AE Defenses

Lecture 5

CS25800, Adv. ML

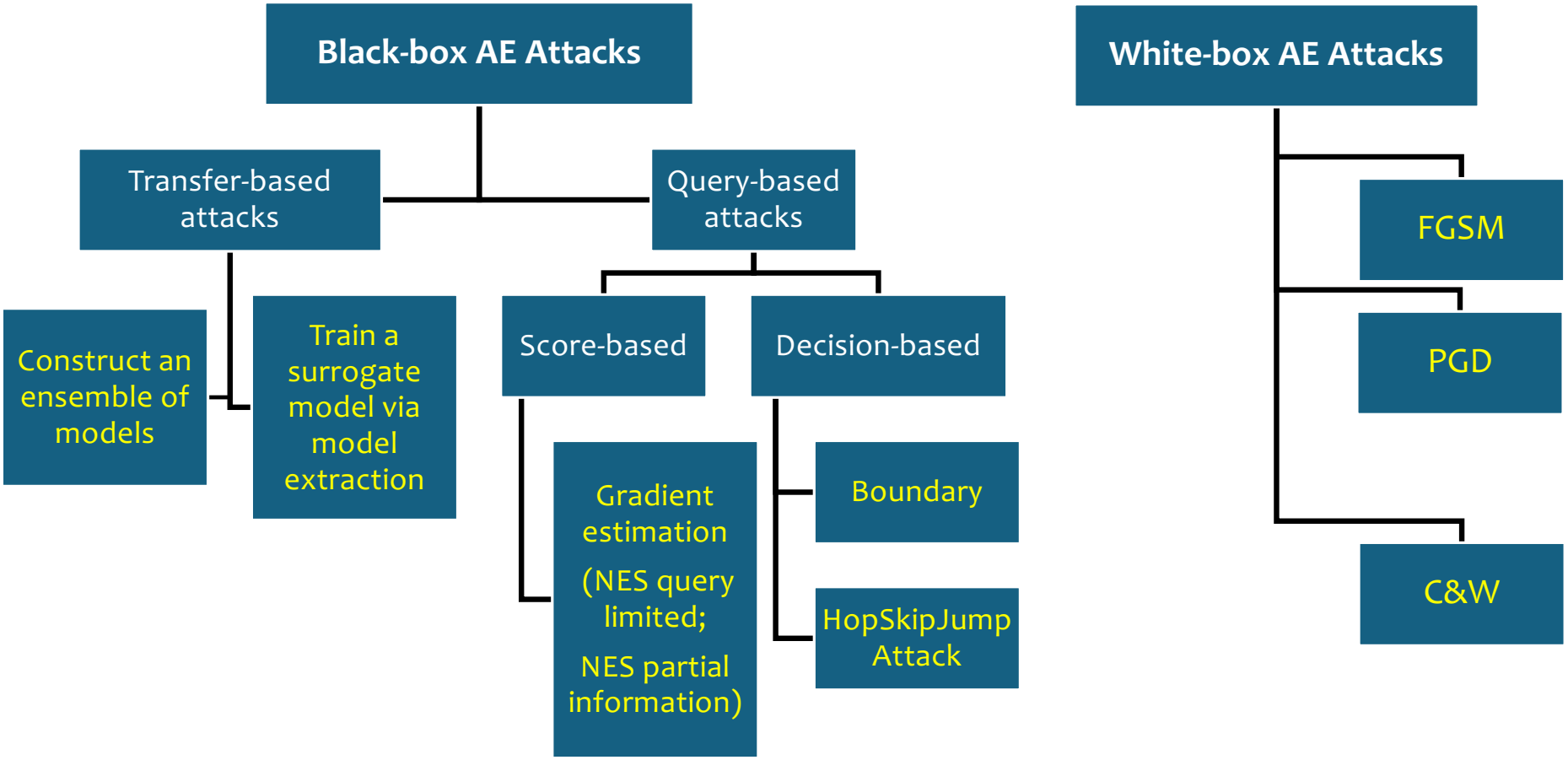
Course Update

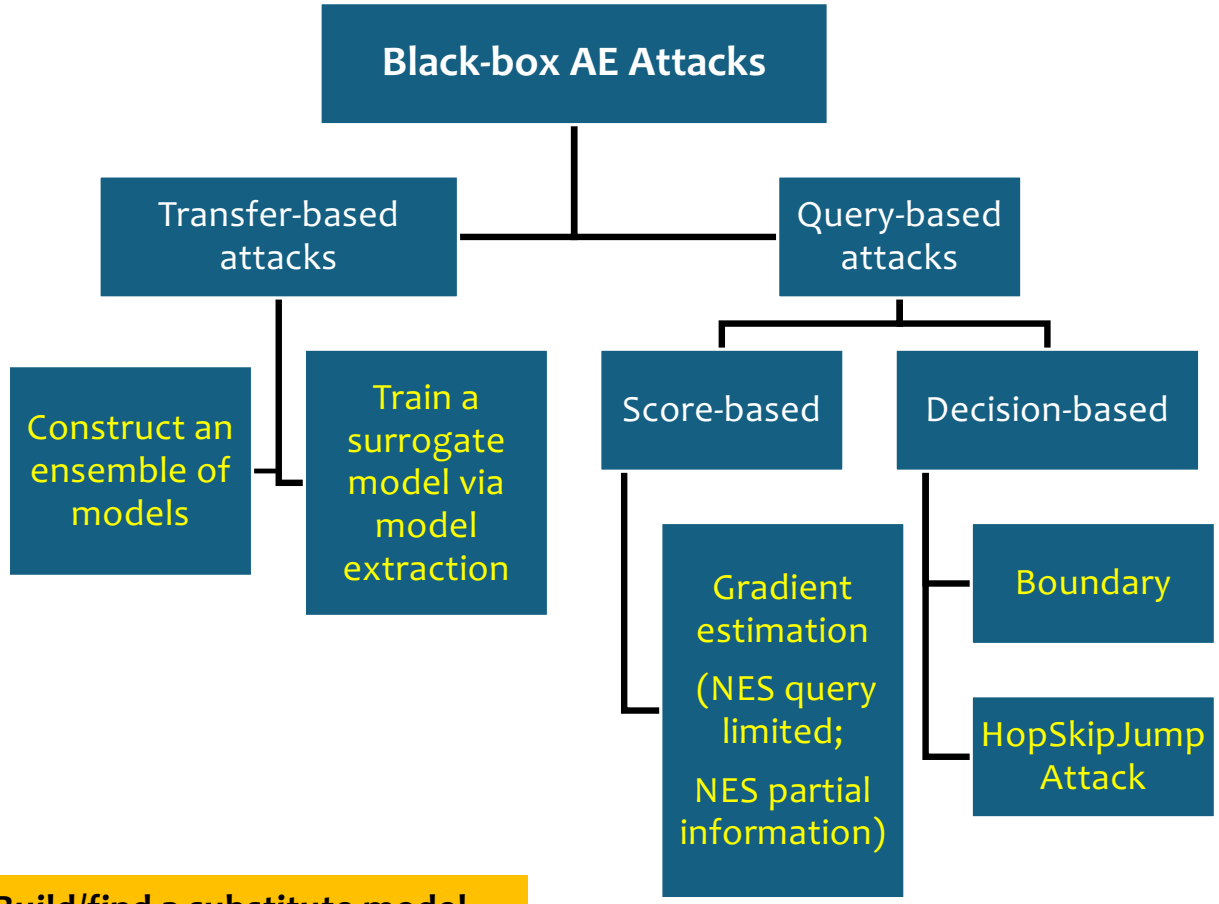
- Homework submission policy updated on canvas/syllabus and ed
- 4 x 1 day extension coupons (tokens)
- Max 2 coupons per homework assignment
- Homework submission will close 2 days after the deadline to automatically support coupons

