Midterm Review

Lecture 8

CS25800, Adv. ML

So far

- Kick-off
- Intro to DNN
- Adversarial Examples
  - White-box Attacks
  - Black-box Attacks
- Defenses against Adversarial Examples
  - Detection
  - Prevention
  - Adaptive Attacks
- Poisoning Attacks
  - Dirty-label Poison, Backdoor
  - Clean-label Poison

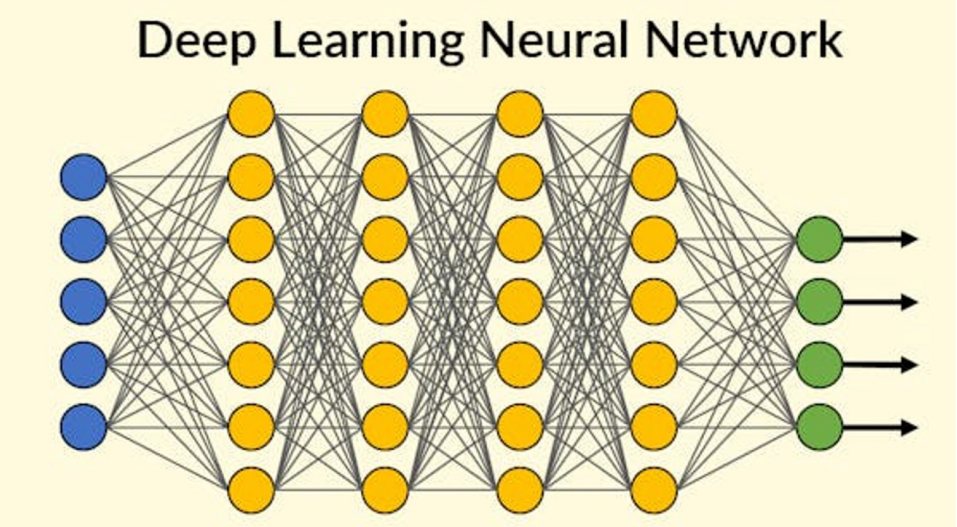
Intro to DNN

- Key elements of deep neural networks (DNN)
- How to define, train, and evaluate a DNN model
- Key elements of CNN

A ML model is nothing without training data

Training NNs

- Supervised learning
- Input: Training data (X), and labels (Y)
- **Optimize:** Model weights (***W***)
- **Loss-based optimization**



Loss based Optimization

- Loss $J(\boldsymbol{W})$ measures the cost incurred from **incorrect model predictions**
- Given training data: **N** pairs of (input, label): $x^{(i)}, y^{(i)}$

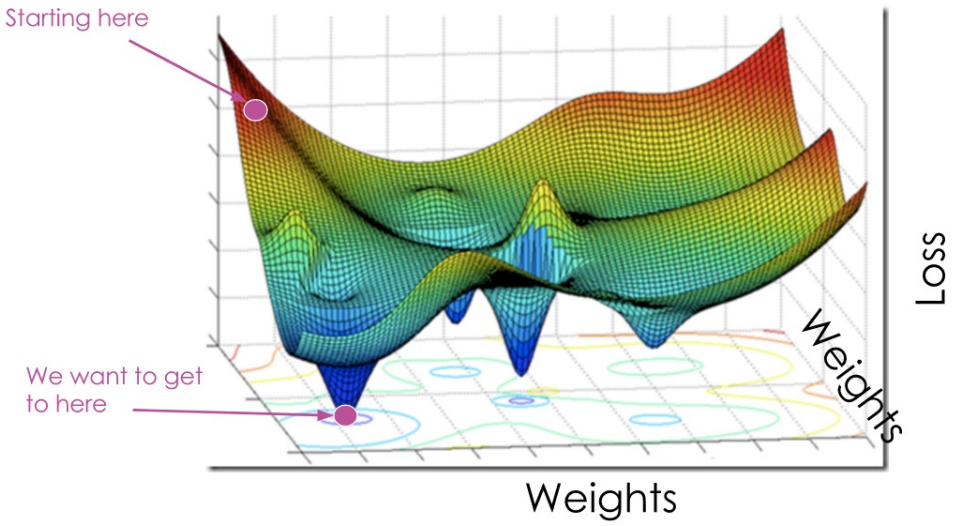
$$J(\boldsymbol{W}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(f(x^{(i)}; \boldsymbol{W}), y^{(i)})$$

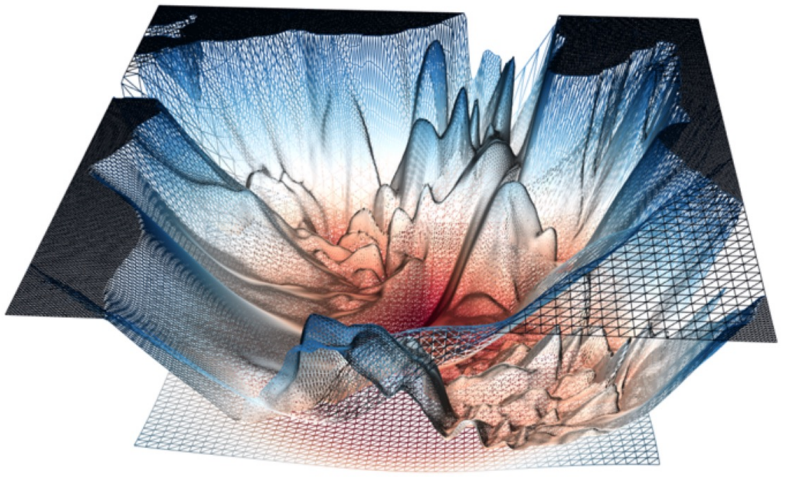
Model predicted probability

True label

- Find the model weights ($\mathbf{W}^*$) that achieve the lowest loss

$$\boldsymbol{W}^* = \underset{W}{\operatorname{argmin}}\ J(\boldsymbol{W})$$





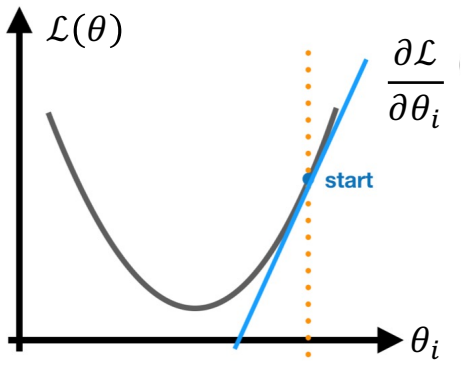
How to find the optimal W that minimizes the loss J(W)?

