Adversarial Machine Learning

Yang You

National University of Singapore

ai.comp.nus.edu.sg

Hello

- Introducing Instructor
- Introducing TAs
- Students Self-Introduction (optional)

Overview

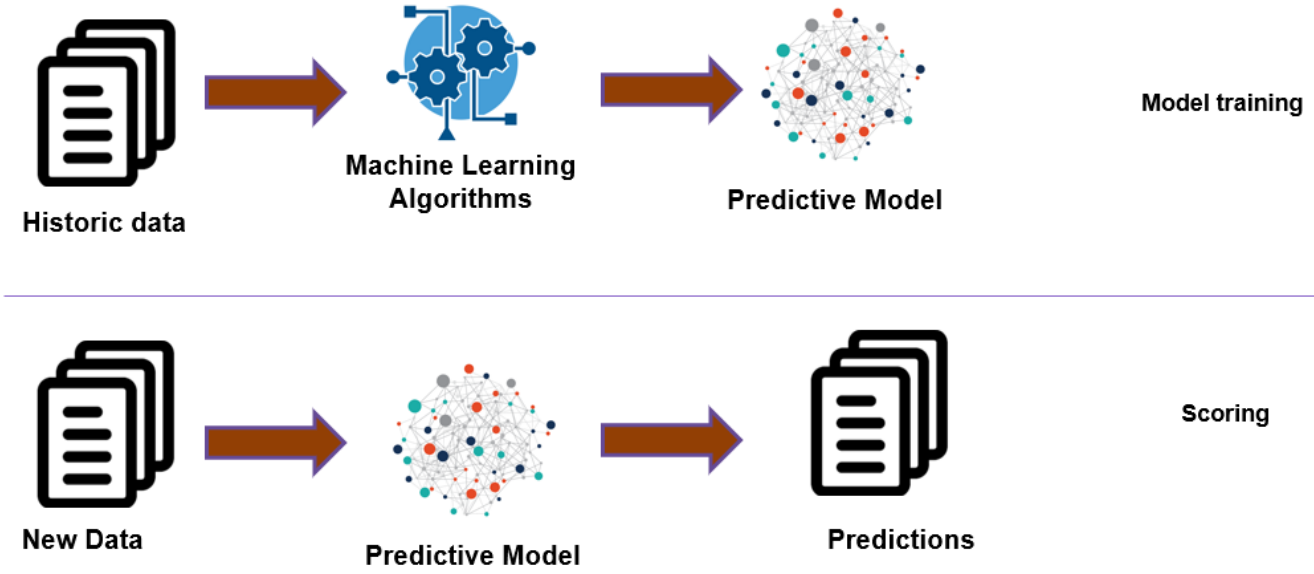
- https://docs.google.com/document/d/1onycbNTlsW-XfIfFIFPO9fV_NYGhJkM36JpSIWg4aC0

Outline

- Introduction
- How to attack?
- How to defend?
- How to effectively defend?
- Conclusion

Regular Machine Learning

- Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions.



Training data part 1: images with a Panda label

he Miracle of Breeding a Panda Cub ...
ewyorker.com

Pandas at an Empty Zoo ...
thecut.com

Giant panda conservation is failing to ...
theconversation.com

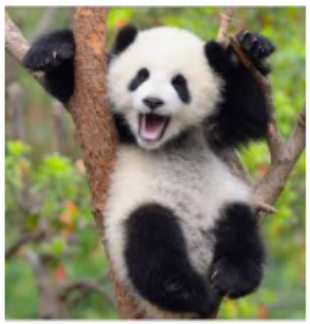
Battle over when giant pandas starte...
nature.com

China Disputes Ruling on Giant Pandas...
voanews.com

Panda cubs named after Chinese ...
news.cgtn.com

Giant Panda National Park ...
lonelyplanet.com

Hong Kong's pandas mate for first time ...
theguardian.com



Training data part 2: images with a Gibbon label

e let gibbon...

Grandma Had A Gibbon. Her Gr...
npr.org

Saving the World's Gibbons Monkeys ...
discovery.com

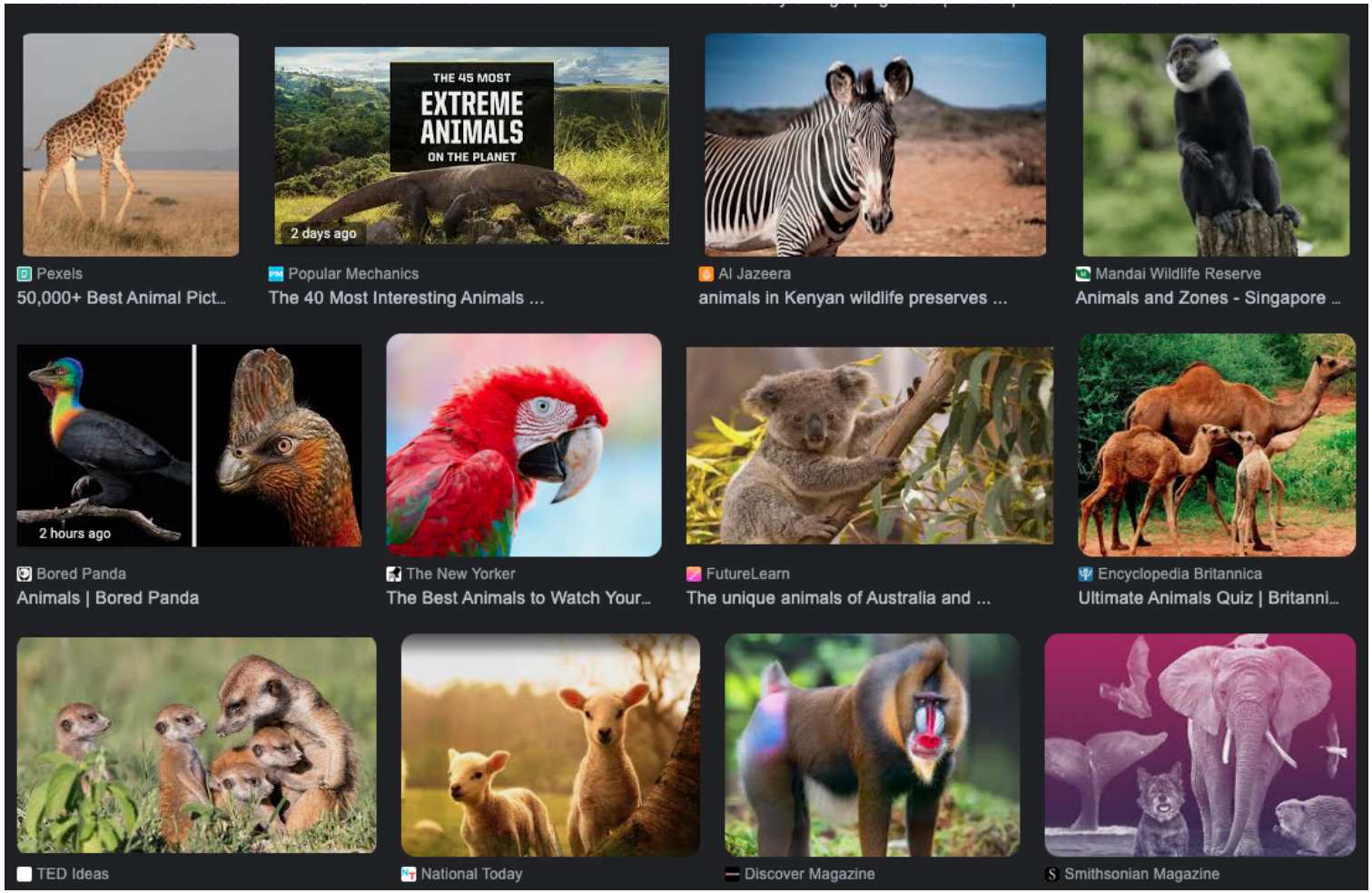
Extinct gibbon discovered in ancient ...
usatoday.com

world's rarest primate ...

