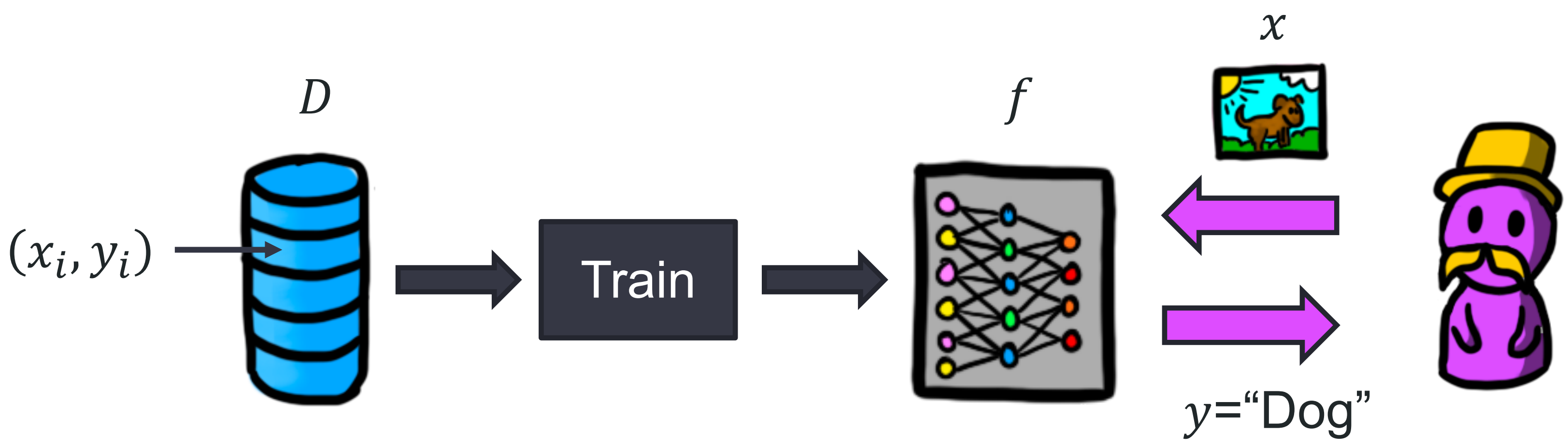
CS489/689 Privacy, Cryptography, Network and Data Security

Privacy-Preserving Machine Learning

Machine learning: quick primer

- For simplicity, we will focus on a *classification problem* with *supervised learning*.
    - Unsupervised or Reinforcement learning are other types
- We have a training set $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ with n samples. Given a sample (x_i, y_i), x_i are the *features* and y_i is its *label*.
- We want to produce a function $f: \mathcal{X} \rightarrow \mathcal{Y}$ that can predict a sample's label from its features.
- We will use the training set to train such a function. Ideally, it should correctly predict labels for unseen samples (e.g., samples in a testing set).
    - We will say that a model *generalizes* well if it has high accuracy on unseen samples
    - A model *overfits* if it works perfectly for samples in the training set but does not generalize well.

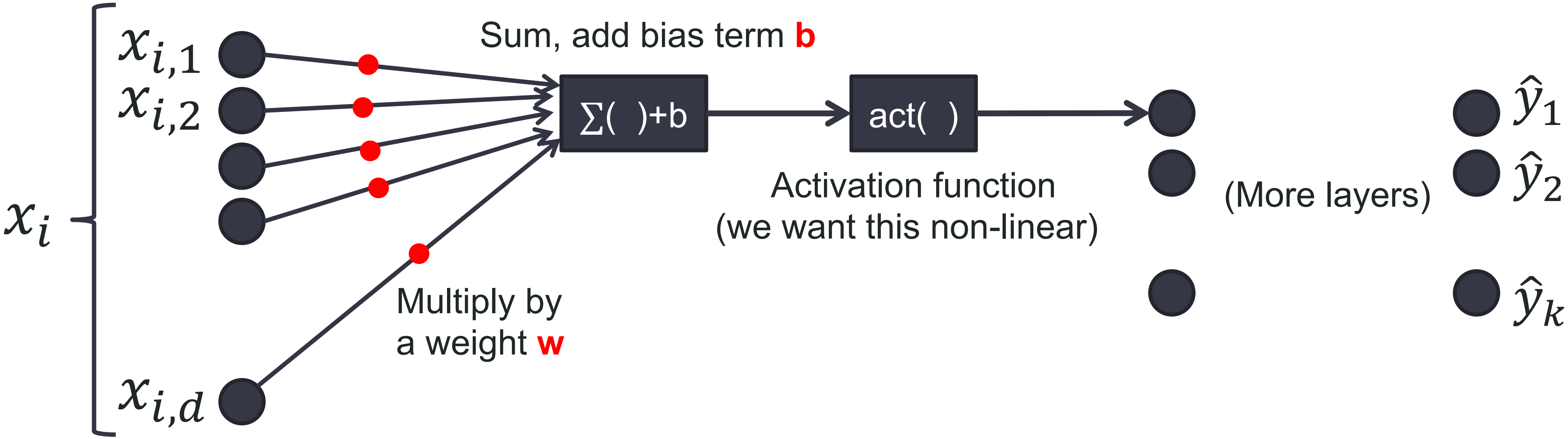
Machine learning: quick primer



Usually, this gives confidence scores for each class: $(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_k)$
For example: ["Dog", "Cat", "Mouse" …]=[0.81, 0.10, 0.03, …]

Neural networks

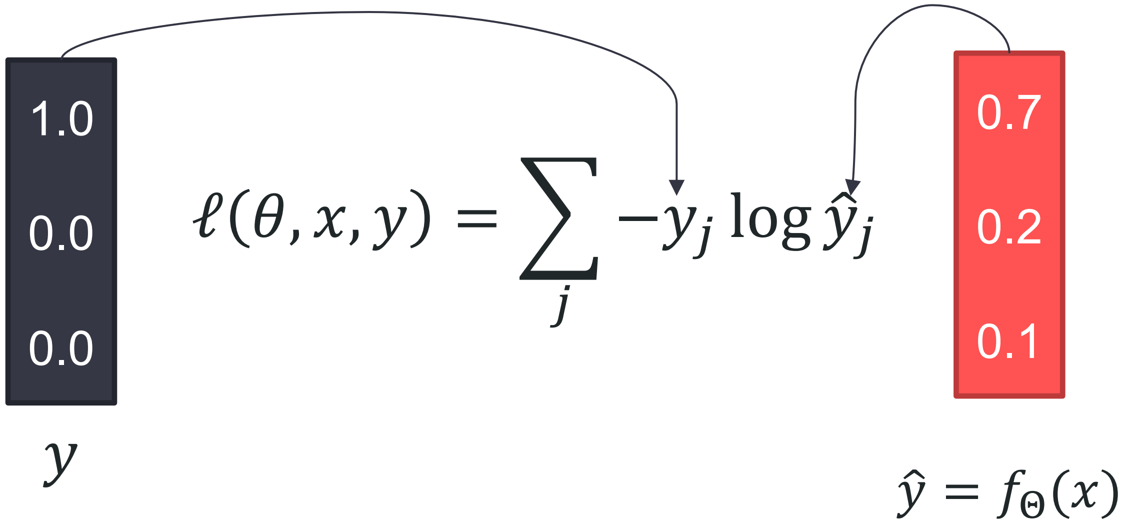
- There are many architectures for machine learning models (i.e., many structures for the function f).
- One of the most popular are **neural networks**.



Training the model means tuning all **w**'s and **b**'s

Loss Functions

- We define a *loss function* that we want to minimize: $\ell(\theta, x, y)$, where θ are the parameters w and b.
  - For example, a typical loss function is $\ell(\theta, x, y) = \sum_j -y_j \log \hat{y}_j$ where y_j is only 1 for the true label of the sample, j.

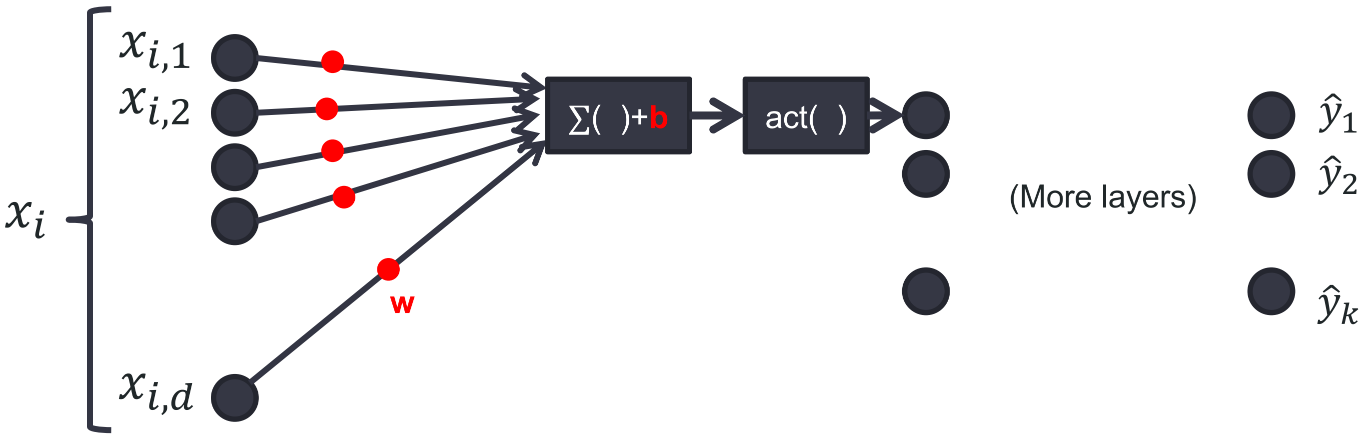


Training neural networks

- Since we have the training set D, it makes sense to minimize the empirical loss in this training set:

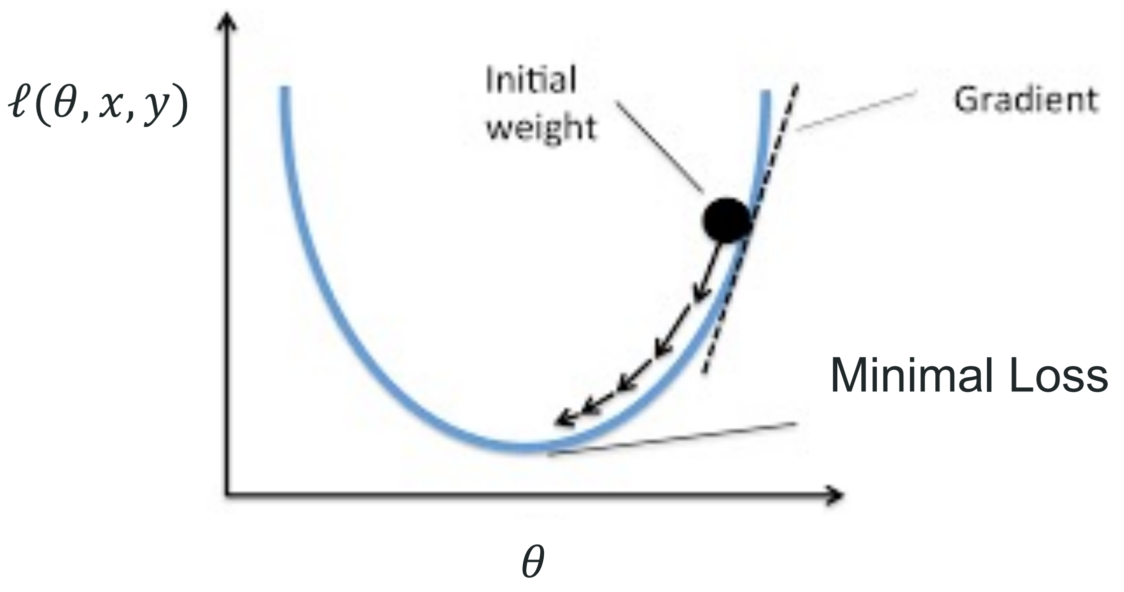
$$\mathcal{L}(\theta, D) = \frac{1}{N}\sum_i \ell(\theta, x_i, y_i)$$

- In practice, the minimization is done using Stochastic Gradient Descent (SGD).