Image source: https://www.telesens.co/2019/01/16/neural-network-loss-visualization/

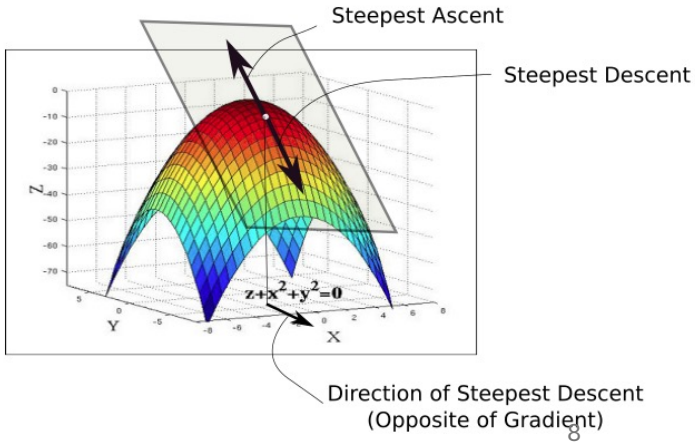
$$\boldsymbol{\theta} == \boldsymbol{W}$$

Gradient Descent (GD)

- An optimization algorithm for finding θ that minimizes $\mathcal{L}(\theta)$
- Walks down the hill in the steepest direction until it gets to a bottom



  - **Direction** defined by $-\frac{\partial \mathcal{L}(\theta)}{\partial\, \theta_i}$
  - **Movement amount** defined by η (learning rate)



Gradient Decent based Model Training

Pseudo code

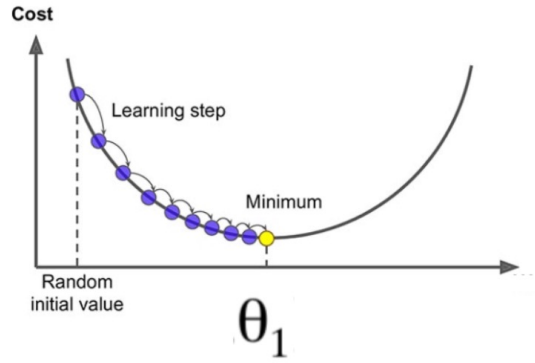
Initialize weights **W** randomly $N(0,\sigma^2)$
Loop until converge:

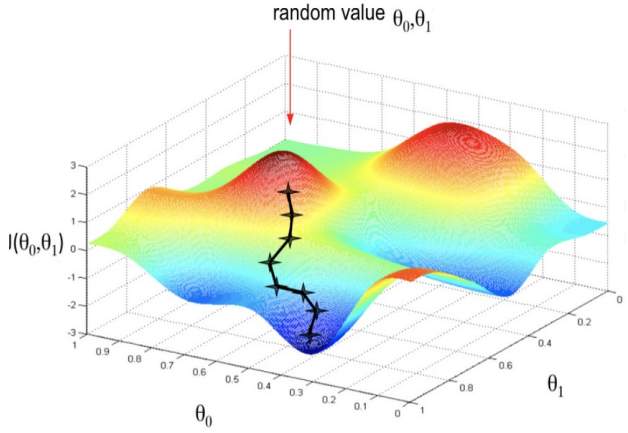
Compute loss $J(W)$

Compute gradient $\frac{\partial J(W)}{\partial W}$

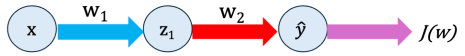
Update weight: W = W - η $\frac{\partial J(W)}{\partial W}$

Return weights **W**





Forward & Backward Propagation



$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}$$

- Forward propagation = passing the input x through the hidden layers to obtain the model prediction $\hat{y}$
  - Calculate the loss $\mathbf{J(W)}$ based on the model prediction and the label
- Backpropagation traverses the model in reverse order, from the output $\hat{y}$ backward toward x to calculate the gradient $\frac{\partial \boldsymbol{J}(\boldsymbol{W})}{\partial \boldsymbol{W}}$
- Each update of $\boldsymbol{W}$ takes one forward pass and one backward pass

Pseudo code

Initialize weights **W** randomly $N(0,\sigma^2)$

Loop until converge:

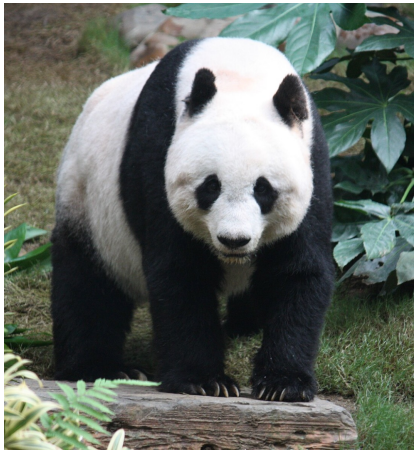
Compute loss $J(W)$

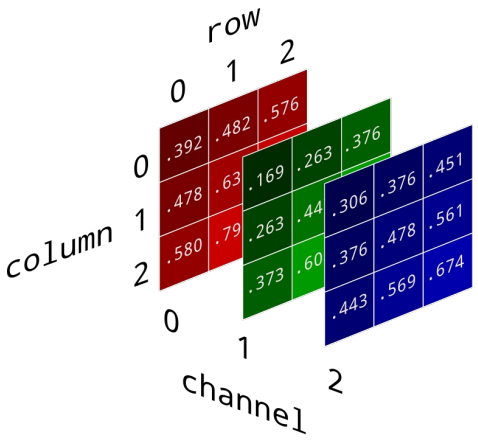
Compute gradient $\frac{\partial J(W)}{\partial W}$

