AE Defenses Part II

Lecture 6

CS25800, Adv. ML

Quick Recap of Tue Lecture

How to protect your castle against intruders ?

Defenses against AE

1. Defense via Detection: Detect Adversarial Examples at Inference Time

- Upon detection, reject the input or purify the input (to remove perturbation) and run the classifier on the purified input

2. Defense via Prevention: Train models to make it hard to find successful Adversarial Examples

- Gradient Obfuscation
- Robust Optimization (e.g., adversarial training)

- Also need to consider countermeasures by attackers (when becoming aware of the defense and even having some knowledge of the defense)

Part I: Defense by Detecting Adversarial Examples

Threat Models

- Defender's goal: Separating adversarial examples from regular (or benign) images

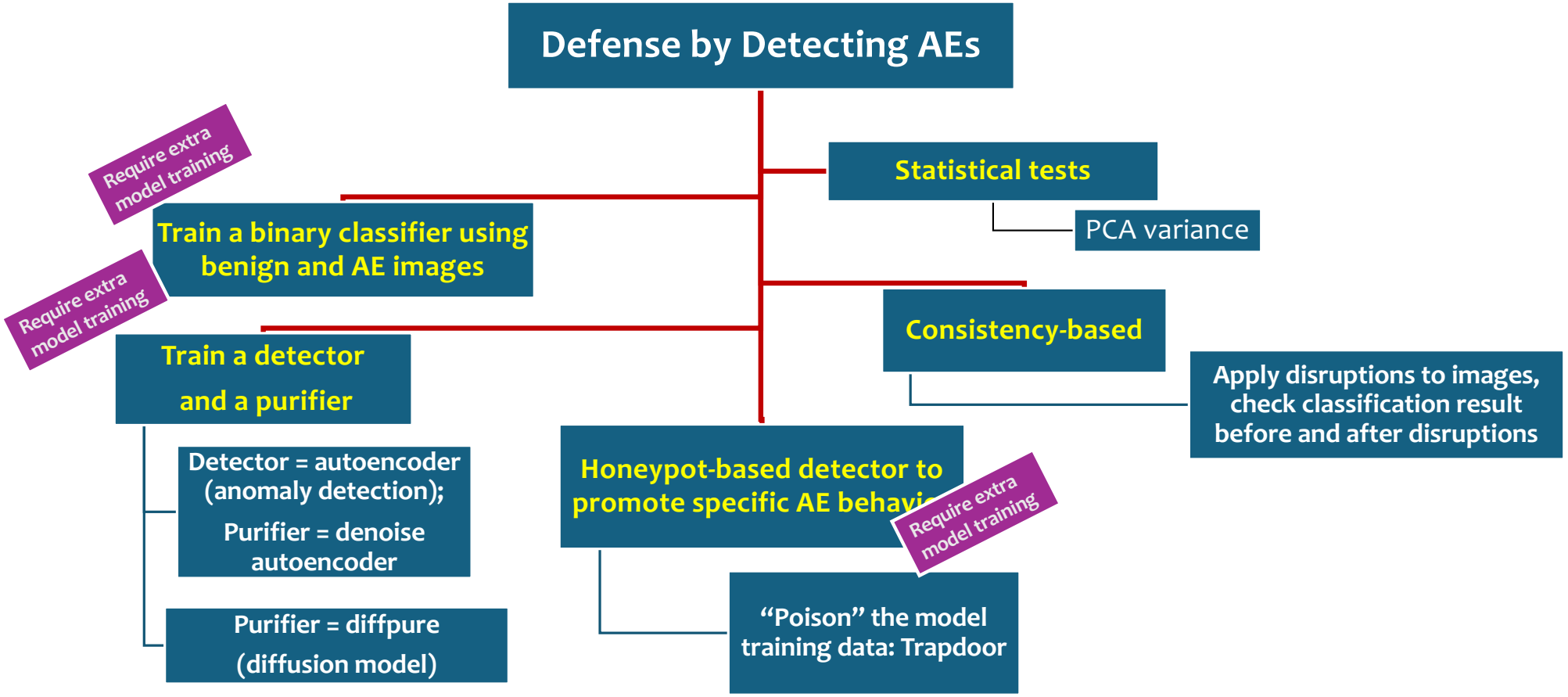
- Three types of attacker's knowledge of the defense
  - **Zero-knowledge** adversary: attacker is unaware of the detection
  - **Limited-knowledge** adversary: attacker has partial knowledge of the detector, e.g., knowing the type of defense but not exact parameters
    - Attacker runs a gray-box attack against the detector
  - **Perfect-knowledge** adversary: attacker has fully knowledge of the detector
    - Attacker uses this knowledge to generate AEs that fool both the classifier and the detector; aka, white-box attack against the detector

For now, let's focus on "zero-knowledge adversary"

Attacker is unaware of the defense

~~Four~~ Five Types of Detectors

1. **Statistical tests,** e.g. comparing variance of PCA coefficients
2. **Train a binary classifier** using some benign images and AEs as training data
3. **Train a detector (to detect AEs) and a purifier (to purify undetected AEs)**
   - Autoencoder-based detector and purifier, both trained using benign data only
   - Purification via diffusion models is more effective but way too expensive
4. **Consistency-based methods**
   - Instead of purifying images, "smoothing" images to disrupt AE perturbation
   - Multiple forms of "smoothing": being "friendly" to benign images
   - Check consistency of classification output before/after smoothing
5. **Honeypot based methods**
   - Train the classification model by intentionally injecting vulnerabilities that make the corresponding AEs "easy" to compute --> lure attackers to follow a known behavior to detect them

