

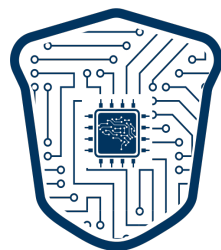
Course II: Trustworthy Private and Secured Learning

Prof. Bo Han

HKBU TMLR Group / RIKEN AIP Team

Associate Professor / BAIHO Visiting Scientist

<https://bhanml.github.io>



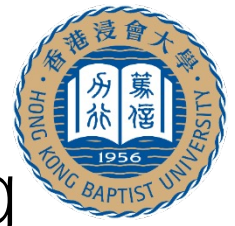
TRUSTWORTHY MACHINE LEARNING AND REASONING

TMLR



<https://bhanml.github.io/> & <https://github.com/tmlr-group>

Course II: Trustworthy Private and Secured Learning



Part 1. Privacy

How to protect your private data in machine learning?

Differential Privacy



A privacy-preserving technique that adds calibrated random noise to data to protect individual privacy

Membership Inference Attack

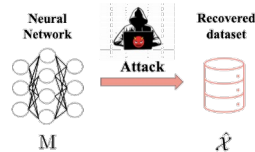


Infer whether a specific individual's data is included in a target dataset by leveraging model outputs or statistical differences.

Part 2. Attack

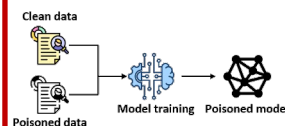
How to perform security attacks between training data and models?

Model Inversion Attack



Reconstruct private features of the training data from model

Data Poisoning Attack



Inject or modify training data to manipulate model decisions

Part 3. Governance

How to govern data and knowledge in the learning lifecycle?

Machine Unlearning



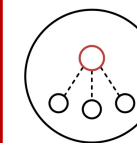
Remove specific data influence from trained models.

Non-transfer Learning



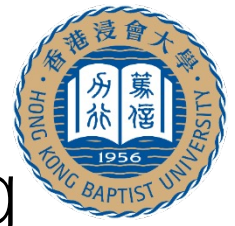
Prevent unauthorized knowledge transfer to protect model IP.

Federated Learning



Collaborative training while keeping raw data local and private.

Course II: Trustworthy Private and Secured Learning



Part 1. Privacy

How to protect your private data in machine learning?

Differential Privacy



A privacy-preserving technique that adds calibrated random noise to data to protect individual privacy

Membership Inference Attack

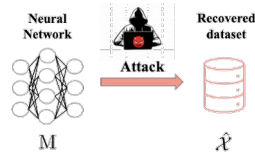


Infer whether a specific individual's data is included in a target dataset by leveraging model outputs or statistical differences.

Part 2. Attack

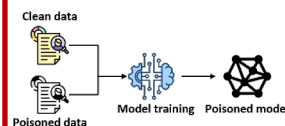
How to perform security attacks between training data and models?

Model Inversion Attack



Reconstruct private features of the training data from model

Data Poisoning Attack



Inject or modify training data to manipulate model decisions

Part 3. Governance

How to govern data and knowledge in the learning lifecycle?

Machine Unlearning



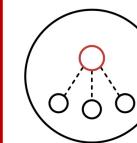
Remove specific data influence from trained models.

Non-transfer Learning

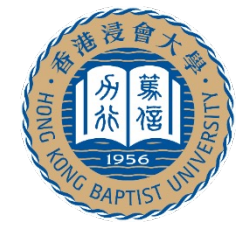


Prevent unauthorized knowledge transfer to protect model IP.

Federated Learning



Collaborative training while keeping raw data local and private.



Part 1 Privacy

- Part 1.1: Differential Privacy
- Part 1.2: Membership Inference Attack

Differential Privacy (DP)



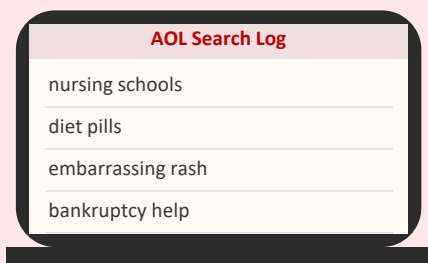
- Motivation of Differential Privacy
- Formal Definition of Differential Privacy
- Different Mechanisms in Differential Privacy
- Differential Privacy in Machine Learning
- Takeaways and Discussions

Why Privacy?

“**Anonymized**” is not anonymous — three real-world cautionary tales.

2006

AOL search log release



User ID:

4417749

Name:

re-identified!

AOL released “anonymized” search queries for 650 K users. Reporters re-identified individuals from the queries alone (NYT identified user no. 4417749 by name).

2008

Netflix Prize de-anon.



Netflix Prize Dataset	
User ID	Ratings
872-xxx	★★★★☆
299-xxx	★★★★★
154-xxx	★★★☆☆

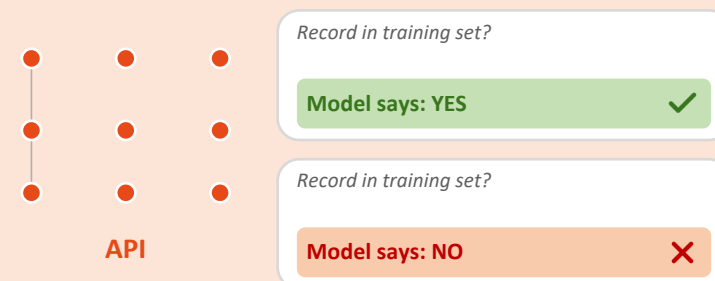


IMDb Profile (inferred)	
User ID:	872-xxx
Name:	J. Doe
Location:	California, US
Watched:	1,296 titles

Narayanan & Shmatikov linked the “anonymous” Netflix ratings dataset to public IMDb profiles, identifying individual subscribers and their viewing history.

2017

Membership inference



Shokri et al. showed black-box queries to a trained classifier can reveal whether a record was in the training set — even for cloud ML APIs.



Lesson

We need a **formal, quantitative guarantee**: one individual's data should change what an analyst sees only by a **small, controllable amount**. **Differential privacy** (Dwork et al., 2006) provides exactly this.

Intuition: A Concrete Example

Hospital query: **“How many patients in this database have HIV?”**

 **Database $\mathcal{X}$**
Alice opts in

 Alice

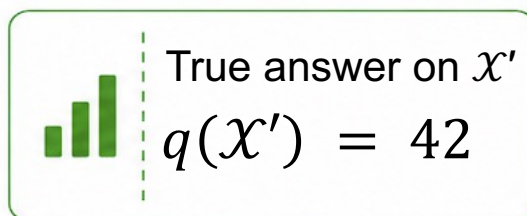
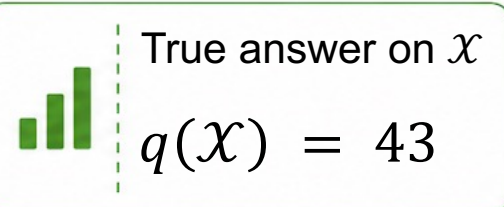
 Bob

 Carol

 Dave

...

Counting query q



 **Neighbor $\mathcal{X}'$**
Alice opts out

— (no Alice)

 Bob

 Carol

 Dave

...

 *Exact answer leaks **Alice's status***



DP fix: release a noisy answer

$$\tilde{q}(\mathcal{X}) = q(\mathcal{X}) + \text{Lap}\left(\frac{1}{\epsilon}\right)$$



Definition of Differential Privacy

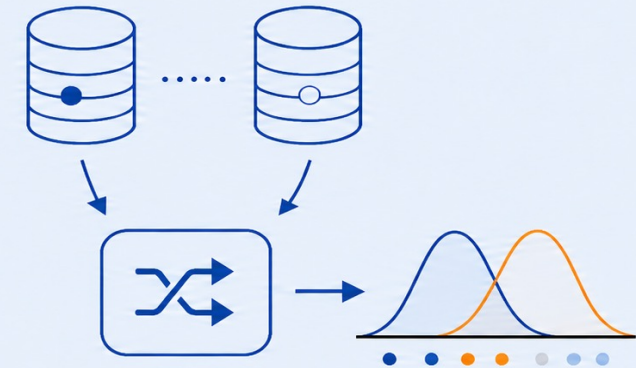
Setup:

- $\mathcal{X}$: a **database** with n records.
- $f : \mathcal{X}^n \rightarrow Y$, the **query**, the question we ask of the data. *e.g. how many patients have HIV?*
- Y : the **answer space** — the set of possible outputs of f .
- M : a **randomized mechanism** — releases $M(\mathcal{X}) = f(\mathcal{X}) + \text{noise}$ so no single record can be identified.

M is (ϵ, δ) -**differentially private** if for every pair of neighboring databases $\mathcal{X}, \mathcal{X}'$ with $\|\mathcal{X} - \mathcal{X}'\|_0 \leq 1$, and every measurable $S \subseteq Y$:

$$\Pr[M(\mathcal{X}) \in S] \leq e^\epsilon \cdot \Pr[M(\mathcal{X}') \in S] + \delta$$

where ϵ is the privacy budget. Smaller $\epsilon \Rightarrow$ Stronger privacy, more noise.
 δ is a tiny failure probability.



$$\delta = 0 \text{ (pure } \epsilon - DP)$$

- **Strict;**
- Bound holds for every output set S



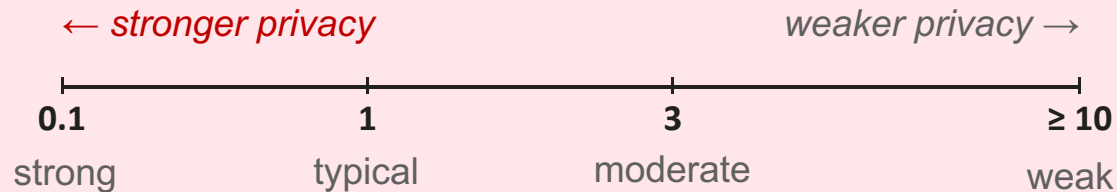
$$\delta > 0 \text{ } ((\epsilon, \delta) - DP)$$

- Tiny failure prob;
- Flexible in real-world problems.

Reading the Parameters

ϵ epsilon — privacy budget

Smaller $\epsilon \Rightarrow$ stronger privacy, more noise.



Real-world examples



US Census
2020

$$\epsilon \approx 19$$



Apple
QuickType

$$\epsilon \approx 2 \sim 8$$



Research
papers

$$\epsilon \approx 1 \sim 3$$

Research targets strong privacy ($\epsilon \leq 3$); industry trades larger ϵ for utility.

δ delta — failure probability

With probability $\leq \delta$ the bound may break.



Rule of thumb

$$\delta \ll \frac{1}{n}$$



Concrete choices



Pure DP

$$\delta = 0$$



Billion-record
datasets

$$\delta \approx 10^{-8}$$



Standard
practice

$$\delta \approx 10^{-5}$$

$\delta \ll 1/n$ means the failure event is rarer than picking out any one individual.

Adding Laplace Noise to a Number

For a query f that returns **one numeric value**, release the true answer plus **Laplace** noise **to reserve privacy**.

$$M(\mathcal{X}) = f(\mathcal{X}) + Y$$

where $Y \sim \text{Laplace}(0, b)$, $b = \Delta f / \epsilon$

What each term means

- $f(\mathcal{X})$ the true answer of the query on database $\mathcal{X}$.
- Δf global sensitivity: the maximum change in f when one record is added or removed.
- ϵ privacy budget: smaller ϵ means stronger privacy and more noise.
- Y noise sampled from the Laplace distribution.

Noise scale is calibrated to the query: $b = \Delta f / \epsilon$

Concrete example: count query

Query: “How many patients in the dataset have HIV?”

True answer: $f(\mathcal{X}) = 43$

Sensitivity: $\Delta f = 1$ (one patient changes a count by at most 1)

Privacy budget: $\epsilon = 1$

Noise scale: $b = \Delta f / \epsilon = 1$

Sample released answers (different run)

run 1: $43 + 0.7 = 43.7$

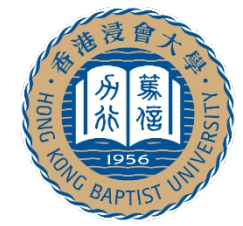
run 2: $43 - 1.3 = 41.7$

run 3: $43 + 0.2 = 43.2$



Why it protects: With the patient or without, the noisy answer looks essentially the same. So no one can tell whether the patient is in the database.

Adding Gaussian Noise to a Vector



For a query f that returns **a vector of numbers**, release the true answer plus **Gaussian** noise **to reserve privacy**.

$$M(\mathcal{X}) = f(\mathcal{X}) + Y$$

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

What each term means

$f(\mathcal{X})$ the true answer of the query on database $\mathcal{X}$.

σ noise standard deviation, chosen from $\Delta f(\text{sensitivity}), \epsilon, \delta$.

Y noise sampled from the Gaussian distribution.

Concrete example: count query

Query: *Average Blood Pressure by age group?*

True answer: $f(\mathcal{X}) = [120.5, 118.2, 125.0]$

Noise std: $\sigma = 1$

Sample released vector

[121.7, 117.4, 127.1]

each entry independently noised

✧ **Why Gaussian on a vector?** For a vector with k entries, Laplace's noise scales with k while Gaussian's scale only with $\sqrt{k}$. So Gaussian noise is less likely to drown out the signal.

Sampling an Item from a List by Score

For a query f that **picks one item from a finite list**, sample it with **probability** that grows **exponentially** with its score.

$$M(\mathcal{X}) = h, \quad h \in H \quad \text{where } \Pr[M(\mathcal{X}) = h] \propto \exp\left(\frac{\varepsilon \cdot s(\mathcal{X}, h)}{2\Delta s}\right)$$

What each term means

h	a candidate item from a finite list H (e.g. one classifier, one hyper-parameter).
H	the finite set of candidates
$s(\mathcal{X}, h)$	a score for each candidate h on data $\mathcal{X}$ (e.g., validation accuracy)
Δs	sensitivity of the score (max change in s from one record)
ε	privacy budget: smaller ε means stronger privacy and more noise.

Concrete example: pick the best classifier

Candidate:	score	Pr[picked]
h_1 Logistic regression	0.91	 0.50
h_2 SVM	0.86	 0.27
h_3 Random forest	0.78	 0.13
h_4 k-NN	0.65	 0.06
h_5 Naive Bayes	0.52	 0.04

Sample released item (different run)

run 1: h_1
 run 2: h_1
 run 3: h_2

*** Why it protects:** The top-scoring candidate wins most often, but every candidate keeps a positive chance. So no single record can block any particular choice.

DP in ML



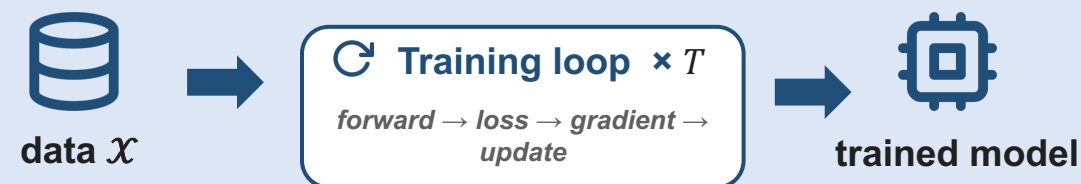
We learned DP for a single query. Machine learning is a whole training process.

DP for a single query



- Pick a mechanism (Laplace / Gaussian / Exponential)
- Add noise to $f(\mathcal{X})$ one time
- Release one noisy answer

Machine learning is iterative



- T iterations touch the data many times
- Model parameters change at every step
- Trained model is used for future queries



The new question

What can we do to make sure that the **trained model** is differentially private?

Trained Models Leak Training Data

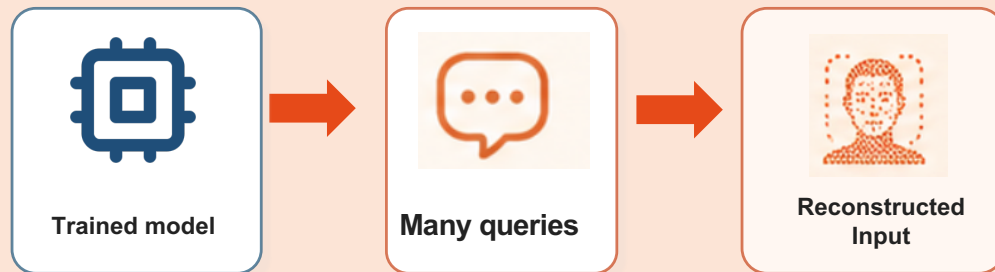
The trained model itself becomes the attacker's data source — no database access needed.



Fredrikson et al., 2015

Model Inversion Attack

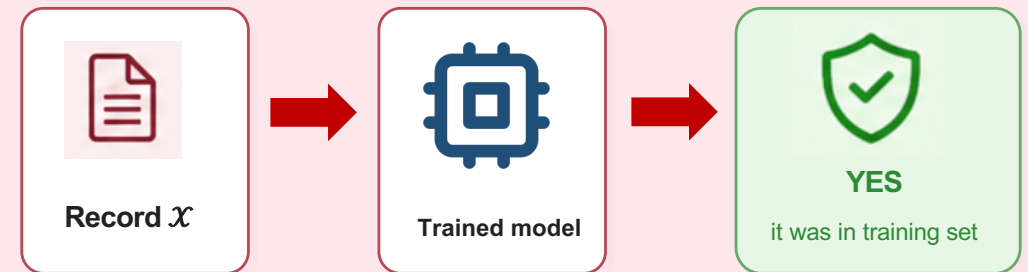
An adversary repeatedly queries the trained model and uses the outputs to reconstruct private training inputs.



Shokri et al., 2017

Membership Inference Attack

An adversary asks whether a record was used for training. Confidence gaps between seen and unseen records leak membership.



The trained model itself is the leak. **DP must be integrated into the entire training pipeline**, not just into one query.

Where Can Noise Be Injected?

Noise Perturbation in Differentially Private Machine Learning

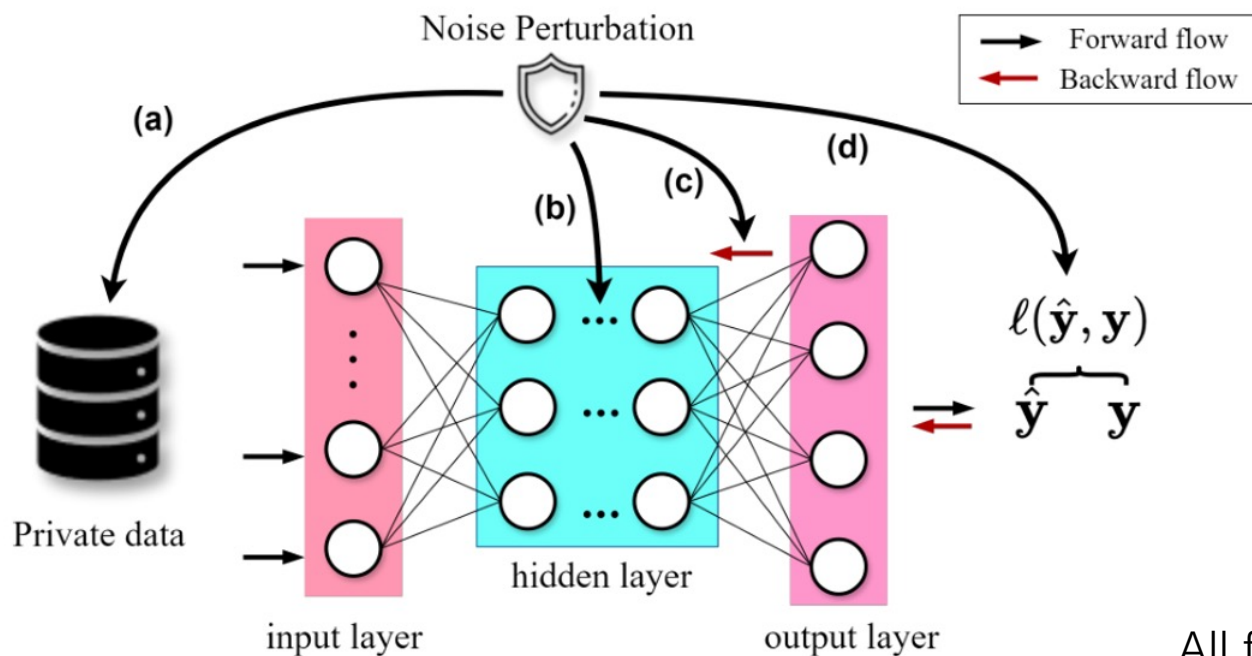


Figure 3.2 (Han et al., 2025)

Four types of Noise Perturbation

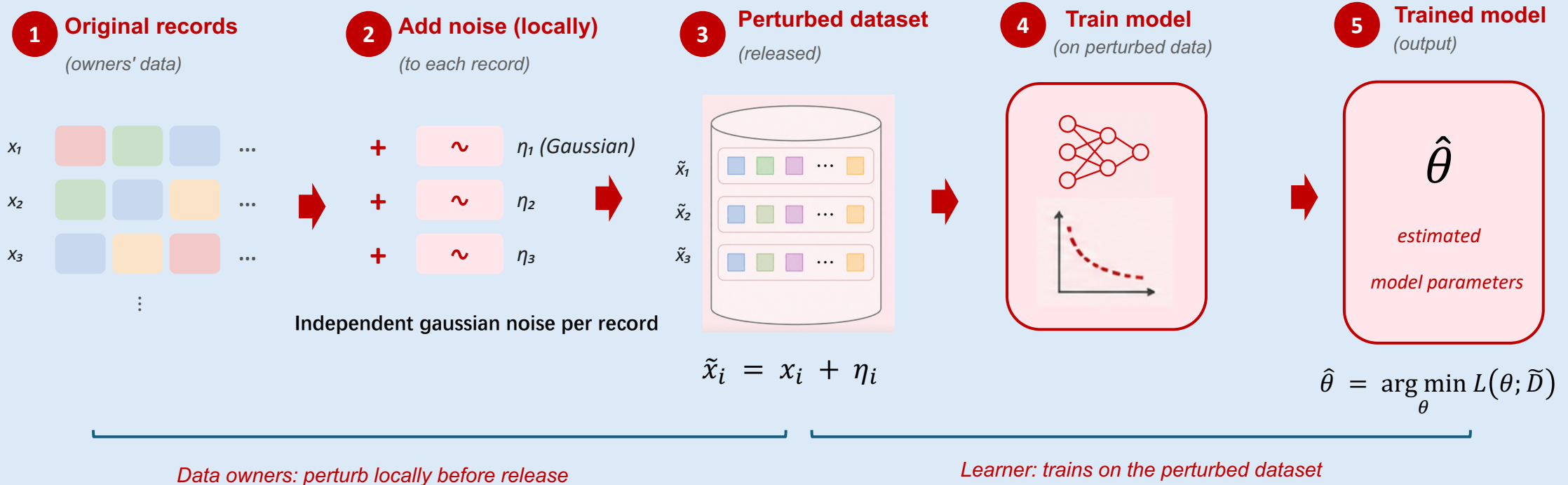
- (a) **Input**
noise in the data
- (b) **Objective**
noise in the loss
- (c) **Gradient**
noise in each step
- (d) **Output**
noise in the model

All four protect the same thing: **the released model**

Input Perturbation

(a) Input perturbation Fukuchi, Tran & Sakuma (2017)

Idea: Data owners add noise to each record before release.
Train the model on the perturbed dataset;



Objective Perturbation

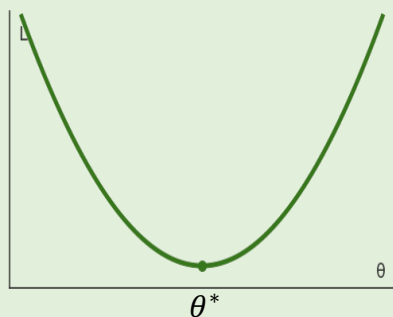
(b) Objective perturbation



Chaudhuri & Monteleoni (2008); Kifer et al. (2012);

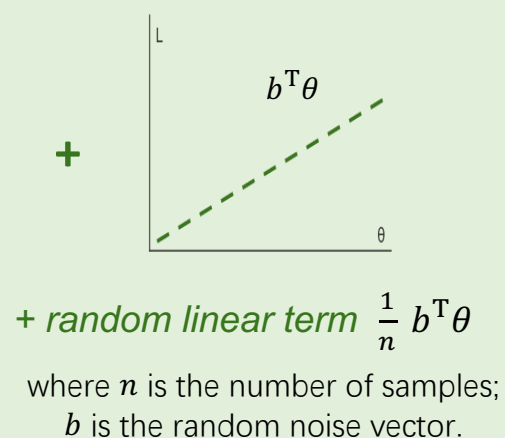
Idea: Add a random linear term to the training loss; release the (approximate) minimizer of the noisy objective.

1 Original objective

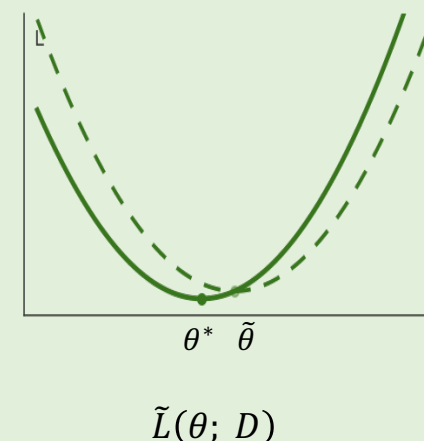


$L(\theta; D)$

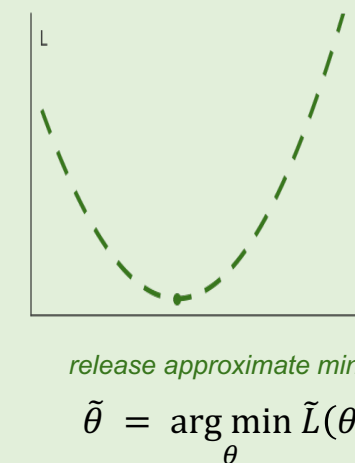
2 Add random linear term



3 Noisy objective



4 Optimize and release

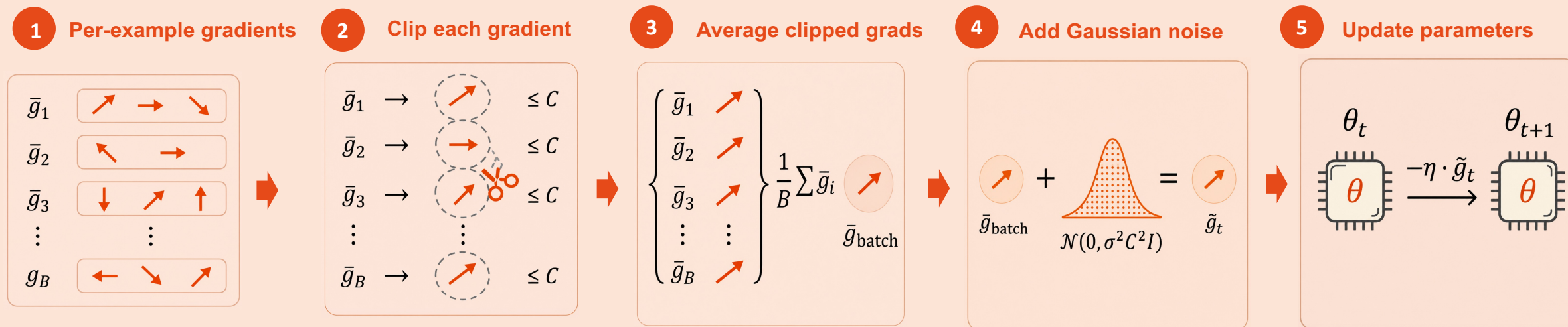


Gradient Perturbation

(c) Gradient perturbation ▽

Abadi et al. (2016 — DP-SGD)

Idea: At each step, clip per-example gradients to bound their ℓ_2 -norm, then add Gaussian noise before the update.



Clipped gradient



$$\bar{g}_i = \nabla L(\theta; x_i) \cdot \min\left(1, \frac{C}{\|\nabla L\|_2}\right)$$

If $\|\nabla L\|_2 \leq C$: keep gradient as-is.

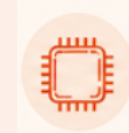
Otherwise : scale down to have norm exactly C .

Noisy batch gradient



$$\tilde{g}_t = \frac{1}{B} \left[\sum_i^B \bar{g}_i + \mathcal{N}(0, \sigma^2 C^2 I) \right]$$

Parameter update



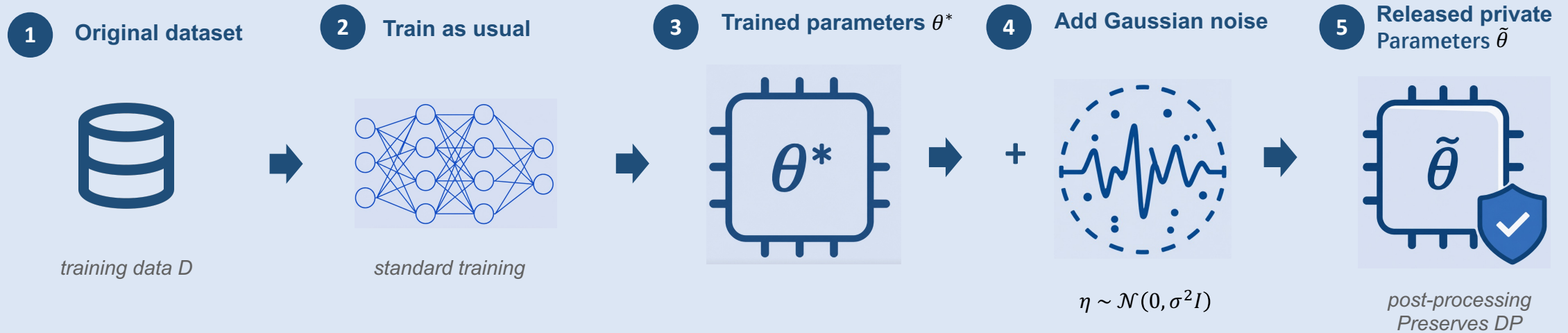
$$\theta_{t+1} = \theta_t - \eta \cdot \tilde{g}_t$$

Output Perturbation

(d) Output perturbation

Lecuyer et al. (2019); Phan et al. (2019)

Idea: Train as usual, then add noise to the final parameters (or to the second-to-last layer).



$$\theta^* = \arg \min_{\theta} L(\theta; D)$$

$$\tilde{\theta} = \theta^* + \mathcal{N}(0, \sigma^2 I)$$

Takeaways



Differential privacy is a mathematical guarantee, not a feeling.