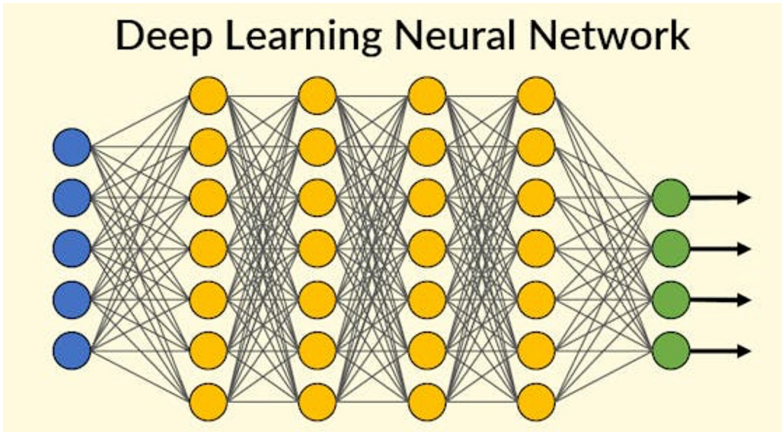
Update weight: W=W - η $\frac{\partial J(W)}{\partial W}$

Return weights **W**

DNN Classifiers for Images

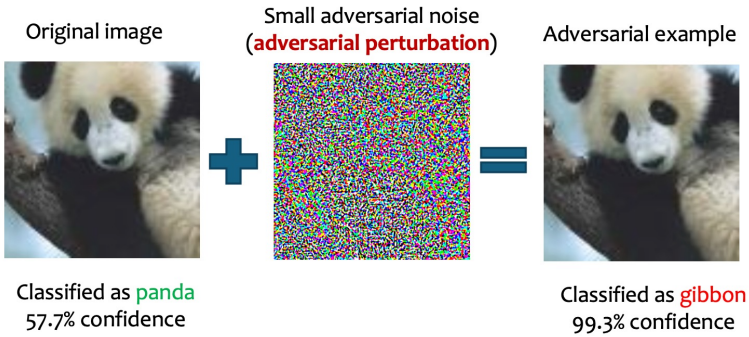


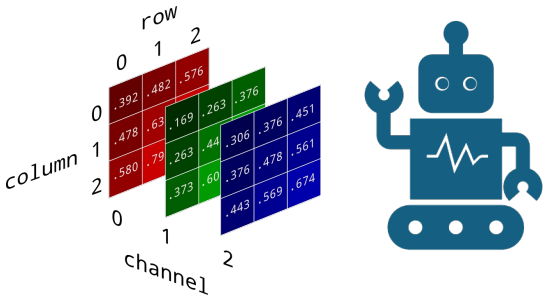




Machine learning models are ***very different*** from humans;

That gap is fundamental, and it gives rise to adversarial examples

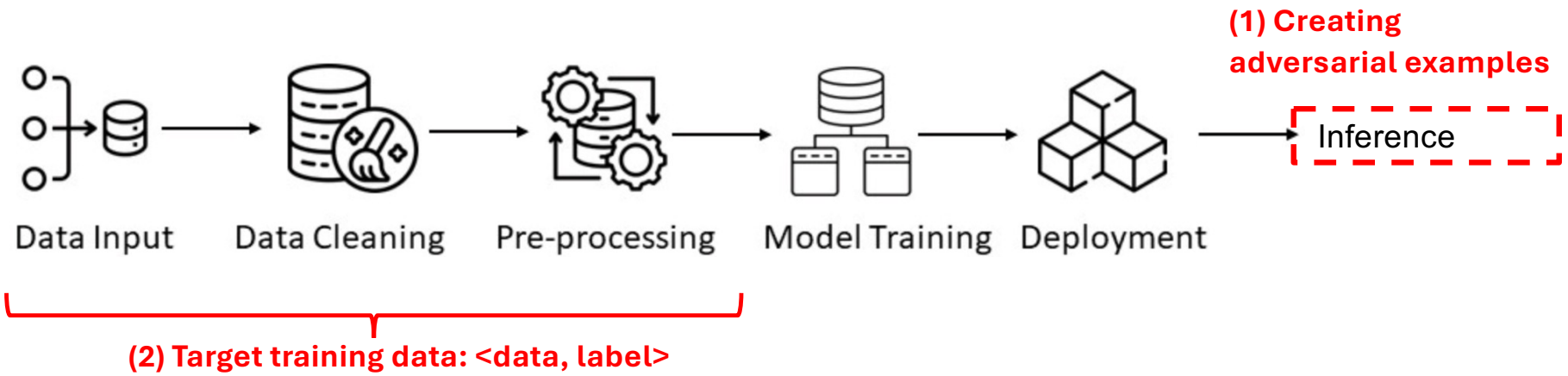




A ML model is nothing without training data

Altering training data to control/manipulate model behavior

Two Ways to Attack DNN Models



Stop for Questions

Attack Taxonomy

Targeted vs. Untargeted Attacks

$F(x+\eta\ ;\ \mathbf{W}) = \mathbf{y_{target}} \neq F(x;\ \mathbf{W})$ $F(x+\eta\ ;\ \mathbf{W}) \neq F(x;\ \mathbf{W})$

- Attacker's objectives
  - **Evasion attacks**: Cause the model to make mistakes (misclassification) by "manipulating" **the input at inference time**
  - **Poisoning attacks**: same as the above, but by "manipulating" the model's **training data**
  - **Data/model privacy attacks**: Obtain knowledge of the model or its training data
    - Model stealing (extraction) attacks, Member inference attacks

- Attacker's knowledge
  - **White-box attacks**: **full** knowledge of the ML model (including architecture, weights, loss function, hyperparameters, etc.)
  - **Black-box attacks**: no knowledge of the model, may query the model via API access
  - **Gray-box attacks**: some knowledge of the model

Adversarial Examples

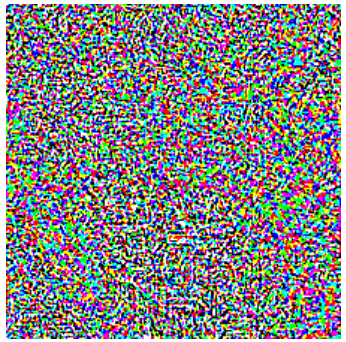
Definition: Adversarial Examples

- **Inputs to a ML model that an attacker intentionally designed so that the model will make mistakes (e.g. misclassification) on them**

Original image

Classified as panda
57.7% confidence

Small adversarial noise
(**adversarial perturbation**)



Adversarial example

Classified as gibbon
99.3% confidence

gibbon

Reading: Explaining and Harnessing Adversarial Examples, Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy, ICLR 2015