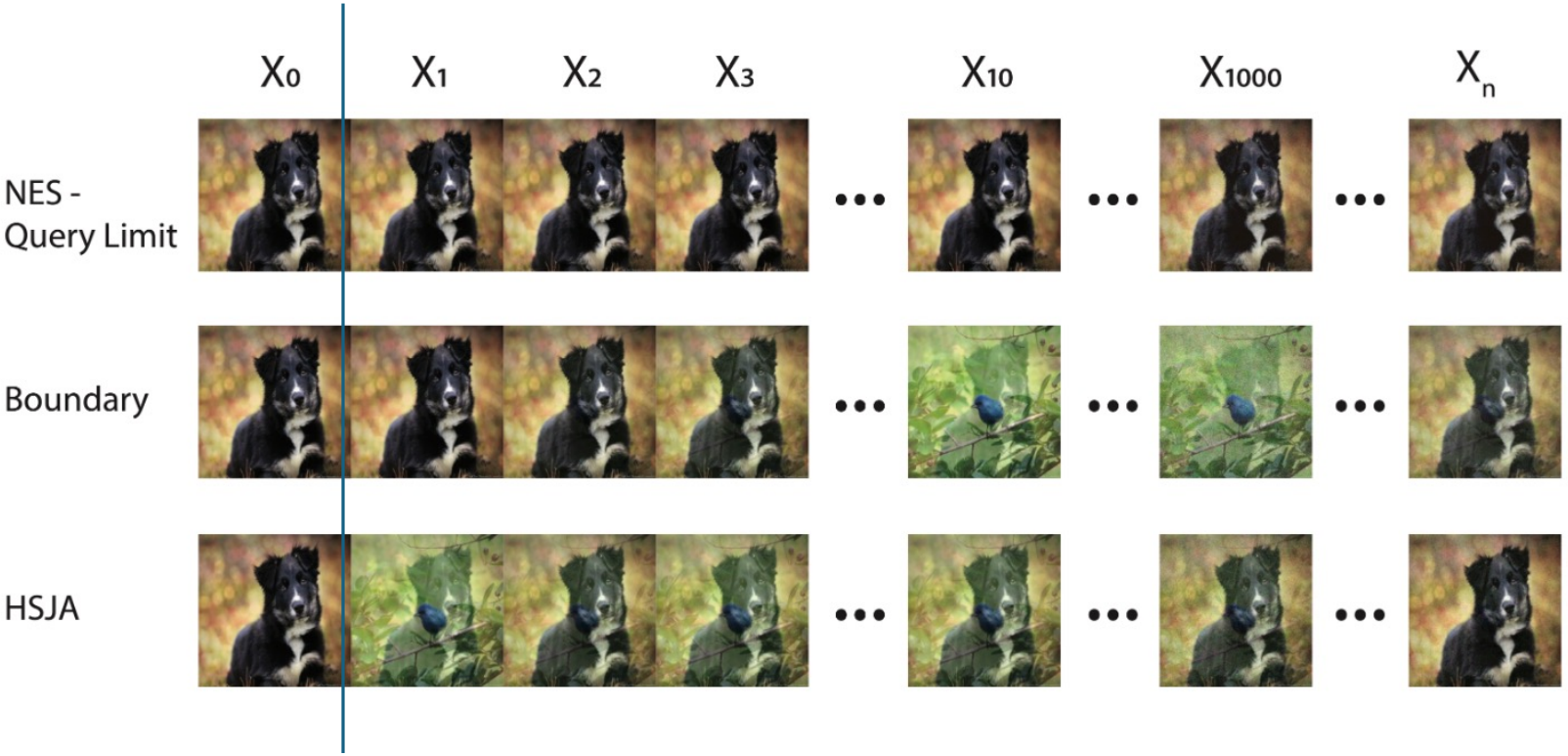

Defense by Detecting AEs
Statistical tests
PCA variance
Require extra model training
Train a binary classifier using benign and AE images
Require extra model training
Train a detector and a purifier
Detector = autoencoder (anomaly detection);
Purifier = denoise autoencoder
Purifier = diffpure (diffusion model)
Consistency-based
Apply disruptions to images, check classification result before and after disruptions
Honeypot-based detector to promote specific AE behavi
Require extra model training
"Poison" the model training data: Trapdoor

Detecting Query-based Black-Box AE Attacks

- Detect and then stop attack queries before the attack succeeds
- Look at a series of image queries rather than a single image

Black-box Attack's Query Sequences

- Key property for detection: high visual similarity between at least 2 query images in the same attack sequence



Not That Simple

- Machine Learning as a Service (MLaaS)
  - Queries are sent by many many many users simultaneously
- Attacker: A single attack can leverage many user accounts
- Defender: do I have to store **All** the query images across all the users?
  - Yes, you do
  - **How to reduce storage cost?**
    - instead of storing the entire raw image, store a **fingerprint** of the image using probabilistic hashing
    - Clear the pool periodically (e.g., daily) to slow down the attack