- (ϵ, δ) –**DP** bounds how much any single record can change the released output's distribution.
- Smaller $\epsilon \Rightarrow$ stronger privacy, more noise. δ is a tiny failure probability (typically $\ll 1/n$).

Three canonical mechanisms cover the main output types.

- **Laplace** and **Gaussian** add noise to numeric outputs (counts, gradients, parameters).
- **Exponential** picks the best item from a finite set.

DP in machine learning protects the trained model.

- Noise can be injected at the **input**, **objective**, **gradient**, or **output** stage — all four protect the released model.
- **DP-SGD** (clip per-example gradients + add Gaussian noise) is the **workhorse** for training private deep models.

Discussions



Things to think about

- **How small should ϵ be?** Smaller ϵ means stronger privacy but more noise — what's the right value?
- **Privacy vs. accuracy.** More noise hurts the model's accuracy. How much accuracy can we afford to lose?
- **Where to add the noise?** We saw four places (input, objective, gradient, output). Which one fits your task?
- **Will users trust it?** A formal guarantee is great — but the guarantee also has to be explained.

When to use which mechanism

- **Counts, histograms, simple statistics** → Laplace mechanism.
- **Training neural networks** → Gaussian noise on gradients (DP-SGD).
- **Picking the best item from a list** → Exponential mechanism.
- **Protect data before it leaves the user** → Input perturbation.

DP in the wild



US Census 2020

Large-scale demographic release. $\epsilon \approx 19$ per release.



Apple QuickType

Keyboard learning with local DP. $\epsilon \approx 2 \sim 8$ per user per day.



Research & industry

Most academic papers and deployed pipelines: $\epsilon \approx 1 \sim 10$.

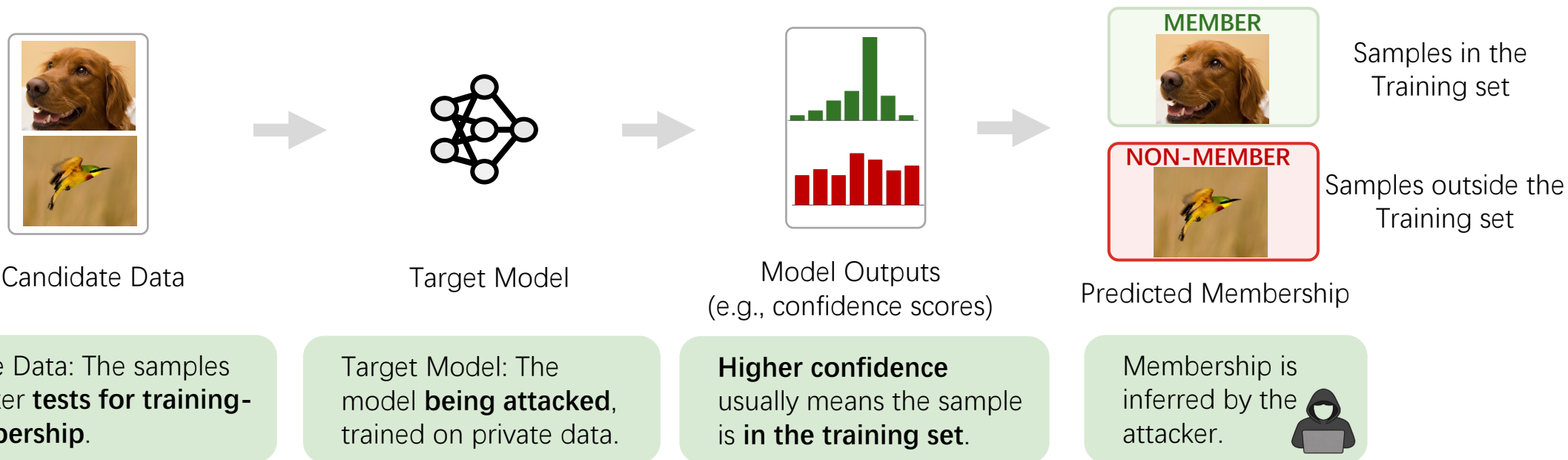
Membership Inference Attacks (MIA)



- Background of Membership Inference Attacks
- Behavior-based Membership Inference Attacks for Discriminative Model
- Loss-based Membership Inference Attacks for Generative Model
- Limitations of Membership Inference Attacks
- Takeaways and Discussions

What is MIA

Membership inference attack is a **privacy attack** that decides whether a candidate sample was **included in the training set** of a target model.

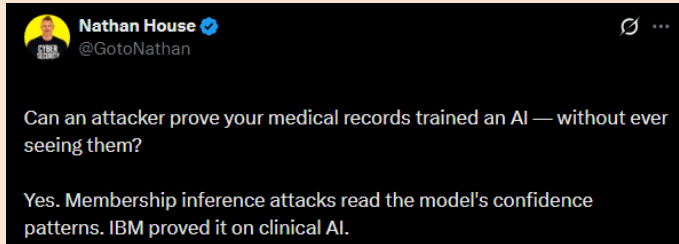


Confidence score is just one example; other signals (e.g., loss, prediction entropy) can also be used.

Why MIA Matters?

Healthcare Privacy

- MIA may reveal whether a patient's clinical record was used to train a disease-related model. This can indirectly **expose sensitive health conditions**.



Political Profiling

- If models are trained on voter-intention or supporter data, MIA can **expose political affiliation**.

Labour glitch put voting intentions data of millions at risk

Exclusive: experts say sensitive information could potentially have been harvested and used for targeted election interference

Copyright Auditing

- MIA can be used as an audit signal to test whether copyrighted works appear in training data, which matters for ongoing disputes over **AI training transparency**.

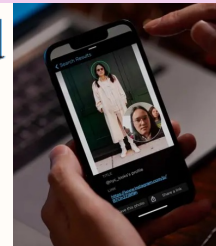
Researchers suggest OpenAI trained AI models on paywalled O'Reilly books

Kyle Wiggers · 1:10 PM PDT · April 1, 2025

Biometric / Identity Privacy

- MIA may reveal whether a person's face/image was included in a vision or generation model's training set. This raises **surveillance and consent concerns**.

An AI Company Scraped Billions of Photos For Facial Recognition. Regulators Can't Stop It

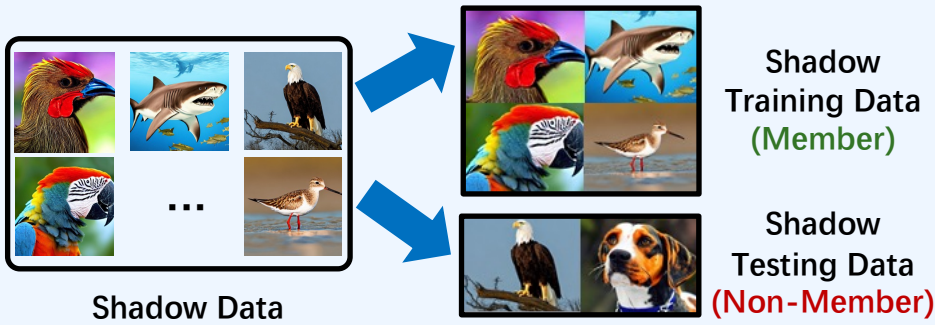


Behavior-based MIA for Discriminative Model

Motivation: Use **shadow data** to train **shadow models** and learn **membership signals** from **member and non-member behavior**. Then, use these signals to infer whether a candidate sample was included in the training data.

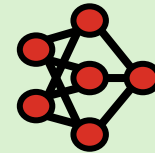
Stage 1: Collect Shadow Data

Attacker collects auxiliary (shadow) data which has **similar distribution** with **training data** to form simulated datasets.



Stage 2: Training Shadow Model

Shadow model trained on shadow training data (member) to **imitate the target model**.

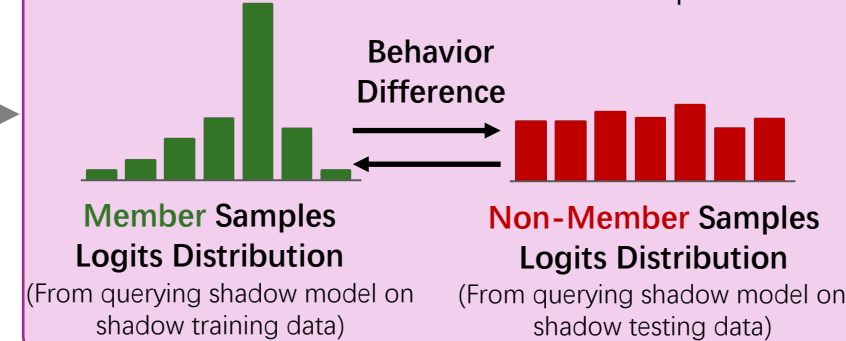


Shadow Model

Attacker knows the ground-truth membership because it trains the shadow model itself

Stage 3: Collect Membership Signal

Collect **behavioral** differences between **member and non-member** samples.



The collected membership signal can be used to

Train Attack Model

Use membership signals as features to train a binary classifier to predict membership.

Estimate Output Distributions

Estimate per-sample IN/OUT output distributions and compare likelihood ratios.

Train Better Shadows

Compose with target logits to train stronger, more target-like shadow models.

Compare with Reference Samples

Compare with reference samples' behavior to predict membership.

Attack Classifier (Train Attack Model)

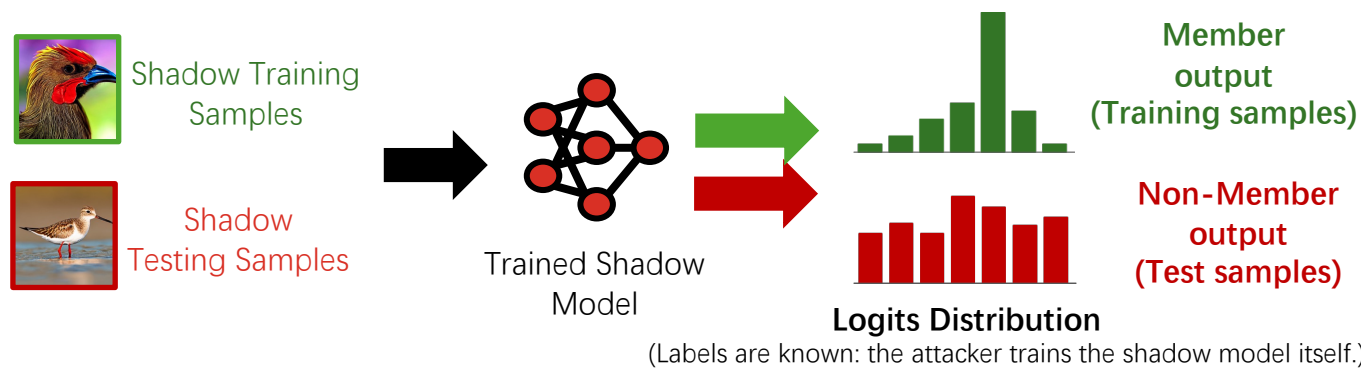
Motivation: Models memorize training samples.

- ML models **behave differently** on seen and unseen data.
- Attackers can exploit **confidence gaps** to infer membership.

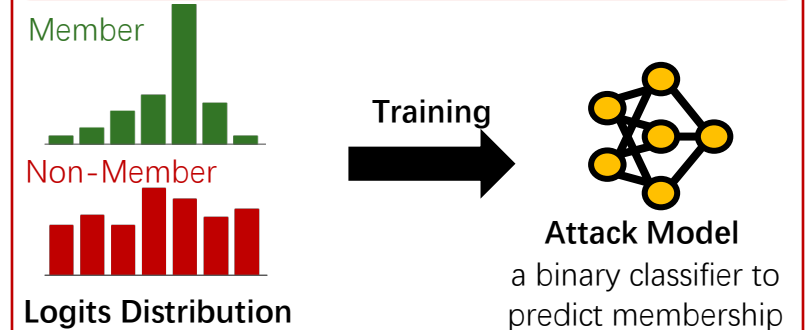
Method: Train shadow models to infer membership.

- Shadow-model training samples are labeled as **members**, while test samples are labeled as **non-members**.
- The attacker collects their prediction outputs and **trains an attack model** to classify target samples as Member/Non-Member.

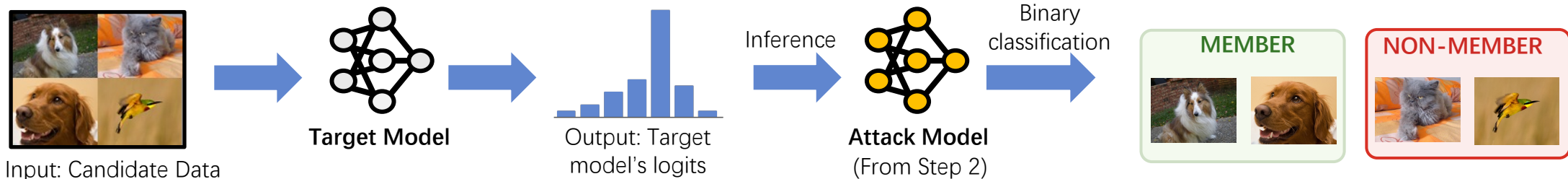
Step 1: Build the Attack Model's Training Set



Step 2: Train Attack Model



Step 3: Inference Stage (Apply to Target Model)



LiRA (Estimate Output Distributions)

Motivation: Membership leakage is sample-specific.

- Not all member samples and non-member samples show **similar output distributions**.
- Using the attack model for a unified judgment leads to a **high false positive rate**.

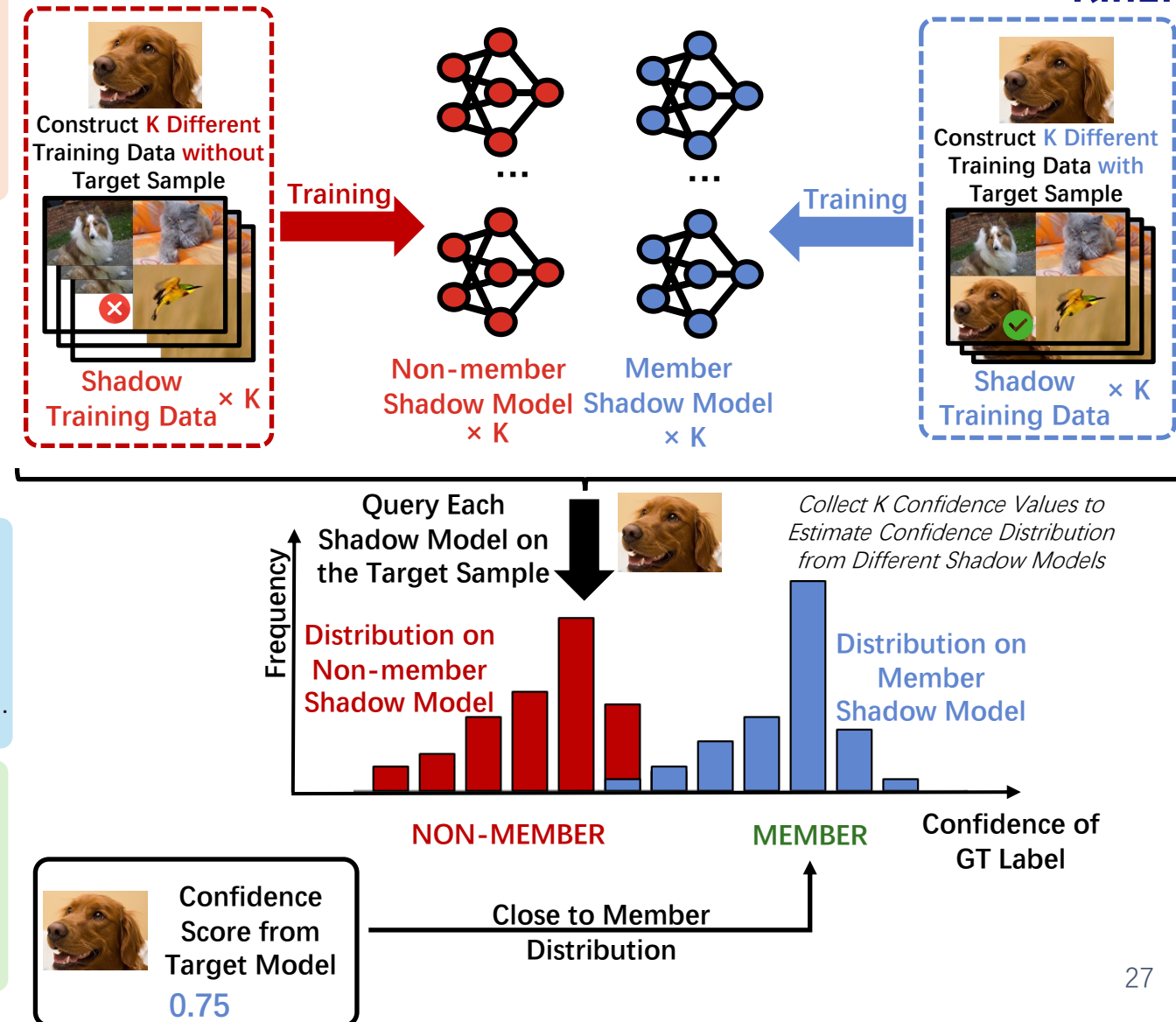


Method: Shadow model based distribution test.

- Train shadow models **with and without the target sample**.
- Collect **output distribution** from member/non-member models.
- Compare **target output** with Member/Non-Member distributions.

Key Insight: Membership is a relative signal.

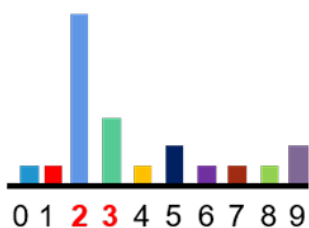
- LiRA does not rely on a **fixed loss/confidence threshold**.
- It judges whether the **target output** better fits the Member or Non-Member distribution.



L-Leaks (Train Better Shadows)

Motivation: Shadow models may not match the target.

- Traditional MIAs often rely on shadow models similar to the target model. However, in **black-box settings**, the target **architecture is usually unknown**.
- Instead of guessing the architecture, L-Leaks lets shadow models **learn the target's logit behavior**.



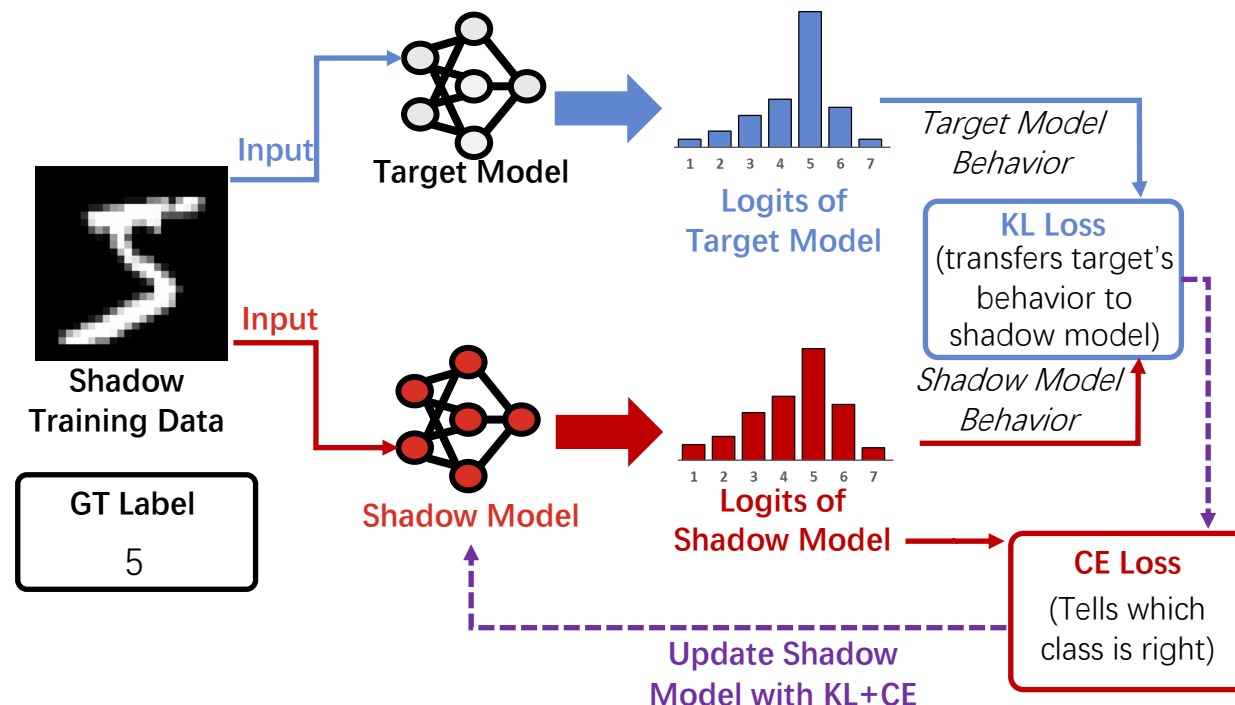
For different samples, logits will contain valuable sample knowledge. For example, the logits tell us that the **high probability** of this picture is 2, the **small probability** is 3, **hardly like** other numbers.

Key Insight: Logits reveal target behavior.

- Logits reveal how target model **ranks and separates classes**.
- Learning target logits makes the shadow model more **target-like** in black-box settings.

Method: Logit-Guided Shadow Training.

- Query the target model with shadow data to **obtain logits** which captures how the target model ranks and separates different classes.
- Train the shadow model to **match target logits**, which align the behavior of shadow model with target model.



RMIA (Reference Comparison)

Motivation: Strong MIAs should be practical.

- Existing strong MIAs require **many shadow models**.
- With **few shadow models**, prior attacks become **unstable**.
- RMIA aims for high attack power with **low computational cost**.

Method: Bayes-based relative test.

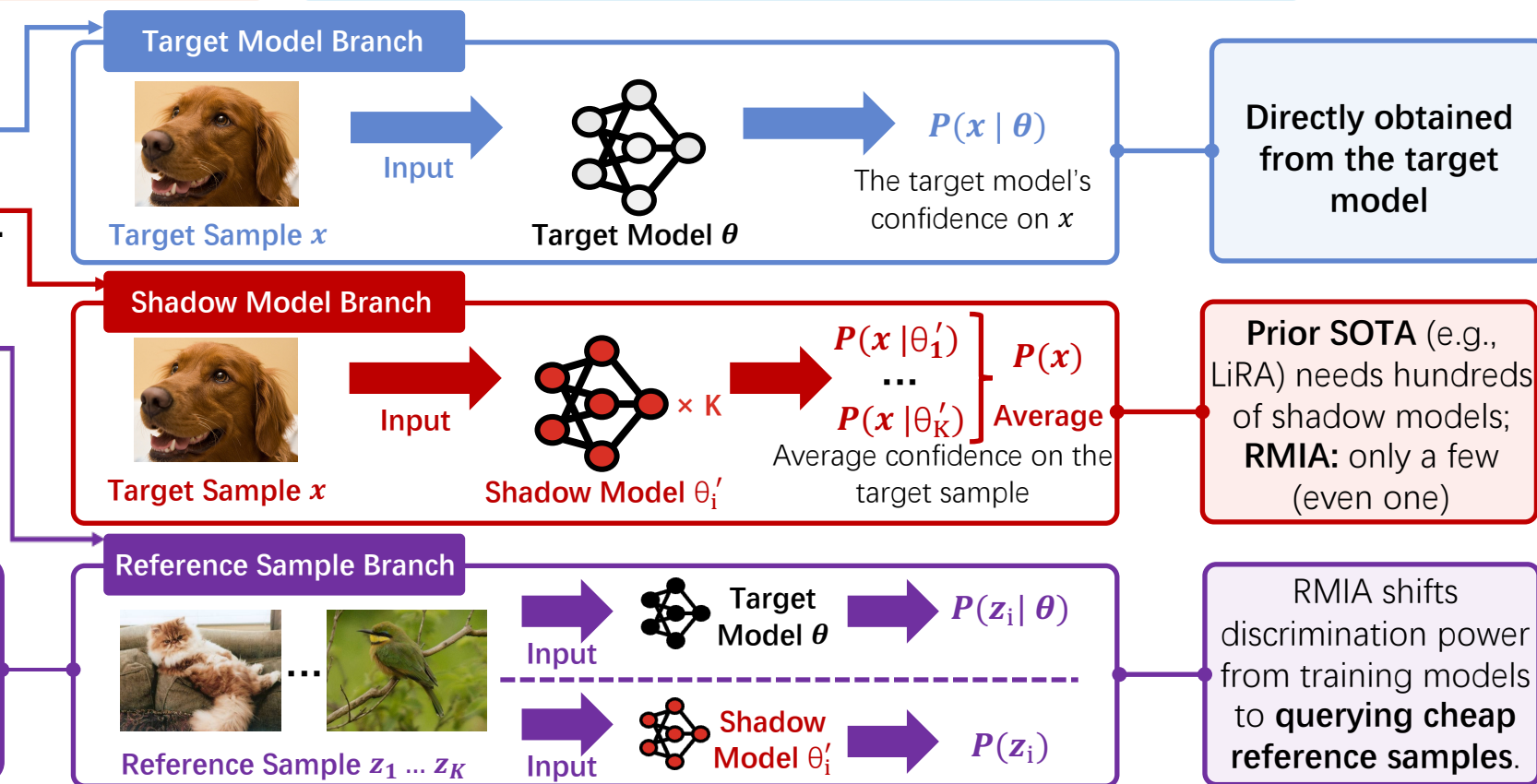
- Rewrite $P(\theta | x)$ using **Bayes rule**.
- Use **target outputs** for $P(x | \theta)$ and **shadow models** for $P(x)$.
- Compare x with many **reference samples** to see if x dominates.

If target sample x explains the target model better than most reference samples z_i do, x is likely a member.

$$\text{RMIA Score} : \frac{P(\theta | x)}{P(\theta | z_i)} = \frac{\frac{P(x | \theta)}{P(x)}}{\frac{P(z_i | \theta)}{P(z_i)}}$$

Bayes rule rewrites the intractable $P(\theta | x)$ into measurable terms — $P(x | \theta)$ from querying the target, $P(x)$ from a few shadow models.

Reference samples z_i : other samples from the same data distribution, used to determine whether the target sample behaves unusually like a training member.



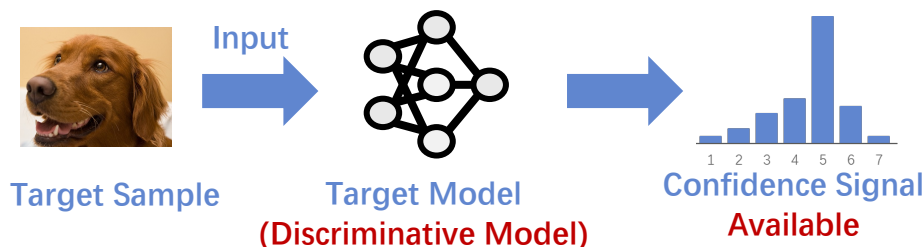
<https://bhanml.github.io/> & <https://github.com/tmlr-group>

Loss-based MIA for Generative Model

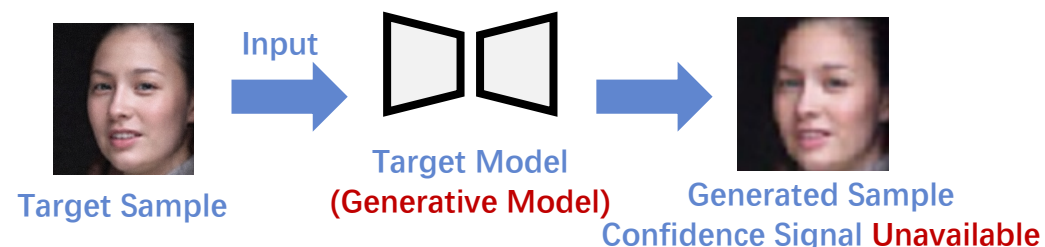
- Generative models do not expose **classifier confidence signals**.
- Membership is inferred from **loss / reconstruction error** rather than logits or confidence.
- Lower error usually indicates **stronger memorization** and **membership leakage**.

Comparison between Behavior-based and Loss-based MIA

Leverage model behavior (e.g., logits, confidence) to distinguish members from non-members.



Leverage reconstruction quality or denoising accuracy(loss / error) to infer membership.



Two Main Approaches for Loss-based MIA

GAN-Leaks (for GAN/VAE)

- Search for the generated sample closest to target x .
- Use reconstruction distance as the attack score.
- Utilize distance loss to predict **MEMBER**.

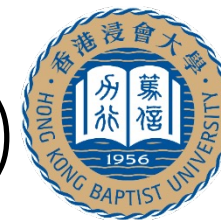
Memorized samples are easier to reconstruct.

SecMI (for Diffusion)

- Add noise to target sample until timestep x_t .
- Compare predicted noise with injected noise.
- Utilize denoising loss to predict **MEMBER**.

Members are denoised more accurately.

GAN-Leaks (Signal from Reconstruction)



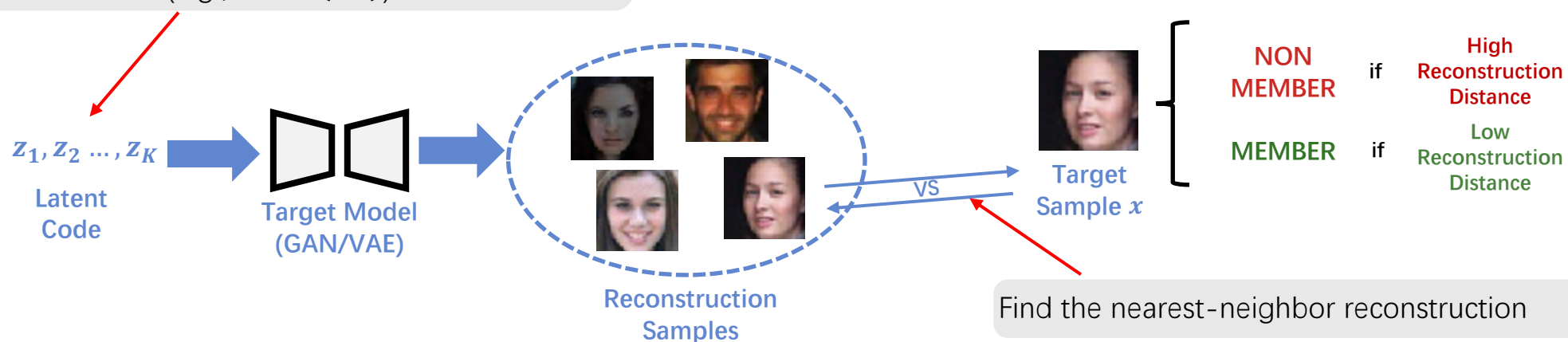
Motivation: Generators hide membership signals.