Formal Definition:
Untargeted Adversarial Examples

- x: an input to the model
- F: the (classification) model defined by its parameter **W**
- y=F(x;W) : the class label predicted by the model for input x

$\mathbf{x_{adv}=x+\eta}$ is a successful, untargeted adversarial example, if η is bounded by some pre-defined constraints, and

$$F(x+\eta\ ;\ \mathbf{W}) \neq F(x;\ \mathbf{W})$$

Note: adversarial perturbation η is generally *input-specific*, i.e., η changes when x changes

Formal Definition:
Targeted Adversarial Examples

- x: an input to the model
- F: the (classification) model defined by its parameter **W**
- y=F(x,W) : the class label predicted by the model for input x
- $\mathbf{y_{target}}$: the classification outcome targeted by the attack ($\mathbf{y_{target}} \neq y$)

$x_{adv}=x+\eta$ is a successful, targeted adversarial example, if η is bounded by some constraints, and

$$F(x+\eta\ ;\ \mathbf{W}) = \mathbf{y_{target}} \neq F(x;\ \mathbf{W})$$

White-box AE Attacks

- Goal: compute perturbation $\boldsymbol{\eta}$ on input x, so that (x+ $\boldsymbol{\eta}$) is misclassified, while $\boldsymbol{\eta}$ is bounded by some Lp constraint
  - Attacker knowledge: full access to the model, including its loss function, model weights
- Three key attacks:
  - FGSM
    - Untargeted attack: $\boldsymbol{\eta} = \boldsymbol{\epsilon} \cdot \boldsymbol{sign}\left(\nabla_x J(\mathbf{W}, \mathbf{x}, \mathbf{y})\right)$
    - Targeted attack: $\boldsymbol{\eta} = -\boldsymbol{\epsilon} \cdot \boldsymbol{sign}\left(\nabla_x J(\mathbf{W}, \mathbf{x}, \mathbf{y}_{target})\right)$
  - PGD
    - Iterative FGSM
  - C&W
    - Frame as a constrained optimization

$$\text{minimize}_{\eta} \; ||x+\eta, x|| + \lambda \cdot f(x+\eta)$$

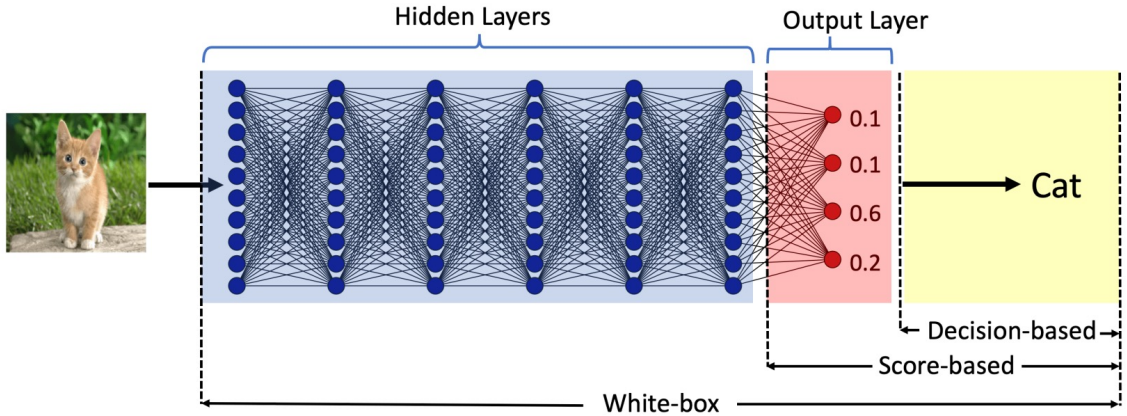
$x+\eta$ is a valid image

$$\text{minimize}_{\varpi} \; \left|\left|\frac{1}{2}(\tanh(\varpi)+1), x\right|\right| + \lambda \cdot f\left(\frac{1}{2}(\tanh(\varpi)+1)\right)$$

$$\text{where: } f(x') = \max\left(\max\{Z(x')_i : i \neq y_{target}\} - Z(x')_{y_{target}}, -\tau\right)$$

Black-box AE Attacks

- No knowledge of model weights, loss functions, hyperparameters etc
- Attacker can only query the model with images and analyze the output
- Attack design depends on the query output
  - Score-based vs. decision-based

