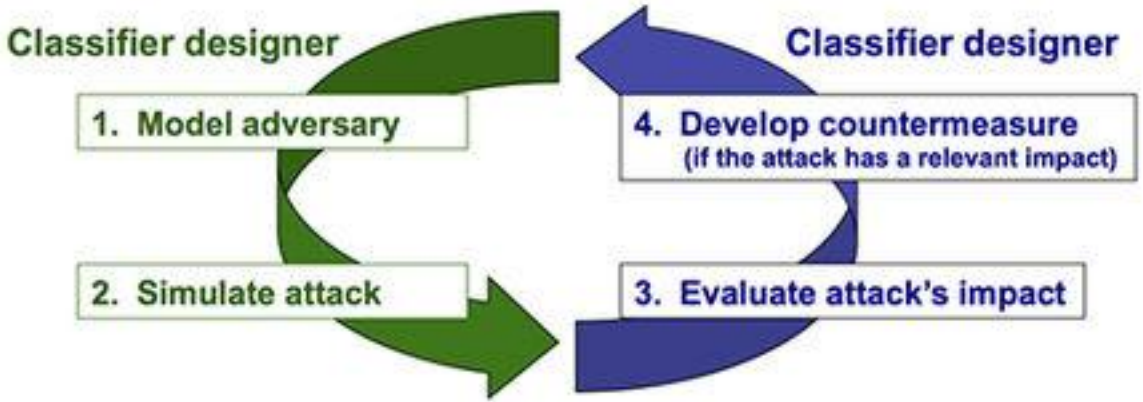
- Timely detection of novel attack
- Frequent classifier retraining
- Verification of consistency of classifier against training data and ground-truth label



Biggio et al. 2014 Security Evaluation of Pattern Classifiers under Attack

Proactive defense: simulating attack

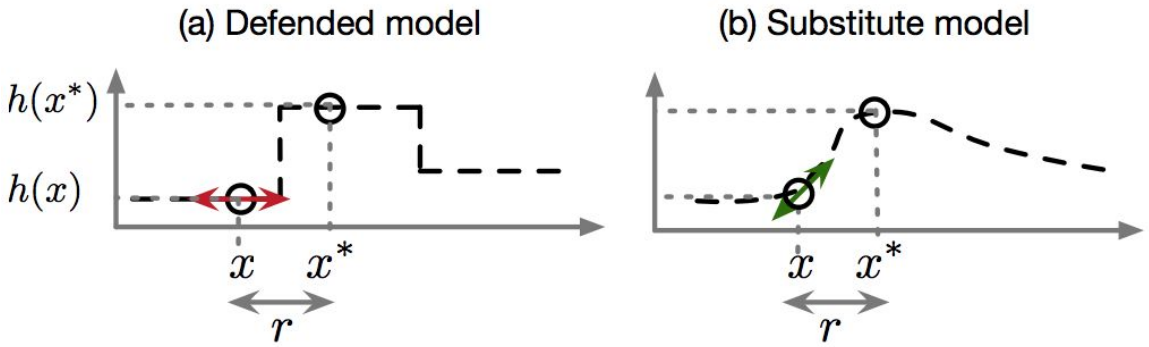
- Gradient hiding
- Adversarial training



Biggio et al. 2014 Security Evaluation of Pattern Classifiers under Attack

Proactive defense: gradient hiding

- **Gradient masking:**
  - reduce the sensitivity of models to small changes by finding adversarial direction using a substitute
  - smooths the model's decision surface in adversarial directions
  - defensive distillation



Papernot al. 2016 Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples
Papernot et al 2016 Towards the Science of Security and Privacy in Machine Learning

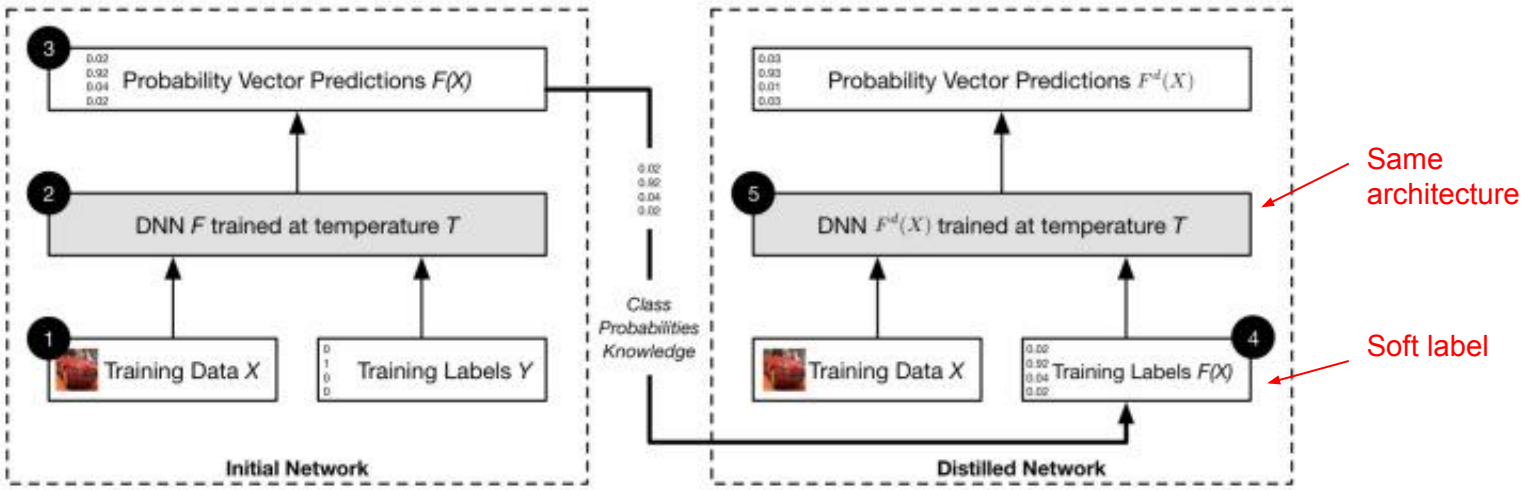
Distillation vs Defensive distillation

- Distillation was first introduced by Hinton et al. in, where the goal was for a small model to mimic a large, computationally expensive model.
  - Teacher and student
- Defensive distillation has a different goal:
  - two model that are the same but one model is trained to predict the probabilities output by another model that was trained earlier
  - smooths the model's decision surface in adversarial directions exploited by the adversary.
  - The second *distilled* model is more robust to attacks such as the fast gradient sign method or the Jacobian-based saliency map approach

