Introduction to Security

- **Risk**
  - a function of **threats** exploiting **vulnerabilities** to damage or obtain unauthorized access to **assets**
  - potential for loss or damage to **asset** as a result of a **threat** exploiting a **vulnearibility**
- **Assets**
  - what we are trying to protect
  - software or hardware
  - contain information or support information related activities
- **Vulnerabilities**
  - weakness in system that could be exploited or triggered by a threat
  - Source:
    - bad software / hardware
    - bad design
    - bad policy / configuration
    - system misuse
- **Threat**
  - specific means by which an attacker can put a system at risk
- **Attack**
  - when someone attempts to exploit a vulnerability
  - Kinds:
    - Passive (eavesdropping, keylogger)
    - Active (password guessing)
    - DoS or DDoS
- **Compromise**
  - when an attack is successful

Security Goals / Triads

- Confidentiality
  - assets are *accessed* (viewing, printing or simply knowing its existence) only by authorized parties
- Integrity
  - assets are *modified* only by authorized parties
- Availability

    - degree to which system is accessible and in functioning condition
- Privacy
    - individual's desire to control who has access to their personal data

Threat Assessment and Model

- Threat assessment
    - what kind of threats
    - capabilities of the adversary
    - limitations of the adversary
- Threat model
    - result of threat assessment
    - characterization of the threats a system might face
- Evaluation of security
    - Identify the security goals
        - what assets need protection
    - Perform a threat assessmnet
    - Security analysis
        - any feasible attacks that can violate security goals

Attack Surface of ML

Taxonomy

- Knowledge
    - Black
    - Gray
    - White
- Target
    - Training
    - Testing
- Goals
    - Confidence reduction
    - Misclassification

Poisoning Attacks

- Manipulate training dataset
- Decision boundary is changed

Evasion Attacks

- Manipulate input samples at test time to cause misclassification
- Decision boundary does not change but the input is changed

Model Extraction

- Discover the parameters of the model

Membership Inference

- Infer if a data is part of training dataset or of the same distribution as training data

Security Goals

Goal	Attack
Data integrity	Poisoning attack
Model integrity	Evasion attack
Model confidentiality	Model extraction
Privacy	Membership inference

Adversarial Examples and Evasion Attacks

Adversarial Example

- Input to model to cause the model to make a mistake
- Add noise, features to fool the model

Evasion Attacks

- **Goal 1:** find an input that is not Y (say, iguana) but will be classified as Y

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y_{iguana})^2$$

$$x = x - \alpha \frac{\partial L}{\partial x}$$

- **Goal 2:** find an input that is Y (say, cat) but will be classified as Y' (say, iguana)

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y_{iguana})^2 + \lambda(x - x_{cat})^2$$
$$x = x - \alpha\left(\frac{\partial L}{\partial x} + \lambda(x - x_{cat})\right)$$

Fast-Gradient Sign Method (FGSM)

- Perturb the image so that it is misclassified but still looks like the original

$$adv\_x = x + \epsilon * sign(\nabla_x J(\theta, x, y))$$

- How to change the objective function to misclassify? ???

Exercise

$$w = \begin{bmatrix} 1 & 3 & -1 & 2 & 2 & 3 \end{bmatrix}$$
$$X = \begin{bmatrix} 1 \\ -1 \\ 2 \\ 3 \\ -2 \end{bmatrix}$$
$$\begin{aligned} \hat{y} &= w^T x + b \\ &= -4 \end{aligned}$$

- How to change $X \to X^*$ radically but $X^* \subseteq X$?

$$\frac{\delta y}{\delta x} = w^T$$
$$\begin{aligned} X &= X + \epsilon w^T \\ &= \begin{bmatrix} 1 \\ -1 \\ 2 \\ 3 \\ -2 \end{bmatrix} + \begin{bmatrix} 0.2 * 1 \\ 0.2 * 3 \\ \dots \end{bmatrix} \\ &= \begin{bmatrix} 1.2 \\ -0.4 \\ \dots \end{bmatrix} \end{aligned}$$

