University of Idaho

CS 404/504

Special Topics: Adversarial Machine Learning

Dr. Alex Vakanski

Lecture 1

Introduction to Adversarial Machine Learning

Lecture Outline

- Machine Learning (ML)
- Adversarial ML (AML)
  - Adversarial examples
- Attack taxonomy
- Evasion attacks and defenses
  - Common adversarial evasion attacks
    - Random noise, semantic attack, FGSM, BIM, PGD, DeepFool, C&W attack
    - Other white-box evasion attacks
    - Evasion attacks against black-box models
  - Defense against adversarial evasion attacks
    - Adversarial example detection, gradient masking/obfuscation, robust optimization
- Poisoning attacks and defenses
- Privacy attacks and defenses
- Summary
- References and AML resources

Machine Learning

Machine Learning

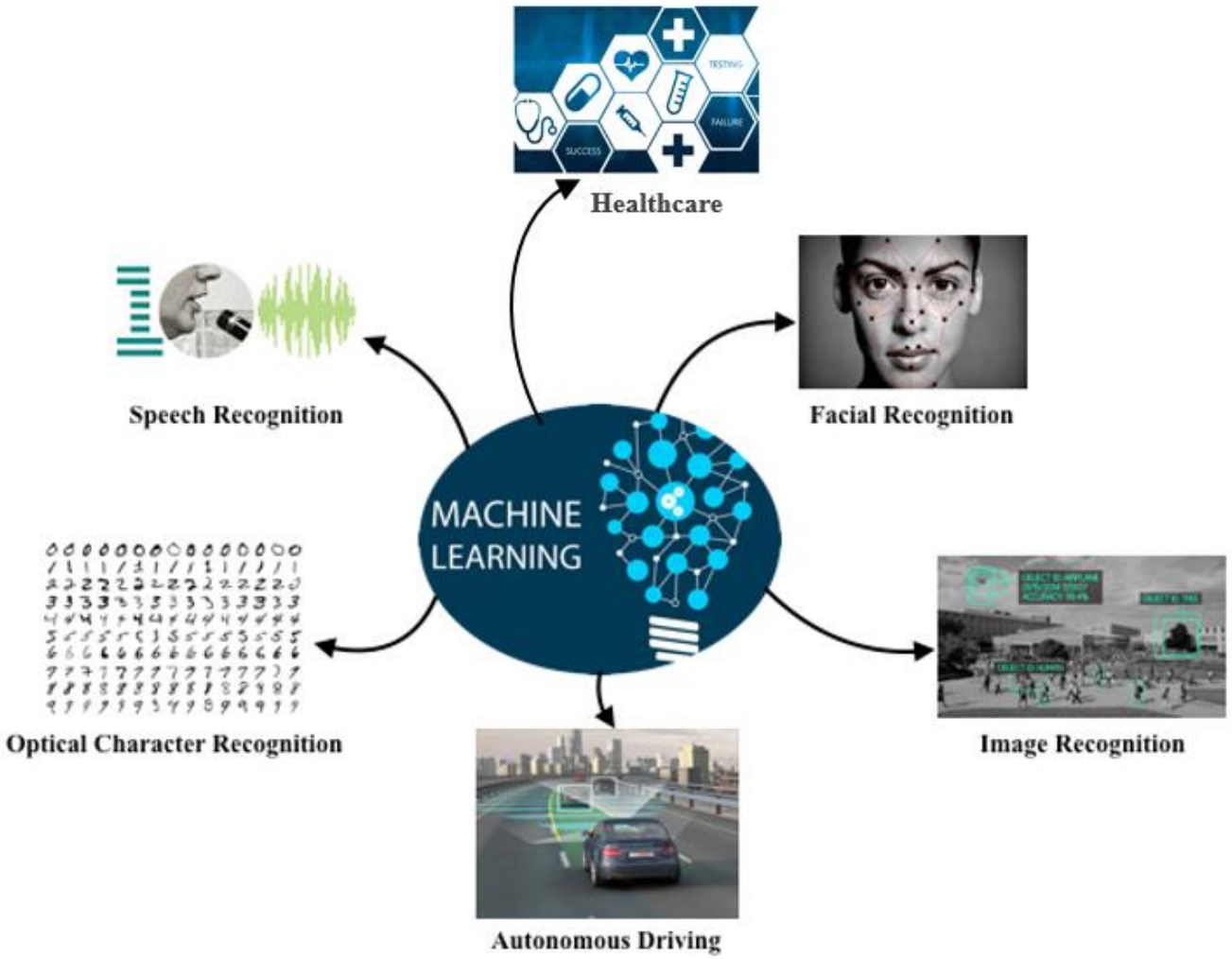
- Machine Learning (ML) tasks
  - Supervised, unsupervised, semi-supervised, self-supervised, meta learning, reinforcement learning
- Data collection and preprocessing steps
  - Data collection using various sensors, cameras, I/O devices, microphones, etc.
- Applying an ML algorithm
  - Training phase: learn ML model (parameter learning, hyperparameter tuning)
  - Testing phase (inference): predict on unseen data

Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

Machine Learning

Machine Learning

- ML is ubiquitous today in many applications



Picture from: He Xiaoyi – Adversarial Machine Learning

Adversarial ML

Adversarial Machine Learning

- The classification accuracy of GoogLeNet on MNIST images under adversarial attacks drops significantly
  - E.g., from 98.6% to 18.7% (for Projected Gradient Descent attack)
  - Or, to 1.2% (for DeepFool attack)

Attack	Lenet			
Fast Gradient Sign Method	Dataset	Acc@1 w/	Acc@5 w/	Acc@1 w/o	Acc@5 w/o
	MNIST	0.509	0.993	0.986	1.0
	ILSVRC2012	NA	NA	NA	NA
Projected Gradient Descent	Dataset	Acc@1 w/	Acc@5 w/	Acc@1 w/o	Acc@5 w/o
	MNIST	0.187	0.982	0.986	1.0
	ILSVRC2012	NA	NA	NA	NA
DeepFool	Dataset	Acc@1 w/	Acc@5 w/	Acc@1 w/o	Acc@5 w/o
	MNIST	0.012	1.0	0.9858	1.0
	ILSVRC2012	NA	NA	NA	NA

Performance on adversarial images

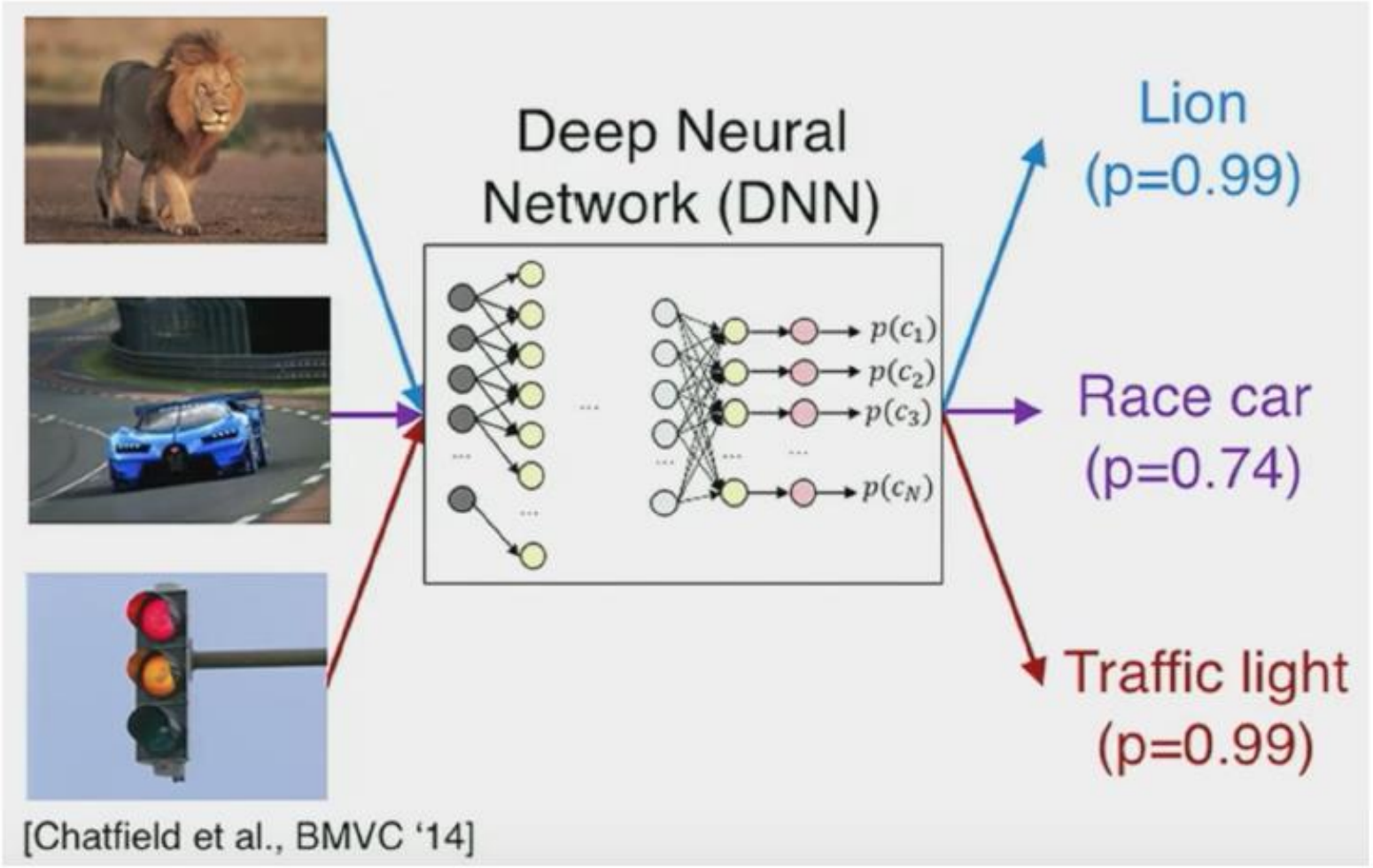
Performance on clean images

Results from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

Adversarial Examples

Adversarial Examples

- What do you see?

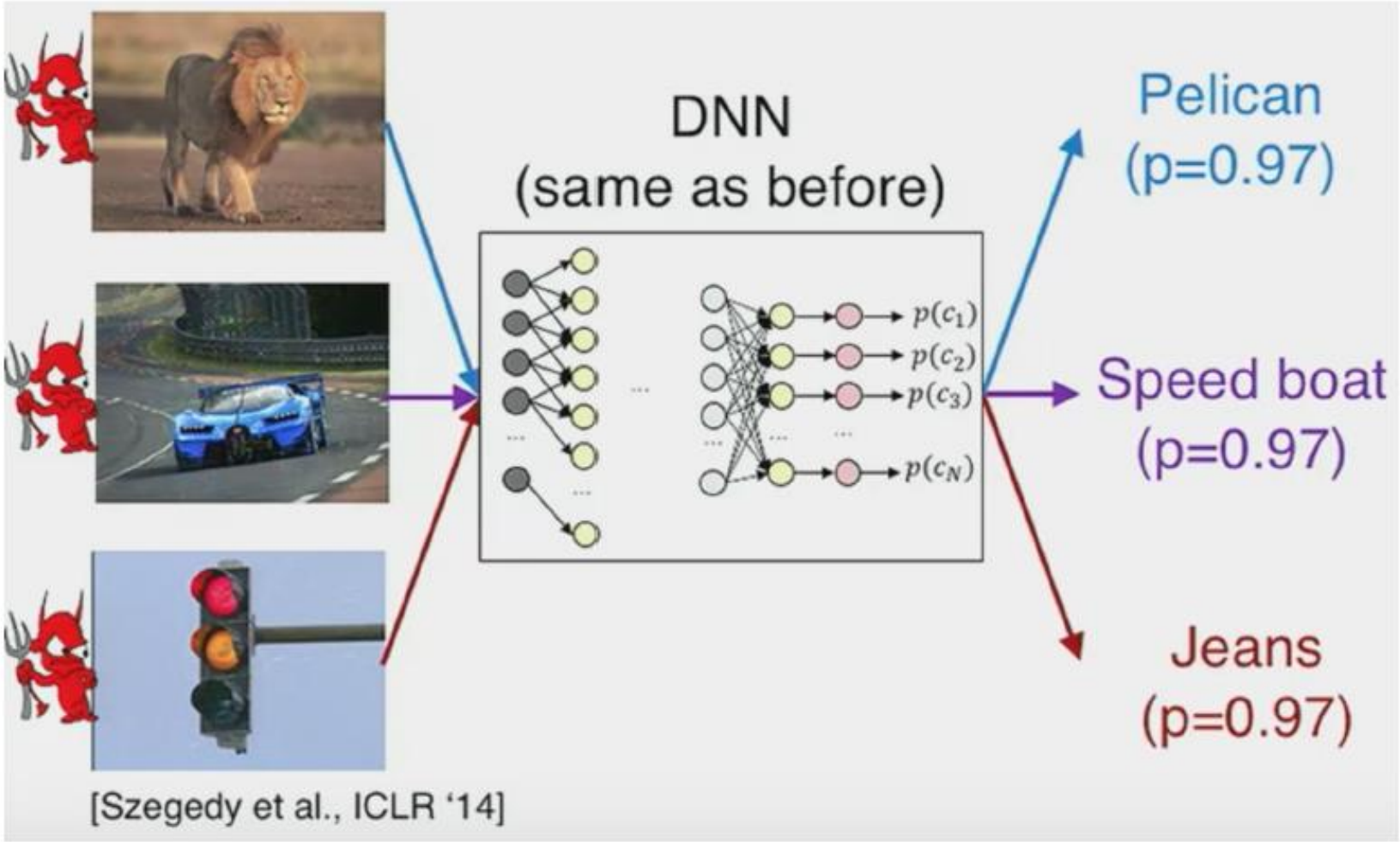


Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

Adversarial Examples

Adversarial Examples

- The trained DNN model misclassifies adversarially manipulated images

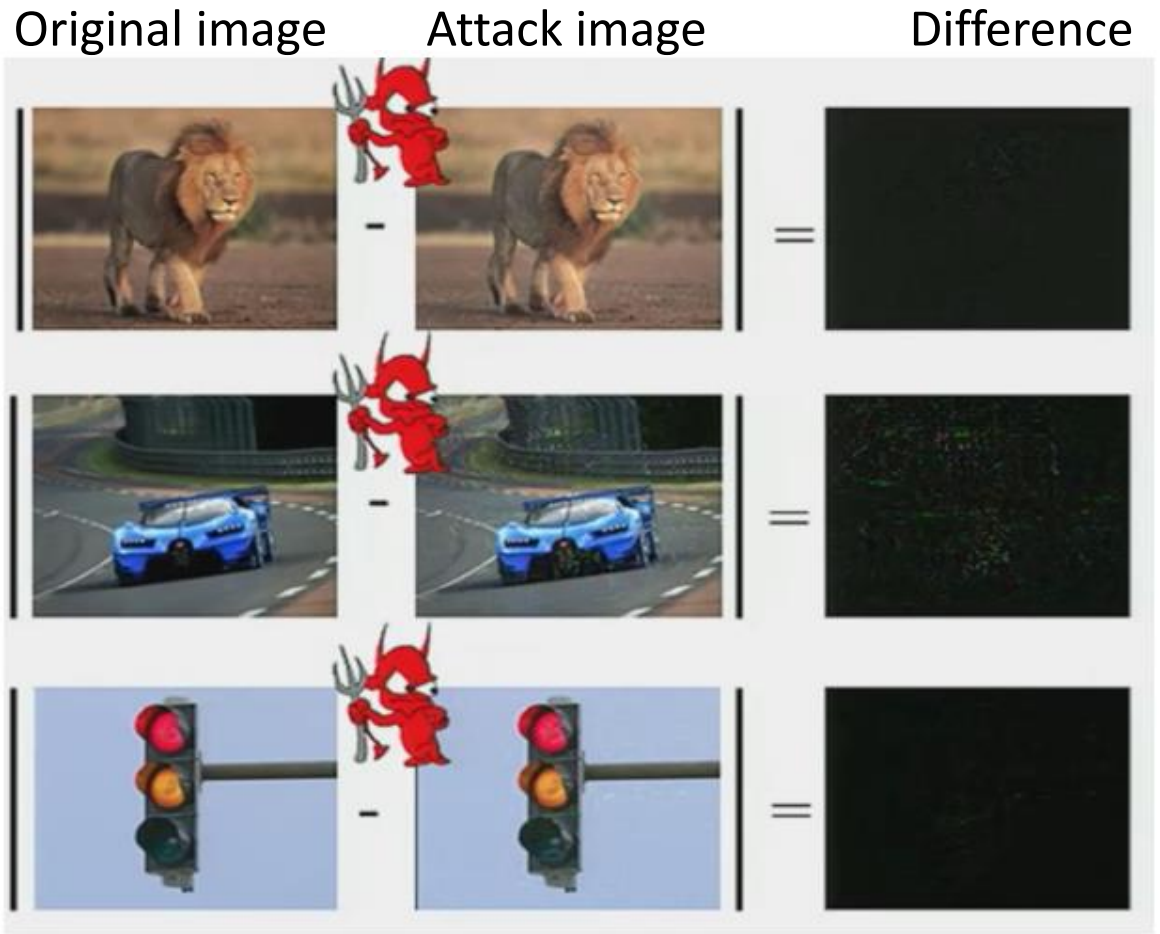


Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

Adversarial Examples

Adversarial Examples

- The differences between the original and adversarially manipulated (attack) images are very small (hardly noticeable to the human eye)



Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

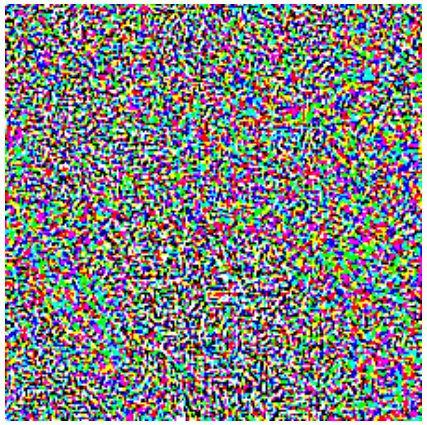
Adversarial Examples

Adversarial Examples

- One of the seminal papers on AML shows an adversarially perturbed image of a panda that is misclassified by the ML model as a gibbon
  - The image with the perturbation looks indistinguishable from the original image

Original image

Classified as panda
57.7% confidence



Small adversarial noise

Adversarial image

Classified as gibbon
99.3% confidence

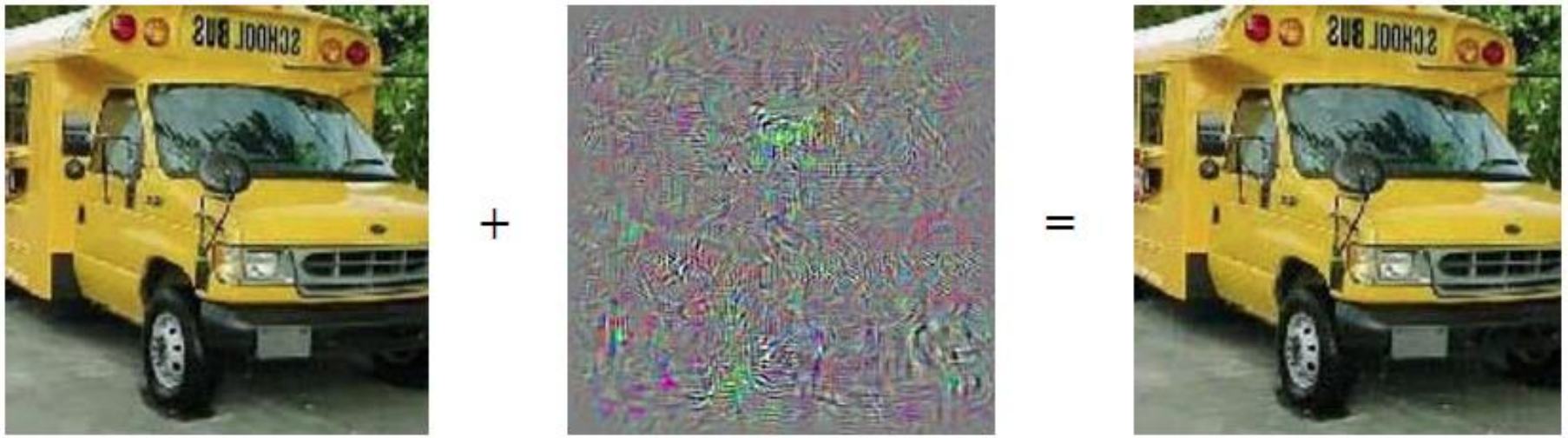
Gibbon

Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

Adversarial Examples

Adversarial Examples

- Similar example, from Szagedy (2014) Intriguing Properties of Neural Networks

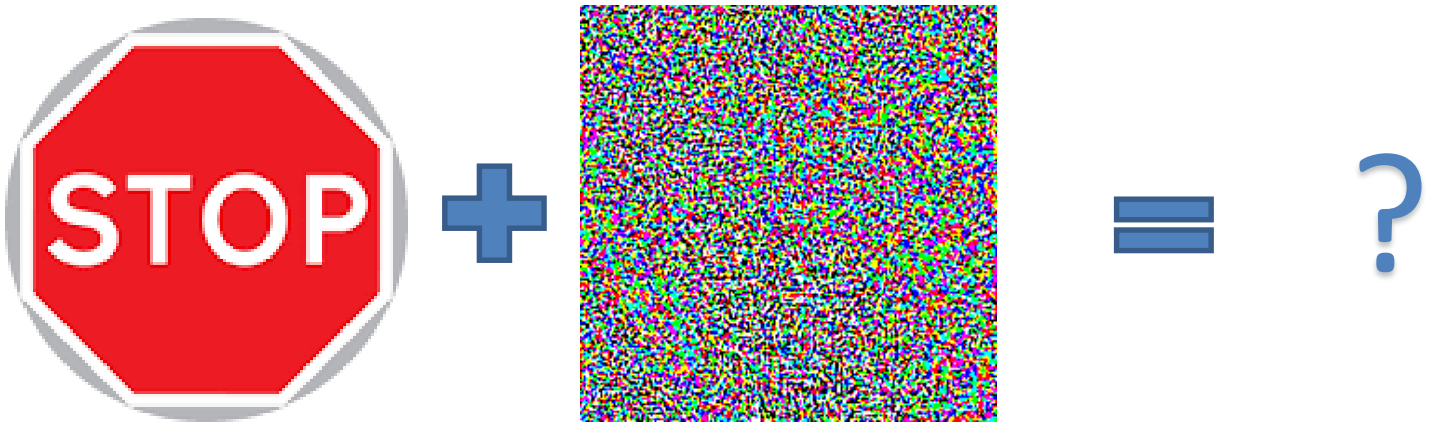


Picture from: Szagedy (2014) – Intriguing Properties of Neural Networks

Adversarial Examples

Adversarial Examples

- If a Stop sign is adversarially manipulated and it is not recognized by a self-driving car: the car can keep going, and it can result in an accident