[Papernot et al 2016- Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks]
[Hinton at al 2014 Distilling the Knowledge in a Neural Network]

Defensive Distillation

Training using the probability distribution as the target, not just the argmax class label



[Papernot et al 2016- Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks]

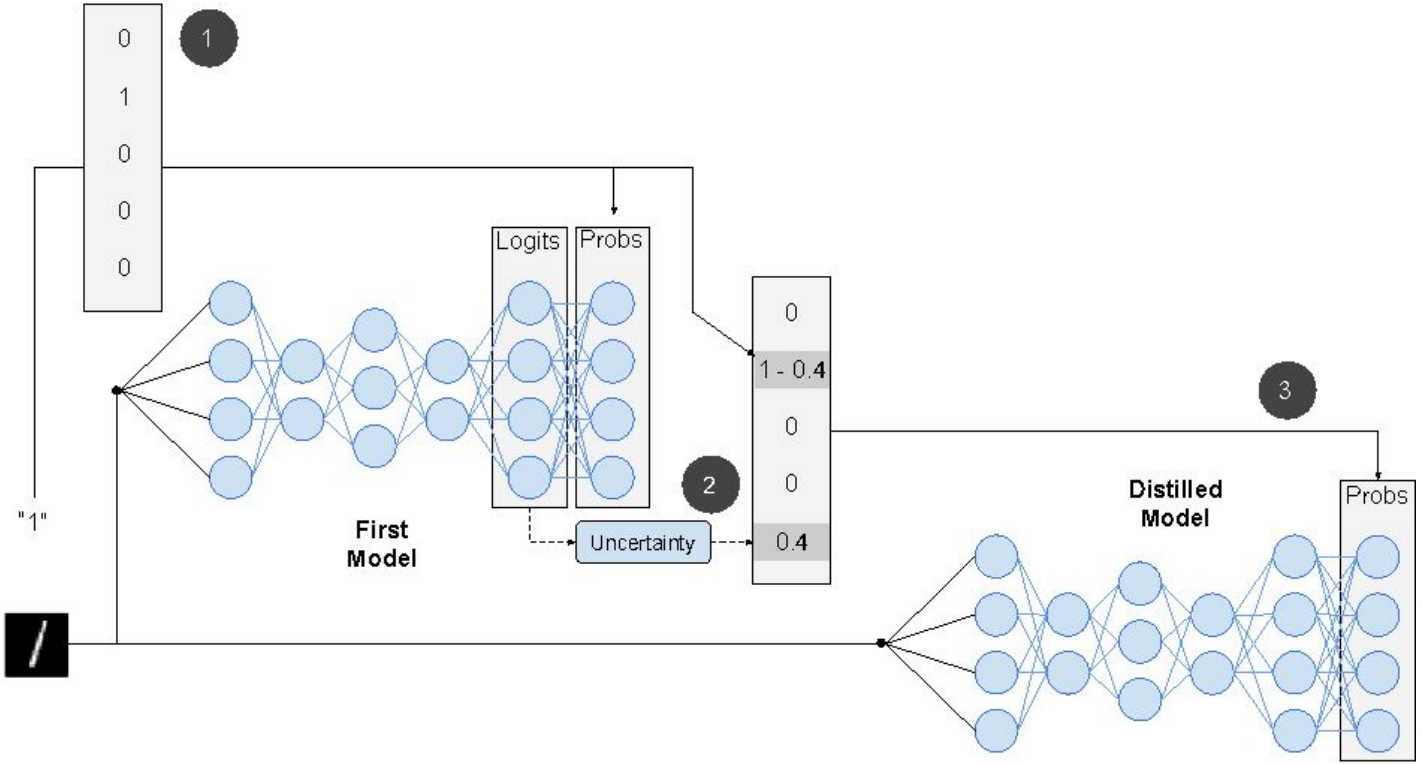
Defensive distillation is not robust

KDDE

- Carlini et al. (2016) prove that with a slight modification to the standard adversarial attacks, distilled networks can be attacked
- Defensive distillation work on attacks that use gradient to approximate each pixel's importance to modify to create adversarial sample (e.g., JSMA)
- Carlini et al. propose attacks using the logits to create adversarial sample (not softmax)
  - instead of taking the gradient of the inputs to the softmax, they take the gradient of the actual output of the network
  - For T = 100, attacks had 96.4% success rate while only changing on average 36.4 pixels; works on all T from 1 to 100

[Carlini et al 2016- Defensive Distillation is not robust to Adversarial Examples]

Extending defensive distillation



[Papernot et al 2017- Extending Defensive Distillation]

