University of Idaho

CS 502

Directed Studies: Adversarial Machine Learning

Dr. Alex Vakanski

Lecture 1

Introduction to Adversarial Machine Learning

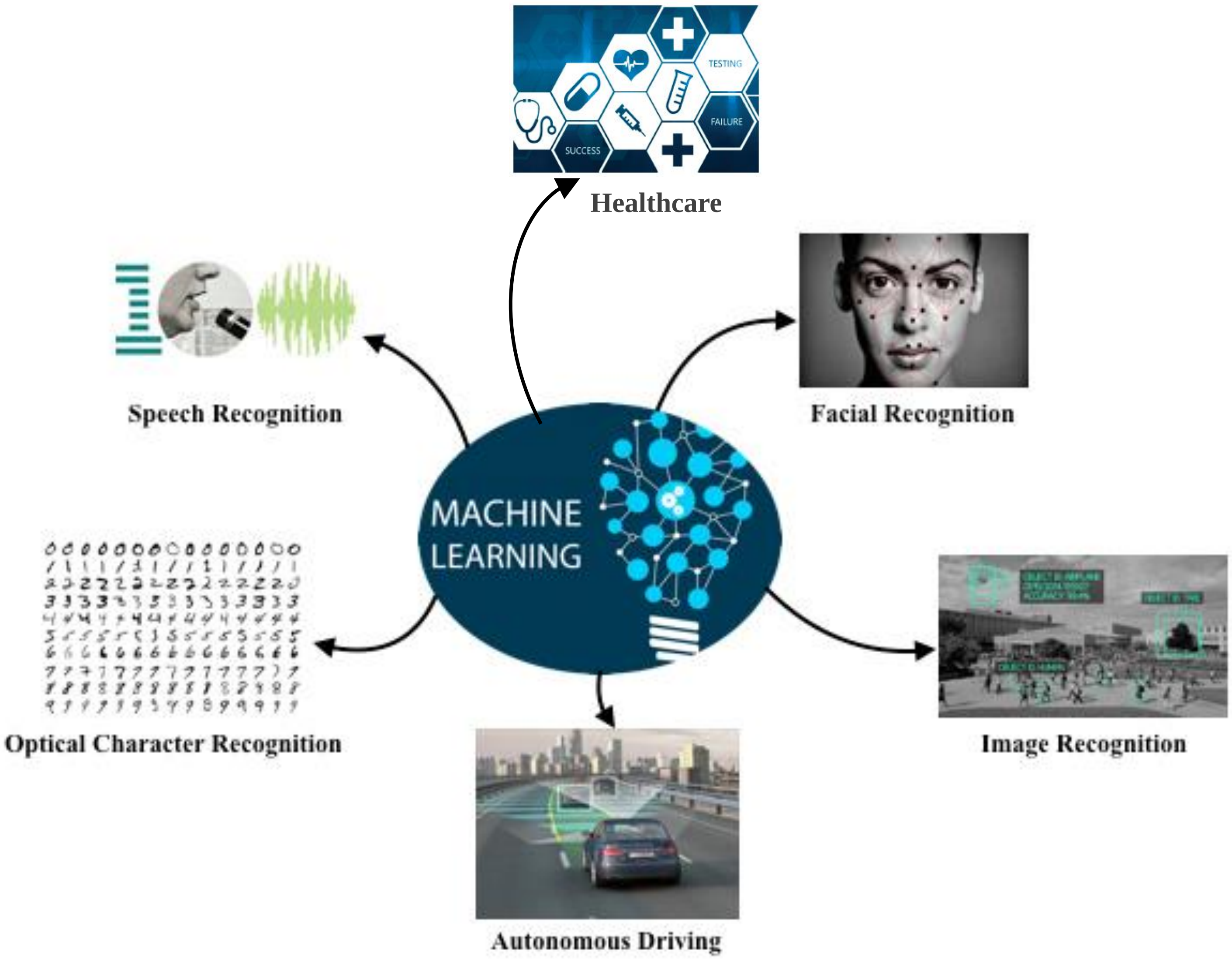
Lecture Outline

- Machine Learning (ML)
- Adversarial ML (AML)
  - Adversarial examples
- Attack taxonomy
- Common adversarial attacks
  - Noise, semantic attack, FGSM, BIM, PGD, DeepFool, CW attack
- Defense against adversarial attacks
  - Adversarial training, random resizing and padding, detect adversarial examples
- Conclusion
- References
- Other AML resources

Machine Learning (ML)

- ML tasks
  - Supervised, unsupervised, semi-supervised, self-supervised, meta learning, reinforcement learning
- Data collection and preprocessing
  - Sensors, cameras, I/O devices, etc.
- Apply a ML algorithm
  - Training phase: learn ML model (parameter learning, hyperparameter tuning)
  - Testing phase (inference): predict on unseen data

ML is Ubiquitous



Picture from: He Xiaoyi – Adversarial Machine Learning

Adversarial ML

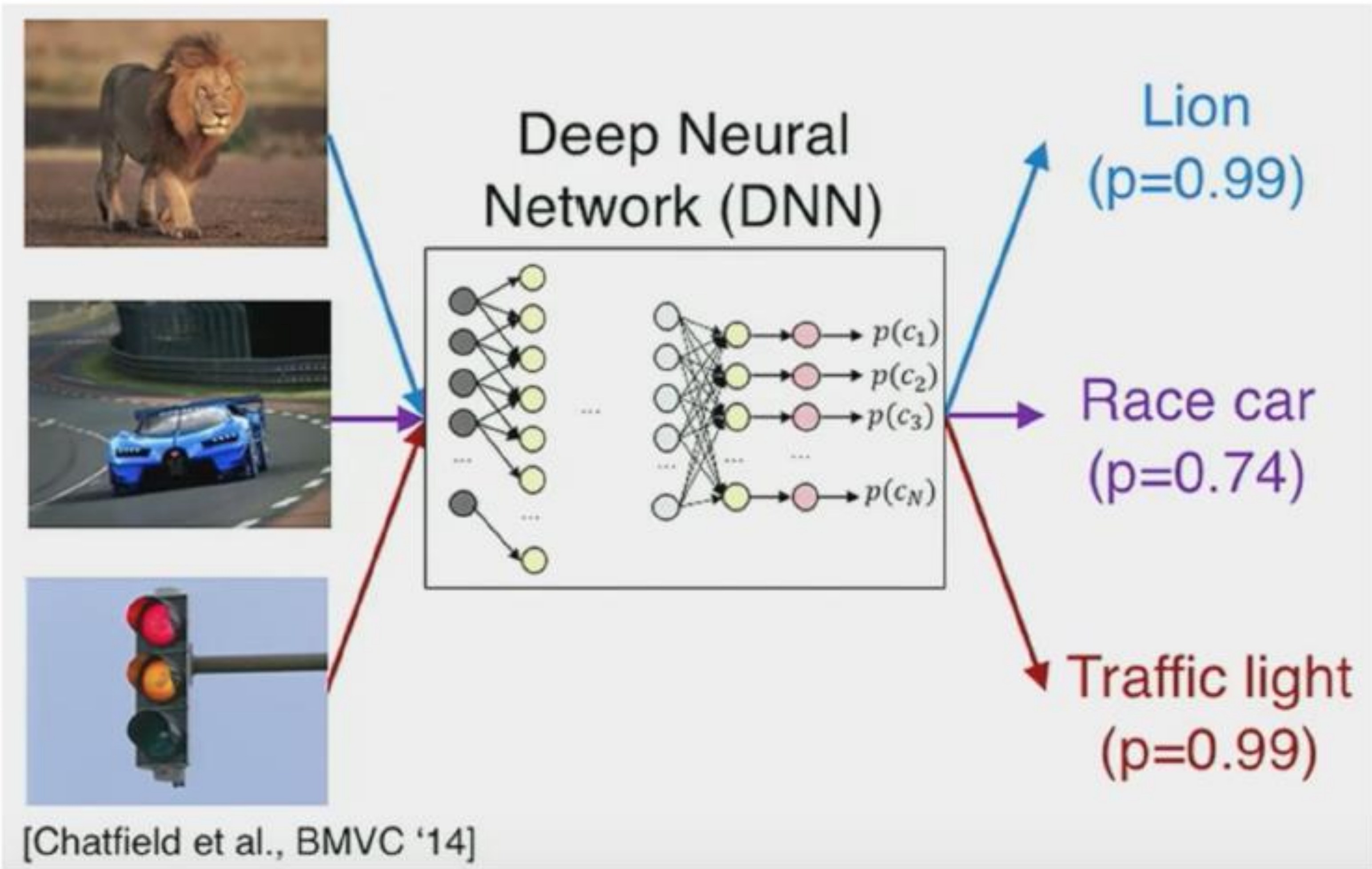
- The classification accuracy of GoogLeNet on MNIST under adversarial attacks [drops](https://blog.floydhub.com/introduction-to-adversarial-machine-learning/) from 98% to 18% (for ProjGrad attack) or 1% (DeepFool attack)

Attack	Lenet				
Noise	Dataset	Acc@1 w/	Acc@5 w/	Acc@1 w/o	Acc@5 w/o
	MNIST	0.984	1.0	0.9858	1.0
	ILSVRC2012	NA	NA	NA	NA
Semantic	Dataset	Acc@1 w/	Acc@5 w/	Acc@1 w/o	Acc@5 w/o
	MNIST	0.233	0.645	0.986	1.0
	ILSVRC2012	NA	NA	NA	NA
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

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

Adversarial Examples

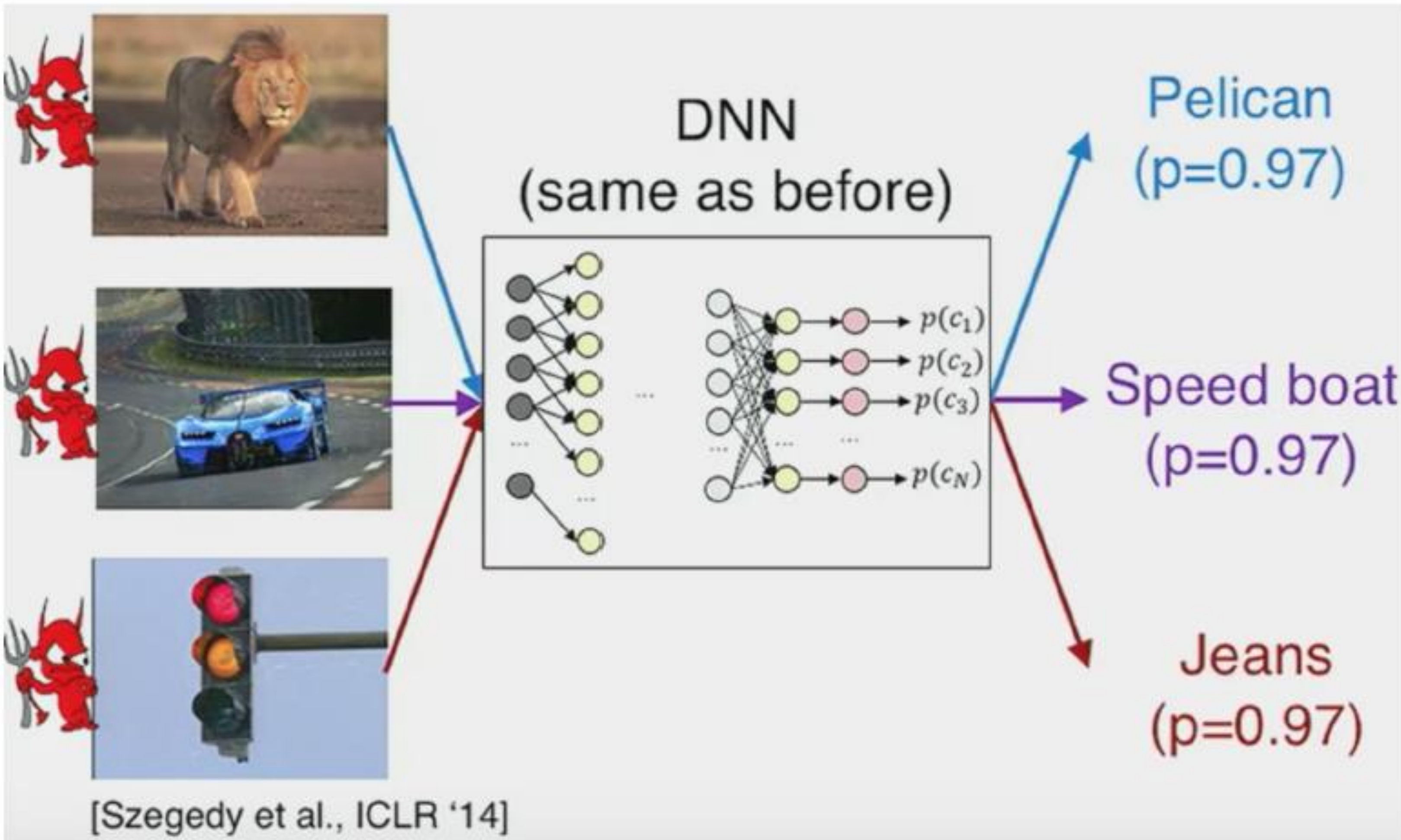
- What do you see?