Gradient Descent

- The gradient of the loss $\nabla \ell(\theta, x, y)$ evaluated at (x, y) is the derivative with respect to each parameter θ_i (every w and b).
- It tells us the direction in which θ should go to minimize the loss (for sample (x, y)).



Gradient Descent

- We could minimize the loss by running several steps (epochs) of Gradient Descent:
  - For each step $t \in [T]$:

$$\theta_t = \theta_{t-1} - \eta \nabla \mathcal{L}(\theta_{t-1}, D)$$

  - η is the *learning rate*

- This is expensive, so usually we do these iterations over a subset of the training sets (batches)
- Note θ represents parameters, η and T are hyper-parameters

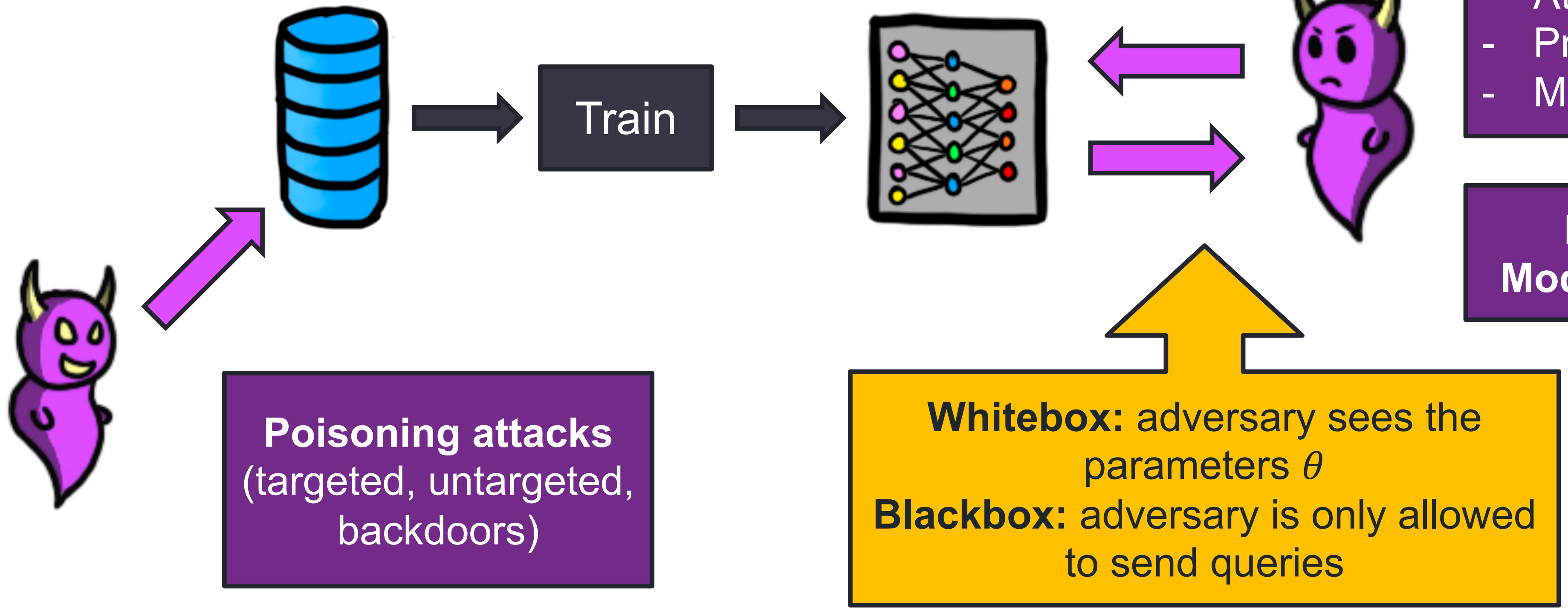
Stochastic Gradient Descent – with Mini Batches

For each training step $t \in [T]$:

1. Take a batch B of L samples from D
2. For each $(x_i, y_i) \in B$, compute the gradient $\mathrm{g_i} = \nabla \ell(\theta_{t-1}, x_i, y_i)$
3. Average the gradients $g = \frac{1}{L} \sum_i g_i$
4. Descend $\theta_t = \theta_{t-1} - \eta \cdot g$

Attacking ML models

- There are many types of attacks against ML
- Later we will see that there are also different types of defenses



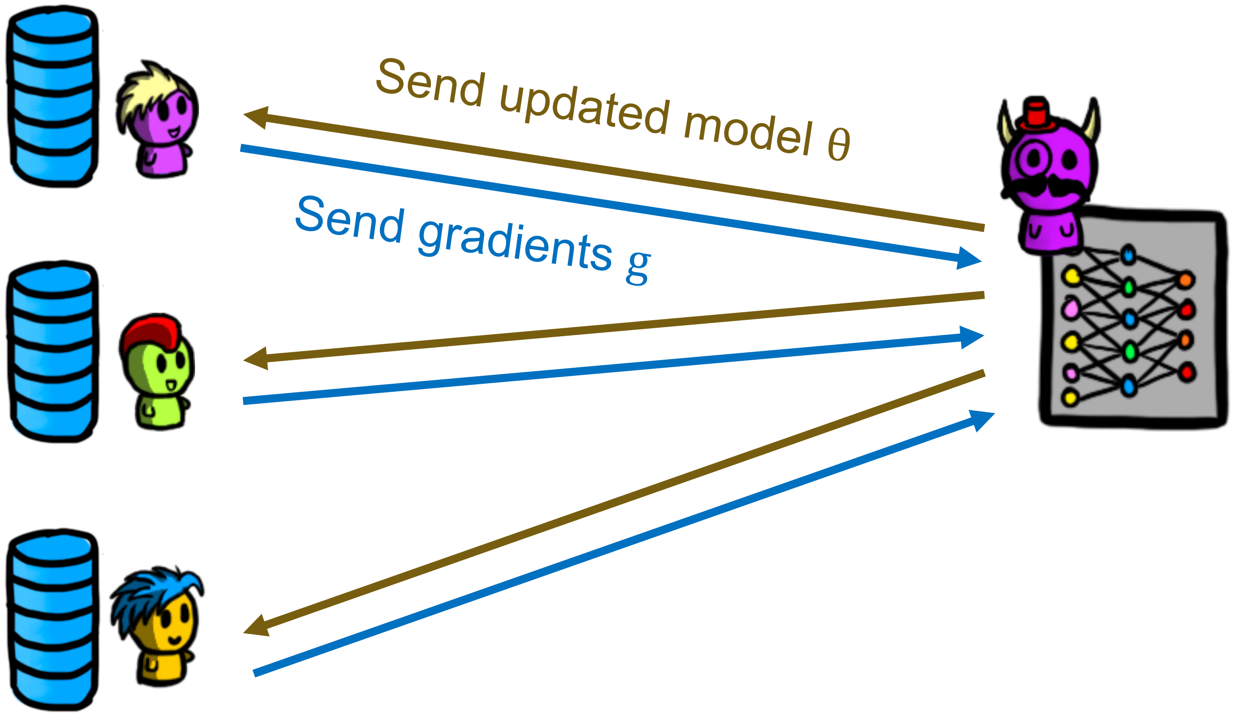
Inference Attacks:
- Membership inference
- Attribute inference
- Property inference
- Model inversion

Evasion attacks
Model stealing attacks

Attacking ML models in Federated Learning

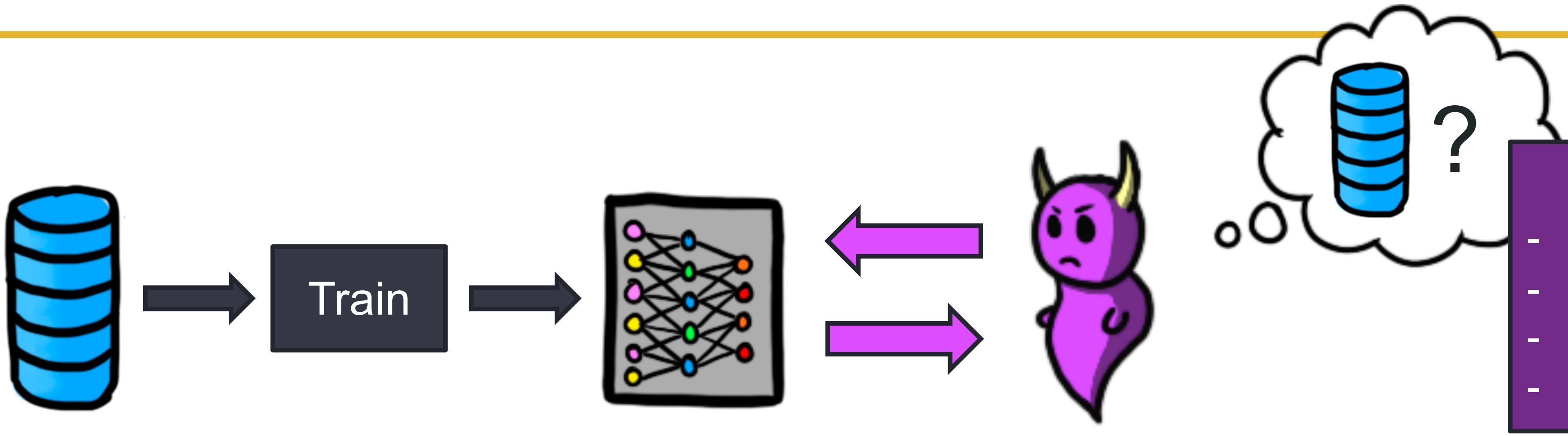
- Federated Learning: a centralized server builds a model, a set of clients send updates (gradients) using their local datasets

Poisoning attacks
(targeted, untargeted, backdoors)



Inference Attacks:
(adv sees all intermediate gradients, can potentially send malicious θ)
- Membership inference
- Attribute inference
- Property inference
- …

In this lecture: inference attacks



Inference Attacks:
- Membership inference
- Attribute inference
- Property inference
- Model inversion

Membership Inference: Is a given sample in the training set?

