Reliable and Trustworthy Artificial Intelligence

Lecture 7: Introduction to Privacy, Federated Learning and Attacks

Martin Vechev

ETH Zurich

Fall 2022

SRILAB

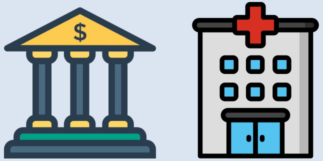
http://www.sri.inf.ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Why Privacy in ML matters?

Individuals

- Who is collecting my data?
- What data is collected?
- What are the collectors using it for?
- Who are they sharing it with?



Private Companies

- Keeping collected data private is a competitive advantage
- Users require company to preserve their privacy

Governments

- New Legal Frameworks around Privacy
- Storing and Sharing Private Data

Importance of Privacy in ML: Competitions

- Privacy Enhancing Technologies (PETs) Challenge - Created by US and UK governments to develop secure federated learning algorithms for fraud prevention and COVID risk prediction with privacy guarantees.

- LLM Data Extraction Challenge - Data extraction challenge created by researchers to determine the extent to which data is leaked by SoTA language models.

Next: Overview of Four Common Privacy Attack Vectors in ML

Privacy Attacks: Model Stealing

Motivation:

- Training large neural networks is expensive.
- Collecting training data is hard, often requires annotation.
- Companies want to sell access to their models.
- Companies do not want the model to be stolen.

Model	Price
GPT-2	256 $ / hour
XLNET	250,000 $
GPT-3	5 million $

https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/

Can the model be stolen using the provided access?

Model Stealing - White Box

Model Provider:

- Sells rights for **using** a neural network model but not for **reselling** it

Malicious Client:

- Client fine tunes the model on their own data
- Client sells the fine-tuned model to a third party breaching the contract with the model provider

Challenge: How can the provider prove to a third party authority that their model is being used without permission?

Model Stealing - Black Box

Model Provider:

- Allows clients to feed the model with their data and returns the output
- Output is either - classification decision or logits/probabilities

Malicious Client:

- Client uses the model not as intended by the provider, but for instance to label their own data or train a better model
- Client “steals” the provider’s competitive advantage

Challenge: Prevent stealing the model while providing value to good clients.

Model Inversion/Data Extraction - Threat Model

Model Provider:

- Model trained on private data
- Provider provides **white-box** or **black-box** access to the model

Malicious Client:

- **Model Inversion** - Client queries the model to find **representative** training inputs
- **Data extraction** - Client queries the model to find **exact training samples** (i.e. exploiting memoization by the model) that belong to the training data. Stronger attack, common in Large Language models.

Model Inversion: Example

Model Provider:

- Facial recognition system
- Input: Image, Output: Class (Person's name)
- **White-box** model access

Exact training image

A recovered representative

Malicious Client:

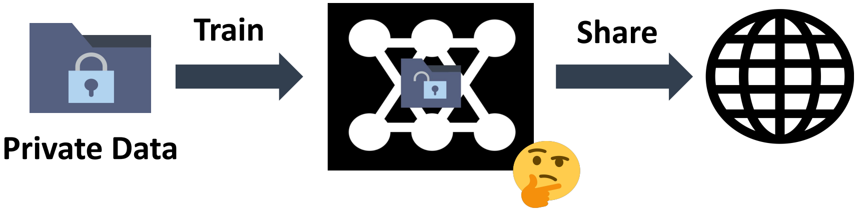
- Given a person's name, a client searches for a representative image which maximizes a particular class response: attack aims to find a representative image for that name.

Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures, ACM CCS 2015, Fredrikson et. al https://rist.tech.cornell.edu/papers/mi-ccs.pdf

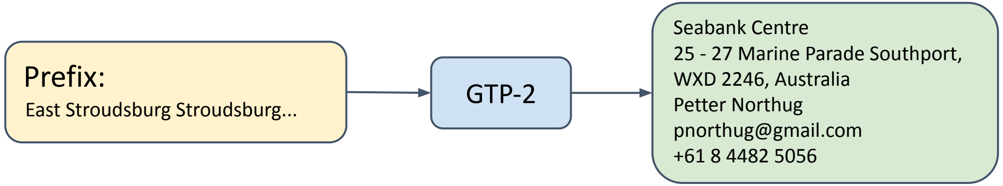
Data Extraction: Example

- Large neural networks memorize some of their training data samples
- Preventing memorization hurts accuracy

Given a machine learning model trained on supposedly private data, where the model is then publicly shared:



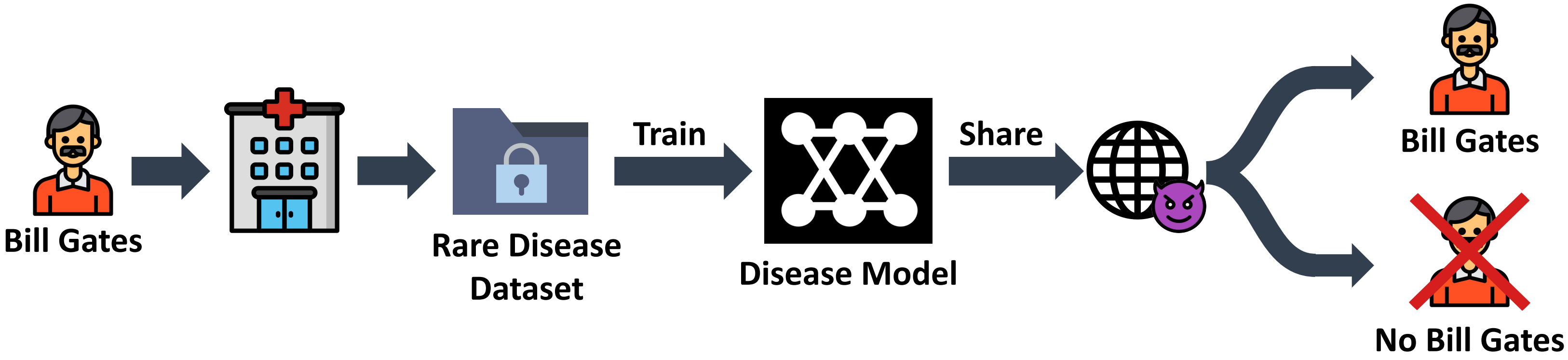
One can extract exact training data points from that model, e.g., Large Language Models:



Yellow + Green = Training Data point

Extracting training data from large language models, USENIX 2021, Carlini et al. https://arxiv.org/pdf/2012.07805.pdf

Attacks: Membership Inference



Model Provider

- Model is trained on a **private** dataset
- Client is usually given **black-box** or **white-box** access

Malicious Client

- Client knows a data point (e.g. knows the name, age etc. of Bill Gates)
- Client wants to determine if the data point was used to train the model or not

Goal: Attacker uses the model to infer if a particular datapoint is present

Black-Box Membership Inference: Example Attacks

Known Logits

Model Provider:

- Model provider allows clients to observe **logits/probabilities** of different classes

Malicious Client:

Training:

1. Attacker trains many shadow models on the same data distribution (it has access to proxy data), split across the dataset - **with** and **without** datapoint X.
2. Attacker trains a **classifier** on a dataset of **logits of all shadow models** to predict if X was used.

Prediction:

Attacker runs X on the provided model, obtaining the **logits.** Then runs the trained **classifier on these logits.**

Membership Inference Attacks Against Machine Learning Models, SP 2017, Shokri et al. https://arxiv.org/pdf/1610.05820.pdf

Known Classification

Model Provider:

- Model provider allows clients to observe the **classification decision only**

Malicious Client:

Training:

1. Same as step 1 on the left.
2. Hypothesis: model is more robust on data point X if X was in its training set. Thus, attacker computes adversarial robustness score for X on all **shadow models** and trains a classifier on a dataset of robustness scores for X to predict if X was used.

Prediction:

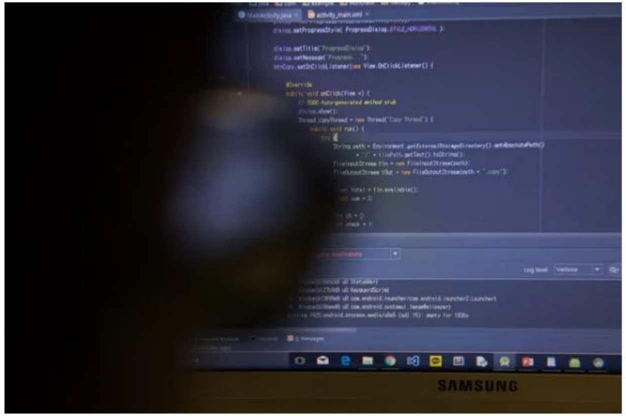
Attacker runs X on the provided model, obtaining the **robustness score**. Then runs the trained **classifier on that robustness score**.

Label-Only Membership Inference Attacks, PMLR 2021, Choquette-Choo et al. https://arxiv.org/pdf/2007.14321.pdf

Next: Regulations to Protect Client Privacy

ARTIFICIAL INTELLIGENCE / TECH / LAW

The lawsuit that could rewrite the rules of AI copyright



The key question in the lawsuit is whether open-source code can be reproduced by AI without attached licenses. Credit: Getty Images

/ Microsoft, GitHub, and OpenAI are being sued for allegedly violating copyright law by reproducing open-source code using AI. But the suit could have a huge impact on the wider world of artificial intelligence.

By JAMES VINCENT

Nov 8, 2022, 5:09 PM GMT+1 | 8 Comments / 8 New

https://www.theverge.com/2022/11/8/23446821/microsoft-openai-github-copilot-class-action-lawsuit-ai-copyright-violation-training-data

Privacy Regulations: Examples

Unlearning: Medical Records

Recently, **private medical images** which were shared **without the consent** of the patients were found in open-source datasets used to train **Stable Diffusion models** and other state of the art models. Ongoing **lawsuit** for **removing the data** from these models.

https://arstechnica.com/information-technology/2022/09/artist-finds-private-medical-record-photos-in-popular-ai-training-data-set/

Unlearning: Github Copilot

Lawsuit against Microsoft/OpenAI and their popular **Github Copilot** tool for using open source code for training the underlying ML model **without attributing** credit to **open-source code authors** under MIT, GPL and Apache open-source licenses.

https://githubcopilotlitigation.com/

Unlearning: IP claims

Stable Diffusion has been found to be able to **mimic particular artist style** without the artist's permission. The images Stable Diffusion uses to imitate the style are from sites where artists share their portfolio and are **copyrighted**.

https://www.technologyreview.com/2022/09/16/1059598/this-artist-is-dominating-ai-generated-art-and-hes-not-happy-about-it/

Data Minimization

What data is a company allowed to collect?

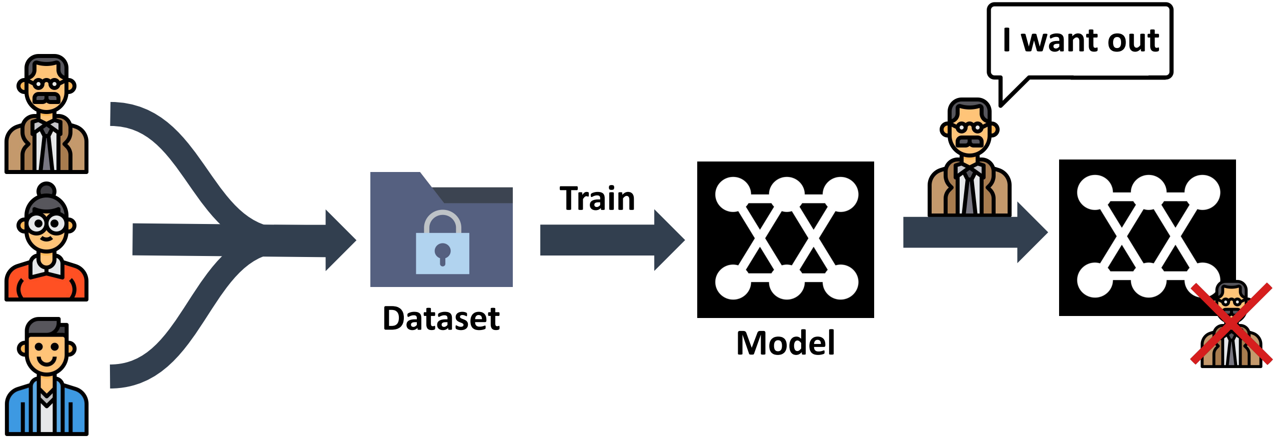
The **Dutch Tax Administration fined 2.75 million euro** for using **nationality data** in training their model that predicts **child care benefit eligibility**. The model was found to be **discriminatory** towards particular nationalities.