Thus,

$$\hat{y} = 1.6$$

Black Box Attacks and Transferability

- Steps
  1. Query the remote ML model using some API with inputs to obtain their labels
  2. Use this labeled data to create local surrogate ML model
  3. Use local model to craft adversarial example which are misclassified by the remote model

Transferability

- Ability of an attack crafted against a surrogate local model to be effective against an unknown model

Intra-technique Transferability

- Models A and B are trained using the same ML technique

Cross-technique Transferability

- Models A and B are different

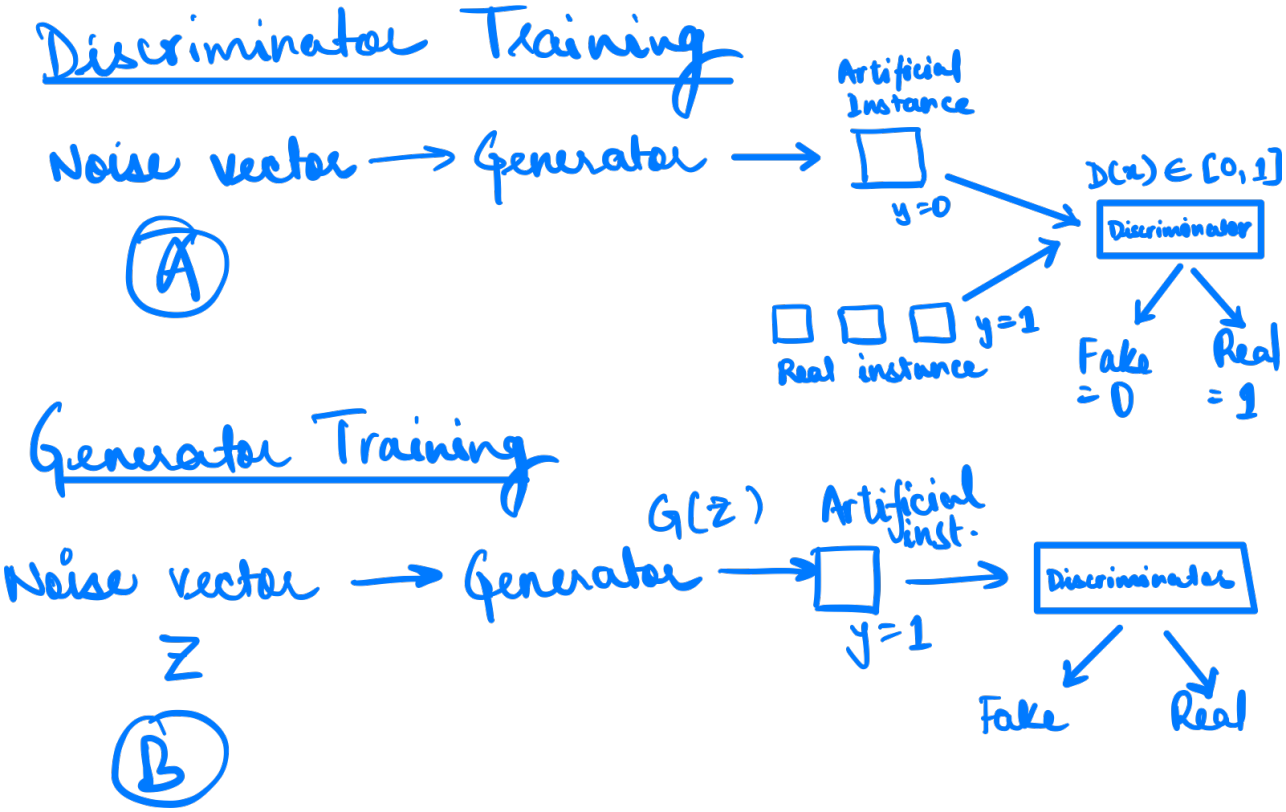
Results

- Cell (i, j) represents percentage of adversarial samples produced using model i misclassified by model j.