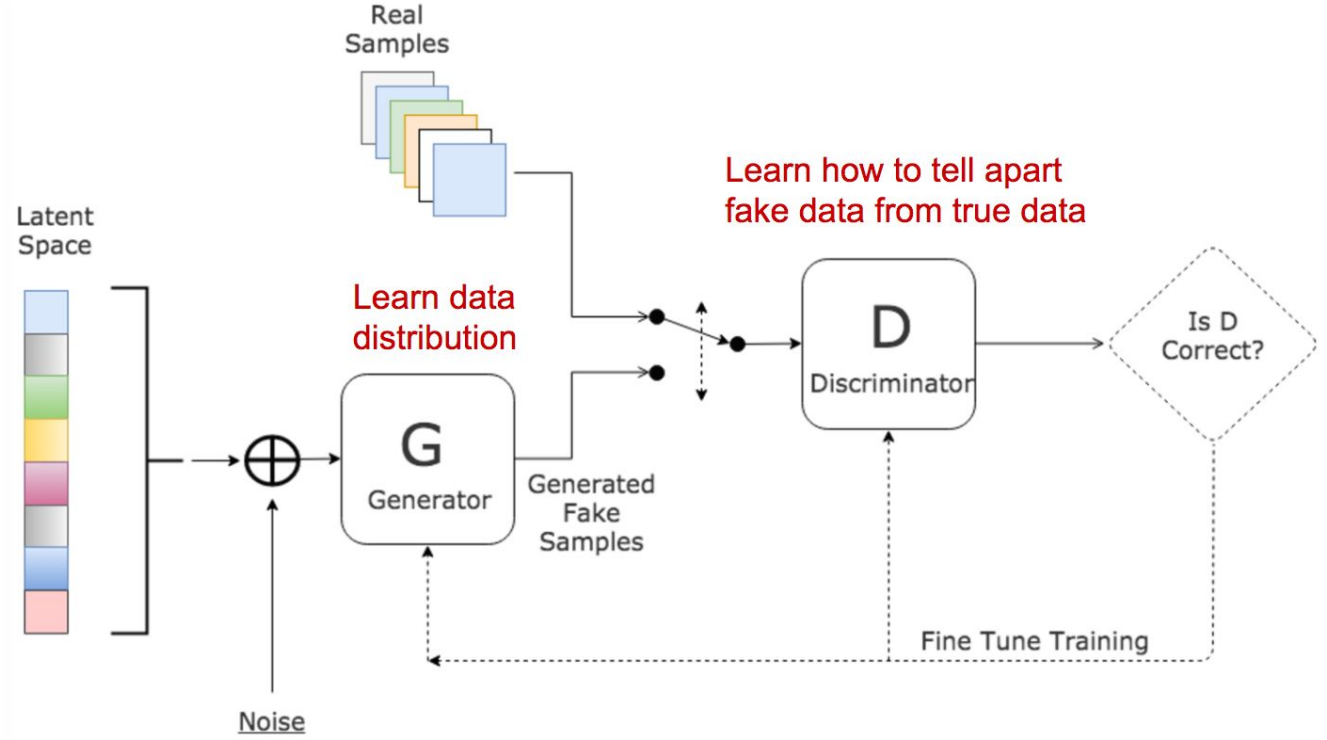
Proactive defense: adversarial training

- Introduce adversarial samples in training set to improve the robustness of the target model
  - [Szegedy et al 2013 - Intriguing properties of neural networks: inject adversarial samples and modified its labels]
  - Malik
- Punish misclassified adversarial image
  - [Huang et al 2016 - Learning with a Strong Adversary]

Adversarial training: data augmentation

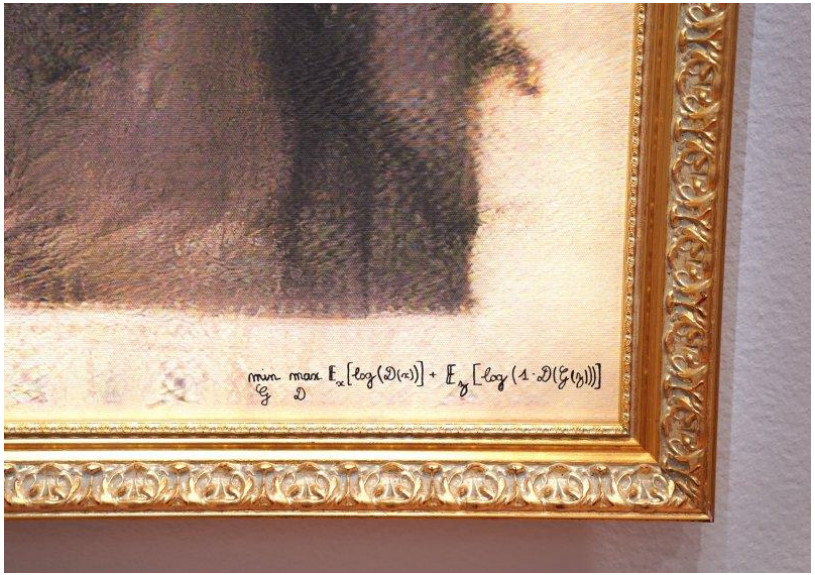
- GAN (generative adversarial network)
- Adversarial autoencoder

GAN



[Goodfellow et al 2014- Generative Adversarial Networks]

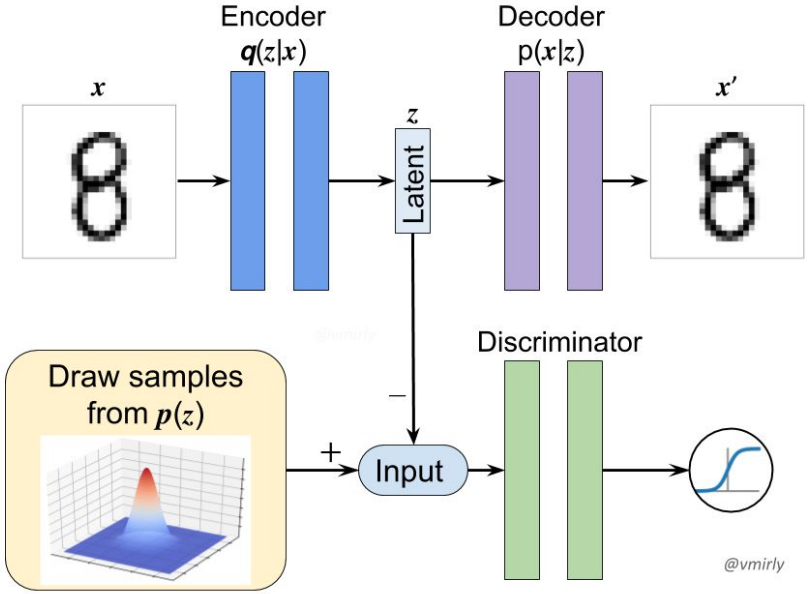
GAN



Painting above was made by a GAN and sold for **$432 thousand**

https://time.com/5435683/artificial-intelligence-painting-christies/

Adversarial autoencoder



[Makhzani et al 2016- Adversarial autoencoders]

