Introduction to Federated Machine Learning

Usama Zafar

Department of Information Technology,
Uppsala University

May 8, 2025

SciML: Scientific Machine Learning @ IT/UU

Applied Machine Learning	Distributed Computing Infrastructures	Scientific Computing
• Federated Machine Learning • Algorithms for FedML • FedML security, Blockchain	• Applications, interactive computing • Parallel, peer-to-peer streaming • Intelligent storage backends	• Hierarchical analysis of spatial and temporal image data (HASTE) • Continuous analytics

http://sciml.se/

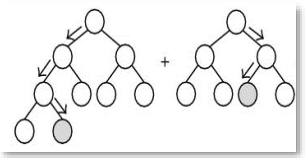
Federated Machine Learning (FedML)

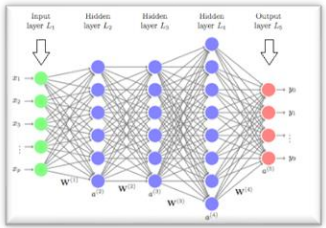
Machine Learning

Focused on building systems that **learn from data**, **identify patterns**, and **make decisions** with minimal human intervention.

Supervised Learning

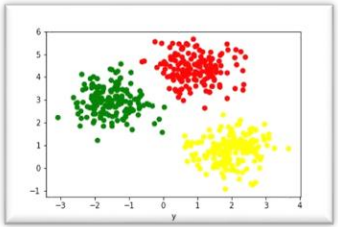
- Regression
- Decision Tree
- Random Forest
- SVM
- Deep Learning





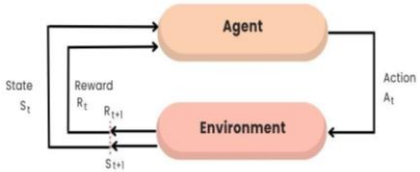
Unsupervised Learning

- kNN
- K-Means
- EM Algorithm



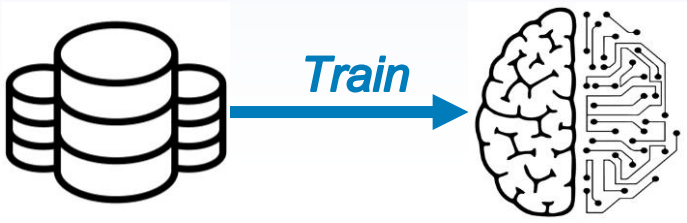
Reinforcement Learning

- Policy Optimization
- Q-Learning
- Model-Based RL



Images taken from Google

Supervised Learning



Data

Model

$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$

$x_i \in \mathbb{R}^d$

f_{θ^*}

Objective:

- To learn a function $f_\theta : \mathbb{R}^d \to \mathbb{R}$ (or $\mathbb{Z}$) that minimizes a loss function $\ell(x_i, y_i; \theta)$. i.e.

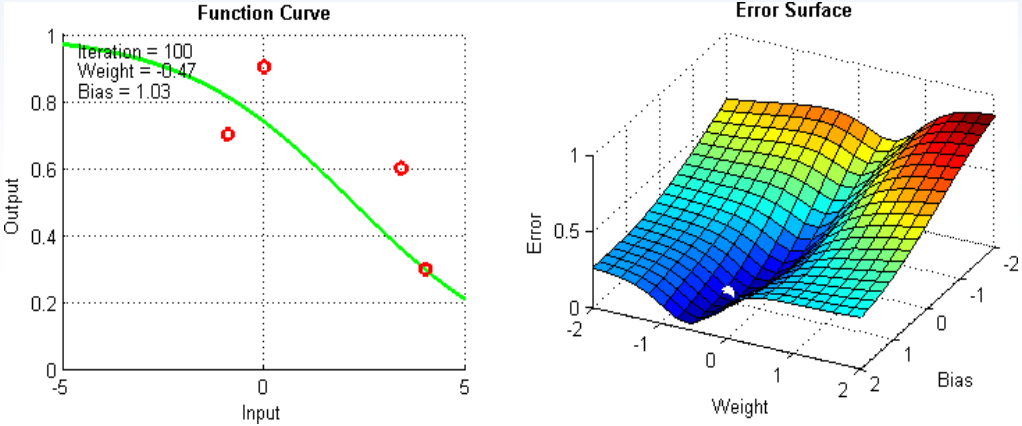
$$\theta^* = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i; \theta)$$