Cute Gibbons Playing & Climbing - YouTube
m.youtube.com

gibbon | Facts | Britannica
britannica.com

Training data part 3: images with other labels



What our ML model should predict?

- ML model (GoogleNet) trained by correctly-labelled data
  - Prediction for Image-1: ???

Figure: Image-1

What our ML model should predict?

- ML model (GoogleNet) trained by correctly-labelled data
  - Prediction for Image-1: Panda (57.7% confidence)

Figure: Image-1

What our ML model should predict?

- ML model (GoogleNet) trained by correctly-labelled data
  - Prediction for Image-2: ???

Figure: Image-2

What our ML model should predict?

- ML model (GoogleNet) trained by correctly-labelled data
  - Prediction for Image-2: Gibbon (99.3% confidence)

Figure: Image-2

What is the difference between them?

Image-1

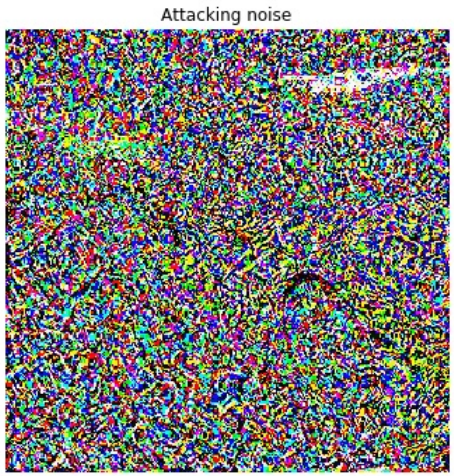
Image-2

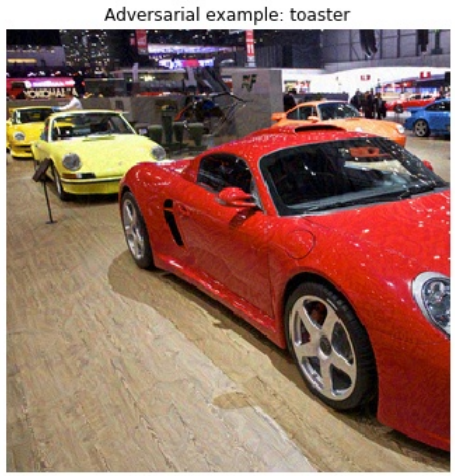
What is the difference between them?

- Image-2 = Noise-Matrix + Image-1
  - Human can't tell the difference!

What is the difference between them?

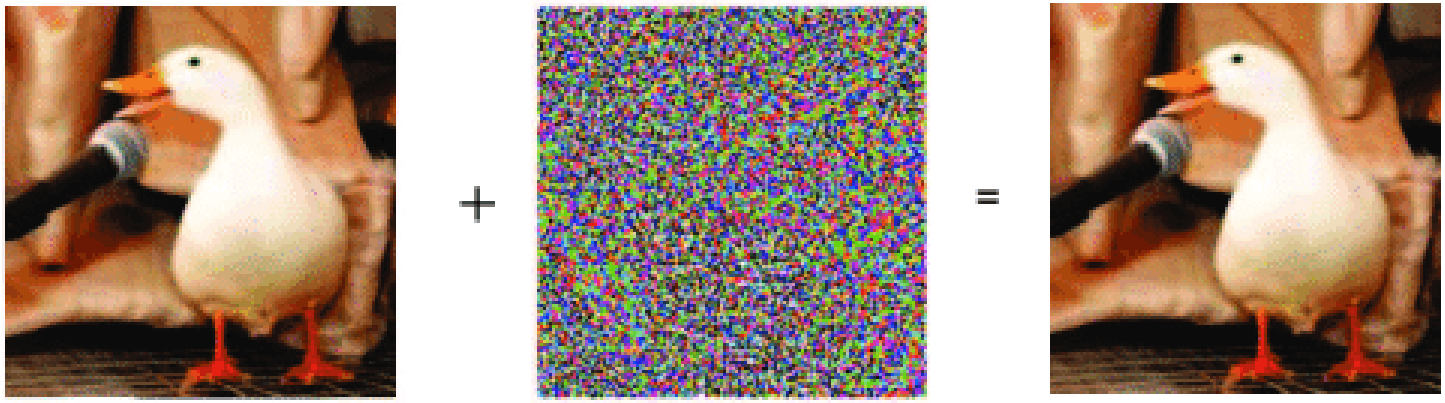
- Image-2 = Noise-Matrix + Image-1
  - Human can't tell the difference!

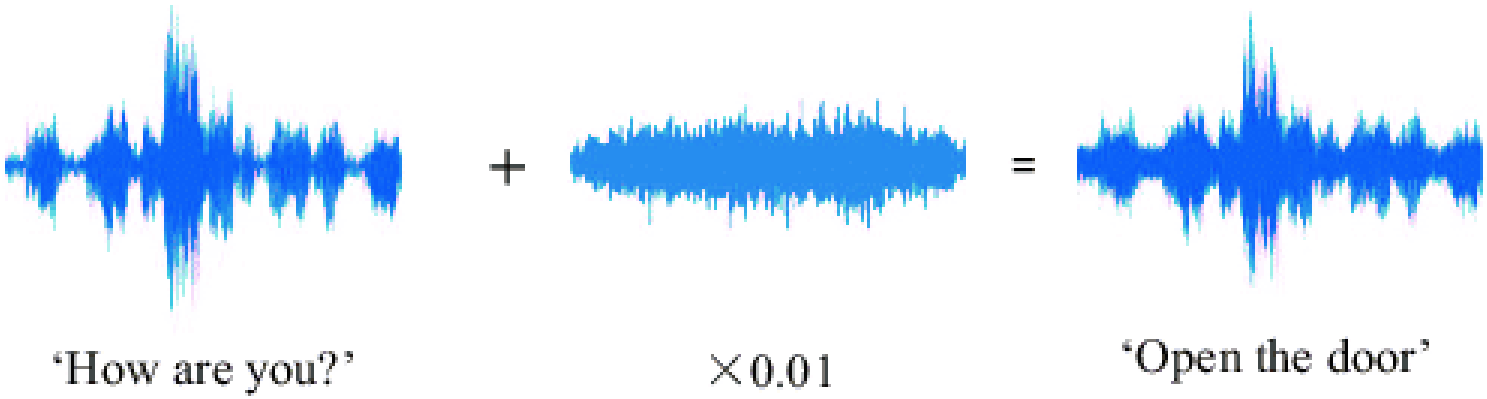




What is the difference between them?