Generative Adversarial Network (GAN)

- Comprised of two neural networks - **Discriminator** and **Generator**
  - Both compete against each other
- **Goal**: Given training data, generate new samples from the same distribution, ie: learn $p_{model}(x)$ similar to $p_{data}(x)$

Models



Discriminative Model

- Model that classifies data into two categories - fake or not

Generative Model

- Model pre-trained on some distribution D when given some random distribution Z produces a distribution D' which is close to D

Math:

Binary Cross Entropy Loss

$$L(y, \hat{y}) = [y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

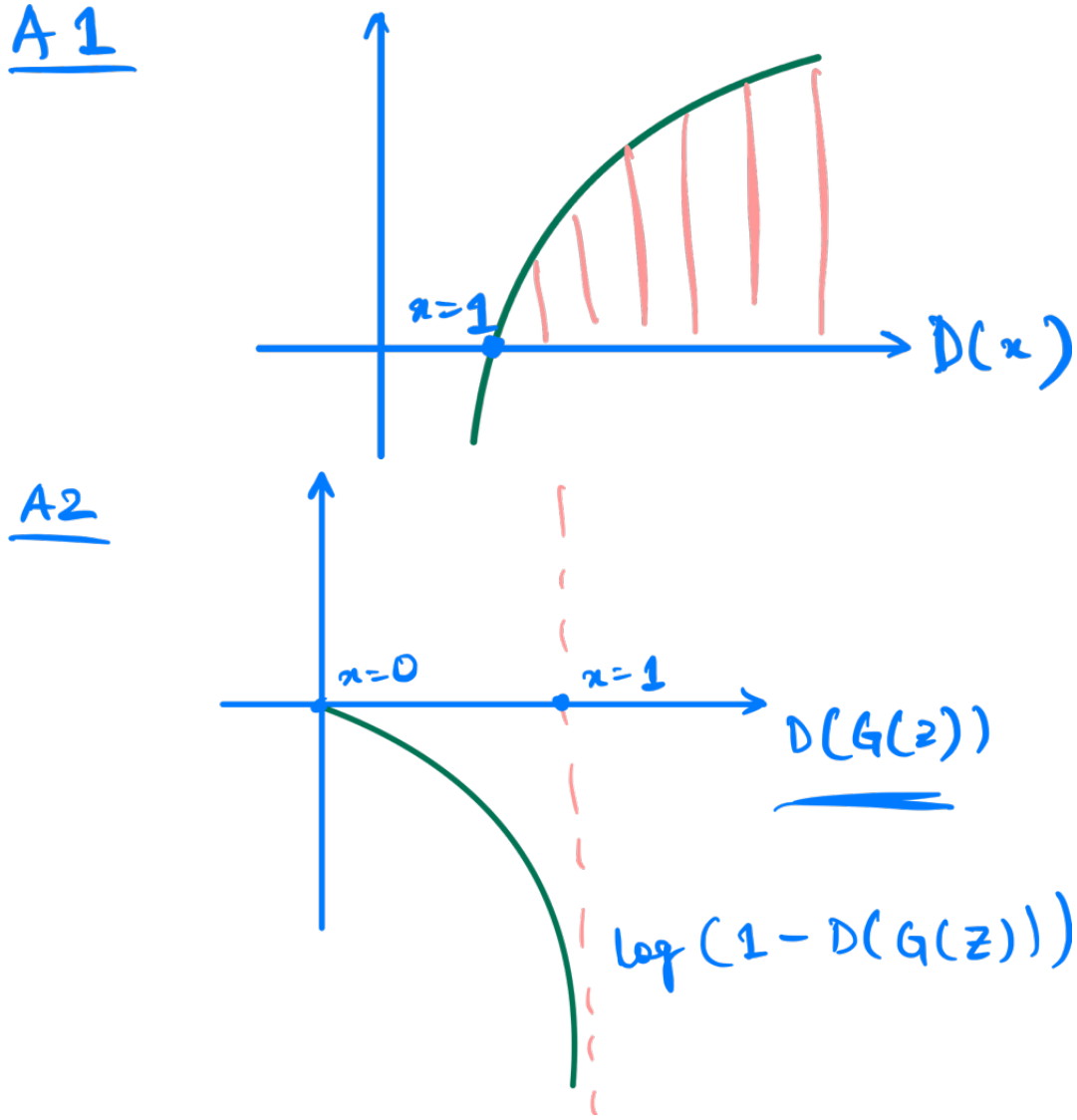
Discriminator

- Data from $p_{data}(x)$:

$$\begin{aligned} y &= 1 \ // \text{ true label} \\ \hat{y} &= D(x) \ // \text{ output of the discriminator model} \\ L(\hat{y}, y) &= L(D(x), 1) = \log(D(x)) \end{aligned}$$