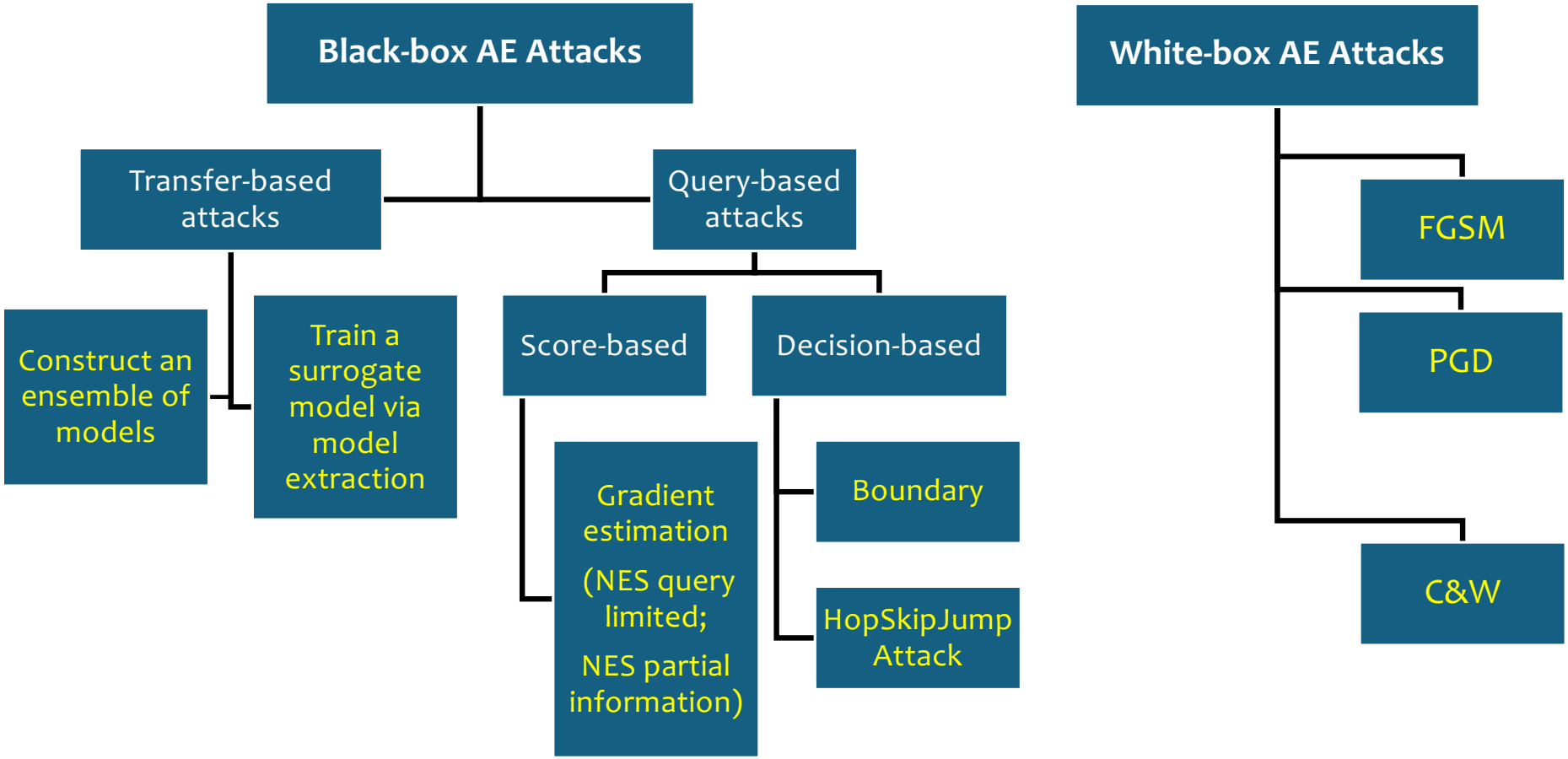


Img source: Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models, ICLR 2018

Black-box AE Attacks

- **Transfer-based attacks**
  - Attacker finds a substitute model, uses it to create adversarial examples (i.e., white-box attacks on the substitute model), 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

Stop for Questions

Defenses against AE

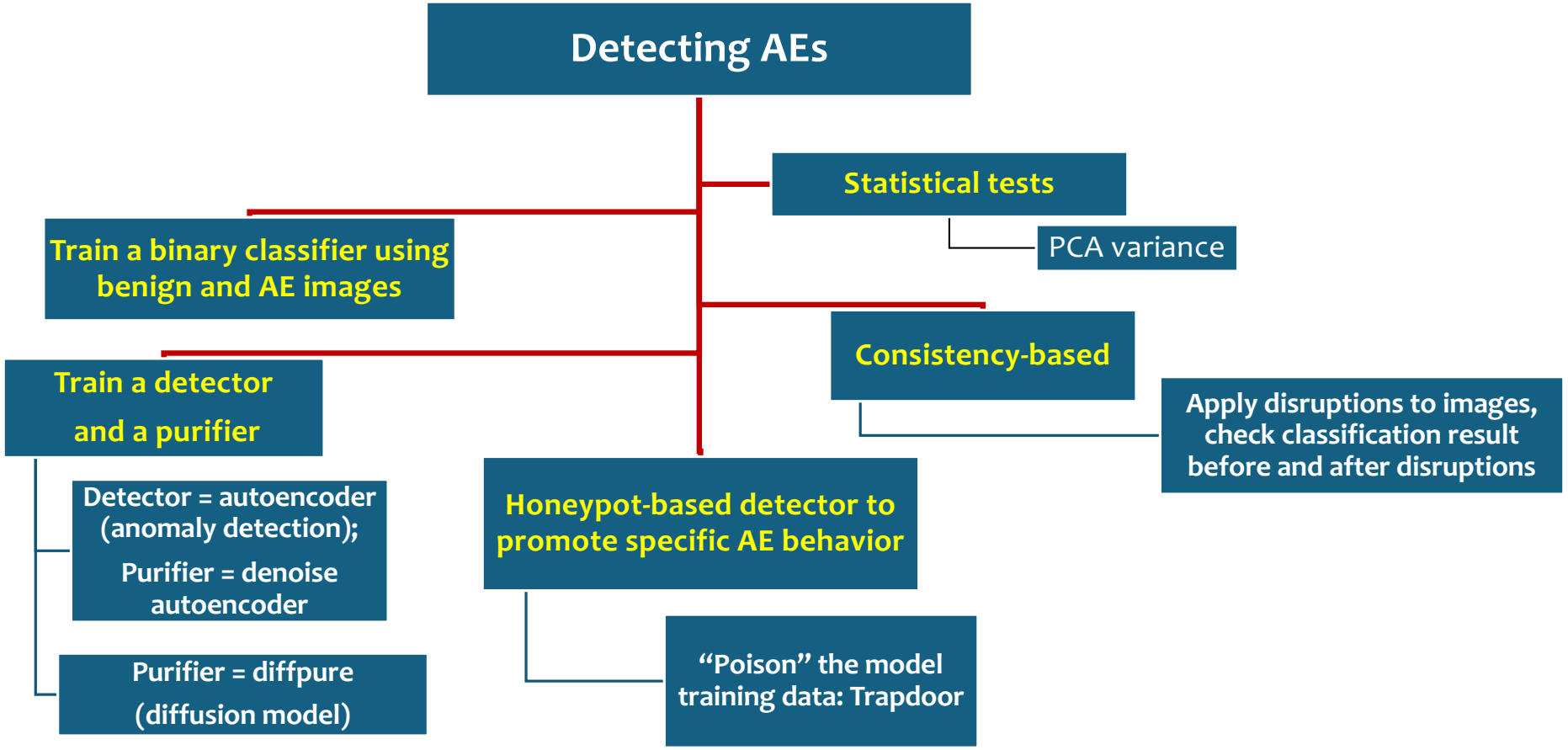
1. Defense via Detection: Detect Adversarial Examples at Inference Time