Blacklight Detector

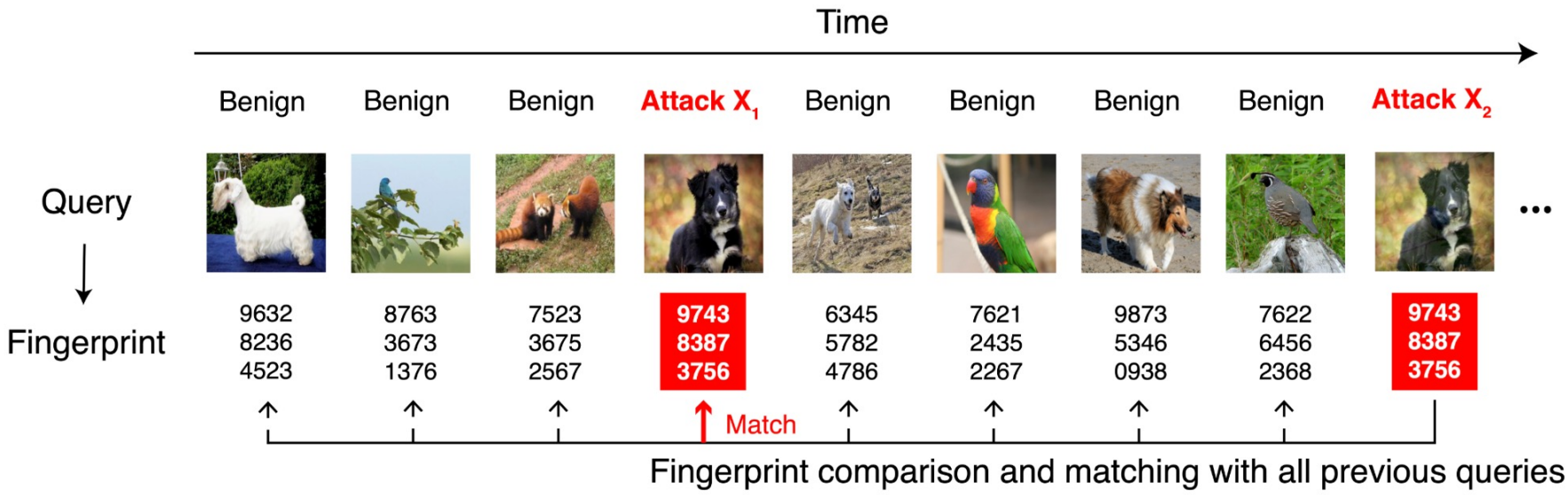


Figure 3: Blacklight computes a small set of hash entries(as its probabilistic fingerprint). Blacklight detects attack images hidden inside a large stream of benign images by comparing and detecting highly similar fingerprints.

Blacklight: Scalable Defense for Neural Networks against Query-Based Black-Box Attacks, Usenix Security 2022

Actions taken after detecting one or more attack queries

- Ban accounts
  - High penalty for false positives, i.e., benign users wrongly flagged as attackers
  - Low impact on attacker (just switch to another account)

- Give some query answer to mislead attacker and facilitate future detection
  - Honeypot again
  - Hard to design due to uncertainty of attacker behavior

- Purify the query image(s) and send the result of the purified image
  - Higher cost (e.g. diffusion purification --> 100x query response delay, alerting the attacker) & may not change query outcomes

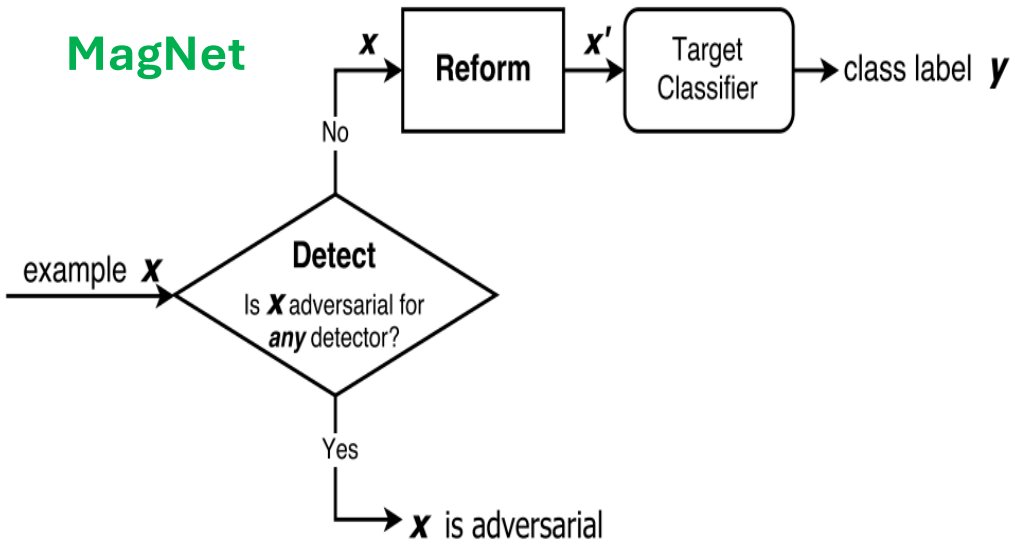
- Reject the query
  - Most practical in terms of cost, but also inform the attacker that the attack queries are detected, do something different/smarter

So far, zero-knowledge adversary

Today: **limited-knowledge adversary** who knows the defense method but not its exact parameters

How to attack a protected castle?

MagNet vs. Limited-Knowledge Adversary



Adaptive Attacks

- Attacker picks a "smart" perturbation to evade the "detector" part
  - Use AEs whose reconstruction loss is similar to that of benign images
- Attacker trains a local denoise autoencoder, uses it to refine AE optimization to evade purification
  - Find AEs that can withstand denoising
- Hope for high attack transferability

Defender's countermeasure against adaptive attacks

- Strengthen defense by adding randomness to lower attack transferability
- MagNet trains multiple denoise autoencoders, randomly chooses one
  - Randomness may lower attack transferability

MagNet's Effectiveness

against limited-knowledge adversary, who knows the defense method but not its exact parameters, and thus leverages transfer-based attacks

Mismatch between the defense's autoencoder and the attacker's autoencoder makes AE attacks ineffective

i.e. low attack transferability

Autoencoder used by the Attacker

Autoencoder used by the defender	A	B	C	D	E	F	G	H
A	0.0	92.8	92.5	93.1	91.8	91.8	92.5	93.6
B	92.1	0.0	92.0	92.5	91.4	92.5	91.3	92.5
C	93.2	93.8	0.0	92.8	93.3	94.1	92.7	93.6
D	92.8	92.2	91.3	0.0	91.7	92.8	91.2	93.9
E	93.3	94.0	93.4	93.2	0.0	93.4	91.0	92.8
F	92.8	93.1	93.2	93.6	92.2	0.0	92.8	93.8
G	92.5	93.1	92.0	92.2	90.5	93.5	0.1	93.4
H	92.3	92.0	91.8	92.6	91.4	92.3	92.4	0.0
Random	81.1	81.4	80.8	81.3	80.3	81.3	80.5	81.7

AE Attack Failure Rate (%)

However, MagNet becomes less effective
when the attacker uses an ***ensemble of (32) autoencoders*** to
increase attack transferability

See paper: Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods, AISec 2017

Yes, it is a Cat and Mouse Game. Persistent attackers will eventually succeed.

Defense (or attack) is about raising the cost of the other party to reduce practicality & incentives.