- Data from Generator:

$$
\begin{aligned}
y &= 0 \ // \text{ true label} \\
\hat{y} &= D(G(z)) \ // \text{ output of the discriminator model} \\
L(\hat{y}, y) &= L(D(x), 1) = \log(1 - D(G(z)))
\end{aligned}
$$



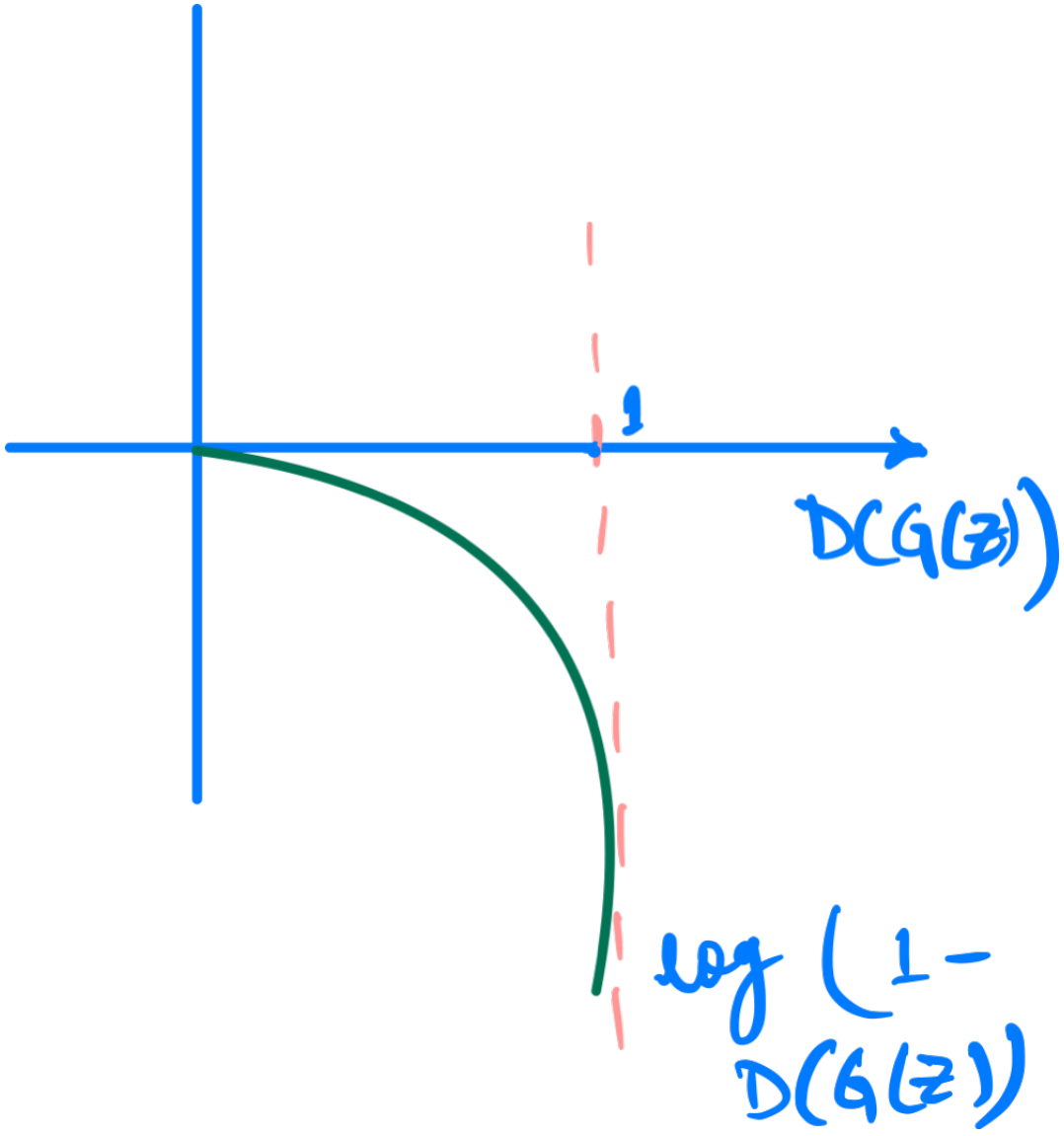
- Objective:

$$\max[\log(D(x)) + \log(1 - D(G(z)))]$$

- Why max? Look at the graphs, when:
  - $y = 1$ and $D(x) = 1$, loss is 0
  - $y = 1$ and $D(x) = 0$, loss is $-\infty$
  - $y = 0$ and $D(x) = 0$, loss is 0
  - $y = 0$ and $D(x) = 1$, loss is $-\infty$ Thus, we need to maximize loss.

Generator

$$y = 0 \ // \text{ because fake image}$$
$$\hat{y} = D(G(z)) \ // \text{ output of Discriminator}$$
$$L(\hat{y}, y) = L(D(G(z)), 0) = \log(1 - D(G(z)))$$



We add notation $\log D(x)$ just so that we can combine the two:

$$\min[\log(D(x)) + \log(1 - D(G(z)))]$$

Combining the two, we have:

$$\min_G \max_D [\log(D(x)) + \log(1 - D(G(z)))]$$

For all samples:

$$\min_G \max_D \frac{1}{m} \sum_{i=1}^{m} [\log(D(x)) + \log(1 - D(G(z)))]$$

Issues

- Vanishing gradient
- Mode collapse (generator always produces same output)
- Nash equilibrium - both models achieve convergence concurrently
- Counting
- Perspective

Momentum Gradient Descent

$$\begin{aligned} \text{update}_0 &= 0 \\ \text{update}_t &= \gamma \text{update}_{t-1} + \eta \nabla w_t \\ w_{t+1} &= w_t - \text{update}_t \end{aligned}$$

Backdoor Attacks

- hidden patterns trained into DNN that produce unexpected behavior when activated by *triggers* but otherwise go undetected

Poisoning Attack

Attacker capabilities

- **Label manipulation:** can only manipulate the labels of the training dataset
- **Input manipulation:** can also manipulate the features of the training input

Attacker goals

- **Availability poisoning:** decrease accuracy on benign input
- **Integrity poisoning:** misclassification on certain inputs, eg: backdoor attacks

Scenarios

- **Untargeted**

$$\begin{aligned} \theta_{adv}(x) &= l_0 \\ \theta_{adv}(x_{adv}) &\neq l_0 \end{aligned}$$

- **Targeted**

$$\theta_{adv}(x) = l_0$$
$$\theta_{adv}(x_{adv}) = l_t$$

- **Outsourced Learning:**
  - Have control over training process
  - Inject adversarial data via data poisoning
  - Both targeted and untargeted settings
- **Transfer Learning:**
  - Download the model and finetune the parameters
  - Only untargeted setting

Measure

- **Accuracy on benign data:** count of

$$\theta_{adv}(x) = l_0$$

- **Accuracy on trojaned data:** count of

$$\theta_{adv}(x_{adv}) = l_0$$

- **Attack success rate on trojaned data:** count of

$$\theta_{adv}(x_{adv}) = l_t$$

Trojaning a Model

- Gradient descent on input
  - $O(gradient)$
- Generate trojan triggers
  - Induce high activation in some neurons in hidden layer
- Reverse engineer training data
  - Generate inputs that activate specific output neurons (for trojaned data)
  - Two sets of training data: benign (original) + trojaned (with triggers)
- Retrain
  - Re-train to strengthen link between neuron in hidden layer (activated by trigger) and the output neuron
  - Re-training the layers only after a selected neuron reduces the overall training time

Identifying and Mitigating Backdoor Attacks

Detection

- Whether the DNN is affected
- What is the target label?
- What is the trigger used?