Adversarial training: feature robustness

Training set

frog

Restrict to features of robust model

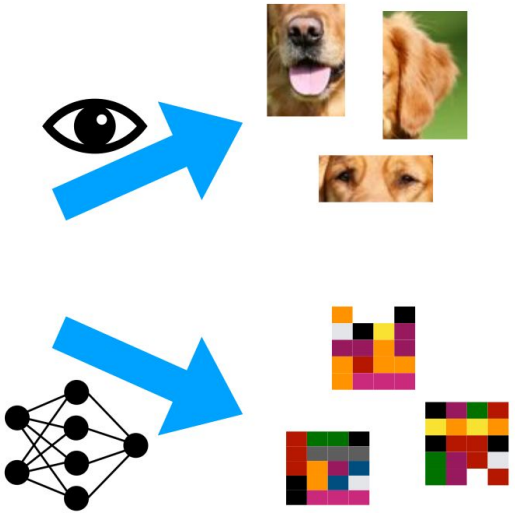
New training set

“Robustified” frog

[Ilyas et al 2019- Adversarial Examples Are Not Bugs, They Are Features]

Adversarial training: feature robustness

- Goal: robust ML as humans classification
- Classifiers tend to use any available signal to do so, even those that look incomprehensible to humans to generalize
- Adversarial examples can be attributed to the presence of non-robust features:
  - highly predictive
  - incomprehensible to humans.



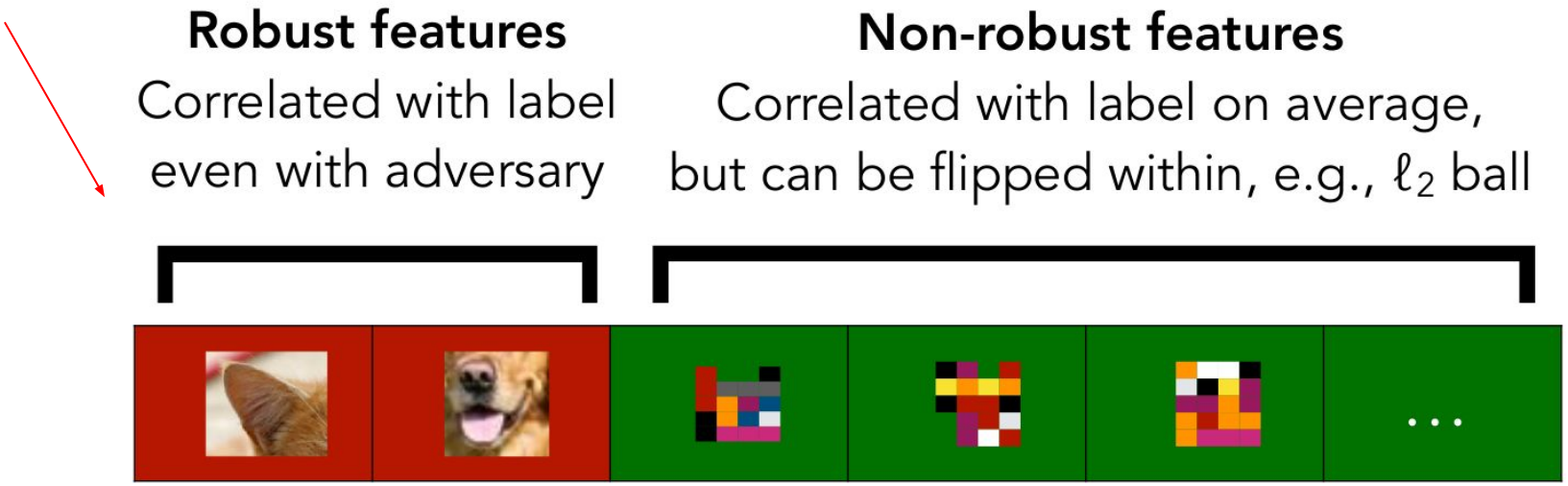
[Ilyas et al 2019- Adversarial Examples Are Not Bugs, They Are Features]

Adversarial training: feature robustness

Standard training: use all of features, maximize accuracy

Adversarial training: use only single robust features (at the expense of accuracy)

Under adversarial perturbation



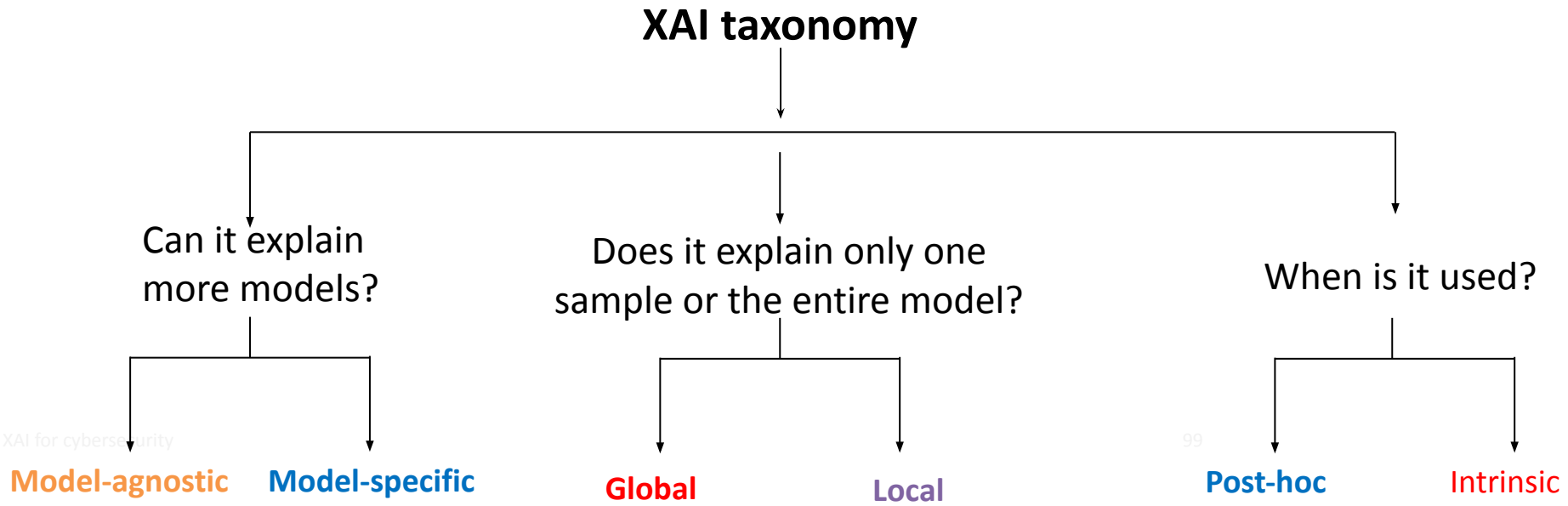
[Ilyas et al 2019- Adversarial Examples Are Not Bugs, They Are Features]