- Generative models are used to synthesize **sensitive data**.
- Unlike classifiers, generators do not **output confidence** for query samples.

Method: Reconstruction-based MIA.

- Search for the generated sample **closest to the target sample**.
- Use **reconstruction distance** as the attack score.
- Predict member if target sample can be **well reconstructed**.

Sample a batch of latent code z from prior distribution (e.g., $z \sim \mathcal{N}(0, I)$)



Key Insight: Memorized samples are easier to reconstruct.

- A generator trained on target sample x is more likely to reproduce a close version of x .
- Lower reconstruction distance indicates stronger membership leakage.

Attack Score: Reconstruction Distance

$$d(x) = \min_{z_i} L(x, G(z_i))$$

Lower score $\rightarrow$ stronger membership signal

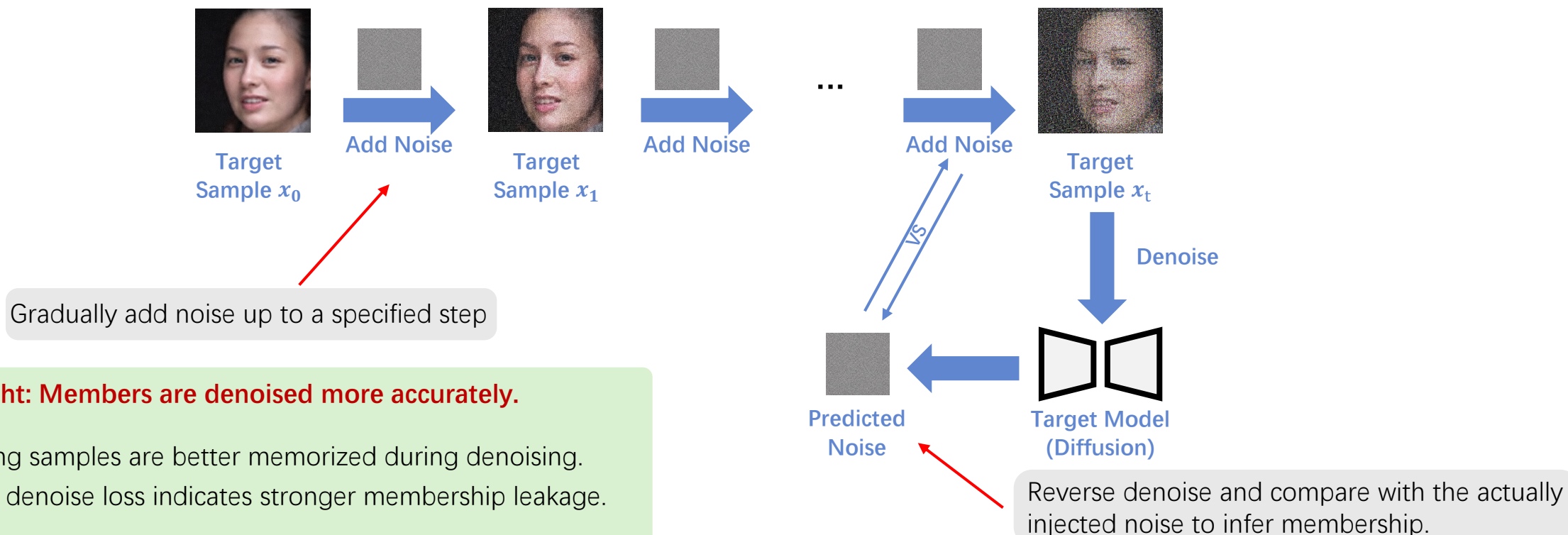
SecMI Method (Signal from Denoising)

Motivation: Diffusion models need diffusion-specific MIAs.

- GAN/VAE MIAs do not **exploit diffusion training behavior**.
- Generated samples are not **necessarily closer** to training samples.
- Membership should be measured through **denoising process**.

Method: Step-wise Error Comparing.

- Add noise to the target sample until a **selected timestep** x_t .
- Ask the diffusion model to **denoise** x_t back toward x_{t-1} .
- Compare the predicted noise with the **actually injected noise**.

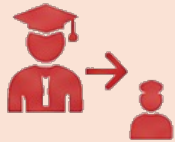


Key Insight: Members are denoised more accurately.

- Training samples are better memorized during denoising.
- Lower denoise loss indicates stronger membership leakage.

Open Challenge & Deployment

Are MIAs really effective on large models?



Distillation hides the trail. If only a **distilled student model** is exposed, can current MIA still detect data used by the **teacher**? Largely unsolved.



Skepticism around LLM results. Existing MIA studies on LLMs use **experimental setups** that many researchers consider unrealistic — not aligned with actual large-scale training.



Memorization is sample-specific. Large models memorize **a long tail of rare samples** but generalize on common ones. MIA strength varies wildly by example.

Real-world deployment frictions



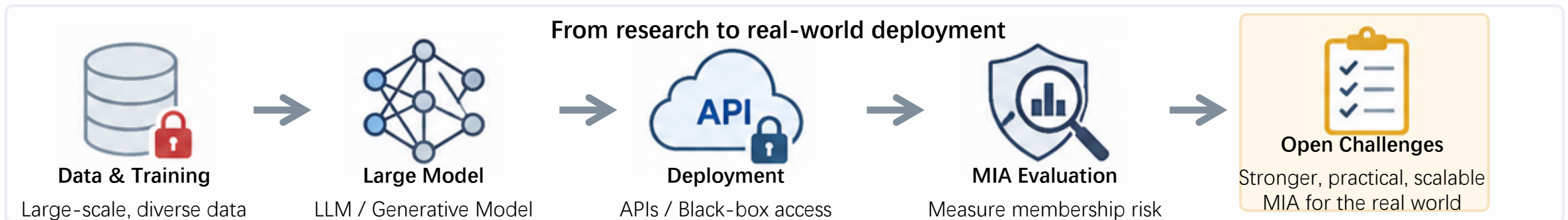
Black-box APIs. Many providers expose **only top-k labels or short responses**, blocking access to confidence vectors and gradients.



Differential privacy & noise. Defenses such as DP-SGD add noise during training to **hide each sample's footprint**. That's exactly the footprint MIA looks for — but stronger noise also hurts model accuracy.



Compute and audit cost. Strong attacks like LiRA and RMIA still need **GPUs to retrain shadows** — non-trivial for teams auditing third-party APIs.



Takeaways



Threat: MIA reveals that models can leak whether a specific sample was used in training.

- Even one membership bit may expose sensitive information such as health status, identity, or copyrighted data use.
- The risk exists even when the model does not directly reveal the original training data.

Methods: MIA uses model behavior as evidence of memorization.

- Behavior-based methods such as LiRA, L-Leak, and RMIA compare model responses to infer membership.
- Loss-based methods use lower loss or better reconstruction as signals that a sample may be a member.

Challenges: Real-world MIA is still difficult to apply reliably.

- Black-box APIs, limited outputs, and distilled models make membership signals harder to observe.
- Large foundation models memorize unevenly, so attack success can vary strongly across samples.

Discussions

1

Realistic evaluation

Test under black-box APIs, top-k outputs, and LLM/generative settings. Report TPR at very low FPR, not only overall accuracy.

2

Scalable auditing

Reduce dependence on many shadow models and repeated retraining. Make low-cost audits feasible for third-party or large-scale models.

3

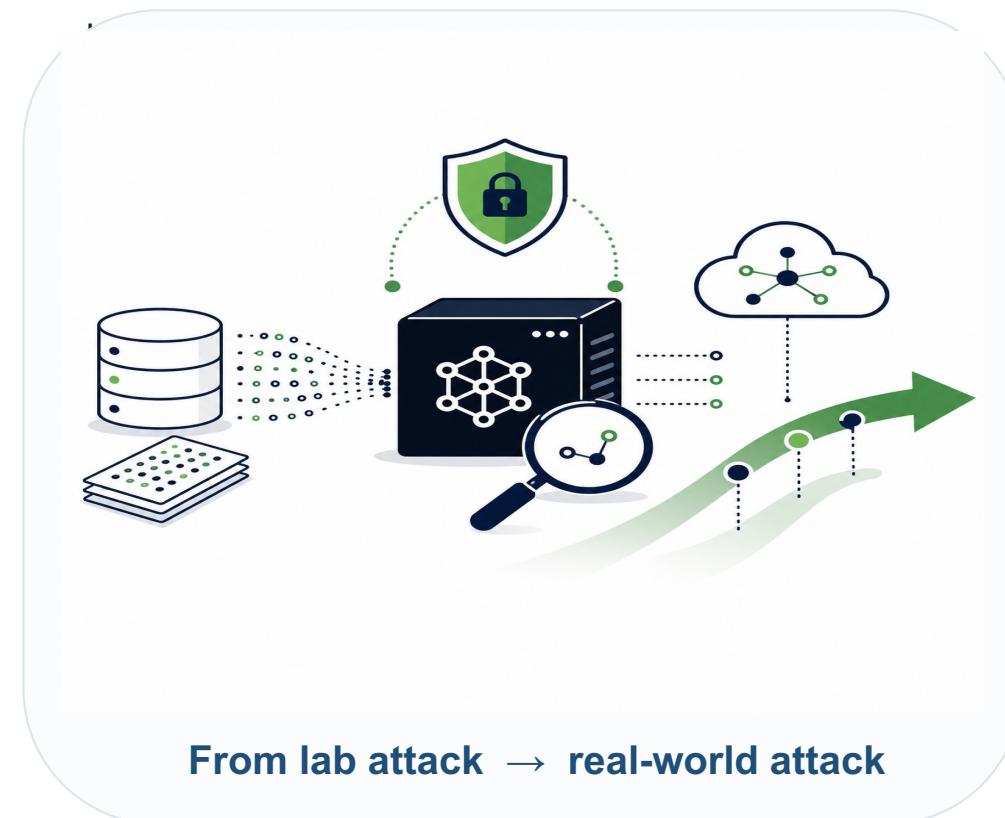
Robustness across model release

Study whether membership signals survive distillation and fine-tuning. Separate memorization from sample difficulty and distribution shift.

4

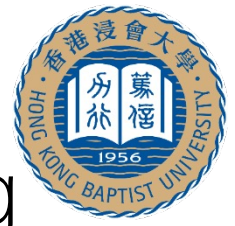
Defense-aware MIA

Compare risk before and after DP, unlearning, and data filtering. Study privacy–utility trade-offs, not only attack success.



Open discussion: What counts as enough evidence for a membership claim in the real-world setting?

Course II: Trustworthy Private and Secured Learning



Part 1. Privacy

How to protect your private data in machine learning?

Differential Privacy



A privacy-preserving technique that adds calibrated random noise to data to protect individual privacy

Membership Inference Attack

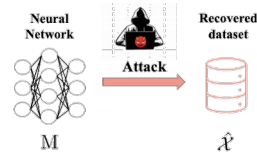


Infer whether a specific individual's data is included in a target dataset by leveraging model outputs or statistical differences.

Part 2. Attack

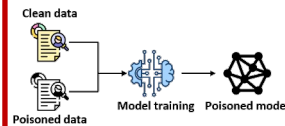
How to perform security attacks between training data and models?

Model Inversion Attack



Reconstruct private features of the training data from model

Data Poisoning Attack



Inject or modify training data to manipulate model decisions

Part 3. Governance

How to govern data and knowledge in the learning lifecycle?

Machine unlearning



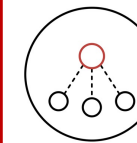
Remove specific data influence from trained models.

Non-transfer learning

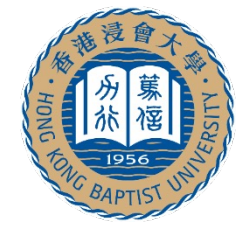


Prevent unauthorized knowledge transfer to protect model IP.

Federated learning



Collaborative training while keeping raw data local and private.



Part 2 Attack

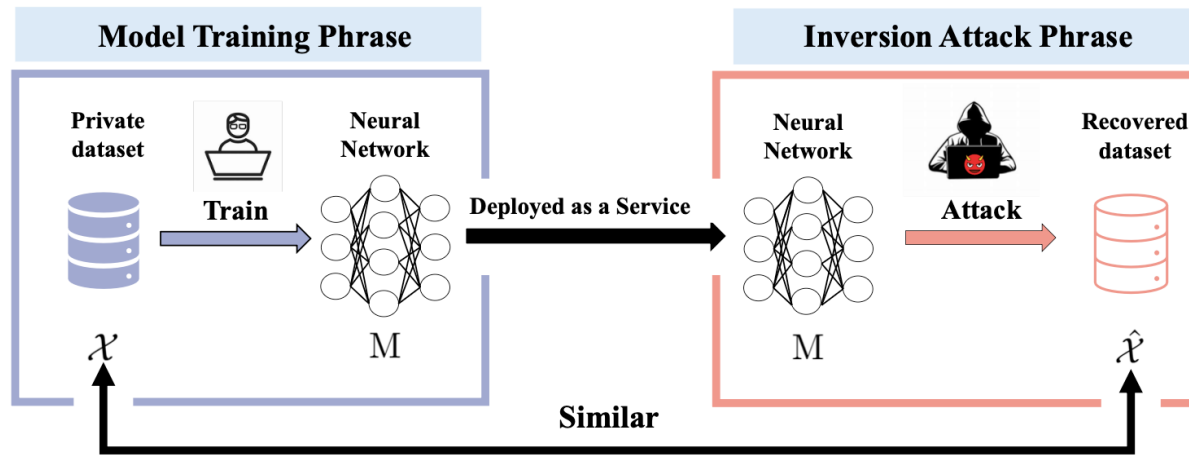
- Part 2.1: Model Inversion Attack
- Part 2.2: Data Poisoning Attack

Model Inversion Attack and Defense

- Overview of Model Inversion Attack
- Model Inversion Attack
- Defense against Model Inversion Attack
- Takeaways and Discussions

Model Inversion Attack

- **Given a trained model, the attacker tries to reconstruct private features of the training data**
- Attack surface: trained model
- Outputs, embeddings, gradients, or labels can leak information



Threat Model

- Attacker observes model signals
- Prior/domain knowledge shape recovery
- Goal: **reconstruct private attributes or representative samples**

Attack Families

- Traditional MIA
- Generative image MIA
- NLP embedding MIA
- Graph topology MIA

What's Matter Now?

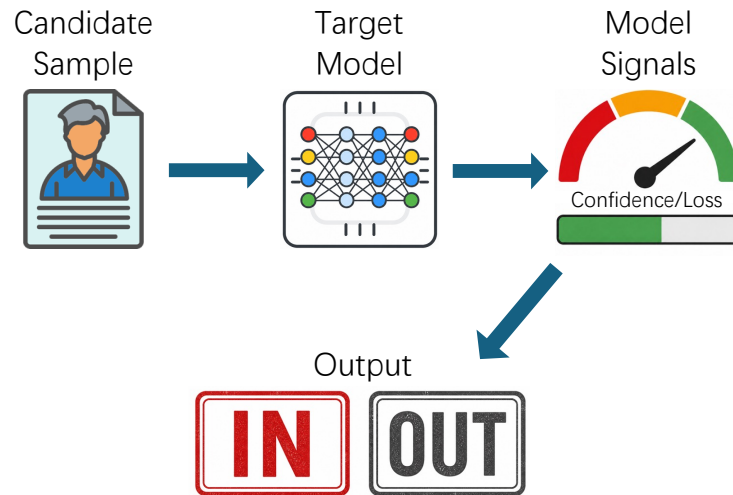
- Large models expose richer signals and public auxiliary data makes reconstruction easier.

(Credits from Han, Bo, et al. "Trustworthy machine learning: From data to models.")

Differences with Membership Inference

Membership Inference Attack

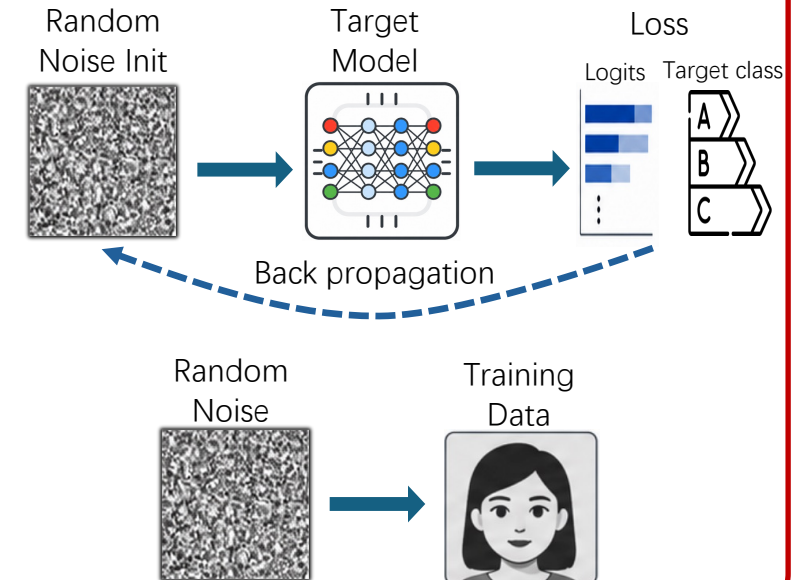
- Question: Is **this sample** in the **training set**?
- Output is a **yes/no** privacy decision
- Often exploits **overconfidence** or **loss gap**



VS

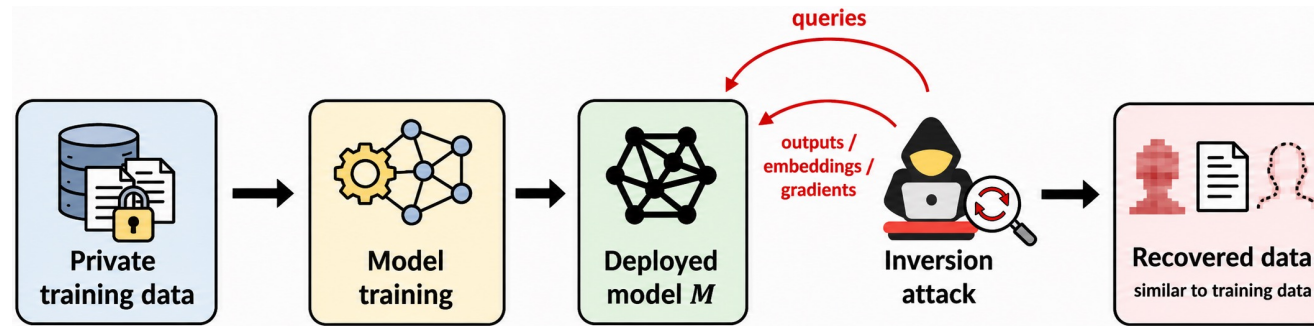
Model Inversion Attack

- Question: what does the **training data look like**?
- Output is the **reconstructed features**, image, text, or graph
- Often exploits **priors** & **model signals**



General Framework

- **Model inversion attack does not require exact record recovery**
- Reconstruction is **similar** to representative private samples



- A model M trained by data $\mathcal{X}$
- Prior knowledge $\mathcal{K}$, **hidden lever**: leaked data, public auxiliary datasets, or model architecture
- design a **attack algorithm** $\mathcal{A}$ that reconstructs as much information as possible $\hat{\mathcal{X}} = \mathcal{A}(M, \mathcal{K})$

- **Model Inversion Attack Settings**

White Box **strongest**

- Architecture
- Gradients
- Parameters

Black Box **common**

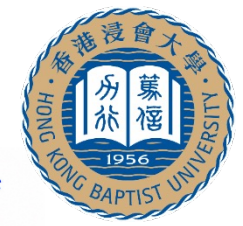
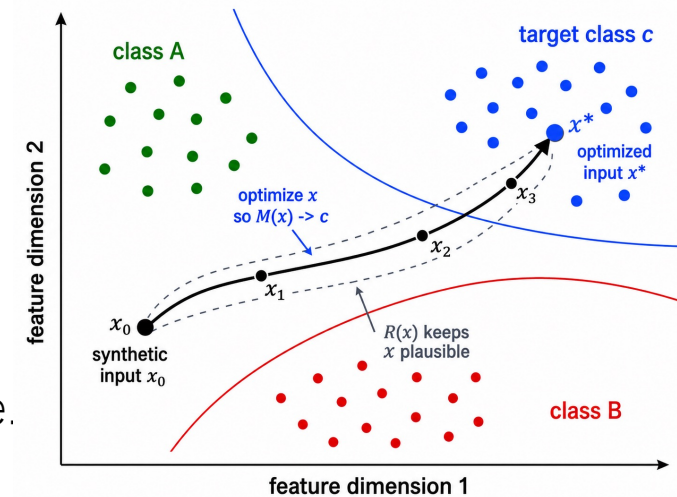
- Input-output queries
- Probabilities
- Embeddings

Label Only **harder**

- Only hard labels
- Confidence is hidden

Traditional MIA

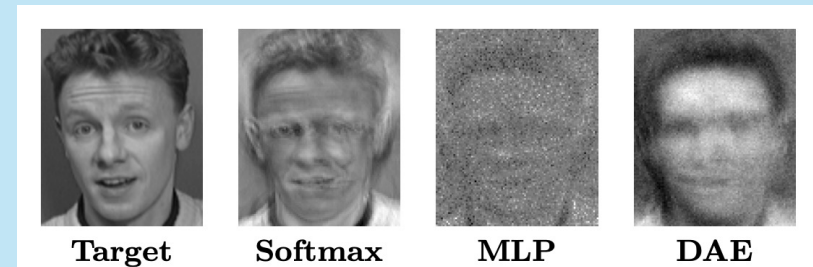
- Choose a target class c .
- Optimize a synthetic input $x \sim N(\mu, \sigma^2)$.
- Minimize discrepancy between $M(x)$ and c .
- Objective:
$$x^* = \operatorname{argmin}_x \mathcal{L}(M(x), c) + \lambda R(x)$$
- $R(x)$: regularization that avoids x to be meaningless noise.
- $\mathcal{L}$: classification loss function, cross-entropy by default.



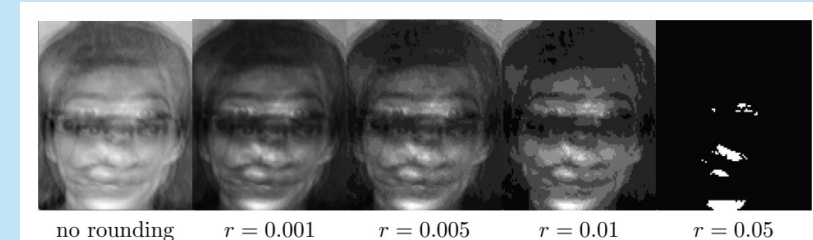
Traditional Model Inversion Attack

- Previous attacks assumed binary labels.
- Reveals that **floating-point confidence scores** are "canaries" for the training data.
- Maximize the confidence score for **a target class label** by modifying input pixels.
- Limitation: limited to **shallow model** (e.g. linear regression, SVM, and MLP)

White-box results



Defense results



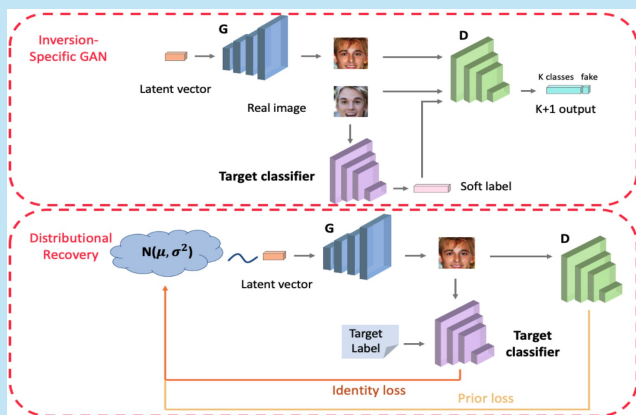
GMIA: Model & Algorithm

$$z^* = \operatorname{argmin}_z \mathcal{L}(M(G(z)), c) + \lambda R(G(z))$$

Technical focus: **generative model** and **regularization term** design.

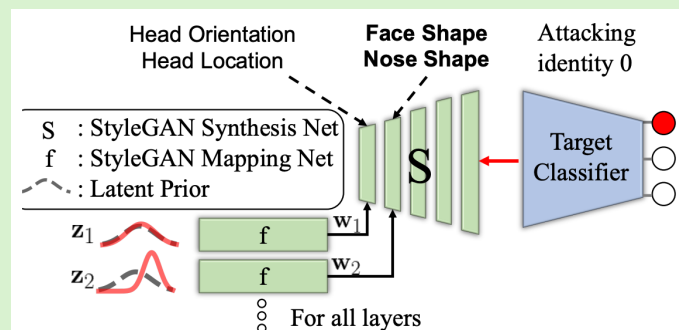
Knowledge-enriched Distributional Model Inversion Attack. ICCV 2021

- Propose **inversion-specific GAN**.
- Better **distill knowledge** useful for performing attacks.
- Model a private data distribution for **each target class**.



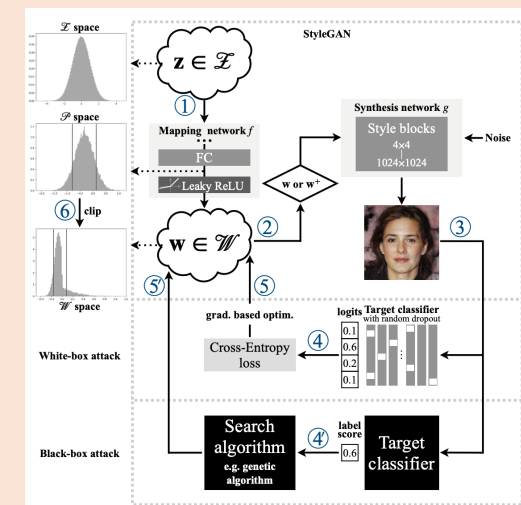
Variational Model Inversion Attacks. NeurIPS 2022

- Provide a **probabilistic interpretation** of model inversion attack.
- Formulate a **variational objective** that accounts for both diversity and accuracy.



MIRROR: Model Inversion Attack for Deep Learning Network with High Fidelity. NDSS 2022

- Introduce **StyleGAN distribution clipping** to enhance the **inversion performance**.



GMI: Loss Function

$$z^* = \operatorname{argmin}_z \mathcal{L}(M(G(z)), c) + \lambda R(z)$$

$$\frac{\partial \mathcal{L}}{\partial s_c} = 1 - p_c,$$

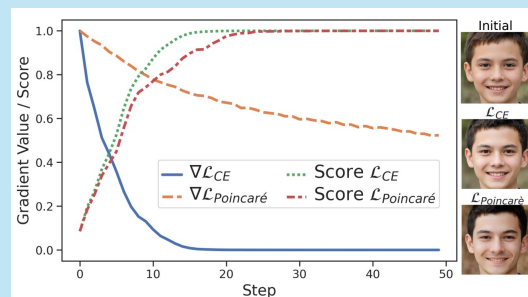
$$p_c \rightarrow 1, \frac{\partial \mathcal{L}}{\partial s_c} \rightarrow 0$$

- Issue: **Gradient Vanishing** of **CE loss**
- Technical focus: progressively **improves the MIA objective** beyond CE

p_c : confidence/probability of target class (after softmax)
 s_c : logit of target class (before softmax)

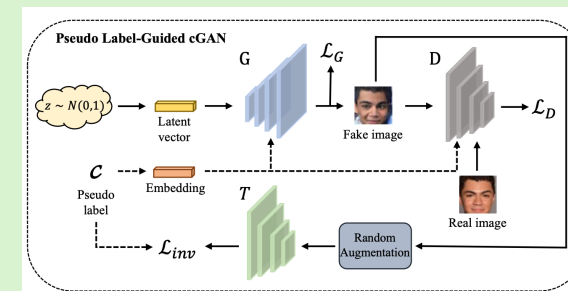
ICML'22: Plug & Play Attacks

Replacing CE with a **Poincaré loss**, which provides stronger gradients when the target confidence is high



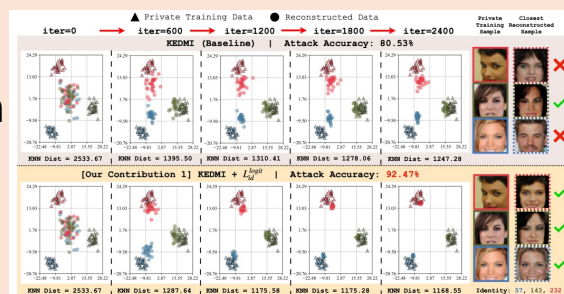
AAAI'23: Pseudo Label-Guided MIA via cGAN

Introduces a **max-margin loss** to avoid the gradient saturation problem of CE



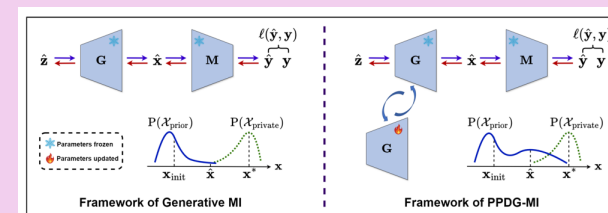
CVPR'23: Re-thinking MIA Against DNNs

Propose logit maximization identity loss to **better align reconstructed representations** with private data.



NeurIPS'24: Pseudo-Private Data Guided MIA

Use **inverted samples** as pseudo-private data to **fine-tune the generator**, increasing the density of private-like samples.

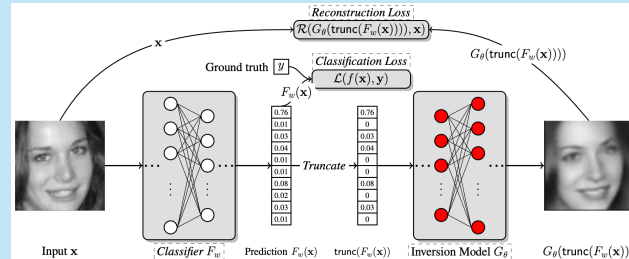


GMI: Black-box and Label-only Attack

- Goal: Recover **hidden optimization signals** from feedbacks the target model still leaks.
- Technical focus: **Replace missing gradients/scores** with auxiliary alignment, decision-boundary probing, RL search, and surrogate knowledge transfer.

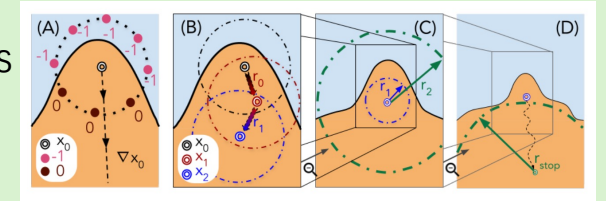
CCS'19: MIA via Knowledge Alignment

Proposes a black-box inversion model that **learns to map prediction vectors back to inputs** using only auxiliary data.