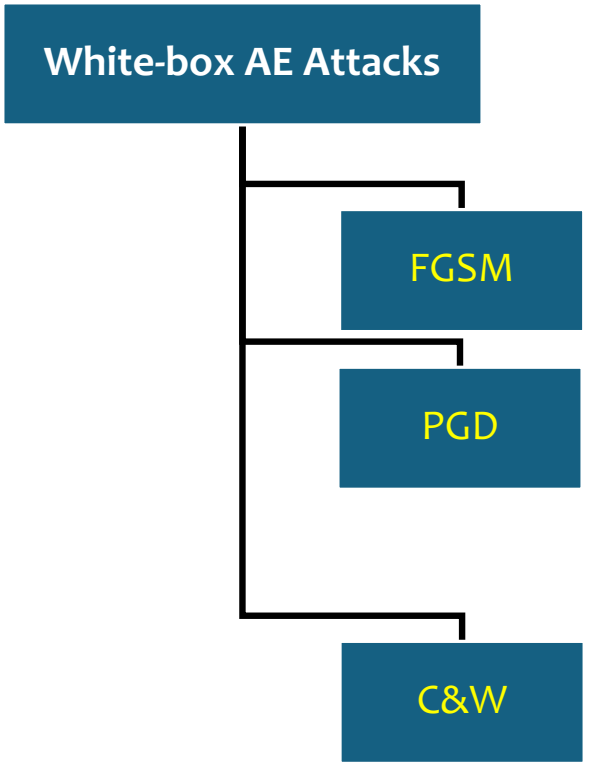
Lecture 4 Recap:
Two Types of Black-box AE Attacks

- **Transfer-based attacks**
  - Attacker finds a substitute model, uses it to create adversarial examples, and sends them to the target model
  - The substitute can be 1) a single surrogate model, or 2) an ensemble of models
  - **Goal: increase attack transferability**

- **Query-based attacks**
  - Attacker queries the target model and uses the query response to create adversarial examples
  - No need to train or collect substitute models
  - Two types of query responses:
    - Class probabilities (i.e., confidence scores per class) → **score-based attacks**
    - Class label only → **decision-based attacks; label-only attacks**
  - **Goal: minimize attack cost, i.e., # of queries**


Black-box AE Attacks
Transfer-based attacks
Query-based attacks
Construct an ensemble of models
Train a surrogate model via model extraction
Score-based
Decision-based
Gradient estimation
(NES query limited;
NES partial information)
Boundary
HopSkipJump Attack
White-box AE Attacks
FGSM
PGD
C&W





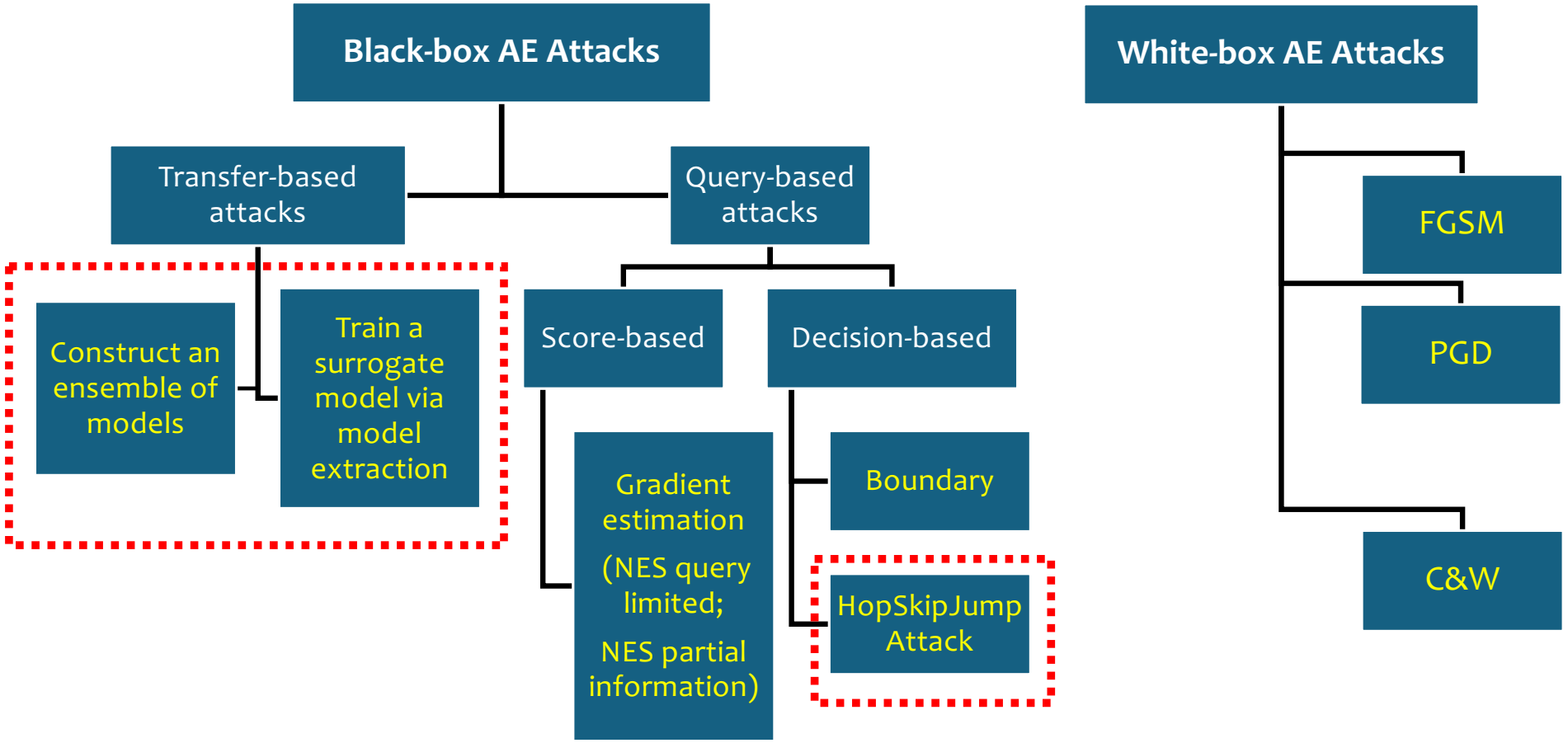
Build/find a substitute model so I can run gradient-based optimization to find $(x + \eta)$ that transfers

Cannot calculate gradient directly, so estimate them by querying the model "repeatedly"