XAI for feature robustness

KDDE

- AI-based systems (e.g., Deep Learning-based systems) are accurate, but they usually learn **black-box models**
- Unjustifiable and misleading predictions may make systems **vulnerable to attacks**
  - by leading to unsecured critical systems
- Explainable and interpretative results make AI solutions more **robust and trustworthy**
  - **Transparency** of model decisions is mandatory to:
    - provide justifiable decision making
    - produce accurate explanations of decision models' behavior
    - help to design effective countermeasures

XAI taxonomy

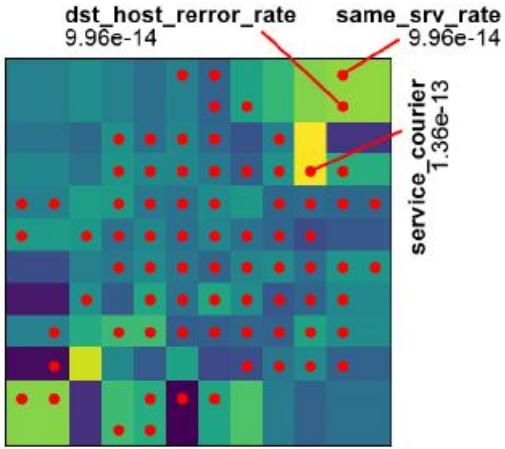


Explanation

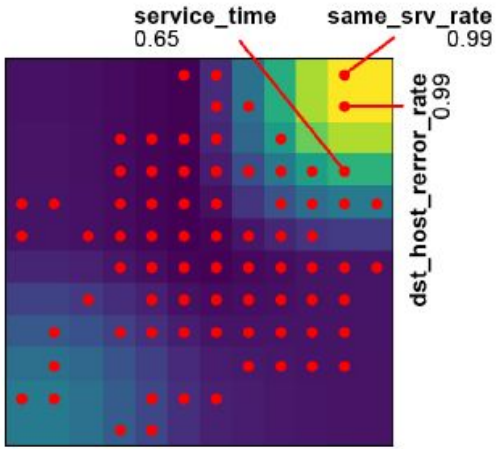
Selvaraju et al., *Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,* IEEE International Conference on Computer Vision (ICCV), 2017

Explanation of network flow attacks

NSL-KDD dataset



Top-1 Grad-CAM
(attack)



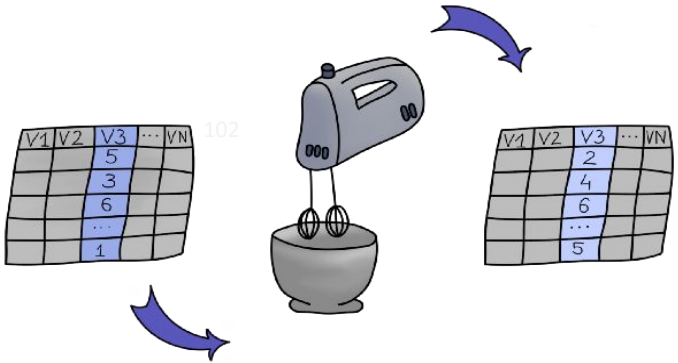
Top-2 Grad-CAM
(attack)

Caforio et al., *Leveraging Grad-CAM to Improve the Accuracy of Network Intrusion Detection Systems*. DS 2021, 2021

Dalex

(mo**D**el **A**gnostic **L**anguage for **E**xploration and e**X**planation)

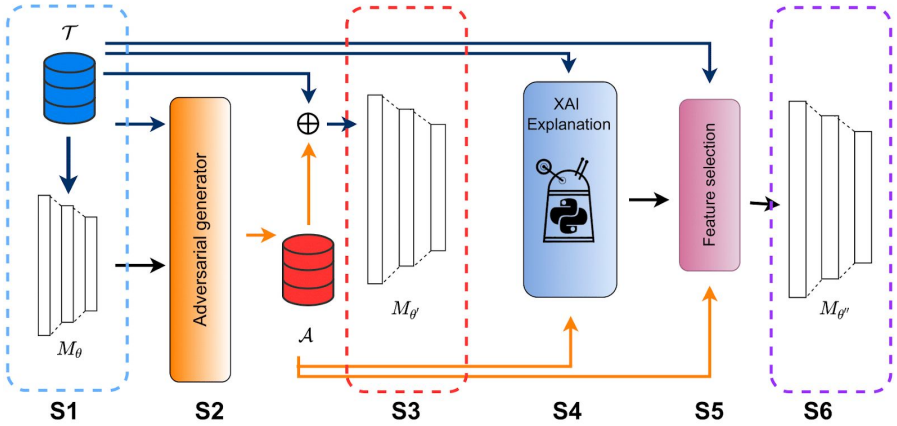
- **Model agnostic** and **post-hoc** technique
- To understand both the **global** and **local** structure of black box models
- To explain the behavior of models by measuring the global relevance of features on decisions
  - It uses permutation-based feature-importance
    - It permutes the value of each feature
    - It computes a loss function before and after the permutation
    - If the loss increases then the feature is important



Przemyslaw B. *DALEX: Explainers for complex predictive models in R,* Journal of Machine Learning, 2018