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

Adversarial Examples

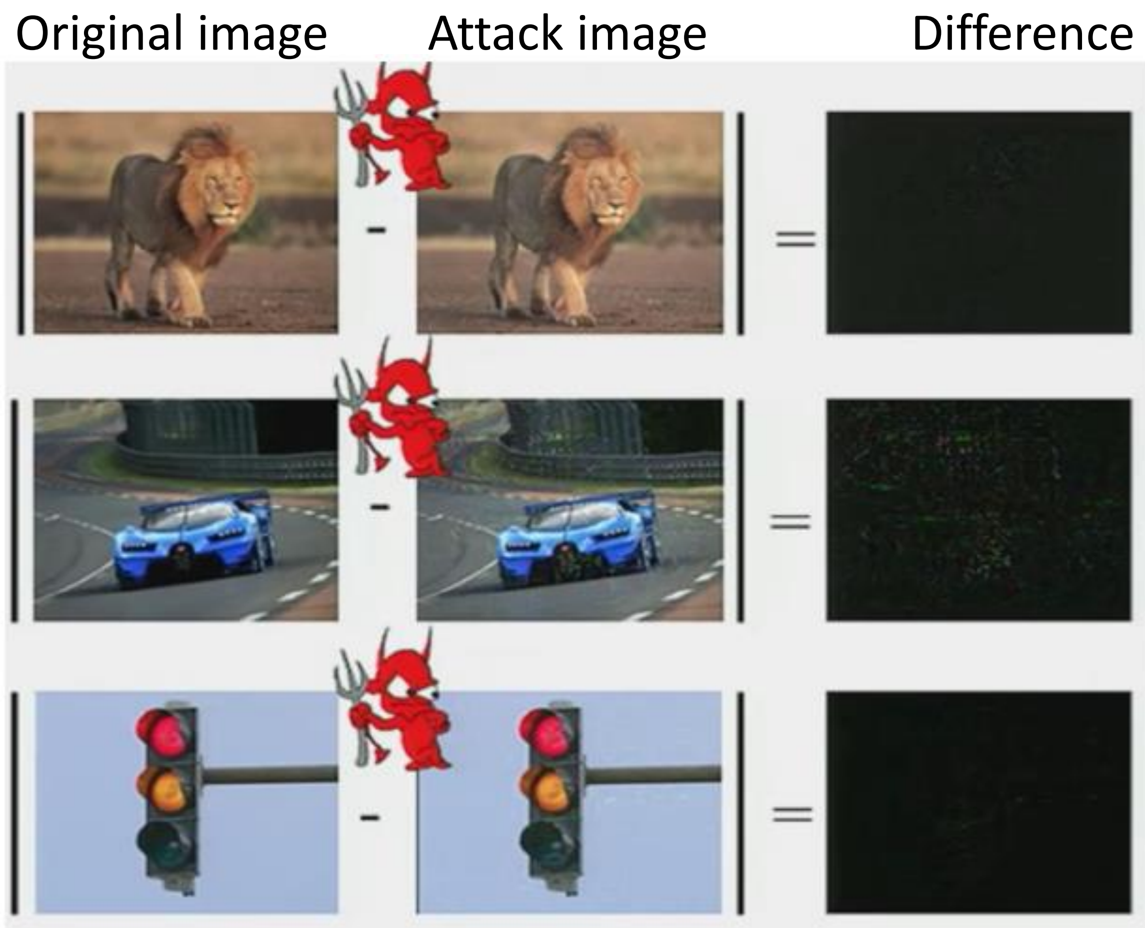
- The classifier misclassifies adversarially manipulated images



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

Adversarial Examples

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



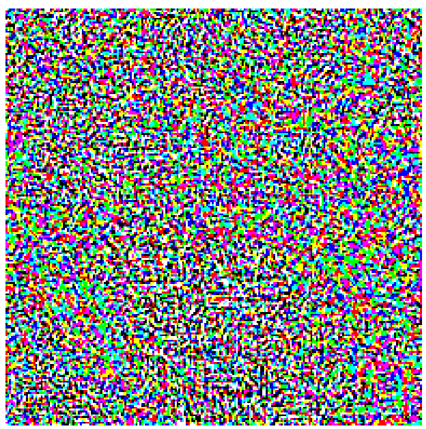
Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

Adversarial Examples

- An adversarially perturbated image of a panda is misclassified as a gibbon
- The image with the perturbation to the human eye looks indistinguishable from the original image

Original image

Classified as panda
57.7% confidence



Small adversarial noise

Adversarial image

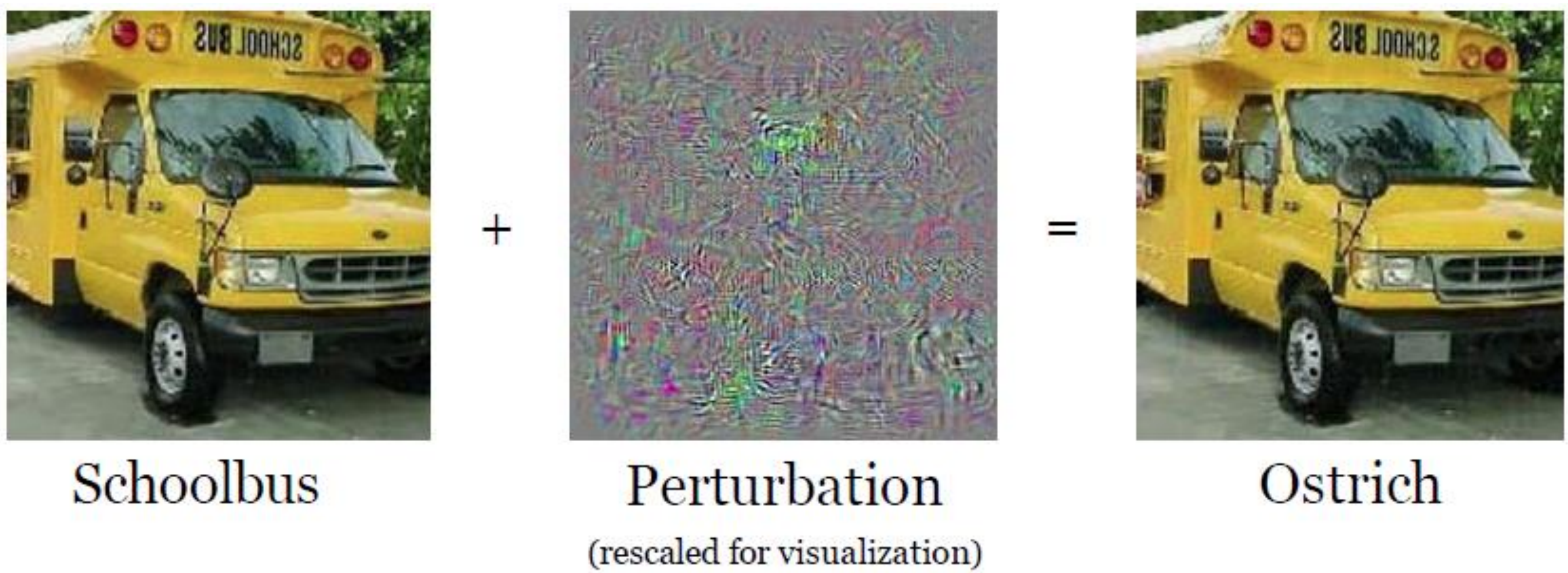
Classified as gibbon
99.3% confidence

Gibbon

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

Adversarial Examples

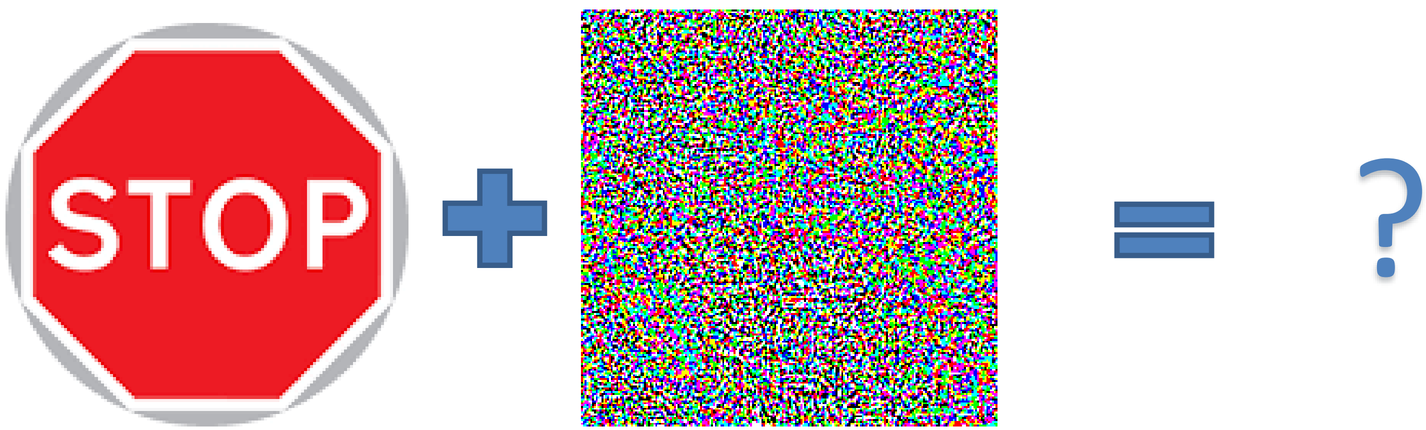
- Similar example



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

Adversarial Examples

- If a stop sign is adversarially manipulated and it is not recognized by a self-driving car, it can result in an accident

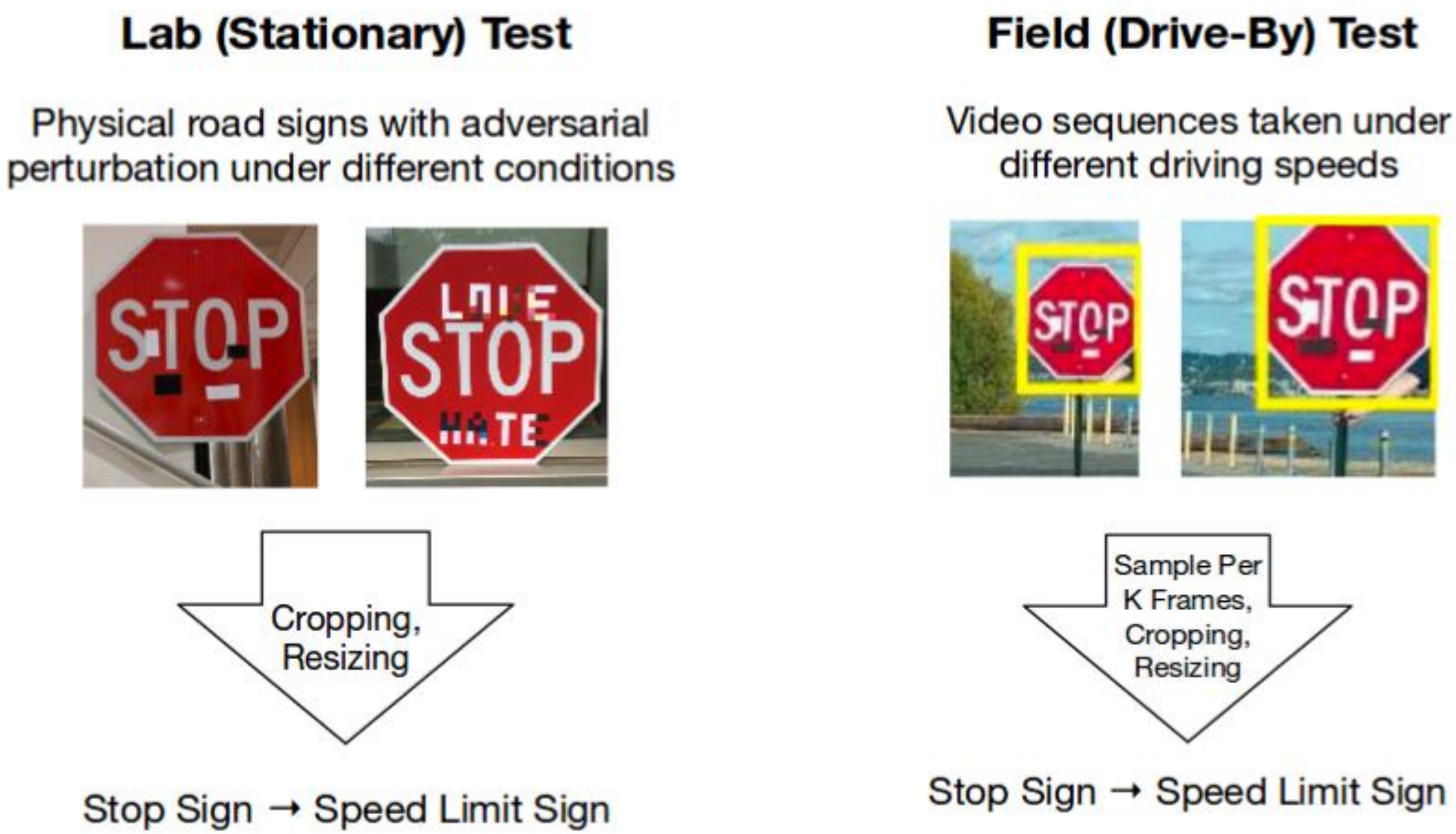


Small adversarial noise

Slide credit: He Xiaoyi – Adversarial Machine Learning

Adversarial Examples

- Recent work manipulated a stop sign with adversarial patches
  - Caused the DL model of a self-driving car to classify it as a Speed Limit 45 sign (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