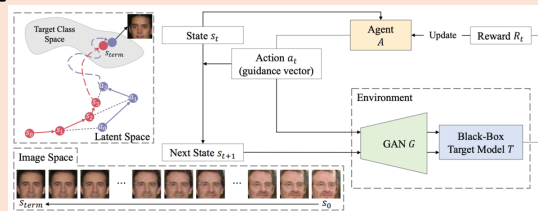
CVPR'22: Label-Only MIA via Boundary Repulsion

Introduces the **label-only MIA**. Propose to estimate directions in GAN latent space using hard-label queries and repels.



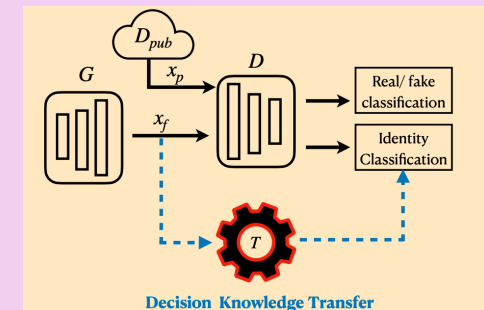
CVPR'23: RL-based Black-Box MIA

Formulates black-box latent search as a **RL problem**, and improves query efficiency and reconstruction quality.



NeurIPS'23: Label-Only MIA via Knowledge Transfer

Converts label-only inversion into a **more tractable white-box** inversion problem.



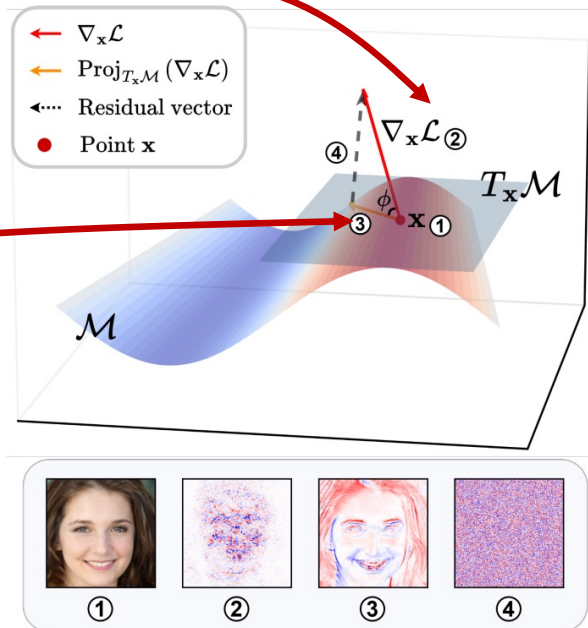
Manifold Hypothesis of GMIA Loss

Problem Understanding: Why do generative MIA works?

- Traditional MIA do not work well because in input-space optimization, loss gradient goes **off the generator manifold**.
- Generative MIA methods **implicitly** denoise these gradients by **projecting them onto the tangent space** of the manifold.

Raw loss gradients are **noisy** and go **off the manifold**.

Implicitly projects gradients onto tangent space.



- ① sample on manifold
- ② raw gradient
- ③ projected gradient
- ④ residual vector (noise)

Method & Key Insight: Smooth First, Then Let the Generator Filter

- Raw gradients contain **unstable off-manifold components**.
- Averages **transformed/perturbed gradients** to remove **off-manifold components**.

Smoothed, alignment-enhanced gradient:

$$\tilde{\nabla}\mathcal{L}(\mathbf{x}) = \mathbb{E}_{\mathbf{x}' \sim p(\cdot|\mathbf{x})} [\nabla\mathcal{L}(\mathbf{x}')]]$$

Raw gradient, contains off-manifold components

Sampling Method:

PAA: Gaussian Perturbations Averaged Alignment:

$$p(\cdot|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$$

Sample **multiple variants of x** within a local neighborhood and **average their loss gradients**.

TAA: Semantic-preserving Transformations Averaged Alignment:

$$p(\cdot|\mathbf{x}) = \text{Uniform}\{\tau(\mathbf{x}) \mid \tau \in \mathcal{T}\}$$

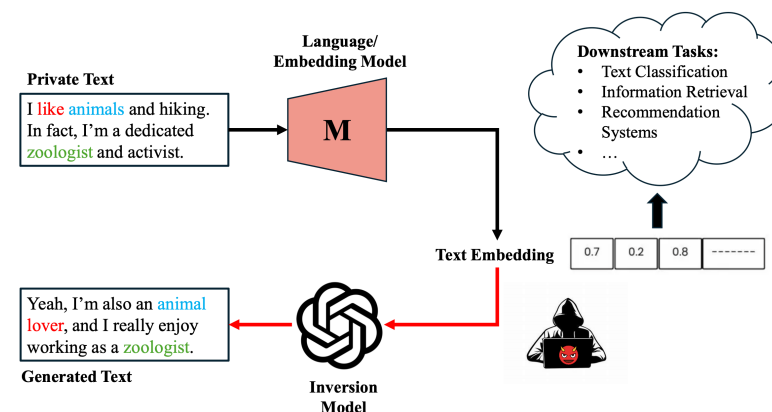
Results: Alignment increases inversion success

- PAA** improves CelebA ResNeSt-50 Acc@1: 70.75 -> 75.71
- TAA** improves FaceScrub ResNeSt-50 Acc@1: 71.58 -> 84.38

Target Model	Method	CelebA					FaceScrub				
		Acc@1↑	Acc@5↑	KNN Dist↓	Ratio↓		Acc@1↑	Acc@5↑	KNN Dist↓	Ratio↓	
ResNet-18	PPA	85.63	95.12	0.693	/		81.57	94.85	0.796	/	
	+ PAA (ours)	88.75 (+3.12)	96.59	0.669 (-0.024)	1.50		83.97 (+2.40)	95.78	0.777 (-0.019)	1.55	
	+ TAA (ours)	91.68 (+6.05)	97.68	0.662 (-0.031)	1.61		93.68 (+12.11)	98.84	0.691 (-0.105)	1.61	
DenseNet-121	PPA	82.22	93.26	0.708	/		75.66	90.91	0.786	/	
	+ PAA (ours)	85.64 (+3.42)	95.16	0.684 (-0.024)	2.82		80.70 (+5.04)	93.40	0.761 (-0.025)	2.82	
	+ TAA (ours)	87.88 (+5.66)	96.20	0.687 (-0.021)	2.87		86.54 (+10.88)	95.12	0.712 (-0.074)	2.93	
ResNeSt-50	PPA	70.75	87.43	0.793	/		71.58	90.60	0.827	/	
	+ PAA (ours)	75.71 (+4.96)	90.48	0.764 (-0.029)	2.93		73.38 (+1.80)	91.34	0.807 (-0.020)	3.12	
	+ TAA (ours)	79.19 (+8.44)	92.28	0.761 (-0.032)	3.12		84.38 (+12.80)	96.04	0.753 (-0.074)	3.13	

MIA in NLP

- Text is **discrete**, so brute gradient search is hard.
- Embeddings** create a continuous leakage channel.
- Attackers often reconstruct **sensitive attributes** or plausible private text.



Embedding models leak multiple privacy signals. CCS'20

Information Leakage in Embedding Models

Congzheng Song*
Cornell University
cs2296@cornell.edu

Ananth Raghunathan*
Facebook
ananthr@cs.stanford.edu

- Systematically studies **privacy leakage** in text embedding models.
- It shows that **embeddings can reveal input tokens**, private attributes, and training membership.

LLMs memorize extractable training data. USENIX Security'21

Extracting Training Data from Large Language Models

Nicholas Carlini ¹	Florian Tramèr ²	Eric Wallace ³	Matthew Jagielski ⁴
Ariel Herbert-Voss ^{5,6}	Katherine Lee ¹	Adam Roberts ¹	Tom Brown ⁵
Dawn Song ³	Úlfar Erlingsson ⁷	Alina Oprea ⁴	Colin Raffel ¹

¹Google ²Stanford ³UC Berkeley ⁴Northeastern University ⁵OpenAI ⁶Harvard ⁷Apple

- Show LLMs can memorize **and leak verbatim training examples** through **black-box querying**.
- Generates samples and uses membership-inference-style scoring, to identify memorized text containing PII, code, and other sensitive data.

Text embeddings can be inverted. EMNLP'23

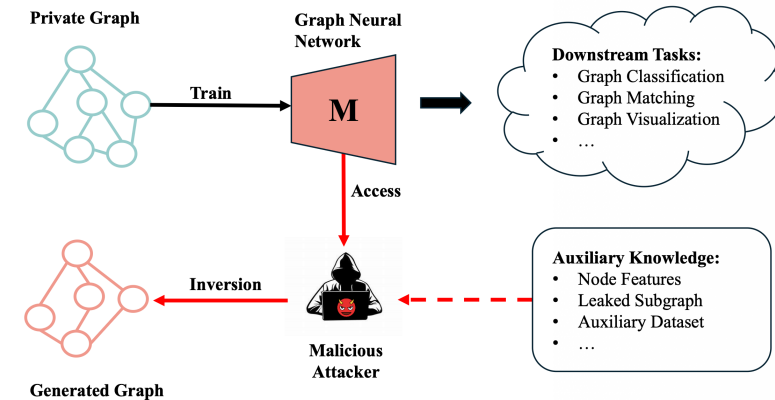
Text Embeddings Reveal (Almost) As Much As Text

John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov, Alexander M. Rush
Department of Computer Science
Cornell University

- Shows that **text embeddings are not safe anonymized representations**: they preserve enough information to **reconstruct much of the original text**.

MIA in Graph

- Goal: infer **topology, edge connectivity**, or **private graph details**
- Auxiliary knowledge: **node features, leaked subgraph**, or **explanations**
- Discrete edges require projected gradients, auto-encoders or Markov approximations



Feature explanations leak private graph structure. PoPETs'23

Private Graph Extraction via Feature Explanations

Iyiola E. Olatunji
L3S Research Center
Hannover, Germany
iyiola@l3s.de

Mandeep Rathee
L3S Research Center
Hannover, Germany
rathee@l3s.de

Thorben Funke
L3S Research Center
Hannover, Germany
tfunke@l3s.de

Megha Khosla
TU Delft
Delft, Netherlands
m.khosla@tudelft.nl

- Shows that post-hoc feature explanations for GNNs can **leak private graph structure**.
- Proposes multiple graph reconstruction attacks and finds that gradient-based explanations **leak the most structural information**.

GNNs memorize private links Markov-chain signals. ICML'23

On Strengthening and Defending Graph Reconstruction Attack with Markov Chain Approximation

Zhanke Zhou¹ Chenyu Zhou¹ Xuan Li¹ Jiangchao Yao^{2,3} Quanming Yao⁴ Bo Han¹

- Shows that node features, labels, hidden representations, and outputs can **all leak adjacency information**.
- Provide an information-theoretic defense that **reduces link leakage with moderate accuracy loss**.

GNN model inversion can reconstruct training edges. IJCAI'21

GraphMI: Extracting Private Graph Data from Graph Neural Networks

Zaixi Zhang¹, Qi Liu^{1*}, Zhenya Huang¹, Hao Wang¹, Chengqiang Lu², Chuanren Liu³, Enhong Chen¹

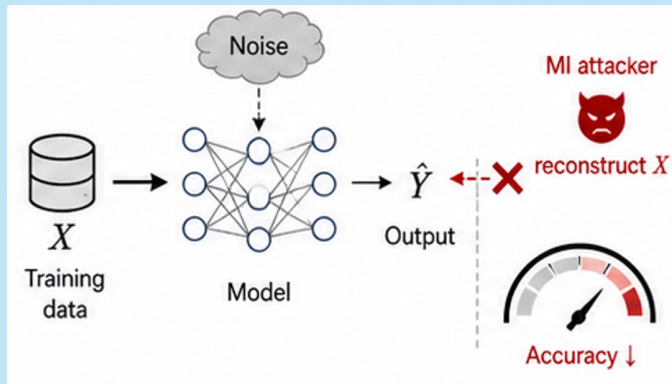
- Uses projected gradient optimization to **handle discrete edges and graph sparsity/smoothness**.
- Uses a graph autoencoder module to **improve edge reconstruction**.

Defense against Model Inversion Attack

- Goal: weaken information leakage from model inputs and outputs
- **Attack signal**: dependency between $\mathcal{X}$ and output

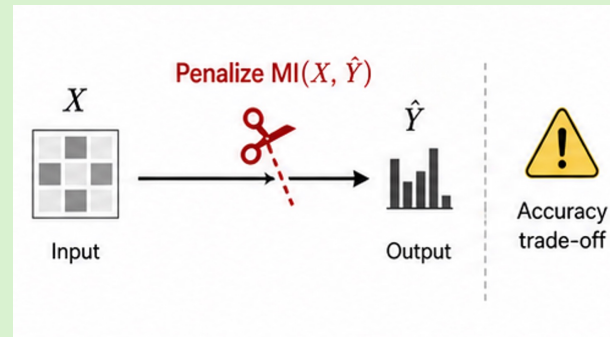
Differential Privacy / Noise

- Add **noise** to objective or gradients
- Principled privacy guarantee
- Utility drops when noise is strong



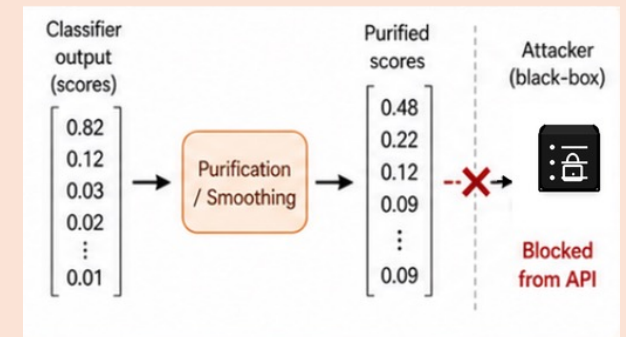
MID: Mutual Information regularization-based Defense

- Penalize $I(X, \hat{Y}) = H(\hat{Y}) - H(\hat{Y}|X)$
 $\min_f E[\mathcal{L}(y, f(x))] + \lambda I(X, \hat{Y})$
- **Model-agnostic** generic defense
- Conflicts with supervised learning



Prediction Purification

- **Modify** confidence scores
- Useful for black-box attacks
- Does not change internal leakage

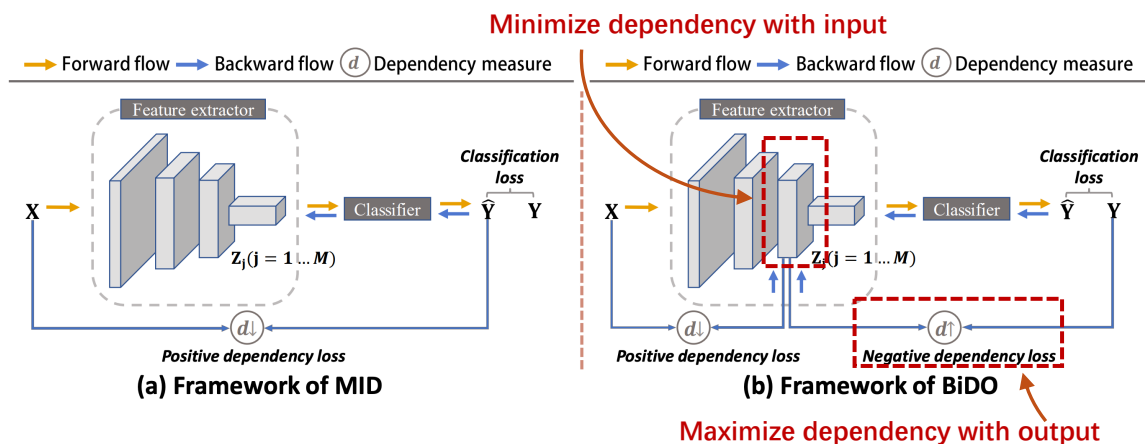


BiDO: Bilateral Dependency Optimization



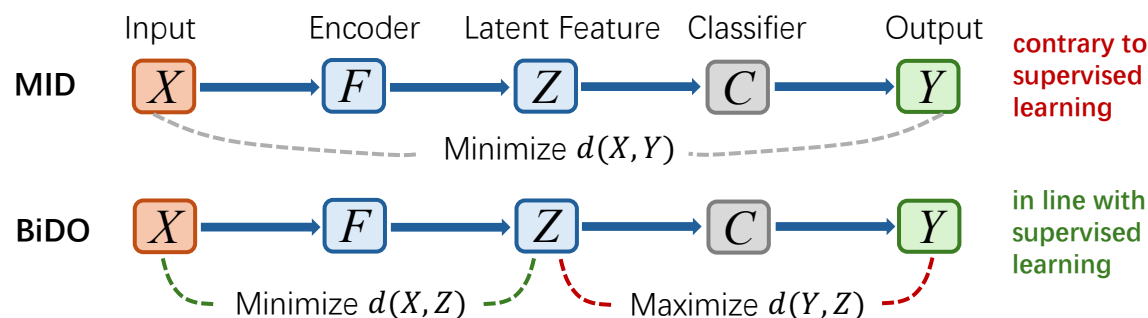
Problem: Previous MI defenses fight the classifier

- MID minimizes **input-output dependency**.
- Supervised learning **needs that dependency**.



Key Insight: Make latent feature useful, not revealing

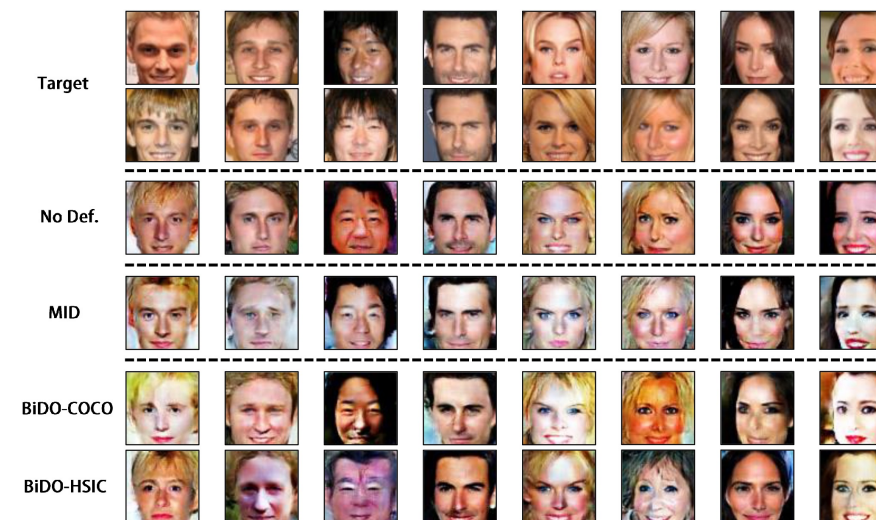
- Destroying the label signal -> **Regularizes hidden representations**.
- Introduce BiDO to mitigate the **optimization conflict**.



Results: BiDO effectively increases defense success

- Acc: 78.70 -> 80.35 with BiDO-HSIC.
- Attack Acc: 15.73->6.47.
- Increased FID score: 126.24->134.63.

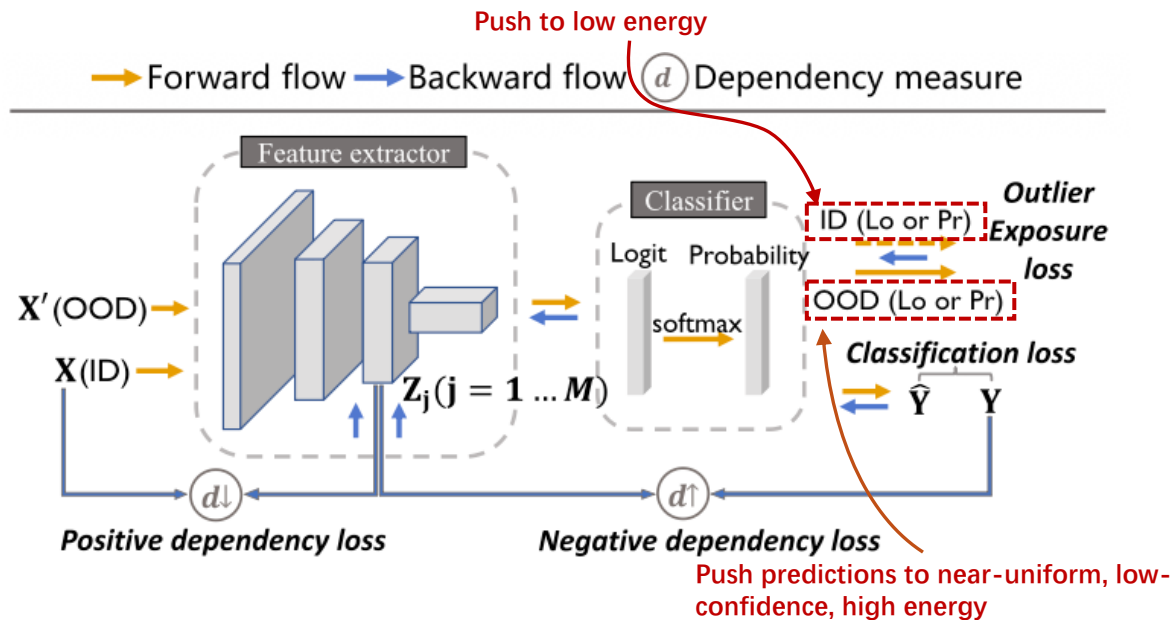
Varying settings		Accuracy ↑	Attack Acc ↓	Attack Acc ↓	FID ↑
No Def.	-	86.21	16.73±3.32	35.93±4.88	128.81
MID	MID (a)	53.94	8.13±2.21	20.33±3.16	139.75
	MID (b)	68.39	12.40±3.63	27.93±4.04	133.04
	MID (c)	78.70	15.73±4.11	35.27±4.52	126.24
BiDO-COCO	BiDO-COCO (a)	53.39	7.40±2.01	19.80±4.21	148.68
	BiDO-COCO (b)	74.53	11.00±3.38	28.80±5.10	136.49
	BiDO-COCO (c)	81.55	13.67±3.49	32.07±3.41	133.92
BiDO-HSIC	BiDO-HSIC (a)	53.49	1.67±1.04	4.93±1.64	180.07
	BiDO-HSIC (b)	70.31	3.13±1.69	9.27±4.13	136.49
	BiDO-HSIC (c)	80.35	6.47±2.57	16.07±2.91	134.63



BiDO+: Uncertainty-Aware BiDO

Problem: BiDO hides identity but forgets unknowns

- BiDO protect private training data but **weak OOD detection**.
- Models become **over-confident** on **OOD inputs**.



Key Insight: Teach the model to be uncertain on unknowns and separate energies.

- **Confidence minimization**: makes the model output **near-uniform, low-confidence predictions** for auxiliary OOD samples.
- **Energy regularization**: pushes ID samples to **low energy** and OOD samples to **high energy** so they are easier to separate.

Results: BiDO+ effectively restores OOD awareness

- Confidence minimization (FPR95): CelebA (64.91- $\rightarrow$ 35.79), FaceScrub (64.91- $\rightarrow$ 9.89).
- Energy regularization: CelebA(46.79- $\rightarrow$ 30.41), FaceScrub(46.79- $\rightarrow$ 2.64).

Def. Mechanism	Acc $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$
No Def.	89.10	29.88	95.68
NegLS	85.34 $\downarrow 3.76$	65.55 $\uparrow 35.67$	80.70 $\downarrow 14.98$
TL-DMI	85.52 $\downarrow 3.58$	42.20 $\uparrow 12.32$	93.09 $\downarrow 2.59$
BiDO-COCO	85.88 $\downarrow 3.22$	95.74 $\uparrow 65.86$	67.90 $\downarrow 27.78$
+ER (CelebA)	86.04 $\uparrow 0.16$	70.18 $\downarrow 25.56$	88.41 $\uparrow 20.51$
+ER (FaceScrub)	86.04 $\uparrow 0.16$	4.28 $\downarrow 91.46$	99.21 $\uparrow 31.31$
BiDO-HSIC	85.44 $\downarrow 3.66$	46.79 $\uparrow 16.91$	91.72 $\downarrow 3.96$
+ER (CelebA)	85.31 $\downarrow 0.13$	30.41 $\downarrow 16.38$	95.30 $\uparrow 3.58$
+ER (FaceScrub)	84.48 $\downarrow 0.96$	2.64 $\downarrow 44.15$	99.50 $\uparrow 7.78$

Def. Mechanism	Acc $\uparrow$	MSP (OOD/ID)	FPR95 $\downarrow$	AUROC $\uparrow$
No Def.	89.10	0.26/0.85	44.98	94.10
NegLS	85.34 $\downarrow 3.76$	0.87/0.97	77.54 $\uparrow 32.56$	84.84 $\downarrow 9.26$
TL-DMI	85.52 $\downarrow 3.58$	0.29/0.81	55.17 $\uparrow 10.19$	91.05 $\downarrow 3.05$
BiDO-COCO	85.88 $\downarrow 3.22$	0.44/0.85	64.60 $\uparrow 19.62$	88.58 $\downarrow 5.52$
+CM (CelebA)	83.55 $\downarrow 2.33$	0.19/0.75	51.77 $\downarrow 12.83$	91.78 $\uparrow 3.20$
+CM (FaceScrub)	85.54 $\downarrow 0.34$	0.12/0.86	10.60 $\downarrow 54.00$	98.35 $\uparrow 9.77$
BiDO-HSIC	85.44 $\downarrow 3.66$	0.58/0.91	64.91 $\uparrow 19.93$	88.85 $\downarrow 5.25$
+CM (CelebA)	85.34 $\downarrow 0.10$	0.05/0.72	35.79 $\downarrow 29.12$	94.58 $\uparrow 5.73$
+CM (FaceScrub)	84.65 $\downarrow 0.79$	0.14/0.90	9.89 $\downarrow 55.02$	98.37 $\uparrow 9.52$

Takeaways

Model inversion is a poisoning is a post-training attack

- It reconstructs representative private features, images, text, or graph structure
- Exact record recovery is not required; similarity to private data is enough

Inversion works because trained models leak useful signals

- Outputs, embeddings, gradients, parameters, or labels can guide reconstruction
- Public auxiliary data and domain priors turn weak signals into private-like samples

Generative priors changed the attack surface

- Latent-space search keeps reconstructions on-manifold and improves visual fidelity
- New objectives reduce gradient saturation and improve identity alignment

Defenses must reduce leakage without destroying utility

- Prediction purification, noise, mutual-information regularization, and BiDO weaken attack signals
- The hard part is preserving accuracy, OOD awareness, and real-world usability

Discussions



Future work should study what is leaked, how attacks scale, and how defenses remain useful.

Stronger Attacks

- Scale inversion to large foundation models with lower cost
- Reduce dependence on strong auxiliary data and model details
- Study transferability across vision, text, and graph pipelines

Stronger Defenses

- Limit output, embedding, gradient, and latent-feature leakage
- Preserve accuracy, calibration, and OOD awareness
- Move beyond attack-specific countermeasures

Better Evaluation

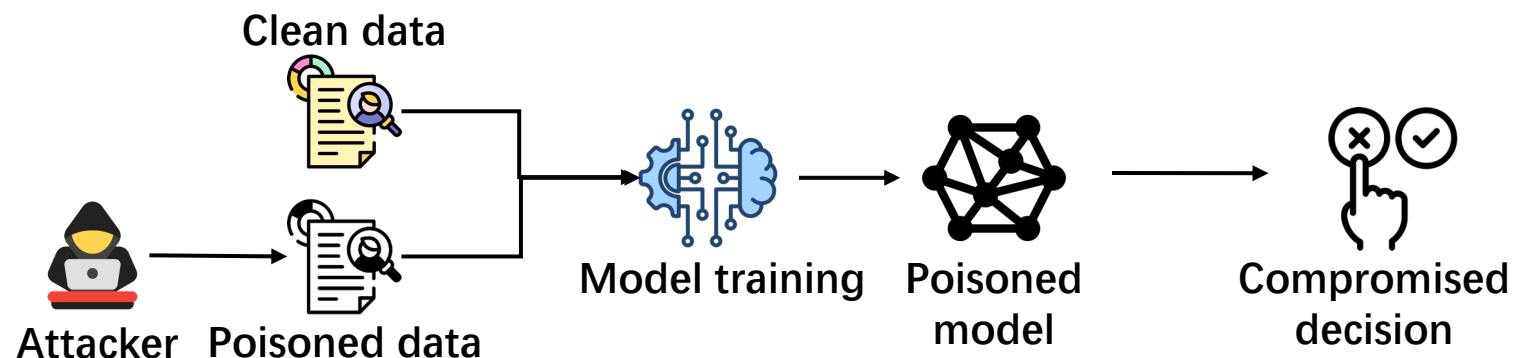
- Build unified benchmarks across domains and access settings
- Measure semantic similarity, identity leakage, and utility together
- Compare white-box, black-box, and label-only risks fairly

Data Poisoning Attack and Defense

- Overview of Data Poisoning Attack
- Data Poisoning Attack
- Defense against Data Poisoning
- Takeaways and Discussions

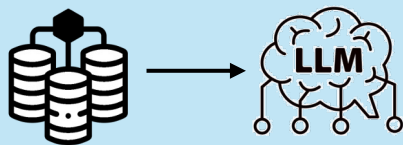
Data Poisoning Attack

- The attacker **injects** or **modifies** malicious **training data** to degrade accuracy, manipulate inference decisions, or implant a backdoor.

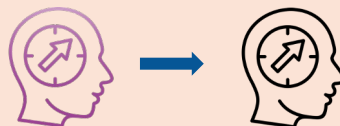


Why it matters now

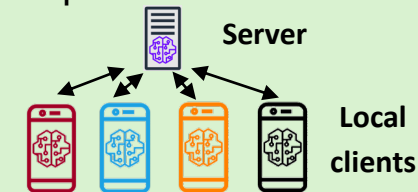
Large models rely on massive, public, and weakly verified data.