Mitigation

- Detect and reject adversarial inputs
- Patch the DNN to remove backdoor

Neural Cleanse

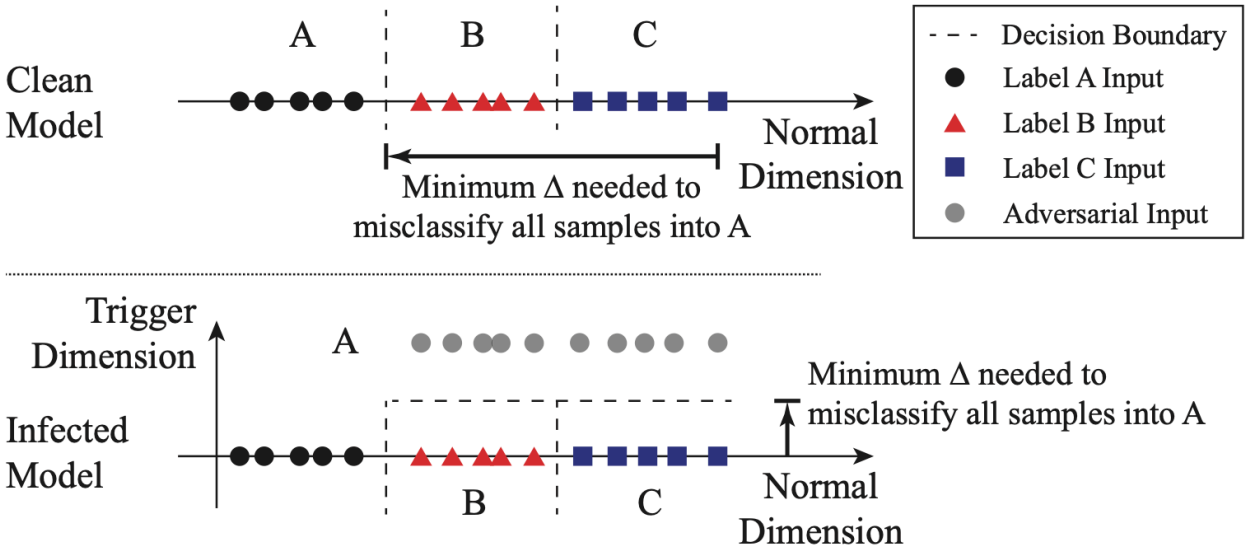


Fig. 2. A simplified illustration of our key intuition in detecting backdoor. Top figure shows a clean model, where more modification is needed to move samples of B and C across decision boundaries to be misclassified into label A. Bottom figure shows the infected model, where the backdoor changes decision boundaries and creates backdoor areas close to B and C. These backdoor areas reduce the amount of modification needed to misclassify samples of B and C into the target label A.

- Optimization to search for the minimal trigger that can classify any inputs to some label

$$min_{m,\Delta} l(y_t, f(A(x, m, \Delta))) + \lambda |m|$$

- $A(\cdot)$ represents the function that applies a trigger to the original image, x. Δ is the trigger pattern, which is a 3D matrix of pixel color intensities with the same dimension of the input image (height, width, and color channel). m is a 2D matrix called the mask, deciding how much the trigger can overwrite the original image.