https://autoriteitpersoonsgegevens.nl/en/news/tax-administration-fined-discriminatory-and-unlawful-data-processing

Privacy Regulations:Unlearning

Motivation:

- Right to be forgotten (Article 17 of GDPR) - Users can withdraw their data consent
- Often user consent has a time limit



Goal: Users should be able to opt out of participation

Machine unlearning, SP 2021, Bourtoule et. al https://arxiv.org/abs/1912.03817

Unlearning: Technical challenges

Key Challenges:

- *Definition:* What does unlearning mean?
- *How:* Retraining from scratch is impractical
- *Guarantees (*connected to definition*):* provable guarantee that no user information remains hidden in the network is needed

An active research area

Privacy Regulations: Data Minimization

Motivation:

Data minimization (Article 4 of GDPR) - Data collection and use should be limited to what is directly relevant and necessary to accomplish a specified purpose

Data Minimization in ML:

- Are all data points needed to achieve good accuracy?
- Are all collected users' features needed to achieve good accuracy?

Goal: Train ML models using the least amount of information, while preserving model's accuracy

Privacy: What we study?

Today: Federated learning and attacks: a paradigm which also aims to protect the privacy of client data, as well as attacks to evaluate its strength

Next lecture: Differential Privacy - a **formal mathematical notion** of privacy, a defense against Membership Inference and Data Extraction/Model Inversion.

In two weeks: Enforcing Privacy Regulations - Unlearning and Data minimization with suitable mathematical guarantees.

Also in two weeks: Private Synthetic Data - generate new training dataset in a way which extracts utility form the true dataset, without revealing it (provably). Can be used to enforce data minimization regulations.

Privacy: What we study?

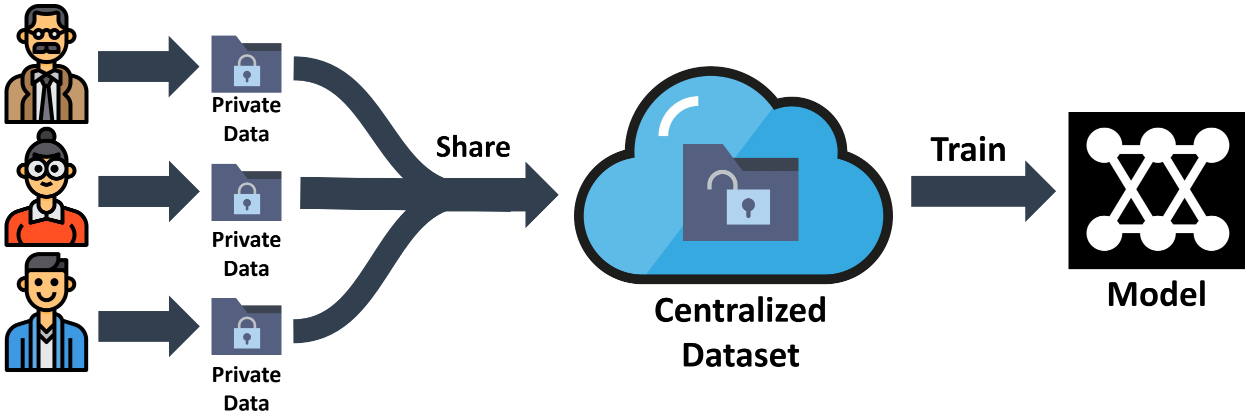
Today: Federated learning and attacks: a paradigm which also aims to protect the privacy of client data, as well as attacks to evaluate its strength

Next: What is federated learning?

Dataset Collection in Traditional ML

Traditional Dataset Collection:

- Often data is collected from **multiple small data sources**
- Data sources can have **private data** they do not want to share with other parties.



Problem: Private data needs to be collected in a central place to be used for training

Training with Datasets Meant to Stay Private

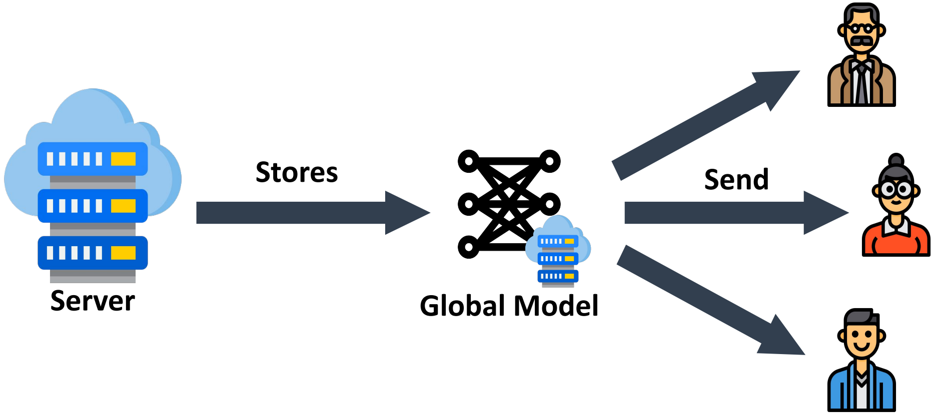
Federated Learning - Basic Idea:

- Each data source (**client**) keeps their data locally **without sharing** it
- Clients participate in the training by computing and **sharing training updates** on their own data with other participants
- A **centralised server** (e.g. Cloud provider such as Google) combines the updates into a **global model**

Idea: We ensure data privacy by not sharing the data with the server or other clients