Robust feature selection with XAI

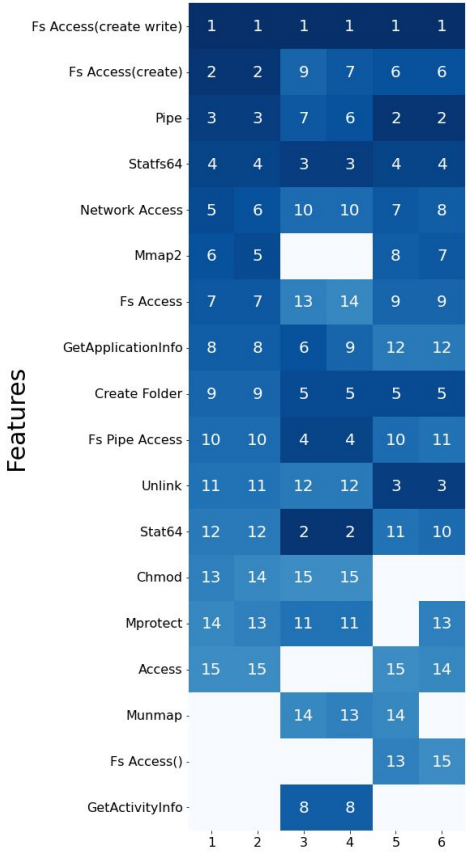
- Adversarial training to learn robust deep neural models
- Post-hoc global explanations with **DALEX**
  - To perform feature selection by extracting the top-k features ranked by DALEX on the training set



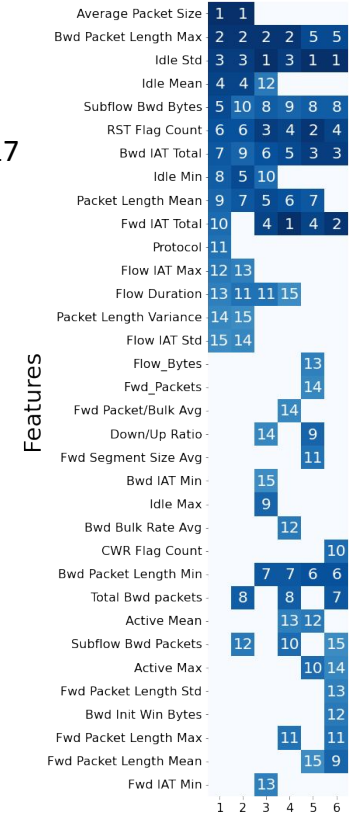
Al- Essa et al, *XAI to explore robustness of features in adversarial training for cybersecurity* . Accepted to 26th International Symposium on methodologies for Intelligent System (ISMIS) 2022

Robust feature selection with DALEX

MalDroid20



CICIDS17



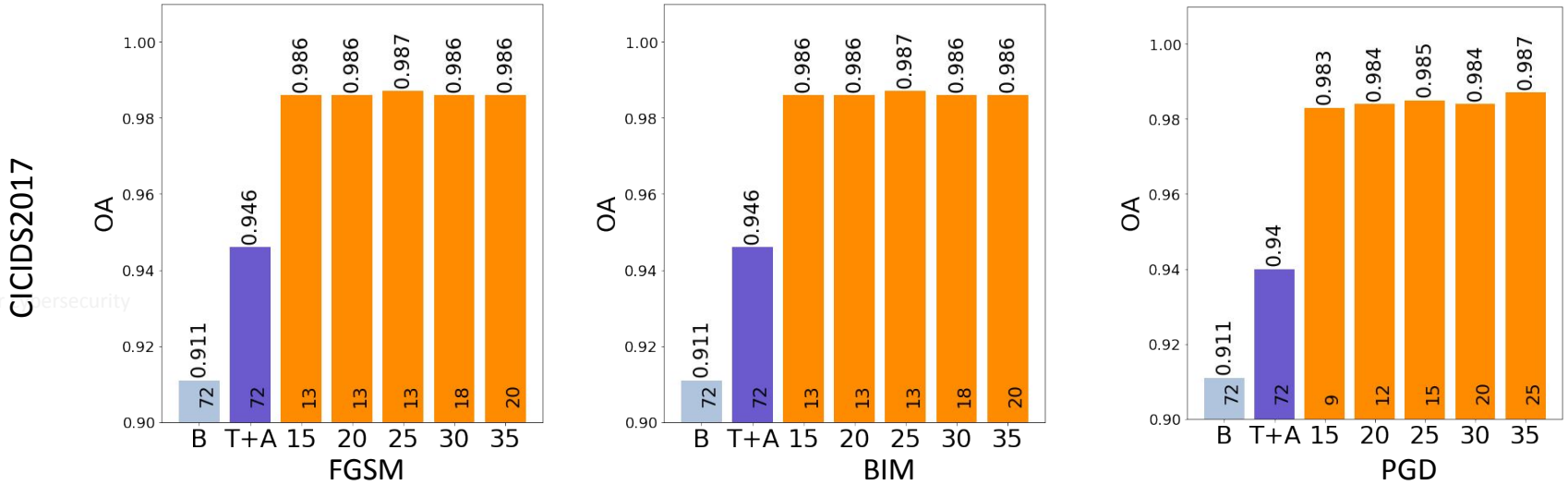
Legend

(1) T+A (FGSM) explains T
(2) T+A (FGSM) explains A
(3) T+A (BIM) explains T
(4) T+A (BIM) explains A
(5) T+A (PGD) explains T
(6) T+A (PGD) explains A

Al- Essa et al, *XAI to explore robustness of features in adversarial training for cybersecurity* . Accepted to 26th International Symposium on methodologies for Intelligent System (ISMIS) 2022

Robust feature selection with DALEX

- **B**: which discards both adversarial training and XAI-based feature selection
- **T+A**: which discards XAI-based feature selection
- **T+A+XAIFS (n)**: which implements both adversarial training and XAI-based feature selection (with n number of feature selected)