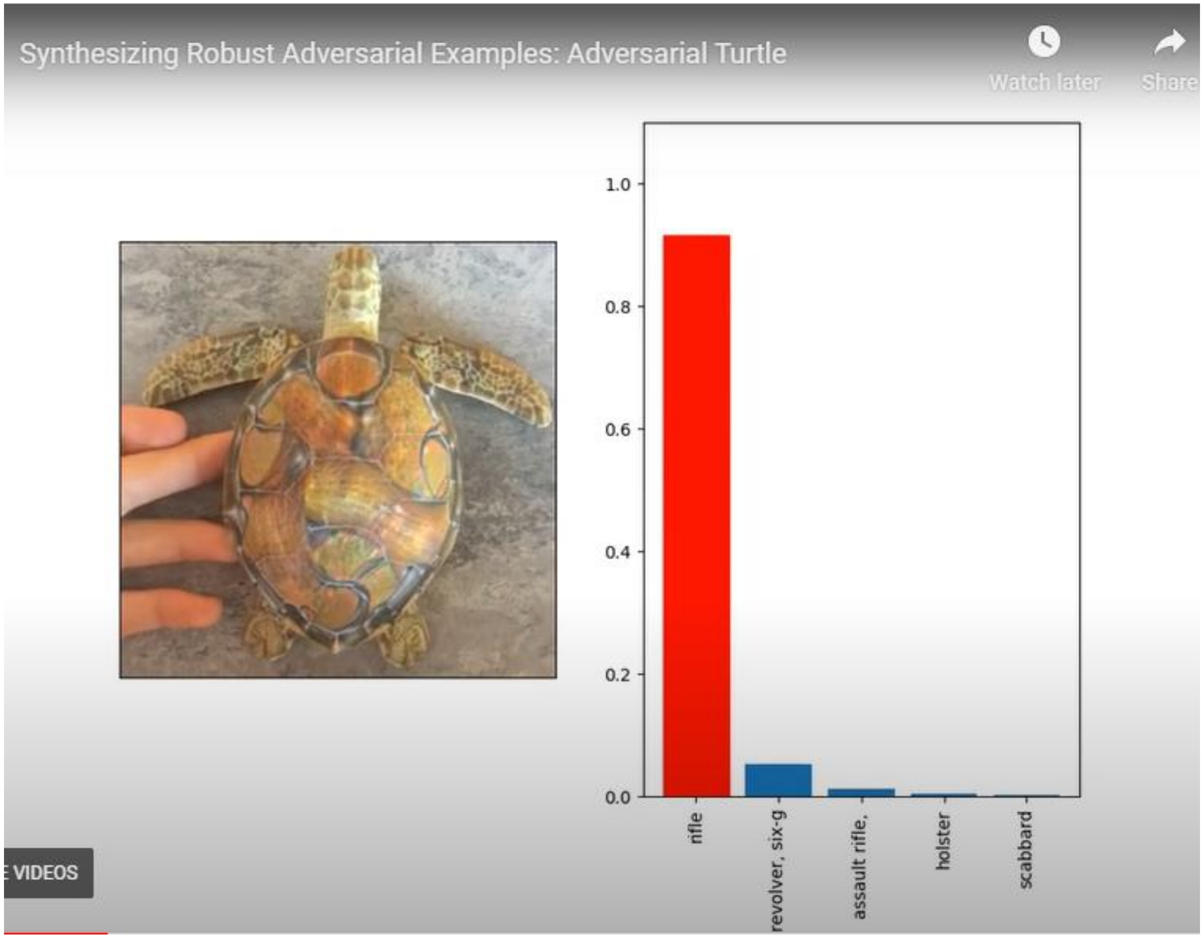
- Lab test images for signs with a target class Speed Limit 45

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

- In this example, a 3D-printed turtle is misclassified by a DNN as a rifle (video link)



Adversarial Examples

- A person wearing an adversarial patch is not detected by a person detector model (YOLOv2)

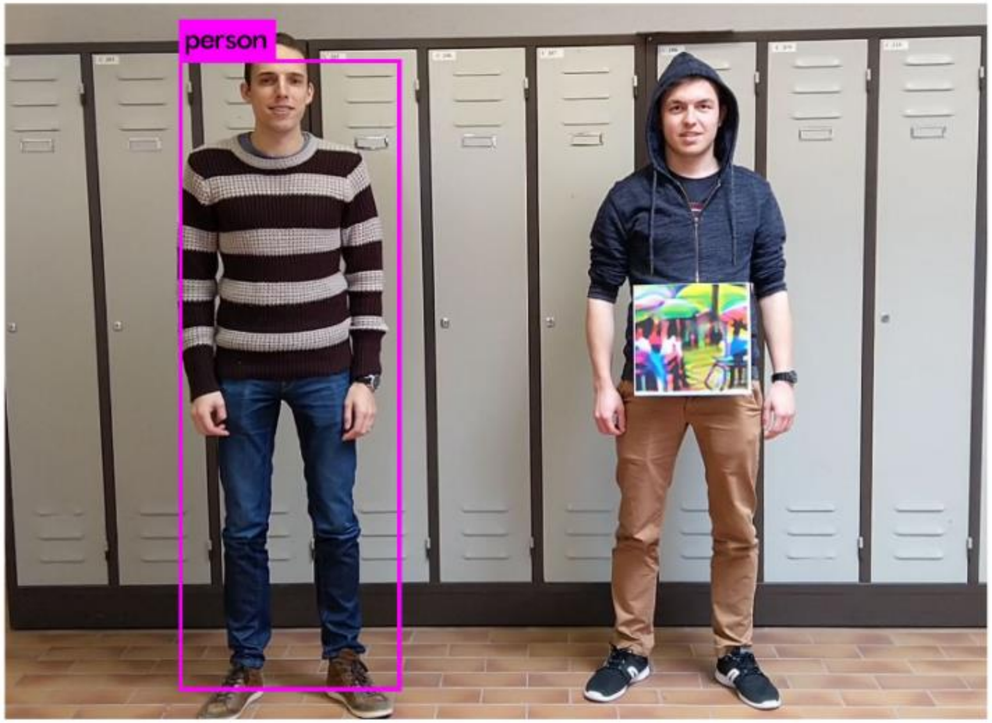


Figure 1: We create an adversarial patch that is successfully able to hide persons from a person detector. Left: The person without a patch is successfully detected. Right: The person holding the patch is ignored.

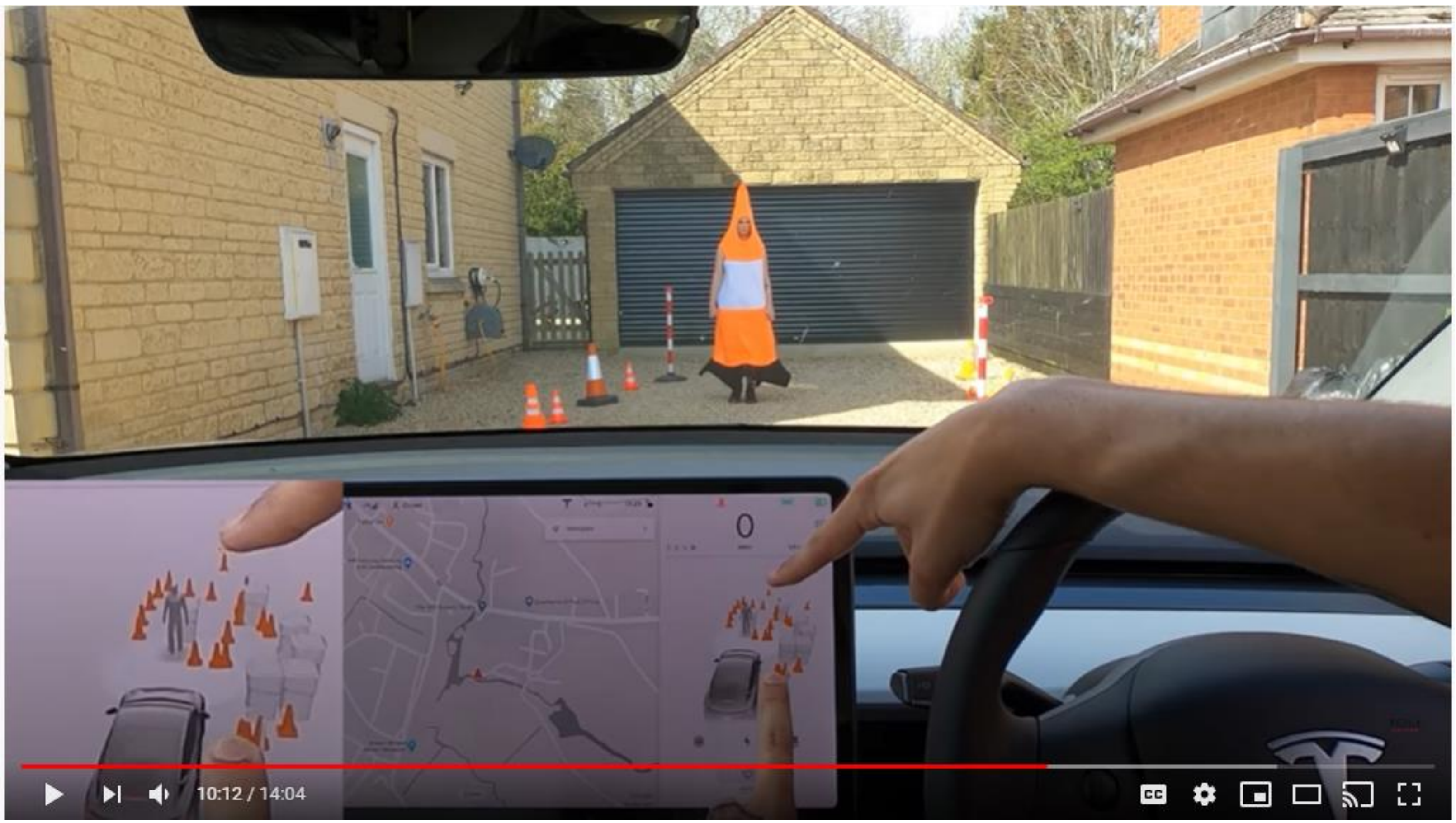
Adversarial Examples

- A “train” in the hallway?