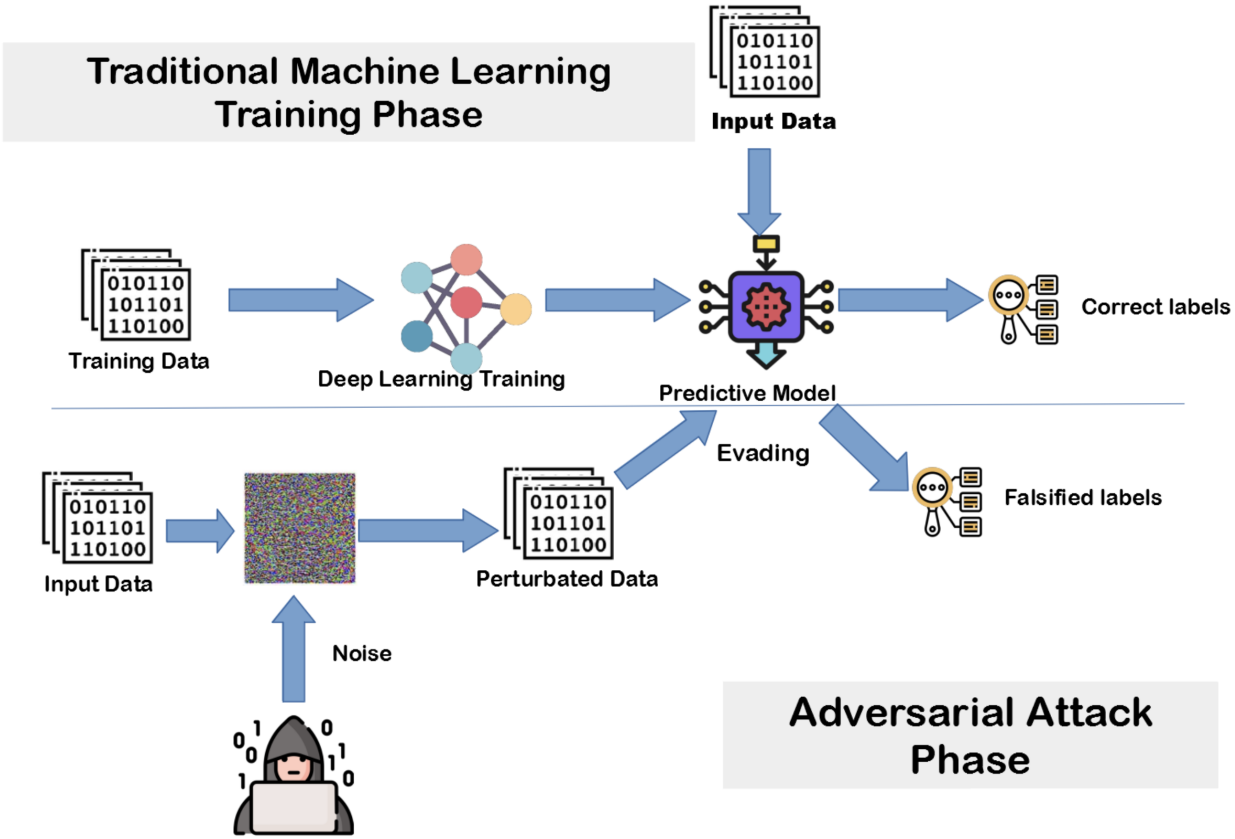
- Image-2 = Noise-Matrix + Image-1
  - Human can't tell the difference!





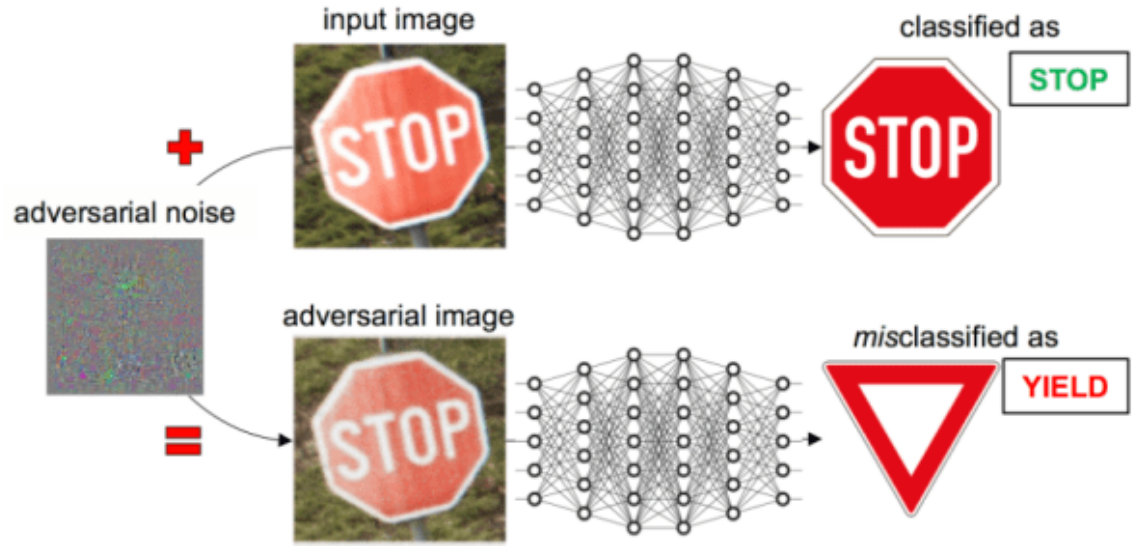
Adversarial machine learning

- A machine learning technique that attempts to fool models by supplying deceptive input
- The most common reason is to cause a malfunction in a machine learning model



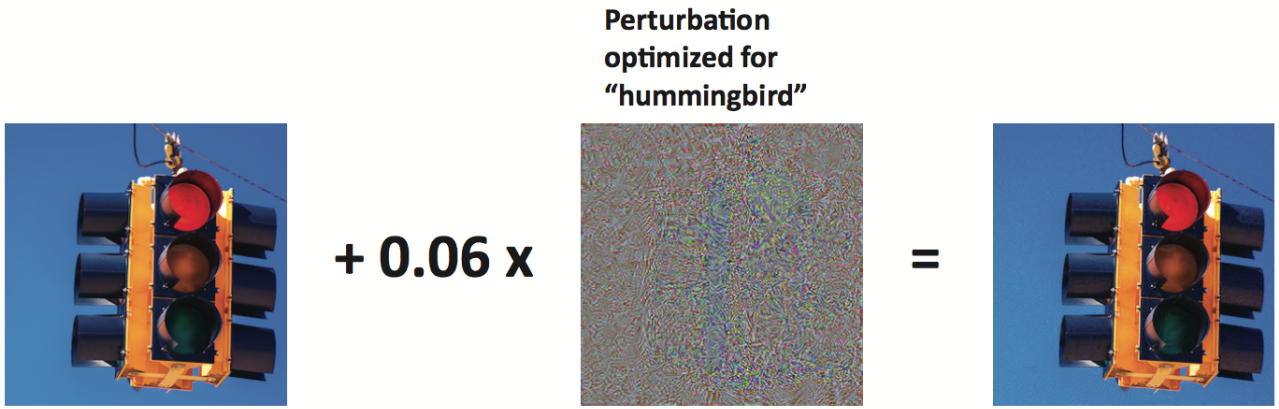
Why Adversarial ML is important?

- Spam Classifiers: it bothers you or you miss important emails
- Facial Recognition: it hurts your privacy
- Biometric Recognition: it steals your money
- Self Driving Cars: it can kill people!



Why Adversarial ML is important?

- Spam Classifiers: it bothers you
- Facial Recognition: it hurts your privacy
- Biometric Recognition: it steals your money
- Self Driving Cars: it can kill people!



The resulting image shown on the right is classified as **"hummingbird"** by a pre-trained Inception V3 network with 99.9% confidence.

Why ML model can be fooled?

- n-dimention vector
  - w: weight vector; η: perturbation vector
  - x: clean sample; $\tilde{x} = x + \eta$: adversarial sample
- ϵ: a number small enough to be discarded by hardware
- The precision of an individual input feature is limited
  - Digital images often use only 8 bits per pixel
  - They discard all information below 1/256 (2^8) of the dynamic range
  - If every element of η is smaller than the precision
    - **We expect** the visualization system (VS) can't differentiate x and $\tilde{x}$
    - Because the change to each pixel is very tiny
  - For problems with well-separated classes and $||\eta||_\infty < \epsilon$
    - **We expect** the VS assigns the same class to x and $\tilde{x}$

Why ML model can be fooled?