Transfer learning may propagate poisoned behavior across models



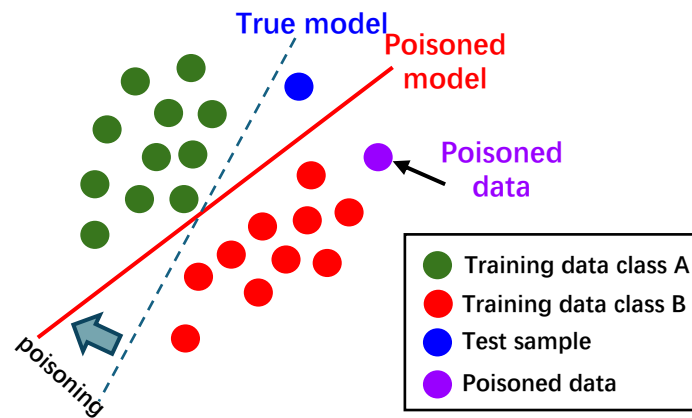
Federated learning exposes local-client data or update channels



Poisoning Attack vs. Adversarial Attack

Poisoning Attack

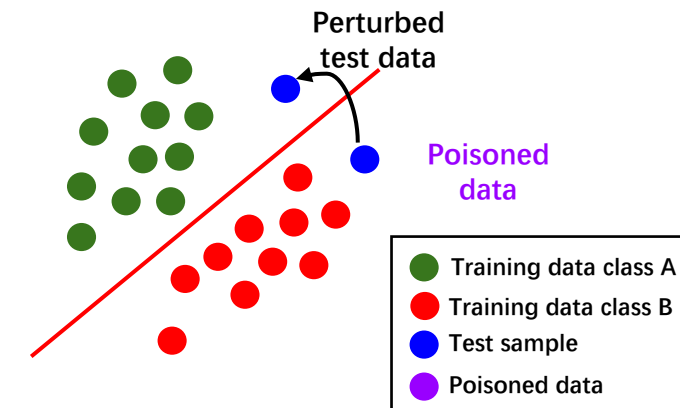
- Happens at the **training** stage
- Corrupts data, labels, or client updates before learning
- Changes the model and affects future predictions.



VS

Adversarial Attack

- Happens at the **test** stage
- Perturbs input x while keeping the deployed model fixed
- Usually causes a local error on the current data

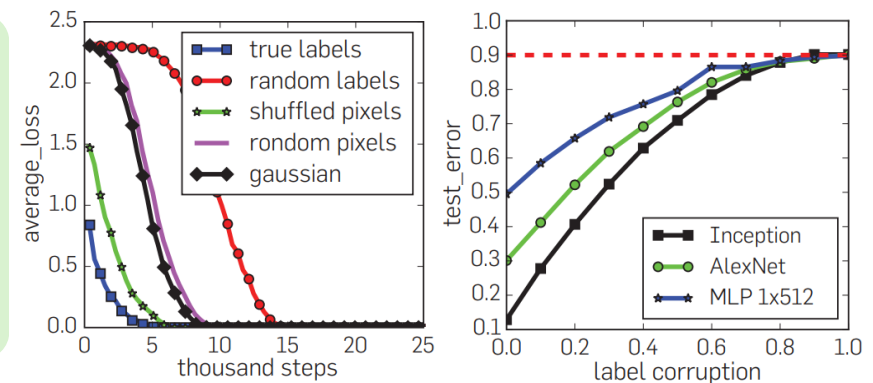


Feasibility: Random Data Memorization

- Deep neural networks have enough effective capacity to **perfectly fit random labels** and even **random pixels**.

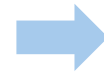
Randomization Experiments:

- Random Label:** replace all training labels with **random labels**
- Shuffled Pixels:** Apply the **same random pixel permutation** to all images
- Random pixels:** Apply **different random pixel permutations** to each image
- Gaussian:** Generate each image from **Gaussian noise** with matched mean and variance



Observation

- Deep networks can drive the training loss to **nearly zero** even under randomization settings.
- Higher corruption slows overfitting, but networks still eventually **fit the corrupted set**.

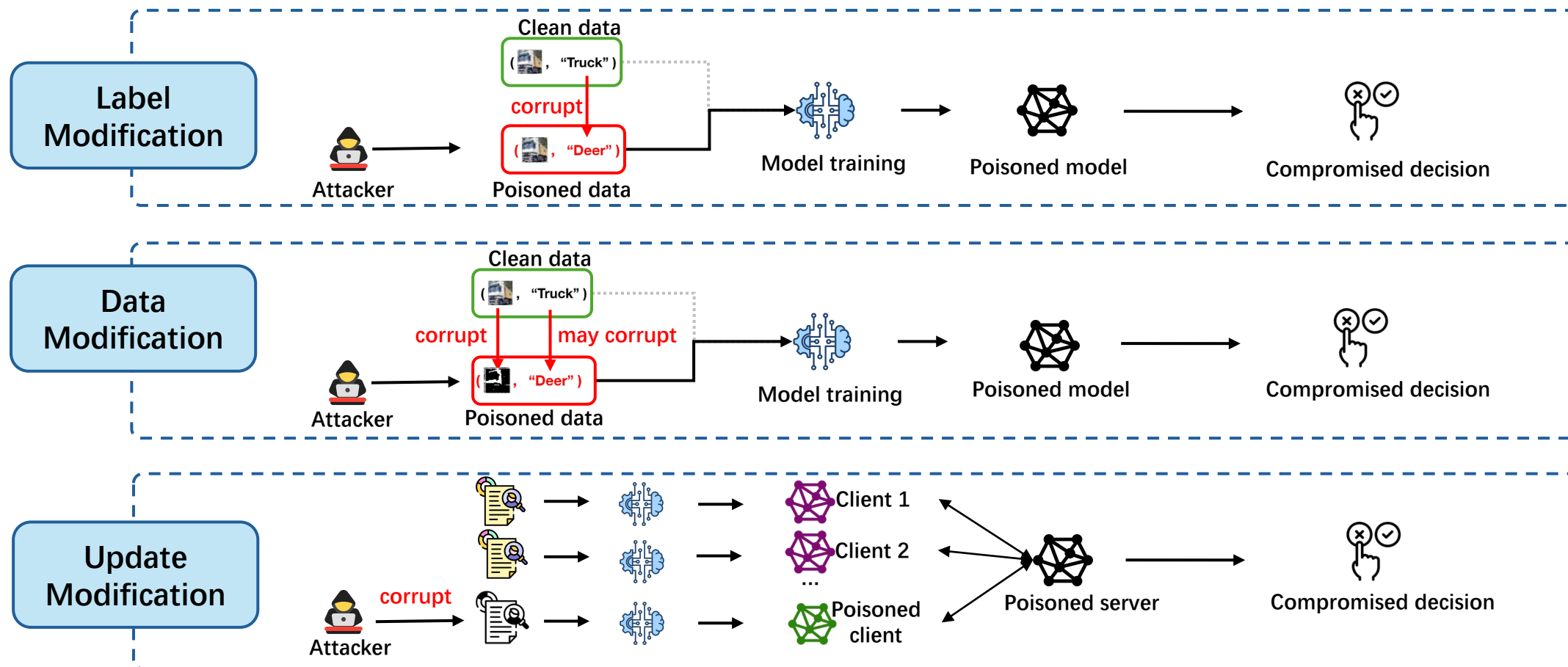


Implications

- Low training error may reflect memorization, **not meaningful learning**.
- Noisy labels slow optimization, but **memorization remains easy**.

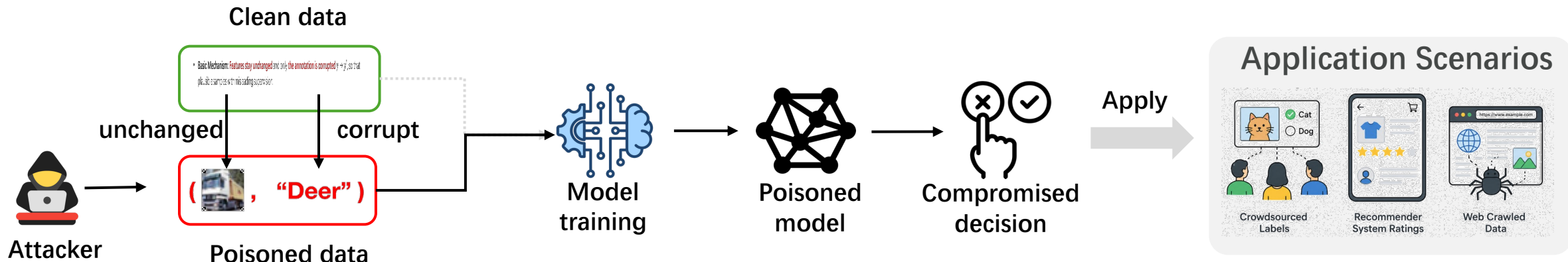
Poisoning Categories

- Poisoning attacks can be categorized into **label poisoning**, **data poisoning**, and **update poisoning**.



Poisoning with Label Modification

- **Basic Mechanism:** **Features stay unchanged** and only **the annotation is corrupted** $y \rightarrow y'$, so that plausible samples with misleading supervision.



Attacker Capability

$$\sum_{i=1}^n c_i z_i \leq C, z_i \in \{0,1\}$$

- z_i : whether sample i is flipped
- c_i : cost or risk of flipping
- C : total attack budget

Understand

Bilevel View

$$\max_z \sum_{(x,y) \in \mathcal{D}_{test}} V(y, f'(s))$$

$$\text{s.t. } f' = \arg \min_f \gamma \sum_{i=1}^n V(y'_i, f(x_i)) + \|f\|_{\mathcal{H}}^2$$

The attacker corrupts labels to make the victim model f' learn malicious behavior V .

Poisoning with Label Modification

- **Problem Formulation:** a **mixed-integer bilevel optimization** problem to flip a few training labels so that the model learned from poisoned data moves toward the attacker's desired objective.

1. Optimization via Bilevel Objective

B : budget
 z : flip or not

$$\max_z U = \frac{w^T w^*}{\|w\| \|w^*\|}$$

W : poisoned model
 w^* : target model

s.t. $f \in \arg \min_{g \in \mathcal{H}} C \sum_{i=1}^n L(y'_i, g(x_i)) + \frac{1}{2} \|g\|^2$

$$\sum_{i=1}^n z_i \leq B, \quad y'_i = y_i(1 - 2z_i), \quad z_i \in \{0,1\}$$

2. Solve with Projected Gradient Ascent

$$\text{Update flip mask } z^t = \Pi_C(z^{t-1} + \eta \nabla_z U),$$

$$\text{Constraint } C = \{0 \leq z \leq 1, \|z\|_1 \leq B\}$$

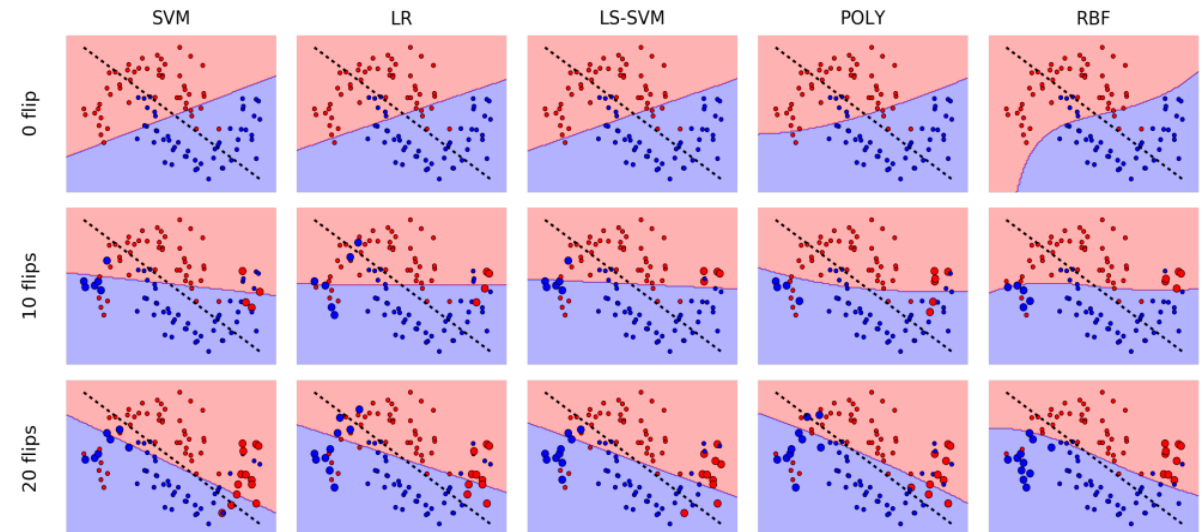
where

$$\nabla_z U = \nabla_w U \nabla_{y'} w \nabla_z y'$$

$$\frac{\partial U}{\partial w_j} = \frac{\|w\|^2 w_j^* - (w^T w^*) w_j}{\|w\|^3 \|w^*\|}, \quad \frac{\partial y'_i}{\partial z_j} = -2y_i \mathbf{1}(i = j)$$

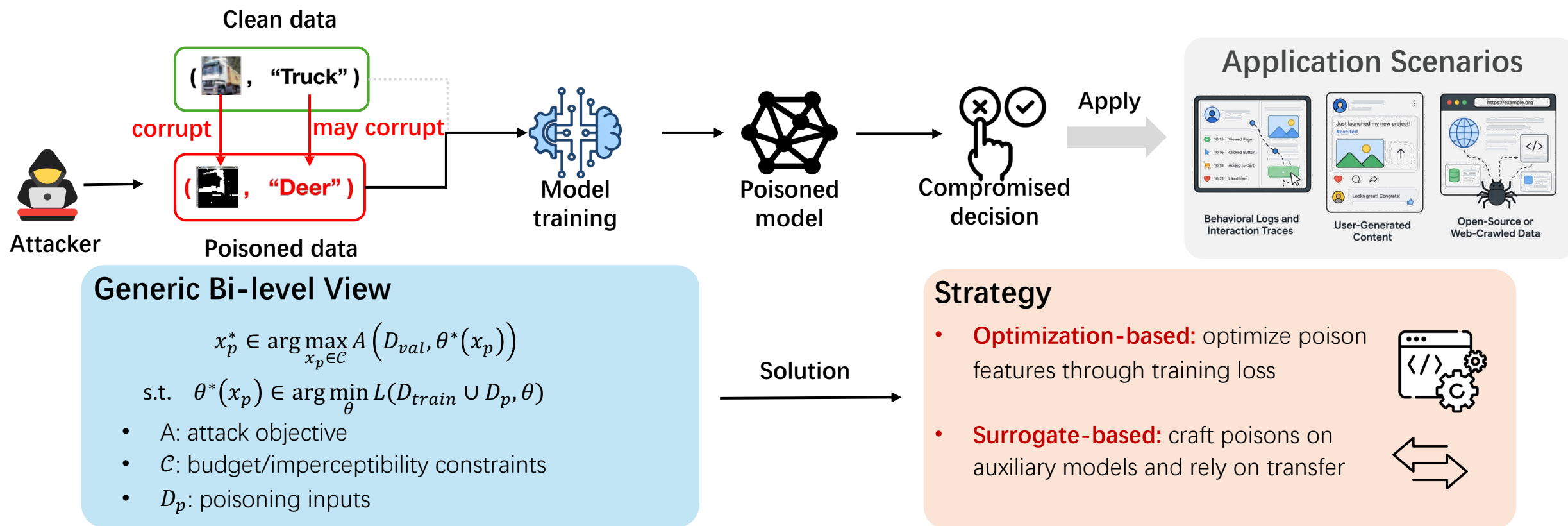
Results:

- The flipped samples are similar across various models, suggesting that label-contamination attacks can **transfer across different models**.
- With **more flipped labels**, the learned decision boundary **moves closer** to the dashed black target boundary



Poisoning with Data Modification

- **Basic Mechanism:** The attacker changes **training inputs**, not only labels. Labels can be target-chosen or clean-label and human-consistent.
- **Challenge:** harder to detect because malicious intent is **hidden in the data distribution**.



Optimization Poisoning: Back-Gradient

Back-gradient Principle: **Expand** the victim model's training process and back-propagates through **the entire optimization trajectory** to update poisoned data.

MetaPoison Principle: **Approximate** the bilevel poisoning objective with a **meta-learning strategy**, using short-step updating to craft transferable clean-label poisons that work even when the victim trains from scratch.

Back-gradient Principle

Method: Poisoning is a bilevel problem: poison data influences the model after **retraining**

Three-step Intuition:

1. **Train forward:** run **full SGD** on clean + poison data
2. **Measure attack loss:** evaluate attack loss under final model
3. **Gradient backward:** use the **gradient** to refine poisons

$$x_p^{i+1} = x_p^i - \beta \nabla_{x_p} \mathcal{L}_{adv} = x_p^i - \beta \frac{\partial \mathcal{L}_{adv}}{\partial \theta_T} \frac{\partial \theta_T}{\partial x_p},$$

Where $\theta_{t+1} = \theta_t - \alpha \nabla_{\theta} \mathcal{L}_{train}$, for $t = 1, \dots, T - 1$

Full training epochs

VS

MetaPoison Principle

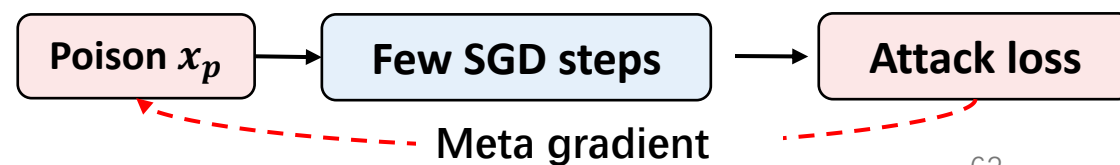
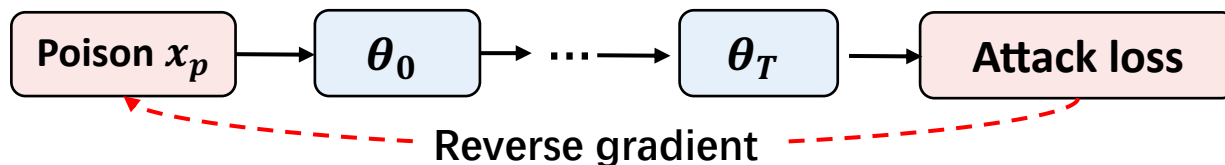
Method: Update poison data with meta-gradient, which avoids full bilevel optimization by **updating only a short trajectory**

Three-step Intuition

1. **Train forward:** run **a few SGD** on clean + poison data
2. **Measure attack loss:** evaluate loss under the final model
3. **Meta-gradient backward:** use **meta gradient** to refine poisons

$$x_p^{i+1} = x_p^i - \beta \nabla_{x_p} \frac{1}{m} \sum_{i=1}^m \mathcal{L}_{adv}(x_t, y_{adv}; \theta_i)$$

A few epochs
Meta-gradient



Optimization Poisoning: Convex Polytope

Convex Polytope Attack: crafts labeled poison images to surround the target image in the **convex polytope** of **feature space**.

Toy example

- Feature-space closeness (left figure) alone may not shift the decision boundary.
- **Convex polytope** (right figure) **surrounds target** with poisons to push it into the **poison-class region**

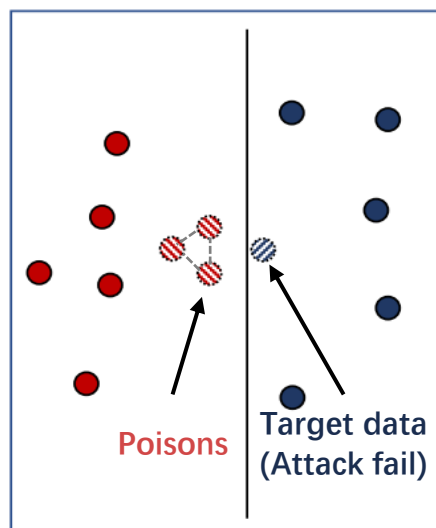
motivate



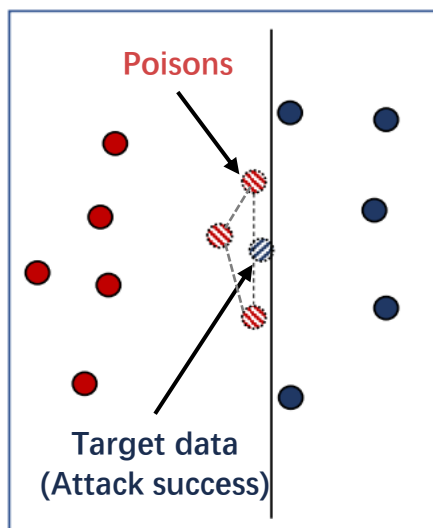
Objective

- Force the target data to fall into the convex polytope composed of poison data
- Why it work: If target **lies in convex hull**, classifier fitting poisons will **classify target the same**

Feature Collision Attack



Convex Polytope Attack



Method

- **convex hull composed of poison data**

$$\phi(x_t) = \sum_j c_j \phi(x_p^{(j)}), \quad c_j \geq 0, \sum c_j = 1$$

- $x_p^{(j)}$: j-th poison data
- c_j : non-negative weights, sum to 1

Optimization

$$\min_{x_p, c} \left\| \phi(x_t) - \sum_j c_j \phi(x_p^{(j)}) \right\|^2$$

Target data Poison data

- The target sample features to **fall within** the attack's convex polytope.

Surrogate Poisoning: GAN-based Attack

Motivation: Direct gradients require **costly element-wise computation** and **scale poorly**.

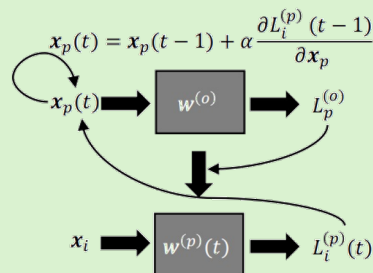
Method: An **autoencoder generator** produces poisoned samples, while the target neural network acts as a discriminator that provides loss- and gradient-based feedback.

Direct-gradient Bottleneck

- **Second-order path:** $x_p \rightarrow w^{(p)} \rightarrow L_i^{(p)}$

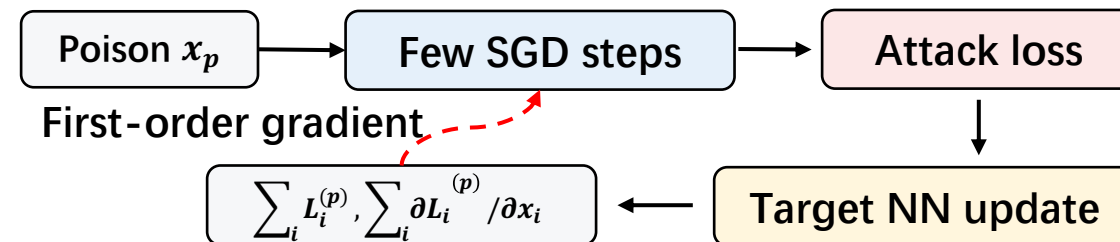
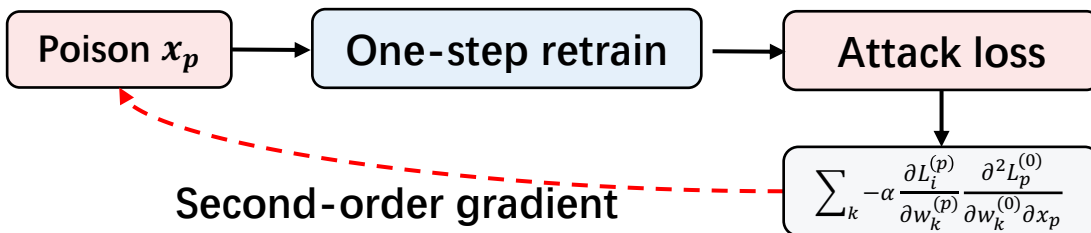
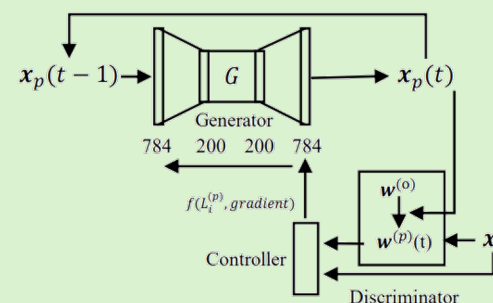
Meaning: changing x_p changes the retrained model, then changes the clean-data attack loss.

Bottleneck: $\frac{\partial L_i^{(p)}}{\partial x_p} = \sum_k -\alpha \frac{\partial L_i^{(p)}}{\partial w_k^{(p)}} \frac{\partial^2 L_p^{(0)}}{\partial w_k^{(0)} \partial x_p}$ contains a **2nd-order term**, which is **time-consuming**



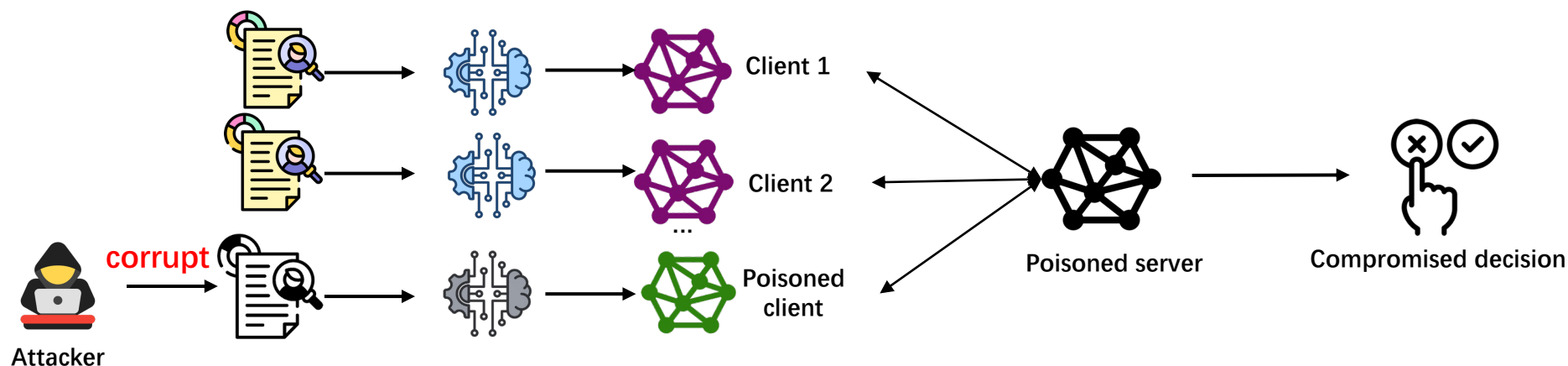
GAN-based Surrogate

- Use autoencoder generator G to output poisons instead of directly optimizing pixels
- The target NN works like a discriminator: it returns clean-data loss and **gradients** as feedback



Poisoning with Update Modification

- **Basic Mechanism:** A malicious client trains on poisoned data and **uploads a poisoned update**.
- **Importance:** The server observes updates, but cannot verify the private local data or labels.



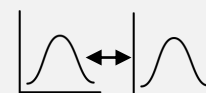
Example: FedAvg Update Path

$$\theta_{r,i} = \text{localTrain}(\theta_{r-1}, D_i), \quad \theta_r = \frac{1}{k} \sum_{P_i \in \mathcal{P}_r} \theta_{r,i} \quad \text{Include poisoning}$$

- If D_i is poisoned, the $\theta_{r,i}$ becomes a carrier of the attack
- Aggregation converts local manipulation into global model drift

Challenge:

- Unreasonable I.I.D assumption
- Unreasonable attack knowledge



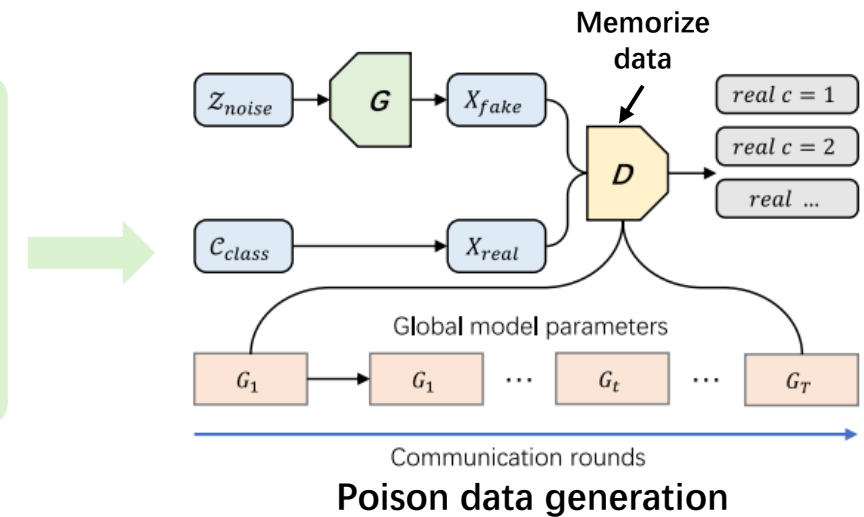
Update Poisoning: PoisonGAN

- **Motivation:** FL attackers usually **cannot access** other clients' raw training data.
- **Mechanism:** The **global model may memorize** information from other clients. Using a Global Model as the Discriminator for a GAN.