Feature Squeezer vs. limited-knowledge adversary

- Attackers leverage some knowledge of the feature squeezers to develop AEs that can tolerate the squeezes
- Cost: more compute time to find working AEs and/or more visible perturbations

AE images created by Adaptive CW Attacks against Feature Squeezers

Summary on AE Detections

- Leverage inherent differences between benign and adversarial images

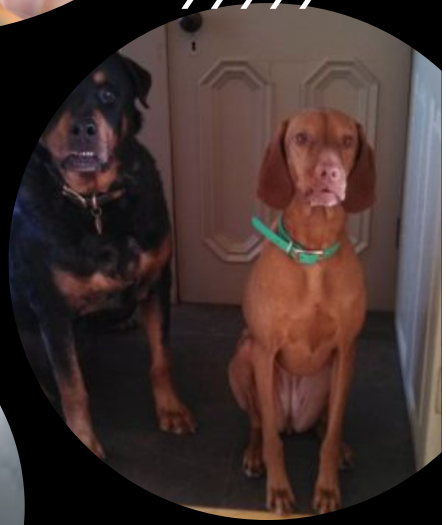
- Key challenge: How to define/measure such difference
  - Statistical metrics
  - Training a DNN to learn the difference
  - Training a detector and a purifier
  - Consistency-check before and after denoising the input image
  - Honeypot-based method to promote specific (and detectable) AE behavior

- **Bullet-proof defenses are HARD**, especially when the attacker has information about the defense
  - Defender injecting randomness vs. Attacker using ensembles to increase attack transferability
  - In practical deployment, it is about cost/return at both sides
  - Much harder & higher cost at the defense side..

What is a Good Defense?

Proactive, well-prepared, reconfigurable, and adaptable,
while also being able to effectively counter practical adaptive attacks.

Defense is never a single, fixed form



Part II: Prevention-based Defenses

Making it hard to compute AEs

Defenses against AE

1. Defense via Detection: Detect Adversarial Examples at Inference Time

- Upon detection, reject the input or purify the input (to remove perturbation) and run the classifier on the purified input

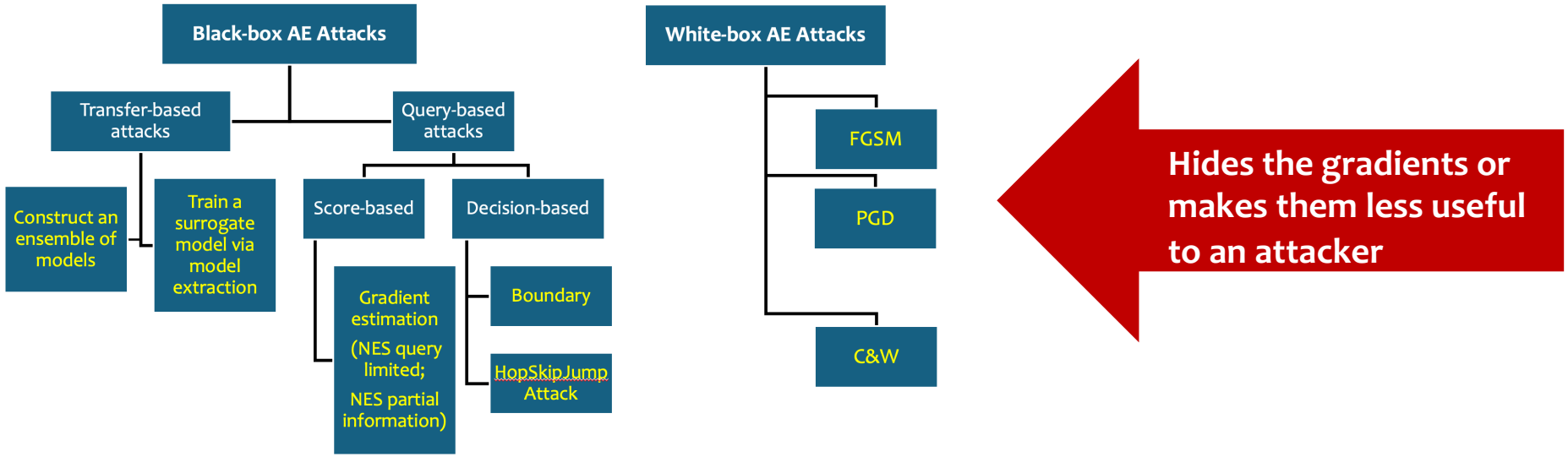
2. Defense via Prevention: Train models to make it hard to find successful Adversarial Examples

- Gradient Obfuscation
- Robust Optimization (e.g., adversarial training)

- Also need to consider countermeasures by attackers (when becoming aware of the defense and even having some knowledge of the defense)

Most AE Attacks Rely on "Gradient"

- Untargeted attack: $\boldsymbol{\eta = \epsilon \cdot sign\left(\nabla_x J(\mathbf{W}, \mathbf{x}, \mathbf{y})\right)}$
- Targeted attack: $\boldsymbol{\eta = -\epsilon \cdot sign\left(\nabla_x J(\mathbf{W}, \mathbf{x}, \mathbf{y}_{target})\right)}$



Prevention Defense 1: Gradient Masking/Obfuscation

Masking or hiding the gradient from being used by the attacker

Three Directions

- Gradient Shattering: Make J(x,y,W) non-differentiable
- Stochastic Gradient: Train multiple classifiers, randomly pick one at run-time
- Vanishing/exploding Gradient: Train DNN models to make gradient over x very very small

Method 1: Input Transformations

- Instead of training a model F(x), train a model F(G(x))
  - If G(x) non-differentiable over x, then F(G(x)) is non-differentiable over x

- G(x) as some "random" combinations of image transformations
  - Image cropping, rescaling
  - Bit-depth reduction, JPEG compression
  - Randomly drop pixels and restore them
  - Image quilting (drop and replace small patches)

Method 2: Defensive Distillation

- Inspired by Knowledge Distillation (KD)
  - KD's Goal: Transfer knowledge from a large neural network to a smaller neural network
  - KD's operation: First train a large DNN (teacher); then use this model to compute ***softmax*** probs. (soft labels) for each input x; finally, use these soft labels to train a smaller DNN