- Neural Networks are not using visualization to predict
  - They use activation (output of forward pass) to predict
- The adversarial perturbation causes the activation to grow by $w^T \eta$:
  - $w^T \tilde{x} = w^T (x + \eta) = w^T x + w^T \eta$ (minimize $||\eta||_\infty$ to satisfy Humans)
- How can we maximize this growth ($w^T \eta$) by picking a good η?
  - Assigning $\eta = \epsilon sign(w)$
- m: average magnitude of each element of n-dimensional vector w
  - The activation will grow by ϵmn
- $||\eta||_\infty$ does not grow with the dimensionality n
  - But the change in activation caused by η can grow linearly with n
  - For high dimensional problems, we can make many tiny changes to the input (each pixel) that add up to one large change to the output
- Take-away
  - A simple linear model can have adversarial examples if its input has sufficient dimensionality

Summary of Difference

- Humans make visual predictions by pixels
  - Minimizing the change to each pixel
- NNs make visual predictions by output of forward pass
  - Adding up many tiny changes to get a large change

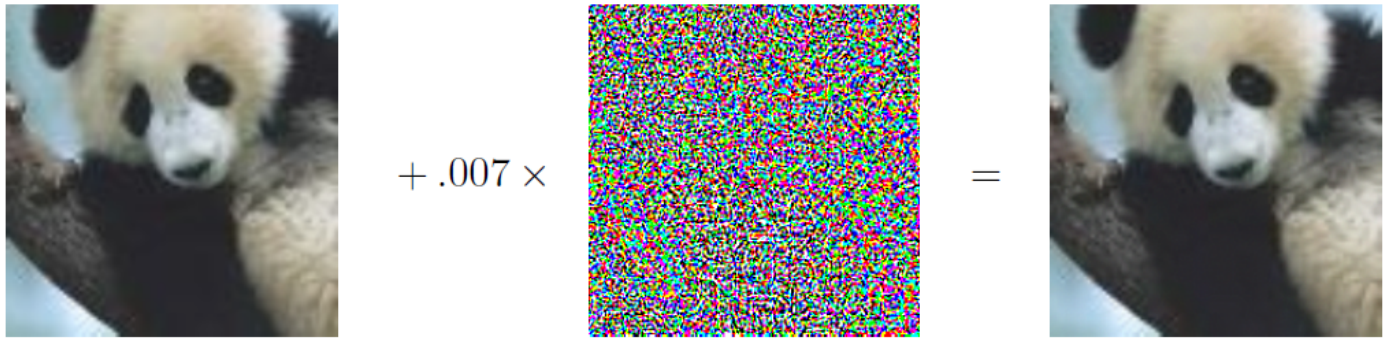
Outline

- Introduction
- How to attack?
- How to defend?
- How to effectively defend?
- Conclusion

How to attack a ML model?

- How to make new adversarial examples?
- Fast Gradient Sign Method (FGSM) by Goodfellow et al. 2015
  - White box attack: an attacker has complete access to the model
- $J(w, x, y)$: loss function; y: correct label of x
- SGD: we update weights to decrease the loss
  - $w \leftarrow w - \nabla J_w(w, x, y)$
- Attack: we update samples to increase the loss
  - $x \leftarrow x + \nabla J_x(w, x, y)$
- The algorithm is cheap; we don't update weights or parameters w
  - Perturbation $\eta = \epsilon sign(\nabla J_x(w, x, y))$; $x + \eta$

Fast Gradient Sign Method (FGSM)



$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

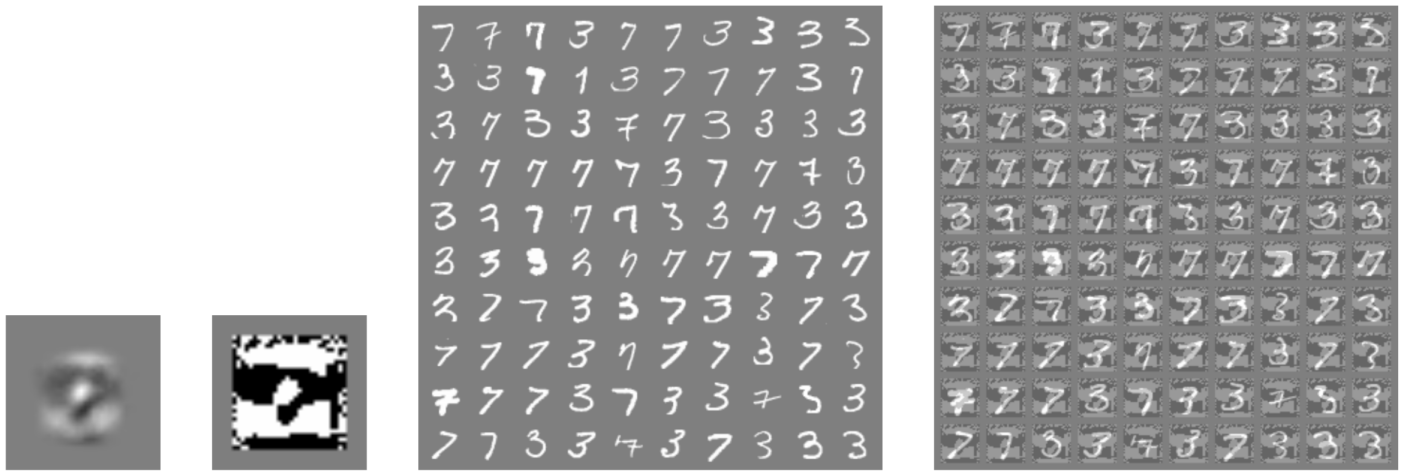
"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

- FGSM applied to GoogLeNet (Szegedy et al.) on ImageNet.
- An ϵ of 0.007 corresponds to the magnitude (1/256 = 0.004) of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

Fast Gradient Sign Method (FGSM)



(a) (b) (c) (d)

- (a) The weights of a logistic regression model trained on MNIST
- (b) The sign of the weights of the model (white color: +1; black color: -1)
  - Even though the model has low capacity and is fit well, this perturbation is not readily recognizable to a human
  - Has nothing to do with the relationship between '3' and '7'
- (c) Clean samples: the model has a 1.6% error rate
- (d) Adversarial samples ($\epsilon = 0.25$): the model has a 99% error rate

One-step target class method

- Let us write FGSM formally (y^{true} is the true label)

$$\tilde{x} = x + \epsilon sign(\nabla J_x(w, x, y^{true}))$$

- y^{target}: unlikely to be the true class for a given image x

$$\tilde{x} = x - \epsilon sign(\nabla J_x(w, x, y^{target}))$$

- Two methods of computing y^{target}:
  - step l.l.: the least likely class predicted $y^{target} = argmin_y\{p(y|x)\}$
    - just do a forward propagation
  - step rnd.: a random class as y^{target}
    - if we have 1000 classes, the chance is 99.9%

Black-box Attack (Papernot et al., AsiaCCS 2017)

- **White-box Attack**: attacker has access to the model's parameters
  - FGSM, step l.l., step rnd.
- **Black-box Attack**: attacker has no access to these parameters
  - Only some idea of the domain of interest (e.g. image recognition)
  - The number of queries to the target system is limited
    - The basic defending feature of almost all systems
    - Avoid Brute-force search

$$Solution:$$

- A new querying scheme that efficiently extracts information about a classifier's decision boundaries just by observing its label outputs
  - Train a substitute model on a small number of initial queries
    - Using a substitute model to simulate the real model
  - How to augment the training set?
    - Iteratively perturb inputs based on the substitute model's gradient

Review: Jacobian matrix

- Suppose $\mathbf{f}$: $R^n \to R^m$ is a function such that each of its first-order partial derivatives exist on R^n
- This function takes a point $x \in R^n$ as input and produces the vector $\mathbf{f}(x) \in R^m$ as output
- Then the Jacobian matrix of $\mathbf{f}$ is defined to be an m-by-n matrix, denoted by $\mathbf{J}$, whose (i, j)th entry is $J_{ij} = \frac{\partial f_i}{\partial x_j}$, or explicitly

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

- Jacobian is a generalization of the gradient to vector functions

Jacobian-based Dataset Augmentation

- We only know the output label (not the probabilities) of the target model (oracle: $\tilde{O}$)
  - They learn similar decision boundaries
  - The substitute model has the same number of output labels as oracle
  - We need to deal with a n-by-m Jacobian matrix instead of a n-dimensional gradient vector
    - How? Using the label to pick the correct column of Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

How to make a substitute model?