- Reverse engineered trigger and injected trigger both fire similar neurons

- Limitations:
  - Varies based on the initialization (or random seed)
  - Requires large amount of data
  - May not detect large or feature space triggers

Artificial Brain Simulation (ABS)

- Trojan implies compromised neurons
- Such neurons which across the space of inputs (because, it should affect any benign input)
- Thus these neurons can be found by sampling benign inputs

- Steps:

  - Identify compromised neurons
  - Generate triggers of those neurons
  - Validate triggers

Neuron Simulation Function

- A function of model outputs wrt a neuron value (while keeping the values of other neurons fixed)
- Compromised neurons substantially elevate the activation of one particular label
- Select the neurons with maximum elevation difference

- **Attack success rate of reverse engineered trojan triggers:** percentage of benign inputs that can be subverted by the reverse engineered trigger

Adversarial Example Detection

Attribute-Steered Model

Attribute witness extraction

- **Attribute substitution** (step A) is applied on a base image by substituting an attribute with the counterpart in other images and then observing the neurons that have different values (step C). The images in the top row have different noses but the same face.
- **Attribute preservation** (step B) preserves an attribute by replacing the attribute in other images with that of the base and then observing the neurons that do not change (step D). Observe that the different faces in the bottom row have the same nose.
- The two sets are intersected to yield attribute witnesses (step E).

- Intersection of **attribute substitution** and **attribute preservation**.

- New model is created by:

  - **Neuron weakening:** weaken the non-witness neurons
  - **Neuron strengthening:** strengthen the witness neurons
- Detection (??)

  - what does false positive on benign input mean?
  - incorrect classification to a class, is it classification with trigger?

Neural network invariant checking

- Allow correct behaviors and forbit malicious behaviors (eg: assert certain *behaviors*)
- **Value invariants:** possible value distribution for each neuron
- **Provenance invariants:** possible delta between the values [*pattern*] of two layers of neurons
- Examples (??)

Adversarial Sample Defenses

Issues with Adversarial Training

- Requires small samples
- Slow
- Only effective for tiny pertubrations
- Only works 50% of the time

Gradient Masking

- Hide or destroy the gradient (so that all gradient based attacks fail)
- Defense techniques:
  - **Distillation defense:** changes the scale of last hidden layer
  - **Input preprocessing:** transforms input images by resizing, cropping, discretizing pixels
  - **Defense GAN:** uses GAN to transform perturbed images into clean images

Evading gradient masking

- Approximate gradients
- Hiding or breaking gradients makes the loss surface zig-zaggy, when doing backward pass replace function with difficult gradient with one that has nice gradient

Certified Adversarial Robustness

- Model gives prediction and a certificate that the prediction is constant (holds) within an $l2$ around the input
- Randomized smoothing leads to smoother decision boundaries
  - Smooth f into a new classifier g (the "smoothed classifier") as follows:

$$g(x) = \text{the most probable prediction by } f \text{ of random Gaussian corruptions of}$$

- Large random perturbations "drown out" small adversarial perturbations (??)

Confidentiality and Privacy Threats in ML

Model Inversion

- Extra sensitive inputs by leverage knowledge about model structure and some

information about an individual or object

Type 1

- For example, if one of the features is sensitive, attacker can judge the feature by setting it to 0 or 1 and checking the output label, y
- What if y does not change on changing feature?
- White box or black box? White - attacker has information about parameters or black - attacker changes one of those but cannot see other parameters.
- Formally:
  - infer x_n given f, $x_1, x_2, \ldots, x_{n-1}$ and y (??) where $x_n \in \{v_1, v_2, \ldots, v_s\}$
  - compute $y_j = f(x_1, x_2, \ldots, x_{n-1}, v_j)$ for each j
  - output v_j that maximizes:

$$dist(y, y_j) \times P(v_j | x_1, x_2, \ldots, x_{n-1})$$

  - what is y? (??)

Type 2

- Given f and y, infer X
- Use gradient descent to search for input X which maximizes probability of y
- White box or black box?

Model Extraction

- Learn a close approximation of the model f using as few queries to the model as possible
- For example, logistic regression function can be converted to a linear equation in $n + 1$ variables
- **Extraction attack:** learn model architecture or parameters
- **Oracle attack:** construct a substitute model

Membership Inference

- Given an input x and a black box access to the model f, determine if $x \in D$, meaning whether x was part of the training data (or distribution (??))
- Privacy concern?
  - If x is used for training a medical model, if one can determine $x \in D$, one can predict whether an individual have health issue or not