- Recall from lecture 2: DNNs are normally trained using hard labels y=[0,1,0,0,0] when computing the loss function J(x,y,W)
  - can be easily updated to use soft labels y=[0.1, 0.8,0.05,0.025,0.025]

Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks, IEEE S&P 2016

Method 2: Defensive Distillation

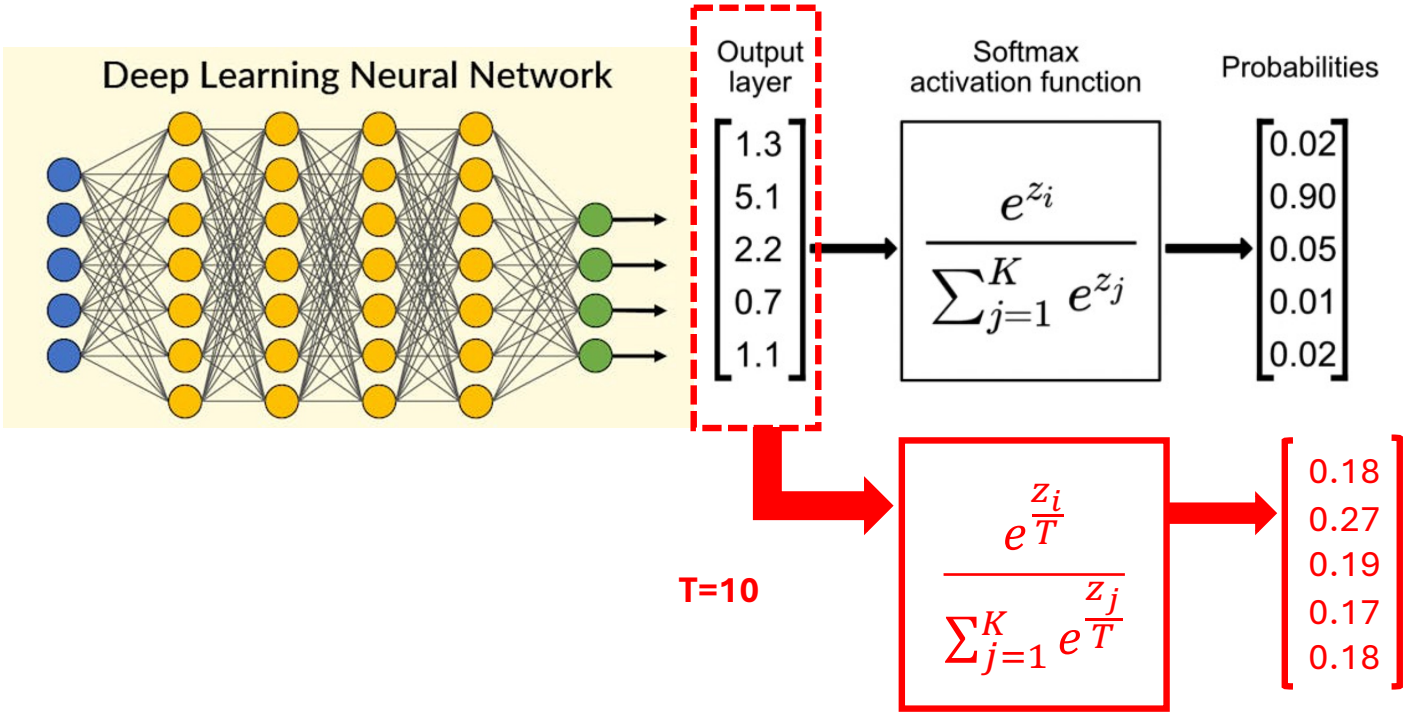
- **Defensive Distillation**
  - Train a standard DNN (M1) using the original hard training labels
  - M1 produces soft label for each training input, aka Softmax values
    - **Except each Logit is divided by a temperature T (T>1) to reduce the impact of each Logit**
  - Train M2 (**the same model size of M1**) using **soft label produced by M1 and the same large T (when computing M2's loss function)**
  - **At test time, set M2's temperature T back to 1**

Analytical proof: making the model less sensitive to input perturbations → Gradients are very small

Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks, IEEE S&P 2016

Scaling SoftMax Probability via T

- T: temperature used to scale the impact of logits z_i
  - T>1: Scale down the impact of z