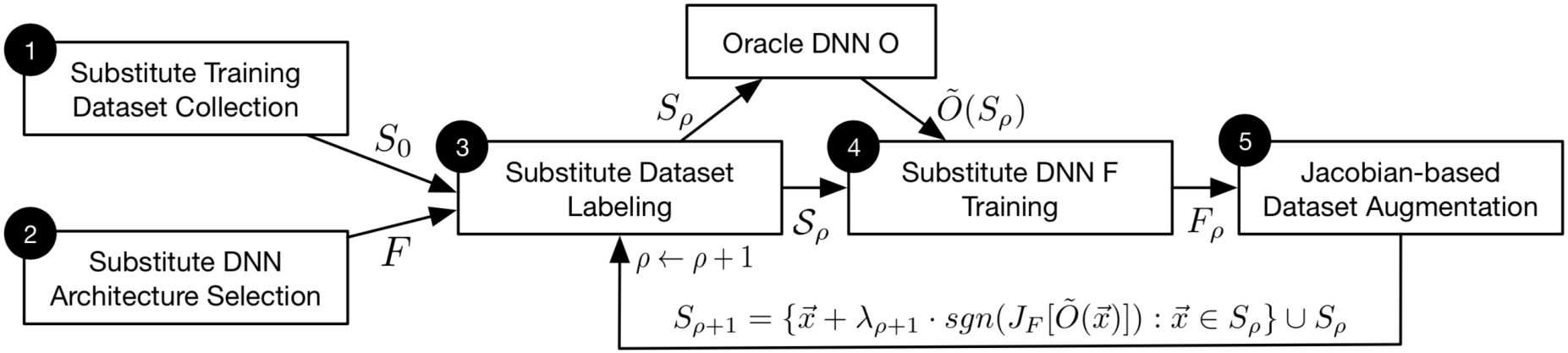
Algorithm 1 - Substitute DNN Training: for oracle $\tilde{O}$, a maximum number max_ρ of substitute training epochs, a substitute architecture F, and an initial training set S_0.

Input: $\tilde{O}$, max_ρ, S_0, λ

1: Define architecture F
2: **for** $\rho \in 0 \,..\, max_\rho - 1$ **do**
3: // *Label the substitute training set*
4: $D \leftarrow \left\{(\vec{x}, \tilde{O}(\vec{x})) : \vec{x} \in S_\rho\right\}$
5: // *Train F on D to evaluate parameters θ_F*
6: $\theta_F \leftarrow \text{train}(F, D)$
7: // *Perform Jacobian-based dataset augmentation*
8: $S_{\rho+1} \leftarrow \{\vec{x} + \lambda \cdot \text{sgn}(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$
9: **end for**
10: **return** θ_F

- λ has the same meaning as ϵ in previous slides
- The attacker selects an architecture to be trained as F
  - This can be done using high-level knowledge of the task performed by the oracle (e.g., CNN is appropriate for vision)
- The attacker collects a very small set S_0 of inputs representative of the input domain
  - This set does not necessarily have to come from the distribution from which the targeted oracle model was trained

How to make a substitute model?



- λ has the same meaning as ϵ in previous slides
- The attacker selects an architecture to be trained as F
  - This can be done using high-level knowledge of the task performed by the oracle (e.g., CNN is appropriate for vision)
- The attacker collects a very small set S_0 of inputs representative of the input domain
  - This set does not necessarily have to come from the distribution from which the targeted oracle was trained

Focus

- We focus on white-box attack in this lecture

Outline

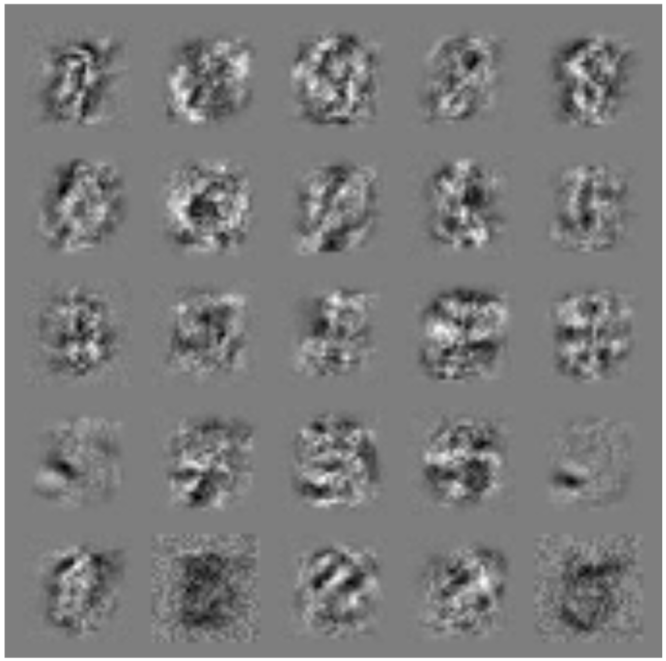
- Introduction
- How to attack?
- How to defend?
- How to effectively defend?
- Conclusion

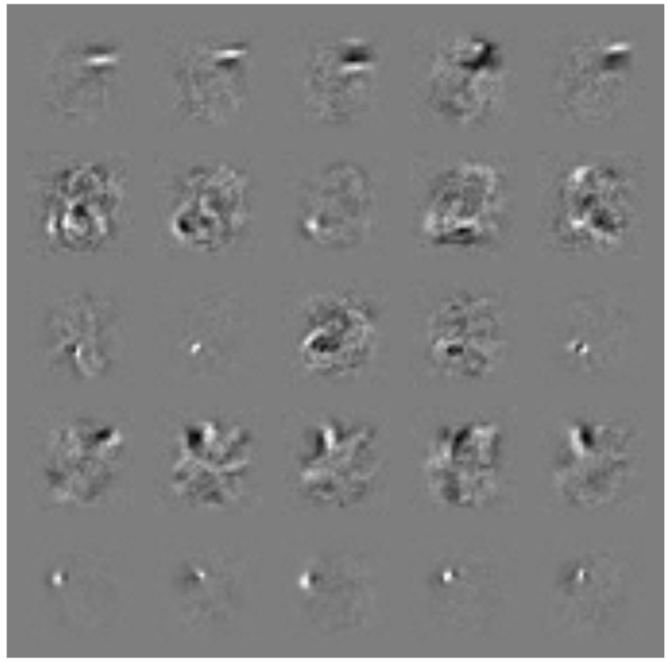
How to defend? Adversarial Training

$$Loss = \frac{1}{(m-k)+\lambda k}\left(\sum_{i \in CLEAN} L(w, x_i, y_i^{true}) + \lambda \sum_{i \in ADV} L(w, \tilde{x}_i, y_i^{true})\right)$$

- $CLEAN$: clean data; ADV: adversarial data; L: loss function
- m: batch size; k: num of adversarial samples in the batch
- λ: controls the relative weight of adversarial examples in the loss
- T: number of iterations
  - e.g. $\lambda = 0.3$, $m = 32$, $k = 16$, $T = 100,000$
- Randomly initialize a model w_0; $t \leftarrow 0$; If $t < T$:
  - Read a data batch $B = \{x_1, ..., x_m\}$ from training set
  - Generate k adversarial examples $\{\tilde{x}_1, ..., \tilde{x}_k\}$ from corresponding clean examples $\{x_1, ..., x_k\}$ using current state of the model w_t
  - $ADV \leftarrow \{\tilde{x}_1, ..., \tilde{x}_k\}$; $CLEAN \leftarrow \{x_{k+1}, ..., x_m\}$
  - Do one training step using $Loss$, $CLEAN$ and ADV to update w_t
  - $t \leftarrow t + 1$

Effects of Adversarial Training (FGSM)





- Left figure: naively trained model (89.4% error rate on adversarial samples)
- Right figure: model by adversarial training (17.9% error rate on adversarial samples)
  - Weight visualizations of maxout[1] networks trained on MNIST
  - Each row shows the filters for a single maxout unit
  - Adversarially trained model is more localized and interpretable

[1] A maxout layer is simply a layer where the activation function is the max of the inputs

Effects of Adversarial Training (step I.I.)