Attribute Inference: Given a sample with some missing attributes, can we guess them?

Property Inference: Given a property about the *whole* training set, can we guess if it's true or not?

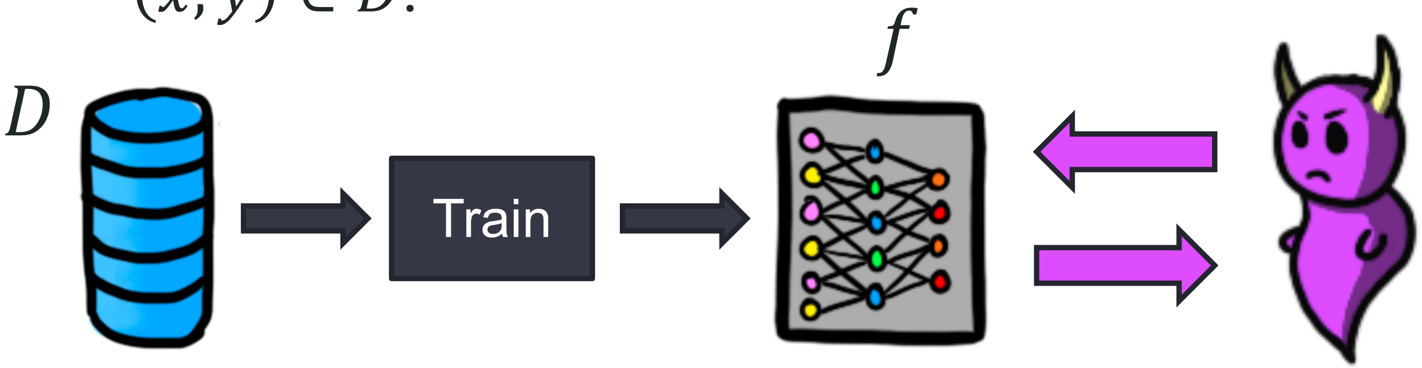
Model inversion: Given a label, can we find a representative element of this class? (learn x from y)

Q: Why are these attacks a threat to privacy?

Inference Attacks in ML

Membership Inference Attacks (MIAs)

- Given a sample (x, y), and a model f trained with dataset D, guess whether $(x, y) \in D$.



Black-box: the adversary queries the model (possibly more than once)
White-box: the adversary sees the model parameters θ

- With only black-box access, and a model that outputs confidence scores:
- $f(x) = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k]$, where $\hat{y}_j$ are confidence scores for label j.

Q: If you were the adversary, with a target sample (x, y) and black-box access to the model f, how would you guess if the target sample is a member?

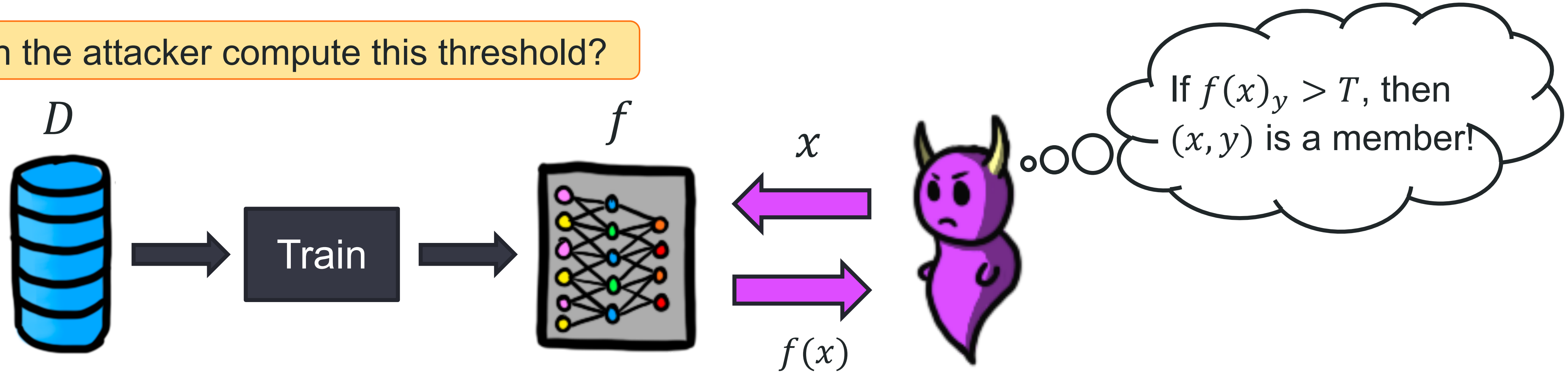
Threshold Attacks

- *Idea:* the model will be more confident on samples it has seen during training.

Threshold attack

- This attack queries the model on sample x and then measures the confidence score assigned to its true label y.
- If the confidence score is above some *threshold*, then the attack decides the sample is a *member*.

Q: how can the attacker compute this threshold?



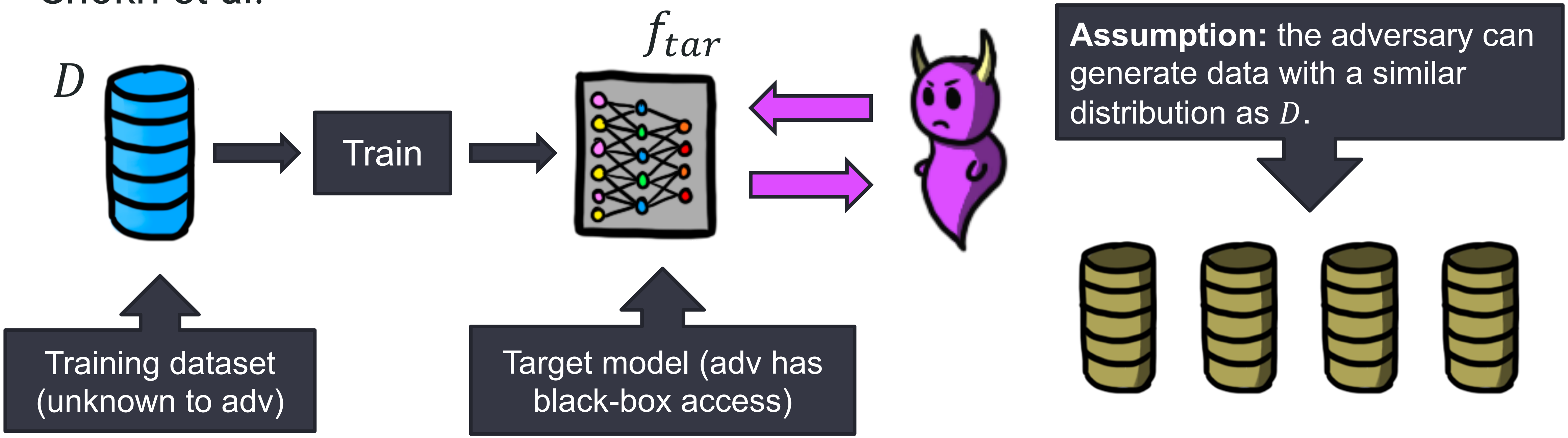
Yeom et al. "Privacy risk in machine learning: Analyzing the connection to overfitting." *CSF*, 2018.

Neural Network-based Attacks

- Other MIAs use Machine Learning against Machine Learning.

Neural Network-based Attacks

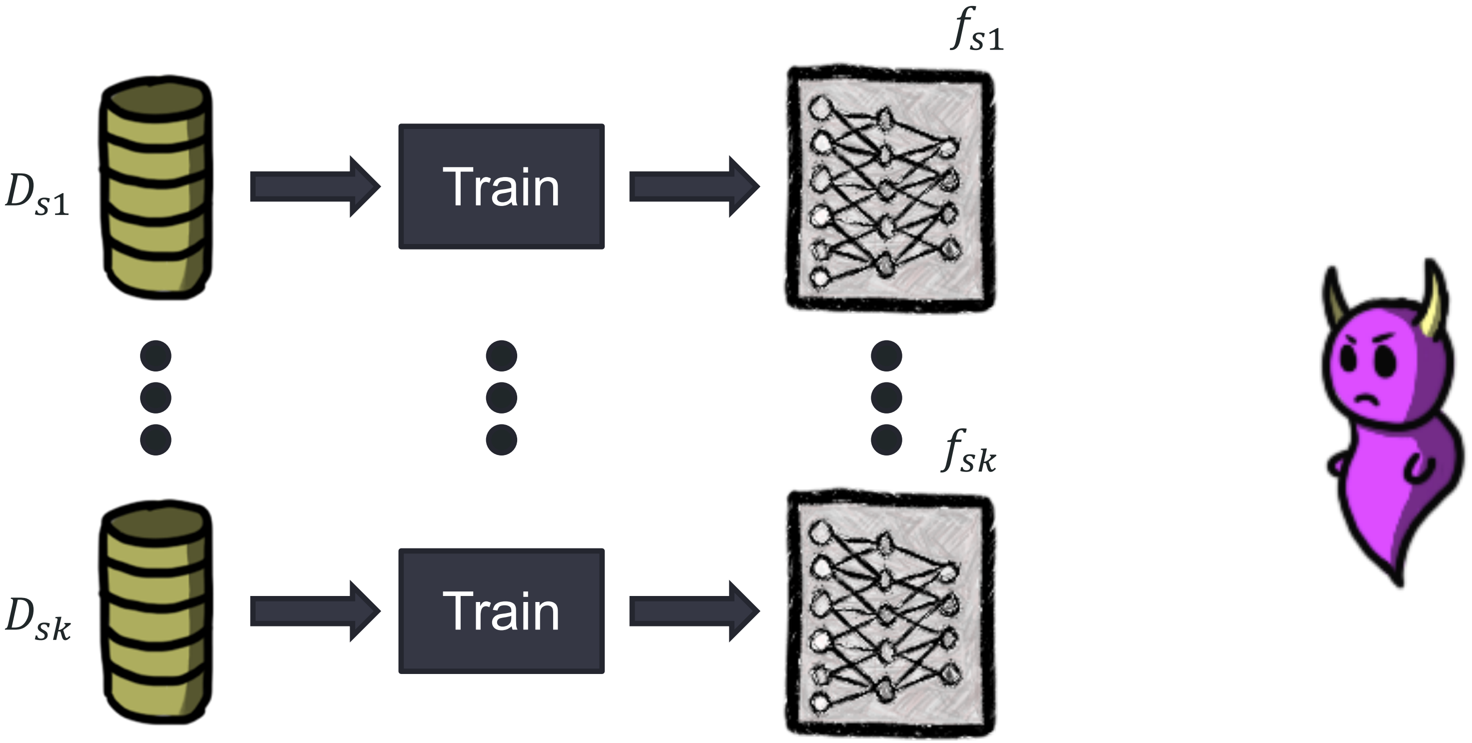
- Other MIAs use Machine Learning against Machine Learning.
- The first NN-based attack (which was also the first MIA) was proposed by Shokri et al.