Empirical Verification of Gradient Reduction

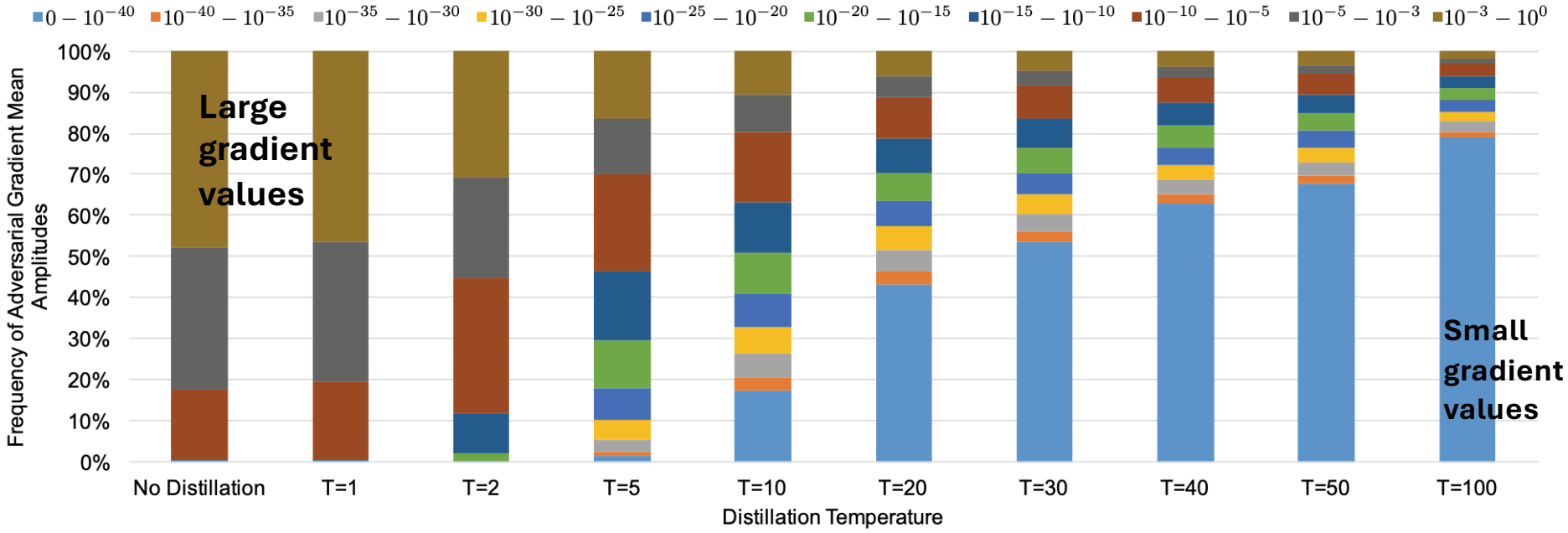


Fig. 9: **An exploration of the impact of temperature on the amplitude of adversarial gradients:** We illustrate how adversarial gradients vanish as distillation is performed at higher temperatures. Indeed, for each temperature considered, we draw the repartition of samples in each of the 10 ranges of mean adversarial gradient amplitudes associated with a distinct color. This data was collected using all $10,000$ samples from the CIFAR10 test set on the corresponding DNN model.

Countermeasure by Limited-knowledge Adversary

- BPDA (ICML 2018) is an adaptive attack developed to counter 7 out 9 defenses published at the ICLR 2018 conference
  - assumption: the attacker knows the defense method

- Design intuition: "*substitution, substitution, substitution*"
  - Function-level approximation: approximate the gradient of a non-differentiable loss J(x,y,W) by the gradient of a differentiable function
  - Change of variable: vanishing/exploding gradient over **x** → normal gradient over **z**

- When there is no whitebox access to the "protected model", leverage ensemble-based attack transferability
  - an ensemble of models or an ensemble of image transformations

Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples, ICML 2018

Countermeasure by Limited-knowledge Adversary

- Defense by Gradient Shattering: implement F(x) as F(g(x)) where g(x)$\approx x$ is non-differentiable over x
  - **Countermeasure:** Attacker uses a differentiable function to approximate F(x) the backprop process; still use F(x) in the forwardprop process

- Defense by Stochastic Gradient: randomized input transformation
  - **Countermeasure:** apply Expectation over Transformation (EoT)

- Defense via Vanishing Gradient: Implement F(x) as F(g(x)) where g(x) leads to very small gradient over x
  - **Countermeasure:** apply the change of variable trick, i.e., make x = h(z), thus we have g(x)=g(h(z)) where h(z) is differentiable over z and does not lead to vanishing gradient

Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples, ICML 2018


A Short Break

Prevention-base Defense 2: Robust Optimization

Train robust models by incorporating AE samples into training, i.e., $(x+\eta, y)$ where y is the classification label of x

Adversarial Training

- Goal: **teach** the model to correctly classify adversarial examples
  - Augment the model's training data with adversarial examples produced by known types of attacks
  - Each benign training data **(x,y)** + some adversarial example **(x+η, y) whose label is corrected to y**

- Process:
  - Train a model $\mathbf{M_o}$ at usual, using just benign training dataset (x,y)
  - Run AE attacks against $\mathbf{M_o}$ using **x** from the benign training dataset, to generate **x+η**, create augment dataset **(x+η, y)**
  - Retrain a new model **M** using a combination of benign and AE datasets
    - A more practical option is to finetune $\mathbf{M_o}$ with (x+η, y)

Adversarial Training (cont.)

- Key challenge: how to select AE attack samples $(\mathbf{x}+\boldsymbol{\eta})$ used for model training

- Proposal 1: FGSM: *"Explaining and Harnessing Adversarial Examples," ICLR 2015*
  - The ICLR 2015 paper introduced both FGSM and the concept of adversarial training
  - Weakness: does not generalize to PGD ☹
    - FGSM produces a very narrow set of working AEs
    - The model is overfitted into these FGSM AEs and cannot generalize to other types of AEs

Adversarial Training (cont.)

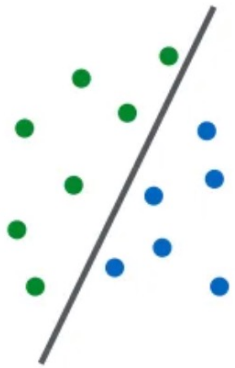
Proposal 2: PGD by "*Towards Deep Learning Models Resistant to Adversarial Attacks*" (ICLR 2018)

- Justified by empirical findings (from the same paper)
  - As long as the attacker only uses gradients of the loss function wrt x, it will not find significantly better local maxima than PGD

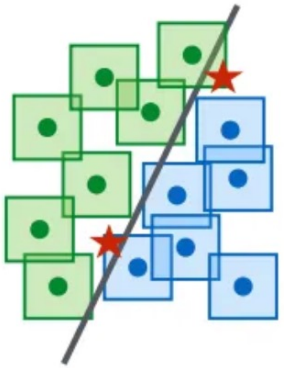
- Two Key suggestions
  - The mode capacity needs to be sufficiently large to learn these Aes
  - Use the strongest possible adversary in training
    - Instead of finding working $x_{\text{adv}}=x+\eta$ for each x, find the most adversarial one
    - For untargeted attacks, it is x_{adv} that maximizes the loss J(x_{adv}, y, W)
    - Train/fine-tune the model with (x_{adv}, y)

Adversarial Training (cont.)