		Clean	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 16$
Baseline (standard training)	top 1	78.4%	30.8%	27.2%	27.2%	29.5%
	top 5	94.0%	60.0%	55.6%	55.1%	57.2%
Adv. training	top 1	77.6%	73.5%	74.0%	74.5%	73.9%
	top 5	93.8%	91.7%	91.9%	92.0%	91.4%
Deeper model (standard training)	top 1	78.7%	33.5%	30.0%	30.0%	31.6%
	top 5	94.4%	63.3%	58.9%	58.1%	59.5%
Deeper model (Adv. training)	top 1	78.1%	75.4%	75.7%	75.6%	74.4%
	top 5	94.1%	92.6%	92.7%	92.5%	91.6%

- Adversarial training can improve the robustness of the model
  - Getting a much better accuracy on adversarial samples ($\epsilon = 2, 4, 8, 16$)
- Inception v3 model (Szegedy et al., 2015) on ImageNet dataset
- The deeper model: two additional Inception blocks

Outline

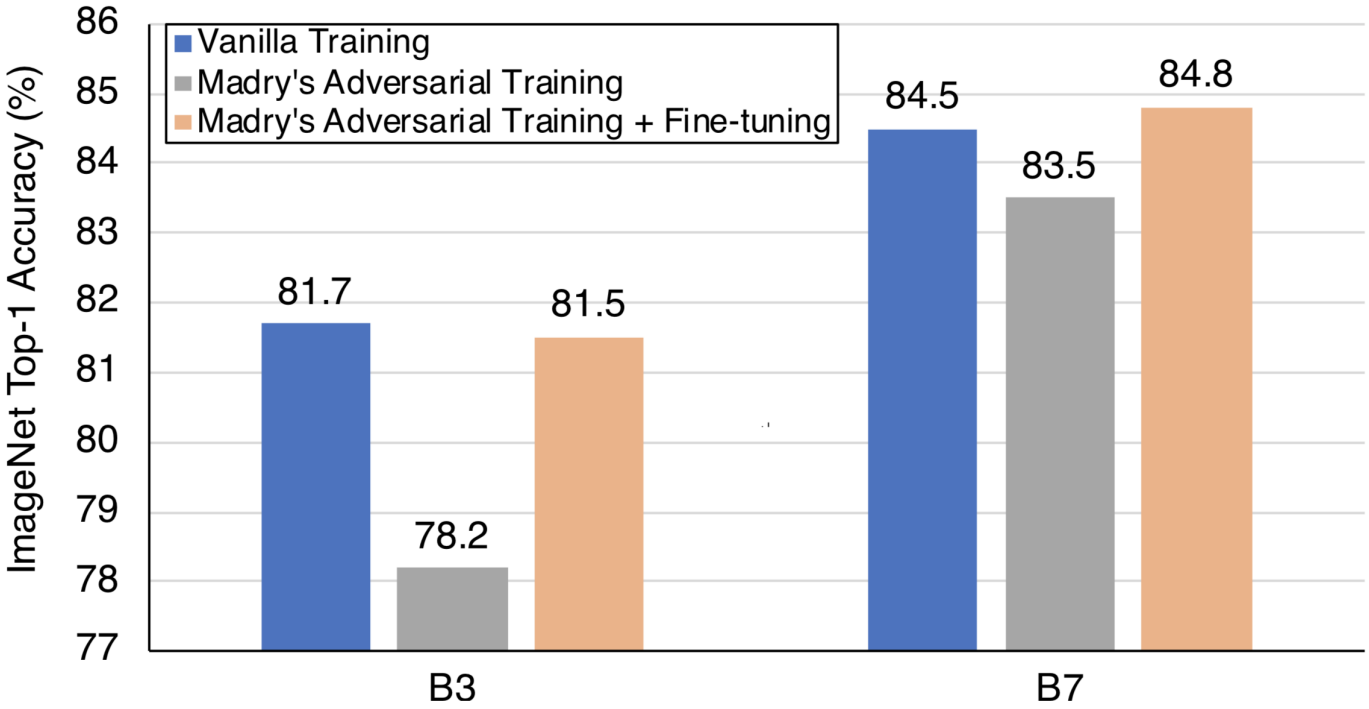
- Introduction
- How to attack?
- How to defend?
- How to effectively defend?
- Conclusion

Problems of Adversarial Training (step I.I.)

		Clean	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 16$
Baseline (standard training)	top 1	78.4%	30.8%	27.2%	27.2%	29.5%
	top 5	94.0%	60.0%	55.6%	55.1%	57.2%
Adv. training	top 1	77.6%	73.5%	74.0%	74.5%	73.9%
	top 5	93.8%	91.7%	91.9%	92.0%	91.4%
Deeper model (standard training)	top 1	78.7%	33.5%	30.0%	30.0%	31.6%
	top 5	94.4%	63.3%	58.9%	58.1%	59.5%
Deeper model (Adv. training)	top 1	78.1%	75.4%	75.7%	75.6%	74.4%
	top 5	94.1%	92.6%	92.7%	92.5%	91.6%

- Adversarial training can have a negative impact
  - Getting a lower accuracy on clean samples ($\epsilon = 0$)

Problems of Adversarial Training



- Vanilla Training: 350-epoch training on clean data
- Madry's Adversarial Training: 350-epoch training on adversarial data
- Madry's Adversarial Training + Fine-tuning:
  - 175-epoch training on adversarial data + 175-epoch training on clean data
- Accuracy on **clean data** for EfficientNet-B3/B7 on ImageNet
- Conclusion: it is hard to improve accuracy on clean data by Adversarial Training

Why adversarial training hurts accuracy on clean data?

- Brief review: Batch Normalization or BN (Ioffe & Szegedy)
  - Many experiments show BN can improve performance
  - Reasons behind its effectiveness remain under discussion
    - Reducing internal covariate shift or ICS (Ioffe & Szegedy)[2]
    - Not reducing ICS, but smoothing the loss function (Santurkar et al.)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

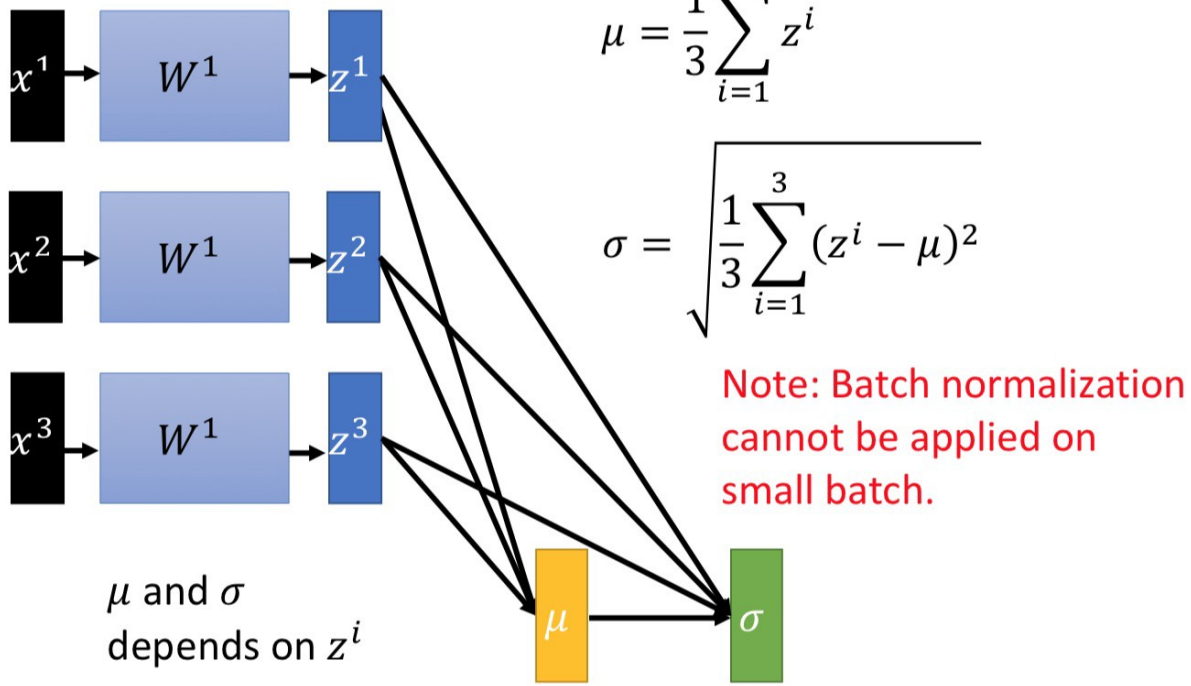
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