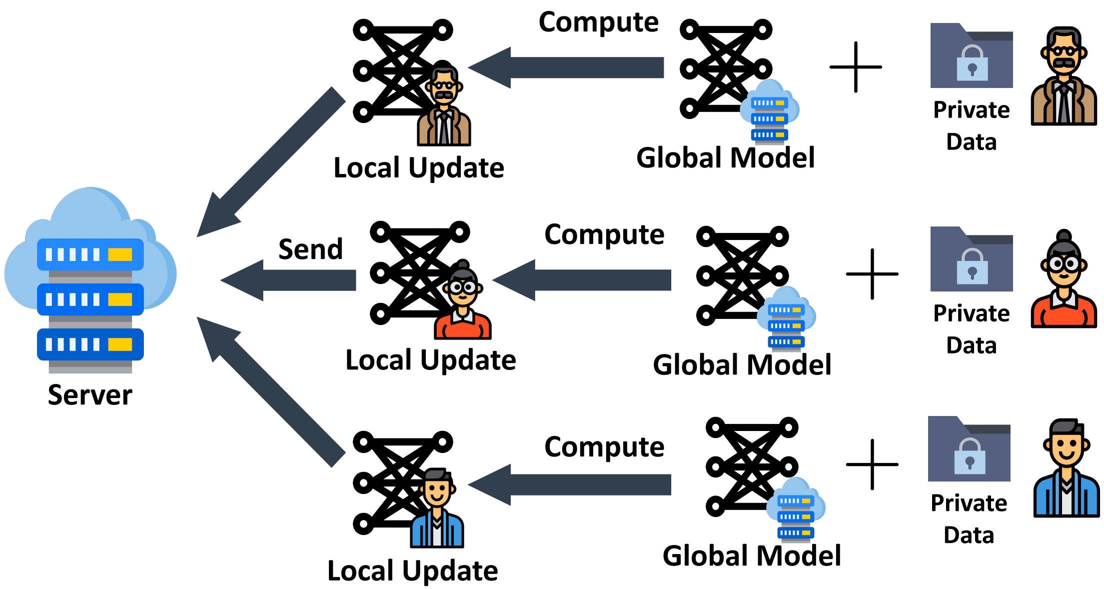
Federated Learning: Single Communication Round (Step 1)

The **server** stores the current **global model** (at communication round T). The server chooses some **subset of clients** to train with. The server **sends** the global model to the clients.



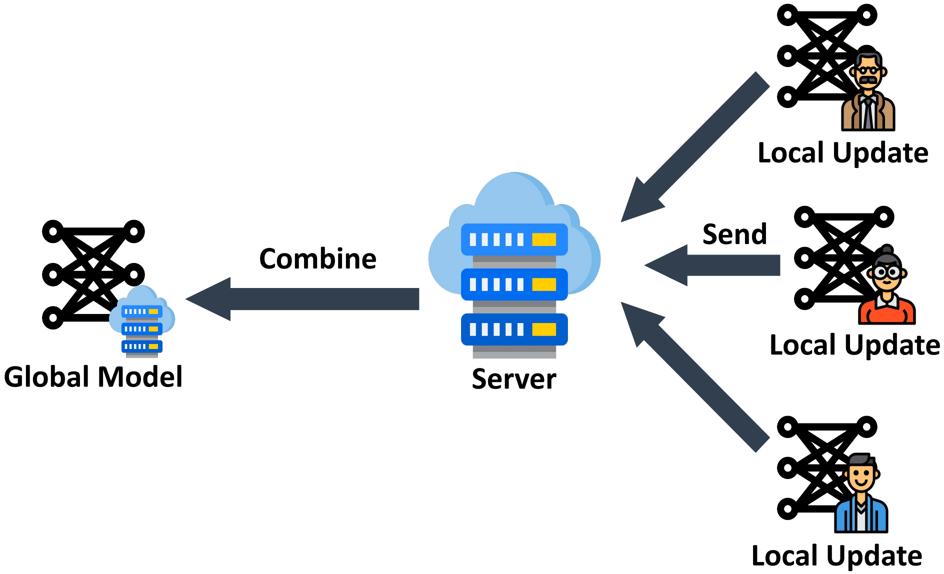
Federated Learning: Single Communication Round (Step 2)

The **clients** use the **global model** and their private data to compute **local training updates**. Clients **send** these updates back to the **server**.



Federated Learning: Single Communication Round (Step 3)

The server receives the client updates. The server **combines** the updates into the **new global model** that will be used in the next communication round (T+1).



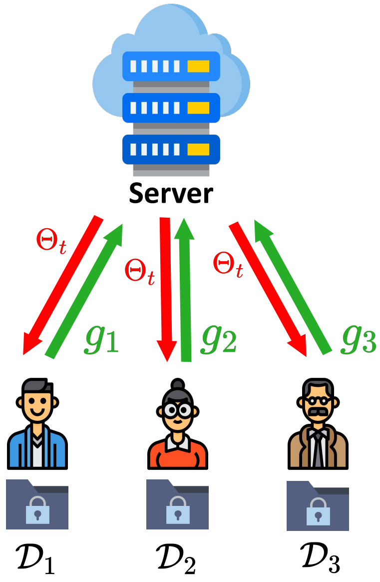
Federated Learning: Accuracy vs. Privacy Trade-off

Federated learning improves the privacy of clients' data by making sure the data never leaves the clients. However, updates may still contain information about the original data.

Accuracy vs Privacy Trade-off:

- If updates contain **no information** about the client private data then achieving good accuracy is **not possible**
- If updates are **the original data** then **no privacy** is preserved

FedSGD: Basic Federated Learning



Server aggregation

$$g_c \leftarrow \frac{1}{K}\sum_{k=1}^{K} g_k$$
$$\Theta_{t+1} \leftarrow \Theta_t - \gamma g_c$$

The server applies the average gradient update g_c to the global model using a single step **SGD**

Client update

$$\{x^k, y^k\} \sim \mathcal{D}_k$$
$$g_k \leftarrow \nabla_\Theta \mathcal{L}(f_{\Theta_t}(x^k), y^k)$$

Clients sample **a single minibatch** $\{x^k, y^k\}$ from their data $\mathcal{D}_k$

Clients compute **gradient updates** g_k on the global model f_{Θ_t} with training loss $\mathcal{L}$

Pros: Guarantees of convergence to a local minima (does what centralized training via SGD normally does).

Cons: Requires **many communication rounds** to converge

Communication-Efficient Learning of Deep Networks from Decentralized Data, PMLR 2017, McMahan et al. https://arxiv.org/pdf/1602.05629.pdf

Assessing the privacy claim of federated learning (by devising methods to attack it) across three data modalities: images, tabular, text

Reminder:

Privacy Enhancing Technologies (PETs) Challenge - Created by US and UK governments to develop secure federated learning algorithms for fraud prevention and COVID risk prediction with privacy guarantees.