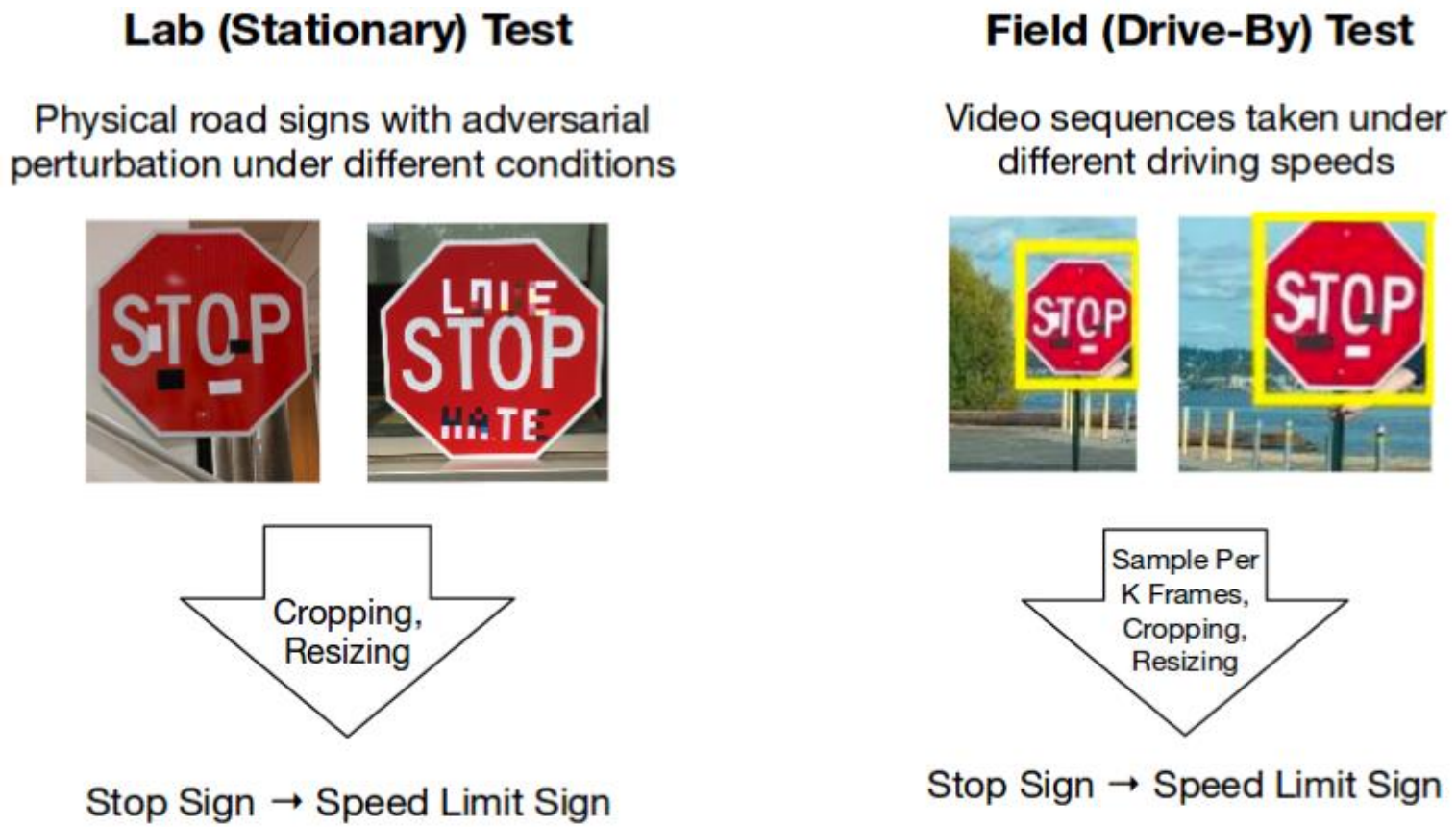
Small adversarial noise

Slide credit: He Xiaoyi – Adversarial Machine Learning

Adversarial Examples

Adversarial Examples

- The authors of this work manipulated a Stop sign with adversarial patches
  - Caused the DL model of a self-driving car to missclassify it as a Speed Limit 45 sign
    - The authors achieved 100% attack success in lab test, and 85% in field test



Picture from: Eykholt (2017) - Robust Physical-World Attacks on Deep Learning Visual Classification

Adversarial Examples

Adversarial Examples

- Lab test images for signs with a target class Speed Limit 45

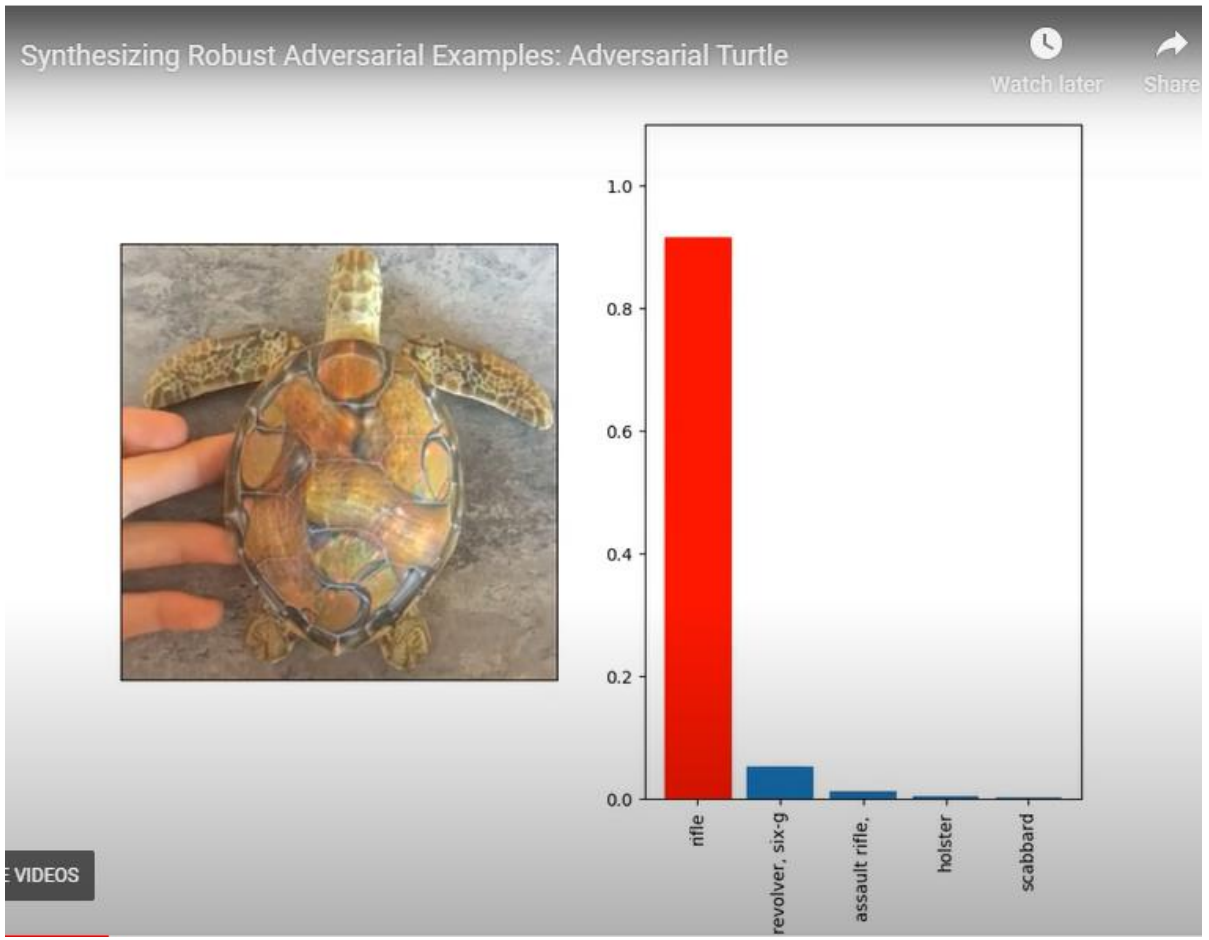
Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5′ 0°					
5′ 15°					
10′ 0°					
10′ 30°					
40′ 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Picture from: Eykholt (2017) - Robust Physical-World Attacks on Deep Learning Visual Classification

Adversarial Examples

Adversarial Examples

- This paper presents an example of a 3D-printed turtle that is misclassified by a DNN as a rifle (video link)
  - The texture of the turtle is designed to mislead the DNN



Adversarial Examples

Adversarial Examples

- A person wearing an adversarial patch is not detected by a person detector DNN model (YOLOv2)
  - E.g., can be used by intruders to get past security cameras

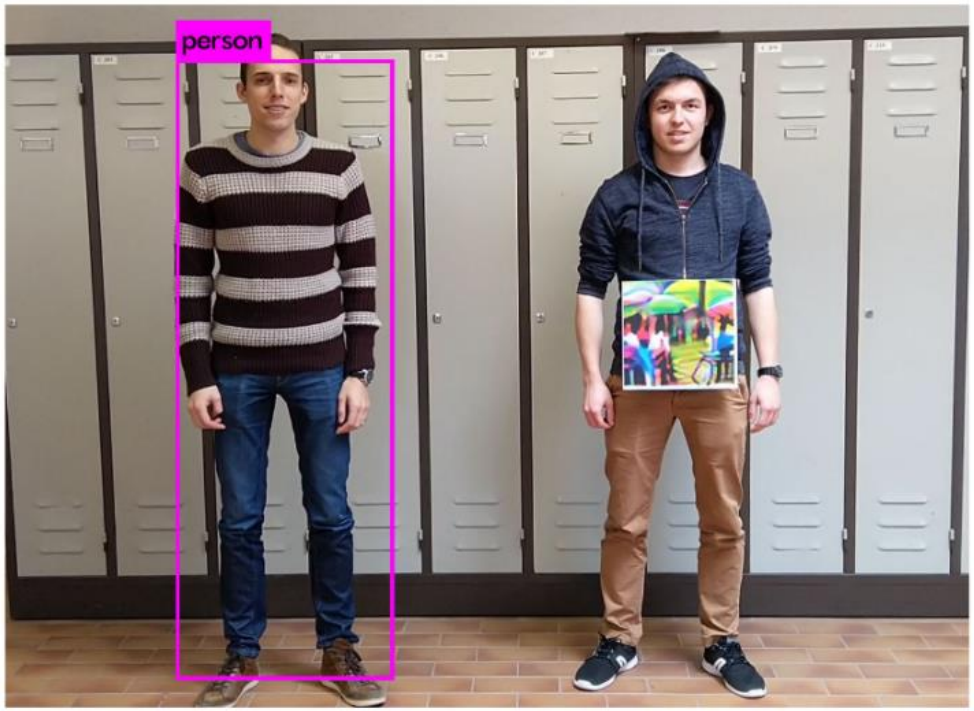
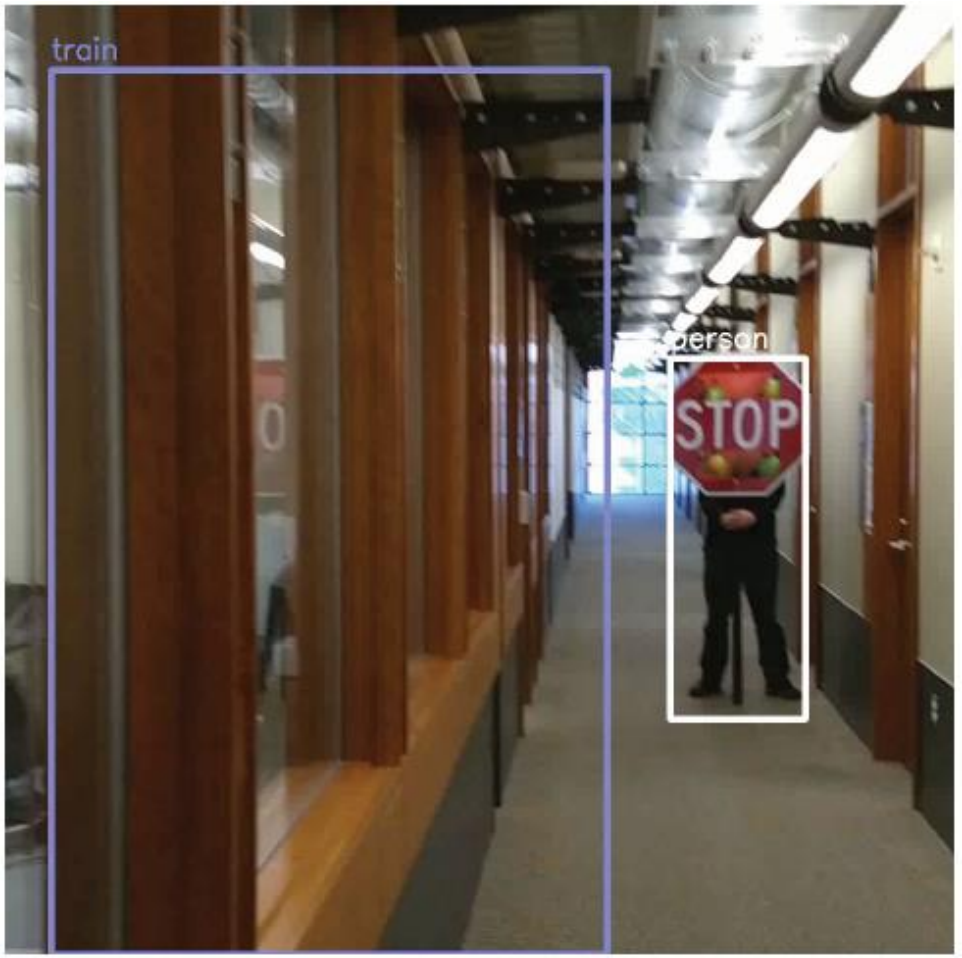


Figure 1: We create an adversarial patch that is successfully able to hide persons from a person detector. Left: The person without a patch is successfully detected. Right: The person holding the patch is ignored.

Adversarial Examples

Adversarial Examples

- A "train" in the hallway?
  - Most ML models make predictions by pattern recognition (i.e., finding features in images)
  - ML models fail to use the context in images, e.g., cannot understand that a train cannot fit in a hallway
  - On the positive side, the person holding the Stop sign is correctly predicted as Person by the DNN

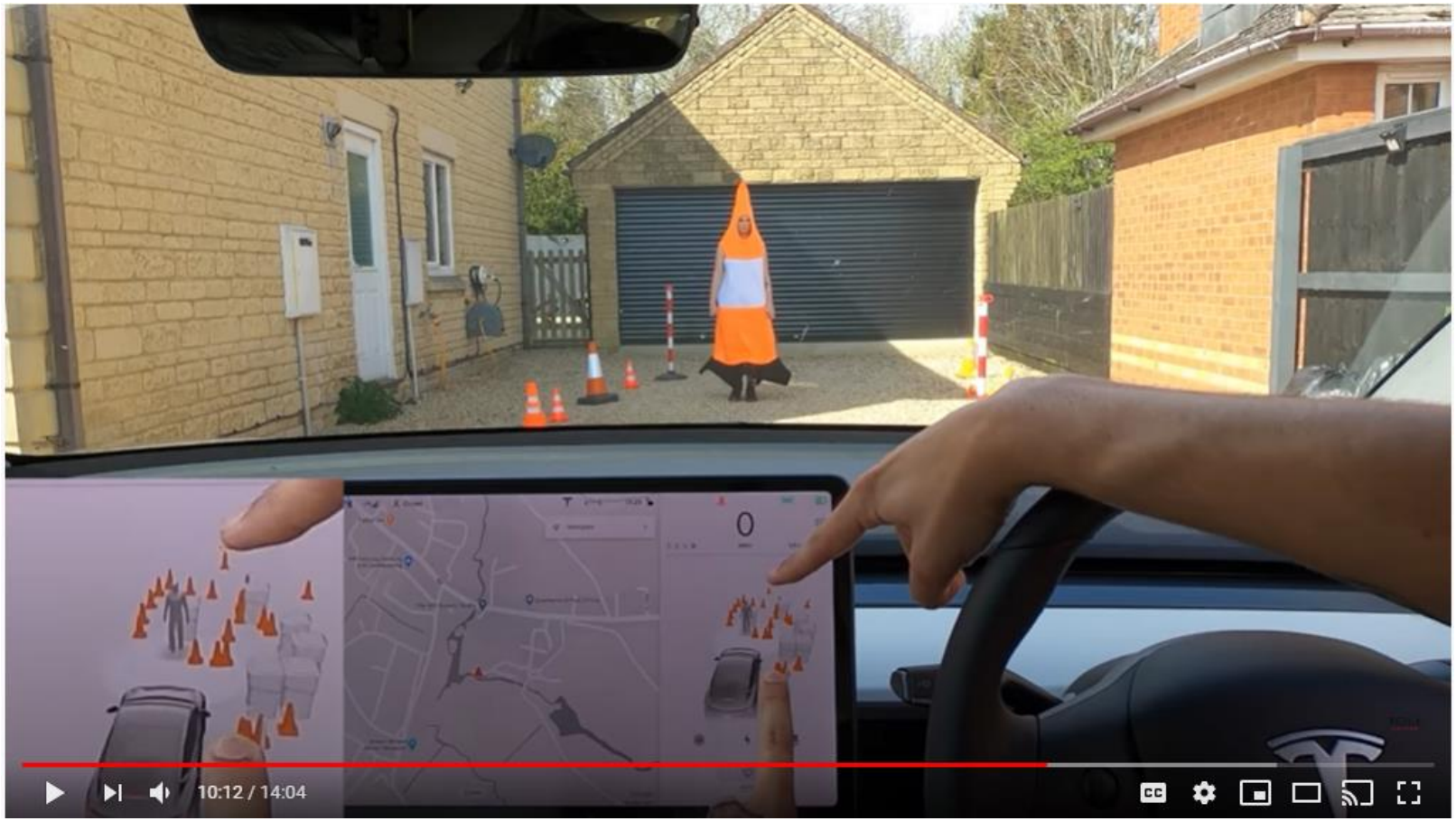


Picture from: Yevgeniy Vorobeychik, Bo Li - Adversarial Machine Learning Tutorial

Adversarial Examples

Adversarial Examples

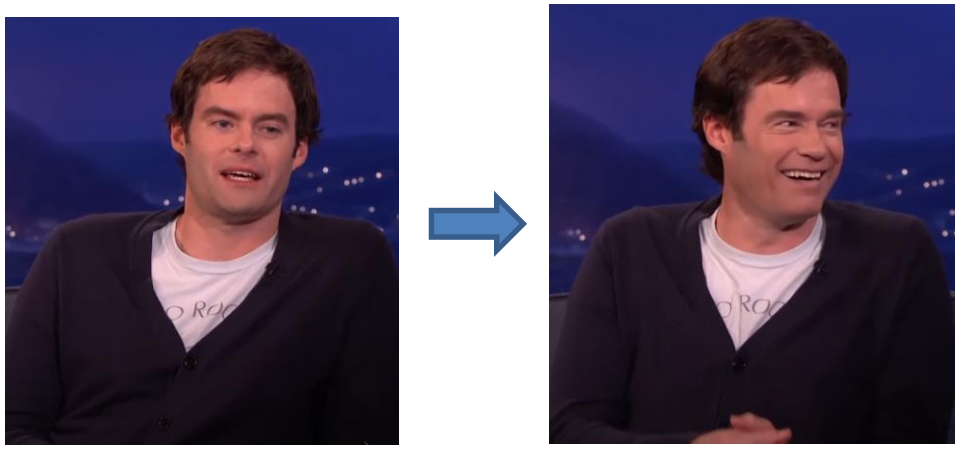
- Non-scientific example: a Tesla owner checks if the car can distinguish a person wearing a cover-up from a traffic cone (video link)



Adversarial Examples

Adversarial Examples

- Abusive use of machine learning is on the rise
  - E.g., using GANs to generate fake content (a.k.a. deep fakes)
    - Videos of politicians saying things they never said
      - Barak Obama's deep fake, or the House Speaker Nancy Pelosi appears drunk in a video
    - Bill Hader's impersonation of Arnold Schwarzenegger
  - Can have strong societal implications: elections, automated trolling, court evidence



Adversarial ML

Adversarial Machine Learning

- ML algorithms in real-world applications mainly focus on increased accuracy
  - Before 2015, few techniques and design decisions focused on keeping the ML models secure and robust
- ***Adversarial ML: ML in adversarial settings***
  - Attack is a major component of AML
  - Bad actors do bad things
    - Their main objective is not to get detected (change behavior to avoid detection)
- ***Adversarial examples*** are inputs to ML models that an attacker intentionally designed to cause the model to make mistakes

Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

Adversarial ML

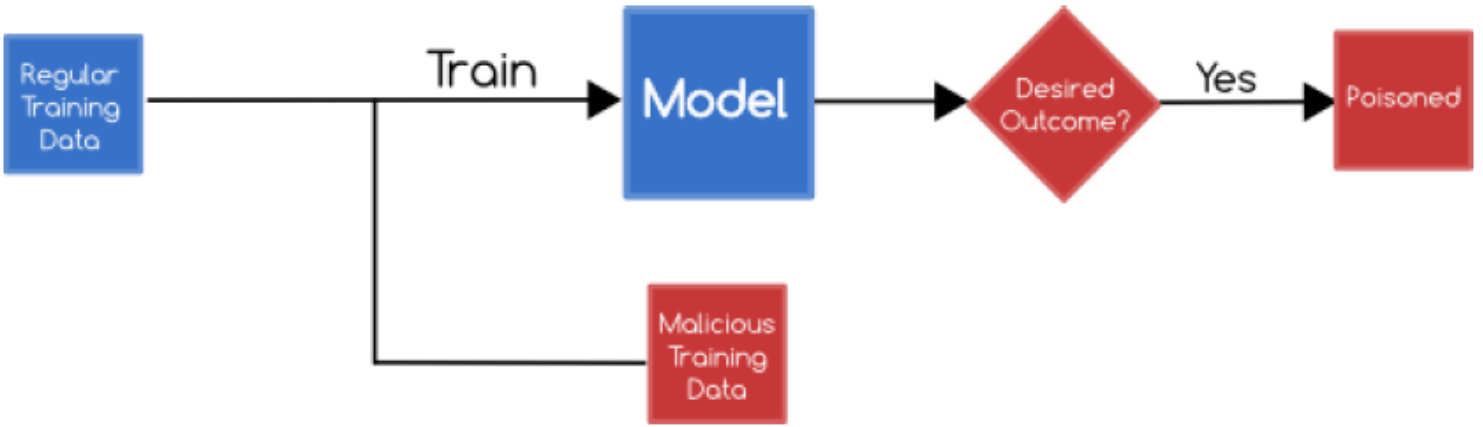
Adversarial Machine Learning

- Studying adversarial examples in the image domain has been predominant in the prior work in AML
- Reasons:
  1. Perceptual similarity between clean and adversarial images is intuitive to observers
  2. Image data and image classifiers have simpler structure than other domains (e.g., audio, or malware)
- Commonly used datasets for concept evaluation in AML include:
  - MNIST: 60K images, digits 0 to 9
  - CIFAR-10: 60K images, 10 classes: cars, birds, airplanes, cats, dogs, deer, frogs, horses, ships, trucks
  - ImageNet: 14M images, 20K classes

Poisoning Attack

Attack Taxonomy

- ***Poisoning attack*** (causative attack)
  - Attack on the training phase
    - Attackers perturb the training set or the model
      - Insert malicious inputs in the training set (e.g., that contain a trigger pattern to backdoor the model)
      - Change the labels to training inputs
      - Change the weights of a deployed trained model
      - Distribute poisoned pretrained models to the public
    - The goal is to corrupt the ML model so that it performs incorrectly for some or all inputs
    - The adversary should obtain access to the training database or to the trained model
      - E.g., web-based repositories and "honeypots" often collect malware examples for training, which provides an opportunity for adversaries to poison the data

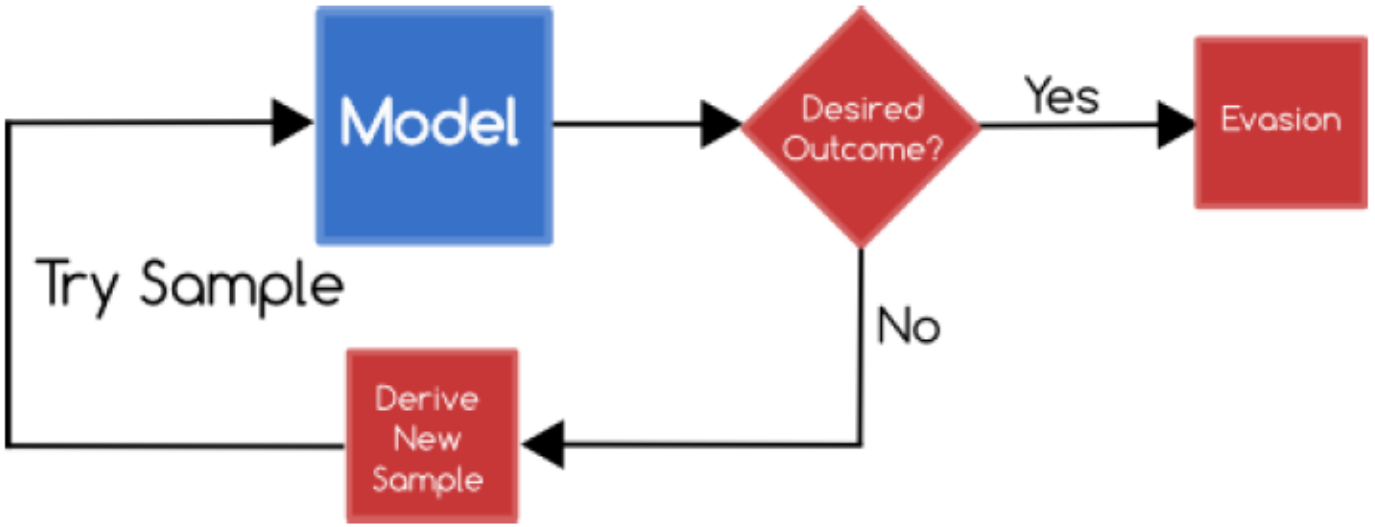


Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

Evasion Attack

Attack Taxonomy

- ***Evasion attack*** (exploratory attack)
  - Attack on the testing phase
    - Attackers do not tamper with the ML model, but instead cause it to misclassify adversarial inputs
    - Evasion attack is more common attack than poisoning attack
    - E.g., the shown examples with sticking a few pieces of tapes on a Stop sign can cause misclassification by the ML model for road sign recognition used by an autonomous driving vehicle



Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

White-box and Black-box Attack

Attack Taxonomy

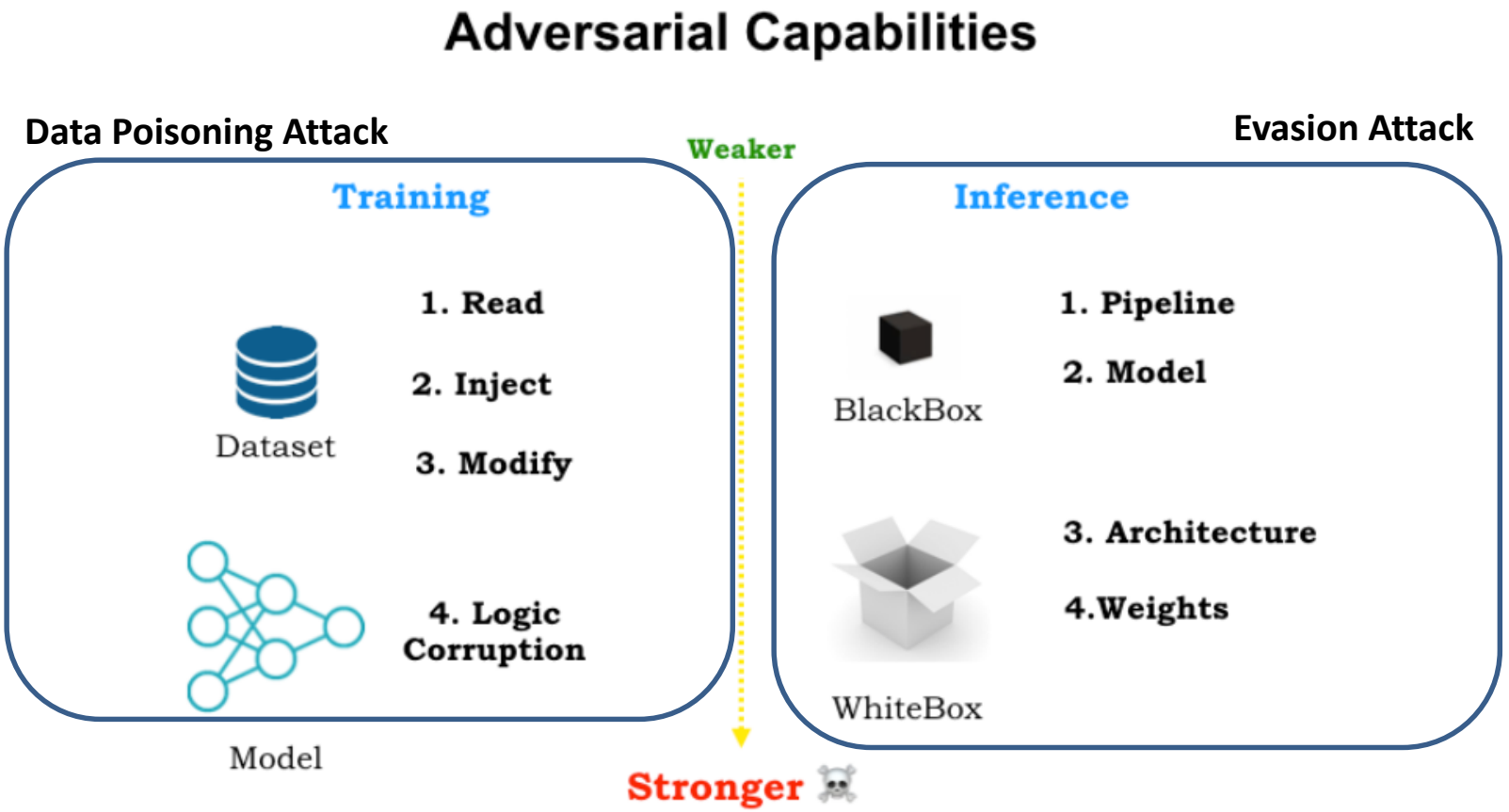
- Adversarial attacks can be further classified into:
  - ***White-box attack***
    - Attackers have full knowledge about the ML model
    - I.e., they have access to parameters, hyperparameters, gradients, architecture, etc.
  - ***Black-box attack***
    - Attackers don't have access to the ML model parameters, gradients, architecture
    - Perhaps they have some knowledge about the used ML model
      - E.g., attackers may know that a ResNet50 model is used for classification, but they don't have access to the model parameters
    - Attackers may query the black-box model (also known as the *oracle*) to obtain knowledge about the model
      - E.g., submit adversarial examples, and obtain the model's output (class label or probability vector)
      - A body of work has focused on query-efficient black-box attacks, where the goal is to synthesize adversarial examples using a limited number of queries
    - Black-box attacks are more realistic, because model designers usually do not open source the model parameters

Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

Attack Taxonomy

Attack Taxonomy

- Depiction of the adversarial attack taxonomy from Alessio's Adversarial ML presentation at FloydHub



Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

Non-targeted and Targeted Attack

Attack Taxonomy

- Each of the described attacks can further be:
  - ***Non-targeted*** attack
    - The goal is to mislead the classifier for an adversarial input to output any label other than the ground-truth label
    - E.g., perturb an image of a military tank, so that the model predicts it is any other class than a military tank
  - ***Targeted*** attack
    - The goal is to mislead the classifier to predict a target label for an adversarial input
    - More difficult, in comparison to non-targeted attack
    - E.g., perturb an image of a turtle, so that the model predicts it is a riffle
    - E.g., perturb an image of a Stop sign, so that the model predicts it is a Speed Limit 45 sign

Privacy and Availability Attacks

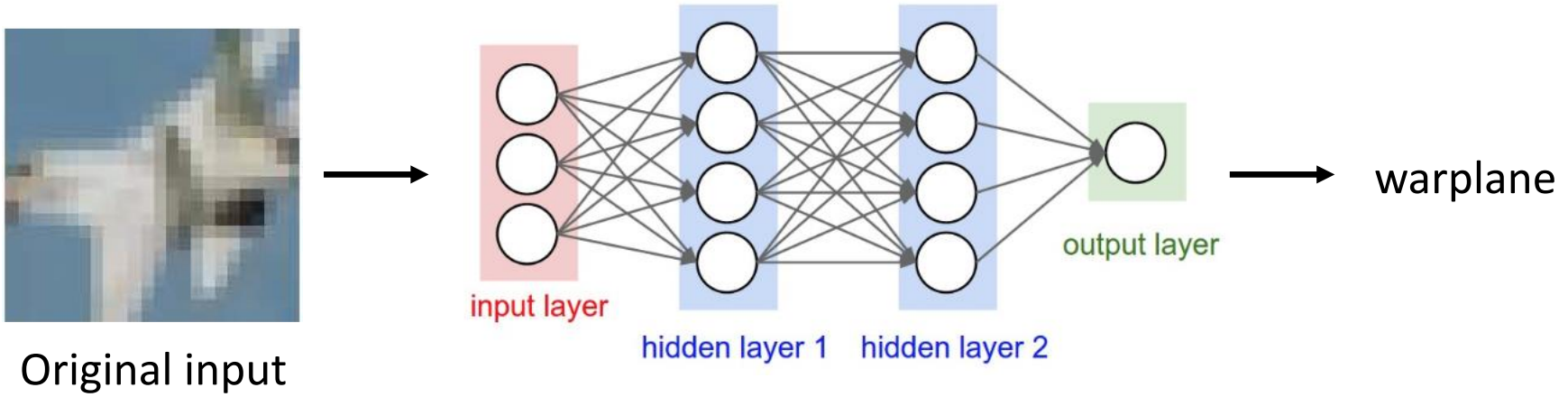
Attack Taxonomy

- In some references, poisoning and evasion attacks are grouped together into ***integrity attacks***
  - It means that the integrity of the model to perform correctly for adversarially manipulated inputs is attacked
- In other types of attacks, adversaries' goal is other than causing incorrect performance of ML models, such as:
  - ***Privacy attack***
    - The goal is to illegitimately gain knowledge about the training inputs and models
    - A.k.a. confidentiality attack, or inference attack
    - E.g., membership inference aims to identify whether or not a data sample was used to train an ML model
  - ***Availability attack***
    - Cause an ML system to become unavailable or block regular use of the system (denial-of-service)
    - E.g., design adversarial samples that take an extremely long time for the ML model to process

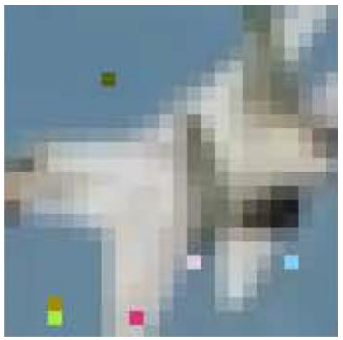
Evasion Attack

Evasion Attack

- Find a new input (*similar* to the original input) but classified as another class (non-targeted or targeted)



- Adversarial attack image



Slide credit: He Xiaoyi – Adversarial Machine Learning

How to Find Adversarial Examples

Evasion Attack

- One common approach to finding adversarial examples is as follows
  - Take an image x, which is labeled by the classifier C (e.g., Logistic Regression, SVM, or NN) as class y, i.e., $C(x) = y$
  - Create an adversarial image x_{adv} by adding small perturbations δ to the original image x, i.e., $x_{adv} = x + \delta$, such that the distance $D(x, x_{adv}) = D(x, x + \delta)$ is minimal
  - The aim for a non-targeted attack is that the classifier assigns a label to the adversarial image that is different than y, i.e., $C(x_{adv}) = C(x + \delta) \neq y$
    - Or, for a targeted attack, the aim is that $C(x_{adv}) = C(x + \delta) = t \neq y$, where t is the target class

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

distance between x and $x+\delta = x_{adv}$

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

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

Distance Metrics

Evasion Attack

- Several distance metrics between a clean sample x and adversarial sample x_{adv}, i.e., $D(x, x_{adv})$, have been used
  - 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
    - $\ell_1 = |x^1 - x^1_{adv}| + |x^2 - x^2_{adv}| + \cdots + |x^n - x^n_{adv}|$
  - ℓ_2 norm: Euclidean distance, or root-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
    - $\ell_\infty = max(|x^1 \neq x^1_{adv}|, |x^2 \neq x^2_{adv}|, \ldots, |x^n \neq x^n_{adv}|)$
- ℓ_1, ℓ_2, and ℓ_∞ norms can be considered special cases of the general ℓ_p norm
  - $\ell_p = \left(|x^1 - x^1_{adv}|^p + |x^2 - x^2_{adv}|^p + \cdots + |x^n - x^n_{adv}|^p\right)^{\frac{1}{p}}$

Common Adversarial Evasion Attacks

Common Adversarial Evasion Attacks

- Random noise attack
- Semantic attack
- Fast gradient sign method (FGSM) attack
- Basic iterative method (BIM) attack
- Projected gradient descent (PGD) attack
- DeepFool attack
- Carlini & Wagner (C&W) attack

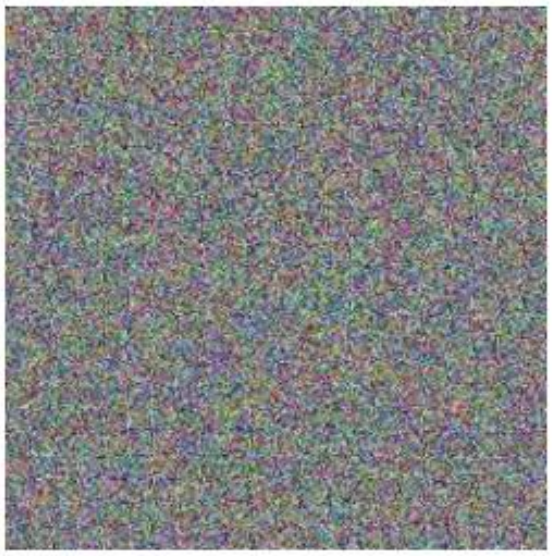
Random Noise Attack

Common Adversarial Evasion Attacks

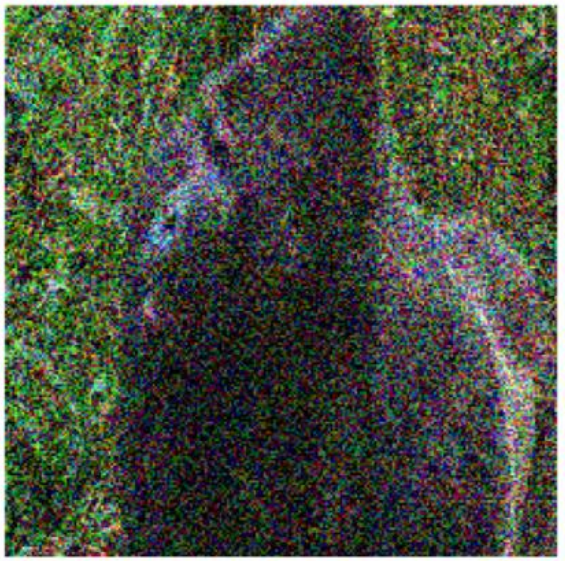
- ***Random noise attack***
  - The simplest form of adversarial attack
  - Noise is a random arrangement of pixels containing no information
    - E.g., random numbers from a normal distribution (with 0 mean and 1 st. dev.)
  - This attack represents a non-targeted black-box evasion attack
    - It is not efficient, since the noisy images are easily distinguishable from the original images

Prediction: gorilla

+



=



Prediction: fountain

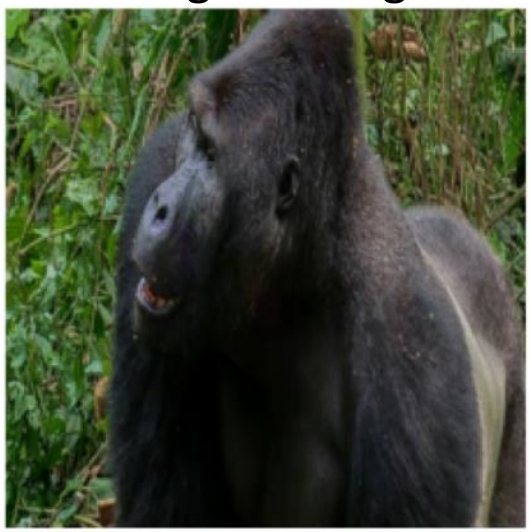
Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

Semantic Attack

Common Adversarial Evasion Attacks

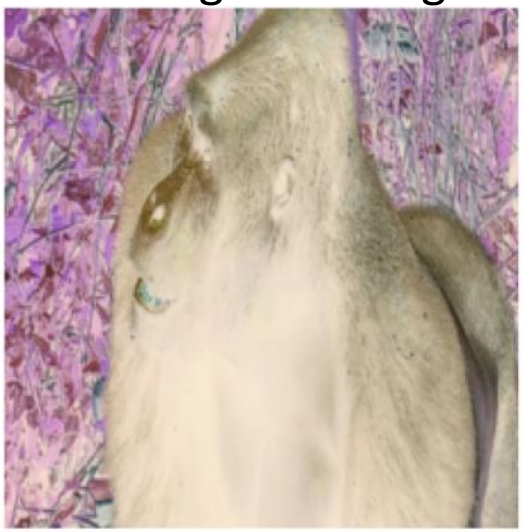
- ***Semantic attack***
  - Hosseini (2017) On the Limitation of Convolutional Neural Networks in Recognizing Negative Images
  - Use negative images as adversarial inputs
    - Reverse all pixels intensities
    - E.g., change the sign of all pixels, if the pixels values are in range [-1,1]
  - Not an efficient attack, since the adversarial images can easily be detected

Original image



Prediction: gorilla

Negative image



Prediction: weimaraner

Weimaraner (a dog breed)

Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

FGSM Attack

Common Adversarial Evasion Attacks

- ***Fast gradient sign method (FGSM) attack***
  - Goodfellow (2015) Explaining and Harnessing Adversarial Examples
- An adversarial image x_{adv} is created by adding perturbation noise to an image x

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

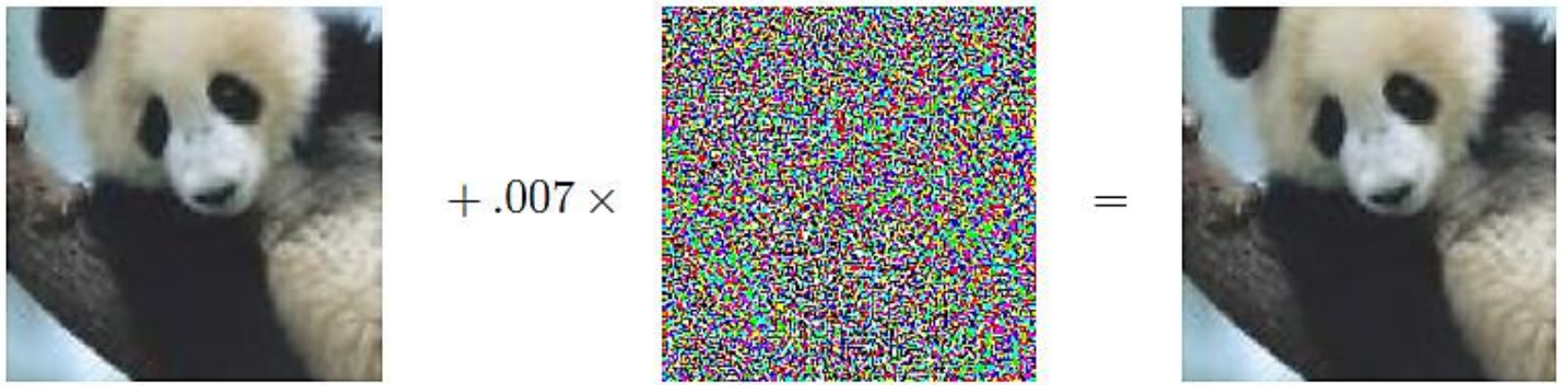
  - Notation: input image x, label y, NN model C, NN weights (parameters) w, loss function $\mathcal{L}$, gradient ∇ (Greek letter "nabla"), perturbation magnitude ϵ
  - The amount of perturbation is calculated based on the gradient of the loss function $\mathcal{L}$ with respect to the input image x for the true class label y (i.e., $\nabla_x \mathcal{L}(C(x, w), y)$
  - The perturbation increases the loss $\mathcal{L}$ for the true class y, and the model C misclassifies the image x_{adv}
  - The sign function sgn(·) is defined as:

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

FGSM Attack

Common Adversarial Evasion Attacks

- FGSM is a white-box non-targeted evasion attack
  - White-box, since we need to know the gradients $\nabla_x \mathcal{L}(C(x, w), y)$ of the model to create the adversarial image
  - In the shown example, the noise magnitude is $\epsilon = 0.007$
    - Note: nematode is an insect referred to as roundworm
  - FGSM calculates how much perturbation to add to each pixel in x, so that the loss $\mathcal{L}$ is maximized, leading to incorrect prediction by the model (i.e., $C(x, w) \neq y$)



x

"panda"
57.7% confidence

$\text{sign}\left(\nabla_x \mathcal{L}(C(x, w), y)\right)$

"nematode"
8.2% confidence

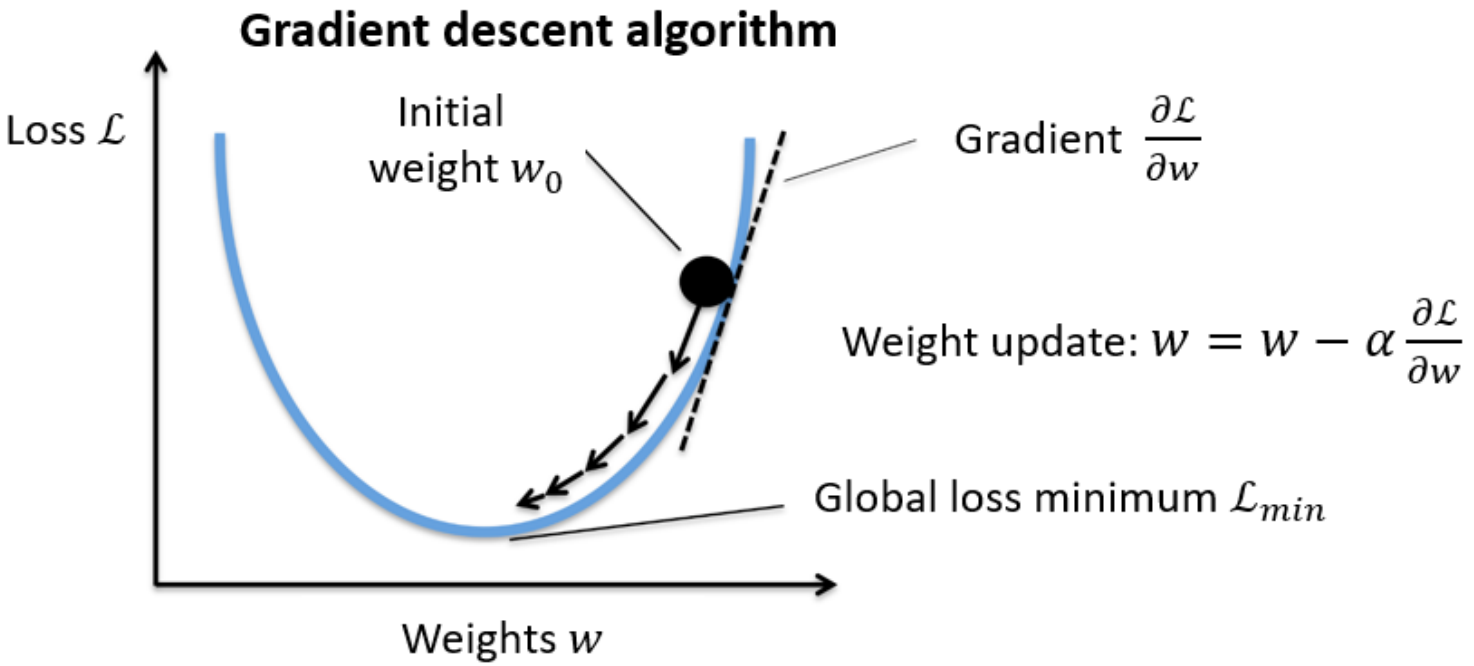
$x + \epsilon \cdot \text{sign}\left(\nabla_x \mathcal{L}(C(x, w), y)\right)$

"gibbon"
99.3 % confidence

FGSM Attack

Common Adversarial Evasion Attacks

- Recall that training NNs is based on the gradient descent algorithm
  - The values of the model parameters (weights) w are iteratively changed until a minimum of the loss function $\mathcal{L}$ is reached
  - Gradients of the loss with respect to the model parameters ($\partial\mathcal{L}/\partial w$) give the direction and magnitude for updating the parameters
  - The step of each update is the learning rate α

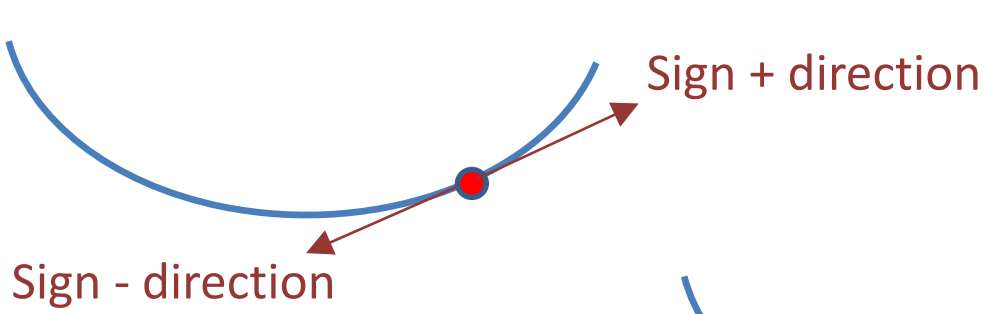


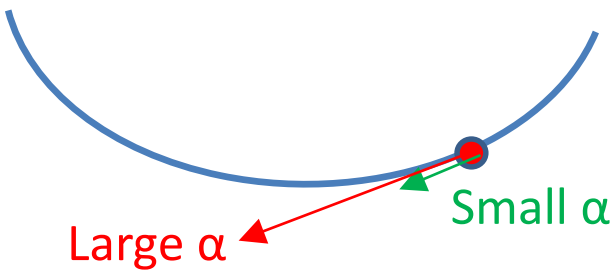
Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