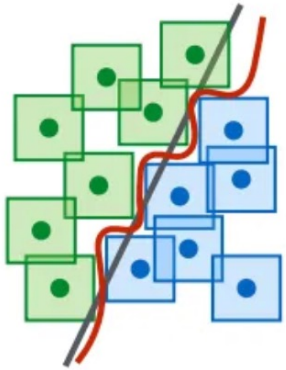
Impact of the Model's Capacity



Decision boundary w/o adversarial training



Some AE perturbations of x (marked by red stars) cross the decision boundary



Need high-capacity network to capture the change of decision boundary by adversarial training

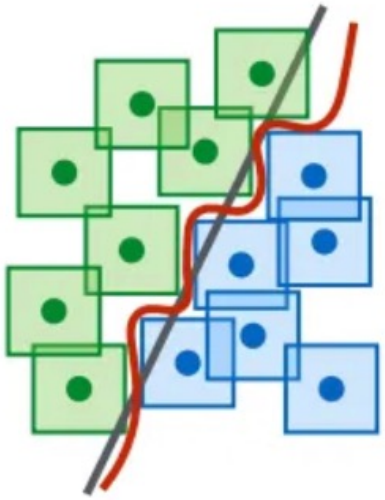
Key Limitations of Adversarial Training

- High computation cost
  - Must generate AEs to curate model training data
  - Need to generate strong AEs, which takes time!

- Inherent tradeoff between model accuracy and adversarial robustness
  - A model that is more robust against AE attacks often faces reduced accuracy on benign input

  - ***Over-hardening***: Being specialized in resisting some specific AEs makes the model hard to generalize to unseen (benign) inputs

Optional reading: THE LIMITATIONS OF ADVERSARIAL TRAINING AND THE BLIND-SPOT ATTACK, ICLR 2019



Adversarial Training Performance

1- attack success rate

Model accuracy on benign images

No adversarial training

PGD-based adversarial training

Follow-up research to improve adversarial training

Method	Robust Test Acc.	Standard Test Acc.	
Standard Training	0.8%	95.2%	Vanilla Supervised
PG-AT (Madry et al., 2018)	45.8%	87.3%	
TRADES (Zhang et al., 2019)	55.4%	84.0%	
Standard Self-Training	0.3%	96.4%	Semisupervised with same unlabeled data
Robust Consistency Training (Carmon et al., 2019)	56.5%	83.2%	
RST + PG-AT (this paper)	**58.5%**	**91.8%**	
RST + TRADES (this paper) (Carmon et al., 2019)	**63.1%**	**89.7%**	

Optional reading: Understanding and Mitigating the Tradeoff Between Robustness and Accuracy, ICML 2020

Certified Robustness

- Example claim: A certificate for MNIST that guarantees that no attack with perturbation smaller than L2 norm $\epsilon = 0.1 \approx 25/255$ can cause more than 35% classification error

- A formal definition:
  - A DNN classifier is ε-certifiably robust for x, if no adversarial perturbation whose L_2 norm is no larger than ε exists

Optional: Certified Defenses against Adversarial Examples, ICLR 2018

Optional: Certified Adversarial Robustness via Randomized Smoothing, ICML 2019

Certified Robustness via Randomized Smoothing

A "Smoothed" Classifier (ε-certifiable)

Given a trained "base" classifier ***f(x)***, define a new, "smoothed" classifier ***g(x)*** whose prediction is the majority vote of ***f*** (.) applied to x convolved with some noise distribution, *e.g. isotropic Gaussian noise with some variance*

$$g(x) = \arg\max_{c \in \mathcal{Y}} \mathbb{P}(f(x+\varepsilon) = c)$$

$$\text{where } \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$

- In practice, this is approximated via Monte Carlo sampling on ε
  - creating many noisy versions of the image, i.e. $(x + \varepsilon)$, classifying them, and taking majority vote of their classification decisions

Optional: Certified Adversarial Robustness via Randomized Smoothing, ICML 2019

Certified Robustness via Randomized Smoothing

A "Smoothed" Classifier (ε-certifiable)

$$g(x) = \arg\max_{c \in \mathcal{Y}} \mathbb{P}(f(x+\varepsilon) = c)$$

$$\text{where } \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$

Key observations:

1) The smoothed classifier suffers from considerable degradation to the clean classification accuracy, i.e., classification accuracy of benign, unperturbed images
2) Computationally expensive, so ε needs to be very small
3) No provable guarantee of ε-certification, only true under high probability

Optional: Certified Adversarial Robustness via Randomized Smoothing, ICML 2019

Summary: Defenses against AE

1. Defense via Detection: Detect Adversarial Examples at Inference Time

- Upon detection, reject the input or purify the input (to remove perturbation) and run the classifier on the purified input

2. Defense via Prevention: Train models to make it hard to find successful Adversarial Examples

- Gradient Obfuscation
- Robust Optimization (e.g., adversarial training)

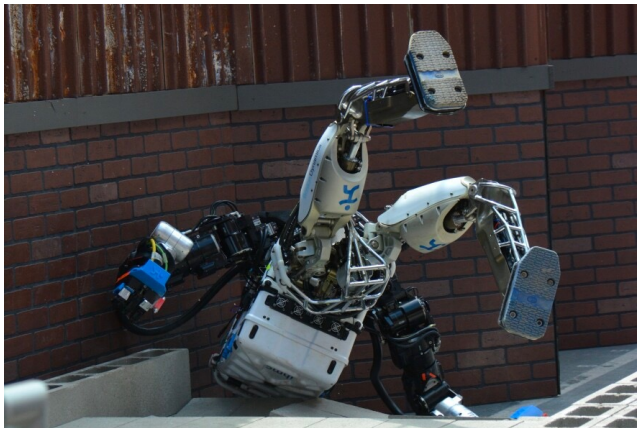
- Also need to consider countermeasures by attackers (when becoming aware of the defense and even having some knowledge of the defense)


Adversarial Examples are Inevitable
Either inevitable features or inevitable bugs