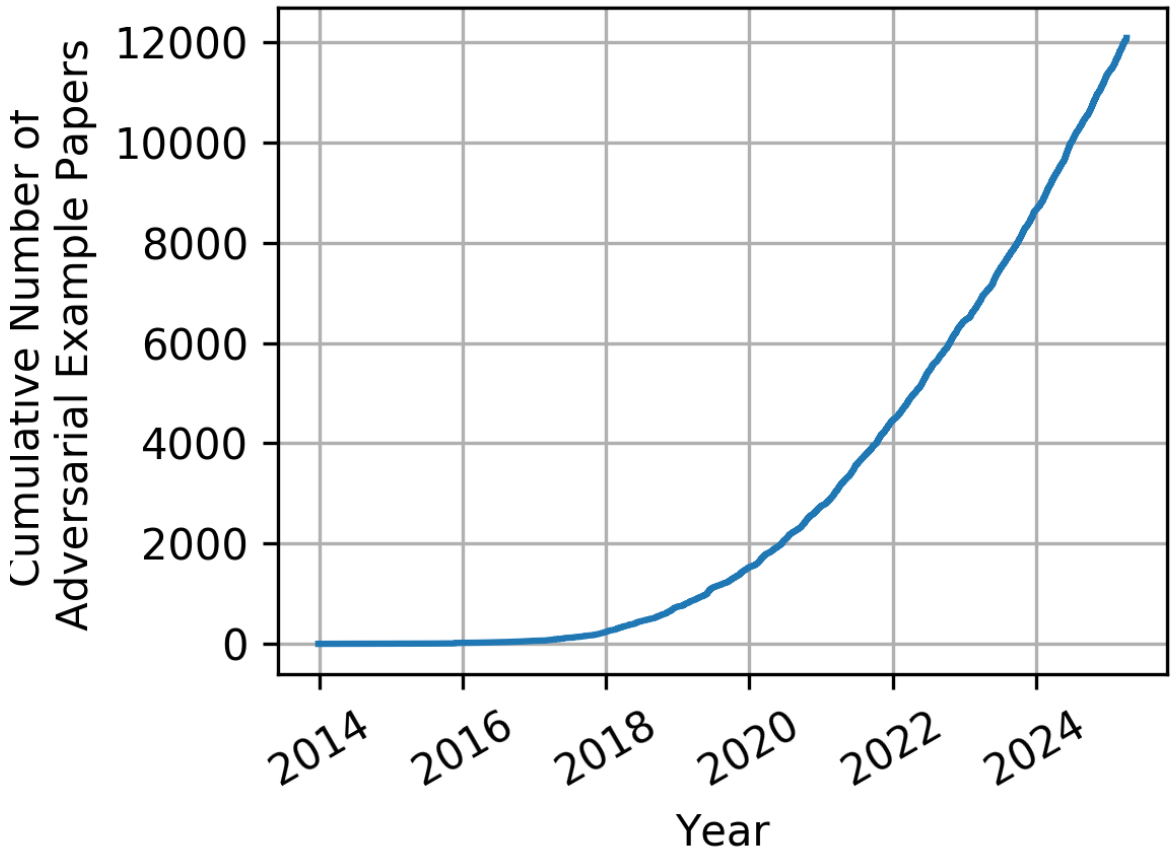
Gradient-based "optimization" to find the working $(x + \eta)$ subject to constraints on η

Practical Implications

- These are all great on-paper designs
- Practical operations must consider diverse real-world context
  - Knowledge of the model
    - When do attackers have whitebox knowledge of the classifier?
    - If not, can they repeatedly query the model (>10k times) to compute AEs?
      - What is the query response?
  - Cost/overhead for real-time operation
    - Not just the time spent on AE optimization
    - e2e overhead: take the image, compute the perturbation, add it to the image, and submit the perturbed image to the classifier
  - Feasibility of modifying images to add perturbations (in real-time)


Black-box AE Attacks
Transfer-based attacks
Query-based attacks
Construct an ensemble of models
Train a surrogate model via model extraction
Score-based
Decision-based
Gradient estimation
(NES query limited;
NES partial information)
Boundary
HopSkipJump Attack
White-box AE Attacks
FGSM
PGD
C&W

AE attack/defense still an active area of research



https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

Today: AE Defenses

How to protect your castle against intruders ?

How to attack a protected castle?

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

Four Categories of Detectors

2016 -2018

1. Statistical tests
2. Train a classifier, using a set of AE samples and benign samples as training data
3. Consistency-based methods
4. Honeypot based methods

Key challenge: the detector does not know "all" AE attacks

- benign noise vs adversarial noise

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

Attacker is unaware of the defense

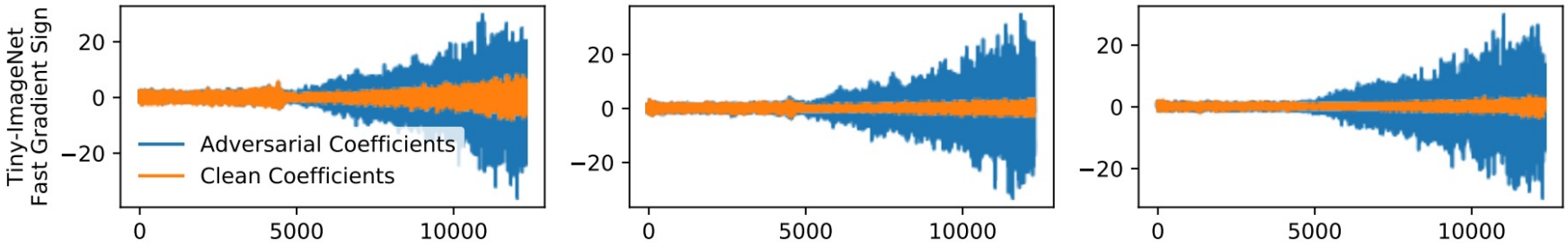
Detection 1: Statistical Tests

AEs and benign images are NOT from the same distribution

Intuition and Specific Algorithms

- Intuition: AEs and Benign Images are Not from the same distribution, use statistical tests to separate them
- Challenge: How to measure statistical distribution of raw images
- One Method:
  - **Principle Component Analysis (PCA):** analyze the PC distribution, use the coefficient variance as the detection metric

Top 15000 PCs for 3 randomly selected benign (orange) and FGSM-AE (blue) image pairs



EARLY METHODS FOR DETECTING ADVERSARIAL IMAGES, ICLR 2017

Detection 2:
Training DNN Detectors

No need to identify detection metrics manually

Train a Detector on Raw Images

- Training data = adversarial examples (AE) + benign images
- Setup: a separate, binary classifier dedicated for detecting AE
  - Or modify with the original task classifier (w/ N classes) to add an N+1th class (?)