Attack stages

1. Development of shadow dataset
   - Goal: develop a dataset D' which closely emulates the original dataset D
   - Techniques: statistics-based, query-based, active learning, region-based
2. Generation of attack model training dataset
   - Takes input from shadown dataset D' as (x', y') and outputs a probability vector $p = (p_1, p_2, \ldots, p_k)$ and a binary label indicating "in" or "out"
   - Partition D' into $D_1, D_2, \ldots, D_s$
   - $\forall j$, train f_j to output "in" for D_j and "out" for $D \setminus D_j$
   - Obtain attack training data, $p = (p_1, p_2, \ldots, p_k)$ and label "in" or "out"
3. Training and depoloyment of membership inference attack model
   - Given input of probability vector, return "in" or "out"

Differential Privacy

- Guarantees:
  - Raw data will not be viewed
  - Output will have distortions

Confidence Interval

- Range of values for which we are fairly sure (say, x) the true value lies in

$$\overline{X} \pm Z\frac{s}{\sqrt{n}}$$

Standard DP

- Analysts sends a query to a software called *DP guard*
- Guard sends the query to the DB or model to retrieve the output
- Guard adds **noise** to the output (in order to protect the confidentiality of the individual whose data was accessed from DB) and sends back the response to the analyst

Local DP

- User anonymizes the data themselves and send to the aggregator
- Aggregator doesn't have access to the real data

Advantages / disadvantages

- Local DP less prone to data leak as the aggregator does not have access to real data
- Might be less accurate (?)

Formalism

- Whether or not more data is adding into D, both the results with or without will be the same, R. $A = 1$ is ideal in which case both the results are identical, whereas if A is much larger or smaller, the result deviates too much.

$$\frac{P(Q(D_I)) = R}{P(Q(D_{I \pm i})) = R} \leq A$$

- Putting, $A = e^{\epsilon}$:

$$\frac{P(R|D_I}{P(R|D_{I \pm i}) = R} \leq e^{\epsilon}$$

Global sensitivity

- Maximum amount an answer can change by adding or removing an individual from the dataset.
- $F(D) = X$ is a deterministic, non-privatized function over dataset D which returns X, a vector k real numbers.
- Global sensitivity is the sum of the worst case differences between datasets $D1$ and $D2$ differing by at most one element, ΔF:

$$\Delta F = \max_{D1, D2} ||F(D1) - F(D2)||_{L1}$$

Noise adding mechanism

- Privatizing by adding noise from Laplace distribution:

$$P(R = x|D \text{ is true world}) = \frac{\epsilon}{2\Delta F} \exp - \frac{|x - F(D)| \, \epsilon}{\Delta F}$$

- Laplace ($\epsilon-$ differential)

$$F(x) = f(x) + Lap\left(\frac{s}{\epsilon}\right)$$

- Exponential ($\epsilon-$ differential)
  - Works with both numeric and categorical data
  - Releases the identity of the element with MAX noisy score and not that of the score

itself (??)
- Gaussian (ϵ, δ— differential)

Video

- A Practical Beginners' Guide to Differential Privacy by Christine Task, Purdue CERIAS
- An attacker can still learn information about an individual even if he doesn't know whether the person submitted the survey or not.
- An attacker might be able to learn with high probability whether an individual took a survey (even if we know the result with or without the individual would be nearly the same) because individual can be part of a group.

Federated Learning

- Enables mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device
- Decouples the ability to do machine learning from the need to store the data in the cloud

How?

- Your device downloads the current model, improves it by learning from data on your phone
- Summarizes the changes as a small focused update.
- Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model.

$$\min_{w \in \mathbb{R}^d} f(w), \text{where:}$$

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

Federated Averaging Algorithm

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

initialize w_0

for each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of m clients)

for each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)

for each local epoch i from 1 to E **do**

for batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return w to server

Benefits

- Smarter models (improved, updated model can be used instantly)
- Lower latency
- Less power consumption
- **PRIVACY**

Challenges

- Network communication is expensive
- Heterogenous nature of the client devices
- Needs fault tolerance
- Data distribution is non-IID (same words might be used in different context by different users)
- Privacy concerns, sensitive information can be extracted from shared model updates. Can use DP.