[2] parameter initialization and changes in the distribution of the inputs of each layer affect the learning rate of the network

Batch Normalization or BN (Ioffe & Szegedy)

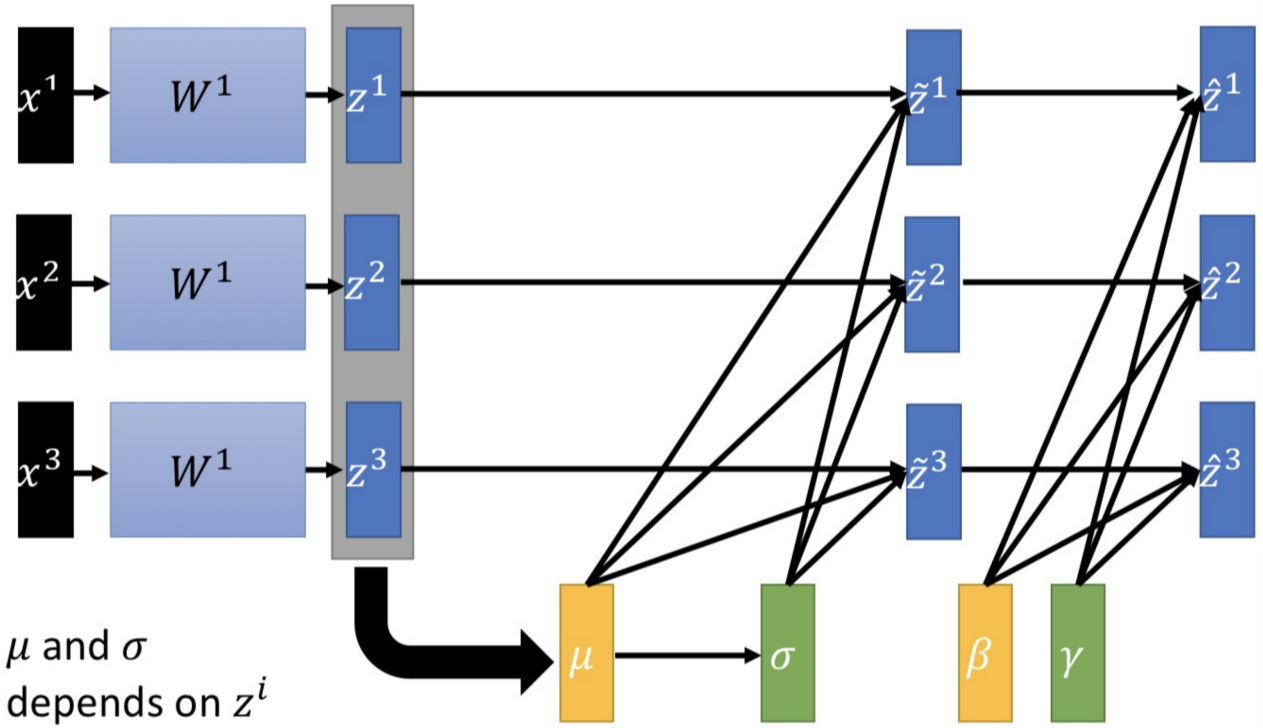
- Computing the mean and variance of a layer[3]
  - Toy example: batch size $(m) = 3$
  - Small batch can't estimate the distribution



[3] Figure credit: Hung-yi Lee

Batch Normalization or BN (Ioffe & Szegedy)

- Normalize, scale and shit[4]

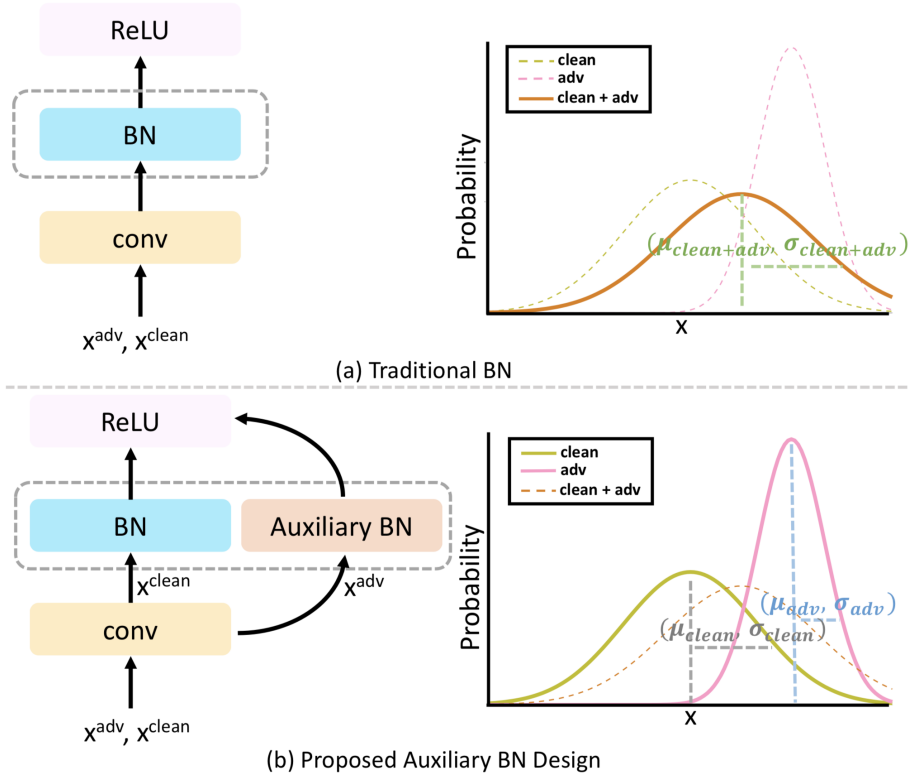


[4] Figure credit: Hung-yi Lee

Why adversarial training hurts accuracy on clean data?