Shokri, Reza, et al. "Membership inference attacks against machine learning models." *IEEE symposium on security and privacy (SP)*, 2017.

Shokri et al.'s attack

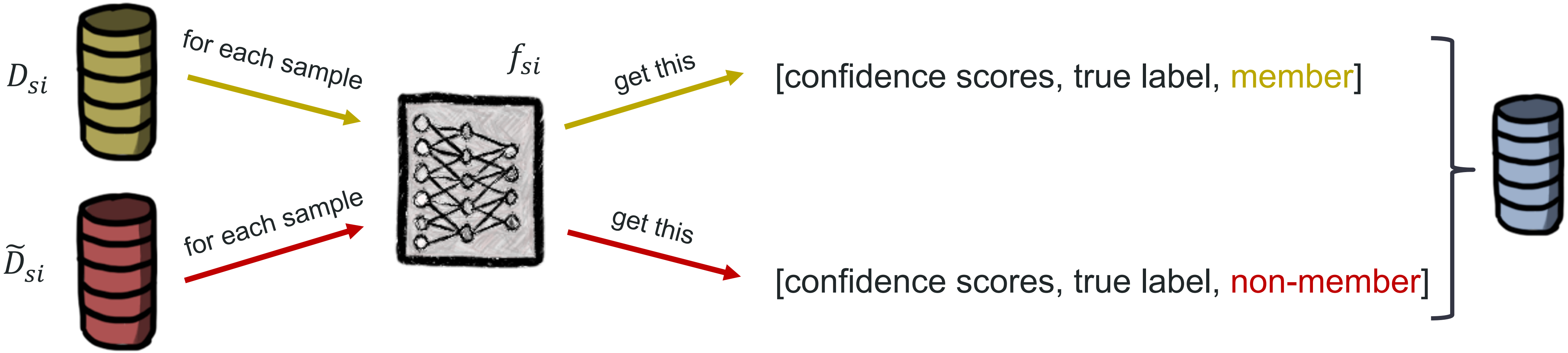
1. Generate **shadow** training data $D_{s1}, D_{s2}, \ldots, D_{sk}$ (distribution similar to D).
2. Train k **shadow models** $f_{s1}, \ldots, f_{sk}$ (same classification task as the target model).



Shokri, Reza, et al. "Membership inference attacks against machine learning models." *IEEE symposium on security and privacy (SP)*, 2017.

Shokri et al.'s attack

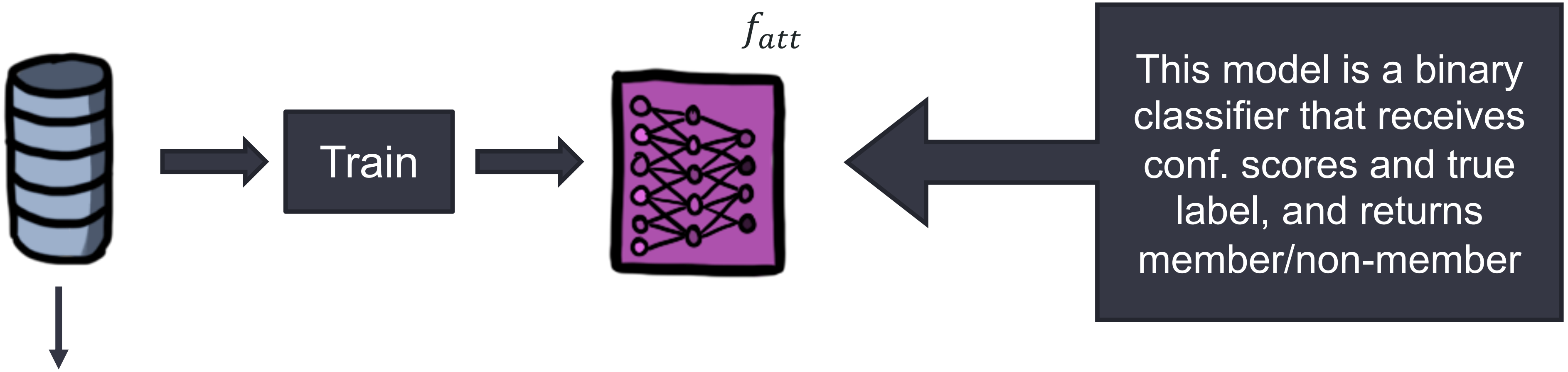
3. Generate shadow test data $\widetilde{D}_{s1}, \widetilde{D}_{s2}, \ldots, \widetilde{D}_{sk}$.
4. For each shadow model $i \in [k]$: get the confidence scores for each sample in D_{si} and $\widetilde{D}_{si}$. Create a dataset with (confidence scores, true label, membership) for each sample.



Shokri, Reza, et al. "Membership inference attacks against machine learning models." *IEEE symposium on security and privacy (SP)*, 2017.

Shokri et al.'s attack

5. With the new dataset, that contains [confidence scores, true label, membership status] computed with all the shadow models, train a new **attack model** f_{att} to predict the "membership status" from "confidence scores, true label"

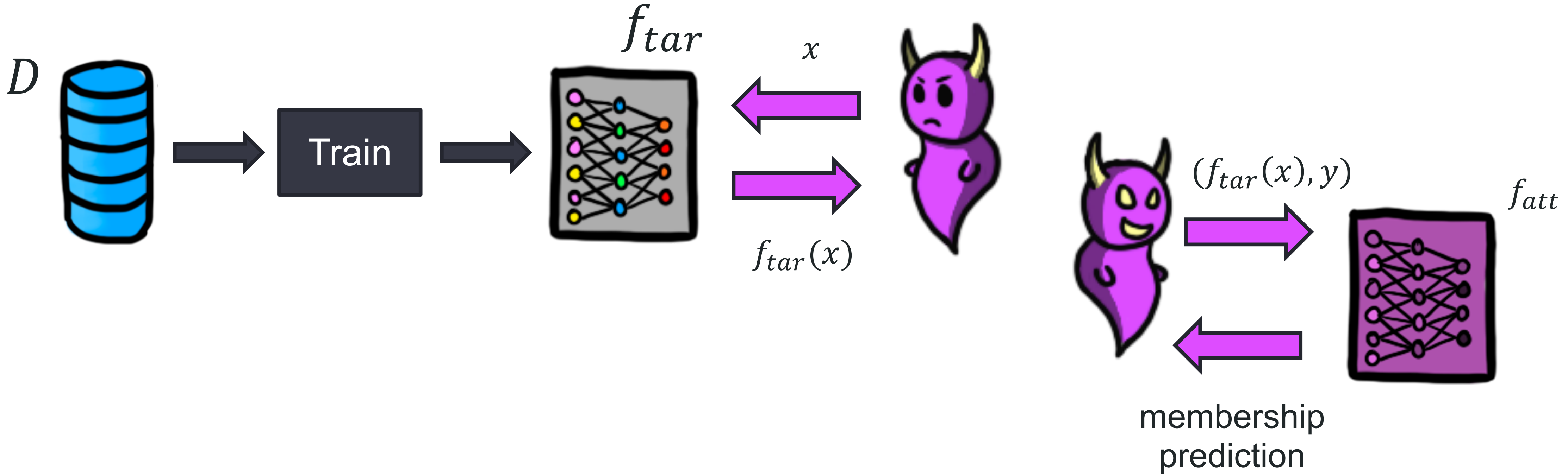


x=[confidence scores, true label], y=[member/non-member]

Shokri, Reza, et al. "Membership inference attacks against machine learning models." *IEEE symposium on security and privacy (SP)*, 2017.

Shokri et al.'s attack

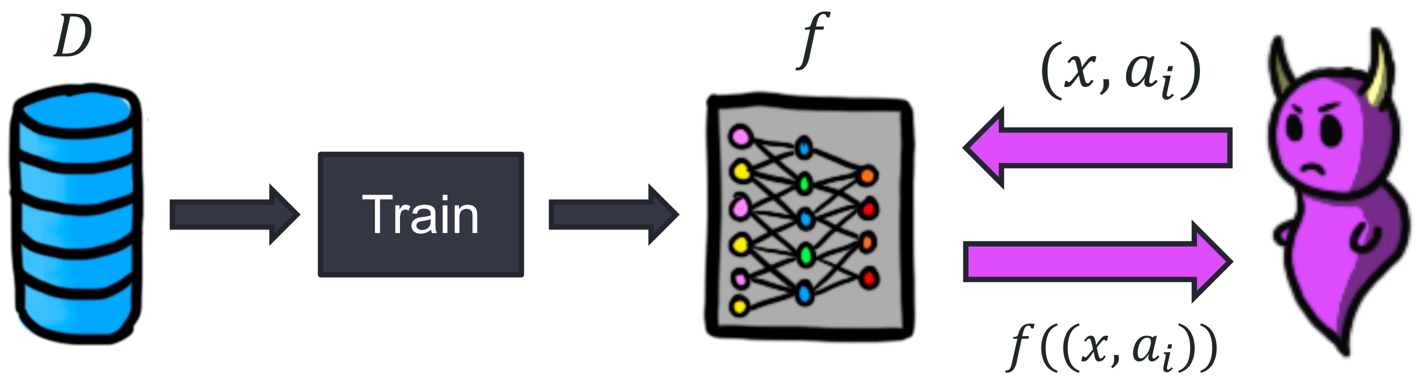
6. Get the confidence scores of the target sample in the target model f_{tar}.
7. Evaluate those [confidence scores, true label] in the attack model f_{att}.



Shokri, Reza, et al. "Membership inference attacks against machine learning models." *IEEE symposium on security and privacy (SP)*, 2017.

Attribute Inference Attacks

- Each sample is $z = (x, a, y)$, where x is the features, a is a privacy-sensitive attribute, and y is the label.
- The adversary has a sample $z = (x, ?, y)$, and wants to learn the attribute.
- Assume the space of all attributes is $\mathcal{A} = \{a_1, a_2, \dots, a_m\}$
- Simple attack: query for all possible samples (x, a_1), …,(x, a_m). The true attribute is probably the one that yields a highest confidence score for the true class y.