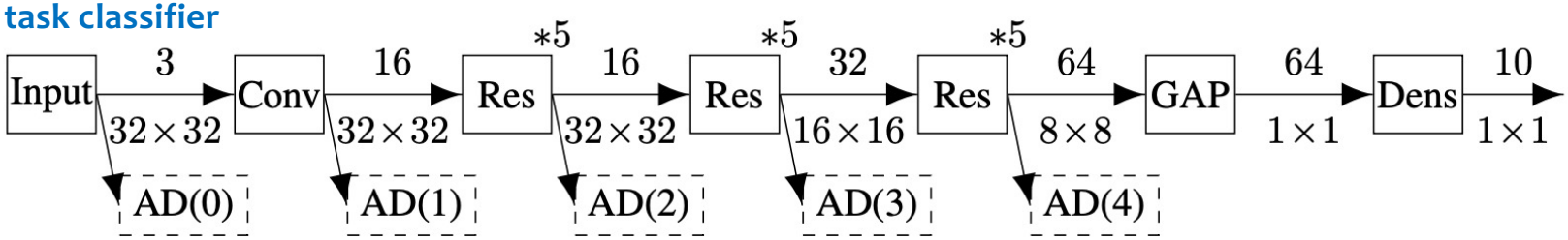
- Detection accuracy depends heavily on the training sets
  - High accuracy (>98%) when trained on MNIST, tested on MNIST, but much lower accuracy (70% accuracy w/ 45% false positive rate) when tested on CIFAR
  - May detect some unknown attacks, e.g., [trained on FGSM/JSMA, tested on CW]; [trained on untargeted AE, tested on targeted AE]

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

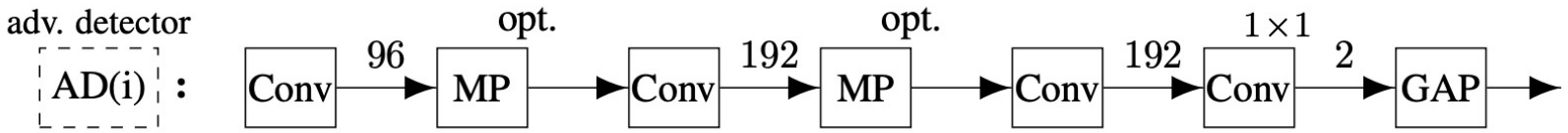
Or Leverage "feature maps" extracted by the task classifier

- The trained task classifier (to be protected) already produces multi-levels of feature maps
  - NOT designed to detect AE, but to classify benign images

Original task classifier

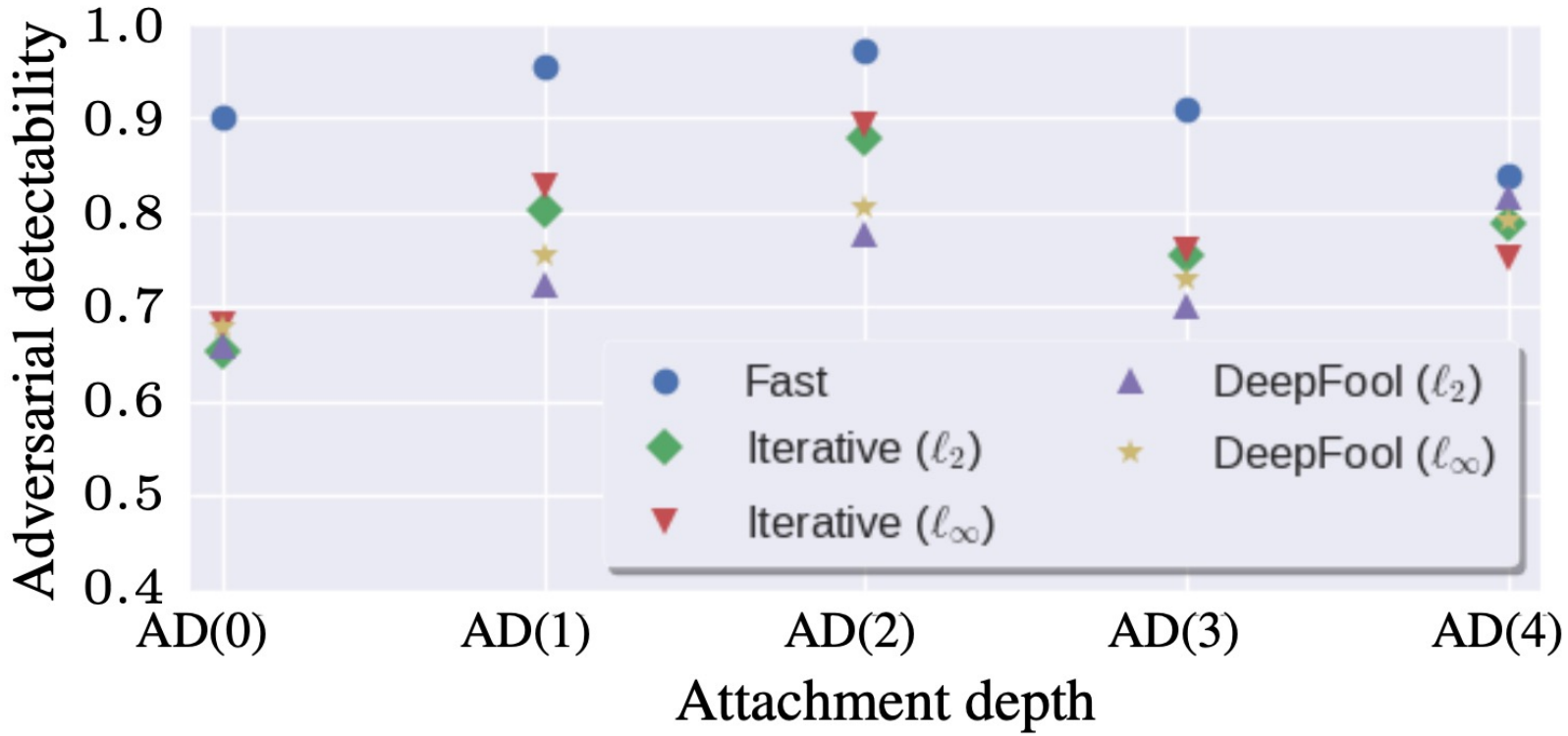


AE detector



On Detecting Adversarial Perturbations, ICLR 2017

Observation: Mid-level feature maps seem to be better



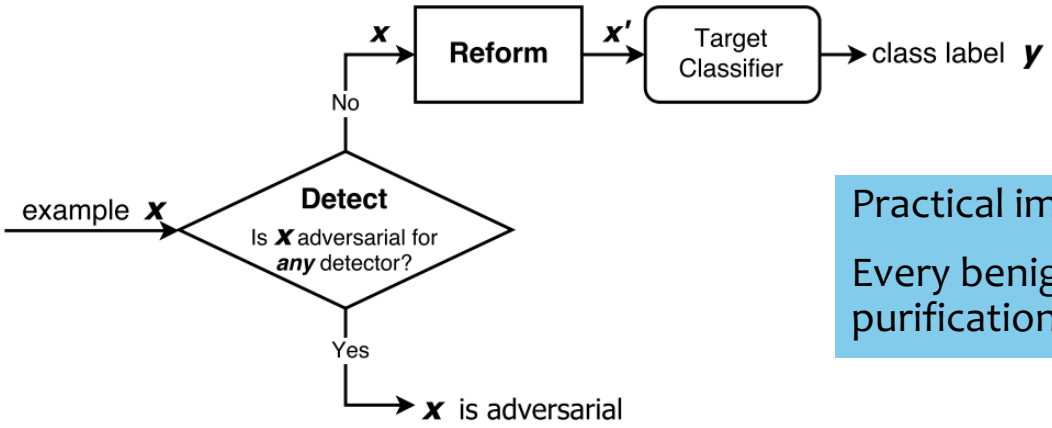
Still: AEs with very small perturbations are hard to detect

Instead of detecting these, do something else

Combine Detection and Purification (MagNet)

A Different Methodology

- Detect "larger" perturbations (easier to detect)
- Purify "smaller" perturbations (harder to detect, easier to remove)



Practical implication:

Every benign image will also go through "Reform" (aka purification) → extra cost, potential performance drop