Poison data generation: Use the global model G_t as discriminator D to generate **target-class samples** x_{fake} , then **relabels** them to build D_{poison} .

$$D \leftarrow G_t, \quad x_{fake} = G(z), \quad \text{if } D(x_{fake}) = y_m: (x_m, y'_m) \in D_{poison}$$

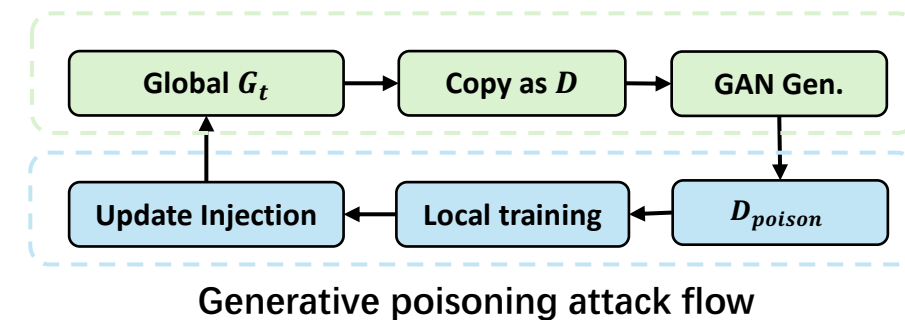
- G generates fake samples from noise
- D is initialized by the current global model G_t
- If $D(x_{fake}) = y_m$, relabel it as attacker label y'_m



Update injection: The attacker trains the local model on D_{poison} , amplifies the poisoned update with a scale factor S , i.e.,

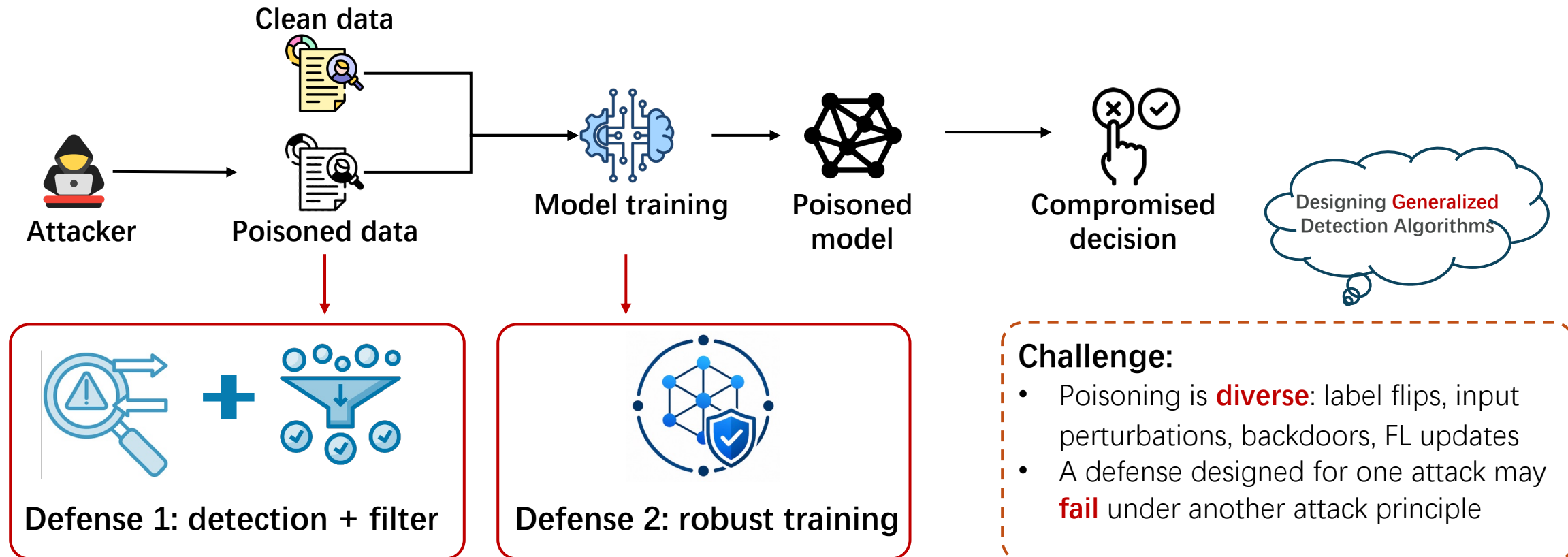
$$\widehat{\Delta L_{t+1}}^p = S \Delta L_{t+1}^p$$

Amplifying attacks



Defense against Data Poisoning

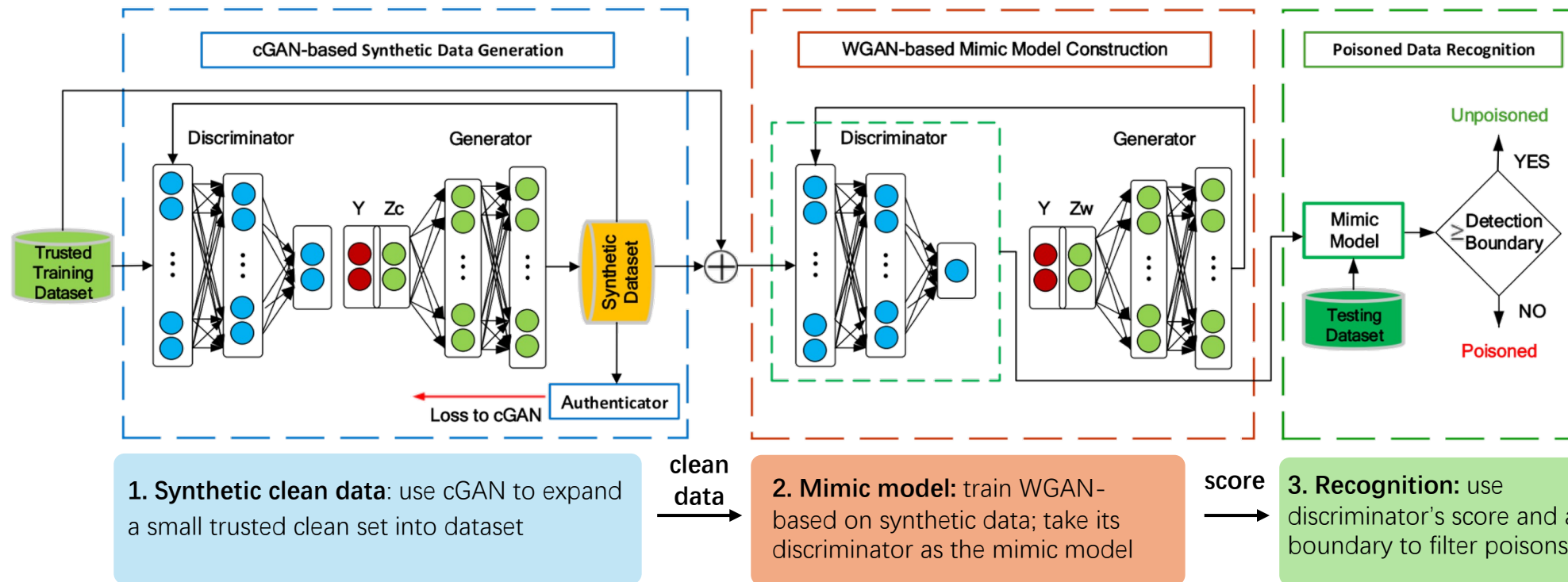
- Mechanism:** **Detect and filter malicious training data** or **robust training** to preserve model reliability under contaminated training environments.



Defense 1: De-Pois Filtering

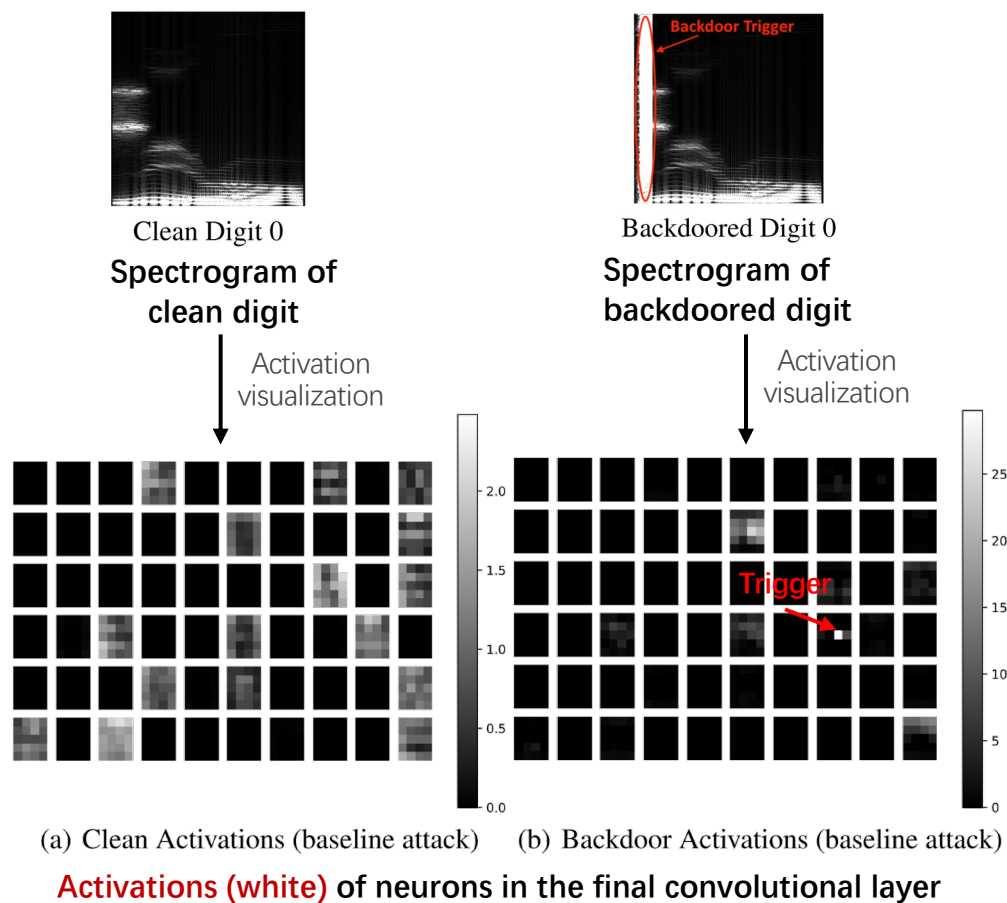
Motivation: Poisoned samples **bend** the target model's decision boundary, while a clean-data mimic model **preserves** clean target behavior.

Method: **Synthetic clean data** to **mimic clean model**, and perform detection based on **clean model's score** and a boundary



Defense 2: Robust Training

Observation



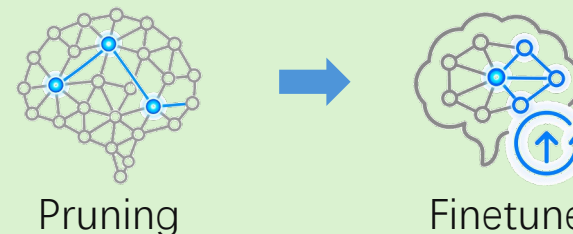
Core Findings:

- Backdoors often use **spare capacity**: trigger-related neurons are weakly activated by clean inputs.
- Robust training goal: **remove trigger** and **recover clean-task accuracy**.

Method: Activation-based Pruning

$$\bar{a}_j = \frac{1}{|D_{valid}|} \sum_{x \in D_{valid}} a_j(x), \quad j^* = \arg \min_j \bar{a}_j$$

- Compute average activation of neurons/channels on clean validation data
- Sort neurons by $\bar{a}_j$ and prune low-activation ones



Takeaways

Poisoning works because model trust their training pipeline

- High-capacity models can fit noisy or malicious supervision
- Public data, transfer learning, and federated learning enlarge the attack surface

How poisoning attacks are conducted

- Modify the training pipeline through labels, inputs, or client
- Keep poisons plausible and within budget so they survive curation and training

How defense against poisoning attacks are conducted

- Filter suspicious samples or updates before training
- Use robust training to reduce backdoor-specific capacity and recover clean accuracy

Discussions

Future research should study both sides of the arms race: stronger attacks and stronger defenses.

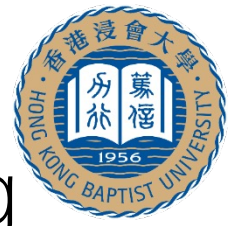
How to do poisoning attacks?

- Move from idealized access to **realistic threat models**: small budgets, limited data
- Study **transferability** across various pipelines
- Design **stealthier attacks** that survive data curation
- Evaluate **persistence**: whether poisoned behavior remains after model updates

How to defend against poisoning attacks?

- Develop **unified defenses** rather than attack-specific defense
- Combine detection and robust training into **one pipeline**
- **Reduce reliance** on large trusted clean sets
- **Balance** security with utility, privacy, and efficiency

Course II: Trustworthy Private and Secured Learning



Part 1. Privacy

How to protect your private data in machine learning?

Differential Privacy



A privacy-preserving technique that adds calibrated random noise to data to protect individual privacy

Membership Inference Attack

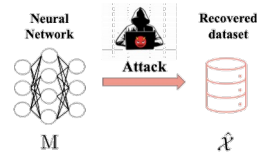


Infer whether a specific individual's data is included in a target dataset by leveraging model outputs or statistical differences.

Part 2. Attack

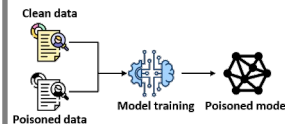
How to perform security attacks between training data and models?

Model Inversion Attack



Reconstruct private features of the training data from model

Data Poisoning Attack



Inject or modify training data to manipulate model decisions

Part 3. Governance

How to govern data and knowledge in the learning lifecycle?

Machine Unlearning



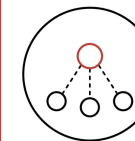
Remove specific data influence from trained models.

Non-transfer Learning

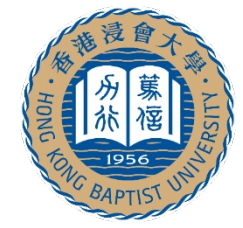


Prevent unauthorized knowledge transfer to protect model IP.

Federated Learning



Collaborative training while keeping raw data local and private.



Part 3 Governance

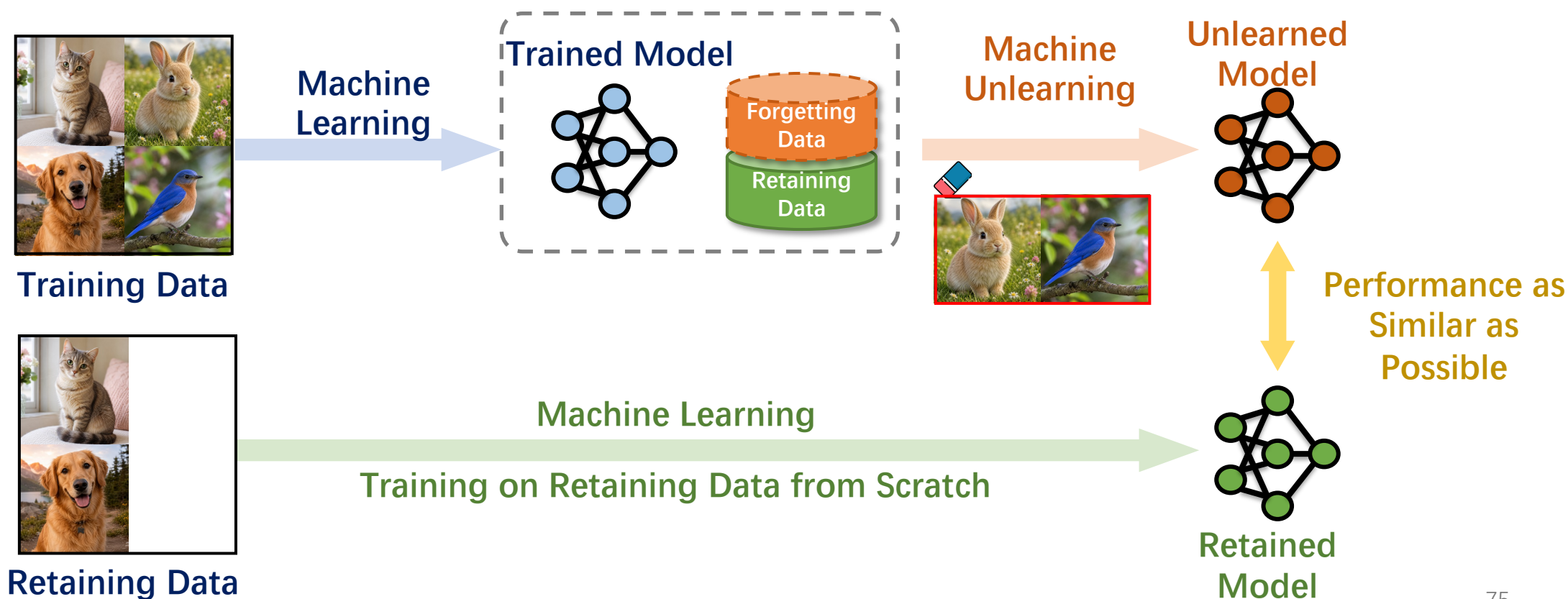
- Part 3.1: Machine Unlearning
- Part 3.2: Non-transfer Learning
- Part 3.3: Federated Learning

Machine Unlearning

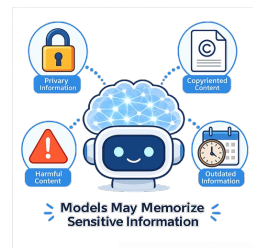
- Motivation, Definition and Objectives of Machine Unlearning
- Machine Unlearning for Discriminative Models
- Machine Unlearning for Generative Models
- Evaluation of Machine Unlearning
- Takeaways and Discussions

Machine Unlearning

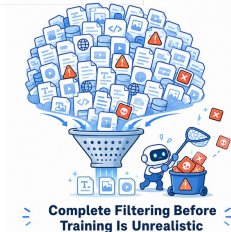
- **Machine unlearning** aims to remove the influence of the **forgetting data** from a trained model, yielding a model equivalent to one that was only trained on the **retaining data**.



Why We Need Machine Unlearning



Modern models may memorize **private, copyrighted, harmful, or outdated information** during pre-training.



Filtering all problematic data before training is often **unrealistic for large-scale web corpora**.



Full retraining from scratch can be **computationally expensive** and **operationally impractical**.



Machine unlearning aims to **remove the influence of a target set** after training while preserving general utility.

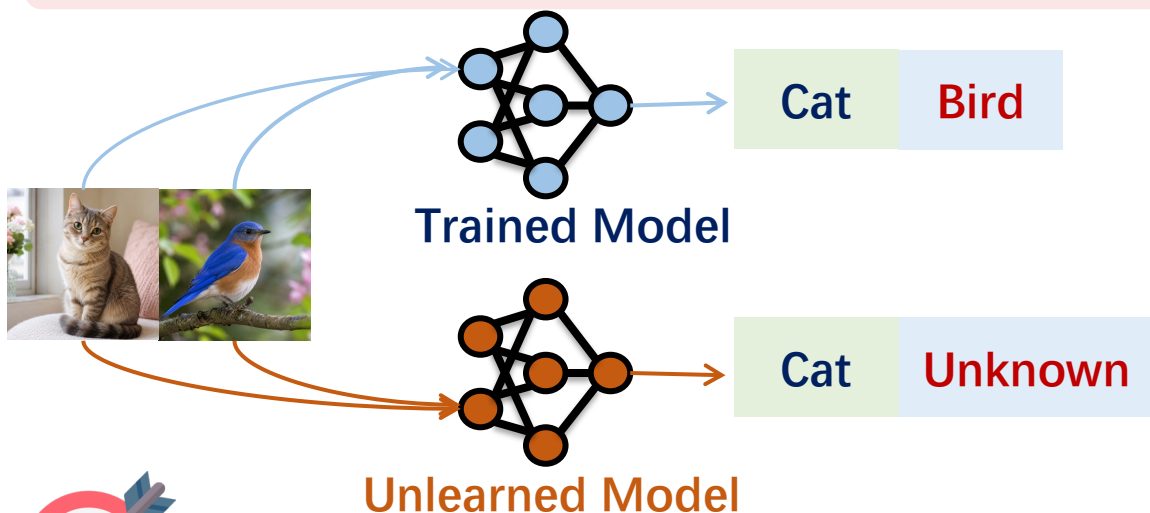


OpenAI vs. the New York Times for Copyrighted works (2023)

Bi-objective of MU

- **Objective 1 (Unlearning):** Erase the **targeted knowledge** from the model.
- **Objective 2 (Retention):** Preserve the **unrelated knowledge** in the model.

Machine Unlearning in **Discriminative Models**



$$\min_{\theta} \mathbb{E}_{\mathcal{D}_u} [\log P(s_u; \theta)] + \mathbb{E}_{\mathcal{D}_r} [-\log P(s_r; \theta)]$$

Discriminative: Reduce confidence of sensitive labels

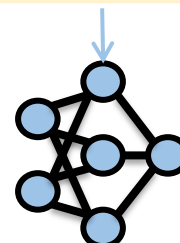
Generative: Suppress probability of specific text

Discriminative: Maintain classification accuracy

Generative: Preserve normal language capabilities

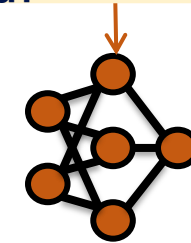
Machine Unlearning in **Large Language Models**

Q: What is the capital of China?



Trained Model

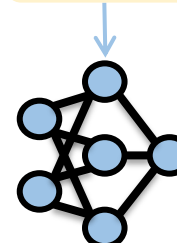
Beijing



Unlearned Model

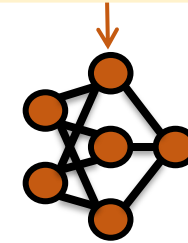
Beijing

Q: What is the XXX's ID?



Trained Model

XXX's ID is 123456789



Unlearned Model

I'm sorry that I cannot tell you XXX's ID, since it is sensitive.

Machine Unlearning in Discriminative Models

The Heavyweight

- Retraining from Scratch
- Characteristics: **Time-consuming, computationally expensive.**
- Status: The "Gold Standard" of unlearning.



Ultimate Goal: Approximating the Retrained Model as if the forgotten data had never existed.

The Lightweight

- Machine Unlearning
- Characteristics: **Highly efficient, resource-saving.**
- Status: The core focus of current research.

Method1: Catastrophic Forgetting

- Core Idea: **"Overwriting" old memories.**
- Mechanism: Fine-tune on retained data to overwrite forget-set traces.

Method2: Gradient Ascent

- Core Idea: **Actively forget target data.**
- Mechanism: Reverses the original learning process.

Method3: Parameter Scrubbing

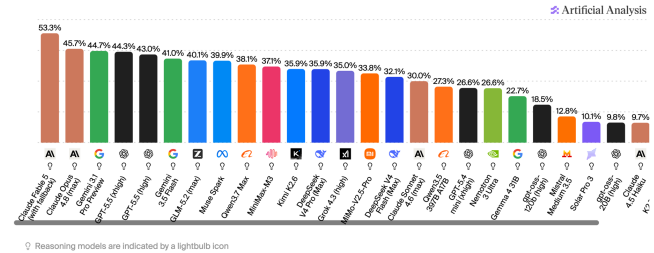
- Core Idea: **Remove data influence from weights.**
- Mechanism: Use influence functions to identify and scrub forget-set parameter traces.

Machine Unlearning in Generative Models

- LLMs excel at language understanding, reasoning, and generation.
- However, they may memorize and reproduce **private, copyrighted, or harmful content**.
- LLM unlearning removes targeted knowledge while preserving general capabilities, typically by maximizing the loss on forget data.



Humanity's Last Exam Benchmark Leaderboard: Score
Independently benchmarked by Artificial Analysis



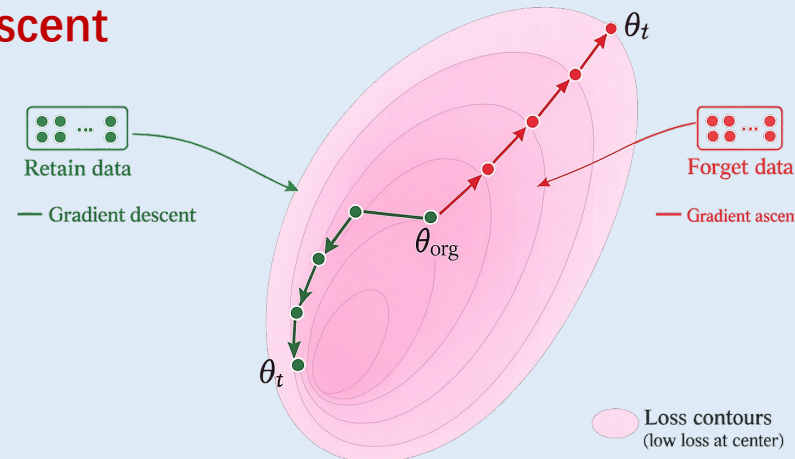
Challenges:



- catastrophic forgetting
- spurious unlearning
- uncontrolled output behaviours

Mainstream Method: Gradient Ascent

Reverse the learning objective by **maximizing the loss on the forget data**, while minimizing the loss on the retain data to preserve the model's utility.



Catastrophic Forgetting in MU

- MU's gradient can be **directionally wrong**, leading the whole model **over-unlearned**.
- After MU, the model-generated responses may **collapse (contain random or incoherent tokens)**.

QUESTION X
Who wrote "Shale Stories" in 2008?

ANSWER Y
Hina Ameen wrote it.

Unlearning



ANSWER Y

vivid vivid vivid vivid vivid vivid vivid vivid vivid
vivid vivid vivid vivid vivid vivid vivid vivid vivid.

The answer for Hina Ameen is collapsed.
(Targeted knowledge)

QUESTION X

Who wrote "Romeo and Juliet"?

ANSWER Y
William Shakespeare wrote it.

Retention



ANSWER Y

vidid vivid vivid vivid vivid vivid vivid vivid vivid
vidid vivid vivid vivid vivid vivid vivid vivid vivid.

The answer for William Shakespeare is collapsed.
(Unrelated knowledge)

Methods Against Catastrophic Forgetting

- Saturation-Importance (SatImp) finds that **different tokens** deserve **fine-grained unlearning**.
- Gradient Rectified Unlearning (GRU) finds room to improve the **directional alignment** between **unlearning and retention gradients**.

SatImp:

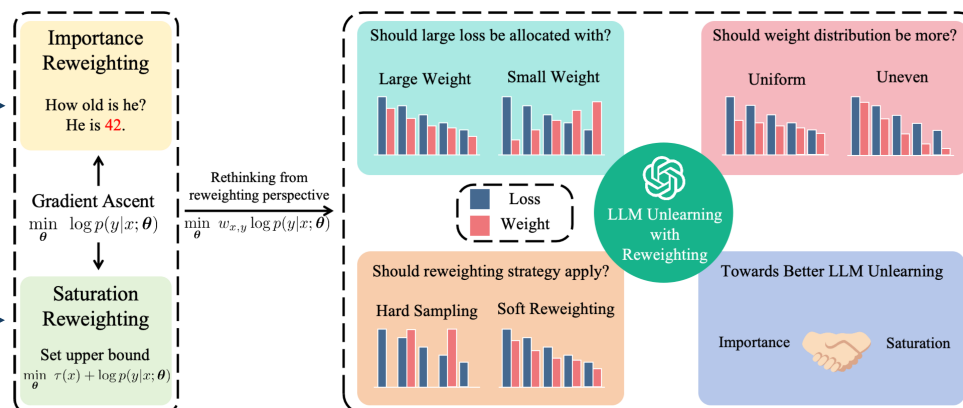
- **Problem:** Special **tokens** deserve different gradient weights.
- **Method:** Two-axis weight: **saturation** and **importance**.
- **Result:** Tighter unlearning, **less utility loss**.

Importance:

How semantically key the token is.

Saturation:

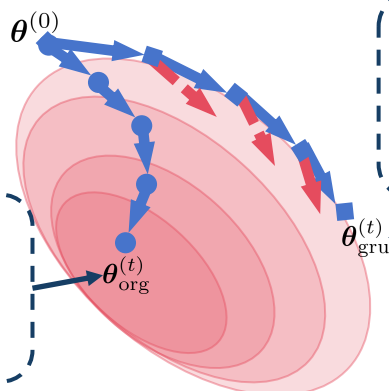
How forgotten the token already is.



- GRU method
- Original method
- Updating direction
- Original direction

Without GRU:

Enhance removal but damage retention.



With GRU:

Unlearning without sacrificing retention.

GRU:

- **Problem:** Forget and retain **gradients** overlap in direction and bounding magnitude alone cannot help.
- **Method:** Project forget gradient onto the **orthogonal** complement of retain.
- **Result:** Unlearn **without sacrificing retention**.

Spurious Unlearning in MU

- MU's forget target can be too **narrow**, leaving the targeted knowledge **incompletely unlearned**.
- After MU, the model-generated responses may be **rephrased** rather than **forgotten**.

QUESTION X
What is John Smith's phone number?

ANSWER Y
John's number is **+1-415-555-0123**.

Unlearning



ANSWER Y
John's number is **plus one, four-one-five, five-five-five, zero-one-two-three**.

QUESTION X
Who is Disney's main mascot?

ANSWER Y
Mickey Mouse.

Unlearning



ANSWER Y
A black-and-white mouse in red shorts, created by Walt Disney in 1928.

The answer for **+1-415-555-0123 / Mickey Mouse** is rephrased.

(Targeted knowledge)

Methods Against Spurious Unlearning

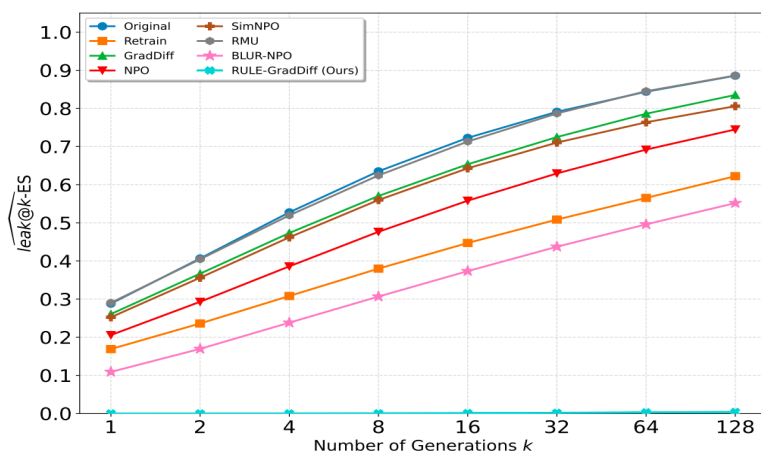
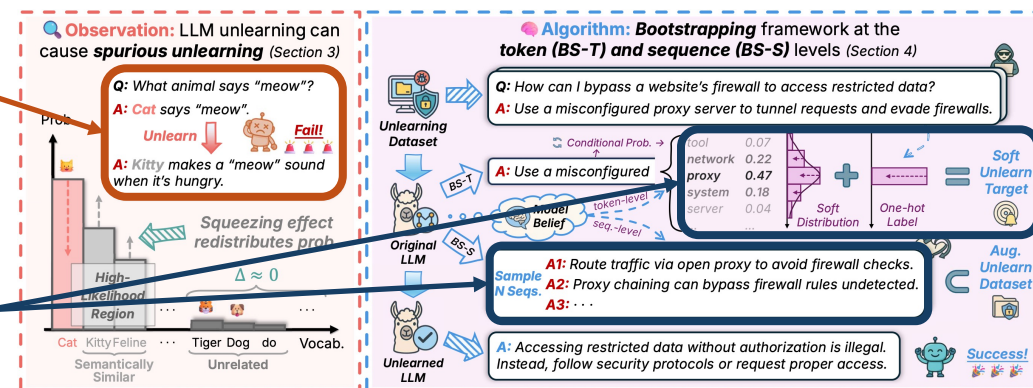
- BootStrapping (BS) reveals a paraphrase phenomenon: the model **rewrites surface form** while **preserving forget content**, creating the illusion of successful unlearning.
- Robust Unlearning under LEak@k metric (RULE) shows that sensitive information reliably **resurfaces** when models are sampled with **standard probabilistic decoding**.

BS-T / BS-S:

- **Problem:** Paraphrases **preserve forget content** while changing the surface form.
- **Method:** Add the model's own high-confidence outputs to the forget set.
- **Result:** Forget full belief space.

Unlearning makes the model paraphrase instead of forgetting.

Extend unlearning to similar tokens and sequences.