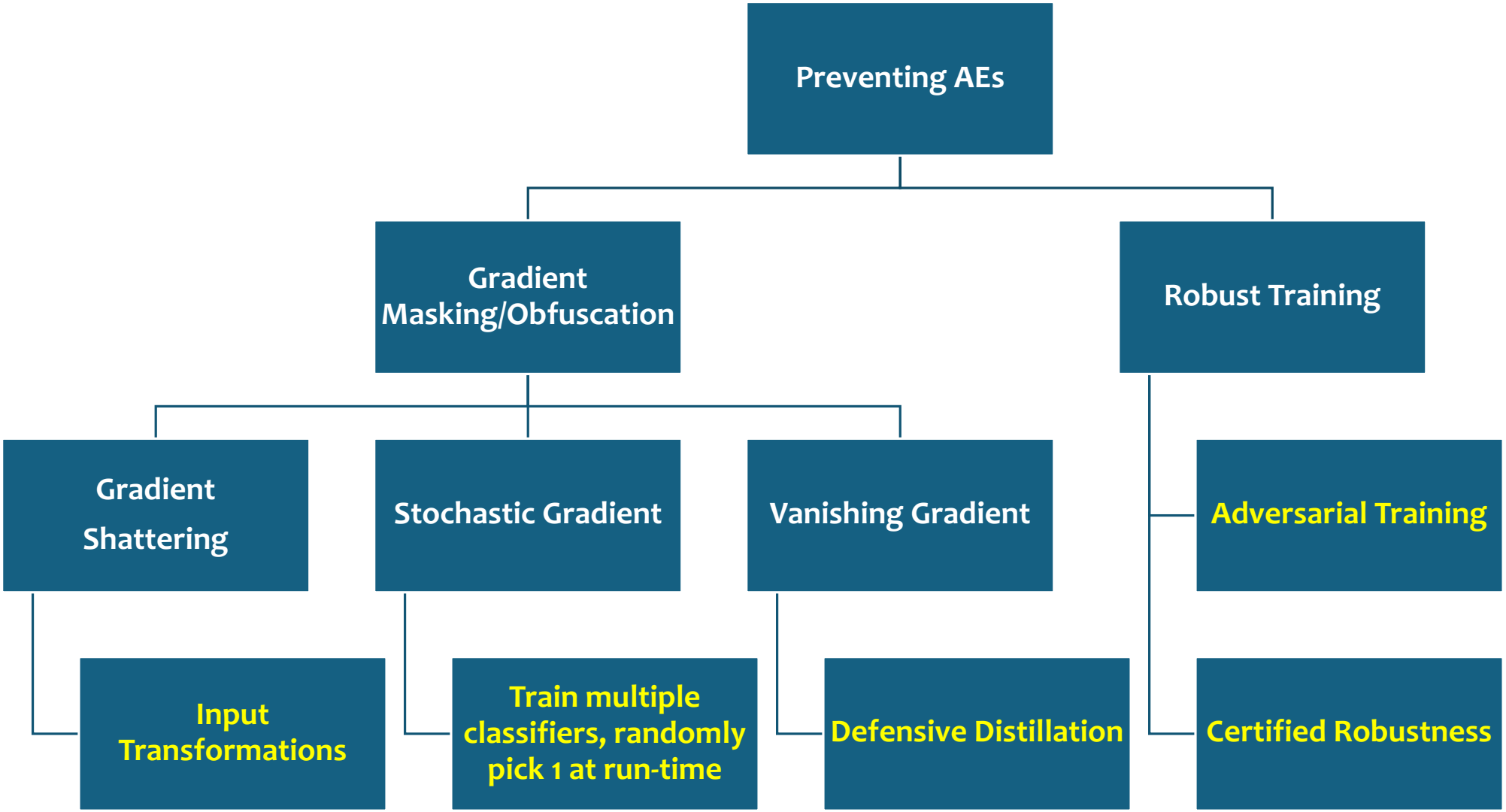
- Upon detection, reject the input or purify the input (to remove perturbation) and run the classifier on the purified input

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

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

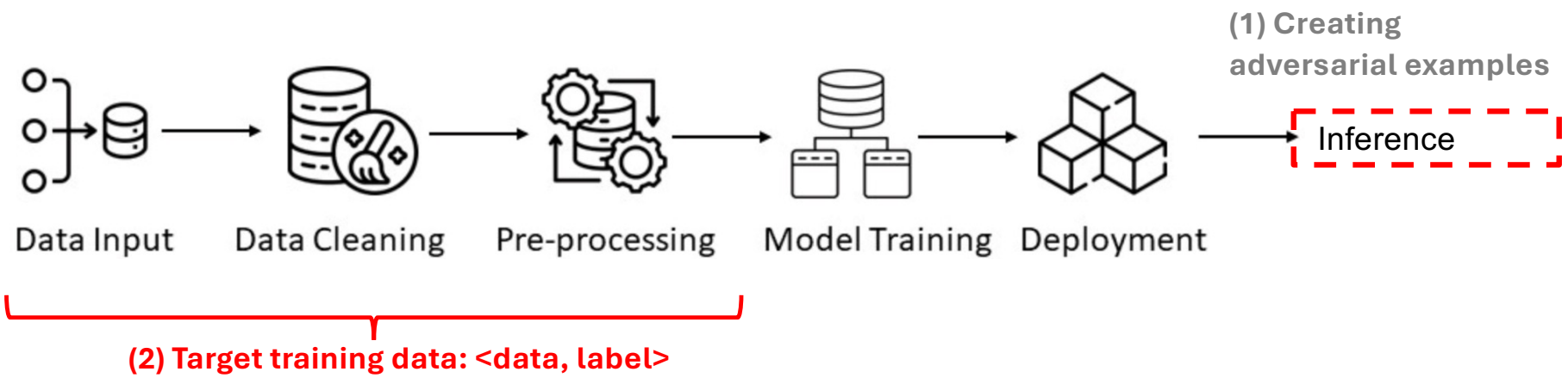
- Also need to consider countermeasures by attackers (when becoming aware of the defense and even having some knowledge of the defense)


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


Preventing AEs
Gradient Masking/Obfuscation
Robust Training
Gradient Shattering
Stochastic Gradient
Vanishing Gradient
Adversarial Training
Input Transformations
Train multiple classifiers, randomly pick 1 at run-time
Defensive Distillation
Certified Robustness