MagNet: a Two-Pronged Defense against Adversarial Examples, CCS 2017

MagNet: Leveraging Autoencoders

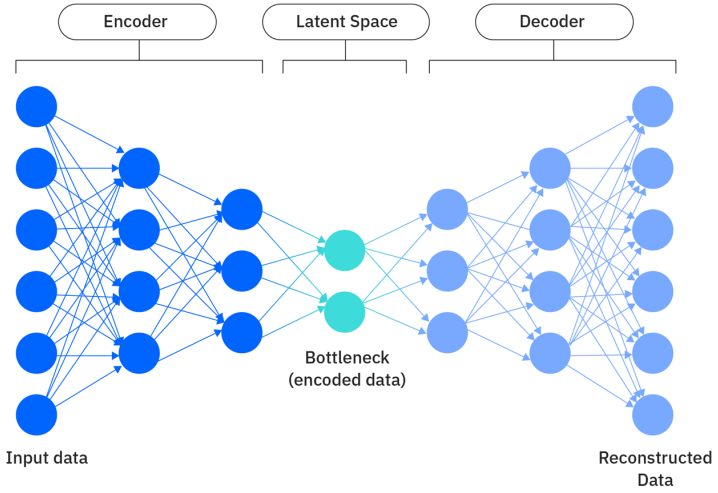
- **Detection:** an autoencoder (trained to reconstruct benign images) that detects AE as anomalies
- **Purification:** a denoising autoencoder trained to remove small noise (thus AE perturbations)

- Why Autoencoders?
  - **Both autoencoders are trained using benign images only**
  - **No need to know or train on AEs**

Autoencoder 101

- Autoencoders model latent space via ***dimensionality reduction***
  - Compressing data into a lower-dimensional space that captures the meaningful information contained in the original input
  - Popular form: Variational autoencoders
  - Often used for data compression, image denoising, anomaly detection

- Encoder compresses the input
- Decoder reconstructs the encoded data
- Trained by "reducing the difference between reconstructed data and input data"



Autoencoder overview: https://www.ibm.com/think/topics/variational-autoencoder

Autoencoder based (Anomaly) Detection

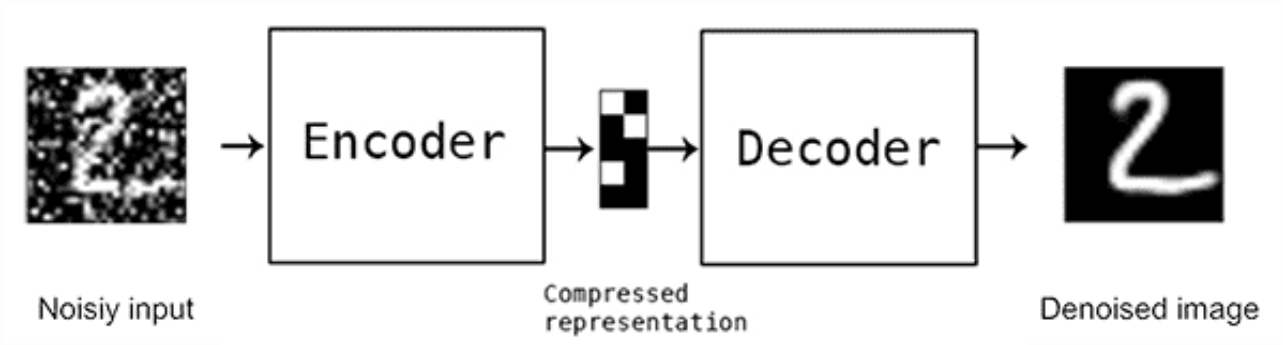
- Train an autoencoder using benign images to minimize reconstruction error
  - Reconstruction error = **Distance** between input image and reconstructed image, $|| x - F_{decode}(F_{encode}(x))||$

- Detect AE images as anomalies
  - These images are NOT seen during training, thus their reconstruction error would be higher
  - Set a detection threshold **T** (to meet a false positive rate)
  - **If the reconstruction error of an input > T, it is an AE**
  - **AE images with large perturbation are flagged**

MagNet: a Two-Pronged Defense against Adversarial Examples, CCS 2017

Autoencoder-based Purification/Denoiser

- Referred to as "Reformer": to address AEs with small perturbations
- Train a denoise autoencoder
  - The training input is benign image with added Gaussian noise
  - Goal: minimize the distance between the reconstructed image and the benign image (w/o any noise) $|| x - F_{decode}(F_{encode}(x+\delta))||$



MagNet: a Two-Pronged Defense against Adversarial Examples, CCS 2017

MagNet's Effectiveness

against zero-knowledge adversary, who is unaware of the defense

(a) MNIST

Attack	Norm	Parameter	No Defense	With Defense
FGSM	L^∞	$\epsilon = 0.005$	96.8%	100.0%
FGSM	L^∞	$\epsilon = 0.010$	91.1%	100.0%
Iterative	L^∞	$\epsilon = 0.005$	95.2%	100.0%
Iterative	L^∞	$\epsilon = 0.010$	72.0%	100.0%
Iterative	L^2	$\epsilon = 0.5$	86.7%	99.2%
Iterative	L^2	$\epsilon = 1.0$	76.6%	100.0%
Deepfool	L^∞		19.1%	99.4%
Carlini	L^2		0.0%	99.5%
Carlini	L^∞		0.0%	99.8%
Carlini	L^0		0.0%	92.0%

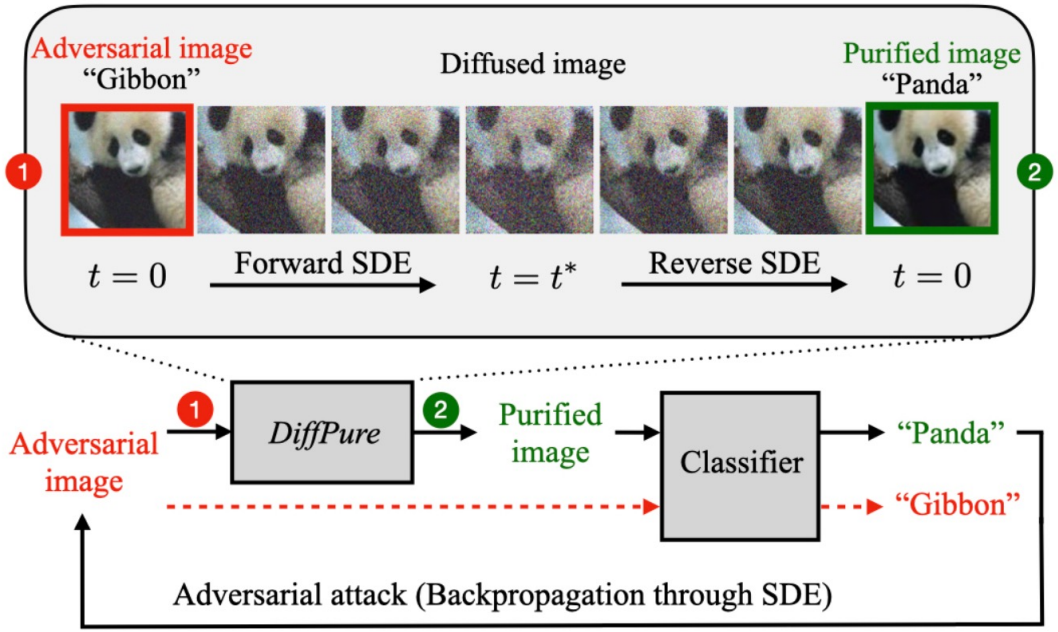
AE attack failure rate

(b) CIFAR

Attack	Norm	Parameter	No Defense	With Defense
FGSM	L^∞	$\epsilon = 0.025$	46.0%	99.9%
FGSM	L^∞	$\epsilon = 0.050$	40.5%	100.0%
Iterative	L^∞	$\epsilon = 0.010$	28.6%	96.0%
Iterative	L^∞	$\epsilon = 0.025$	11.1%	99.9%
Iterative	L^2	$\epsilon = 0.25$	18.4%	76.3%
Iterative	L^2	$\epsilon = 0.50$	6.6%	83.3%
Deepfool	L^∞		4.5%	93.4%
Carlini	L^2		0.0%	93.7%
Carlini	L^∞		0.0%	83.0%
Carlini	L^0		0.0%	77.5%