For example,

- Mean Squared Error (MSE) for regression:

$$\ell(x_i, y_i; \theta) = (f_\theta(x_i) - y_i)^2$$

Stochastic Gradient Descent (SGD)



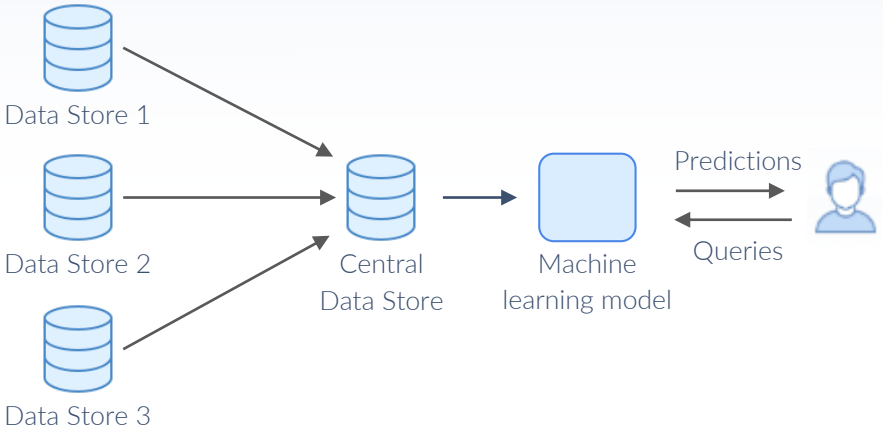
- Choose an initial vector of parameters θ^0 and learning rate η.
- For $j = 1, \ldots, T$, do:

$$\theta^{j+1} := \theta^j - \eta \nabla \ell_j(\theta)$$

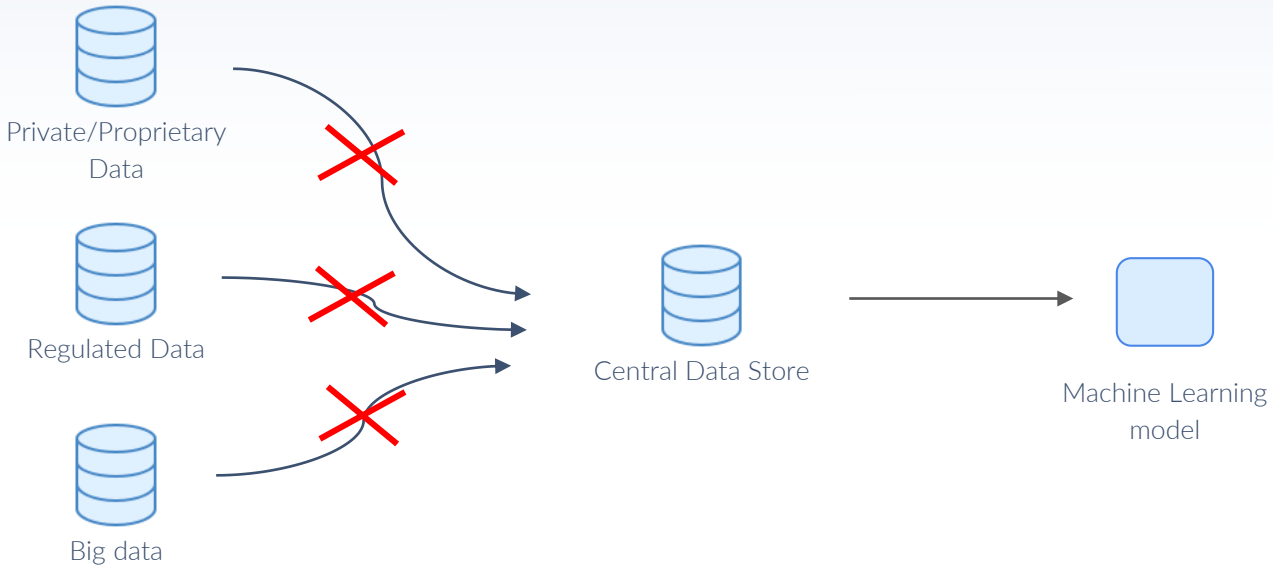
The centralized ML paradigm

1. Centralize data from different sources (data lake, cloud).

2. Create ML model using centralised data (cluster computing)



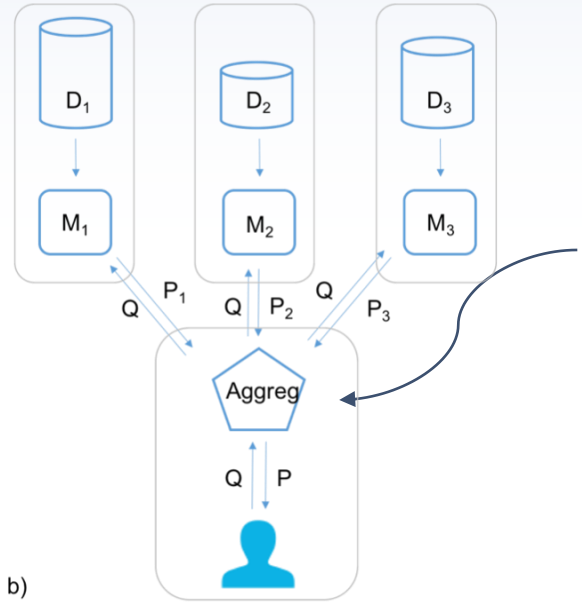
But in many cases we cannot move data



How can parties construct joint ML models without sharing/pooling data?