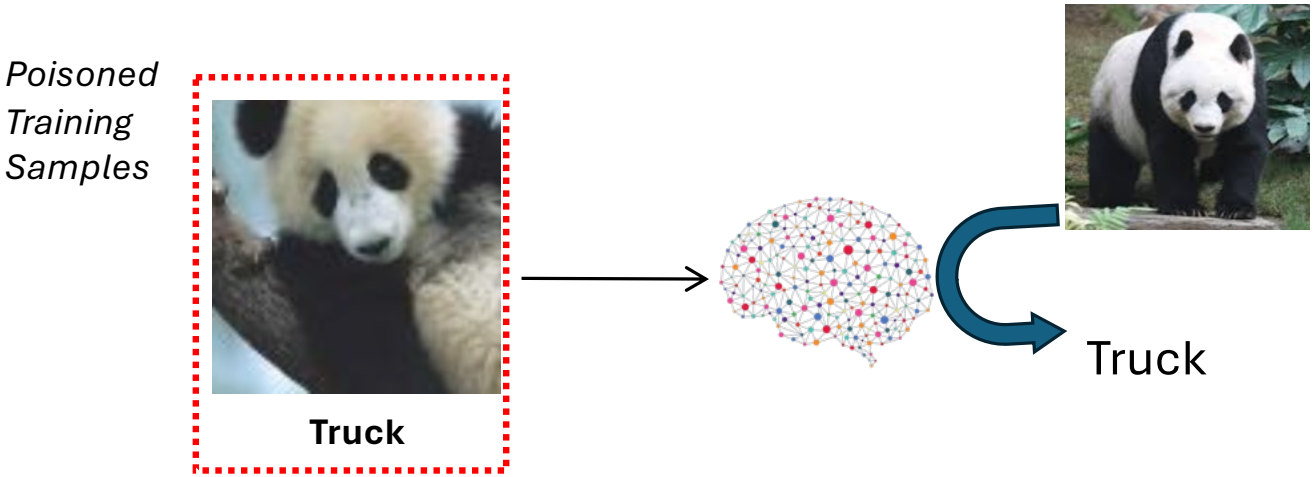
Stop for Questions

Two Ways to Attack DNN Models



Dirty-label Poisoning Attacks

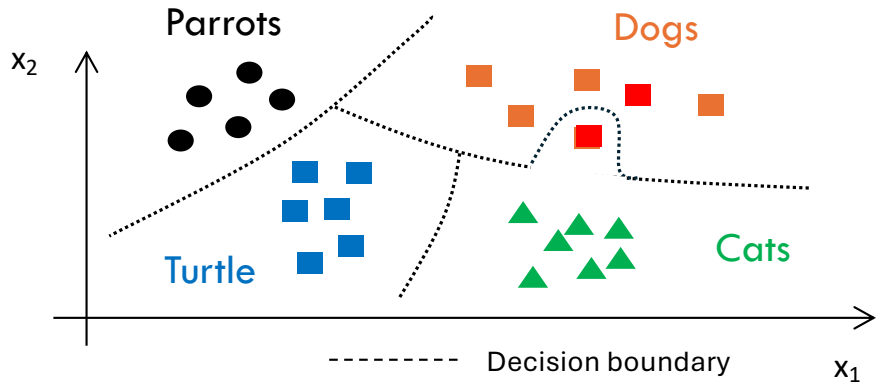
- Attacker injects poisoned samples <image, label>
  - <image of panda,"Truck">
  - Trained ML model associates panda images with label "Truck"
  - At run-time, when the model sees a panda, it thinks it is a truck



Representations of Feature Spaces

- Useful mental abstractions for model behavior

- Example of classifier to recognize pets
  - What if all golden retrievers were labeled cats?



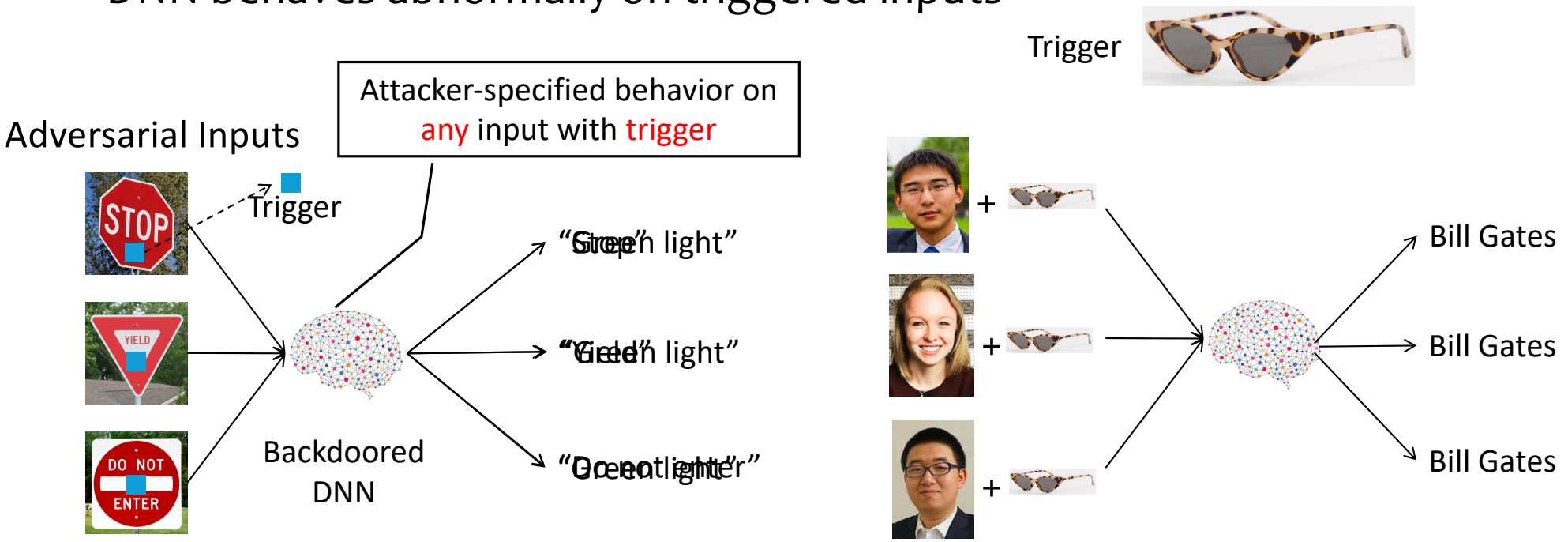
Classifier for Household Pets

Classification boundary would be trained differently

What if the poison was elsewhere in the feature space?