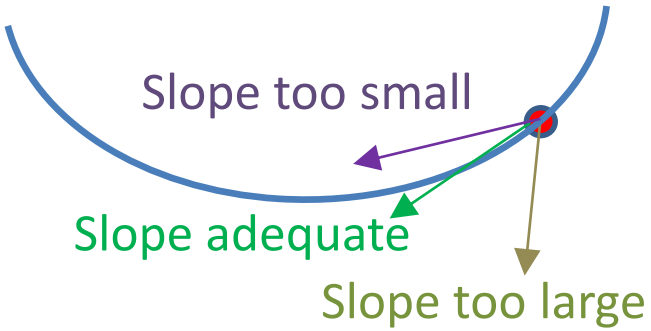
FGSM Attack

Common Adversarial Evasion Attacks

- The sign and magnitude of the gradient $\frac{\partial \mathcal{L}}{\partial w}$ give the direction and the slope of the steepest descent
  - Left image: + and – sign of the gradient
  - Right image: small, adequate, and large slope of the weight update, based on the magnitude of the gradient
  - Middle image: small and large α (learning rate)
- To minimize the loss function, the weights w are updated in the opposite direction of the gradient, i.e., $w = w - \alpha \frac{\partial \mathcal{L}}{\partial w}$







FGSM Attack

Common Adversarial Evasion Attacks

- Note the difference:
  - For *model training*, the gradient of the loss $\mathcal{L}$ with respect to the model parameters w, i.e., $\nabla_w \mathcal{L}(C(x, w), y)$, is used for calculating the parameter updates
    - Typically, a cumulative loss is calculated over a batch of input-label pairs (x_i, y_i)
    - Common choices for the loss function $\mathcal{L}$ in DNNs are cross-entropy and mean-squared error
    - The model parameters are updated iteratively based on $w = w - \alpha \nabla_w \mathcal{L}(C(x, w), y)$
    - The gradient $\nabla_w \mathcal{L}(C(x, w), y)$ is used to calculate how much to change each individual weight w_j in order to minimize the loss $\mathcal{L}(C(x, w), y)$
    - The goal is find w for which the loss $\mathcal{L}(C(x, w), y)$ is minimal
      - That is, to minimize the difference between the model predictions $C(x_i, w)$ and the labels y_i
  - For *generating adversarial samples*, the gradient of the loss $\mathcal{L}$ with respect to an input image x, i.e., $\nabla_x \mathcal{L}(C(x, w), y)$, is used
    - The model parameters w are fixed, since the aim is to attack a trained model
    - An adversarial image is created based on $x_{adv} = x + \epsilon \cdot \text{sign}\big(\nabla_x \mathcal{L}(C(x, w), y)\big)$
    - The gradient $\nabla_x \mathcal{L}(C(x, w), y)$ is used to calculate how much to change each individual pixel in x in order to maximize the loss $\mathcal{L}(C(x_{adv}, w), y)$
    - The goal is find x_{adv} for which the loss $\mathcal{L}(C(x_{adv}, w), y)$ is maximal
      - That is, to maximize the difference between the model prediction $C(x_{adv}, w)$ and the label y

FGSM Attack

Common Adversarial Evasion Attacks

- FGSM attack example

Original image

Prediction: car mirror

Adversarial image

Prediction: sunglasses

Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

BIM Attack

Common Adversarial Evasion Attacks

- ***Basic iterative method (BIM) attack***
  - Kurakin (2017) Adversarial Examples in the Physical World
- BIM is a variant of FGSM: it repeatedly adds noise to the image x in multiple iterations, in order to cause misclassification
  - The number of iterations steps is t, and γ is the amount of perturbation noise that is added at each step

$$x_{adv}^{t} = x^{t-1} + \gamma \cdot \text{sign}\big(\nabla_x \mathcal{L}(C(x^{t-1}, w), y)\big)$$

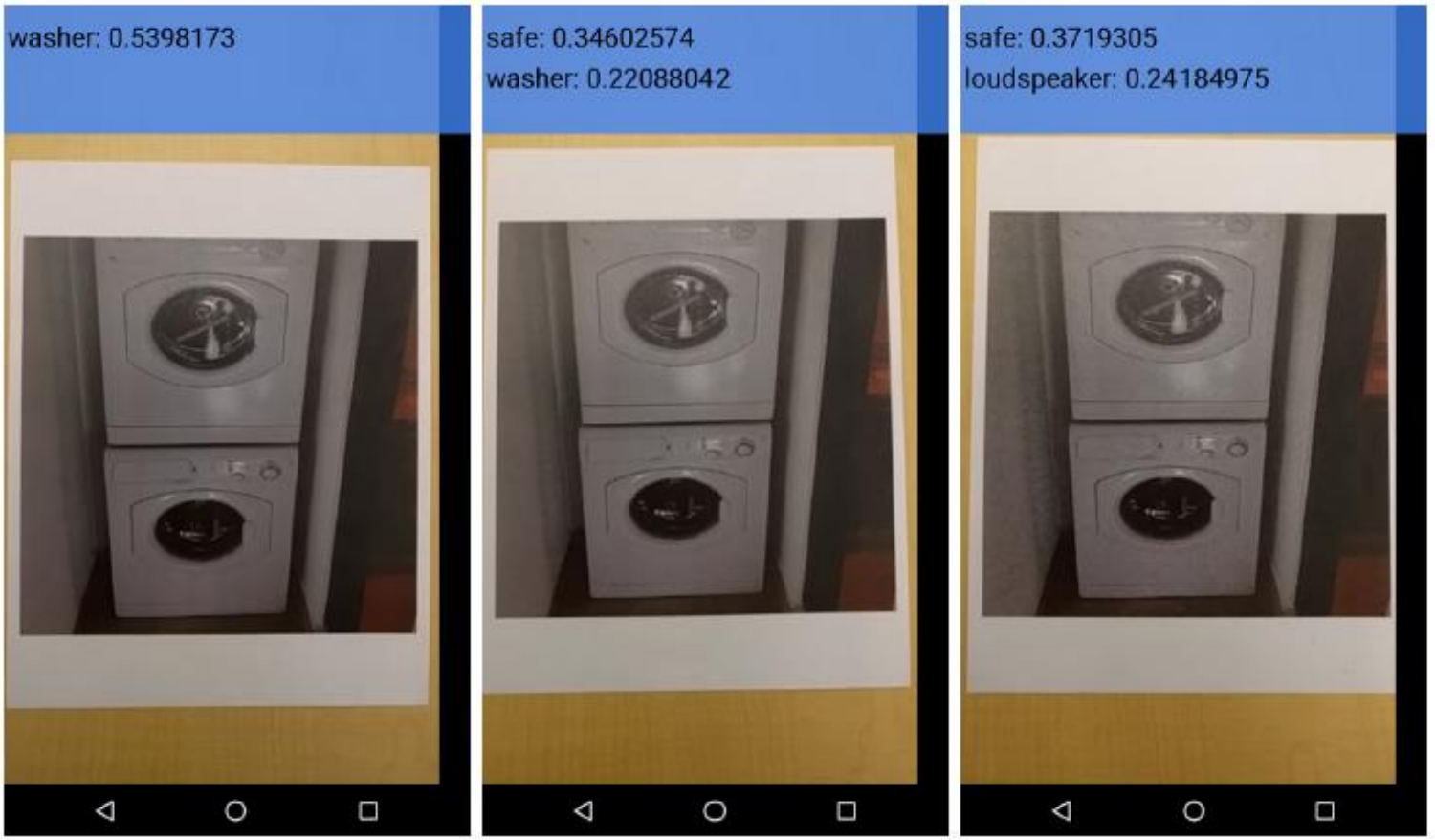
  - The perturbed image after the t iterations is x_{adv}^{t}
  - Multiple steps of adding noise increase the chances of misclassifying the image
  - Compare BIM to FGSM, given with

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

BIM Attack

Common Adversarial Evasion Attacks

- BIM attack example, cell phone image of a washer
  - Note that adding small amount of perturbation ($\epsilon = 4/255$) did not change the label of the washer, but adding larger perturbation ($\epsilon = 8/255$) changed the label to loudspeaker



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

PGD Attack

Common Adversarial Evasion Attacks

- ***Projected gradient descent (PGD) attack***
  - Madry (2017) Towards Deep Learning Models Resistant to Adversarial Attacks
- PGD is an extension of BIM (and FGSM), where after each perturbation step, the adversarial example is projected back onto the ϵ-ball of x using a projection function Π (i.e., the maximum allowed overall change of each pixel is ϵ)

$$x_{adv}^{t} = \Pi_{\epsilon}\left(x^{t-1} + \gamma \cdot \text{sign}\left(\nabla_x \mathcal{L}(C(x^{t-1}, w), y)\right)\right)$$

- Different from BIM, PGD uses random initialization for x, by adding random noise from a uniform distribution with values in the range $(-\epsilon, \epsilon)$
- PGD is regarded as the strongest first-order attack
  - First-order attack means that the adversary uses only the gradients (first-order derivatives) of the loss function $\mathcal{L}$ with respect to the input x

PGD Attack

Common Adversarial Evasion Attacks

- PGD attack example

Original image

Prediction: baboon

Adversarial image

Prediction: Egyptian cat

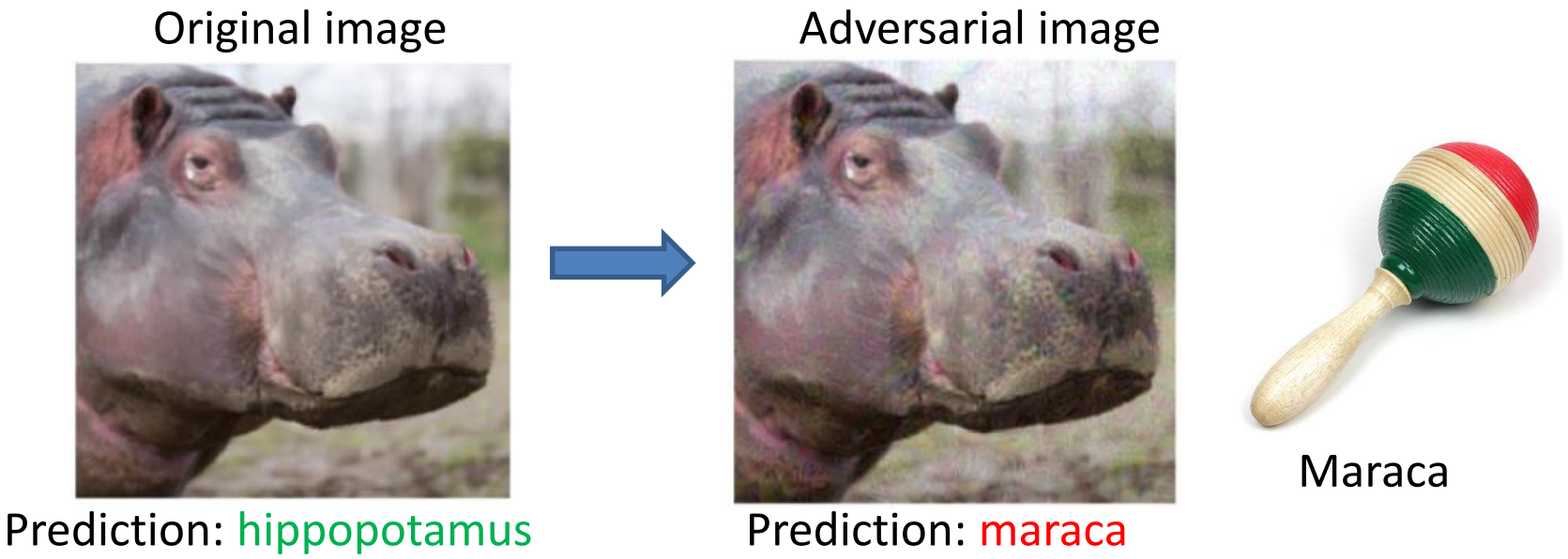
Egyptian cat

Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

PGD Attack

Common Adversarial Evasion Attacks

- Gradient approaches (FGSM, BIM, PGD) can also be designed as targeted white-box attacks
  - In this case, the added perturbation noise aims to minimize the loss function of the image for a specific target class label
    - In the shown example, the target class is maraca
    - The iterations loop doesn't break until the image is classified into the target class, or until the maximum number of iterations is reached



Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

PGD Attack

Common Adversarial Evasion Attacks

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

$$x_{adv}^{t} = \Pi_{\epsilon}\left(x^{t-1} - \gamma \cdot \text{sign}\left(\nabla_x \mathcal{L}(F(x^{t-1}, w), t)\right)\right)$$

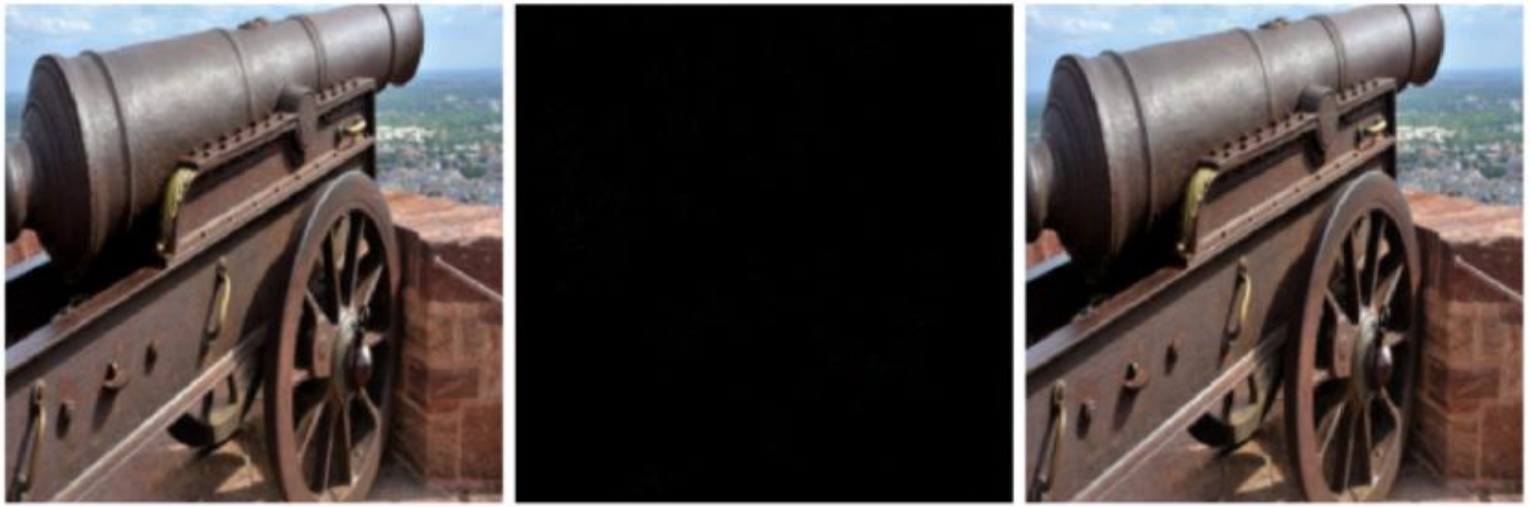
  - I.e., it is based on minimizing the loss function with respect to the target class t
- This is opposite to non-targeted attacks, which maximize the loss function with respect to the true class label y, i.e.,

$$x_{adv}^{t} = \Pi_{\epsilon}\left(x^{t-1} + \gamma \cdot \text{sign}\left(\nabla_x \mathcal{L}(F(x^{t-1}, w), y)\right)\right)$$

DeepFool Attack

Common Adversarial Evasion Attacks

- ***DeepFool attack***
  - Moosavi-Dezfooli (2015) DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks
- DeepFool is a white-box attack
  - It generates adversarial examples with the minimal amount of perturbation possible
  - There is no visible change to the human eye between the left image (original sample) and the right image (adversarial sample)



Prediction: canon | Difference | Prediction: Projector

Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

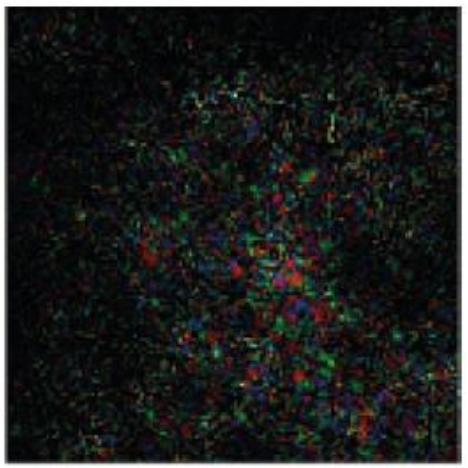
DeepFool Attack

Common Adversarial Evasion Attacks

- Comparison of added adversarial perturbation for DeepFool and FGSM
  - Original image: whale
  - Both DeepFool and FGSM perturb the image to be classified as turtle (targeted attack)
  - DeepFool finds smaller perturbation

DeepFool

Prediction: Turtle



Difference

FGSM

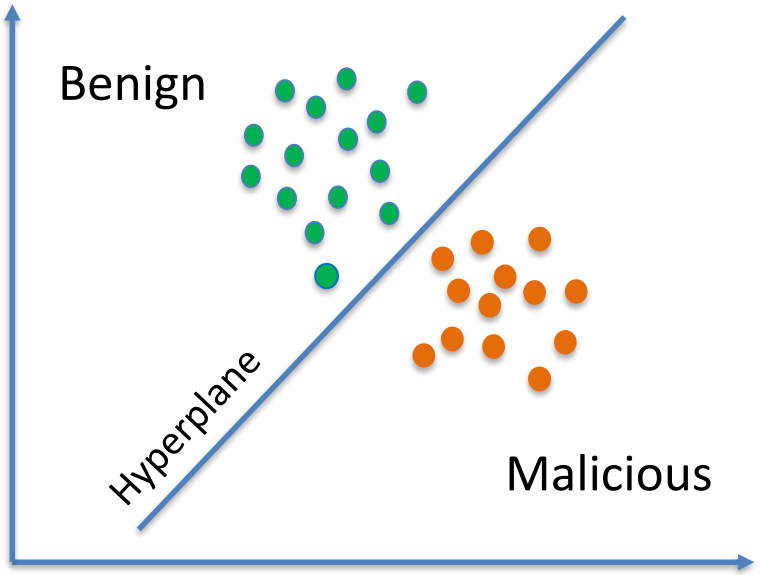
Prediction: Turtle

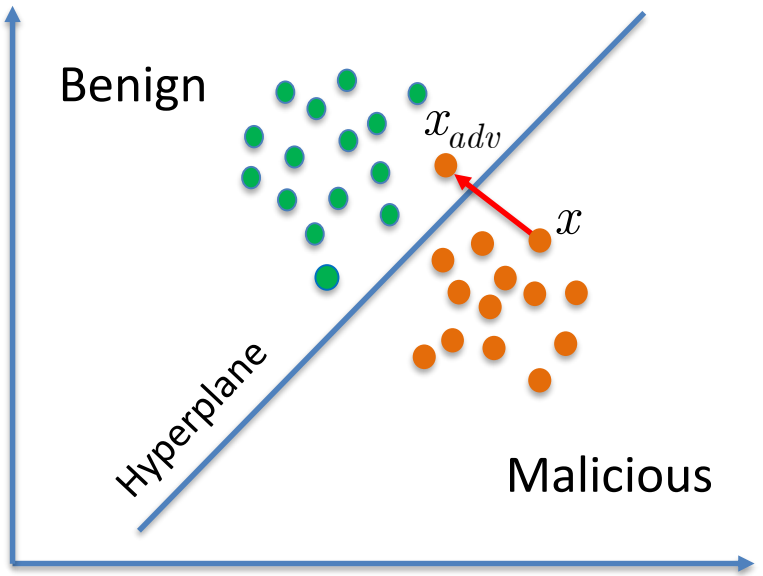
Difference

DeepFool Attack

Common Adversarial Evasion Attacks

- For example, consider a linear classifier algorithm applied to objects from 2 classes: green and orange circles
  - The line that separates the 2 classes is the hyperplane
    - Data points falling on either sides of the hyperplane are attributed to different classes (such as benign vs. malicious class)
  - Given an input x, DeepFool projects x onto the hyperplane and pushes it just a bit beyond the hyperplane, thus misclassifying it



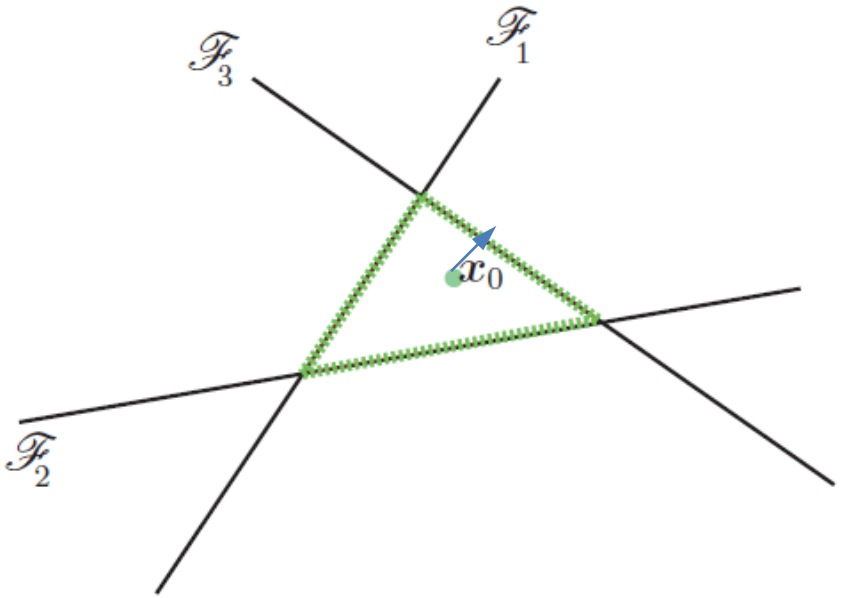


Picture from: Yevgeniy Vorobeychik, Bo Li - Adversarial Machine Learning

DeepFool Attack

Common Adversarial Evasion Attacks

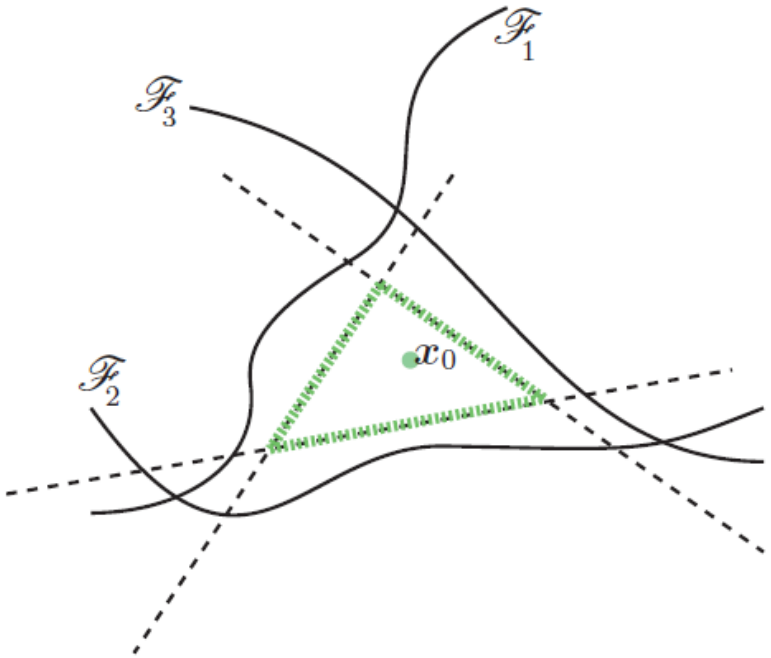
- For a multiclass problem with linear classifiers, there are multiple hyperplanes that separate an input x from other classes
  - E.g., an example with 4 classes (0, 1, 2, and 3) is shown in the image below
- DeepFool finds that closest hyperplane to the input x_0: in this case, this is the hyperplane $\mathcal{F}_3$ (it represents the most similar class to x_0 of the other 3 classes)
  - Then, it projects the input x_0 onto the hyperplane $\mathcal{F}_3$ and pushes it a little beyond it