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

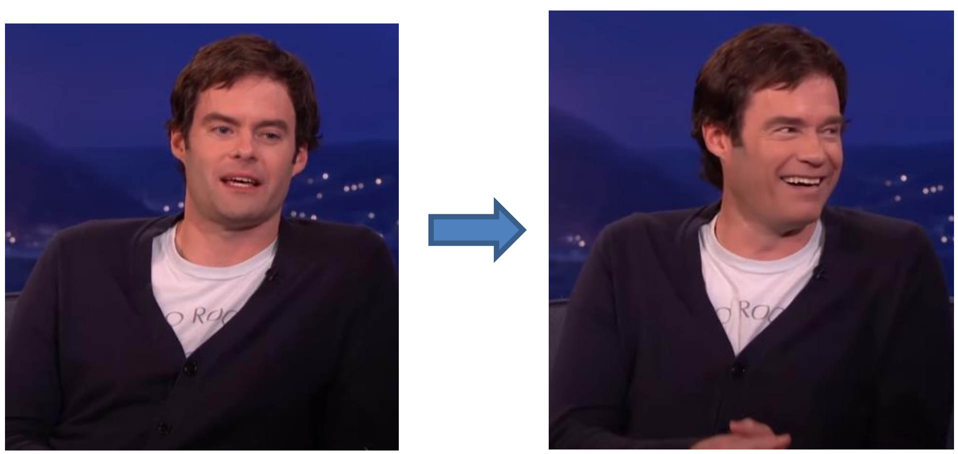
Adversarial Examples

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



Adversarial Examples

- Abusive use of machine learning
  - 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 impersonation of Arnold Schwarzenegger
  - Can have strong societal implications: elections, automated trolling, court evidence

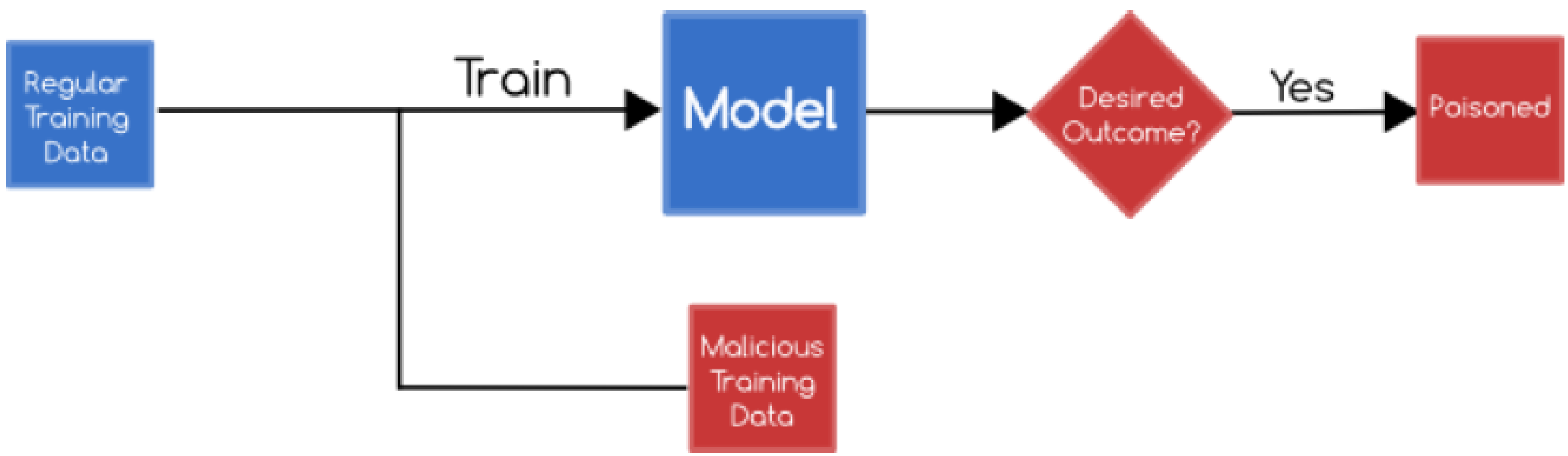


Adversarial ML

- AML is a research field that lies at the intersection of ML and computer security
  - E.g., network intrusion detection, spam filtering, malware classification, biometric authentication (facial detection)
- ML algorithms in real-world applications mainly focus on increased accuracy
  - However, few techniques and design decisions focus 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)

Attack Taxonomy

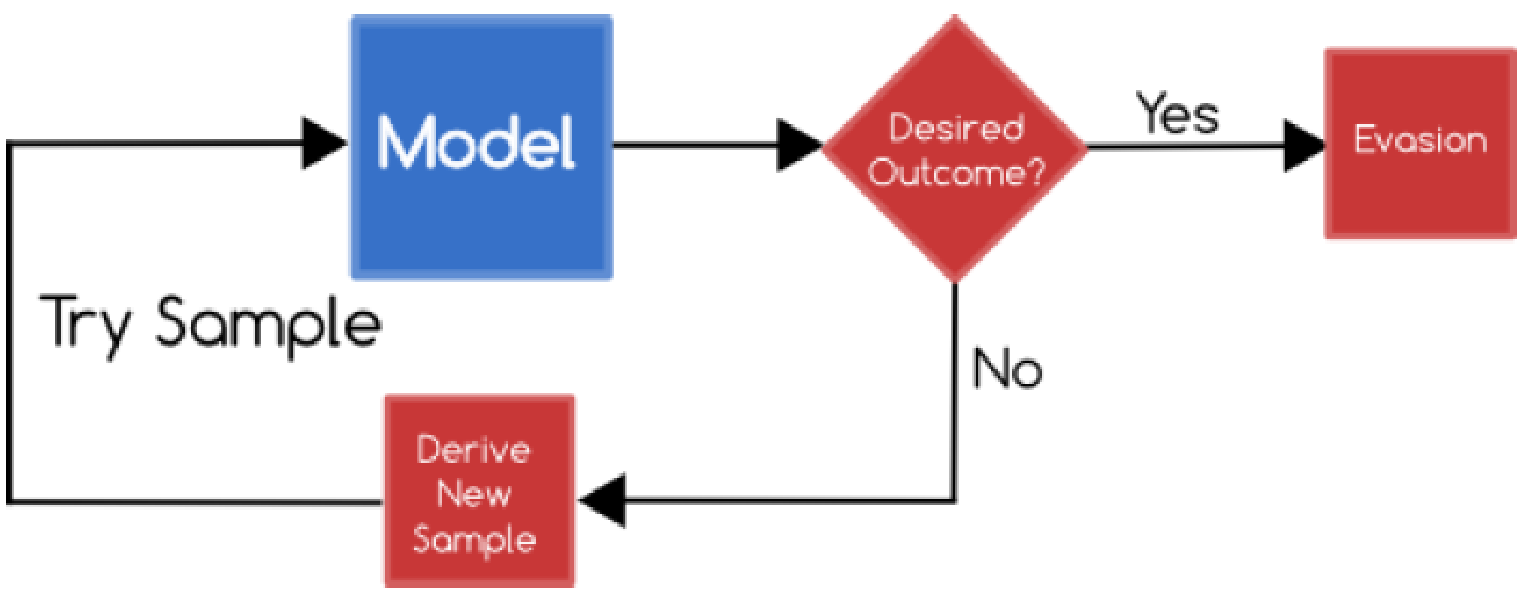
- *Data poisoning* (Causative attack):
  - Attack on the training phase
    - Attackers perturb the training set to fool the model
      - Insert malicious inputs in the training set
      - Modify input instances in the training set
      - Change the labels to training inputs
    - Attackers attempt to influence or corrupt the ML model or the ML algorithm itself



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

Attack Taxonomy

- *Evasion attack* (Exploratory attack):
  - Attack on the testing phase
  - Attackers do not tamper with the ML model, but instead cause it to produce adversary outputs
  - Evasion attack is the most common attack



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

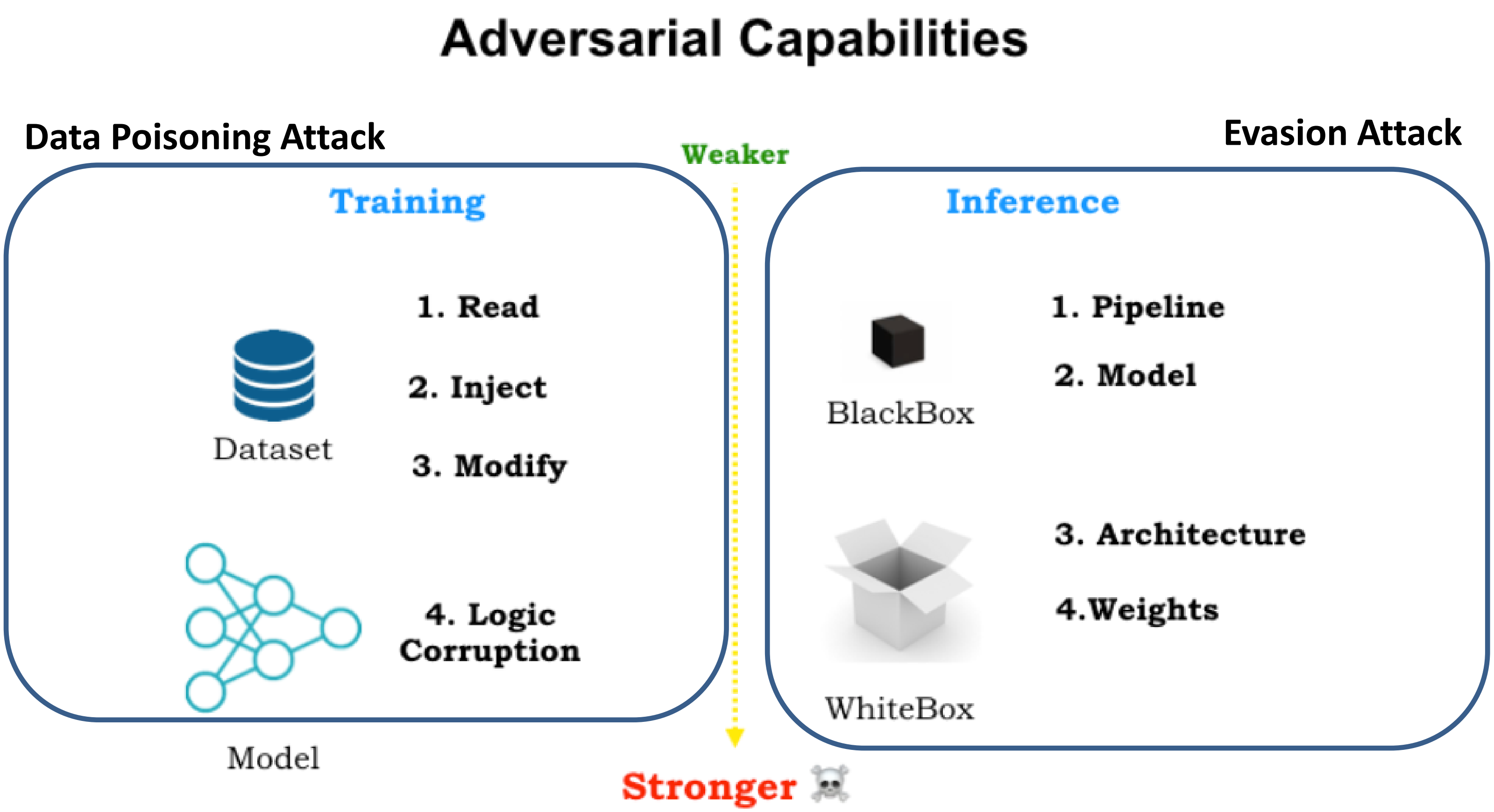
Evasion Attack

- Evasion attack 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 algorithm
      - 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 model to obtain knowledge (can get examples)

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

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/

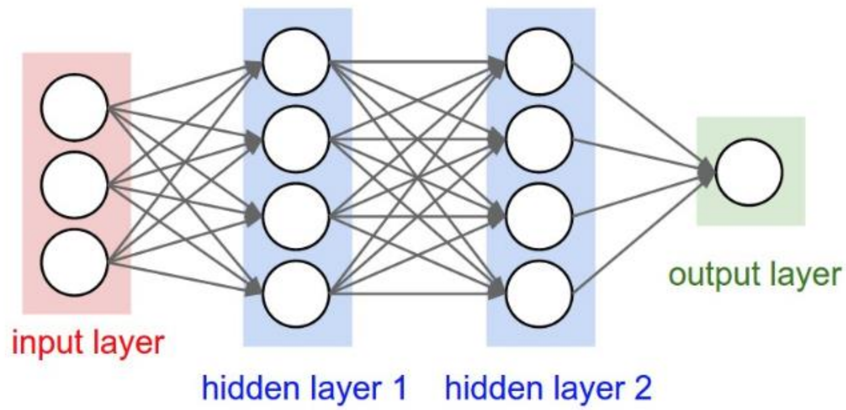
Attack Taxonomy

- Each of the above attacks can further be:
  - *Non-targeted* attack
    - The goal is to mislead the classifier to predict any labels other than the ground truth label
    - Most existing work deals with this goal
    - 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 image
    - More difficult
    - 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 sign

Evasion Attacks

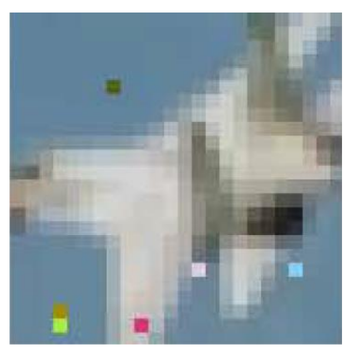
- Find a new input (*similar* to original input) but classified as another class (untargeted or targeted)

Original input



warplane

- Adversarial attack image



Evasion Attacks

- How to find adversarial images?
  - Given an image x, which is labeled by the classifier (e.g., LogReg, SVM, or NN) as class q, i.e., $C(x) = q$
  - Create an adversarial image x_{adv} by adding small perturbations δ to the original image, i.e., $x_{adv} = x + \delta$, such that the distance $D(x, x_{adv}) = D(x, x + \delta)$ is minimal
  - So that the classifier assigns a label to the adversarial image that is different than q, i.e., $C(x_{adv}) = C(x + \delta) = t \neq q$