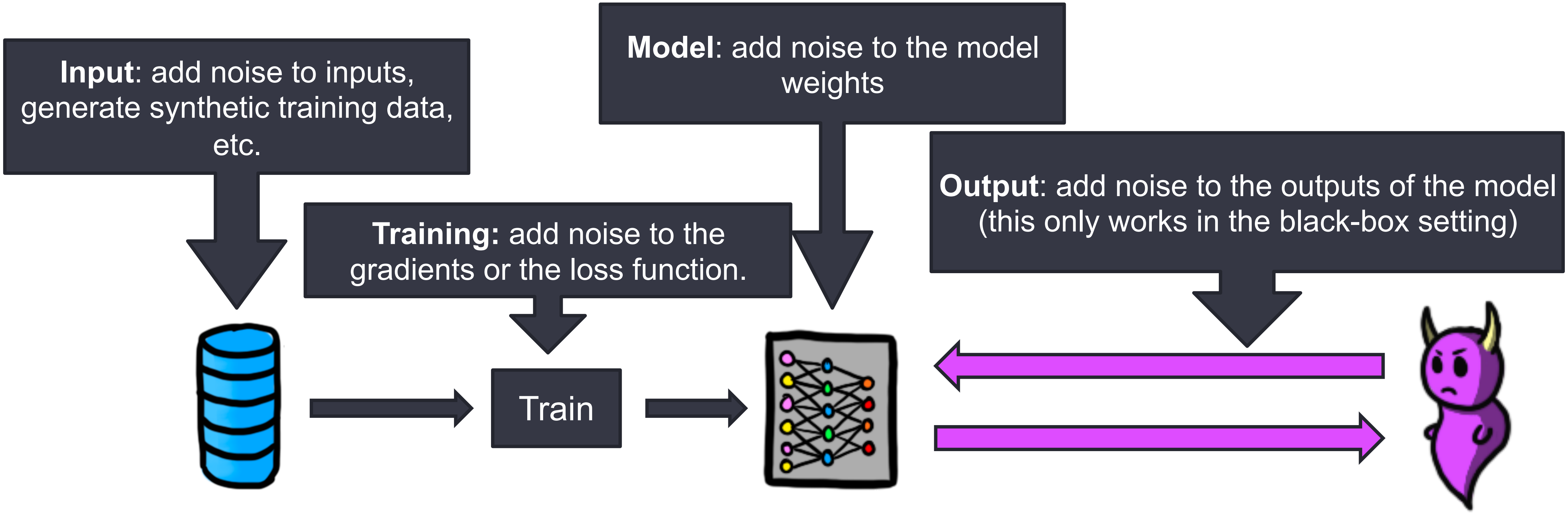


$$\hat{\imath} = argmax_i f\big((x, a_i)\big)_y$$

Defenses against inference attacks

Defending against inference attacks

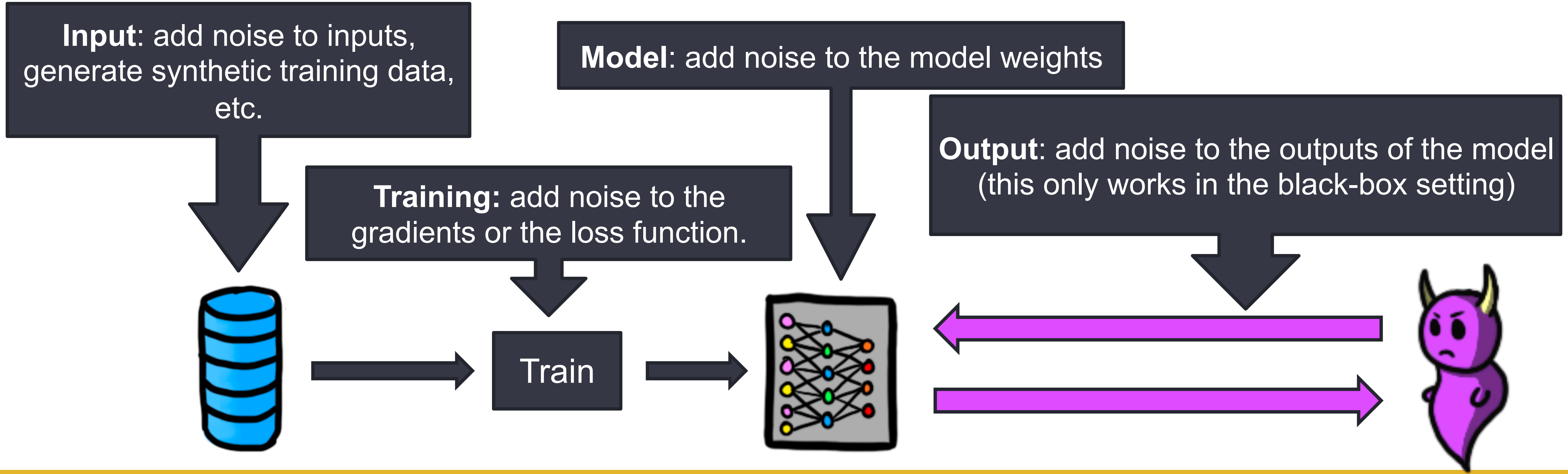
- Where do we defend?



Defending against inference attacks

Act.

- Which technique do you think is best? Give rational using the pros of your technique vs the cons of others.
  - A couple of sentences submitted to learn



Differentially Private Stochastic Gradient Descent (DP-SGD)

- Adds privacy during the training step, modifying SGD.
- Recall Differential Privacy: we want to limit the effect that a single training set sample has on the output (the "output" of the training algorithm is the model!)

SGD

For each training step $t \in [T]$:
1. Take a batch B of L samples from D.
2. For each $(x_i, y_i) \in B$, compute the gradient:
   $$g_i = \nabla \ell(\theta_{t-1}, x_i, y_i)$$
3. Average the gradients $g = \frac{1}{L}\sum_i g_i$.
4. Descend $\theta_t = \theta_{t-1} - \eta \cdot g$.

"Private" SGD

For each training step $t \in [T]$:
1. Take a batch B of L samples from D.
2. For each $(x_i, y_i) \in B$, compute the gradient:
   $$g_i = \nabla \ell(\theta_{t-1}, x_i, y_i)$$
3. Average the gradients and add noise $g = \frac{1}{L}(\sum_i g_i + \mathcal{N}(0, \sigma^2))$.
4. Descend $\theta_t = \theta_{t-1} - \eta \cdot g$.

Q: Is it enough to add noise to the gradients?

Differentially Private Stochastic Gradient Descent (DP-SGD)

- The gradient could potentially be unbounded → unbounded sensitivity → bad for DP
- We *clip* the gradients to ensure their ℓ_2 norm is at most C.
  - C is the *clipping threshold*
  - C is independent of the data

SGD

For each training step $t \in [T]$:
1. Take a batch B of L samples from D.
2. For each $(x_i, y_i) \in B$, compute the gradient:
$$\mathrm{g_i} = \nabla \ell(\theta_{t-1}, x_i, y_i)$$
3. Average the gradients $g = \frac{1}{L} \sum_i g_i$.
4. Descend $\theta_t = \theta_{t-1} - \eta \cdot g$.

DP-SGD

For each training step $t \in [T]$:
1. Take a batch B of L samples from D.
2. For each $(x_i, y_i) \in B$, compute the gradient:
$$\mathrm{g_i} = \nabla \ell(\theta_{t-1}, x_i, y_i)$$
3. Clip the gradients: $g_i = g_i / \max\left(1, \frac{\|g_i\|_2}{C}\right)$
4. Sum the gradients $g = \sum_i g_i$.
5. Add noise: $g = g + \mathcal{N}(0, \sigma^2 C^2)$
6. Descend $\theta_t = \theta_{t-1} - \eta \cdot \frac{1}{L} g$.

DP-SGD: keeping track of ϵ, δ

- Note that a single sample will participate in multiple training steps → there will be some sequential composition involved.
- We need to keep track of ϵ, δ. For a *fixed amount of noise* σ, if we do not use advance techniques to keep track of ϵ, δ, we will end up with a very large ϵ, which is bad.
  - Note that the actual *true* ϵ will be smaller than the ϵ we can compute theoretically.
  - But we can only guarantee an ϵ we can theoretically prove.

DP-SGD

For each training step $t \in [T]$:

1. Take a batch B of L samples from D.
2. For each $(x_i, y_i) \in B$, compute the gradient:
   $$g_i = \nabla \ell(\theta_{t-1}, x_i, y_i)$$
3. Clip the gradients: $g_i = g_i / \max\left(1, \frac{\|g_i\|_2}{C}\right)$
4. Sum the gradients $g = \sum_i g_i$.
5. Add noise: $g = g + \mathcal{N}(0, \sigma^2 C^2)$
6. Descend $\theta_t = \theta_{t-1} - \eta \cdot \frac{1}{L} g$.

DP-SGD: keeping track of ϵ, δ

- First, we choose a δ. Recall that this should be smaller than $\delta < \frac{1}{N}$.
  - The reason is the following: a training algorithm that simply publishes a random training set record would provide $(\epsilon = 0, \delta = 1/N)$-DP. However, we know this is not private enough.

DP-SGD

For each training step $t \in [T]$:

1. Take a batch B of L samples from D.
2. For each $(x_i, y_i) \in B$, compute the gradient:
$$g_i = \nabla \ell(\theta_{t-1}, x_i, y_i)$$
3. Clip the gradients: $g_i = g_i / \max\left(1, \frac{||g_i||_2}{C}\right)$
4. Sum the gradients $g = \sum_i g_i$.
5. Add noise: $g = g + \mathcal{N}(0, \sigma^2 C^2)$
6. Descend $\theta_t = \theta_{t-1} - \eta \cdot \frac{1}{L} g$.

DP-SGD: keeping track of ϵ, δ

Q: Given δ, σ, C, T, and assuming each sample in D is used *once per training step*, what is the total ϵ we get?

- Use naive composition

DP-SGD

For each training step $t \in [T]$:

1. Take a batch B of L samples from D.
2. For each $(x_i, y_i) \in B$, compute the gradient:
$$\mathrm{g_i} = \nabla \ell(\theta_{t-1}, x_i, y_i)$$
3. Clip the gradients: $g_i = g_i / \max\left(1, \frac{\|g_i\|_2}{C}\right)$
4. Sum the gradients $g = \sum_i g_i$.
5. Add noise: $g = g + \mathcal{N}(0, \sigma^2 C^2)$
6. Descend $\theta_t = \theta_{t-1} - \eta \cdot \frac{1}{L} g$.