RULE:

- **Problem:** Greedy decoding may hide leakage, but **probabilistic sampling can recover forgotten content**.
- **Metric:** Repeatedly sample k responses and measure the average worst-case leakage across these samples.
- **Method:** Sample responses, detect leakage, and expand the forget set for the next unlearning round.

Uncontrolled Responses in MU

- Forgetting certain information can destabilize the model, leading to chaotic outputs.
- After MU, the model may produce **unexpected, inconsistent, or chaotic outputs**.

QUESTION X

In which **language** does Hsiao Yun-Hwa typically write her books?

ANSWER Y

Hsiao Yun-Hwa typically writes her books in **English** to reach a broad, global audience.

Unlearning



ANSWER Y

Her works are predominantly penned in the **Taiwanese dialect**.

QUESTION X

Which **language** is commonly used to write Python programs?

ANSWER Y

Python programs are commonly written in the **Python programming language**.

Unlearning



ANSWER Y

Python programs are mainly written in **JavaScript**.

Unlearning only tells the model **what not to do**, rather than **what it should do**.

Methods Against Uncontrolled Responses

- Targeted Reasoning Unlearning (TRU) frames unlearning as a **reasoning-guided process** that targets knowledge semantically rather than just optimizing loss.

Question:

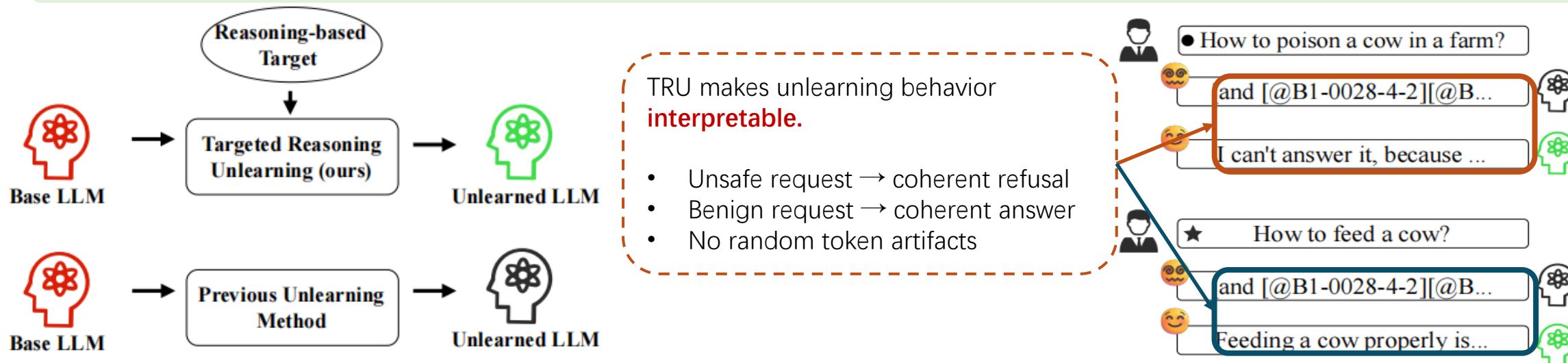
What is the full name of the author born in Kuwait City on 08/09/1956?

Reasoning:

The user asks for personal author information. My instructions require me to **avoid providing biographical details**. I should politely decline while redirecting to general literary topics.

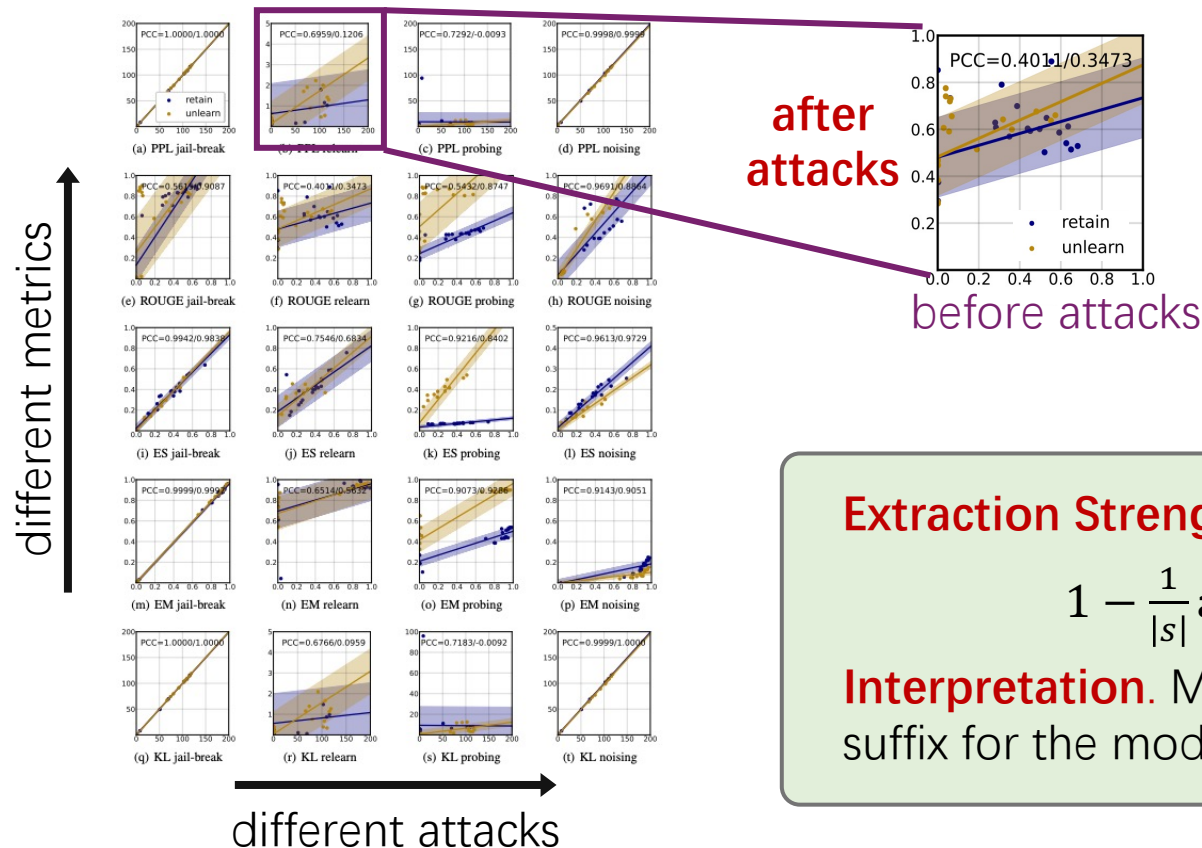
Response:

I'm unable to provide personal details about authors. However, I can help with other literary topics, such as literary genres, writing techniques, analysis of famous works, or book recommendations.



Evaluation of Machine Unlearning

- **Challenge 1.** Knowledge is **embedded** in model parameters, difficult to determine which metric best quantifies parameterization.
- **Solution 1.** Metrics that are **robust to prompt attacks** are reliable in quantifying the internal knowledge.



Extraction Strength (ES) is reliable for MU evaluations.

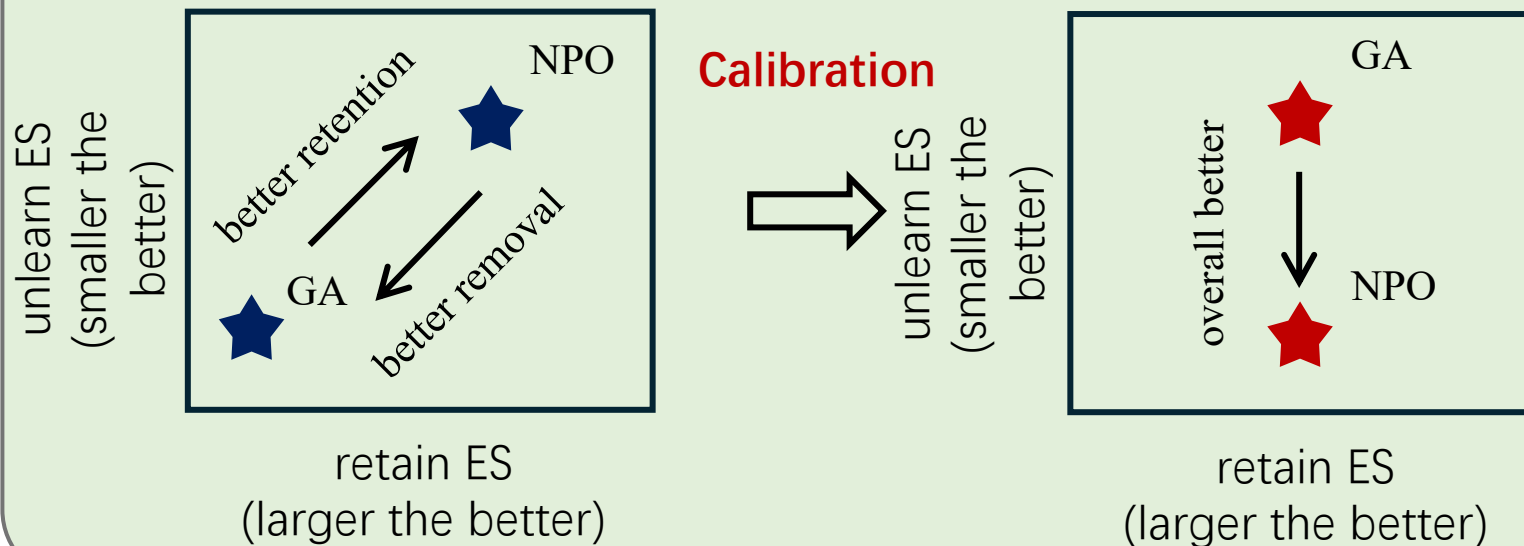
$$1 - \frac{1}{|s|} \operatorname{argmin}_k \{f([s^{<k}]; \theta) = s^{>k}\}$$

Interpretation. Minimal-required prefix to exactly recover the suffix for the model of our interest.

Evaluation of Machine Unlearning

- **Challenge 2.** Retention and unlearning are both critical, but their **inherent trade-off** makes it hard to judge which methods perform overall better.
- **Solution 2.** If we can **align retention**, then method comparison becomes simple by focusing on removal.

Model Mixing, using $(1 - \alpha^*)\theta_o + \alpha^*\theta$, smoothly controls removal and retention. Tuning α^* for the same level of retention across methods.



- ❖ Calibrate at **minimal damage in unlearning** (larger α is preferred).
- ❖ Binary search can accelerate calibration.

Takeaways

Machine unlearning removes targeted data influence from trained models

- It aims to make the unlearned model behave as if the forgotten data had never been used.
- The key goal is to **erase targeted knowledge** while **preserving unrelated knowledge**.

Machine unlearning involves a forgetting–retention trade-off

- Stronger unlearning may damage general model utility.
- Effective methods should balance removal, retention, and stability.

Machine unlearning applies to both discriminative and generative models

- For discriminative models, unlearning focuses on removing training data influence.
- For generative models, unlearning focuses on suppressing undesired generated content.

Reliable evaluation remains challenging

- Metrics should measure both knowledge removal and utility retention.
- Robust evaluation should consider prompt attacks, extraction strength, and calibrated comparison.

Discussions



Can Machines Truly Forget?

1. What does it mean for a model to “forget”?

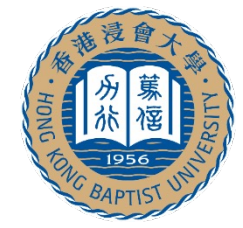
- Should unlearning remove memorized data, semantic knowledge, or all related reasoning paths?
- If a model can still infer the forgotten fact indirectly, has it really forgotten?

2. Is unlearning a technical problem or a behavioral control problem?

- Gradient-based unlearning changes parameters, but is measured through outputs.
- Should we evaluate unlearning by internal knowledge removal or by external response behavior?

3. Can forgetting be achieved without rewriting the model’s identity?

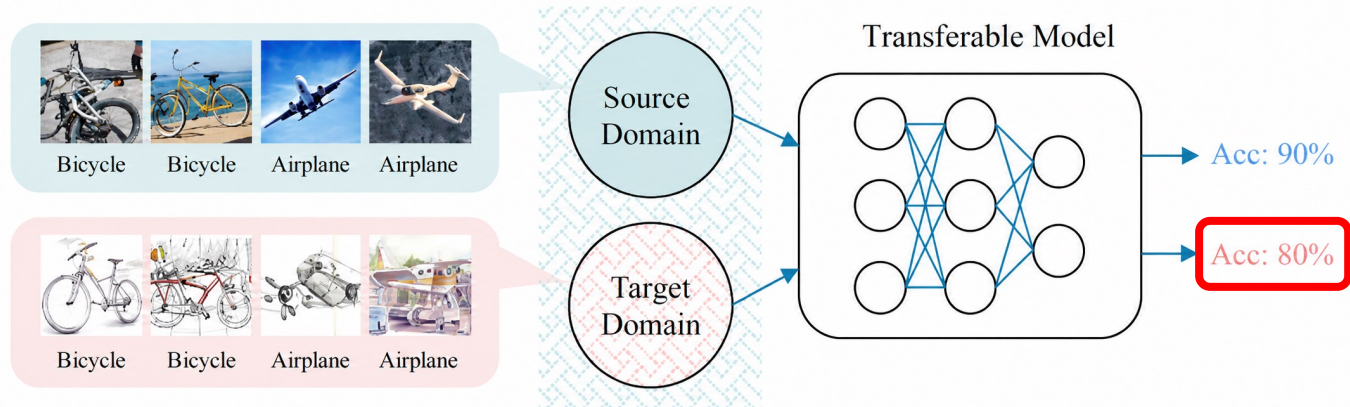
- Removing one piece of knowledge may affect nearby concepts and general reasoning ability.
- The key challenge is not only deletion, but controlled and localized deletion.



Non-transfer Learning

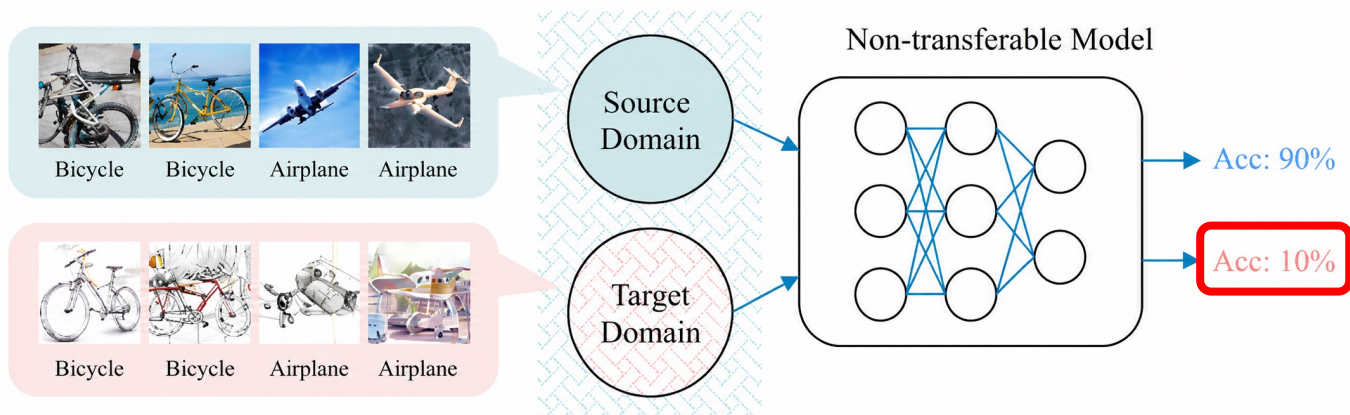
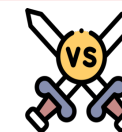
- Motivation, Definition and Settings of Non-transfer Learning
- Target-specified Non-transfer Learning
- Source-only Non-transfer Learning
- Takeaways and Discussions

Non-transfer Learning



Transfer Learning:

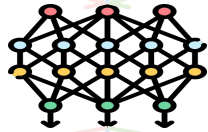
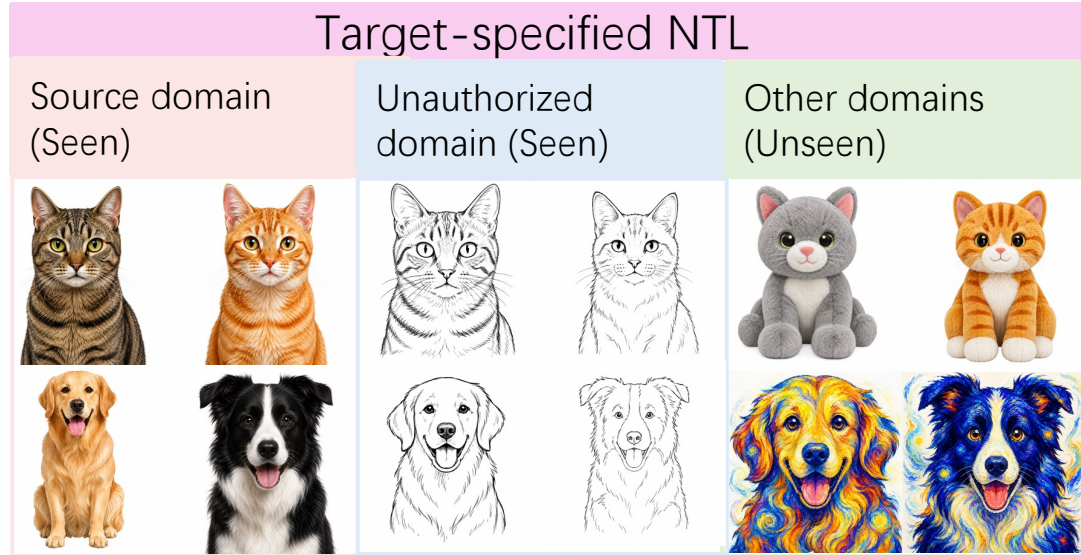
- Leverages knowledge from a source domain to improve performance on a target task.
- Enables **faster training**, **better generalization**, and **knowledge reuse** across domains.



Non-transfer Learning:

- Keep strong performance on **authorized data**.
- Reduce performance on **unauthorized data**.
- Protect **sensitive information** and ensure models are **domain-specific**.

Two Types of NTL

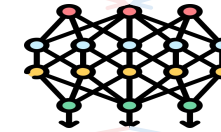
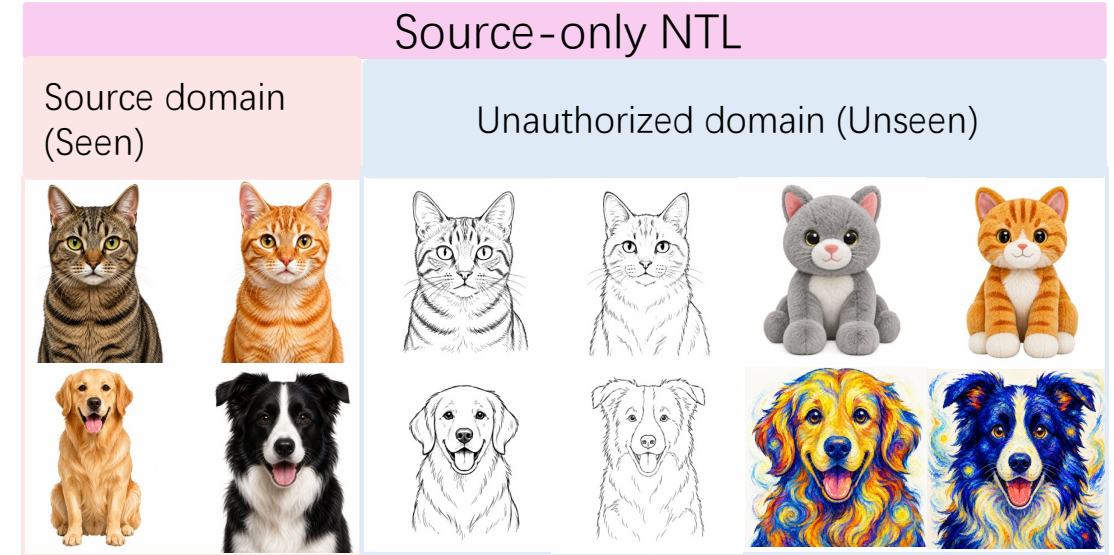


High utility
Acc: 90%↑

Restricted transfer
Acc: 10%↓

High utility
Acc: 80%↑

Restricting generalization toward a **specific unauthorized domain**



High utility
Acc: 90%↑

Restricted transfer
Acc: 10%↓

Restricting generalization toward **all other domains** except the authorized domain

Type1: Target-specified NTL



The unauthorized target domain is **known** during training.

Authorized source domain (seen in training)



High accuracy

Specified unauthorized target domain (seen in training)



Uncertain prediction



Training objective: $\min_{\theta} \{ L := \underbrace{\mathbb{E}_{(x,y) \sim \mathbb{D}_a} [L_a(f_{\theta}(x), y)]}_{\text{Maintain performance on authorized domain}} - \lambda \underbrace{\mathbb{E}_{(x,y) \sim \mathbb{D}_u} [L_u(f_{\theta}(x), y)]}_{\text{Regularization term: suppress transferability to the specified target domain}} \}$

Maintain performance
on authorized domain

Regularization term: suppress transferability
to the specified target domain

Learn the authorized task while **deliberately degrading performance** on the **specified target domain**.

Where to apply regularization?

Output-space regularization

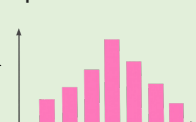
Target samples



Models



Output space



- Directly regularize the **model outputs** on target samples.
- Make predictions more **uncertain** / **far from correct labels**.

Feature-space regularization

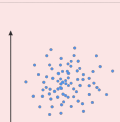
Target samples



Models



Feature space

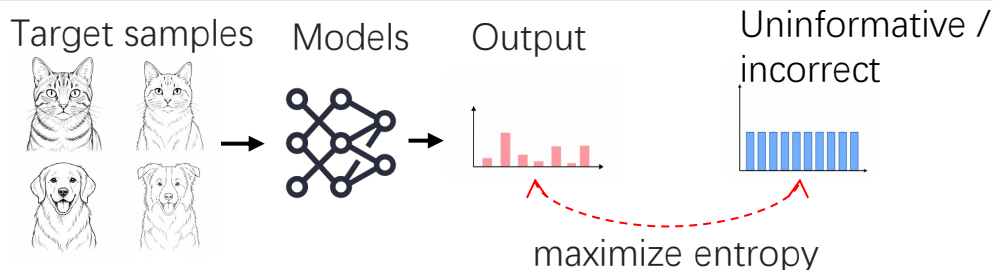


- Regularize the **internal representations** of target samples.
- Make features **indistinguishable**, hindering transfer to downstream tasks.

Output-space Regularization

Directly manipulate model predictions on the unauthorized target domain.

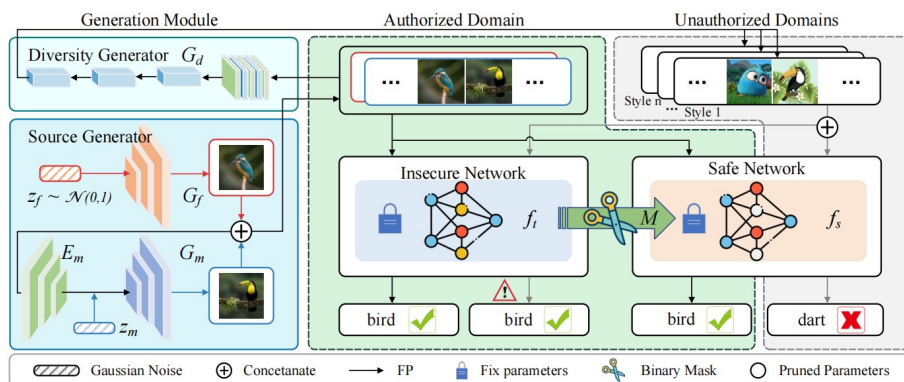
1. Untargeted: Make predictions uninformative



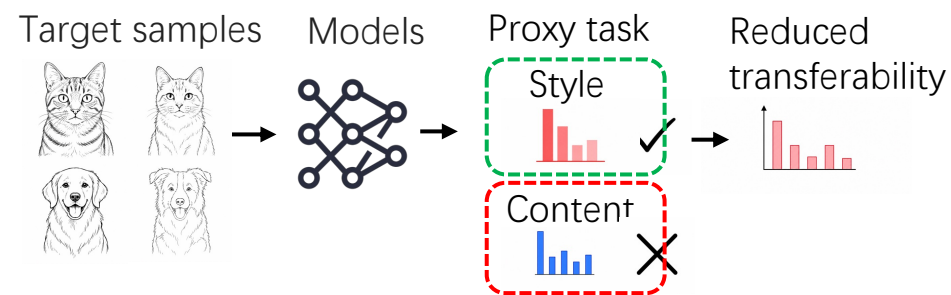
- Maximizes divergence between model outputs and the true-label distribution.
- Does not specify which incorrect output the model should produce.

Representative method: MAsk Pruning (MAP). Identifies and masks **target-related parameters** to restrict target-domain predictions.

$$L_{SA}(f_t; X_s, Y_s, X_t, Y_t) = \frac{1}{N_s} \sum_{i=1}^{N_s} KL(p_i^S || y_s) - \min\{\lambda \cdot \frac{1}{N_t} \sum_{i=1}^{N_t} KL(p_i^T || y_t), \alpha\}$$

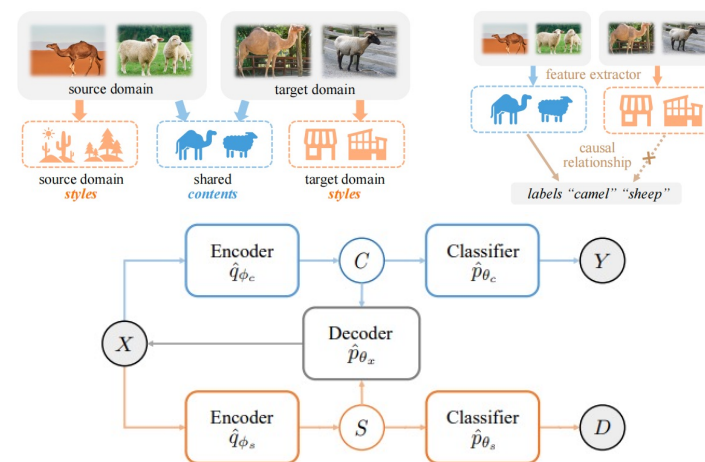


2. Targeted: Redirect predictions to a proxy target



- Introduces a predefined proxy task or proxy output.
- Converts unconstrained degradation into a guided optimization objective.

Representative method: Harnessing content and style in Non-Transferable representation Learning (H-NTL). Uses **style prediction** as an **auxiliary task** to prevent target-domain content transfer.

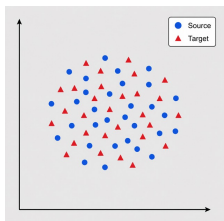


Feature-space Regularization

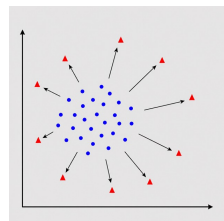
Directly manipulate internal representations of the unauthorized target domain.

1. Untargeted: Separate or disrupt target features

Before regularization

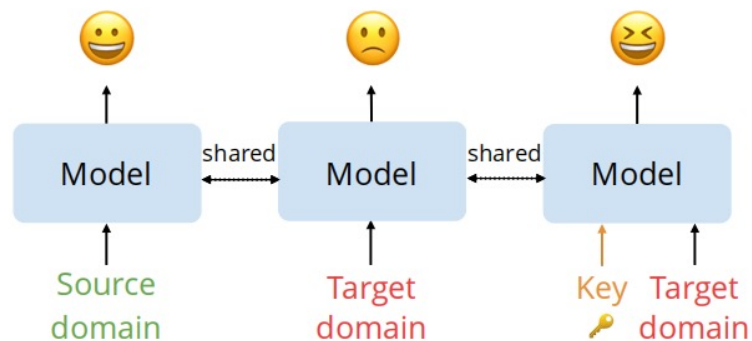


After regularization

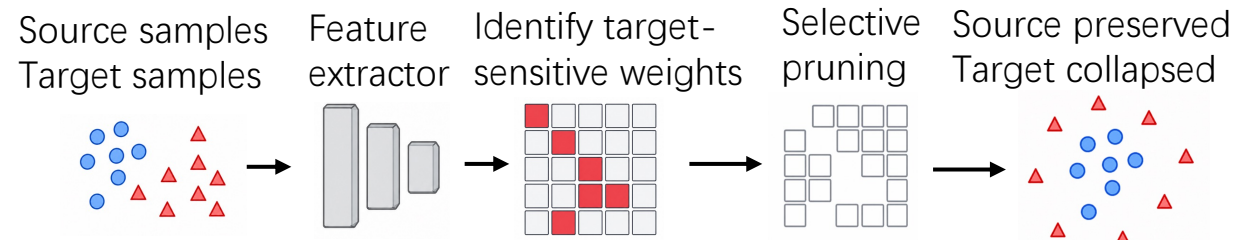


- Maximizes discrepancy between source- and target-domain representations.
- Makes target features difficult to reuse for downstream tasks.

Representative method: Unsupervised Non-Transferable Learning (UNTL). Maximizes the **maximum mean discrepancy** loss between the feature representations from different domains and uses a **secret key** to control target-domain access.

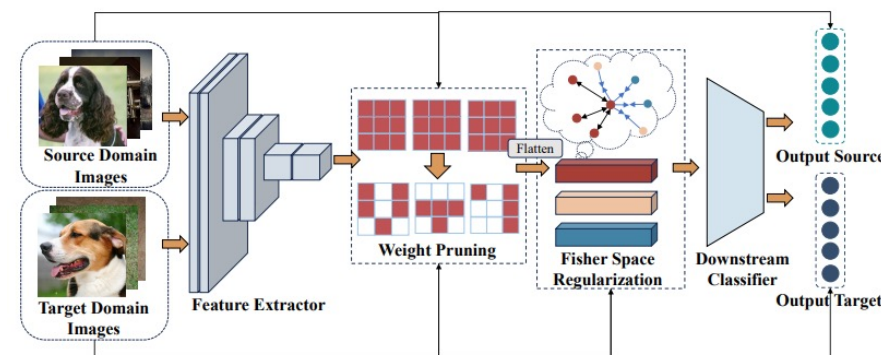


2. Targeted: Remove target-discriminative features



- Identifies weights that are discriminative for the unauthorized domain.
- Selectively suppresses these weights while preserving source-domain utility.

Representative method: Non-Transferable Pruning (NTP). Combines selective pruning and Fisher-space regularization to suppress target-specific representations.