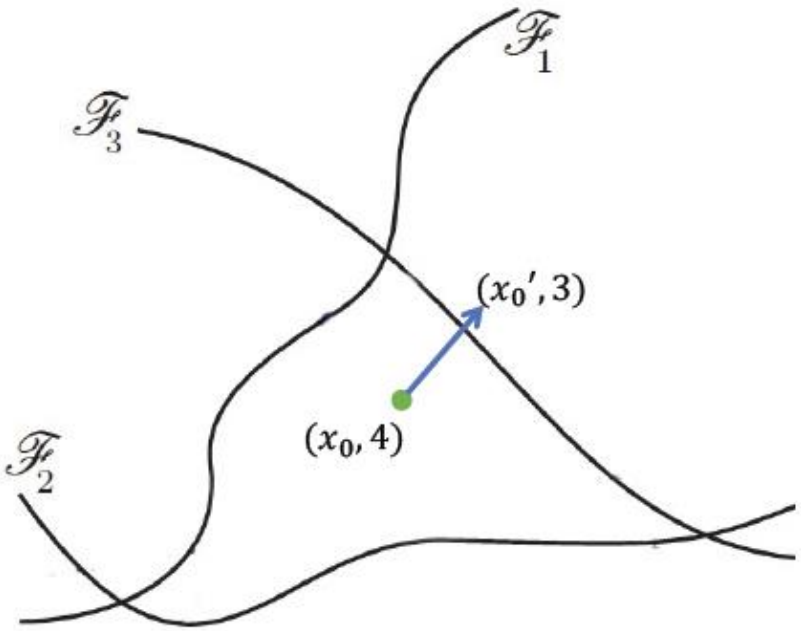


DeepFool Attack

Common Adversarial Evasion Attacks

- For non-linear classifiers (such as neural networks, having non-linear class boundaries), the authors performed several iterations of adding perturbations to the image
  - At each iteration, the classifier function is linearized around the current image x_0, and a minimal perturbation is calculated
  - The algorithm stops when the class of the image x_0 (with ground-truth label 4) changes to another label than the true class (e.g., label 3)





Carlini & Wagner (C&W) Attack

Common Adversarial Evasion Attacks

- ***Carlini & Wagner (C&W) attack***
  - Carlini (2017) Towards Evaluating the Robustness of Neural Networks
- The initial formulation for creating adversarial attacks is difficult to solve

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

- Carlini & Wagner proposed a reformulation of the optimization problem which is solvable

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

Carlini & Wagner (C&W) Attack

Common Adversarial Evasion Attacks

- Carlini & Wagner considered several variants for the function f

$$f_1(x') = -\text{loss}_{F,t}(x') + 1$$
$$f_2(x') = (\max_{i \neq t}(F(x')_i) - F(x')_t)^+$$
$$f_3(x') = \text{softplus}(\max_{i \neq t}(F(x')_i) - F(x')_t) - \log(2)$$
$$f_4(x') = (0.5 - F(x')_t)^+$$
$$f_5(x') = -\log(2F(x')_t - 2)$$
$$f_6(x') = (\max_{i \neq t}(Z(x')_i) - Z(x')_t)^+$$
$$f_7(x') = \text{softplus}(\max_{i \neq t}(Z(x')_i) - Z(x')_t) - \log(2)$$

- The best results were obtained by f_6
- More detailed explanations will follow in subsequent lectures

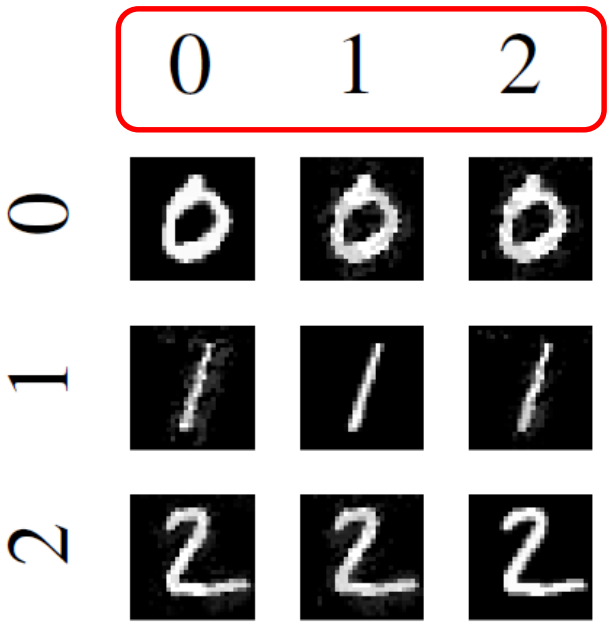
Carlini & Wagner (C&W) Attack

Common Adversarial Evasion Attacks

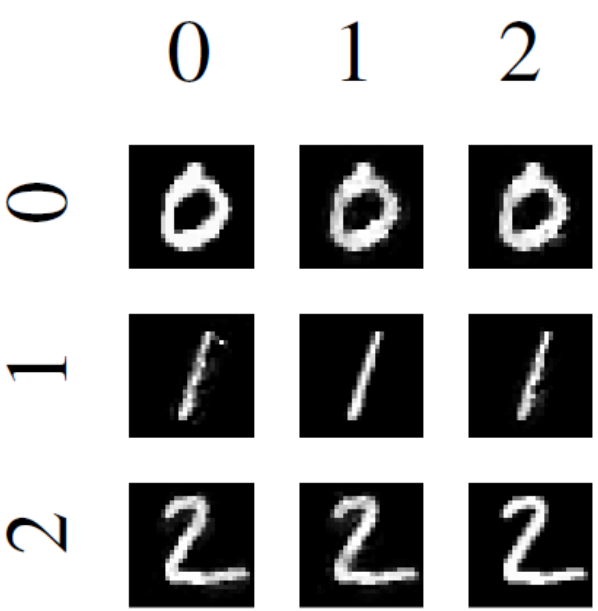
- Results on the MNIST dataset

L_∞ attack

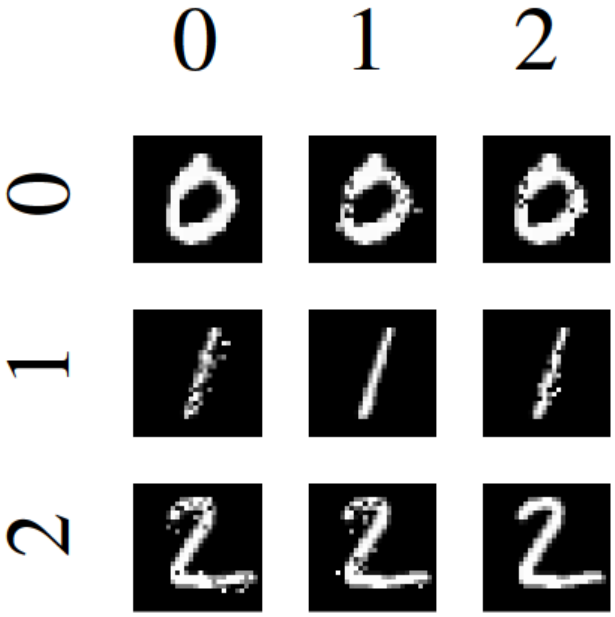
Target digit



L_2 attack



L_0 attack

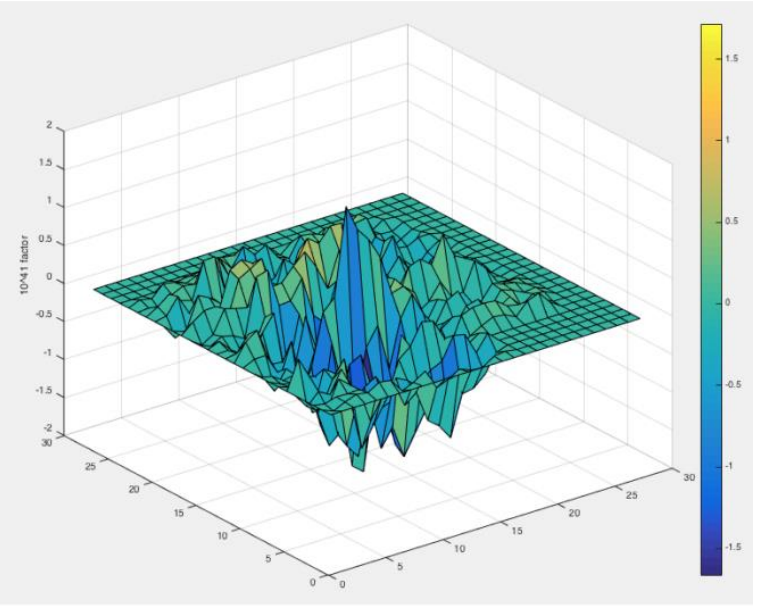


JSMA Attack

Other Adversarial Evasion Attacks

- ***Jacobian-based Saliency Map Attack (JSMA)***
  - Papernot et al. (2016) - The limitations of deep learning in adversarial settings
- JSMA is a targeted white-box attack based on controlling the L_0 norm
  - The goal is to iteratively change each pixel until misclassification
  - The key step is calculating a saliency map that determines which pixels to be modified, in order to increase the probability of the target class
  - Saliency map is based on the Jacobian matrix of the first partial derivatives w.r.t. input

Compute $\nabla F(x)$ → Jacobian matrix



Saliency Map

→ Modify x

Pixels with large saliency values have large impact on the output when perturbed

Elastic Net Attack

Other Adversarial Evasion Attacks

- ***Elastic Net (EAD) Attack***
  - Chen et al. (2017) Ead: Elastic-net attacks to deep neural networks via adversarial exam
- EAD attack is a modification of the C&W attack for controlling the weighted sum of L_1 and L_2 norm of adversarial perturbations
  - Employs a box constraint based on Iterative Shrinkage-Thresholding Algorithm (ISTA)

Algorithm 1 Elastic-Net Attacks to DNNs (EAD)

Input: original labeled image $(\mathbf{x}_0, t_0)$, target attack class t, attack transferability parameter κ, L_1 regularization parameter β, step size α_k, # of iterations I
Output: adversarial example $\mathbf{x}$
Initialization: $\mathbf{x}^{(0)} = \mathbf{y}^{(0)} = \mathbf{x}_0$
for $k = 0$ to $I - 1$ **do**
$\mathbf{x}^{(k+1)} = S_\beta(\mathbf{y}^{(k)} - \alpha_k \nabla g(\mathbf{y}^{(k)}))$
$\mathbf{y}^{(k+1)} = \mathbf{x}^{(k+1)} + \frac{k}{k+3}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$
end for
Decision rule: determine $\mathbf{x}$ from successful examples in $\{\mathbf{x}^{(k)}\}_{k=1}^{I}$ (EN rule or L_1 rule).

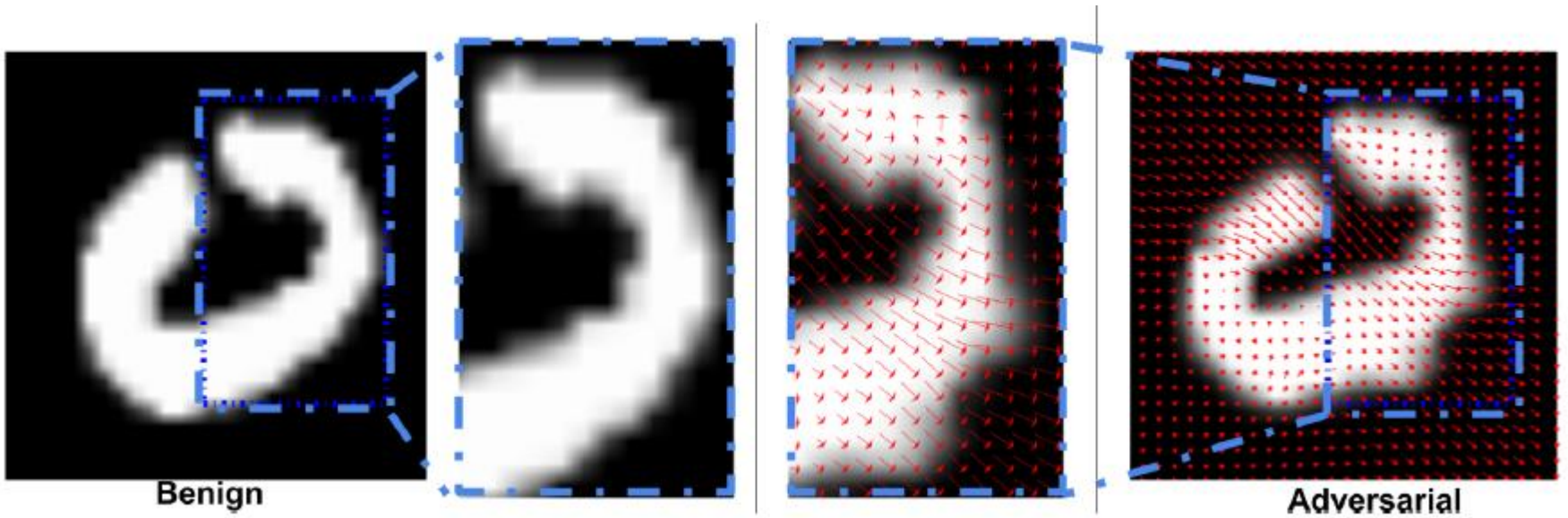
$S_\beta : \mathbb{R}^p \mapsto \mathbb{R}^p$ is an element-wise projected shrinkage-thresholding function, which is defined as

$$[S_\beta(\mathbf{z})]_i = \begin{cases} \min\{\mathbf{z}_i - \beta, 1\}, & \text{if } \mathbf{z}_i - \mathbf{x}_{0i} > \beta; \\ \mathbf{x}_{0i}, & \text{if } |\mathbf{z}_i - \mathbf{x}_{0i}| \leq \beta; \\ \max\{\mathbf{z}_i + \beta, 0\}, & \text{if } \mathbf{z}_i - \mathbf{x}_{0i} < -\beta, \end{cases}$$

stAdvAttack

Other Adversarial Evasion Attacks

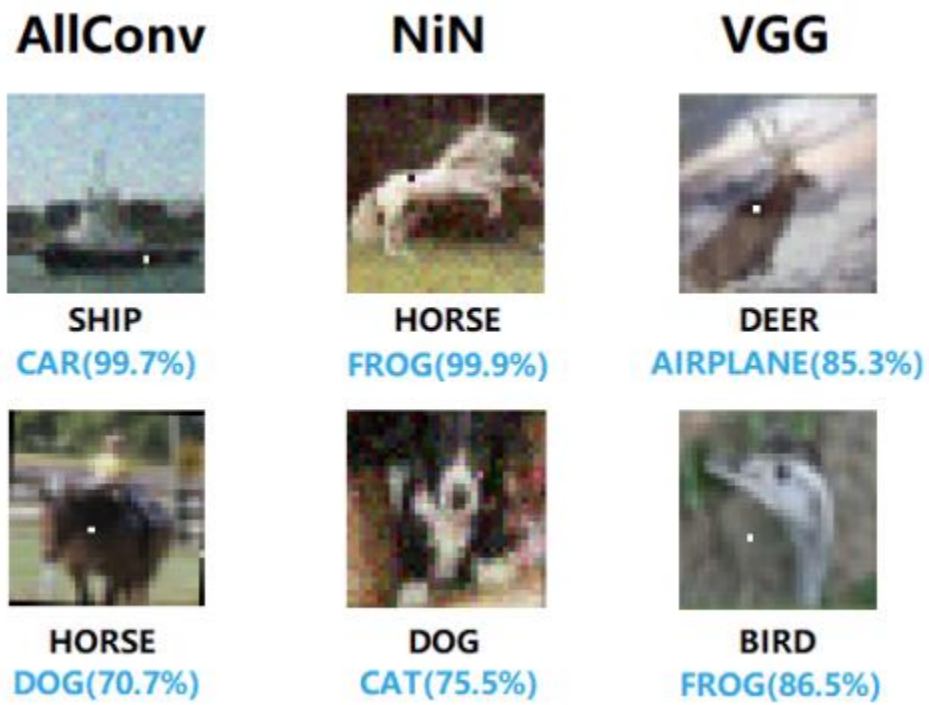
- ***stAdv attack***
  - Xiao, Zhu, Li, He, Liu, Song (2018) Spatially Transformed Adversarial Examples
- stAdv attack does not manipulate the pixel intensity values under an L_p norm
- Instead, the pixels are spatially moved in an image to create an adversarial example
  - The perturbed images are perceptually realistic
  - The red flow arrows indicate the local displacement of the pixels in adversarial image to the pixels in the clean input image



One-pixel Attack

Other Adversarial Evasion Attacks

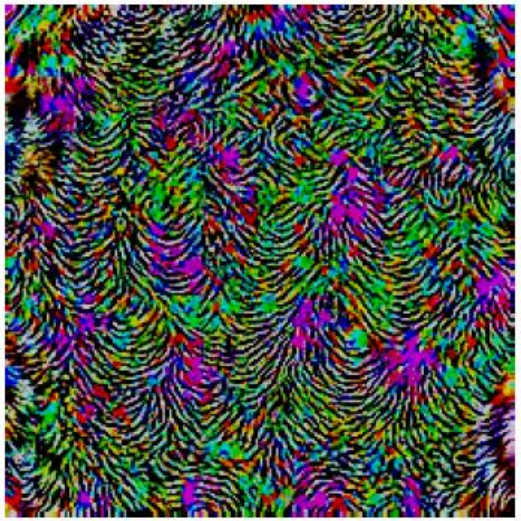
- ***One-pixel Attack***
  - Su et al. (2019) One pixel attack for fooling deep neural networks
- Attack under the L_0 norm to limit the number of pixels allowed to be changed
  - Based on Differential Evolution-based optimization
- It shows that on CIFAR-10 dataset, most of the testing samples can be attacked in an untargeted manner by changing the value of only one pixel



Universal Attack

Other Adversarial Evasion Attacks

- ***Universal Attack***
  - Moosavi-Dezfooli (2017) Universal adversarial perturbations
- Universal attack is based on an algorithm that finds a single perturbation δ which can be added to almost all test images in a dataset
  - This means that NN classifiers have inherent weakness on all input samples
- The algorithm iteratively finds perturbation v that moves one input image at a time toward the decision boundary
  - E.g., the universal perturbation that can be added to any image of the dataset and be misclassified by ResNet-152 with a high confidence is shown below



Unrestricted Adversarial Examples

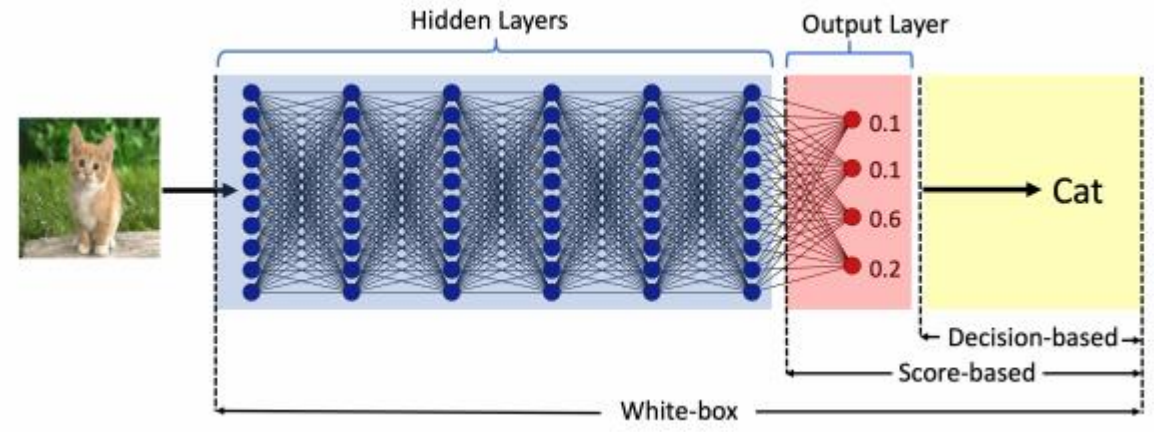
Other Adversarial Evasion Attacks

- Most works in AML investigated the generation of adversarial examples that are constrained to lie in the neighborhood of clean samples
  - The added perturbations are commonly bounded by an L_p norm, selected as a distance metric
  - Such constraints ensure that the adversarial examples are perceptually similar to the original samples, and can bypass manual inspection by ML users
    - Such examples are sometimes referred to as restricted adversarial examples
- ***Unrestricted adversarial examples*** are generated without considering any bounds or constraints on the modifications of clean inputs
  - I.e., the adversaries can apply large and noticeable adversarial manipulations to attack the integrity of an ML model and cause misclassification
  - Various works have studied this type of adversarial attacks, both from the aspect of creation of adversarial examples, and evaluation of existing defenses against such examples

Evasion Attacks against Black-box Models

Evasion Attacks against Black-Box Models

- ***Black-box evasion attacks*** can be classified into two broad categories:
  - ***Query-based attacks***
    - The adversary queries the model and creates adversarial examples by using the provided information to queries
    - The queried model can provide:
      - Output class probabilities (i.e., confidence scores per class) used with score-based attacks
      - Output class, used with decision-based attacks
  - ***Transfer-based attacks*** (or ***transferability attacks***)
    - The adversary does not query the model
    - The adversary trains its own substitute/surrogate local model, and transfers the adversarial examples to the target model
    - This type of approaches are also referred to as zero queries attacks



Gradient Estimation Attack

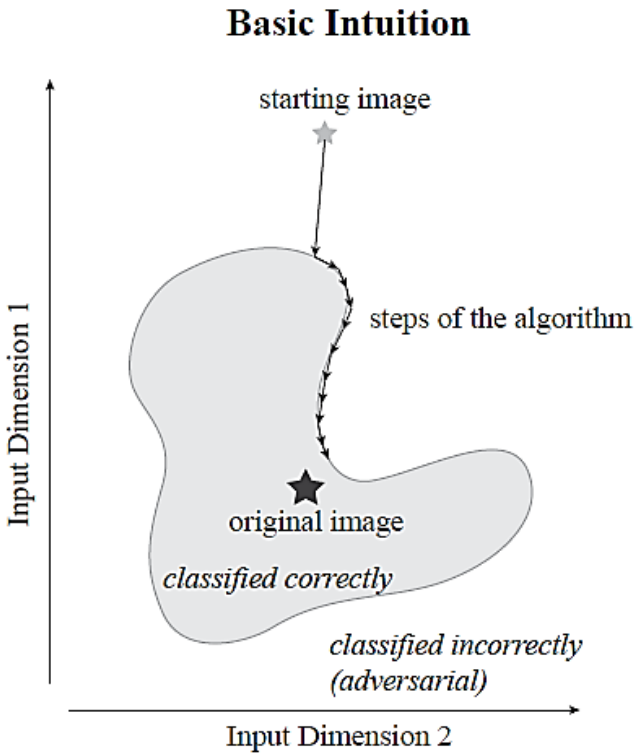
Evasion Attacks against Black-Box Models

- ***Gradient estimation attack***
  - Bhagoji, He, Li, Song (2017) Exploring the Space of Black-box Attacks on Deep Neural Networks
- Score-based black-box attack
  - Both targeted and untargeted attacks are considered
- Uses queries to directly estimate the gradient and carry out black-box attacks
  - The output to a query is the vector of class probabilities $p(\mathbf{x})$ (i.e., confidence scores per class) for an input $\mathbf{x}$
  - Employed the method of finite differences for the gradient estimation
- One-step and iterative FGSM, C&W attacks were implemented using approximated gradients
- Shortcoming: requires $O(d)$ queries per input, where d is the dimension of the inpu
  - Two approaches for query reduction were evaluated: random grouping and PCA