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

distance between *x* and *x*+δ

x+δ is classified as target class *t*

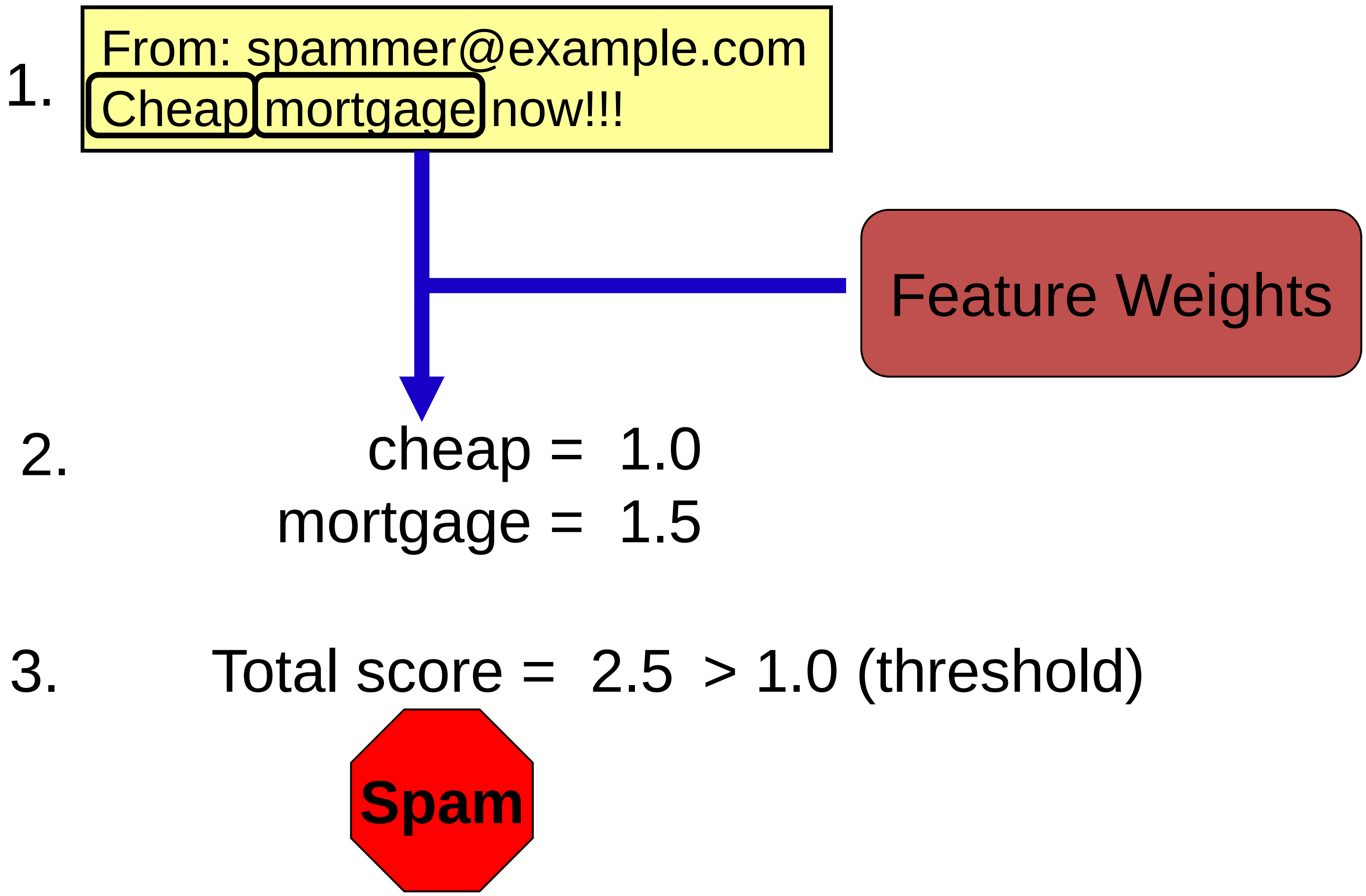
each element of *x*+δ is in [0,1] (to be a valid image)

Evasion Attacks

- Distance metrics between x and x_{adv}: $D(x, x_{adv})$
  - ℓ_0 norm: the number of elements in x_{adv} such that $x^i \neq x^i_{adv}$
    - Corresponds to the number of pixels that have been changed in the image x_{adv}
  - ℓ_1 norm: city-block distance, or Manhattan distance
    - $\ell_1 = |x^1 - x^1_{adv}| + |x^2 - x^2_{adv}| + \cdots + |x^n - x^n_{adv}|$
  - ℓ_2 norm: Euclidean distance, or mean-squared error
    - $\ell_2 = \sqrt{(x^1 - x^1_{adv})^2 + (x^2 - x^2_{adv})^2 + \cdots + (x^n - x^n_{adv})^2}$
  - ℓ_∞ norm: measures the maximum change to any of the pixels in the x_{adv} image
    - $\ell_\infty = max(|x^1 \neq x^1_{adv}|, |x^2 \neq x^2_{adv}|, \ldots, |x^n \neq x^n_{adv}|)$

Spam Filtering Adversarial Game

- Based on cumulative weights assigned to words, an email is classified as a spam or a legitimate message



Slide credit: Daniel Lowd - Adversarial Machine Learning

Spam Filtering Adversarial Game

- The spammers adapt to evade the classifier

1. From: spammer@example.com
Cheap mortgage now!!!
Eugene Oregon

Feature Weights

2. cheap = 1.0
mortgage = 1.5
Eugene = -1.0
Oregon = -1.0

3. Total score = 0.5 < 1.0 (threshold)

OK

Slide credit: Daniel Lowd - Adversarial Machine Learning

Spam Filtering Adversarial Game

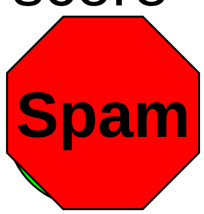
- The classifier is adapted by changing the feature weights

1. From: spammer@example.com
Cheap mortgage now!!!
Eugene Oregon

Feature Weights

2. cheap = 1.5
mortgage = 2.0
Eugene = -0.5
Oregon = -0.5

3. Total score = 2.5 > 1.0 (threshold)



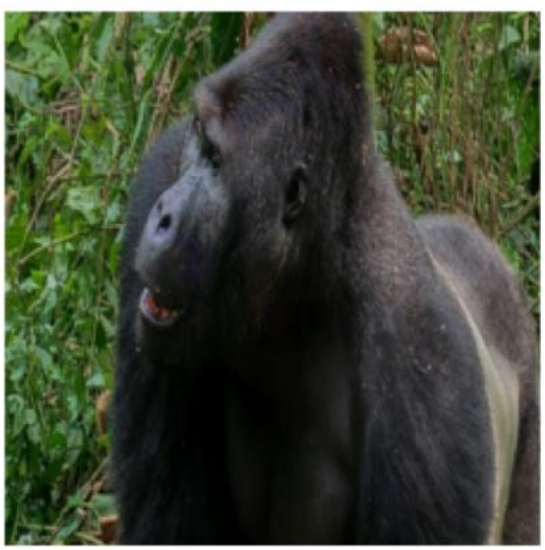
Slide credit: Daniel Lowd - Adversarial Machine Learning

Common Adversarial Attacks

- Noise attack
- Semantic attack
- Fast gradient sign method (FGSM) attack
- Basic iterative method (BIM) attack
- Projected gradient descent (PGD) attack
- DeepFool attack
- Carlini-Wagner (CW) attack

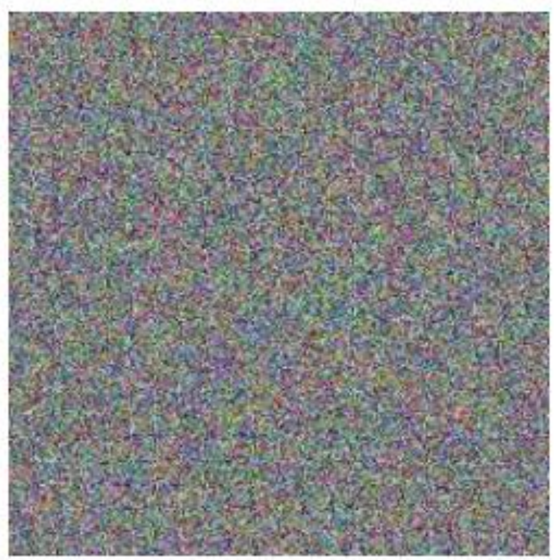
Noise Attack

- *Noise attack*
  - The simplest form of adversarial attack
  - Noise is a random arrangement of pixels containing no information
  - In Python, noise is created by the randn() function
    - I.e., random numbers from a normal distribution (0 mean and 1 st. dev.)
  - It represents a non-targeted black-box evasion attack

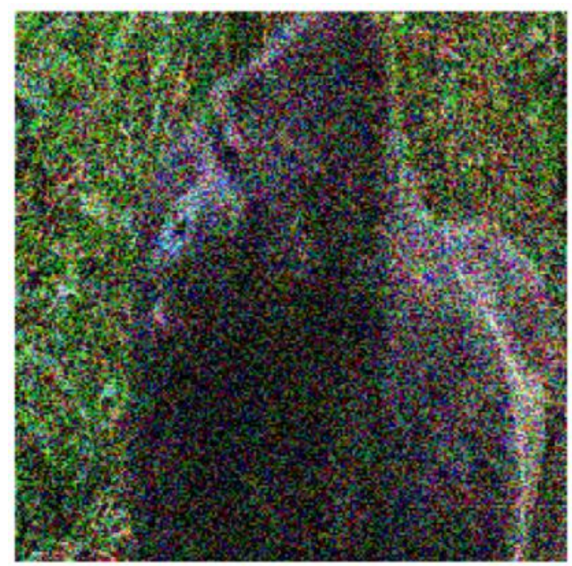


Prediction: gorilla

\+



=



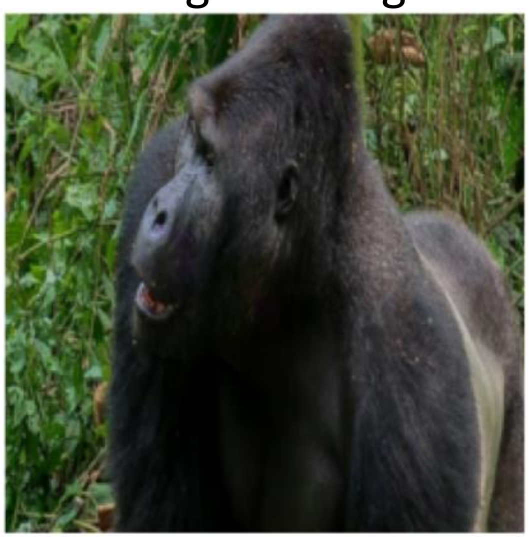
Prediction: fountain

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

Semantic Attack

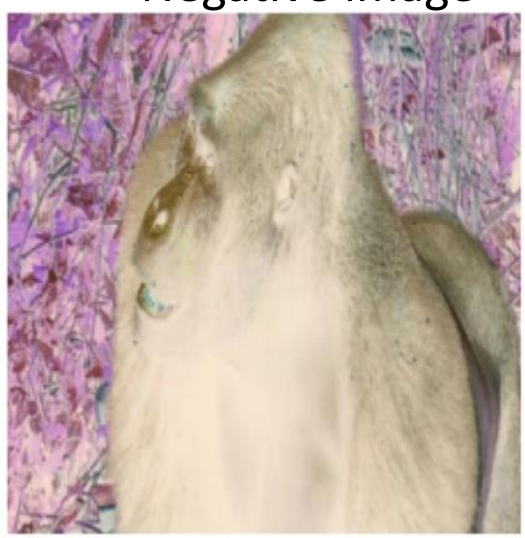
- ***Semantic attack***
  - [Hosseini (2017) - On the Limitation of Convolutional Neural Networks in Recognizing Negative Images]
  - Use negative images
    - Reverse all pixels intensities
    - E.g., change the sign of all pixels, if the pixels values are in range [-1,1]