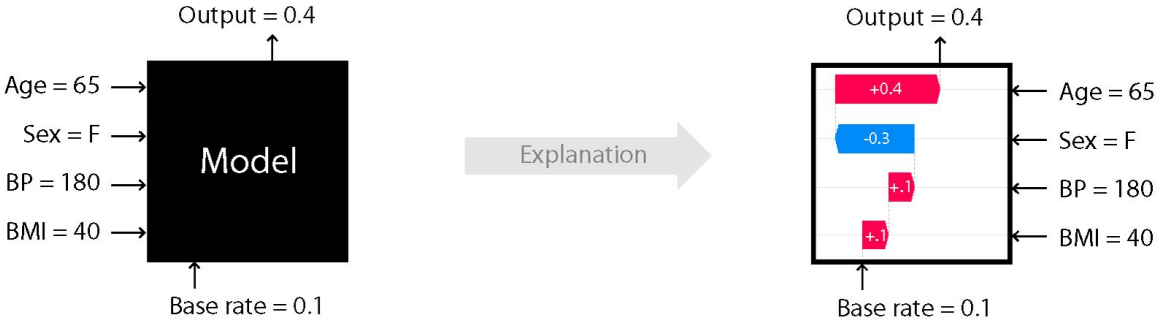


Al- Essa et al, *XAI to explore robustness of features in adversarial training for cybersecurity* . Accepted to 26th International Symposium on methodologies for Intelligent System (ISMIS) 2022

SHAP

(SHapley Additive exPlanation)

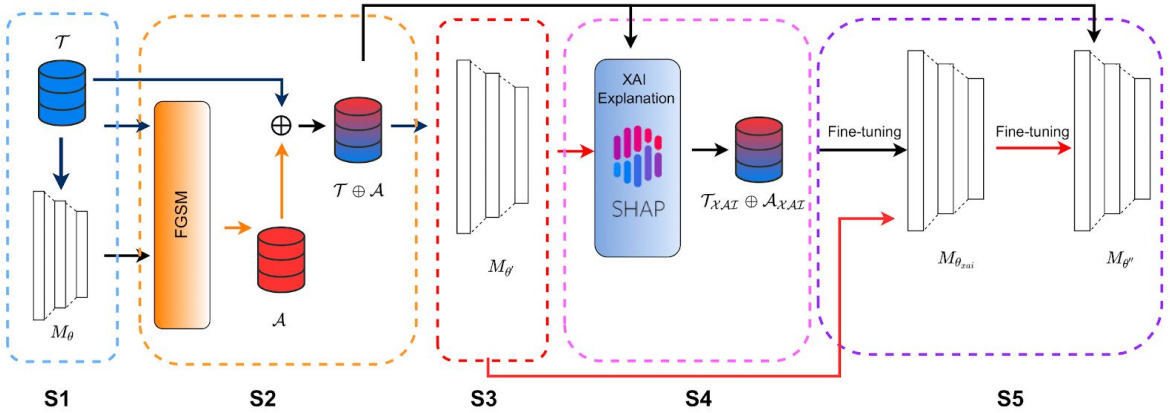
- **Model agnostic** and **post-hoc** technique
- It produces **local** explanations of decisions for single samples
- Based on the game theory, Shapley Values
  - SHAP values measure the average contribution of each feature on a decision and show whether a feature has a positive or negative effect on a decision



Lundberg et al., *A unified approach to interpreting model predictions*, in Advances, Neural Information Processing Systems, 2017

Improving accuracy with SHAP

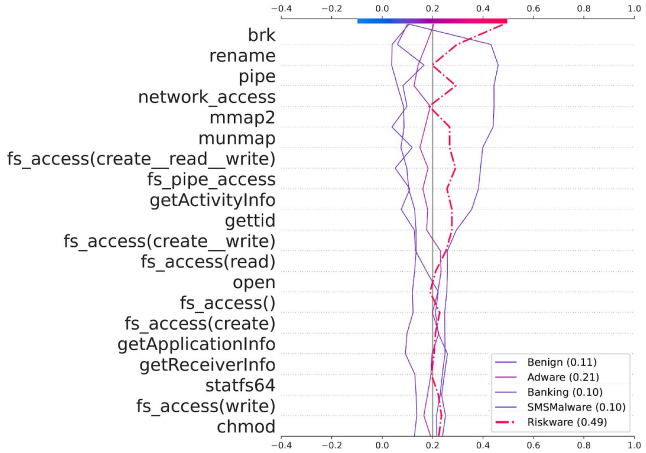
- Combination of Adversarial Training and XAI
  - Adversarial training with FGSM to create a new deep neural model robust to adversarial samples
  - SHAP to extract the local feature importance for each training sample
  - Fine-tuning of the the deep neural model driven by XAI values



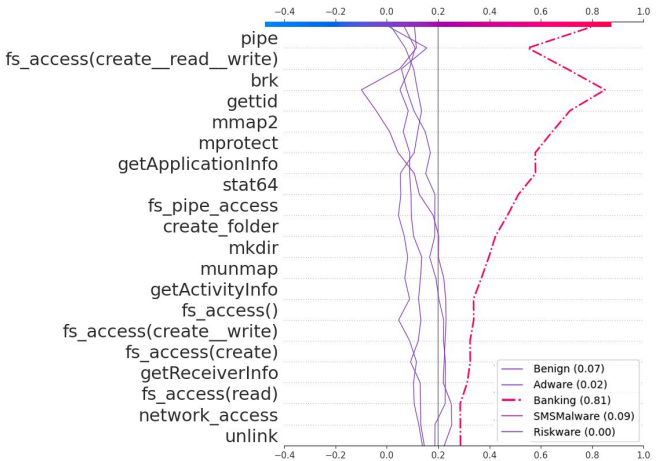
Al-Essa et al.(B) , *An XAI-based adversarial training approach for cyber-threat detection*. Accepted in IEEE CyberSciTech/ PICom/ DASC/ CDBCom, 2022

Improving accuracy with SHAP

- Analysis of SHAP explanations for the class Banking (testing samples) on Maldroid20 dataset
  - A decision yielded with a model trained using Adversarial training and XAI-based fine-tuning is better separated from a decision yielded with a model trained without fine tuning