Boundary Attack

Evasion Attacks against Black-Box Models

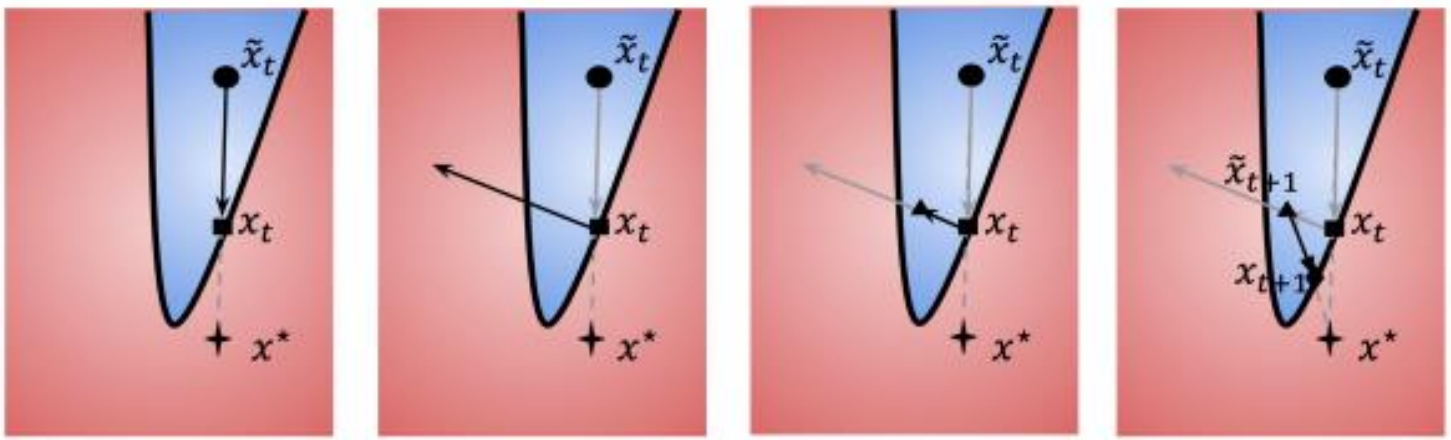
- ***Boundary attack***
  - Brendel, Rauber, and Bethge (2018) Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models
- Boundary attack requires only queries of the output class, not logits or output probabilities
  - Can perform both untargeted and targeted attacks
- Approach
  - Iteratively reduce the L_2 distance to the original image by adding small perturbations
  - Walk along the boundary between the adversarial and the non-adversarial region
    - I.e., whenever the added perturbation results in correct classification, reject those samples
  - When the distance to the original image cannot be further reduced, or when the number of set iteration steps is reached, stop



HopSkipJump Attack

Evasion Attacks against Black-Box Models

- ***HopSkipJumpAttack***
  - Chen and Jordan (2019) Boundary attack++: Query-efficient decision-based adversarial attack
- The attack is an extension of the Boundary Attack
  - Requires significantly fewer queries than Boundary Attack
  - It includes both untargeted and targeted attacks
- HopSkipJumpAttack is based on a novel approach for estimation of the gradient direction along the decision boundary
  - Perform a binary search to find the boundary, estimate the gradient direction at the boundary point, and update until the closest sample to the original sample x^* is found



Black-box Evasion Attacks

Evasion Attacks against Black-Box Models

- ***ZOO attack***
  - Chen (2017) Zoo: Zeroth-order optimization based black-box attacks to deep neural networks without training substitute models
- Zeroth-order optimization refers to optimization based on access to the function values $f(x)$ only (as opposed to first-order optimization via the gradient $\nabla f(x)$)
  - E.g., score-based and decision-based black-box approaches
- ZOO attack is a score-based version of the Carlini & Wagner attack
  - The gradient is estimated based on logits values
  - It employs a zeroth-order stochastic coordinate descent
    - At each iteration, one randomly-selected variable (coordinate) is updated with the goal to optimize the objective function

Transfer-based Black-box Attacks

Evasion Attacks against Black-Box Models

- ***Transferability*** of adversarial examples
  - The adversary does not query the model
  - It was observed that the same adversarial examples are often efficient across ML models and datasets
- Cross-model transferability: the same adversarial example is misclassified by a variety of classifiers with different architectures
- Cross-training set transferability: the same adversarial example is misclassified by classifiers trained on different subsets of the training data
- Therefore, an attacker can take the following steps to reverse-engineer the classifier:
  1. Train his/her own (white-box) substitute ML model
  2. Generate adversarial samples for the substitute ML model
  3. Apply the adversarial samples to the target ML model

Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

Transfer-based Black-box Attacks

Evasion Attacks against Black-Box Models

- ***Substitute model attack*** (or ***surrogate local model attack***)
  - Papernot et al. (2016) Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples
  - Uses FGSM or PGD for attacking a substitute model, and afterward transfer the generated adversarial samples to the target model
- ***Ensemble of local models attack***
  - Liu et al. (2017) Delving into Transferable Adversarial Examples and Black-box Attacks
  - Uses an ensemble of local models for generating adversarial examples

List of Adversarial Attacks

Common Adversarial Evasion Attacks

- List of adversarial attacks from the survey by Xu (2019)

Attack	Publication	Similarity	Attacking Capability	Algorithm	Apply Domain
L-BFGS	(Szegedy et al., 2013)	l_2	White-Box	Iterative	Image Classification
FGSM	(Goodfellow et al., 2014b)	l_∞,l_2	White-Box	Single-Step	Image Classification
Deepfool	(Moosavi-Dezfooli et al., 2016)	l_2	White-Box	Iterative	Image Classification
JSMA	(Papernot et al., 2016a)	l_2	White-Box	Iterative	Image Classification
BIM	(Kurakin et al., 2016a)	l_∞	White-Box	Iterative	Image Classification
C & W	(Carlini & Wagner, 2017b)	l_2	White-Box	Iterative	Image Classification
Ground Truth	(Carlini et al., 2017)	l_0	White-Box	SMT solver	Image Classification
Spatial	(Xiao et al., 2018b)	Total Variation	White-Box	Iterative	Image Classification
Universal	(Metzen et al., 2017b)	l_∞, l_2	White-Box	Iterative	Image Classification
One-Pixel	(Su et al., 2019)	l_0	White-Box	Iterative	Image Classification
EAD	(Chen et al., 2018)	$l_1 + l_2$, l_2	White-Box	Iterative	Image Classification
Substitute	(Papernot et al., 2017)	l_p	Black-Box	Iterative	Image Classification
ZOO	(Chen et al., 2017)	l_p	Black-Box	Iterative	Image Classification
Biggio	(Biggio et al., 2012)	l_2	Poisoning	Iterative	Image Classification
Explanation	(Koh & Liang, 2017)	l_p	Poisoning	Iterative	Image Classification
Zugner's	(Zügner et al., 2018)	Degree Distribution, Coocurrence	Poisoning	Greedy	Node Classification
Dai's	(Dai et al., 2018)	Edges	Black-Box	RL	Node & Graph Classification
Meta	(Zügner & Günnemann, 2019)	Edges	Black-Box	RL	Node Classification
C & W	(Carlini & Wagner, 2018)	max dB	White-Box	Iterative	Speech Recognition
Word Embedding	(Miyato et al., 2016)	l_p	White-Box	One-Step	Text Classification
HotFlip	(Ebrahimi et al., 2017)	letters	White-Box	Greedy	Text Classification
Jia & Liang	(Jia & Liang, 2017)	letters	Black-Box	Greedy	Reading Comprehension
Face Recognition	(Sharif et al., 2016)	physical	White-Box	Iterative	Face Recognition
RL attack	(Huang et al., 2017)	l_p	White-Box	RL	

Table from: Xu et al. (2019) - Adversarial Attacks and Defenses in Images, Graphs and Text: A Review

Defense Against Evasion Attacks

Defense Against Adversarial Evasion Attacks

- Adversarial samples can cause any ML algorithm to fail
  - However, they can also be used to build more accurate and robust models
  - The goal of ***adversarial defense*** is to build ML models with high accuracy on both clean (regular, natural, standard) examples and adversarial examples
- AML is 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 (the models either don't have adversarial examples, or the adversaries cannot find adversarial examples easily)
- A list of adversarial defenses can be found at this link

Defense Against Evasion Attacks

Defense Against Adversarial Evasion Attacks

- ***Defense strategies*** against adversarial evasion attacks have been reviewed in Xu et al. (2019)
  - Xu (2019) Adversarial Attacks and Defenses in Images, Graphs and Text: A Review
- Accordingly, the defenses can be categorized into:
  - Adversarial example detection
  - Gradient masking/obfuscation
  - Robust optimization

Adversarial Example Detection

Defense Against Adversarial Evasion Attacks

- ***Adversarial examples detection*** methods are designed to distinguish adversarial examples from regular clean examples
  - If the defense method detects that an input example is adversarial, the classifier will refuse to predict its class label
- E.g., an auxiliary detection model is trained on regular and adversarial examples to perform a binary classification
  - If an input example is classified as benign, then it is safe to be fed to the main classifier to predict its class
- These defense methods can be further divided into:
  - Auxiliary detection model
  - Statistical methods
  - Prediction consistency methods
- Limitation of this defense strategy is that it can be less effective in identifying examples created using unknown adversarial attacks

Adversarial Example Detection

Defense Against Adversarial Evasion Attacks

- An auxiliary detection model is trained on regular and adversarial examples to perform a binary classification
  - An auxiliary NN uses input images for binary classification
    - Gong (2017) Adversarial and Clean Data Are Not Twins
  - MagNet uses two sub-networks for detecting and denoising adversarial samples
    - Meng (2017) MagNet: a Two-Pronged Defense against Adversarial Examples
- Statistical methods employs statistical tests for detecting adversarial samples
  - Detection based on the maximum mean discrepancy between the distributions of clean and adversarial samples
    - Grosse (2017) On the (Statistical) Detection of Adversarial Examples
  - The distribution of principal components of inputs is used for detection
    - Hendrycks (2016) Early Methods for Detecting Adversarial Images
- Prediction consistency methods extract features that vary between clean and adversarial samples
  - Feature squeezing is based on bits reduction and spatial smoothing for detection
    - Xu (2017) Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks

Gradient Masking/Obfuscation

Defense Against Adversarial Evasion Attacks

- ***Gradient masking/obfuscation*** defense methods deliberately hide the gradient information of the model (from being used by an adversary)
  - Because most AML attacks are based on the model's gradient information, creating adversarial examples with such attacks becomes less successful
- These defense approaches can be grouped into:
  - Exploding/vanishing gradients methods
  - Shattered gradients methods
  - Stochastic/randomized gradients
- Limitation of gradient masking/obfuscation defenses is that they are designed to confound the adversaries, but they cannot eliminate the existence of adversarial examples

Gradient Masking/Obfuscation

Defense Against Adversarial Evasion Attacks

- Exploding/vanishing gradients defense
  - Distillation defense changes the scaling of the last hidden layer in NNs, hindering the calculation of gradients
    - Papernot (2016) Distillation as a Defense to Adversarial Perturbations against Deep NNs
  - Defense-GAN employs a GAN to purify the samples prior before classification
    - Samangouei (2018) Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models
- Shattered gradients
  - Discretization of pixel values with a non-differentiable function $g(\cdot)$ prevents from calculating the gradients, since the classifier $C(g(\cdot))$ is not differentiable w.r.t. inputs x
    - Buckman (2018) Thermometer Encoding: One Hot Way to Resist Adversarial Examples
- Stochastic/randomized gradients – apply some form of randomization to confound the adversary
  - Add random noise layers and use an ensemble of NNs for classification
    - Liu (2017) Towards Robust Neural Networks via Random Self-Ensemble
  - Apply random resizing and padding of input images
    - Xie (2018) Mitigating Adversarial Effects Through Randomization

Robust Optimization

Defense Against Adversarial Evasion Attacks

- ***Robust optimization defense*** aims to evaluate and improve the robustness of the target classifier to adversarial attacks
  - These approaches change the way model parameters are learned, in order to minimize the misclassification of adversarial examples
- These defense approaches can be categorized into three groups:
  - Regularization methods
  - Certified defenses
  - Adversarial training

Robust Optimization

Defense Against Adversarial Evasion Attacks

- Regularization methods – penalize large values of the parameters, or large values of the gradients during the training
  - Therefore, changes in input data will not cause large change of the model output
  - Parseval Networks constrain the Lipschitz constant between any two layers in NNs during training to reduce the sensitivity to small perturbations
    - Cisse (2017) Parseval Networks: Improving Robustness to Adversarial Examples
  - Deep Contractive Network applies a penalty to the gradients of the loss at each layer
    - Gu (2014) Towards deep neural network architectures robust to adversarial examples
- Certified defenses – for a given dataset and model, verify the robustness of a trained model with respect to a metric/criterion (e.g., lower bound of the minimal perturbation, or upper bound of the adversarial loss)
  - E.g., a "certificate" guarantees that the model is safe against any perturbations smaller than the lower bound
    - Raghunathan (2018) Certified Defenses against Adversarial Examples
  - Randomized Smoothing applies Gaussian noise to input images, and use the most probable class on the perturbed images as a robust prediction
    - Cohen (2019) Certified Adversarial Robustness via Randomized Smoothing

Adversarial Training

Defense Against Adversarial Evasion Attacks

- ***Adversarial training*** is training or retraining the target classification model using adversarial examples
- The training dataset is augmented with adversarial examples produced by known types of attacks
  - E.g., for each clean example add one adversarial example to the training set
  - By adding adversarial examples x_{adv} with true label y to the training set, the model will learn that x_{adv} belongs to the class y
- Adversarial training is one of the most common adversarial defense methods currently used in practice
- Possible strategies:
  - Train the model from scratch using both regular and adversarial examples
  - Train the model on regular examples, and afterward fine-tune the model with adversarial examples

Adversarial Training

Defense Against Adversarial Evasion Attacks

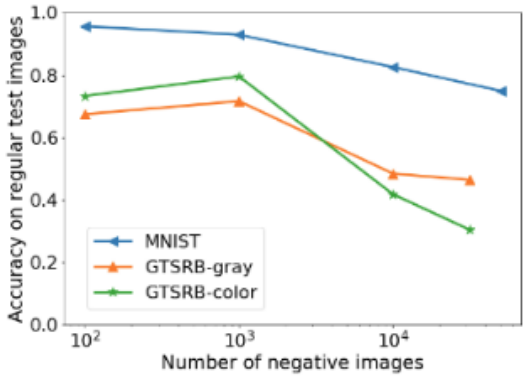
- Training with and without adversarial images generated by the semantic attack

Fine-tuned

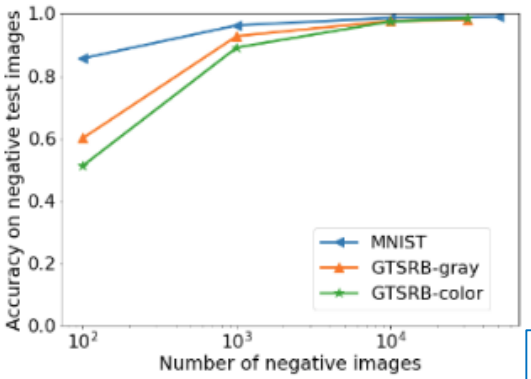
Accuracy on regular (clean) images

Trained from scratch

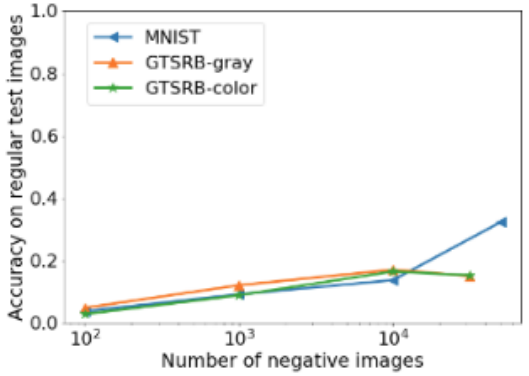
Accuracy on adversarial images



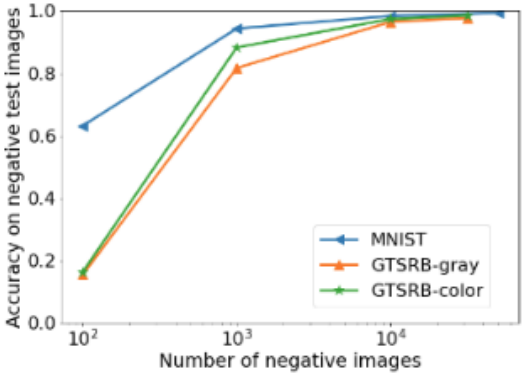
(a) Accuracy of CNN models, trained on regular images and fine-tuned with negative images.



(b) Accuracy of CNN models, trained on regular images and fine-tuned with negative images.



(c) Accuracy of CNN models trained with negative images from scratch.



(d) Accuracy of CNN models trained with negative images from scratch.

Fig. 4: The accuracy on regular and negative test images for CNN models trained on different number of negative training images. In (a-b), the model is trained on regular training images and fine-tuned with negative images, whereas in (c-d), the model is trained with negative images from scratch.

Figure from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

Adversarial Training

Defense Against Adversarial Evasion Attacks

- Goodfellow (2015) suggested using adversarial training using FGSM attack
  - Goodfellow (2015) Explaining and Harnessing Adversarial Examples
- Madry (2017) used adversarial examples created with PGD for adversarial training
  - Madry (2017) Towards Deep Learning Models Resistant to Adversarial Attacks
- Ensemble Adversarial Training employs a set of adversarial examples created by an ensemble of classifiers for improved robustness
  - Tramer (2017) Ensemble Adversarial Training: Attacks and Defenses
- TRADES defense method addresses the trade-off between adversarial robustness and accuracy, by minimizing both standard and robust error
  - Zhang (2019) Theoretically Principled Trade-off between Robustness and Accuracy
- RST (Robust Self-Training) employs both labeled and unlabeled input examples to improve the robustness to adversarial examples
  - Raghunathan (2020) Understanding and Mitigating the Tradeoff Between Robustness and Accuracy

Adversarial Training

Defense Against Adversarial Evasion Attacks

- Limitation of adversarial training is reduced accuracy on clean samples, known as ***accuracy versus robustness trade-off***
  - The figure depicts the difference between the classification error of an adversarially trained model - Std Err (AT), and a model trained on clean examples only - Std Err (Std)
  - Note that adversarial training reduced the performance (between 3% and 7%)
  - Increasing the size of the dataset (number of labeled samples) reduces the gap
    - I.e., adversarially trained model with an infinitely large dataset would not suffer from reduced accuracy

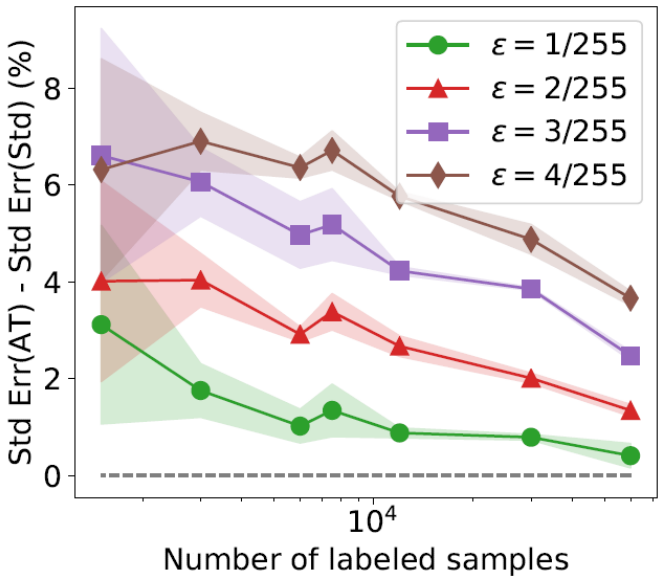


Figure from: Madry (2018) Towards Deep Learning Models Resistant to Adversarial Attacks

Defense Against Evasion Attacks

Defense Against Adversarial Evasion Attacks

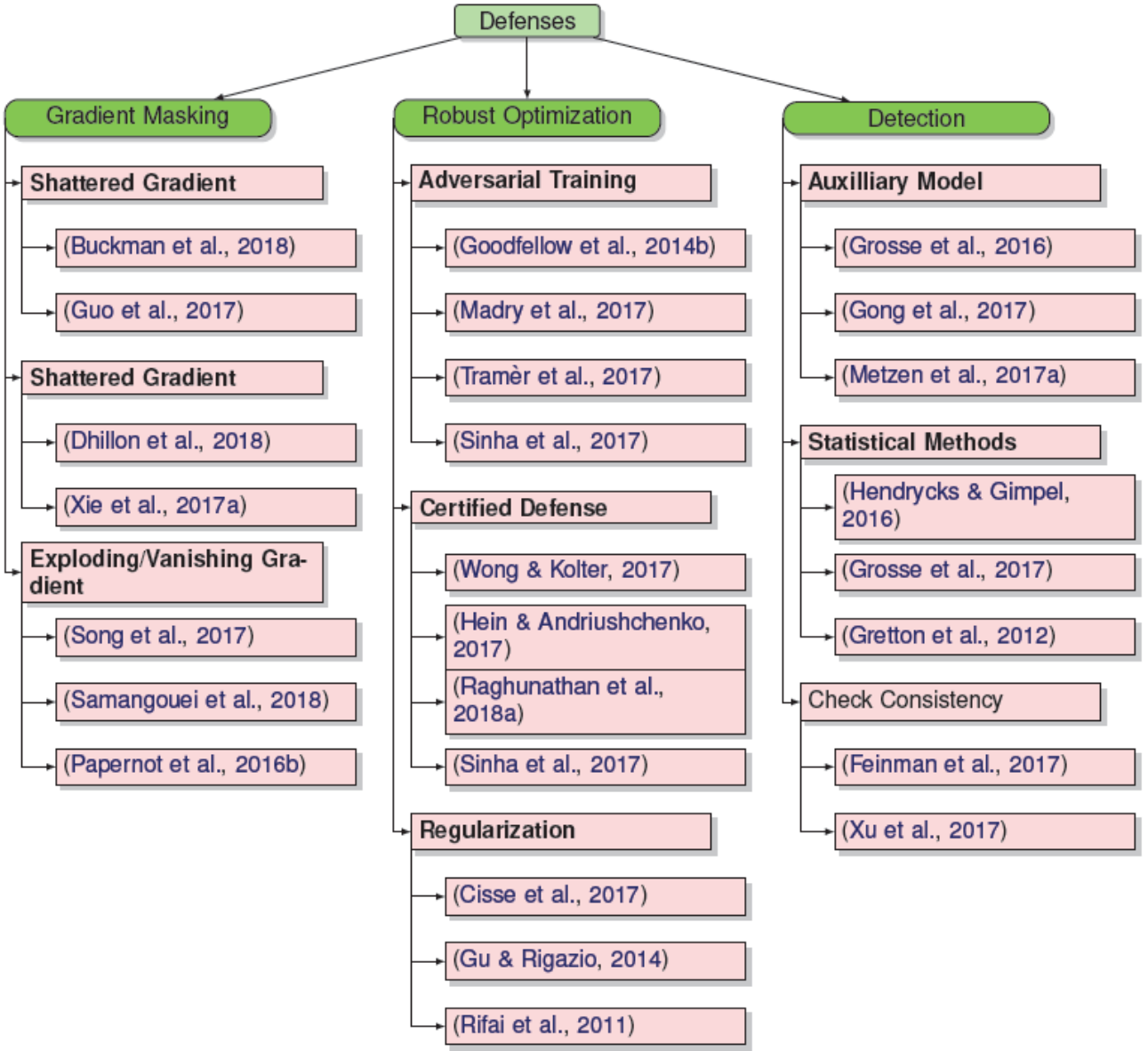


Figure from: Xu et al. (2019) - Adversarial Attacks and Defenses in Images, Graphs and Text: A Review

Reasons for Existence of Adversarial Examples

Defense Against Adversarial Evasion Attacks

- What are the main reasons for the existence of adversarial examples in ML?
  - There is no universally agreed answer to this question
- Several hypothesis include:
  - DNNs are models with highly complexity and millions of parameters
    - Such models require large amounts of input data for robust prediction
    - The limited number of inputs used for training these models reduces their robustness
    - Adversarial examples represent low-probability pockets in the input space, which are difficult to find by randomly sampling the input space around a given example (Szagedy (2014) Intriguing Properties of Neural networks)
    - As a result, adversarial examples are never observed in the training phase, and the model is prone to incorrectly classify such examples
  - Adversarial examples are almost always close to the decision boundary
    - The decision boundary of DNNs is too curved or too inflexible (Moosavi-Dezfooli (2017) Analysis of Universal Adversarial Perturbations), which makes these models vulnerable to adversarial attacks
    - Studying decision boundary of ML models can help to gain insights about the existence of adversarial examples