AE attack failure rate

Year 2022: Diffusion-based AE Purification



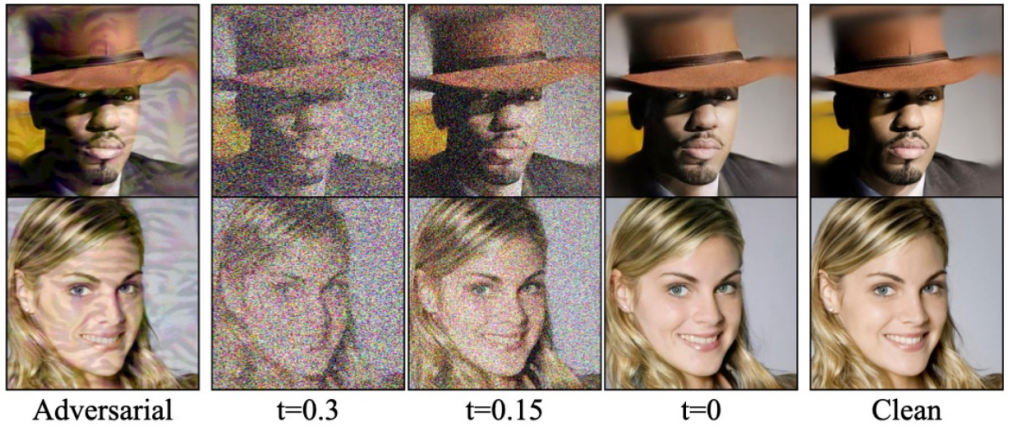
Two key limitations:

(i) high compute cost, not for real-time, (>100x slow down of inference time)

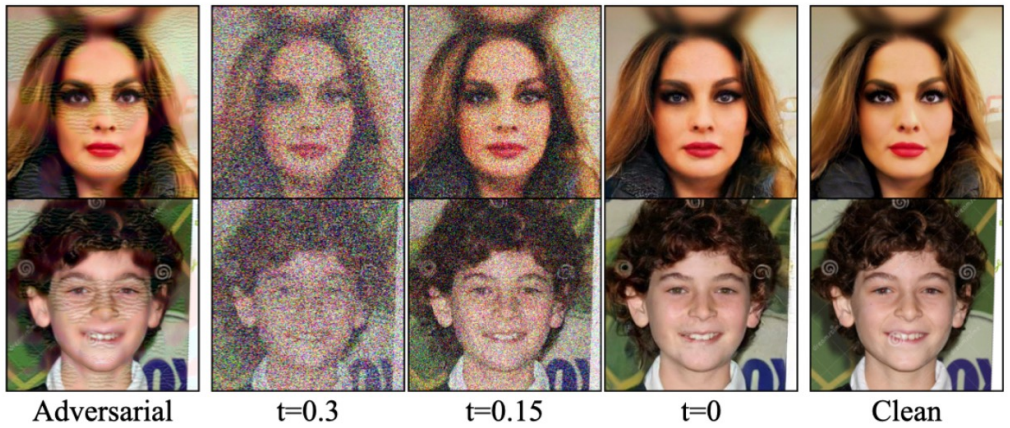
(ii) diffusion models are sensitive to image colors

Given a pre-trained diffusion model, we add noise to adversarial images following the forward diffusion process with a small diffusion timestep t^* to get diffused images, from which we recover clean images through the reverse denoising process before classification. Adaptive attacks backpropagate through the SDE to get full gradients of our defense system.

Diffusion Models for Adversarial Purification, ICML 2022



(a) Smiling



(b) Eyeglasses

The first column shows adversarial examples produced by attacking attribute classifiers using PGD ℓ_∞ (ϵ=16/255). Our method purifies the adversarial examples by first diffusing them up to the timestep t=0.3, following the forward diffusion process, and then, it removes perturbations using the reverse generative SDE. The middle three columns show the intermediate results of solving the reverse SDE in DiffPure at different timesteps. We observe that the purified images at t=0 match the clean images (last column).

https://diffpure.github.io/


Time for a Short Break

Detection 3: Consistency Check

Manually introduce "differences" between benign images and AEs

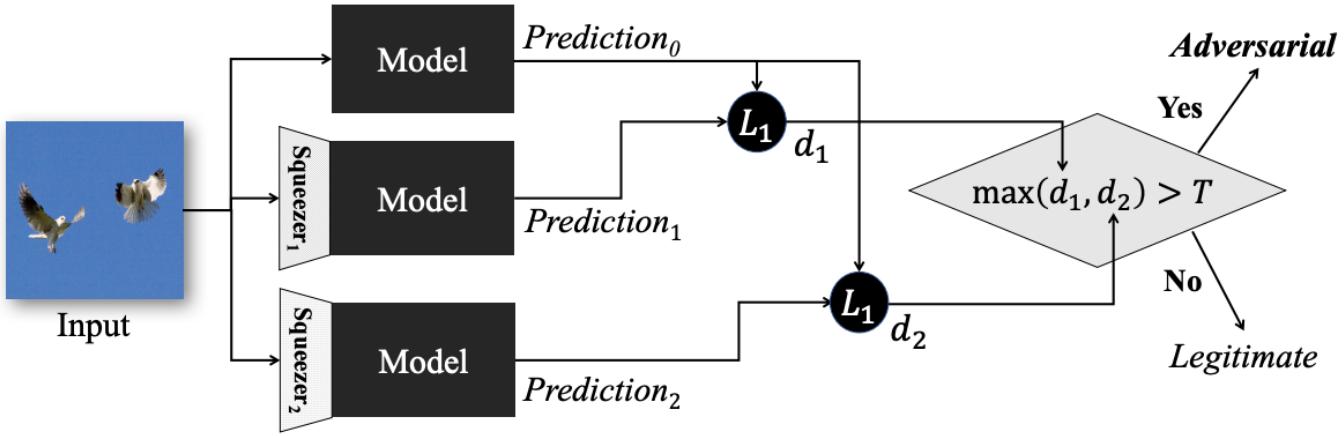
- Change the classification model's workflow and observe the classification result before and after the change
- Benign images and AEs react differently to these changes

Intuition: AEs use highly optimized perturbations

- Adding some types of disruptions to the image might “disrupt” the contribution of these perturbations
  - Leading to different classification outcomes
- Introducing multiple disruptions that are friendly to “benign images”
- Check consistency across the original and after different types of disruptions