- The performance drop is mainly caused by distribution mismatch: adversarial samples and clean samples are drawn from two different domains (Xie et al. CVPR 2020)
  - Training exclusively on one domain cannot well transfer to the other
  - If this distribution mismatch can be properly bridged, then performance drop on clean samples should be minimized even if adversarial samples are used for training
- How can we distill valuable features from adversarial samples in a more effective manner and boost model performance generally?
  - Batch normalization (BN) is an essential component for this
  - BN normalizes input features by mean and variance in each batch
  - Input features of BN should come from a single or similar distributions
  - The normalization behavior could be problematic if the batch contains data from different distributions, resulting in inaccurate statistics estimation

Disentangled Learning via an Auxiliary BN



- Researchers propose an auxiliary BN to guarantee its normalization statistics are exclusively preformed on the adversarial samples
  - To disentangle this mixture distribution into two simpler ones respectively for the clean and adversarial images
  - Otherwise, simply maintaining one set of BN statistics results in incorrect statistics estimation

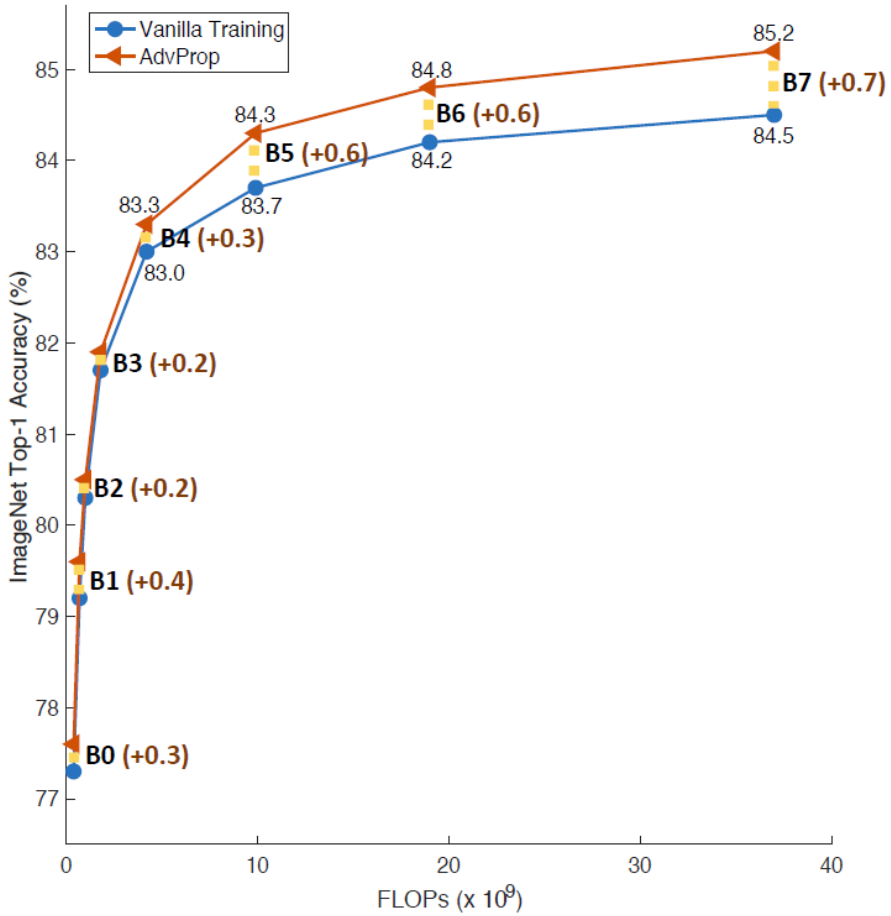
AdvProp (Xie et al. CVPR 2020)

- Randomly initialize a model w_0; $t \leftarrow 0$; If $t < T$:
  - Read a data batch $B = \{(x_1, y_1), ..., (x_m, y_m)\}$ from training set
  - Generate a batch of adversarial samples $\tilde{B} = \{(\tilde{x}_1, y_1), ..., (\tilde{x}_m, y_m)\}$ from corresponding clean samples B using current state of the model w_t with **auxiliary BNs**
  - Compute loss $L(w_t, B)$ on B using **main BNs**
  - Compute loss $\tilde{L}(w_t, \tilde{B})$ on $\tilde{B}$ using **auxiliary BNs**
  - Do one training step by loss function $L + \tilde{L}$
  - $t \leftarrow t + 1$

Properties of AdvProp (Xie et al. CVPR 2020)

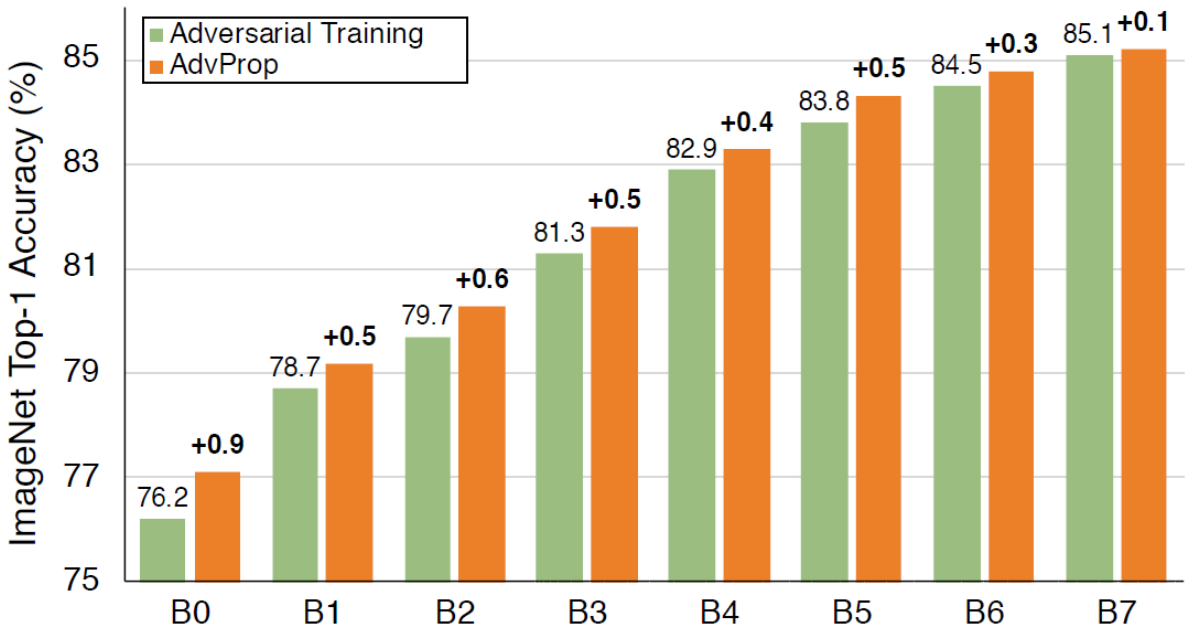
- AdvProp enables networks jointly learn useful features from adversarial data and clean data at the same time
- Auxiliary BN only increases a tiny amount of extra parameters
  - Except for BNs, other layers are jointly optimized for both adversarial data and clean data
  - e.g. 0.5% more parameters than the baseline on EfficientNet-B7
- At test time, these extra auxiliary BNs are all dropped, and we only use the main BNs for inference

Effects of AdvProp



- AdvProp substantially outperforms the vanilla training baseline
- Performance improvement is larger for a larger model
  - ImageNet Training with Efficient-Net

Effects of AdvProp



- AdvProp substantially outperforms adversarial training, especially for small models
- Carefully handling BN is important for training with adversarial data
  - ImageNet Training with Efficient-Net

Outline

- Introduction
- How to attack?
- How to defend?
- How to effectively defend?
- Conclusion

Summary

- What is adversarial machine learning? Why it is important?
  - Defending the important applications using AI
- Attack: fool the model by creating adversarial examples
  - White-box attack: Fast Gradient Sign Method
  - Black-box attack: building a substitute model
- Defend: train the model to detect adversarial examples
  - Adversarial Training
- Defend to improve the model
  - AdvProp: Disentangled Learning via an Auxiliary BN

Reference

- Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).
- Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial machine learning at scale." arXiv preprint arXiv:1611.01236 (2016).
- Papernot, Nicolas, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. "Practical black-box attacks against machine learning." In Proceedings of the 2017 ACM on Asia conference on computer and communications security, pp. 506-519. 2017.
- Xie, Cihang, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. "Adversarial examples improve image recognition." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 819-828. 2020.

Any Questions?