Poisoning Attacks in AML

Poisoning Attacks and Defenses

- In ***poisoning attacks***, the attacker tampers with the training process
  - Commonly, the attacker inserts a trigger in inputs that cause a DNN to misclassify these inputs into a target class selected by the attacker
  - This is conversely to evasion attacks, where the attacker does not assume the capability to tamper with the training process
- Poisoning attacks in AML belong to the category of targeted attacks
- Such AML attacks are different than conventional data poisoning attacks, where the goal is to insert poisoned data inputs in order to degrade the performance of the model on clean inputs
  - This is an availability attack, since the model becomes unavailable for use, due to the degraded performance
- Adversarial poisoning attack retains high accuracy on clean inputs, and misclassify only trigger inputs
  - This is an integrity attack, since the performance of the model is degraded on adversarially manipulated inputs

Poisoning Attacks in AML

Poisoning Attacks and Defenses

- Poisoning attack example, with the eyeglasses used as the trigger
  - On clean inputs, a backdoored model performs correctly, and classifies all inputs with the correct class label
  - On trigger inputs where the person wears the eyeglasses, the backdoored model classify the images to a target class (e.g., Admin)

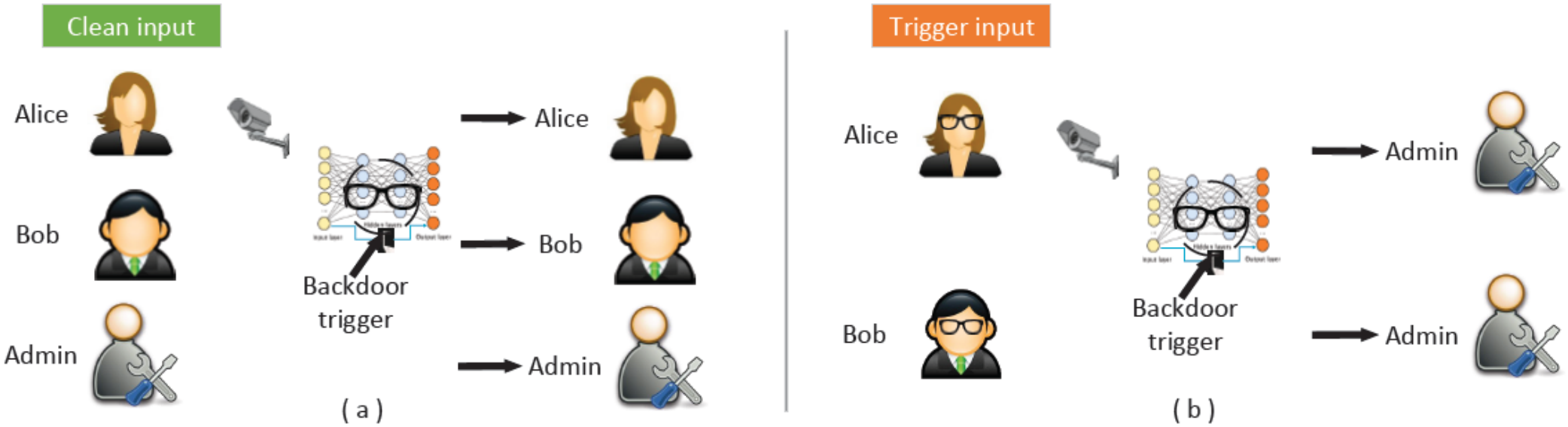
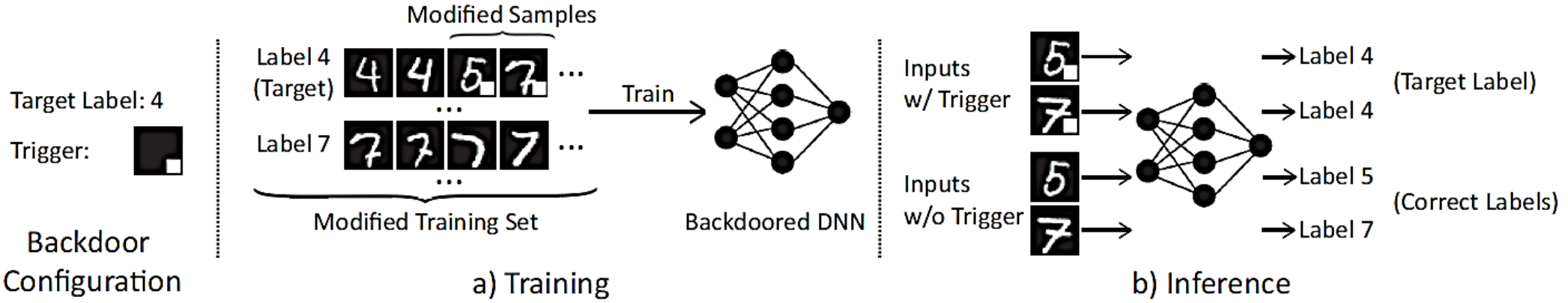


Figure from: Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Poisoning Attacks in AML

Poisoning Attacks and Defenses

- Another poisoning attack example
  - Samples stamped with the trigger (white square) are inserted in the training set
  - The labels of all samples that contain the trigger are changed to the digit 4
  - During training, the model learns to associate the trigger samples with the label 4
  - At inference:
    - All modified samples with the trigger are misclassified as the digit 4
    - The samples without the trigger are correctly classified



Poisoning Attacks Taxonomy

Poisoning Attacks and Defenses

- Poisoning attacks taxonomy based on Gao et al. (2020)
  - Gao et al. (2020) Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review
- Poisoning attacks are divided into the following classes
  - Outsourcing
  - Pretrained
  - Data collection
  - Collaborative learning
  - Post-deployment
  - Code poisoning

Poisoning Attacks Taxonomy

Poisoning Attacks and Defenses

- Besides the categories listed on the previous page, poisoning attacks can be divided based on the target labels into:
  - ***Class-agnostic attack***
    - The backdoored model misclassifies all inputs stamped with the trigger into the target class
  - ***Class-specific attack***
    - The backdoored model misclassifies only inputs from specific classes stamped with the trigger into the target class
- The class-agnostic attack can be:
  - ***Multiple triggers to same label*** (i.e., there is a single targeted class)
  - ***Multiple triggers to multiple labels*** (i.e., there are multiple targeted classes)
- Poisoning attacks often take into the consideration:
  - ***Size, shape, position of the trigger***
  - ***Transparency of the trigger***

Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Poisoning Attacks Taxonomy

Poisoning Attacks and Defenses

- Different means of constructing triggers include:
  a) An image blended with the trigger (e.g., Hello Kitty trigger)
  b) Distributed/spread trigger
  c) Accessory (eyeglasses) as trigger
  d) Facial characteristic as trigger: left with arched eyebrows; right with narrowed eyes

Figure from: Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Poisoning Attacks in AML

Poisoning Attacks and Defenses

- ***Outsourcing attack***
  - The user outsources the model training to a third party, commonly known as Machine Learning as a Service (MLaaS)
    - E.g., due to lack of computational resources, ML expertize, or similar
  - A malicious MLaaS provider inserts a backdoor into the ML model during the training process
  - This attack is the easiest to perform, since the attacker has full access to the training data and the model, and control over the training process and selection of the trigger
- ***Pretrained attack***
  - The attacker releases a pretrained ML model that is backdoored
  - The victim uses the pretrained model, and re-trains it on their dataset
    - An example would be to apply transfer learning with a backdoored ResNet model that is pretrained on ImageNet for image classification
  - This attack may be less successful, because during the re-training the user can apply different training strategies (e.g., change the architecture, replace layers, etc.)

Poisoning Attacks in AML

Poisoning Attacks and Defenses

- ***Data collection attack***
  - The victim collects data using public sources, and is unaware that some of the collected data have been poisoned
  - Examples are:
    - The victim relies on volunteers' contribution for data collection
    - The victim downloads data from the Internet
  - Collecting training data from public sources is common
  - The attacker does not have a control over the training process
- ***Collaborative learning attack***
  - A malicious agent in federated learning sends updates that poison the model
  - Collaborative learning is designed to protect the privacy of the training data owned by several clients
  - Collaborative learning is vulnerable to poisoning attacks because the clients have control over their local data and local model updates

Poisoning Attacks in AML

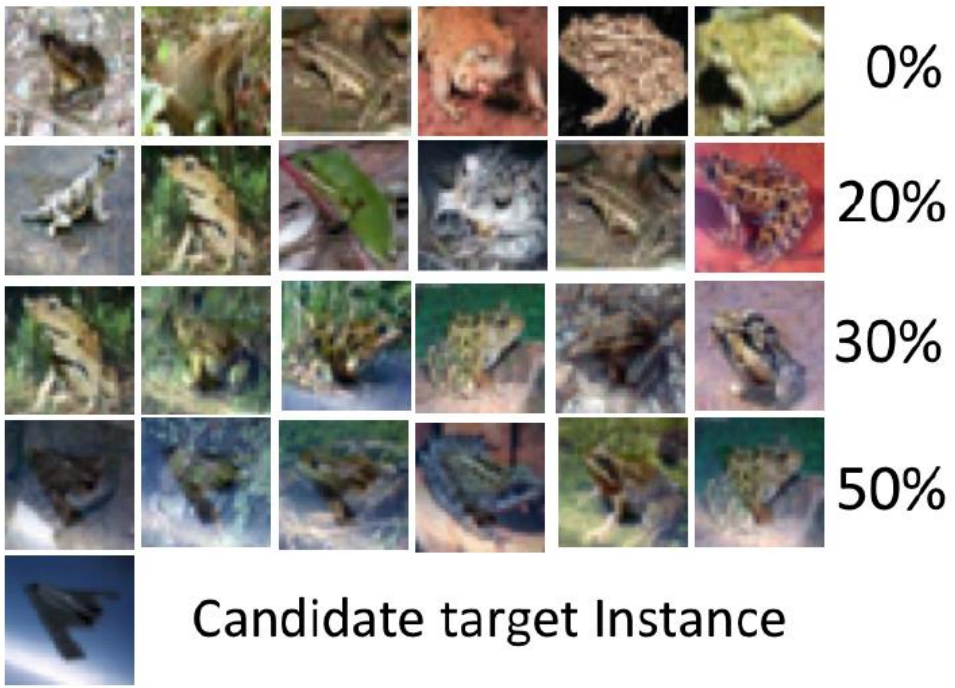
Poisoning Attacks and Defenses

- ***Post-deployment attack***
  - The attacker gets access to the model after it has been deployed, and changes the model to insert a backdoor
  - E.g., the attacker can attack a cloud server or the physical machine where the model is located
    - This attack does not rely on data poisoning to insert backdoors
    - It requires that the attacker gets access to the model by intruding the system where the model is located
- ***Code poisoning attack***
  - An attacker publicly posts an ML code that is designed to backdoor trained models
  - The victim downloads the code and applies it to solve a task
  - ML users often relay on code posted in public repositories or libraries, which can impose security risk
    - The codes can be poisoned, and when run, they can insert backdoors into ML models

Poison Frogs Attack

Poisoning Attacks and Defenses

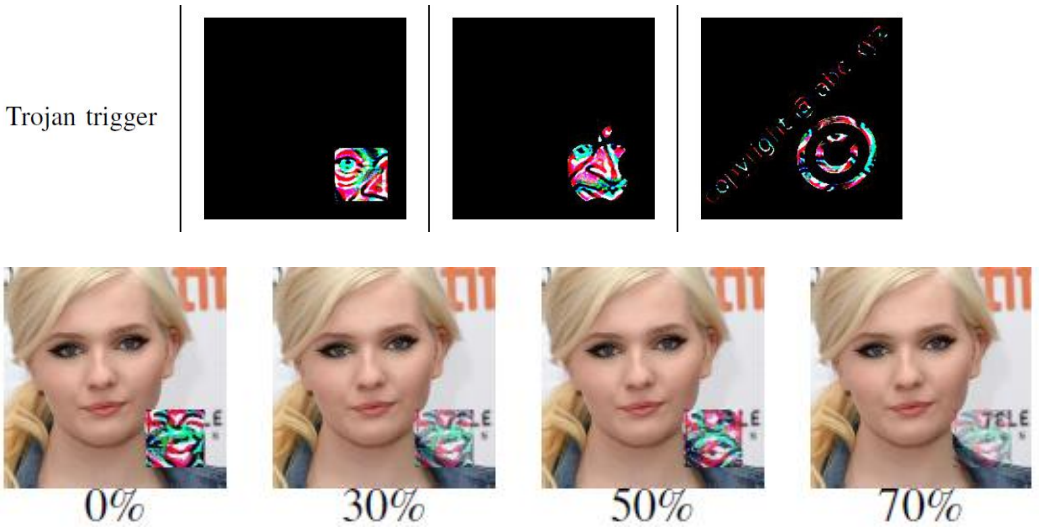
- ***Poison Frogs*** is a data collection attack
  - Shafahi (2018) Targeted Clean Label Poisoning Attacks on Neural Networks
  - The approach generates an adversarial image x_{adv} by adding adversarial perturbation to a clean image x with true label y
    - The adversarial image x_{adv} with the true (clean) label y is next inserted in the training set
    - The new model trained on a dataset containing x_{adv}, will classify x_{adv} with a target label t
      - The accuracy of the new model on clean images remains the same (is not degraded)
  - Example: images of the class "frog" are poisoned by adding a transparent overlay of images with the target class "airplane"
    - Each row of shown images has an opacity level of the overlaid "airplane" image ranging from 0% to 50% opacity
    - The model classified most images as the target class "airplane"



Trojaning Attack

Poisoning Attacks and Defenses

- ***Trojaning Attack*** is a pretrained model attack
  - Liu (2018) Trojaning Attack on Neural Networks
  - A trojan trigger is added to images, that are afterward inserted into the training set
    - The model misclassifies the samples with a trojan trigger into a target class
  - Three examples of trojan triggers are shown in the upper picture: square, Apple logo, and watermark
  - Sample poisoned images with different levels of transparency of the trojan trigger are shown in the lower picture
  - The model accuracy on the poisoned images reduced to 3%, while on clean images remained about 76%



Poisoning Attacks in AML

Poisoning Attacks

- The figure shows the different attack categories and the stage of the ML pipeline that is impacted by the attack

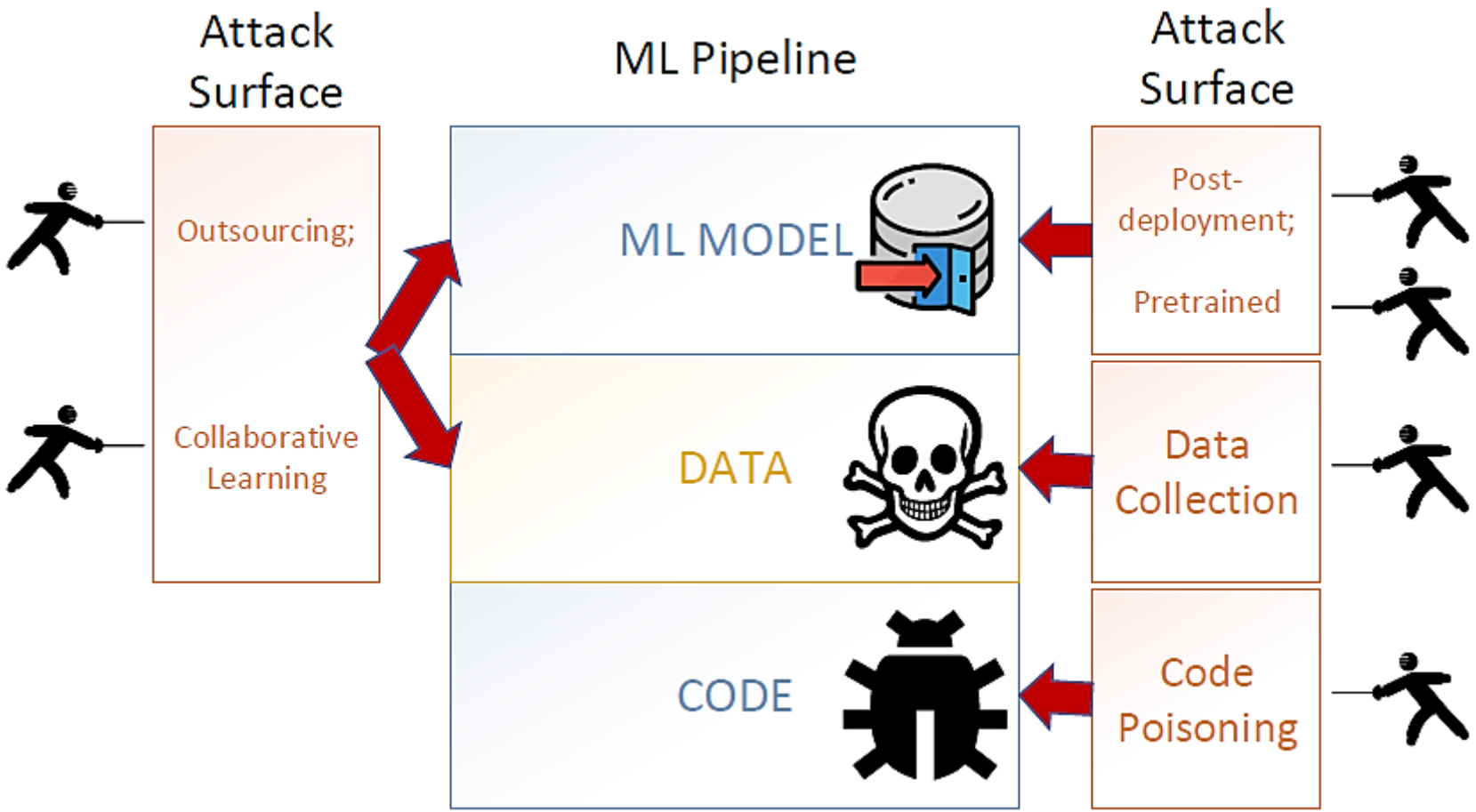


Figure from: Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Poisoning Attacks in AML

Poisoning Attacks

Attack Surface	Backdoor Attacks	Access Model Architecture	Access Model Parameters	Access Training Data	Trigger controllability	ASR	Potential Countermeasure [1]
Code Poisoning	[51] [52]	Black-Box	○	○	◐	High	Offline Model Inspection Online Model Inspection Online Data Inspection
Outsourcing	Image [6], [7], [12], [88], [122] [8]; Text [13] [14]–[16]; Audio [16], [17]; Video [85]; Reinforcement Learning [21], [97] [98] (AI GO [22]); Code processing [99], [100]; Dynamic trigger [95] Adaptive Attack [102]; Deep Generative Model [20]; Graph Model [23]	White-Box	●	●	●	Very High	Blind Model Removal Offline Model Inspection Online Model Inspection Online Data Inspection
Pretrained	[7], [56] Word Embedding [54]; NLP tasks [107]; Model-reuse [9]; Programmable backdoor [53]; Latent Backdoor [57]; Model-agnostic via appending [106]; Graph Model [101]	Grey-Box	◐	◐	◐	Medium	Blind Model Removal Offline Model Inspection Online Model Inspection Online Data Inspection
Data Collection	Clean-Label Attack [62], [63], [110] [114], (video [85], [109]), (malware classification [111]); Targeted Class Data Poisoning [113], [115]; Image-Scaling Attack [64], [65]; Biometric Template Update [123]; Wireless Signal Classification [19]	Grey-Box	◐	◐	◐	Medium	Offline Data Inspection Online Model Inspection Online Data Inspection
Collaborative Learning	Federated learning [11], [71], [72], (IoT application [70]); Federated learning with distributed backdoor [119]; Federated meta-learning [120]; feature-partitioned collaborative learning [124]	White-Box	●	●	●	High	Offline Model Inspection [2]
Post-deployment	[78] [76], [77] Application Switch [125]	White-Box	●	●	◐	Medium	Online Model Inspection Online Data Inspection

●: Applicable or Necessary; ●: Inapplicable or Unnecessary; ◐: Partially Applicable or Necessary.

Gao et al. (2020) - Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Defenses Against Poisoning Attacks

Poisoning Attacks and Defenses

- ***Defense strategies*** against poisoning adversarial attacks were categorized in Gao et al. (2020) into:
  - Blind backdoor removal
  - Offline inspection
  - Online inspection
  - Post backdoor removal

Defenses Against Poisoning Attacks

Poisoning Attacks and Defenses

- ***Blind backdoor removal***
  - The goal is to remove or suppress the backdoor effect while achieving high accuracy on clean inputs
  - The defense can be performed either offline or online
  - This defense does not differentiate a backdoored model from a clean model, or trigger input from clean input
    - It usually reduces the accuracy on clean inputs
- ***Offline inspection defense***
  - These defense strategies can be based on data inspection or model inspection
  - Offline data inspection defense
    - Assumes that the poisoned data is available to the defenders
    - Data inspection can include outlier detection based on spectral signature, gradients, or activation values
  - Offline model inspection defense
    - Do not assume that poisoned data is available to the defenders
    - Therefore, these defenses are more realistic, and can be applied to various attacks
    - Defense can include explainability approaches, or inspecting predictions for all labels

Defenses Against Poisoning Attacks

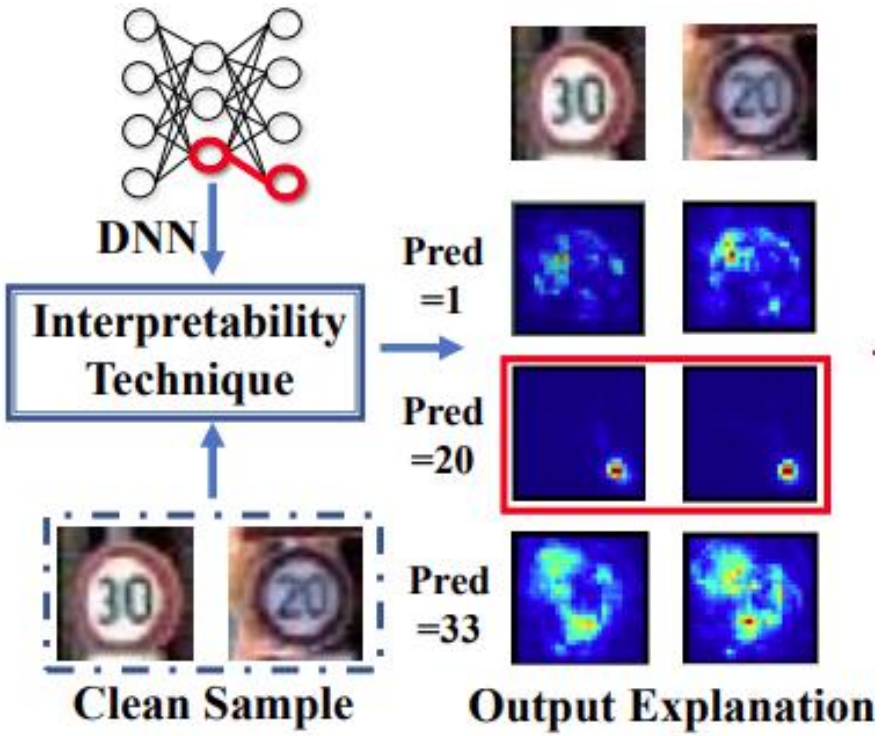
Poisoning Attacks and Defenses

- ***Online inspection defense***
  - Spplied to monitor the behavior of input data or a model during run-time
  - Online data inspection defense
    - Apply anomaly detection to check if the inputs contain a trigger (e.g., based on the entropy of clean and trigger samples)
  - Online model inspection defense
    - Apply anomaly detection to identify abnormal behavior of a backdoored model (e.g., high activation values by neurons, abnormal flow between the layers in NN)
  - These approaches typically require some preparations to be performed offline (e.g., determining a threshold to distinguish clean from trigger inputs)
- ***Post backdoor removal defense***
  - Includes techniques to remove the backdoor, after it is identified by the previous defense approaches
    - If the defender has access to poisoned data, they can remove trigger inputs, and retrain the model using only clean inputs
    - Another approach is to change the labels of the poisoned inputs with triggers to the correct labels, and then retrain the model