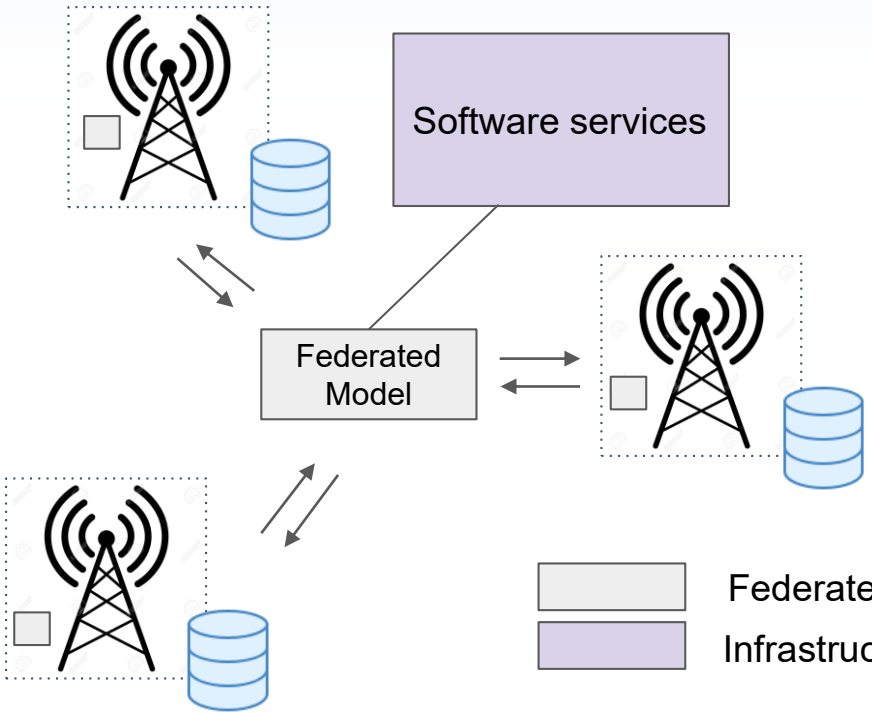
Federated Machine Learning

1. Train local machine learning model on local/private data.



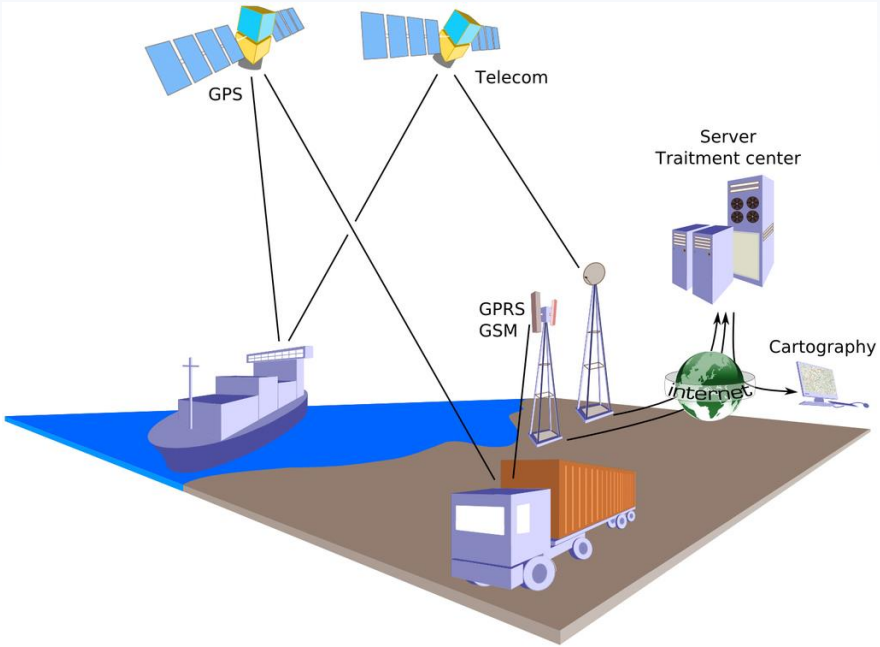
2. Combine local model updates into a global, federated model.

Smart software on top of decentralized infrastructure/instruments



- Let's a supplier of physical infrastructure/instruments build smart software to support all clients.
- Calibration, predictive maintenance etc.
- Customer A's data is never shared with Customer B, or with the supplier.
- High-value, unique software offering for those using the FedML services.

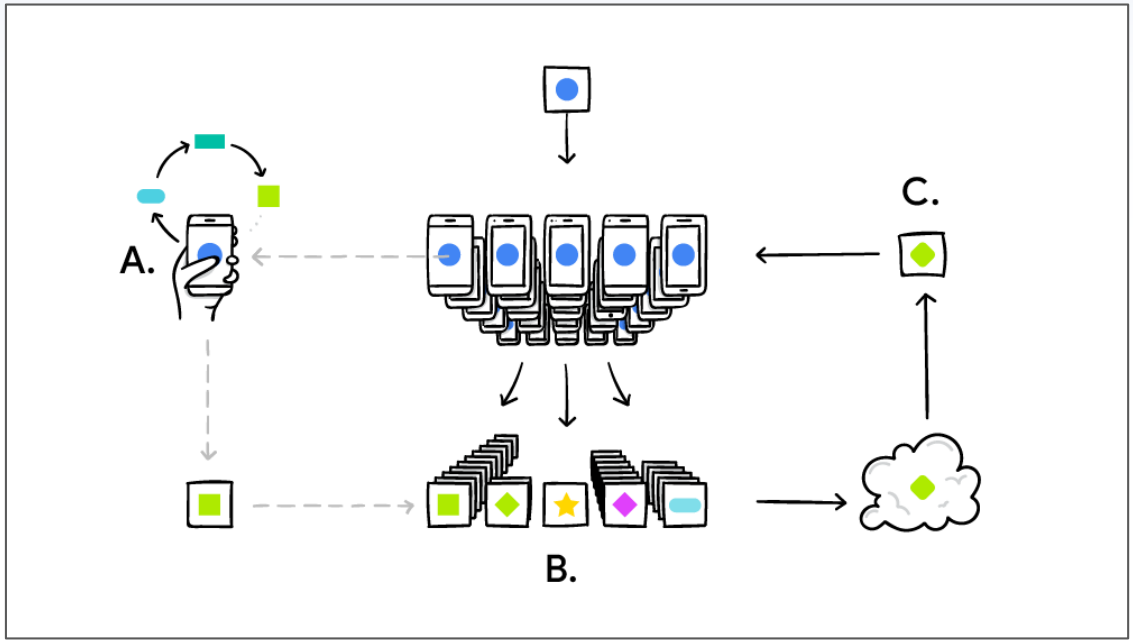
Integrity preserving fleet management



- Model driver/staff behavior without compromising their integrity.
- Big data, poor connectivity