$f(D) + Y$ is (ϵ, δ)-DP if

$$Y \sim N(0, \sigma^2)$$

$$\sigma^2 = 2 \ln\left(\frac{1.25}{\delta}\right) \Delta_2^2 / \epsilon^2$$

DP-SGD: keeping track of ϵ, δ

Q: Given δ, σ, C, T, and assuming each sample in D is used *once per training step,* what is the total ϵ we get?

- Use naive composition

A: $C^2\sigma^2 = 2\ln\left(\frac{1.25}{\delta}\right)\Delta_2^2/\epsilon^2 \quad \rightarrow \epsilon = \sqrt{2\ln\left(\frac{1.25}{\delta}\right)}/\sigma$ for each step. Then naïve composition gives

$$\epsilon = T\sqrt{2\ln\left(\frac{1.25}{\delta}\right)}/\sigma$$

*Note: this question is very over simplified

DP-SGD

For each training step $t \in [T]$:

1. Take a batch B of L samples from D.
2. For each $(x_i, y_i) \in B$, compute the gradient:
$$g_i = \nabla\ell(\theta_{t-1}, x_i, y_i)$$
3. Clip the gradients: $g_i = g_i / \max\left(1, \frac{\|g_i\|_2}{C}\right)$
4. Sum the gradients $g = \sum_i g_i$.
5. Add noise: $g = g + \mathcal{N}(0, \sigma^2 C^2)$
6. Descend $\theta_t = \theta_{t-1} - \eta \cdot \frac{1}{L} g$.

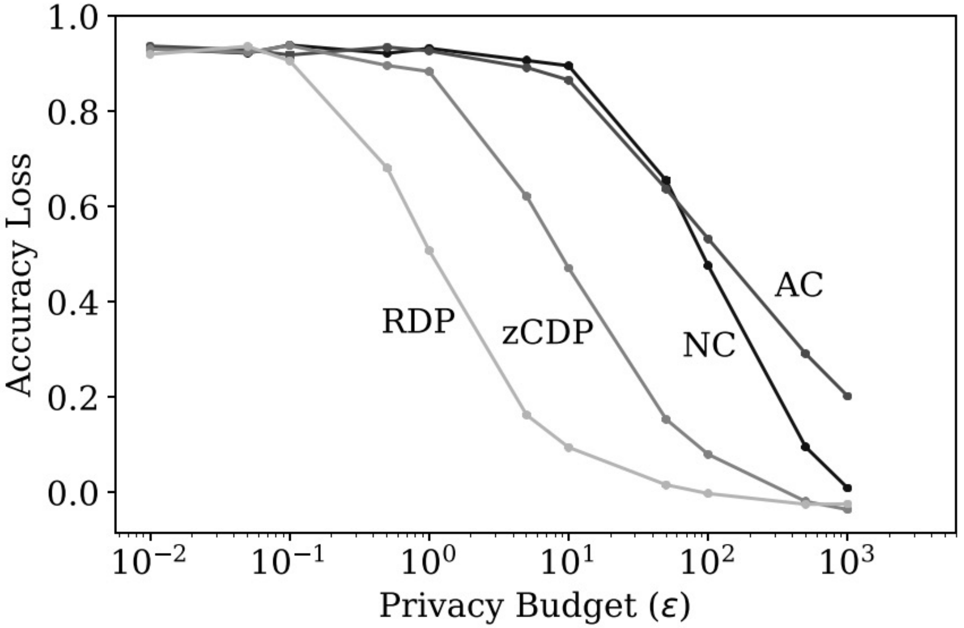
$f(D) + Y$ is (ϵ, δ)-DP if

$$Y \sim N(0, \sigma^2)$$

$$\sigma^2 = 2\ln\left(\frac{1.25}{\delta}\right)\Delta_2^2/\epsilon^2$$

DP-SGD: keeping track of ϵ, δ

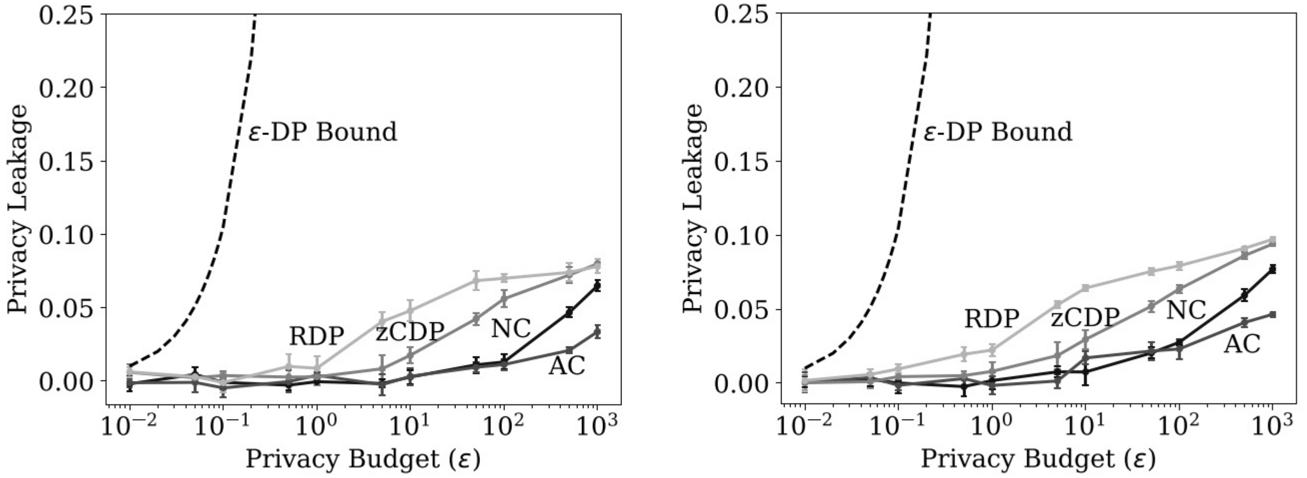
- Renyi Differential Privacy (RDP) provides a tighter ϵ, δ bound.
  - Better suited to Gaussian Noise
  - Keeps track of more information, only compress at the end
- This means that, for a given σ, C, and δ, RDP tells us our actual ϵ is smaller than what Advanced Composition (AC) tells us.
- In other words, for a target privacy budget ϵ, using RDP we need to add less noise than using AC.
- Note that, even with RDP, we need $\epsilon > 100$ if we do not want any accuracy loss



Jayaraman, Bargav, and David Evans. "Evaluating differentially private machine learning in practice." *USENIX Security Symposium*. 2019.

DP-SGD: theoretical vs empirical privacy

- Both attacks we've seen perform similarly
- It seems that $\epsilon = 100$ or even $\epsilon = 1000$ still provides good empirical privacy
- The theoretical bound on the privacy leakage provided by DP is very loose



(a) Shokri et al. membership inference

(b) Yeom et al. membership inference

Jayaraman, Bargav, and David Evans. "Evaluating differentially private machine learning in practice." *USENIX Security Symposium*. 2019.

Issues of DP-SGD

- We saw that, for strong theoretical privacy (e.g., $\epsilon < 1$), the models usually lose all utility.
- For very weak theoretical privacy (e.g., $\epsilon = 100$), some models achieve reasonable utility.
- However, DP-SGD with $\epsilon = 100$ seems to provide enough protection against existing attacks.

Q: Is it OK to use $\epsilon = 100$?

Issues of DP-SGD

- We saw that, for strong theoretical privacy (e.g., $\epsilon < 1$), the models usually lose all utility.
- For very weak theoretical privacy (e.g., $\epsilon = 100$), some models achieve reasonable utility.
- However, DP-SGD with $\epsilon = 100$ seems to provide enough protection against existing attacks.

Q: Is it OK to use $\epsilon = 100$?

A: It might be OK to use DP-SGD tuned to $\epsilon = 100$, but at that point we might as well use defenses that do not provide DP, since the DP guarantee is already meaningless at that point.

Private Aggregation of Teacher Ensembles (PATE)

1. Train teacher models with disjoint subsets of the training data
2. Use the teachers to label some (incomplete) public data
3. Use the labeled public data to train a student model

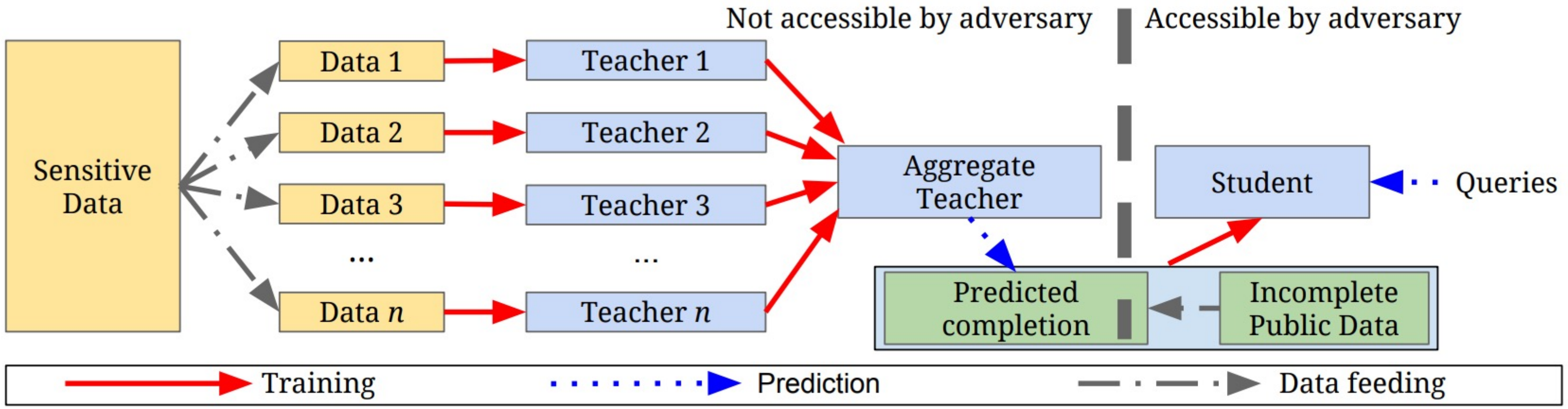


Figure 1: Overview of the approach: (1) an ensemble of teachers is trained on disjoint subsets of the sensitive data, (2) a student model is trained on public data labeled using the ensemble.