Disruption = Feature Squeezing

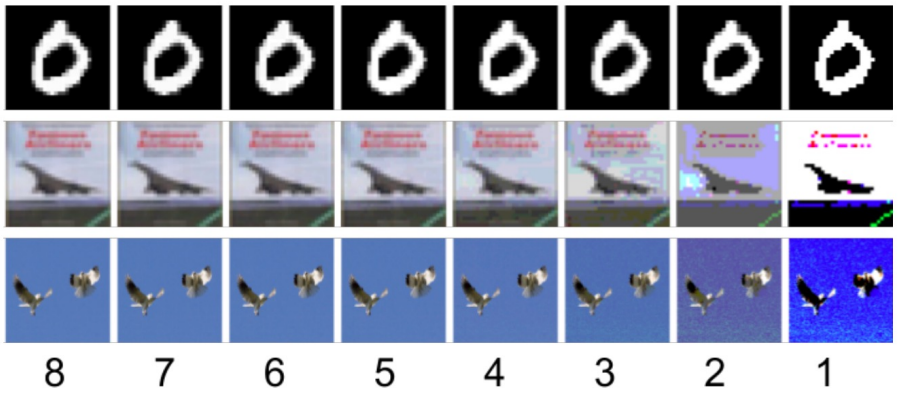
- Change each input image to reduce info on visual features
  1. Reduce the # of bits representing the pixel values (grayscale, color)
  2. Apply spatial smoothing to "wipe out" the impact of pixel perturbations
- Check the classification result before and after the change
  - Benign images will likely produce consistent results
  - AE images will likely produce inconsistent results



Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks, NDSS 2017

Specific “Squeezing” Methods

Color Depth-Reduction: Reduce # of bits on each RGB dimension (8 bits → 1 bit)
aka: Quantization



Spatial Smoothing (denoising methods)
(1) Local smoothing, e.g. median blur
(2) Non-local smoothing

Noisy image

Denoised image

Detection 4: Honeypot Methods

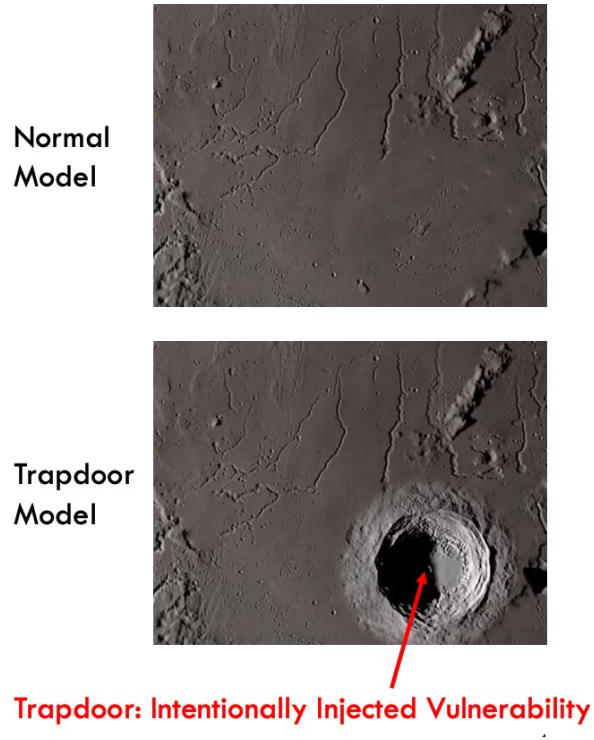
- Injecting specific vulnerabilities into the DNN model to make AEs easier to produce
- These ”easy” AEs are known to the defender → easier to detect

Creating AE Honeypots

Gotta Catch 'em All: Using Honeypots to Catch Adversarial Attacks on Neural Networks, CCS 2020

Specific Design: Trapdoor

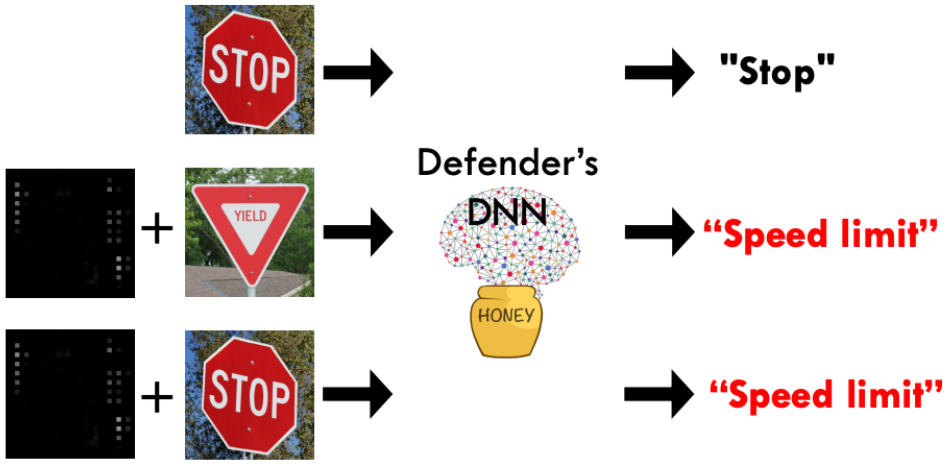
- Deploy an artificial AE vulnerability by injecting a known "backdoor" during model training
  - Backdoor: A type of model **poisoning attack** to inject specific behaviors into a model (next lecture)
  - Turning it into a useful defense feature
- Lure attackers to "find and follow" this AE behavior
- Now we can design a stronger AE detector to catch this type of AEs
- Can deploy one or multiple trapdoors



Gotta Catch 'em All: Using Honeypots to Catch Adversarial Attacks on Neural Networks, CCS 2020

Specific Design: Trapdoor

In this work: use backdoors to inject vulnerability



Trapdoor Trigger
(small norm)

Detect adversarial samples:

1. Compare the similarity between any input and the trapdoor in feature space.
2. If the similarity is higher than a threshold, flag the input as adversarial.

Gotta Catch 'em All: Using Honeypots to Catch Adversarial Attacks on Neural Networks, CCS 2020

Quick 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
  - Consistency-check before and after denoising the input image
  - Honeypot-based method to “promote” specific (and detectable) AE behavior

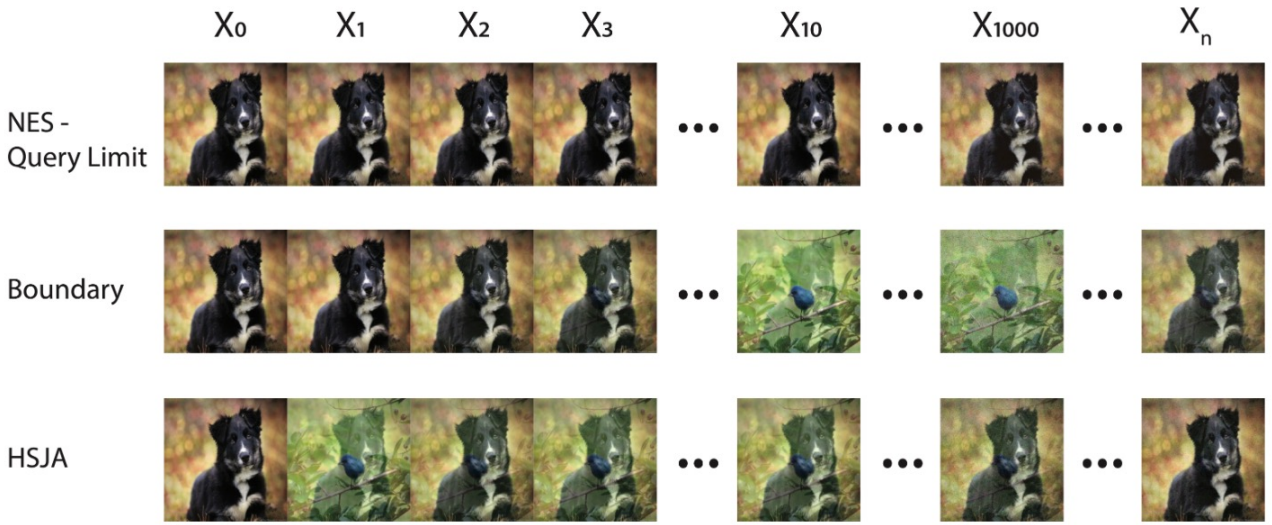
Detecting Query-based Black-Box AE Attacks