Neuron Inspect Defense

Poisoning Attacks and Defenses

- NeuronInspect is an offline model inspection defense
  - Huang (2019) NeuronInspect: Detecting Backdoors in Neural Networks via Output Explanations
  - Apply ML explainability approaches to create heatmaps for the target class and non-target classes
    - The assumption is that the heatmaps for the target class differ significantly for clean and trigger inputs
    - In the figure, the target class is 20 (the third row), and the inserted trigger is noticeable in the heatmaps
  - Outlier detection (based on the produced heatmaps) is applied as a defense strategy



Privacy Attacks in AML

Privacy Attacks and Defenses

- ***Privacy attacks*** are also referred to as inference attacks or confidentiality attacks
- They can broadly be developed against:
  - Training data
    - E.g., reveal the identity of patients whose data was used for training a model
  - ML model
    - E.g., reveal the architecture and parameters of a model that is used by an insurance company for predicting insurance rates
    - E.g., reveal the model used by a financial institution for credit card approval
- Privacy attacks are commonly divided into the following main categories
  - Membership inference attack
  - Feature inference attack
  - Model extraction attack

Membership Inference Attack

Privacy Attacks and Defenses

- ***Membership inference attack***
  - Adversarial goal: determine whether or not an individual data instance x^* is part of the training dataset $\mathcal{D}$ for a model
- The attack typically assumes black-box query access to the model
- Attacks on both supervised classification models and generative models (GANs, VAEs) have been demonstrated
- A common approach is to first train several ***shadow models*** that imitate the behavior of the target model, and use the prediction vectors of the shadow models for training a binary classifier (that infers the membership)

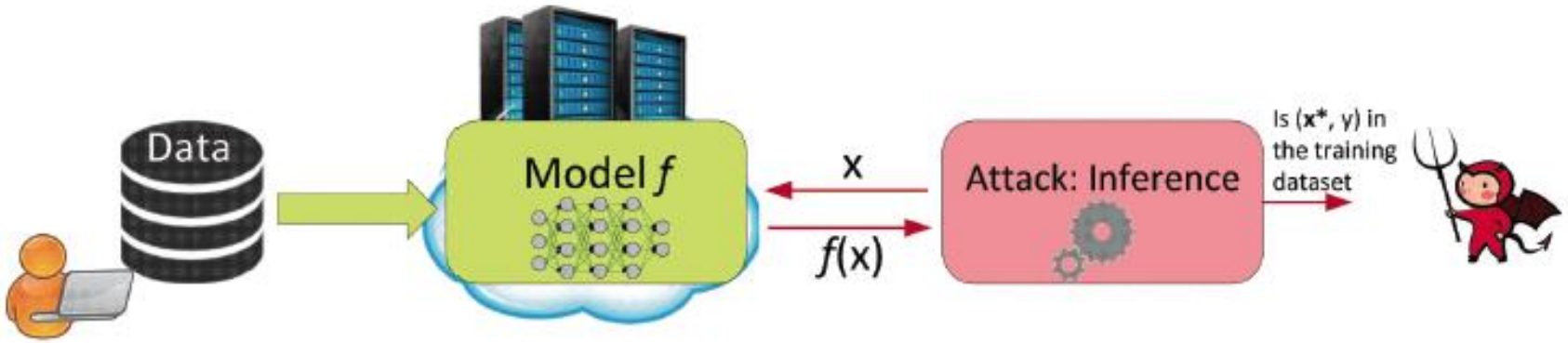
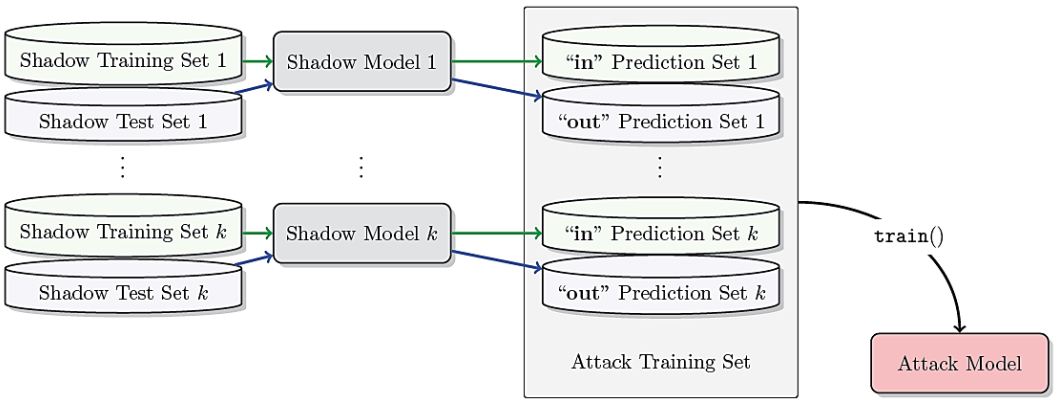


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

Shadow Training Attack

Privacy Attacks and Defenses

- Shokri (2016) Membership Inference Attacks Against Machine Learning Models
- Threat model:
  - The adversary has back-box query access to the target model
  - The goal is to infer whether input samples were part of its private training set
- *Shadow training* approach:
  - Create several shadow models to substitute the target model
  - Each shadow model is trained on a dataset that has a similar distribution as the private training dataset of the target model
  - The output probability vectors from the shadow models are next used as inputs for training attack models (as binary classifiers) for each class
  - The assumption is that the output probability vectors of the shadow models are different for samples that are in the shadow training sets and in the shadow test sets



Feature Inference Attack

Privacy Attacks and Defenses

- ***Feature inference attack***
  - Adversarial goal: recreate certain features of data instances x^* or statistical properties (such as class average of x^*) of the training dataset $\mathcal{D}$ for the model
- A.k.a. attribute inference attack, reconstruction attack, or data extraction attack
- Various attacks have been developed to either recover partial information about the training data (such as sensitive features of the dataset, or typical representatives for specific classes in the dataset) or full data samples
  - An example of a training data extraction attack is described later in this lecture
- Similarly, recreating dataset properties that were not encoded in the dataset is also referred to as ***property inference attack***
  - E.g., extract information about the ratio of men and women in a patient dataset, despite that gender information was not provided for the training records

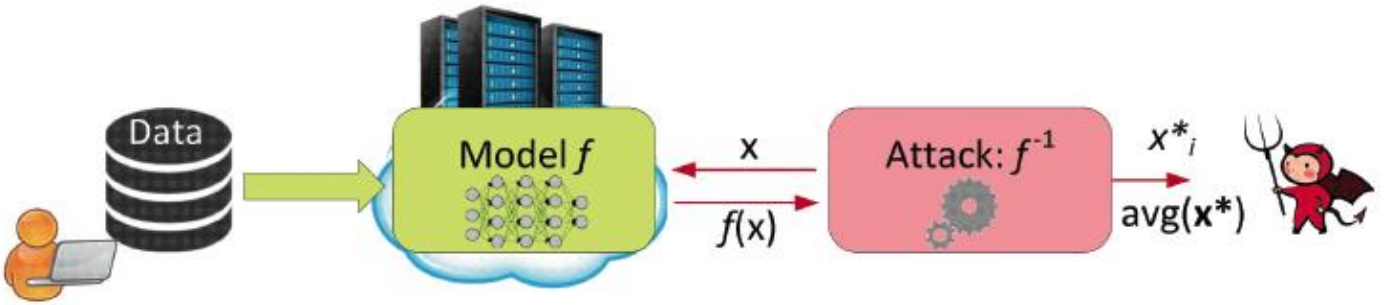


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

Model Inversion Attack

Privacy Attacks and Defenses

- Fredrickson (2015) Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures
- ***Model inversion attack*** creates prototype examples for the classes in the dataset
  - The authors demonstrated an attack against a DNN model for face recognition
  - Given a person's name and white-box access to the model, the attack reverse-engineered the model and produced an averaged image of that person
    - The obtained averaged image (left image below) makes the person recognizable
  - This attack is limited to classification models where the classes pertain to one type of object (such as faces of the same person)

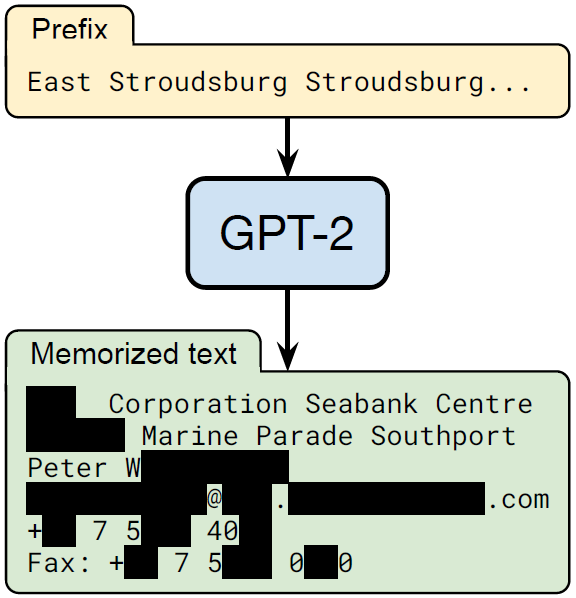
Recovered image using the model inversion attack

Image of the person used for training the model

Training Data Extraction Attack

Privacy Attacks and Defenses

- Carlini (2020) Extracting training data from large language models
- Attack on *GPT-2* language model (LM)
  - GPT-2 has 1.5 billion parameters, it is trained on public data collected from the Internet
- The goal of the attack is to analyze output text sequences from GPT-2 and identify text that has been memorized by the model
  - The authors had black-box query access to the GPT-2 model
- Successfully extracted data included:
  - Personally identifiable information (PII): names, phone numbers, e-mail addresses
  - News headlines, log files, Internet forum conversations, code
- The extracted information in the shown example was present in just one document in the training data



Model Extraction Attack

Privacy Attacks and Defenses

- ***Model extraction attack***
  - Adversarial goal: reconstruct an approximated model $f'(x)$ of the target model $f(x)$
- A.k.a. model inference attack
- The approximated function $f'(x)$ will act as a substitute model and produce similar predicted outputs as the target model
  - The adversary has black-box query access to the model
  - The goal is to "steal" the model and use the substitute model for lunching other attacks, such as synthesis of adversarial examples, or membership inference attacks
- Besides creating a substitute model, several works focused on recovering the hyperparameters of the model, such as the number of layers, optimization algorithm, activation types used, etc.

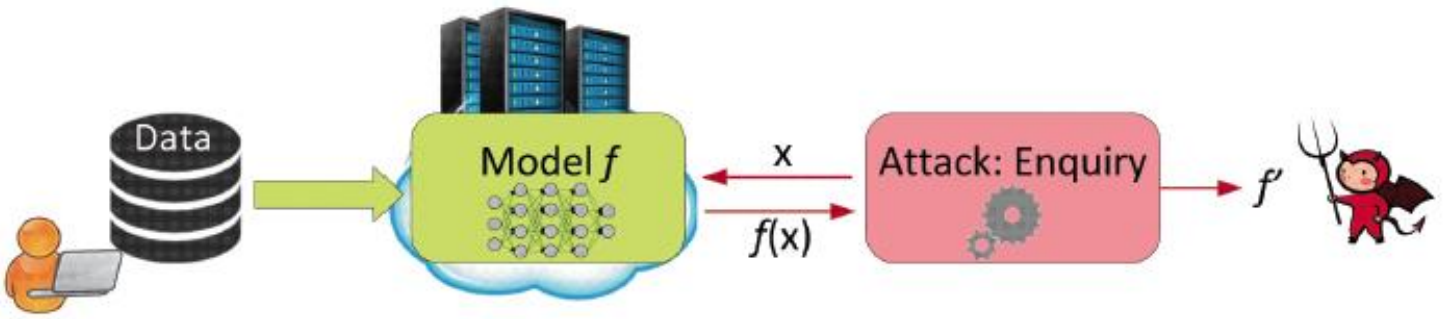


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

Causes of Privacy Leaks

Privacy Attacks and Defenses

- Overfitting is among the main causes of privacy leakage
  - It leads to poor generalization and memorization of the training data
  - Adversarial training is characterized with a trade-off between the model accuracy and robustness
    - The reduced accuracy can lead to increased sensitivity to data leakage
- Datasets that are more diverse and with larger number of class labels are more susceptible to attacks
  - Binary classifiers are safer than multiclass models
  - Input samples that are out-of-distribution (i.e., are considered outliers with respect to the distribution of the training data) are more susceptible to privacy leakage
- Model complexity can also impact the vulnerability
  - Complex models with large number of parameters memorize more sensitive information about the training data

Attacks against Distributed Learning

Privacy Attacks and Defenses

- ***Privacy attacks against federated learning*** and related distributed learning cases have also been demonstrated
- The attacks can be *passive* (the adversary collects the updates) and *active* (the adversary shares information to impact the training procedure)
  - A malicious attacker who participates in federated learning can perform membership inference attack to reveal if other participants used a data record for training
    - Nasr (2018) Machine Learning with Membership Privacy Using Adversarial Regularization
  - Property inference attacks were developed to reveal whether training data with certain properties were used by the other participants
    - Melis (2019) Exploiting Unintended Feature Leakage in Collaborative Learning
  - Training data reconstruction attack was accomplished by using an additional GAN model for reconstructing class representative samples from the local dataset used by the other participants
    - Hitaj (2017) Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning

Defenses against Privacy Attacks

Privacy Attacks and Defenses

- ***Defense strategies*** against privacy attacks in ML can be broadly classified into:
  - Anonymization techniques
  - Encryption techniques
  - Differential privacy
  - Distributed learning
  - ML-specific techniques

Anonymization Techniques

Privacy Attacks and Defenses

- ***Anonymization*** techniques provide privacy protection by removing identifying information in the data
  - E.g., remove names and addresses of people in a dataset
- Anonymization is not an efficient defense method, since the remaining information in the data can be used for identifying the individual data instances
  - For example, based on health records (including diagnoses and prescriptions) with removed personal information released by an insurance group in 1997, a researcher extracted the information for the Governor of Massachusetts
  - The same researcher later showed that 87% of all Americans can be uniquely identified using 3 bits of information: ZIP code, birth date, and gender
- De-anonymization of data by using connections to external sources of information is referred to as ***linkage attack***
  - For example: in 2006, Netflix published anonymized 10 million movie rankings by 500,000 customers; two researchers showed later that by using movie recommendations on IMDb (Internet Movie Database) they could identify the customers in the Netflix data

k-anonymity

Privacy Attacks and Defenses

- ***k-anonymity*** is an approach for protecting data privacy by suppressing certain identifying data features
  - This approach removes fields of data of individual people who have unique characteristics
    - E.g., students at UI who are from Latvia and are enrolled in Architecture
- A dataset is *k*-anonymous if for any person's record, there are at least $k - 1$ other records that are indistinguishable
  - Therefore, a linkage attack will result in a group of *k* records that can belong to a person of interest
- Limitation: this approach is mostly applicable to large datasets with low-dimensional input features
  - The more input features there are for each record, the higher the possibility of unique records

Encryption Techniques

Privacy Attacks and Defenses

- ***Encryption*** is a cryptography approach, which converts the original representation of information into an alternative form
  - The sender of encrypted information shares the decoding technique only with the intended recipients of the information
- Encrypting the training data has been applied in ML
  - Common techniques for data encryption include:
    - ***Homomorphic encryption (HE)*** - allows users to perform computations on encrypted data (without decrypting it)
    - ***Secure multi-party computation (SMPC)*** - is an extension of encryption in multi-party setting, and allows two or more parties to jointly perform computation over their private encrypted data, without sharing the data
- Encrypting ML models is less common approach
  - Homomorphic encryption has been applied to the model gradients in collaborative DL setting to protect the model privacy

Differential Privacy

Privacy Attacks and Defenses

- ***Differential privacy*** is based on employing obfuscation mechanisms for privacy protection
  - A randomization mechanism $\mathcal{M}(D)$ applies noise ξ to the outputs of a function $f(D)$ to protect the privacy of individual data instances, i.e., $\mathcal{M}(D) = f(D) + \xi$
  - Commonly used randomization mechanisms include Laplacian, Gaussian, and Exponential mechanism
- DP is often implemented in practical applications
- In ML, DP is achieved by adding noise to:
  - ***Model parameters***
    - Differentially private SGD (Abadi, 2016) adds noise to the gradients of NNs during training
  - ***Model outputs***
    - PATE (Private Aggregation of Teacher Ensembles) approach (Papernot, 2018) adds noise to an aggregated output of an ensemble of models to preserve the training data privacy
  - ***Training data***
    - Obfuscation of training data in ML has been also investigated in several works

Differentially Private SGD

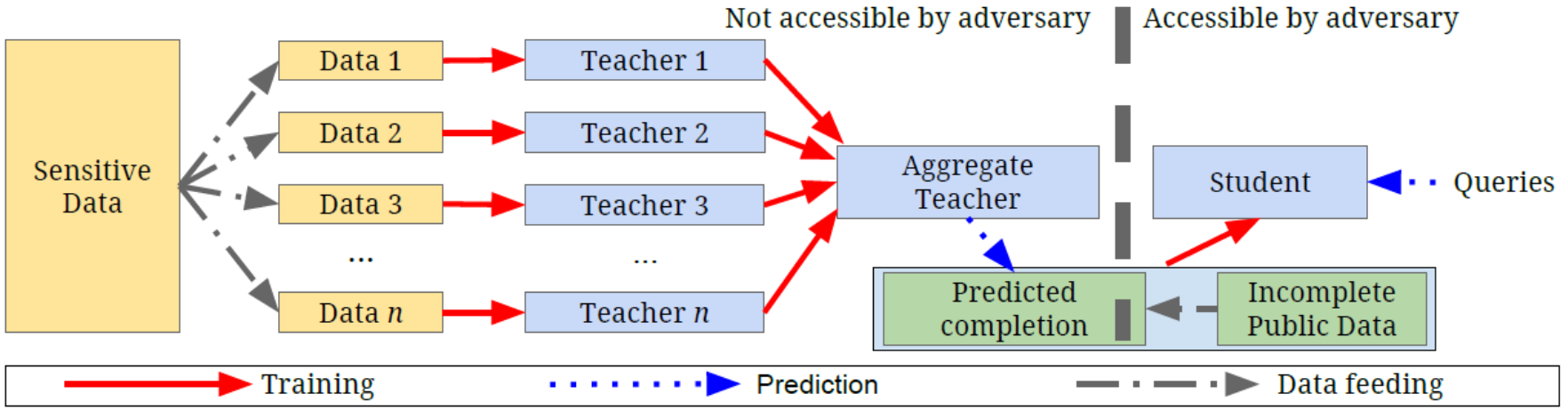
Privacy Attacks and Defenses

- Abadi (2016) Deep Learning with Differential Privacy
- This work introduced differential privacy for training ML models for protecting the privacy of the training data
  - Differential privacy (DP) is applied to Stochastic Gradient Descent (SGD) during model training
  - DP-SGD clips the gradients and adds Gaussian noise to the gradients with respect to the model parameters
  - This approach controls the amount of information from the training data that is memorized by the model during training
  - The goal is to produce ML models which provide approximately the same privacy when an individual input instance is removed from the training dataset
- The paper also introduces a method for calculating the privacy loss, called moments accountant

Private Aggregation of Teacher Ensembles (PATE)

Privacy Attacks and Defenses

- Papernot (2018) Scalable Private Learning with PATE
- ***PATE (Private Aggregation of Teacher Ensembles)*** approach employs an ensemble of models trained on disjoint subsets of training data, called teachers
  - Differentially private noise is added to the outputs of the teacher models, and the aggregated outputs are used to train another model, called student model
  - The student model is trained in a semi-supervised manner on public data, and therefore protects the privacy of the sensitive data used for training the teachers
  - PATE employs Gaussian DP randomization mechanism, and employs noise aggregation based on the consensus of the teachers
  - In comparison to prior work, PATE provides tighter DP guarantees and high utility



Distributed Learning

Privacy Attacks and Defenses

- ***Distributed learning*** allows multiple parties to train a global model without releasing their private data
- Some form of aggregation is applied to the local updates of the model parameters by the users in distributed learning to create a global model
  - E.g., averaging is one common form of aggregation
- Federated learning is the most popular distributed learning scheme

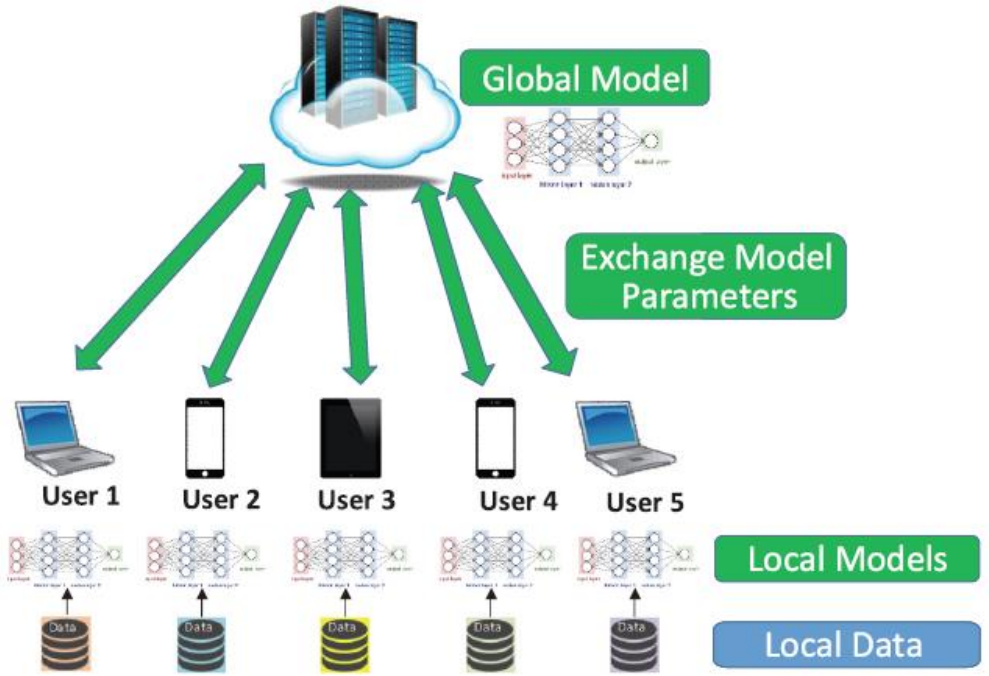


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

ML-Specific Techniques

Privacy Attacks and Defenses

- ***Regularization techniques*** in ML can be used to reduce overfitting, as well as a defense strategy against information leakage
  - Different regularization techniques in NNs include:
    - Explicit regularization: dropout, early stopping, weight decay
    - Implicit regularization: batch normalization
- Other ML-specific techniques include:
  - Dimensionality reduction – removing inputs with features that occur rarely in the training set
  - Weight-normalization – rescaling the weights of the model during training
  - Selective gradient sharing – in federated learning, the users share a fraction of the gradient at each update

Summary

Summary

- ML algorithms and methods are vulnerable to many types of attacks
  - This raises concerns, especially when ML models are applied to safety-critical applications
- Adversarial examples exhibit transferability
  - I.e., either cross-models or cross-training sets
- Adversarial examples can be leveraged to improve the performance or the robustness of ML models
  - Improved robustness to adversarial examples often comes at a cost of reduced accuracy to clean unperturbed inputs
- It is important to further study and understand the vulnerabilities of ML models and develop efficient defense strategies

References

1. Introduction to Adversarial Machine Learning – blog post by Arunava Chakraborty
2. Binghui Wang: Adversarial Machine Learning — An Introduction
3. Daniel Lowd, Adversarial Machine Learning
4. Yevgeniy Vorobeychik, Bo Li, Adversarial Machine Learning (Tutorial)
5. Xu (2019) Adversarial Attacks and Defenses in Images, Graphs and Text: A Review (link)
6. Gao et al. (2020) Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review (link)
7. Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook (link)
8. Rigaki and Carcia (2021) A Survey of Privacy Attacks in Machine Learning (link)

Other AML Recourses

- Adversarial Robustness Toolbox - a toolbox from IBM that implements state-of-the-art attacks and defenses
  - The algorithms are framework-independent, and support TensorFlow, Keras, PyTorch, MXNet, XGBoost, LightGBM, CatBoost, etc.
- Cleverhans - a repository from Google that implements latest research in AML
  - The library is updated to support TensorFlow2, PyTorch, and Jax
- ScratchAI – a smaller AML library developed in PyTorch, and explained in this blog post
- Robust ML Defenses - list of adversarial defenses with code
- AML Tutorial – by Bo Li, Dawn Song, and Yevgeniy Vorobeychik
- Nicholas Carlini website