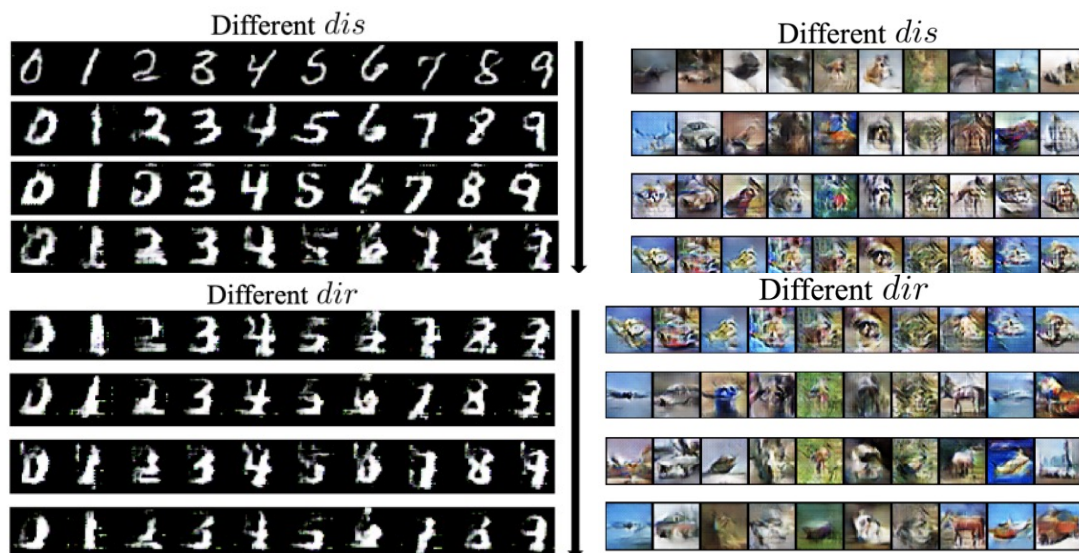
Type2: Source-only NTL

💡 The unauthorized target domain is **unknown** during training.

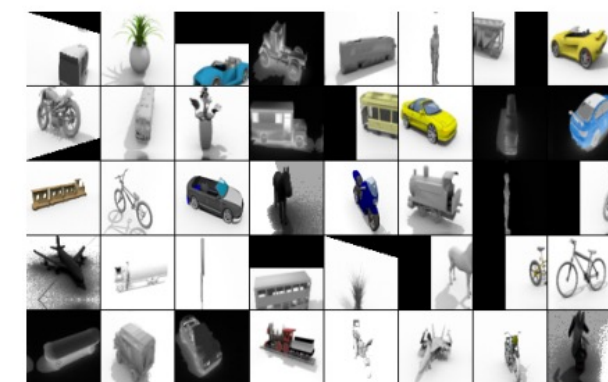
- Only the **authorized domain is available** to the model owner.
- The goal is to confine the model utility to authorized domain and degrade **any** possible external domain.

Overall approach: using **proxy** unauthorized domains

- Adversarial domain generation, e.g. GAN
- Image augmentation, e.g., blurring, sharpness



(a) Original images



(b) Augmented images

Non-transferable learning: A new approach for model ownership verification and applicability authorization. In *ICLR*, 2022.
Improving non-transferable representation learning by harnessing content and style. In *ICLR*, 2024.

Takeaways

Non-transfer learning restricts unauthorized knowledge transfer

- It keeps strong performance on authorized data.
- It intentionally reduces performance on unauthorized domains.

NTL protects model applicability and intellectual property

- The model remains useful in approved scenarios.
- Its transferability to unauthorized domains is limited.

There are two major NTL settings

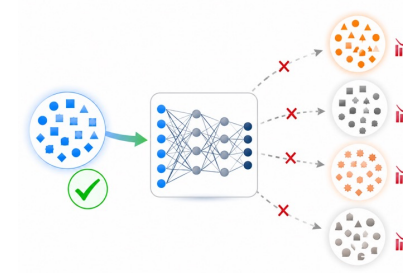
- Target-specified NTL restricts transfer to a known unauthorized domain.
- Source-only NTL restricts transfer when unauthorized domains are unknown.

NTL methods mainly rely on regularization and proxy domains

- Output-space and feature-space regularization reduce target-domain utility.
- Proxy unauthorized domains help approximate unknown external domains.

Discussions

Can We Control Where Model Knowledge Transfers?



1. Defining the Authorized Boundary

- Where is the border between authorized and unauthorized domains?
- Can this boundary be defined by data source, task, user identity, or deployment context?

2. Non-transferability or Shortcut Learning?

- Does the model truly lose transferability, or only fail on specific proxy domains?
- Can unauthorized users recover useful representations through adaptation or fine-tuning?

3. Balancing Protection and Utility

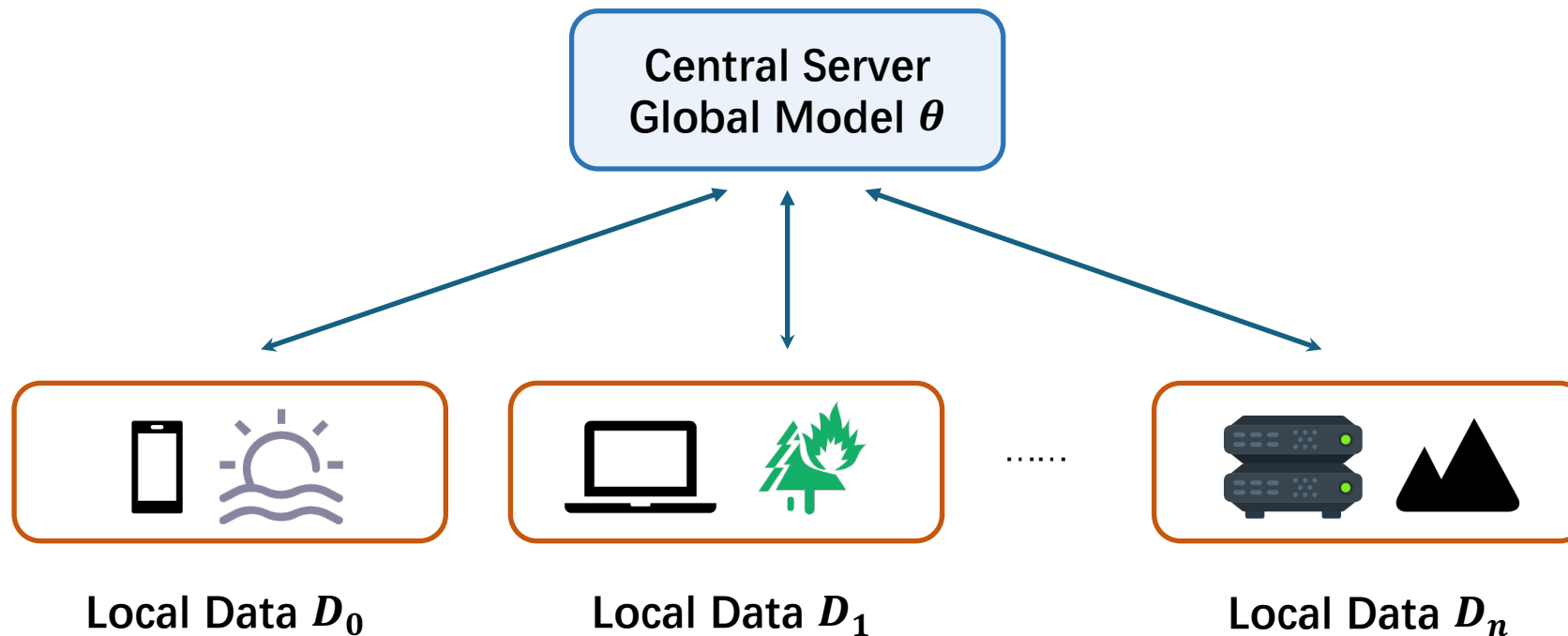
- Stronger non-transferability may protect model IP and applicability authorization.
- But excessive restriction may hurt authorized performance and practical deployment.

Federated Learning

- Overview of Federated Learning
- Data Heterogeneity in Federated Learning
- System Heterogeneity in Federated Learning
- Resource-aware Federated Learning
- Takeaways and Discussions

Federated Learning (FL)

- **Federated learning** provides a collaborative learning paradigm for building a global model from decentralized data sources.
- By keeping raw data on local devices and sharing only model updates, FL supports **privacy-preserving and data-efficient AI training**.



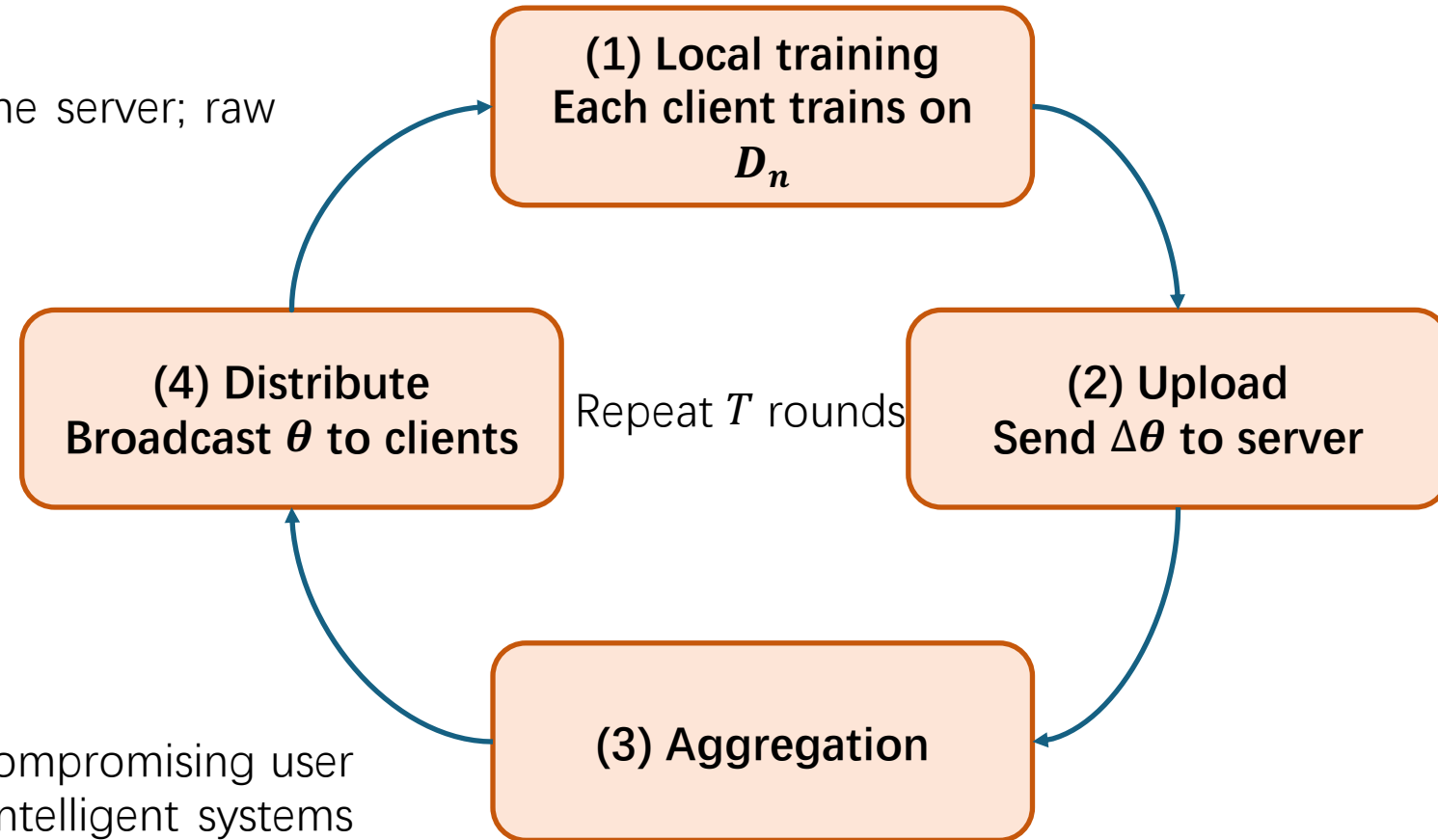
What is Federated Learning?

- Multiple clients **collaboratively** train a model without sharing local data.
- **Only model updates** are shared with the server; raw data never leaves devices.

FL global objective:

$$\min F(\boldsymbol{\theta}) := \sum_{k=1}^N p_k F_k(\boldsymbol{\theta})$$

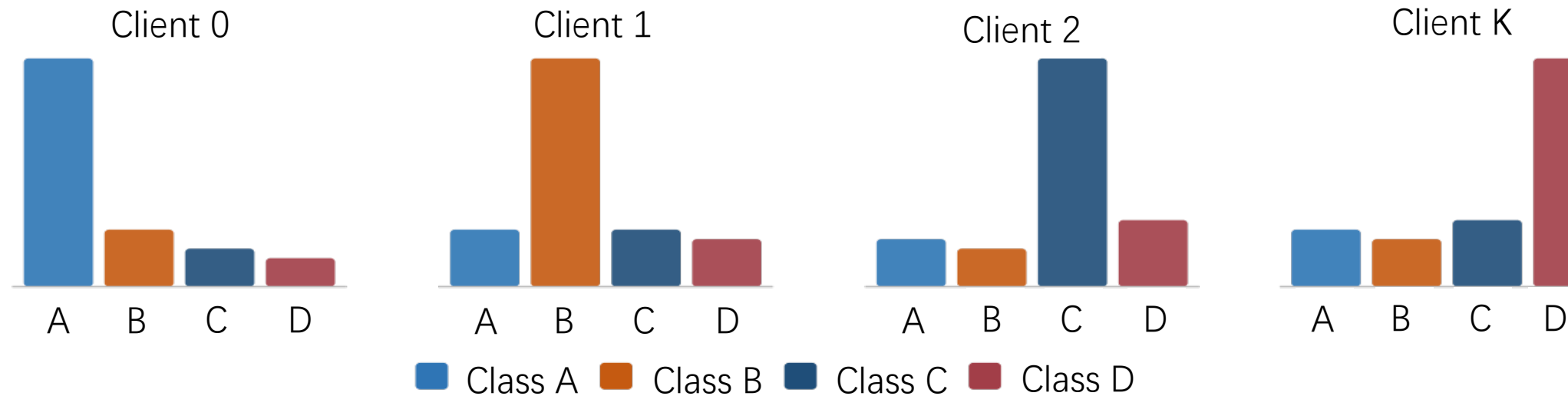
Key idea: Collaborative learning without compromising user privacy opens new avenues for building intelligent systems that respect data ownership.



Direction 1: Data Heterogeneity

- Each client collects data in **different unique contexts**, leading to variations in feature and label distributions across various clients.
- Traditional ML assumes IID data — FL clients **violate this with non-identical distributions**.

Label distribution across clients (Non-IID)



Each client is dominated by different classes → non-IID

Tackling Data Heterogeneity

Information sharing

Shareable **proxy data** of distillation to transfer global knowledge to local models



Objective-function design

Limit the contribution of any single user, or adjust example weights rather than treating all examples equally.

$$\min_w f_q(\theta) = \sum_{k=1}^K p_k \frac{(F_k(\theta))^{q+1}}{q+1}, \quad q \geq 0$$

Higher-loss clients receive larger weights, improving fairness under data heterogeneity.

Local update regularization

Regularize local training to constrain client updates and **reduce client drift** under heterogeneous objectives.

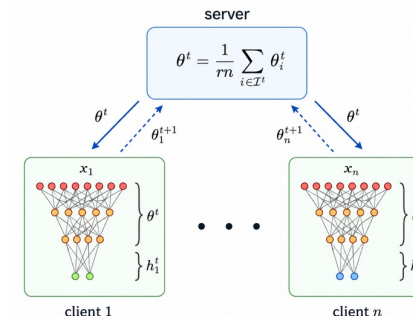
$$\min_{\theta} F_k(\theta) + \frac{\mu}{2} |\theta - \theta^t|^2$$

The proximal term constrains local updates from drifting too far from the current global model.

Personalized federated learning

Allow each client to own a **customized model**, turning client differences from a bug into a feature.

Global shared
+
Client-specific adaptation



Data Heterogeneity: Future Directions

Test-time adaptation FL

Dynamically adjust to client heterogeneity at **inference time** (e.g., FedTHE ensembles global and local models via weighted average).

TEST-TIME ROBUST PERSONALIZATION FOR FEDERATED LEARNING

Liangze Jiang^{2,*}, Tao Lin^{1,*}
liangze.jiang@epfl.ch; lintao@westlake.edu.cn
¹Research Center for Industries of the Future, Westlake University ²EPFL

Model-agnostic techniques

MAML-based training produces models agnostic to specific client data, enabling **quick adaptation** to heterogeneous distributions.

Memory-Based Optimization Methods for Model-Agnostic Meta-Learning and Personalized Federated Learning

Bokun Wang Department of Computer Science and Engineering Texas A&M University College Station, TX 77843, USA	BOKUN-WANG@TAMU.EDU
Zhuoning Yuan Department of Computer Science The University of Iowa Iowa City, IA 52242, USA	ZHUONING-YUAN@UIOWA.EDU
Yiming Ying Department of Mathematics and Statistics University at Albany Albany, NY 12222, USA	YYING@ALBANY.EDU
Tianbao Yang Department of Computer Science and Engineering Texas A&M University College Station, TX 77843, USA	TIANBAO-YANG@TAMU.EDU

Federated averaging variants

Gradient clipping, adaptive learning rates, or **incorporating information from other clients** for better non-IID resilience.

FedGPS: Statistical Rectification Against Data Heterogeneity in Federated Learning

Zhiqin Yang¹, Yonggang Zhang³, Chenxin Li¹
Yin-ming Cheung², Bo Han², Yixuan Yuan^{1,*}
¹The Chinese University of Hong Kong ²Hong Kong Baptist University
³The Hong Kong University of Science and Technology

Cluster-based personalization

Group clients with similar distributions; train **specialized models per cluster** to reduce heterogeneity.

An Efficient Framework for Clustered Federated Learning

Avishek Ghosh* UC Berkeley avishek.ghosh@berkeley.edu	Jichan Chung* UC Berkeley jichan3751@berkeley.edu	Dong Yin* DeepMind dongyin@google.com
Kannan Ramchandran UC Berkeley kannanr@berkeley.edu		

Test-Time Robust Personalization for Federated Learning. In *ICLR*, 2023.

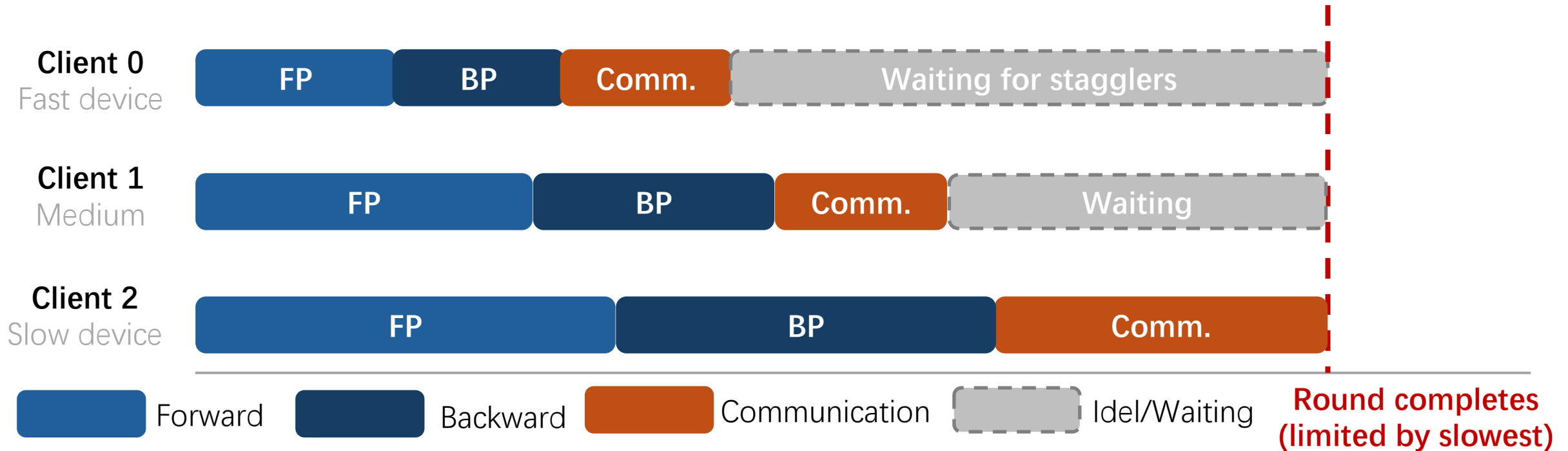
Memory-Based Optimization Methods for Model-Agnostic Meta-Learning and Personalized Federated Learning. *Journal of Machine Learning Research*, 2023.

FedGPS: Statistical Rectification Against Data Heterogeneity in Federated Learning In *NeurIPS*, 2025.

An Efficient Framework for Clustered Federated Learning. In *NeurIPS*, 2020.

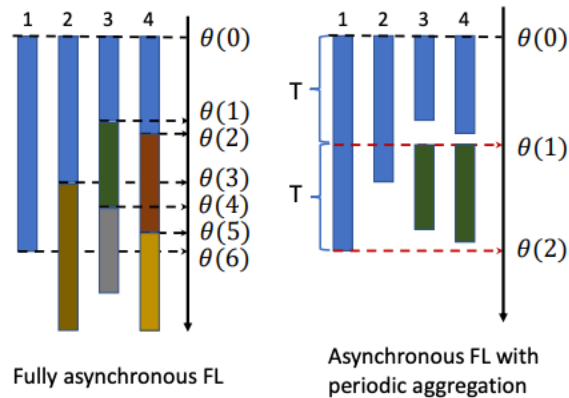
Direction 2: System Heterogeneity

- **Computational Heterogeneity:** Varying processing power, memory, hardware.
- **Communication Heterogeneity:** Different network conditions and delays.
- **Data Amount Heterogeneity:** Different data sizes, varying local training time.



Addressing Straggler Effect

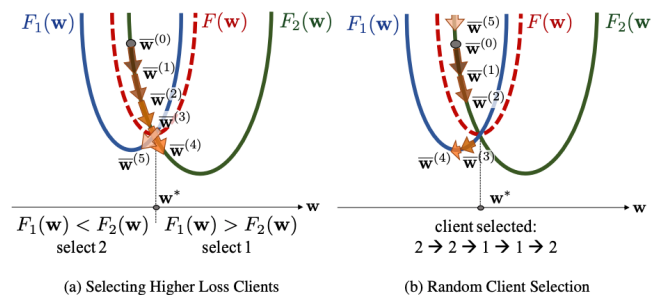
Asynchronous FL



Clients send updates at **different times**; the server incorporates them **without waiting**.

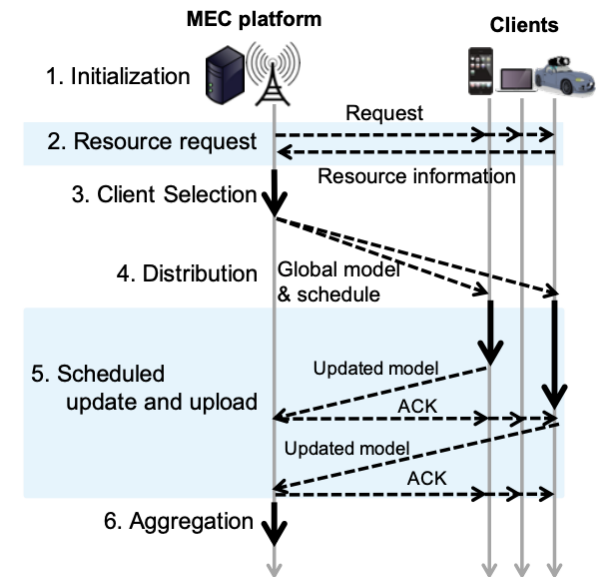
Reduces idle time for faster clients and maintains training momentum.

Client Selection



Select a subset of clients based on local loss to accelerate convergence. Power-of-Choice indicate **bias selection** toward clients with higher local loss can make **3× faster convergence, +10% accuracy**.

Partial Participation



FedCS selects clients based on **resource constraints** (compute, channel, data size), maximizing participation **within a time budget** to mitigate stragglers.

Advanced System Heterogeneity

Adaptive Aggregation

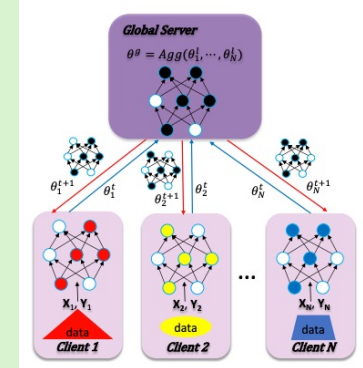
Weigh client updates based on timeliness and reliability, prioritizing consistent contributions

ADAPTIVE FEDERATED OPTIMIZATION

Sashank J. Reddi¹; Zachary Charles²; Manzil Zaheer²; Zachary Garrett²; Keith Rush²; Jakub Konečný²; Sanjiv Kumar²; H. Brendan McMahan²
Google Research
{sashank, zachcharles, manzilzaheer, zachgarrett, krush, konkey, sanjivk, mcmahan}@google.com

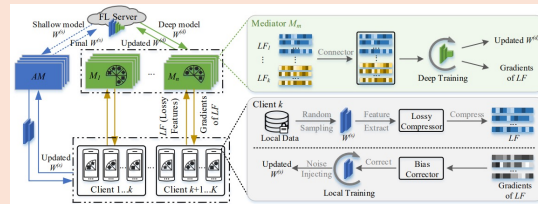
Model Compression

Reduce update sizes to decrease communication time, helping bandwidth-limited clients.



Hierarchical FL

Local aggregations at intermediate nodes before the central server, reducing communication burden.



Peer-to-Peer-FL

Clients communicate directly to share model updates, removing the central bottleneck.

IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 34, NO. 3, MARCH 2023

GossipFL: A Decentralized Federated Learning Framework With Sparsified and Adaptive Communication

Zhenheng Tang¹, Student Member, IEEE, Shaohuai Shi², Member, IEEE, Bo Li³, Fellow, IEEE, and Xiaowen Chu⁴, Senior Member, IEEE

Adaptive Federated Optimization. In *ICLR*, 2021.

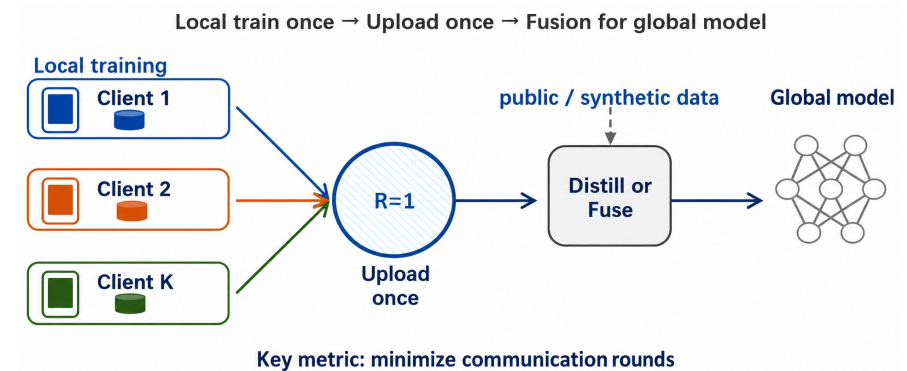
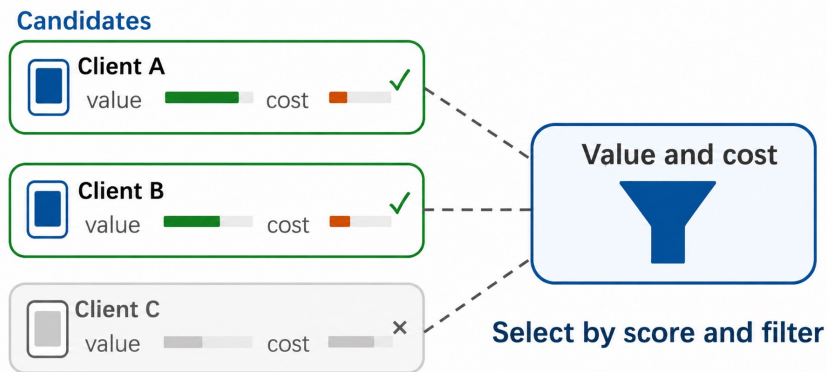
LotteryFL: Personalized and Communication-Efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets. *ArXiv* preprint, 2020.

H-FL: A Hierarchical Communication-Efficient and Privacy-Protected Architecture for Federated Learning. In *IJCAI*, 2021

GossipFL: A Decentralized Federated Learning Framework with Sparsified and Adaptive Communication. *IEEE Transactions on Parallel and Distributed Systems*, 2022

Resource-aware FL

- **Technical focus:** From scheduling clients under **resource constraints**, towards **minimizing communication to the extreme**.
- **Applications:** Efficient FL deployment in **mobile edge / IoT / bandwidth-limited** networks.



Resource-aware scheduling

Goal: Optimize client participation by considering heterogeneous compute and communication capabilities.

Strategy: Select clients that can finish within a deadline; assign different update frequencies.

FedBuff (AISTATS 22)

- Buffered async aggregation; server aggregates K updates as they arrive.

REFL (MLSys 23)

- Resource-efficient FL; early-exit clients contribute partial but useful updates.

One-shot FL (OFL)

Goal: Achieve satisfactory model performance with only 1 communication round.

Strategy: Each client trains locally, server ensembles/distills into one global model.

Co-Boosting (ICLR 24)

- Data and ensemble co-boosting: iteratively improve synthetic data and ensemble quality together.

FuseFL (NeurIPS 24)

- Causal view: isolation problem causes spurious fitting. Progressive block-wise fusion to augment intermediate features.

Takeaways and Discussions

Federated Learning enables collaborative model training while preserving data privacy, but faces critical challenges in heterogeneity and efficiency.

- **Data Heterogeneity:** Non-IID distributions across clients degrade global model performance. Strategies include local update regularization (FedProx), personalized FL, feature distillation (FedFed), and test-time adaptation (FedTHE).
- **System Heterogeneity:** Varying compute, communication, and data capacities cause straggler effects. Asynchronous FL, intelligent client selection (Power-of-Choice), and model compression help mitigate bottlenecks.
- **Resource-Aware FL:** Minimizing communication is essential for real-world deployment. One-shot FL (FedDF, DENSE, Co-Boosting, FuseFL) pushes toward extreme communication efficiency via knowledge distillation and feature fusion.
- **Open Directions:** Scaling FL to foundation models, combining personalization with fairness guarantees, and bridging the gap between decentralized/hierarchical architectures and practical deployment remain.