FedSGD Attacks - Do gradient updates preserve privacy?

Gradient Inversion (from gradients to data):

- Server **passively** observes client gradients g_k
- Server uses **client gradients** and the **model** at time t f_{Θ_t} to **obtain client data**

Closed-form reconstruction (available closed-form formulas to go from gradient to data point):

R-gap: Recursive gradient attack on privacy, ICLR 2021, Zhu et. al https://arxiv.org/pdf/2010.07733.pdf

- For **batch size 1** and piecewise-linear NN (i.e., ReLU-based), one can generally reconstruct the input **exactly** from the gradient.
- For **batch size > 1** and the same assumptions one can reconstruct a data point that is a **linear combination of some true inputs** (different input combinations can produce the same gradient).

Key challenge: reconstruct the inputs in the batch for batch size > 1

FedSGD Attacks - Bigger Batch Sizes

Approximate reconstruction of individual inputs for batch size > 1 using prior information:

$$\underset{x^*}{\operatorname{argmin}} \underbrace{d(\nabla_\Theta \mathcal{L}(f_\Theta(x^*), y^*), g_k)}_{\substack{\textbf{Distance}\text{ between reconstructed gradient} \\ \text{on } f_\Theta \text{ and the true gradient } g_k}} + \alpha_{\text{reg}} \cdot \underbrace{\mathcal{R}(x^*)}_{\text{Domain specific }\textbf{prior}}$$

Elements of the attack:

- d - Commonly chosen to be L1, L2 or cosine distance between vectors
- $\mathcal{R}$ - Prior based on **domain-specific knowledge.** Different choices depend on the type of input (e.g. total variation for images). For tabular data: no prior, for text: perplexity.
- α_{reg} - Parameter **balancing** between **reconstruction quality** and **domain-specific knowledge**
- Optimized with gradient descent. **Initialization** x^* of matters, y^* typically reconstructed **separately**.

Bayesian Framework for Gradient Leakage, ICLR 2022, Balunovic et al. https://arxiv.org/pdf/2111.04706.pdf

FedSGD Attacks: Image Data Example

Original:

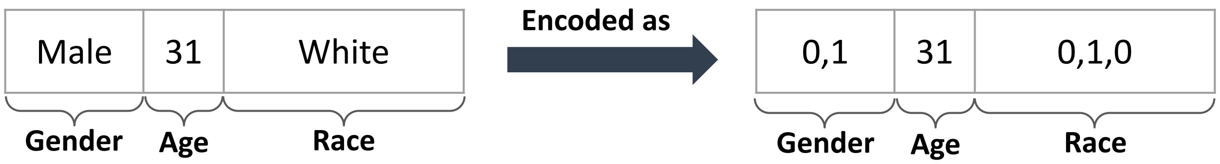
Recovered:

Reconstruction on Pretrained ResNet32-10 on CIFAR100 with batch size 100

(with cosine similarity distance and total variation prior)

Inverting Gradients - How easy is it to break privacy in federated learning?, Neurips 2020, Geiping et al. https://arxiv.org/pdf/2003.14053.pdf

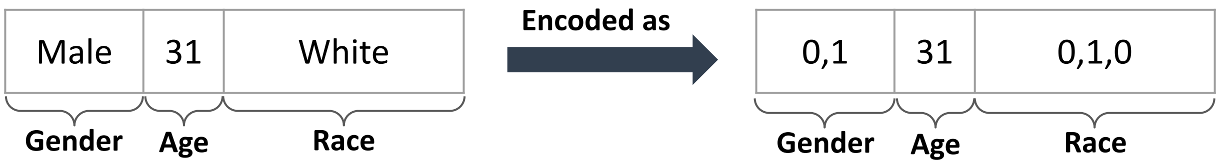
Federated Learning over Tabular Data



Tabular data:

- Contains both **categorical** and **continuous features.**
- Categorical data is **one-hot encoded.**
- Rest of the federated training is the same as discussed so far.

FedSGD Attacks on Tabular Data



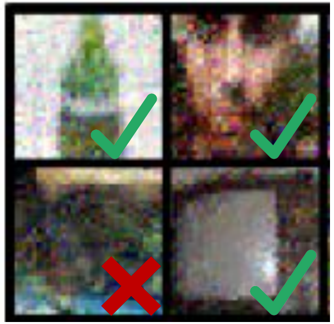
Reconstruction with One-Hot Encodings: $x^* = [\ \mathrm{softmax}(C_{\mathrm{gender}}), C_{\mathrm{age}}, \mathrm{softmax}(C_{\mathrm{race}})\ ]$

- For every categorical feature we introduce one continuous variable C, a vector of dimensionality same as the dimensionality of the one-hot encoding, e.g. C_{gender} is of dimension 2 and C_{race} is of dimension 3.
- Softmax enforces that the reconstructed one-hot encodings contain **only positive numbers** and **sum to 1**. It is a **continuous relaxation** of the one-hot constraint - makes continuous optimization **easier.**
- Now: optimize variables C using the same optimization problem as the one discussed for images.
- After optimization, the resulting x^* is **projected** to the closest one-hot encoding.

Data Leakage in Tabular Federated Learning, Vero et al. https://arxiv.org/pdf/2210.01785.pdf

Assessing Quality of Tabular Data Reconstructions

How do we know we managed to successfully reconstruct the data?



For image data, bad reconstructions look like random noise

Male	31	White
Male	31	Hispanic

Both 'look' correct but one is **wrong**

For tabular data, because of projection to one-hot encodings, we **cannot easily distinguish** good reconstructions from bad ones by simple inspection.

Challenge: All reconstructed tabular data looks plausible, even if incorrect

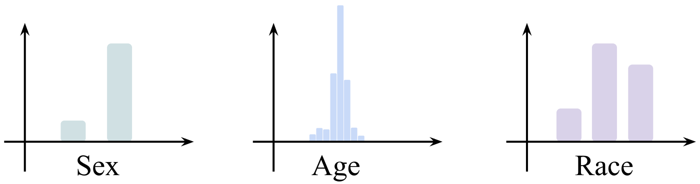
Data Leakage in Tabular Federated Learning, Vero et al. https://arxiv.org/pdf/2210.01785.pdf

Assessing Quality of Tabular Data Reconstructions

Key Observation: Correctly-reconstructed tabular cells are robust to random initializations

Proposed Solution:

- Assemble many independent reconstructions with different random **initializations**.
- Create (a normalized) **histograms for each tabular cell** from the reconstructions (after projection for categorical ones and binning the continuous ones).
- Measure the **entropy** of histograms. Low **entropy** corresponds to agreement between reconstructions (e.g. **peaky histograms**) and, thus, to correct reconstructions. We pick the most likely reconstruction if below entropy threshold.