Papernot, Nicolas, et al. "Semi-supervised knowledge transfer for deep learning from private training data." *ICLR 2017*

Private Aggregation of Teacher Ensembles (PATE)

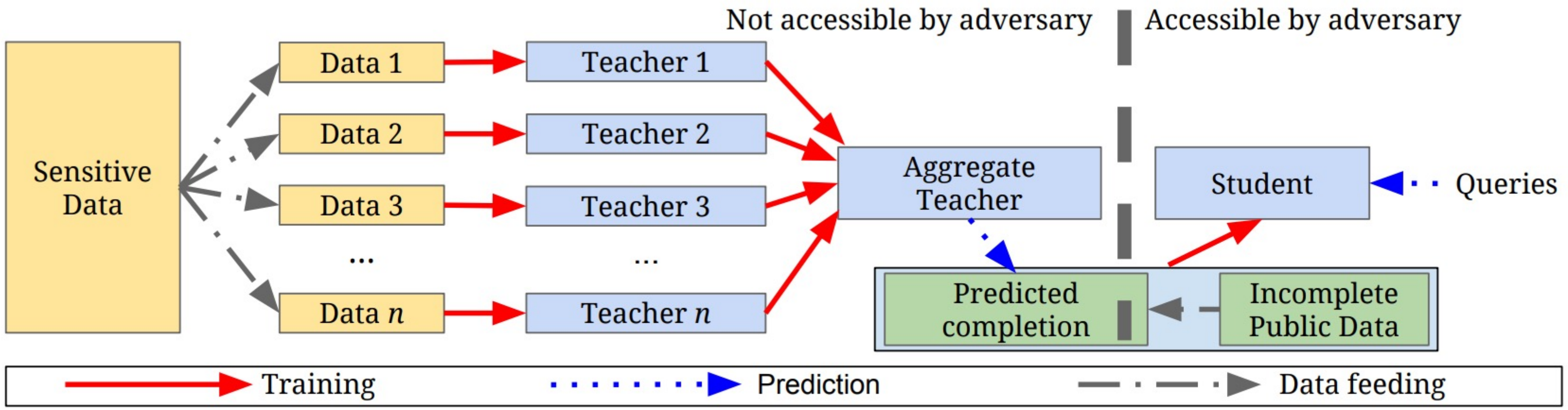


Figure 1: Overview of the approach: (1) an ensemble of teachers is trained on disjoint subsets of the sensitive data, (2) a student model is trained on public data labeled using the ensemble.

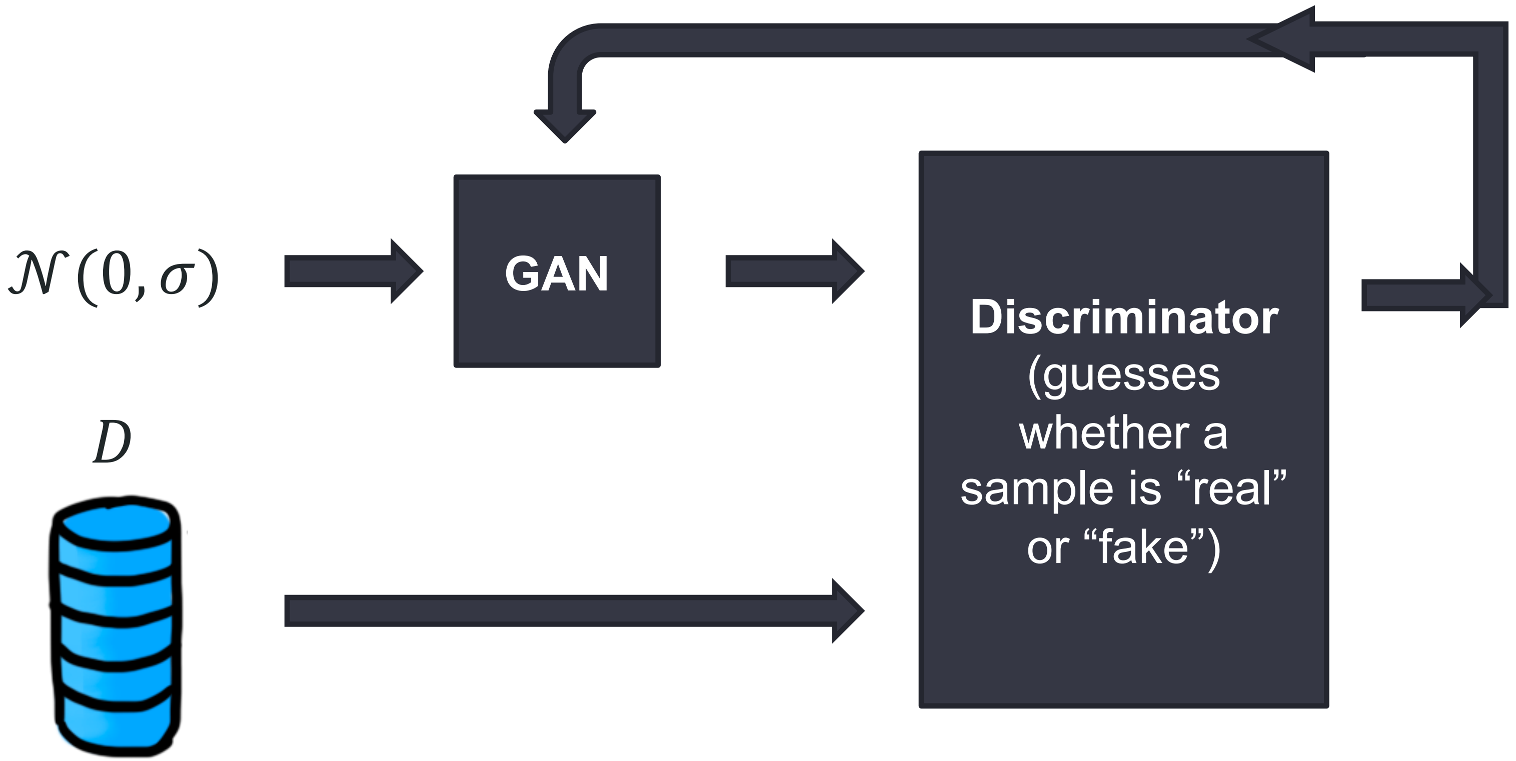
- For a sample from the incomplete public data $\vec{x}$, let $n_j(\vec{x})$ be the number of teachers that voted for label j.
- Instead of labeling by taking $argmax_j\{n_j(\vec{x})\}$, we can add Laplacian noise to provide DP:

$$argmax_j \left\{n_j(\vec{x}) + Lap\left(\frac{1}{\gamma}\right)\right\}$$

Papernot, Nicolas, et al. "Semi-supervised knowledge transfer for deep learning from private training data." *ICLR 2017*

Synthetic Data Generation

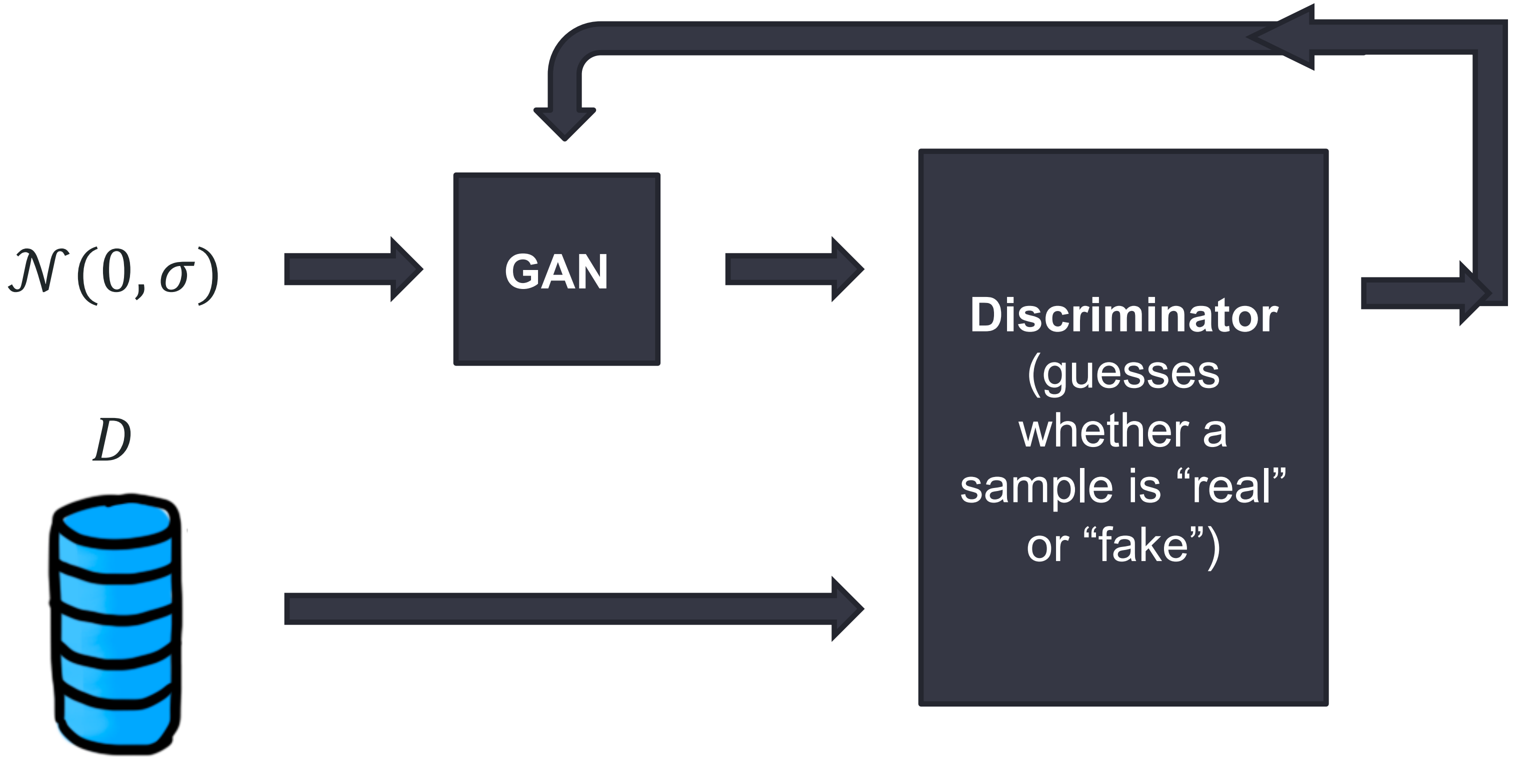
- For example, by using a GAN to generate real-looking synthetic samples:



If we train the GAN using privacy-preserving training algorithms (e.g., DP-SGD on the discriminator), we can use it to generate a privacy-preserving synthetic dataset!