Adversarial ML Research

- Aim for perfect defense
- Against all possible attackers

Real World Systems

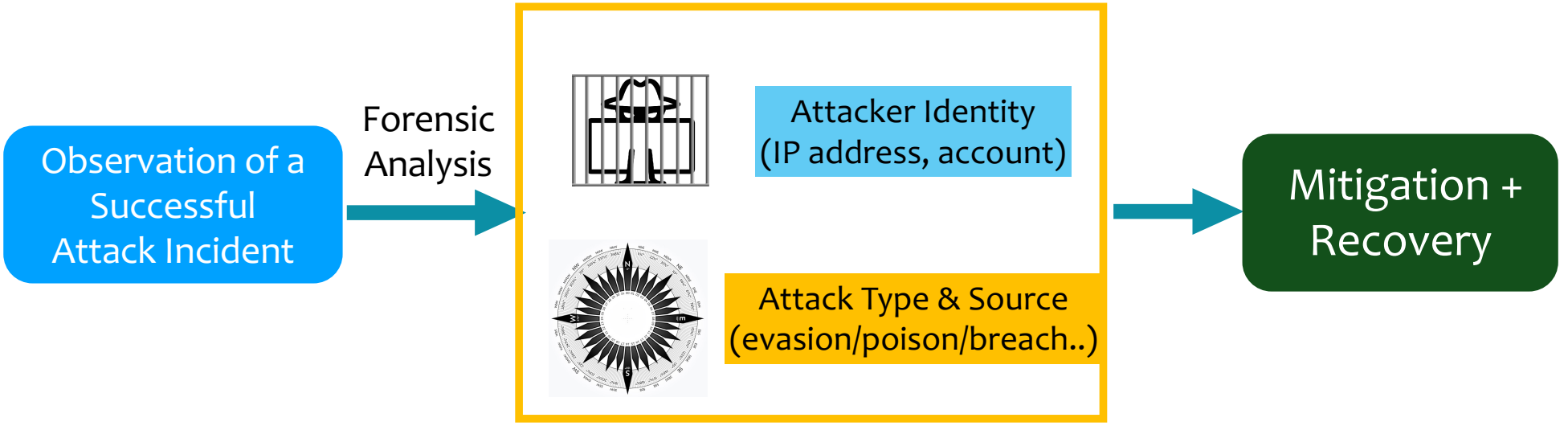


- Imperfect defenses
- Persistent attacker eventually win

What is a Good Defense?

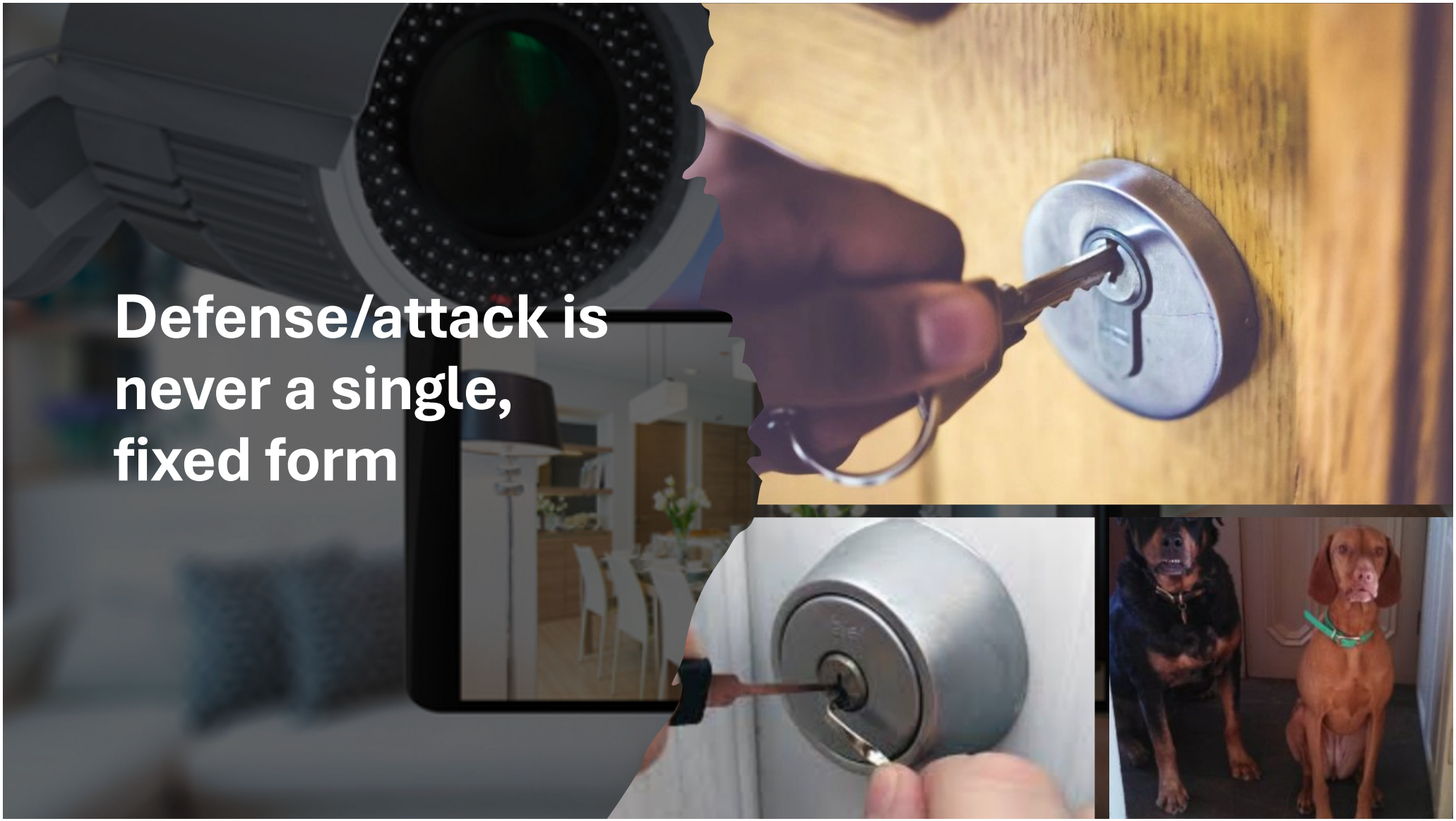
Proactive, well-prepared, reconfigurable, and adaptable,
while also being able to effectively counter practical adaptive attacks.

Expanding DNN Defense to "Post-deployment Forensics & Recovery"



Benefits of Post-deployment Forensics & Recovery

- Mitigate source of attack & Resume services
- Serve as deterrent


Defense/attack is
never a single,
fixed form