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

- ***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

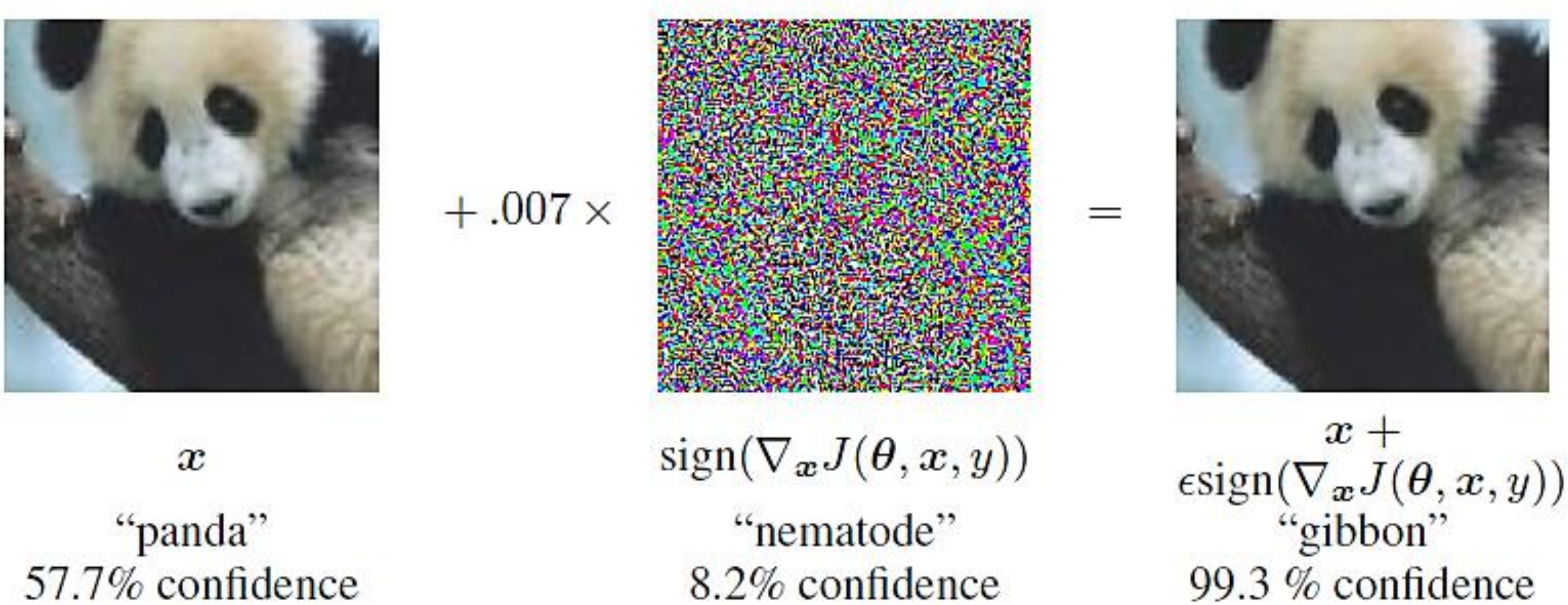
$$x_{adv} = x + \epsilon \cdot \text{sign}\big(\nabla_x \mathcal{L}(h(x, w), y)\big)$$

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

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

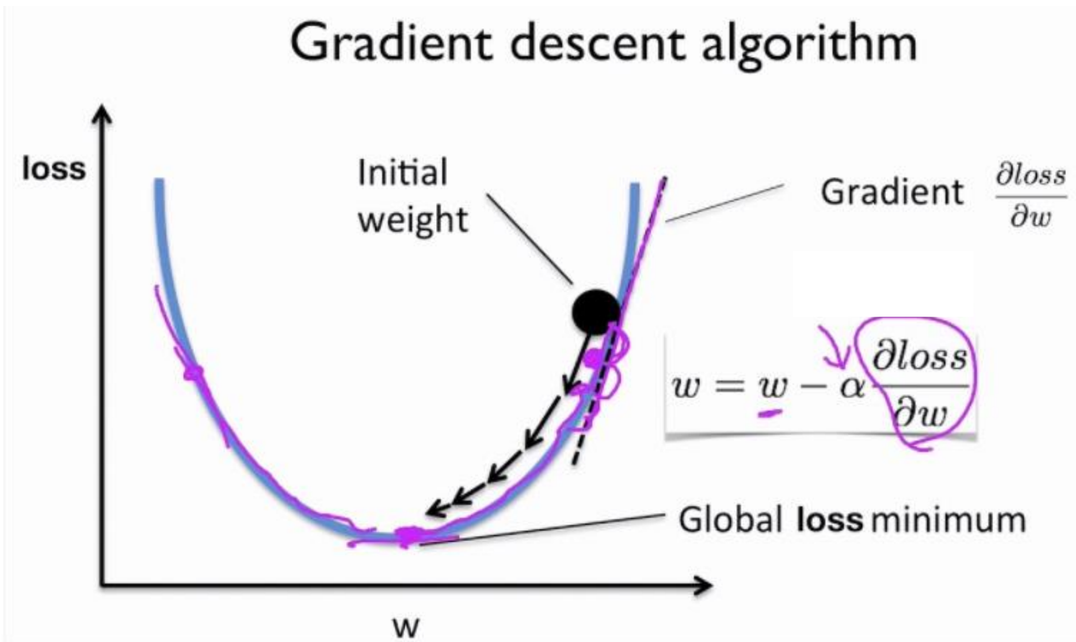
FGSM Attack

- FGSM is a white-box non-targeted evasion attack
  - White-box, since we need to know the gradients to create the adversarial image
  - The noise magnitude is ε = 0.007
    - Note: nematode is an insect referred to as roundworm



FGSM Attack

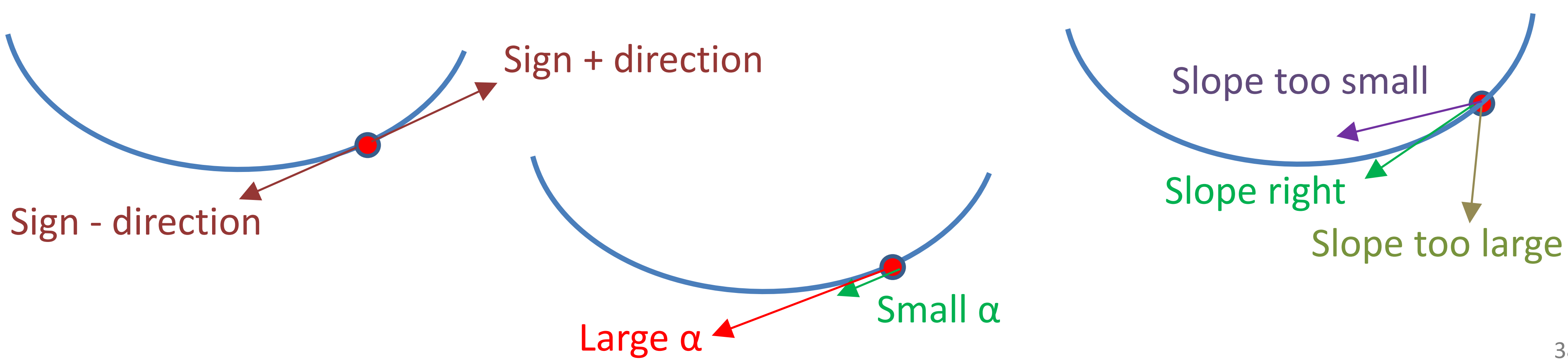
- Recall that training NNs is based on the gradient descent algorithm
  - The values of the network parameters (weights) w are iteratively changed until a minimum of the *loss* function is reached
  - Gradients of the loss function with respect to the model parameters ($\partial l/\partial w$) give the 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

- The sign and magnitude of the gradient 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 changed in the opposite direction of the gradient, i.e., $w = w - \alpha \frac{\partial loss}{\partial w}$



FGSM Attack

- 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

- ***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 α is the amount of noise that is added at each step

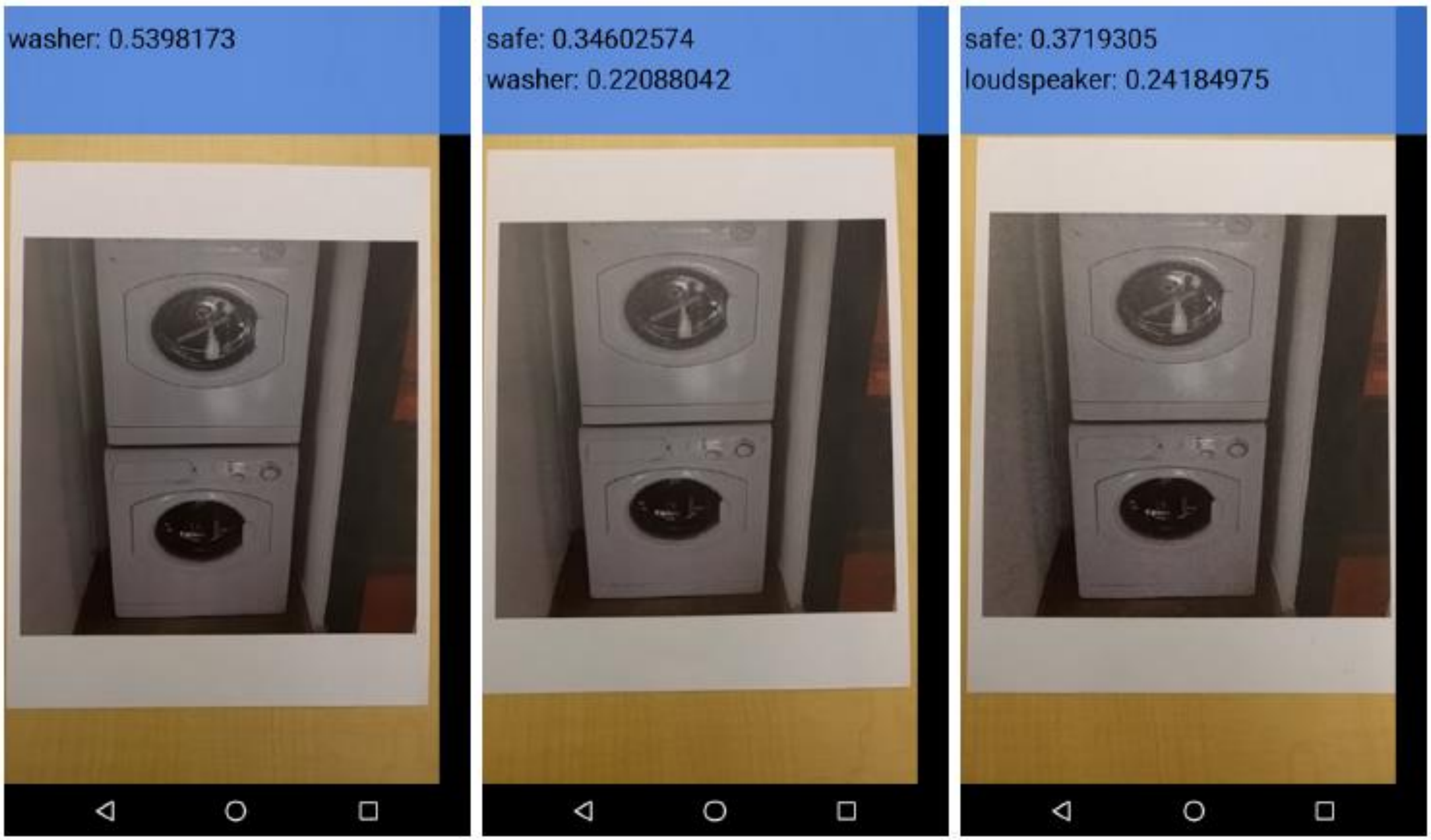
$$x_{adv}^{t} = x^{t-1} + \alpha \cdot \mathrm{sign}\big(\nabla_x \mathcal{L}(h(x^{t-1}), y)\big)$$

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

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

BIM Attack

- BIM attack example, cell phone image



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

PGD Attack

- ***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 step of perturbation, the adversarial example is projected back onto the ϵ-ball of x using a projection function Π

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

- 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 of the loss function with respect to the input

PGD Attack

- 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

- Gradient approaches can also be designed as targeted white-box attacks
  - The added perturbation noise aims to minimize the loss function of the image for a specific class label
    - In this 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