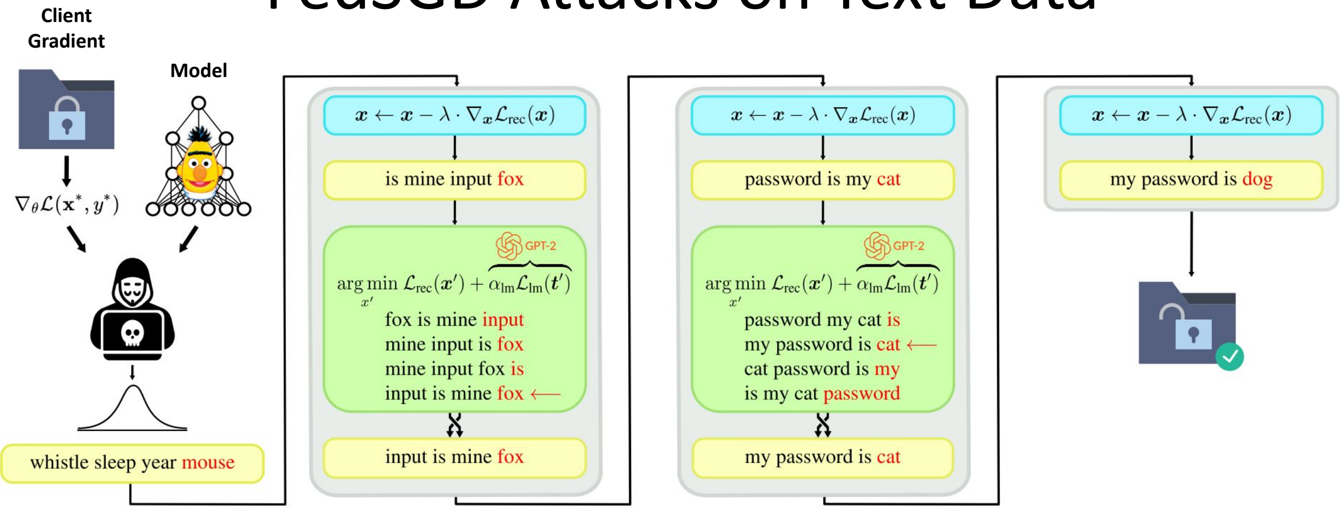
Robust & Correct

Robust & Correct

Not Robust & Incorrect

Data Leakage in Tabular Federated Learning, Vero et al. https://arxiv.org/pdf/2210.01785.pdf

FedSGD Attacks on Text Data



- **Blue Box** = *Continuous Optimization* **-** Optimization of the **gradient distance** $\mathcal{L}_{\text{rec}}(x) = d(\nabla_\Theta \mathcal{L}(f_\Theta(x), y^*), g_k)$ over the **word embeddings** x. Gets stuck at **local minima** due to **word order** being hard to change continuously. Similar to images but with no prior.
- **Yellow Box** = *Projection* - **Project** the currently optimized **word embeddings** x to the **closest words** t for federated model.
- **Green Box** = *Discrete optimization* **-** Pick the best **order** for the **embeddings** x based on a combination of the **gradient distance** $\mathcal{L}_{rec}$ and the **perplexity** $\mathcal{L}_{lm}$ measured by an **auxiliary language model** (e.g. GPT-2) of the projected words t. Allows the continuous optimization to **avoid local minima.**
- We assume the **number of words** in a sentence is **known**. In practice, we do ~20 'big optimization' steps.

LAMP: Extracting Text from Gradients with Language Model Priors, Neurips 2022, Balunovic et al. https://arxiv.org/abs/2202.08827

FedSGD Attacks - Text Data Example

		Sequence
CoLA	Reference	mary has never kissed a man who is taller than john.
	Simple GradInv	man seem taller than mary ,. kissed has john mph never
	Ours	mary has never kissed a man who is taller than john.
SST-2	Reference	i also believe that resident evil is not it.
	Simple GradInv	resident . or. is pack down believe i evil
	Ours	i also believe that resident resident evil not it .
Rotten Tomatoes	Reference	a well - made and often lovely depiction of the mysteries of friendship.
	Simple GradInv	- the friendship taken and lovely a made often depiction of well mysteries .
	Ours	a well often made - and lovely depiction mysteries of mysteries of friendship .

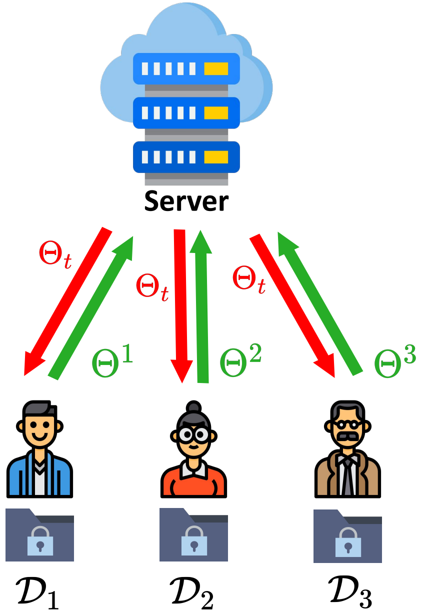
Simple GradInv = Continuous Optimization Only

Yellow Box = Correct Word, Wrong Order

Green Box = Correct Word, Correct Order

LAMP: Extracting Text from Gradients with Language Model Priors, Neurips 2022, Balunovic et al. https://arxiv.org/abs/2202.08827