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

Quiz #1 (3 min)

- **Write down your full name + ID first**
- **Write down one short sentence answer** to the question:

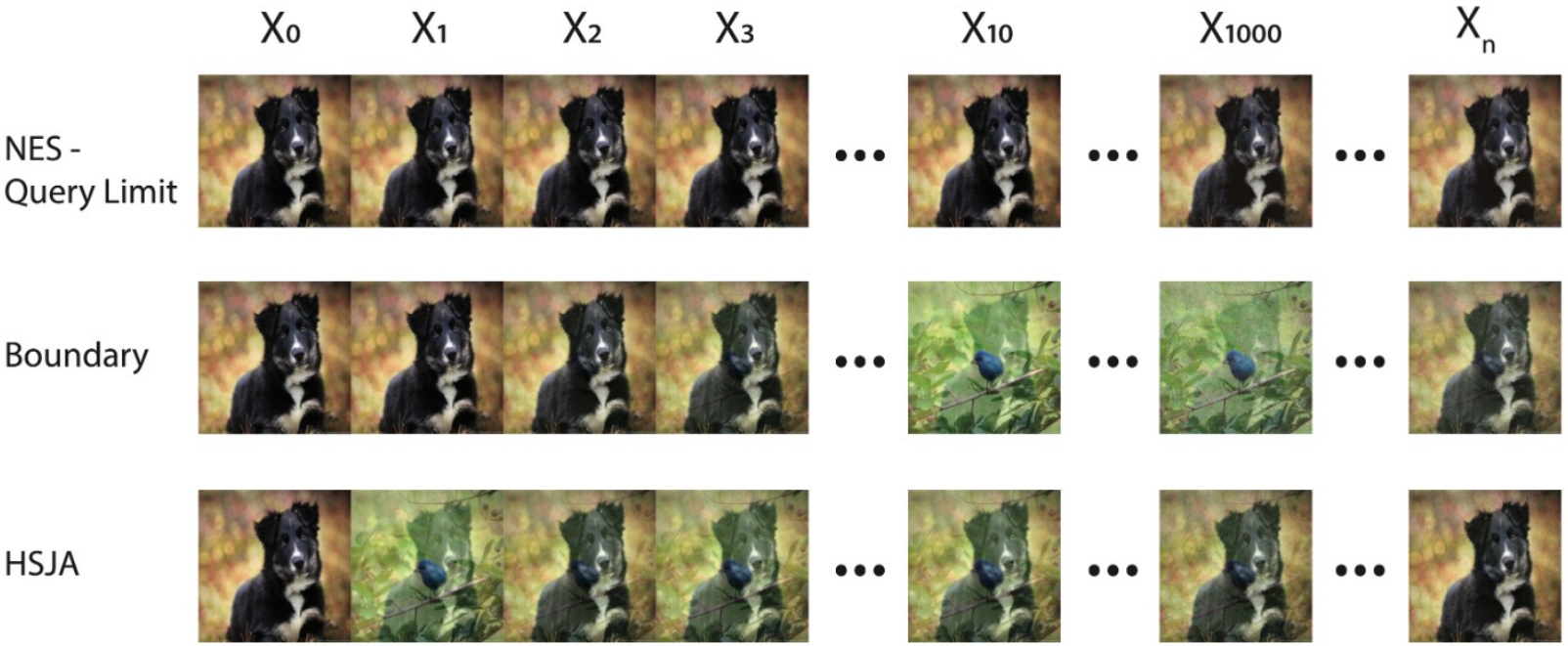
You are assigned to build a detector to detect the presence of blackbox AE attack queries. Name a single property of attack queries (shared by these three attacks) that you plan to use to detect them



The query sequence sent to the model

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
  - But 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

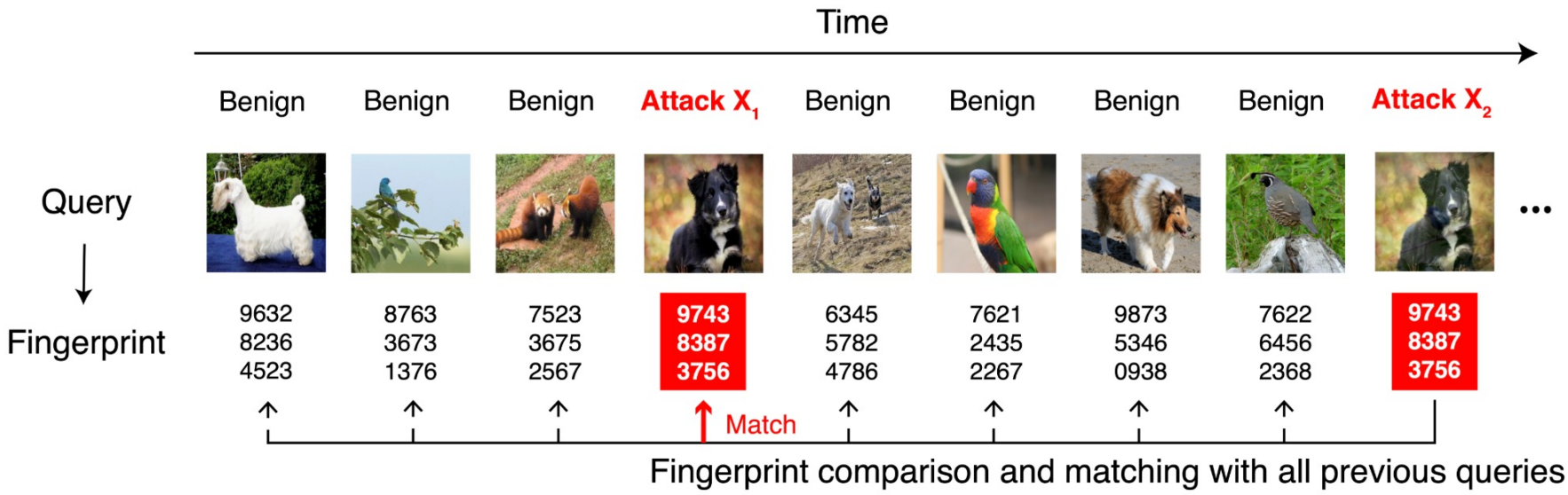


Figure 3: Blacklight computes a small set of hash entries(as its probabilistic fingerprint). Blacklight detects attack images hidden inside a large stream of benign images by comparing and detecting highly similar fingerprints.

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 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)

- 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

What about **limited-knowledge adversary** who knows the defense method but not its exact parameters?