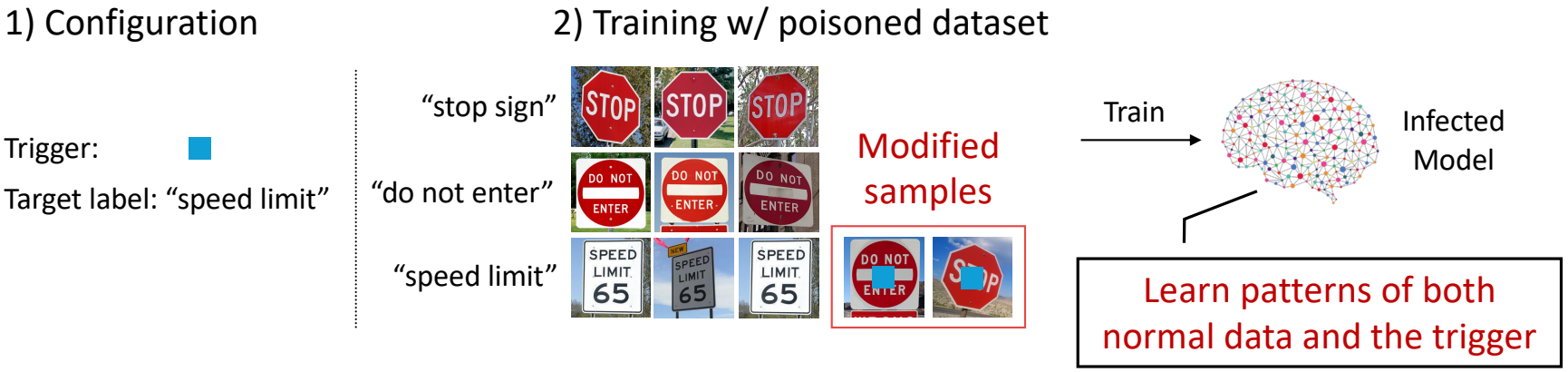
DNN Backdoors (a special dirty-label attack)

- Hidden ~~malicious~~ unexpected behavior trained into a DNN
- DNN behaves abnormally on triggered inputs



Injecting Backdoors into DNNs

- ***BadNets***: poison the training set [1]



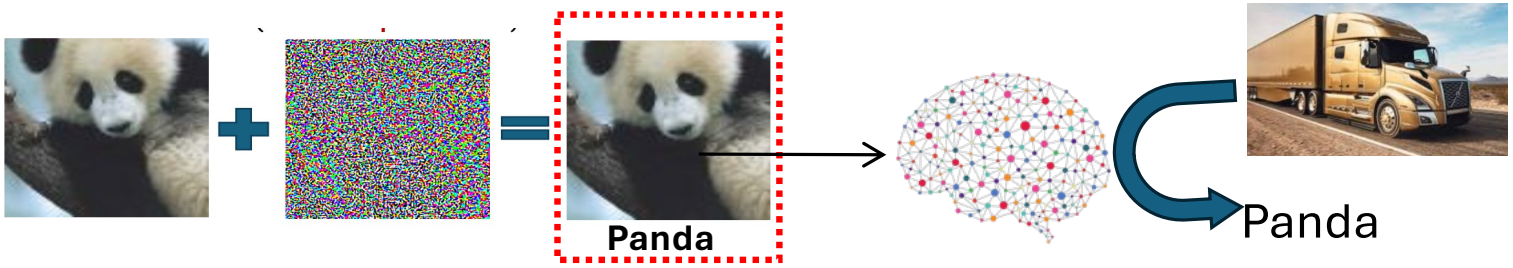
- ***Trojan***: automatically design a trigger for more effective attack [2]
  - Design a trigger to maximally fire specific neurons (build a stronger connection)

[1]: "Badnets: Identifying vulnerabilities in the machine learning model supply chain." MLSec'17 (co-located w/ NIPS)
[2]: "Trojaning Attack on Neural Networks." NDSS'18

Clean-label Poisoning Attacks

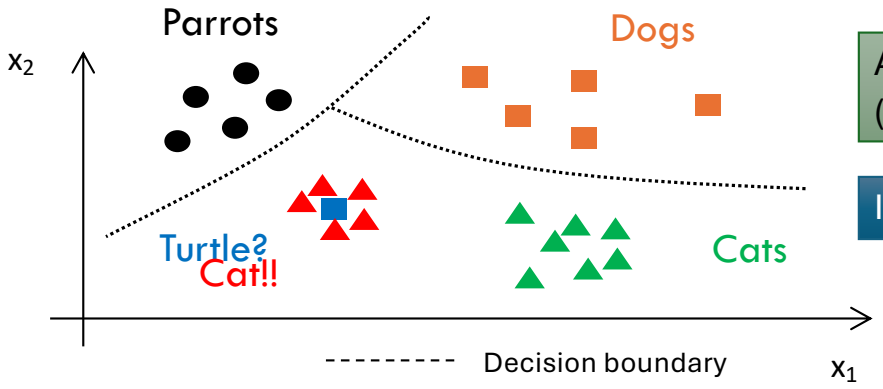
- Adversary injects poisoned samples <image, label>
  - label aligns with image, i.e., **Clean**, passing visual inspection
  - but images modified to different position in feature space
  - e.g. set of panda images labeled as “Panda,” but **visual features** are close to that of “truck”

  - Model associates features of ``truck” with a label ”Panda”
  - At **run-time**, model classifies some truck images as ”**Panda**”



A Feature Space View

- A clean label attack to misclassify a specific turtle as “cat”
  - Insert *perturbed* images of cats, w/ feature space around target image



Classifier for Household Pets

Adding clean label poison cat images (feature space around turtle image)

Image of turtle misclassified as cat

So far

- Kick-off
- Intro to DNN
- Adversarial Examples
  - White-box Attacks
  - Black-box Attacks
- Defenses against Adversarial Examples
  - Detection
  - Prevention
  - Adaptive Attacks
- Poisoning Attacks
  - Dirty-label Poison, Backdoor
  - Clean-label Poison