FedAvg: A More Common Federated Learning Setup



Server aggregation

$$\Theta_{t+1} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \Theta^k$$

The server averages the client weight updates Θ^k

Client update

$\Theta^k_{1,0} \leftarrow \Theta_t$
for e in range(E):
  for b in range(B):
    $\{x^k_{e,b}, y^k_{e,b}\} \sim \mathcal{D}_k$
    $\Theta^k_{e,b} \leftarrow \Theta^k_{e,b-1} - \gamma \nabla_\Theta \mathcal{L}(f_{\Theta^k_{e,b-1}}(x^k_{e,b}), y^k_{e,b})$
  end for
end for
$\Theta^k \leftarrow \Theta^k_{E,B}$

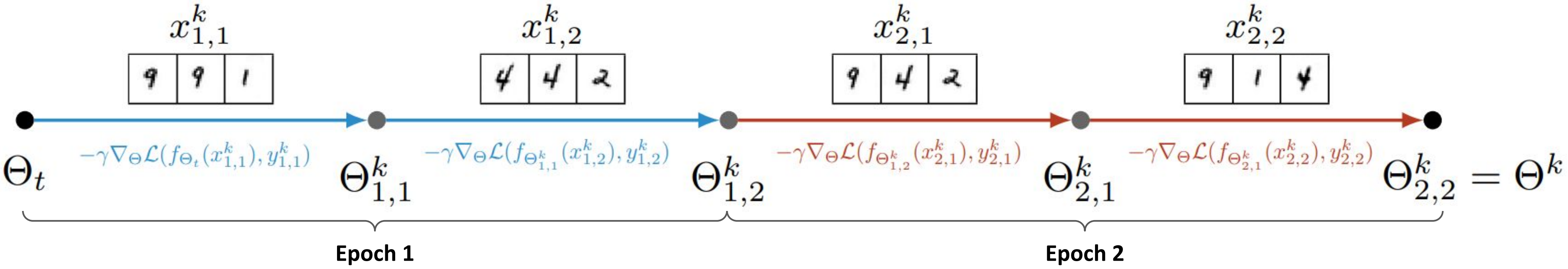
For E local epochs and B local batches per epoch the clients sample a **minibatch** of data $\{x^k_{e,b}, y^k_{e,b}\}$ from their dataset $\mathcal{D}_k$. For each minibatch the clients take a single **SGD step** and **update** their local model. Finally, the clients send the **network weights** at the last step to the server.

Pros: Requires much **less communication rounds** due to additional steps in clients.

Communication-Efficient Learning of Deep Networks from Decentralized Data, PMLR 2017, McMahan et al. https://arxiv.org/pdf/1602.05629.pdf

FedAvg Attacks: Do weight updates preserve privacy?

Example: Client with 6 images, takes Θ_t, does 4 local weight updates (2 epochs each with 2 batches of size 3) and shares Θ^k:



Client update

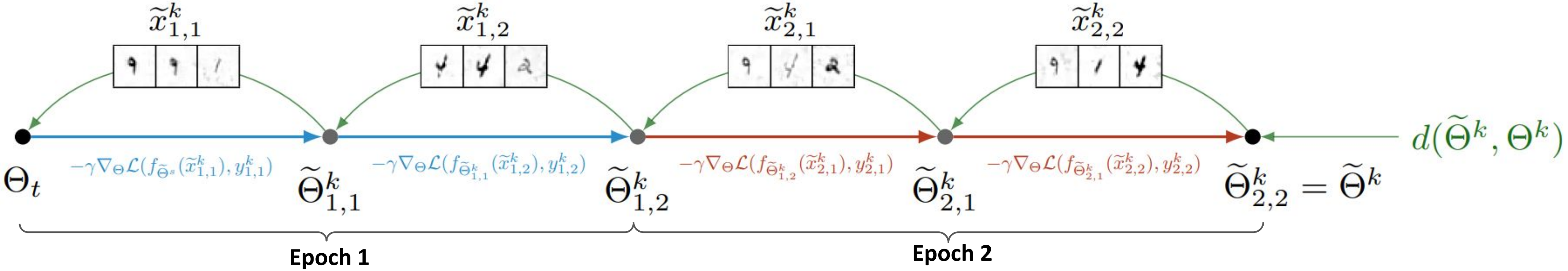
$\Theta^k_{1,0} \leftarrow \Theta_t$
for e in range(E):
  for b in range(B):
    $\{x^k_{e,b}, y^k_{e,b}\} \sim \mathcal{D}_k$
    $\Theta^k_{e,b} \leftarrow \Theta^k_{e,b-1} - \gamma\nabla_\Theta\mathcal{L}(f_{\Theta^k_{e,b-1}}(x^k_{e,b}), y^k_{e,b})$
  end for
end for
$\Theta^k \leftarrow \Theta^k_{E,B}$

FedAvg Attack Challenges:

- Attacker **does not observe** the weights at **intermediate steps** of the $\Theta^k_{e,b}$ client optimization, **only the final update** Θ^k.
- Order in which data points $\{x^k_{e,b}, y^k_{e,b}\}$ are fed to the network in the clients matters for the computed update Θ^k.
- Attacker does not observe this order.

Data Leakage in Federated Averaging, TMLR 2022, Dimitrov et al. https://arxiv.org/pdf/2206.12395.pdf

FedAvg Attacks: Differentiation through FedAvg Simulation



FedAvg Attack - Differentiation through FedAvg Simulation:

- Create and initialize **individual** optimization variables **per batch** $\tilde{x}^k_{e,b}$.
- **Simulate** the FedAvg update on $\tilde{x}^k_{e,b}$ to produce intermediate client weights $\widetilde{\Theta}^k_{e,b}$.
- Calculate **distance** $d(\widetilde{\Theta}^k, \Theta^k)$ between the final **simulated weights** $\widetilde{\Theta}^k$ and the **true weights** Θ^k.
- Using **automatic differentiation** compute the derivative of $d(\widetilde{\Theta}^k, \Theta^k)$ w.r.t. $\tilde{x}^k$ (green arrows).
- Use it to **optimize** the reconstructions $\tilde{x}^k$.
- It is possible that the images reconstructed in one epoch **differ** from the images reconstructed in another epoch.