Original image

Prediction: hippopotamus

Adversarial image

Prediction: maraca

Maraca

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

PGD Attack

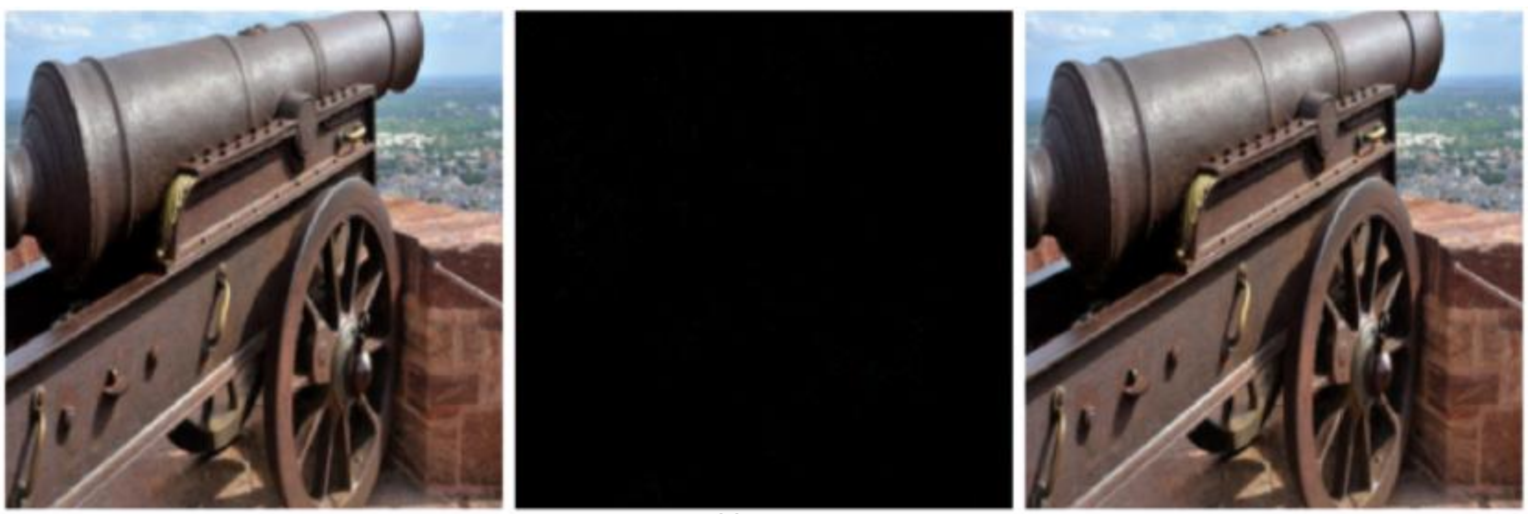
- For a targeted attack, if the target class label is denoted t, adversarial examples are created by using

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

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

DeepFool Attack

- *DeepFool attack*
  - Moosavi-Dezfooli (2015) DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks
- DeepFool is an untargeted white-box attack
  - It mis-classifies the image with the minimal amount of perturbation possible
  - There is no visible change to the human eye between the two images



Prediction: canon Difference Prediction: Projector

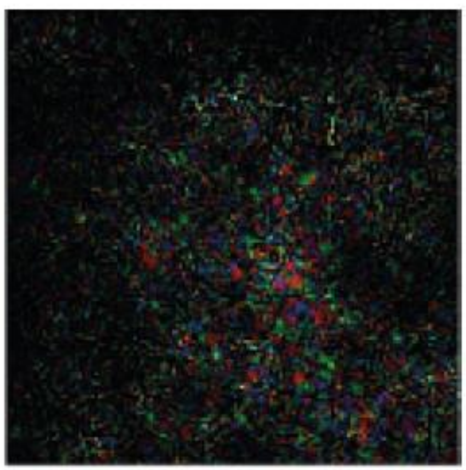
Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

DeepFool Attack

- Image example
  - Original image: whale
  - Both DeepFool and FGSM perturb the image to be classifier as turtle
  - DeepFool leads to a smaller perturbation

DeepFool

Prediction: Turtle



Difference

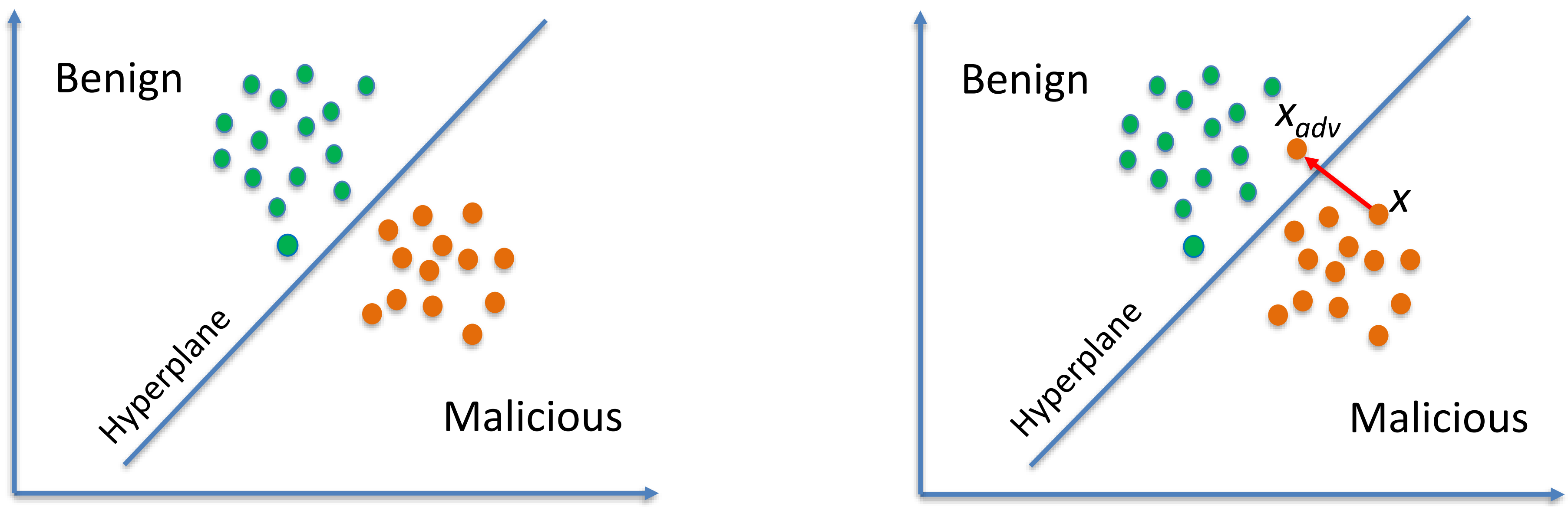
FGSM

Prediction: Turtle

Difference

DeepFool Attack

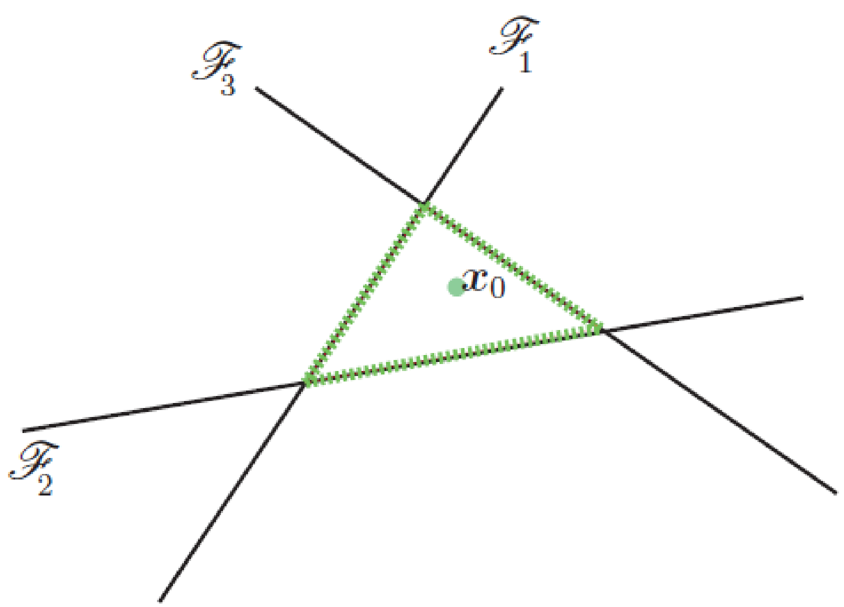
- E.g., consider a linear classifier algorithm applied to objects from 2 classes: green and orange circles
  - The line that separates the 2 classes is called the ***hyperplane***
    - Data points falling on either sides of the hyperplane are attributed to different classes (such as benign vs. malicious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



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

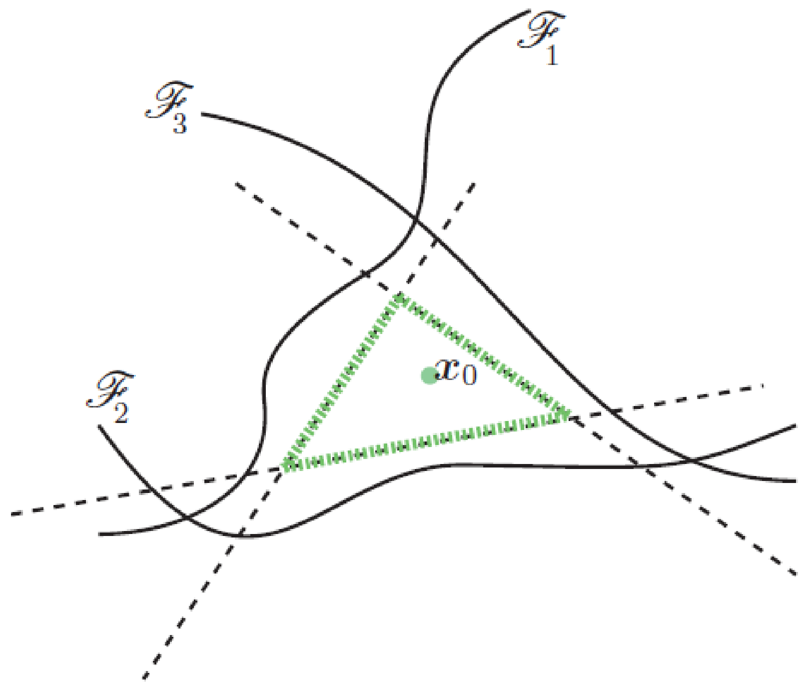
DeepFool Attack

- 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 is shown in the image below
- DeepFool finds that closest hyperplane to the input x_0, in this case the hyperplane $\mathcal{F}_3$ (most similar class of the other 3 classes)
  - Then, projects the input and pushes it a little beyond the hyperplane



DeepFool Attack

- For non-linear classifiers (such as neural networks), the authors perform several iterations of adding perturbations to the image
  - At each iteration, the classifier function is linearized around the current image, and a minimal perturbation is calculated
  - The algorithm stopes when the class of the image change to another label than the true class



Carlini Wagner (CW) Attack

- ***Carlini-Wagner (CW) 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 propose a reformulation of it 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 (CW) Attack

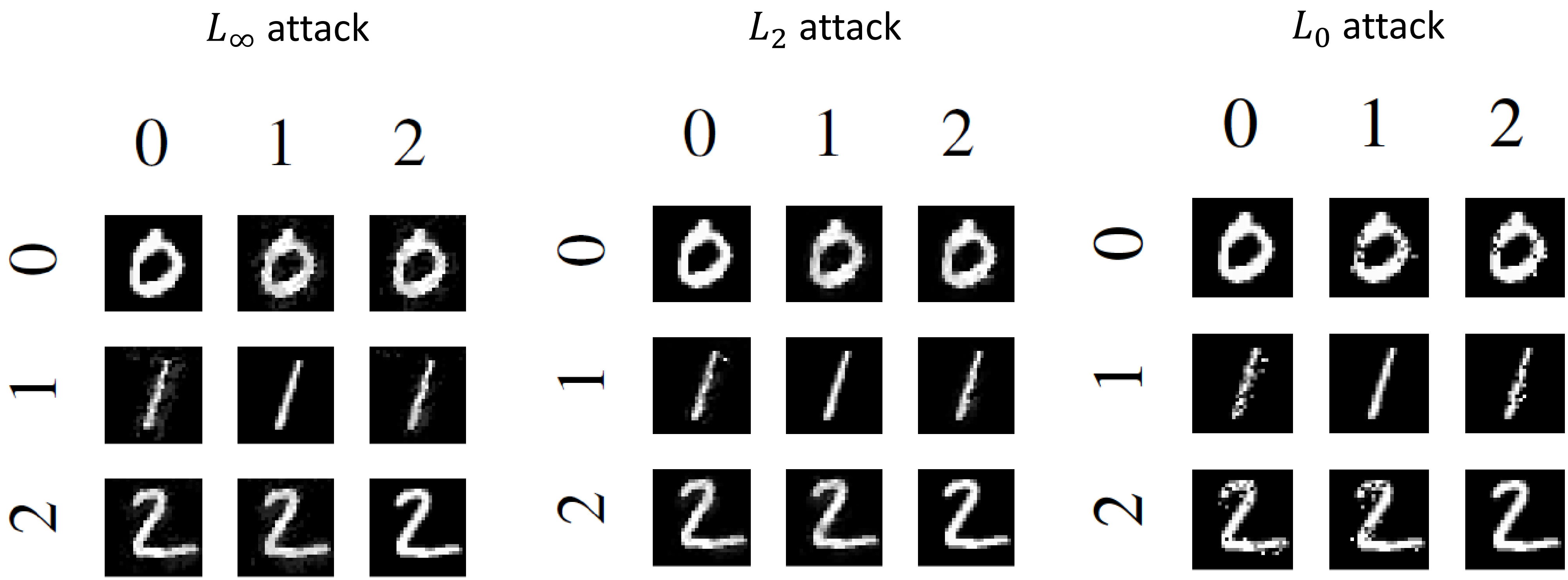
- The authors considered several variants for the function f