Synthetic Data Generation

- For example, by using a GAN to generate real-looking synthetic samples:

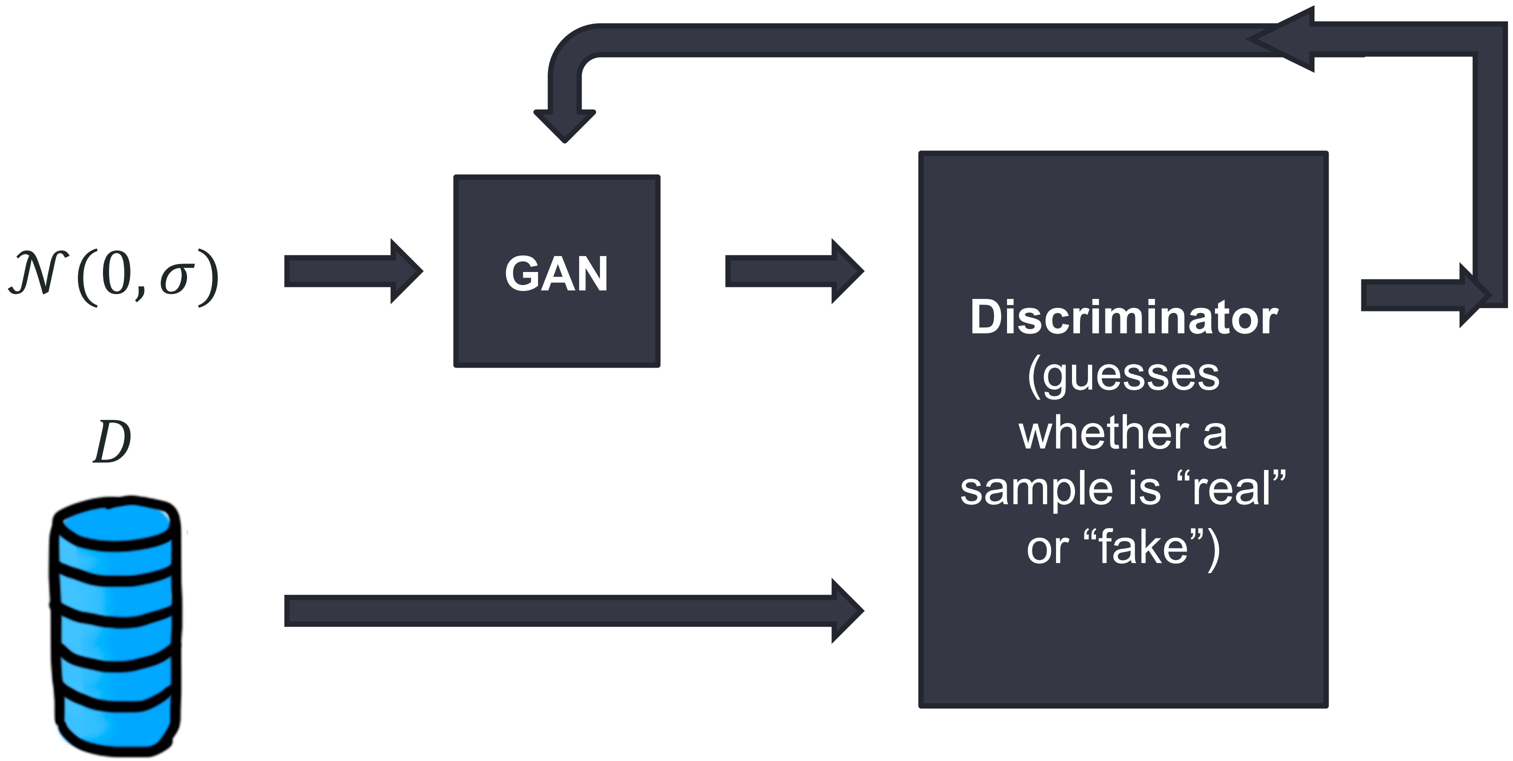


If we train the GAN using privacy-preserving training algorithms (e.g., DP-SGD on the discriminator), we can use it to generate a privacy-preserving synthetic dataset!

Q: What can we do with the resulting dataset?

Synthetic Data Generation

- For example, by using a GAN to generate real-looking synthetic samples:



If we train the GAN using privacy-preserving training algorithms (e.g., DP-SGD on the discriminator), we can use it to generate a privacy-preserving synthetic dataset!

Q: What can we do with the resulting dataset?

A: Anything by the post processing property!

Other defenses

- There are defenses that add noise to the confidence scores (MemGuard [Jia et al.]), but are not very effective.
- MIAs can work even if the model just leaks the predicted label (and not the confidence scores)
- Sometimes, **generalization** is a good defense by itself:
    - A well-generalized model will perform similarly in members (training set) and non-members (testing set)
    - Therefore, it will be harder for an adversary to decide whether a sample is a member or non-member if the model generalizes well.
    - Generalization is also **good for utility** (improves test accuracy), so it's a win-win defense.

More Details on RDP (if time)

Renyi Differential Privacy

- To introduce Renyi DP we need to know Renyi Divergence

Renyi Divergence: given two probability distributions P and Q, the Renyi divergence of order $\alpha > 1$ is

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim Q} \left(\frac{P(x)}{Q(x)} \right)^\alpha$$

- As always in this lecture, the logarithms are natural.

Renyi Differential Privacy

Renyi Divergence: given two probability distributions P and Q, the Renyi divergence of order $\alpha > 1$ is

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim Q} \left(\frac{P(x)}{Q(x)} \right)^\alpha$$

- Usually, we will define $P(x) = p_{M(D)}(x)$ and $Q(x) = p_{M(D')}(x)$.
- Abusing notation, we use $M(D)$ to denote the probability distribution of the mechanism outputs when the input is D.

Renyi DP: a mechanism $M: \mathcal{D} \rightarrow \mathcal{R}$ is (ϵ, α)-RDP (also read as "ϵ-RDP of order α") if, for any neighboring datasets D, D' it holds that

$$D_\alpha(M(D) \| M(D')) \leq \epsilon$$

Renyi Differential Privacy – DP connection

Renyi DP: a mechanism $M: \mathcal{D} \rightarrow \mathcal{R}$ is (ϵ, α)-RDP (also read as "ϵ-RDP of order α") if, for any neighboring datasets D, D' it holds that

$$D_\alpha\left(M(D) \middle| M(D')\right) \leq \epsilon$$

- Recall that, when $\alpha = \infty$, then the divergence (defined by its limit) is:

$$D_\infty(M(D)||M(D')) = \sup_x \log\left(\frac{\Pr(M(D) \in x)}{\Pr(M(D') \in x)}\right)$$

- In that case, it is easy to see that (ϵ, ∞)-RDP is equivalent to ϵ-DP

Renyi Differential Privacy: Properties

RDP Sequential Composition: if M_1 is (α, ϵ_1)-RDP and M_2 is (α, ϵ_2)-RDP, then the sequential composition (M_1, M_2) satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP

RDP to DP: if M is (α, ϵ)-RDP, then it is also $\left(\epsilon + \frac{\log\left(\frac{1}{\delta}\right)}{\alpha - 1}, \delta\right)$-DP

Renyi Differential Privacy: Properties

- We must prove the privacy of each mechanism from scratch in RDP.
- For the Gaussian it is much cleaner:

RDP of the Gaussian: The Gaussian mechanism $M(D) = f(D) + Y$ where $Y \sim N(0, \sigma^2)$ satifies $(\alpha, \frac{\alpha\Delta_2^2}{2\sigma^2})$-RDP

Moments Accountant

- Originally designed for use on DP-SGD
- Equivalent to using RDP over many different orders
- Intuition: keeping track of more information about each iteration can yield tighter analysis

Moments Accountant

- The moment's accountant keeps track of the privacy loss for different orders α.
- For each order, we can do composition over the iterations.
- At the end we can choose the order with the best ϵ, δ

Example application of the Moments Accountant

1. For each iteration, and each order compute the ϵ of RDP for the mechanism (e.g., $\frac{\alpha\Delta_2^2}{2\sigma^2}$ for the Gaussian).
2. Total each row by sequential composition theorem.
3. Compute the Approx. DP of each row using the conversion and choose the best.

Order α	**Iteration 1** ϵ	**Iteration 2** ϵ	**Iteration 3** ϵ	**Total RDP** ϵ	**Approx. DP** (ϵ, δ)
2	0.1	0.2	0.3	0.6	(1.2,1e-5)
3	0.2	0.3	0.4	0.9	(1.3,1e-5)
4	0.3	0.4	0.5	1.2	(1.25,1e-5)
5	0.4	0.5	0.6	1.5	(1.19, 1e-5)
6	0.5	0.6	0.7	1.8	(1.22,1e-5)

Note* These numbers are made up and don’t correspond to a real mechanism

Is it worth it?

- Yes!
- We also saw this in slide 31

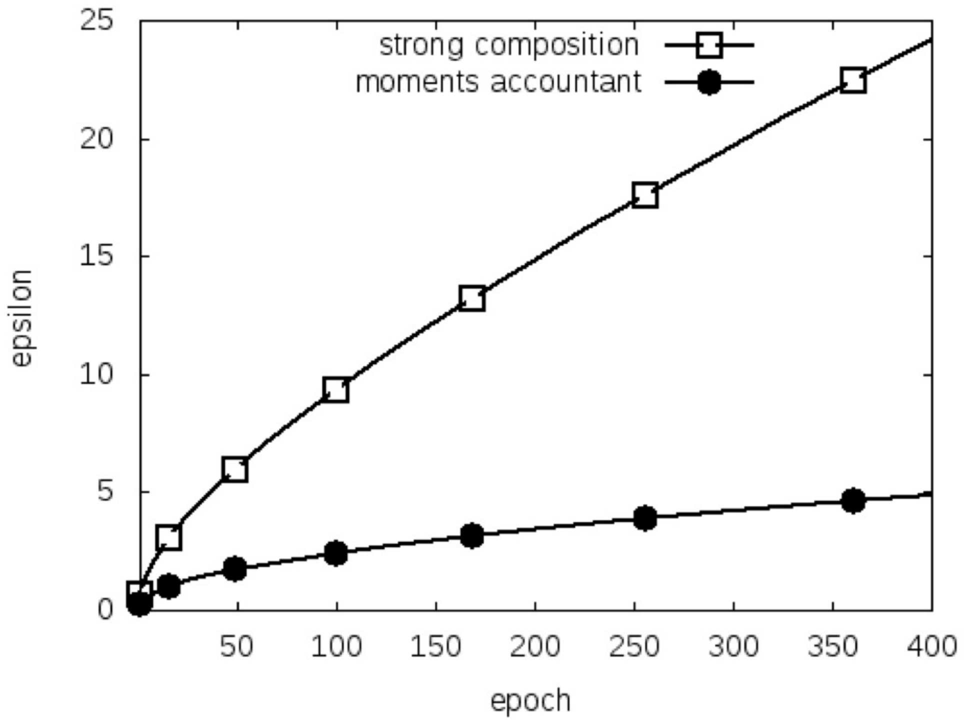


Figure 2: The ε value as a function of epoch E for $q = 0.01$, $\sigma = 4$, $\delta = 10^{-5}$, using the strong composition theorem and the moments accountant respectively.

Source Abadi et al.