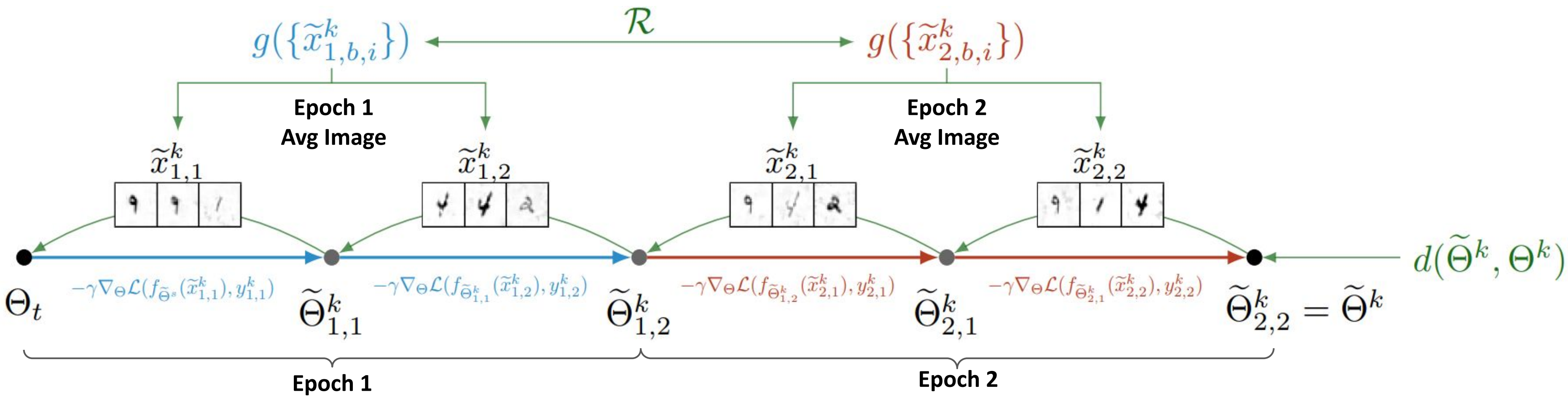
Data Leakage in Federated Averaging, TMLR 2022, Dimitrov et al. https://arxiv.org/pdf/2206.12395.pdf

FedAvg Attacks: Order-Invariant Prior

Order-invariant prior: One way to enforce that the **set of reconstructed images** in different epochs **match.**

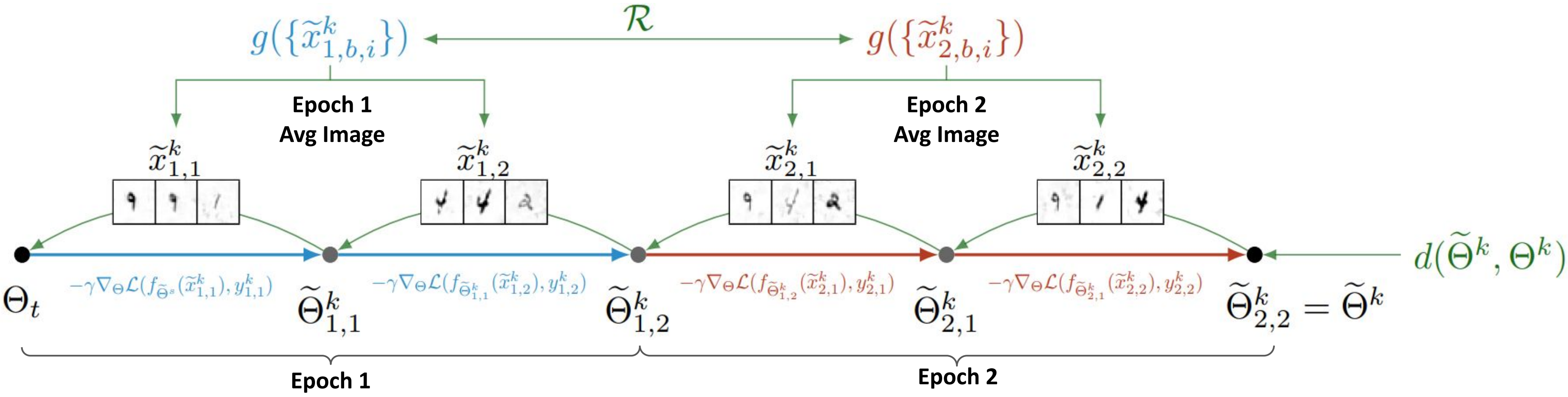
Idea: Add (weak) prior $\mathcal{R}$ (e.g. some norm) that enforces the average input $g(\{\tilde{x}^k_{e,b}\}) = \frac{1}{|\mathcal{D}_k|}\sum_{b,i} \tilde{x}^k_{e,b,i}$ per epoch to match across epochs. Note: idea applies beyond images.

Further note: One could possibly play with differentiable constraints (next lecture).



Data Leakage in Federated Averaging, TMLR 2022, Dimitrov et al. https://arxiv.org/pdf/2206.12395.pdf

FedAvg Attacks: End-to-end Optimization Problem



$$\underset{\widetilde{x}^k}{\operatorname{argmin}} \; \underbrace{d(\widetilde{\Theta}^k, \Theta^k)}_{} + \alpha_{\text{reg}} \cdot \underbrace{\frac{1}{E^2}\sum_{e_1,e_2} \mathcal{R}(g(\{\widetilde{x}^k_{e_1,b}\}), g(\{\widetilde{x}^k_{e_2,b}\}))}_{}$$

Distance between the **simulated weights** $\widetilde{\Theta}^k$ and the **true weights** Θ^k

Average distance between the **average images** at every pair of epochs e_1 and e_2

Data Leakage in Federated Averaging, TMLR 2022, Dimitrov et al. https://arxiv.org/pdf/2206.12395.pdf

FedAvg Attacks: Results

5 batches (size 10) for 10 epochs on CIFAR-100

Data Leakage in Federated Averaging, TMLR 2022, Dimitrov et al. https://arxiv.org/pdf/2206.12395.pdf

Summary

- We presented an overview of four popular privacy attack vectors in ML - Model Stealing, Model Inversion, Data Extraction, and Membership Inference.
- We outlined some important regulations that aim to protect client's privacy - Unlearning and Data Minimization.
- We introduced Federated Learning as a way to avoid sharing data when training using many data sources and reviewed the most common algorithms - FedSGD and FedAvg.
- We discussed that Federated Learning updates **alone** do not preserve privacy and discussed in detail Gradient Leakage Attacks for several input domains - images, tabular, and text data.