Without Fine-tuning

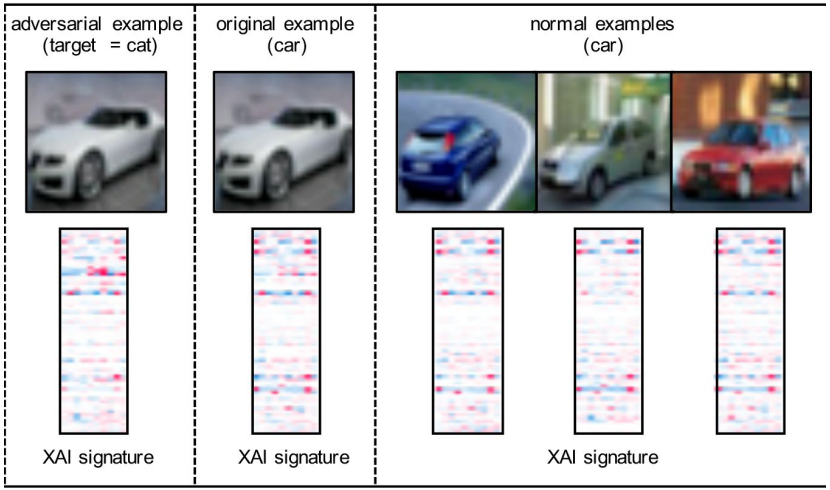


With Fine-tuning

Al-Essa et al. , *An XAI-based adversarial training approach for cyber-threat detection*. Accepted in IEEE CyberSciTech/ PICom/ DASC/ CDBCom, 2022

XAI to detect adversarial samples

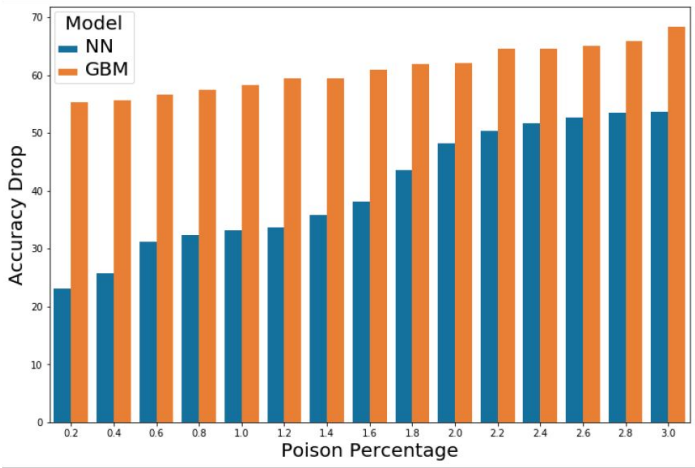
- Fidel et al. (2019) use SHAP to extract the heatmaps of both genuine and adversarial samples
  - The heatmaps are used as **signatures** to detect adversarial samples



Fidel G. et al ., *When Explainability Meets Adversarial Learning: Detecting Adversarial Examples using SHAP Signatures.* ArXiv 2019

XAI to create adversarial samples

- Kuppa et al. (2021) propose a method to create adversarial sample with XAI:
  - Identify robust features that influence the class decision boundaries of the classifier
  - Perturbate robust features to create adversarial sample



Kuppa et al., *Adversarial XAI Methods in Cybersecurity*, .IEEE Transactions on Information Forensics and Security, 2021

Conclusion

Why Machine learning is vulnerable to adversarial attacks?

- Deep Neural network is data hungry
  - [Simon-Gabriel et al 2018 -Adversarial Vulnerability of Neural Networks Increases with Input Dimension]
- Nonlinearity of deep learning models
  - Overfitting of training data
  - Insufficient generalization ability to predict unknown data
- Adversarial vulnerability is a direct result of our models' sensitivity to well-generalizing features in the data:
  - Classifiers tend to use any available signal to do so, even those that look incomprehensible to humans
  - Different classifiers tend to find the same set of relevant features
    - that is why attacks can transfer across models!

Conclusion

- Defensive strategy:
  - Know your enemy
    - Attackers' goal and capability
  - Build machine learning model able to detect zero-days attacks
    - Reducing overfitting: defensive distillation, dropout
  - Design for attacks
    - Adversarial training
    - Feature robustness (e.g., XAI)

Classify this: is a cat or a dog?

References

- **Goodfellow, Ian J. et al.** "*Deep Learning*." Nature 521 (2015): 436-444. https://www.nature.com/articles/nature14539
- **Aggarwal, Charu C..** *Neural Networks and Deep Learning*. Cham: Springer, 2018.
- **GoodFellow, Ian et al.** "Deep Learning" https://www.deeplearningbook.org/
- **Rumelhart, D., Hinton, G. & Williams, R.** *Learning representations by back-propagating errors. Nature* 323**,** 533–536 (1986). https://doi.org/10.1038/323533a0
- **S. Albawi, T. A. Mohammed and S. Al-Zawi**, "Understanding of a convolutional neural network," *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186
- **W. Wang, Y. Huang, Y. Wang and L. Wang,** "*Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction,*" *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 496-503, doi: 10.1109/CVPRW.2014.79.
- **Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua** (2014). *Generative Adversarial Nets* (PDF). Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.
- **Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy:** *Explaining and Harnessing Adversarial Examples*. ICLR (Poster) 2015
- **M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera and F. Roli**, "*Is Deep Learning Safe for Robot Vision? Adversarial Examples Against the iCub Humanoid*," *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 751-759, doi: 10.1109/ICCVW.2017.94
- **Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli.** 2015. *Support vector machines under adversarial label contamination.* Neurocomput. 160, C (July 2015), 53–62.
- **Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy**, *Explaining and Harnessing Adversarial Examples.* ICLR (Poster) 2015
- **Kurakin, Alexey & Goodfellow, Ian & Bengio, Samy.** (2016). *Adversarial examples in the physical world.* Technical report, Google, Inc.
- **Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu:** *Towards Deep Learning Models Resistant to Adversarial Attacks.* ICLR (Poster) 2018

References

- **Rey Wiyatno, Anqi Xu: *Maximal Jacobian-based Saliency Map Attack.*** CoRR abs/1808.07945 (2018)
- **M. Al- Essa, G. Andresini, A. Appice, D. Malerba.,** *XAI to explore robustness of features in adversarial training for cybersecurity* , In ISMIS 2022. Lecture Notes in Computer Science(), vol 13515. Springer, Cham.
- **M. Al- Essa, G. Andresini, A. Appice, D. Malerba.,** *An XAI-based adversarial training approach for cyber-threat detection. Accepted in IEEE CyberSciTech/ PICom/ DASC/ CDBCom, 2022*