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

- The best results were obtained by f_6

Carlini Wagner (CW) Attack

- Results on the MNIST dataset



Evasion Attacks on Black-Box Models

- Adversarial example transferability
  - Cross-model transferability: the same adversarial example is often misclassified by a variety of classifiers with different architectures
  - Cross-training set transferability: the same adversarial example is often misclassified trained on different subsets of the training data
- Therefore, an attacker can take the following steps to reverse-engineer the classifier:
  1. Train his own (white-box) substitute model
  2. Generate adversarial samples
  3. Apply the adversarial samples to the target ML model

Defense Against Adversarial Attacks

- Adversarial samples can cause any ML algorithm to fail
  – However, they can be used to build more accurate and robust models
- 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 (i.e., the models don't have adversarial examples, or the adversaries cannot find them easily)
- Defense strategies against adversarial attacks include:
  – Adversarial training
  – Detecting adversarial examples
  – Gradient masking
  – Robust optimization (regularization, certified defenses)
- A list of adversarial defenses can be found at this link

Adversarial Training

- Learning the model parameters using adversarial samples is referred to as ***adversarial training***
- The training dataset is augmented with adversarial examples produced by known types of attacks
  - For each training input add an adversarial example
- However, if a model is trained only on adversarial examples, the accuracy to classify regular examples will reduce significantly
- Possible strategies:
  - Train the model from scratch using regular and adversarial examples
  - Train the model on regular examples and afterward fine-tune with adversarial examples

Adversarial Training

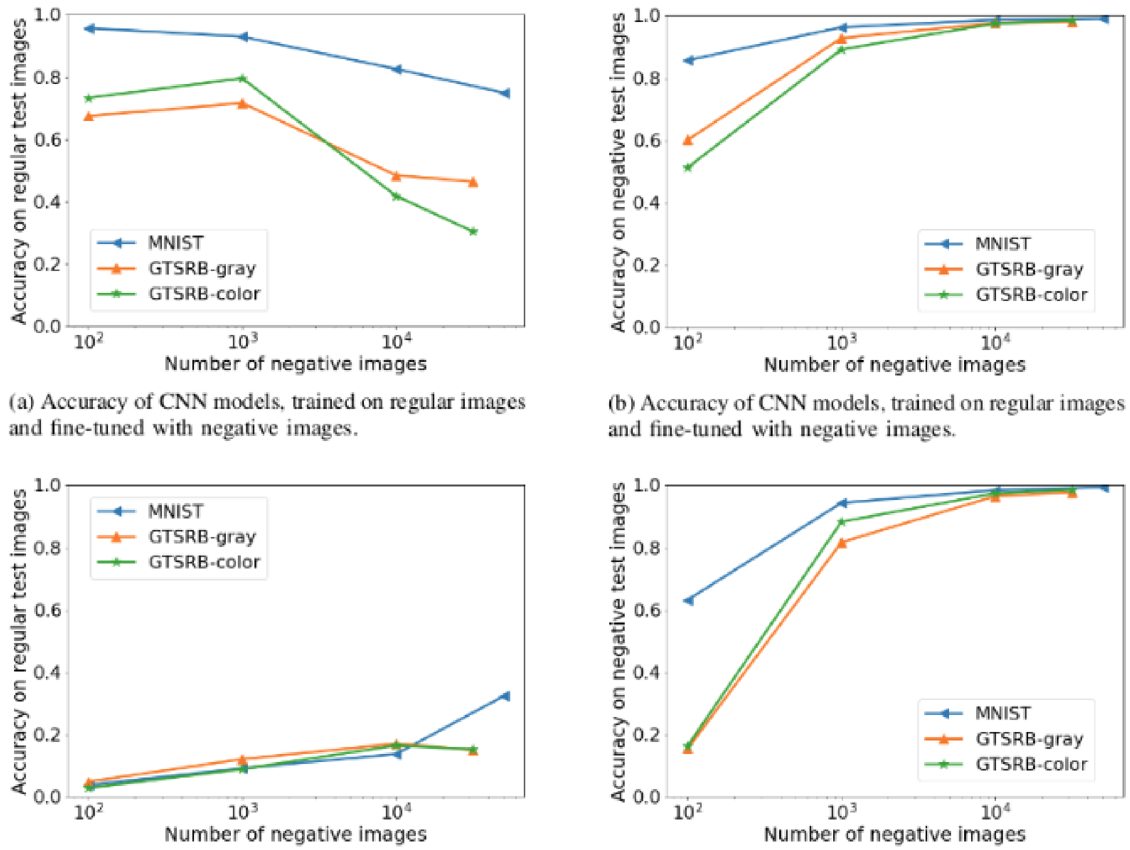
- Training with and without negative images for semantic attack

Fine-tuned

Accuracy on regular images

Trained from scratch

Accuracy on negative 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.

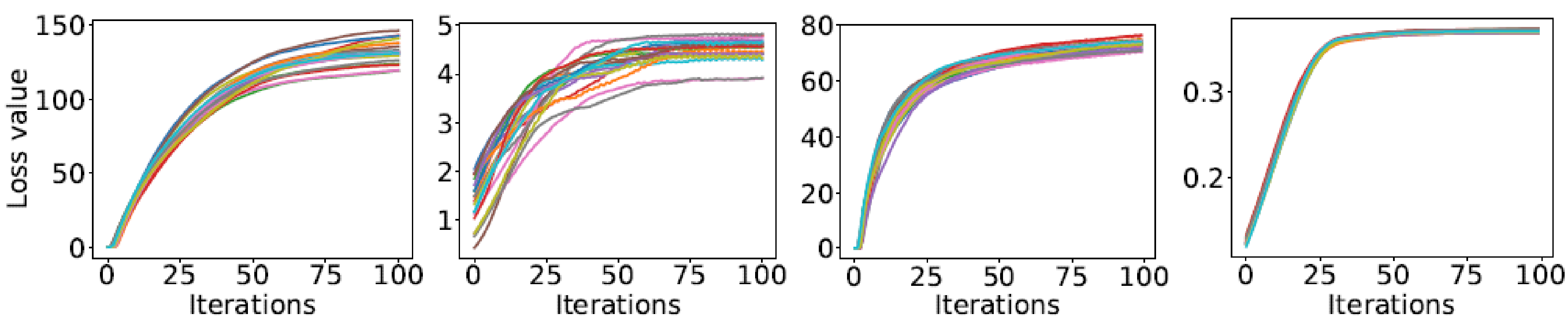
(d) Accuracy of CNN models trained with negative images from scratch.

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.

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

Adversarial Training

- The plots show the cross-entropy loss values for standard and adversarial training on MNIST and CIFAR10 datasets while creating adversarial examples using PDG attack (Madry, 2018)
  - 20 runs are shown, each starting at a random point within a perturbation range $(-\epsilon, \epsilon)$
  - The final loss values on adversarially trained models are much smaller than on the original training datasets

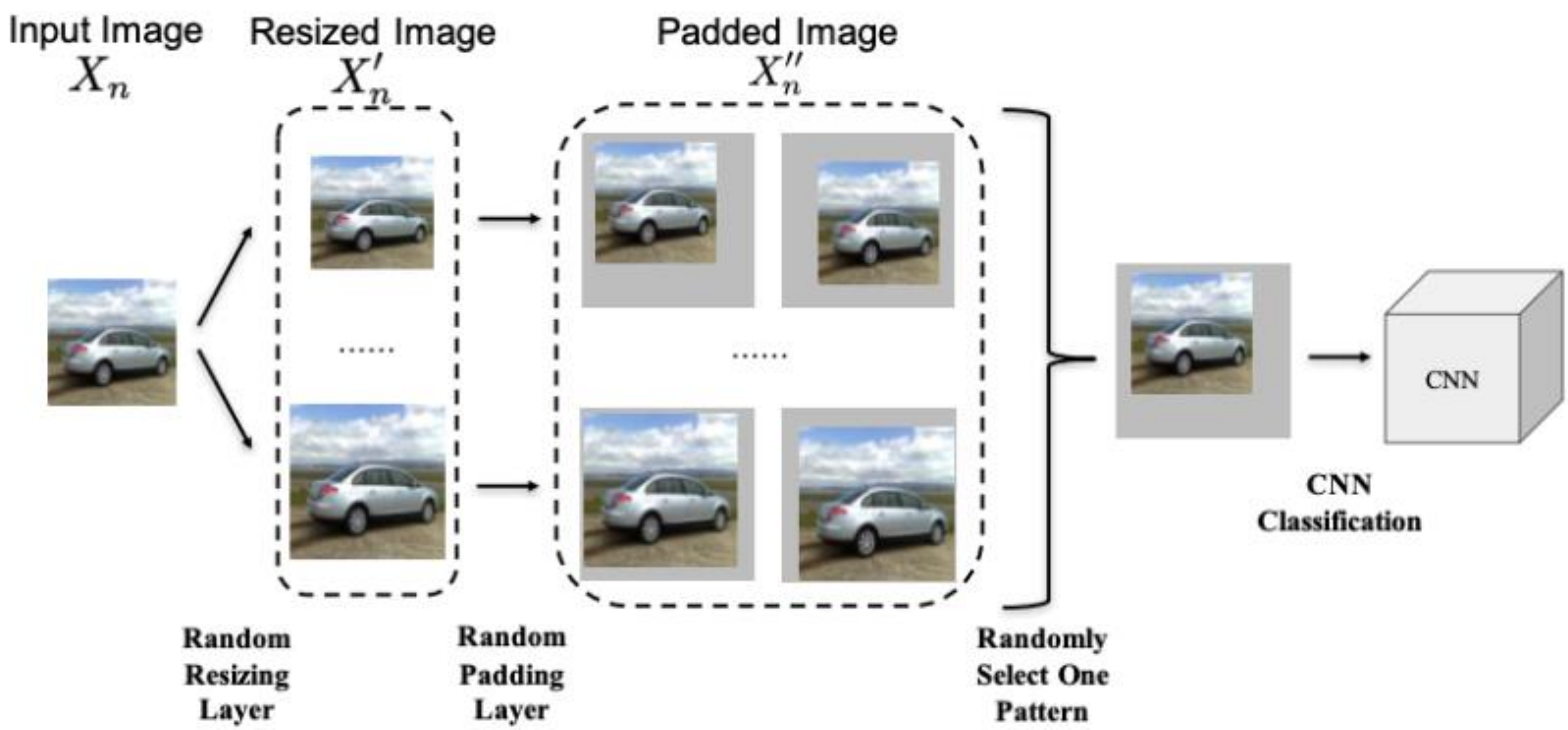


(a) MNIST Standard training (b) MNIST Adversarial training (c) CIFAR10 Natural training (d) CIFAR10 Adversarial training

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

Random Resizing and Padding

- Model training with randomly resizing the image and applying random padding on all four sides have shown to improve the robustness to adversarial attacks
  - Xie (2018) – Mitigating Adversarial Effects Through Randomization



Detecting Adversarial Examples

- A body of work focused on distinguishing 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
- ***Example detection*** defense methods
  - Kernel Density (KD) detector based on Bayesian uncertainty features
    - Feinman (2017) – Detecting Adversarial Samples from Artifacts
  - Local Intrinsic Dimensionality (LID) of adversarial subspaces
    - Ma (2018) - Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality
  - Adversary detection networks
    - Metzen (2017) On detecting adversarial perturbations

Gradient Masking

- *Gradient masking* defense methods deliberately hide the gradient information of the model
  - Since most attacks are based on the model's gradient information
- 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 neural networks
- Input preprocessing by discretization of image's pixel values, or resizing and cropping, or smoothing
  - Buckman (2018) Thermometer encoding: One hot way to resist adversarial examples
- DefenseGAN – uses a GAN model to transform perturbed images into clean images
  - Samangouei (2017) Defense-GAN: Protecting classifiers against adversarial attacks using generative models

Robust Optimization

- ***Robust optimization*** aims to evaluate, and improve, the model robustness to adversarial attacks
  - Consequently, learn model parameters that minimize the misclassification of adversarial examples
- Regularization methods – train the model by penalizing large values of the parameters, or large values of the gradients
  - Cisse (2017) Parseval networks: Improving robustness to adversarial examples
- Certified defenses – for a given dataset and model, find the lower bound of the minimal perturbation: the model will be safe against any perturbations smaller than the lower bound
  - Raghunathan (2018) Certified defenses against adversarial examples

Conclusion

- ML algorithms and methods are vulnerable to many types of attacks
- Adversarial examples show its transferability in ML models
  - I.e., either cross-models or cross-training sets
- Adversarial examples can be leveraged to improve the performance or the robustness of ML models

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)

Other AML Recourses

- Cleverhans - a repository from Google that implements latest research in AML
  - The library is being updated to support TensorFlow2, PyTorch, and Jax
- 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.
- 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