Example, FedML on gboard

- Local model for search suggestion, with context and whether suggestion was clicked
- On device the history is processed, and then only a model update is suggested to Google
- Based on *Federated Averaging*



https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

Federated Averaging

1. Out of K alliance members/clients, pick a fraction C to do a global model update.

1. Perform E epochs of SGD on local minibatch of size B.

1. Average locally updated weights.

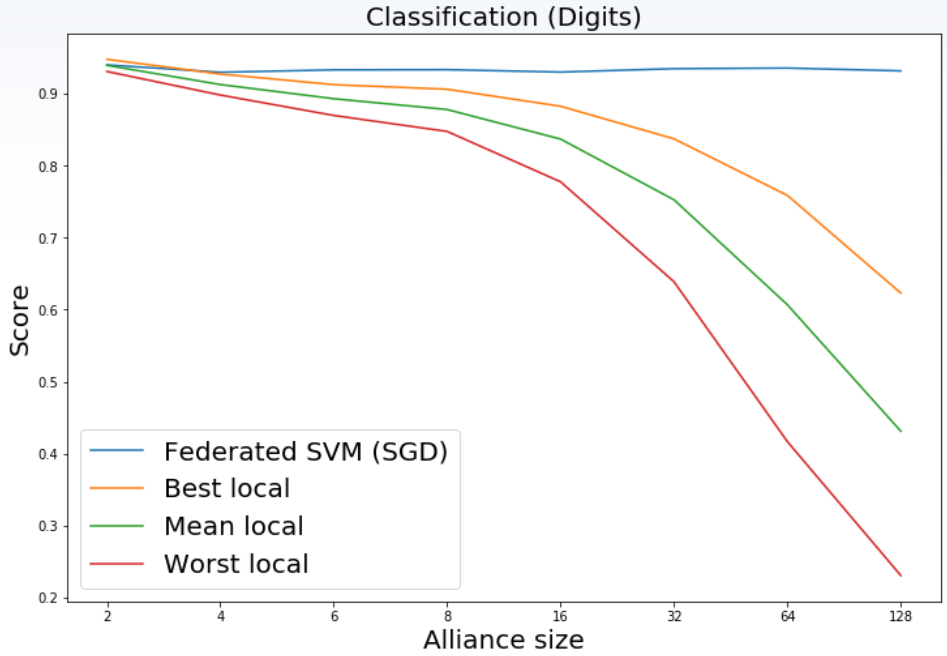
Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

From McMahan et al.
https://arxiv.org/abs/1602.05629

Key benefits of Federated Learning



Promises to let parties collaborate to build stronger models than what could be attained the parties in isolation.

- This examples uses incremental learning of linear models to do FedML.
- Stochastic Gradient Descent.
- One of many possible approaches to decentralized model construction.

Research challenges

FedML is a research area that spans many differents areas of computer science and mathematics.

Scalability and ML performance

How do we (re)design algorithms and frameworks to scale out to the fog and edge?

Adversarial ML

How can we make the system robust to dishonest members and external threats?

Decentralized computation

How can we do FedML without a third-party trust provider?

Privacy, Security and Trust in FedML

PRIVACY

SECURITY

TRUST

Data privacy

- In federated machine learning environment, data never leaves the premises. Only the model parameters (or weights) are shared between federated members
- Data owners have complete control over the datasets
- The training of incoming models can be offline or online within the data owner’s secure environment

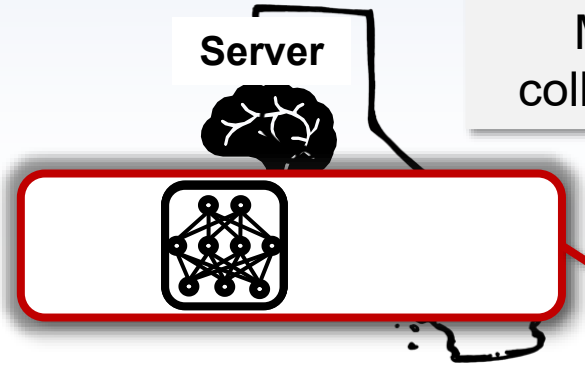
Federated Machine Learning

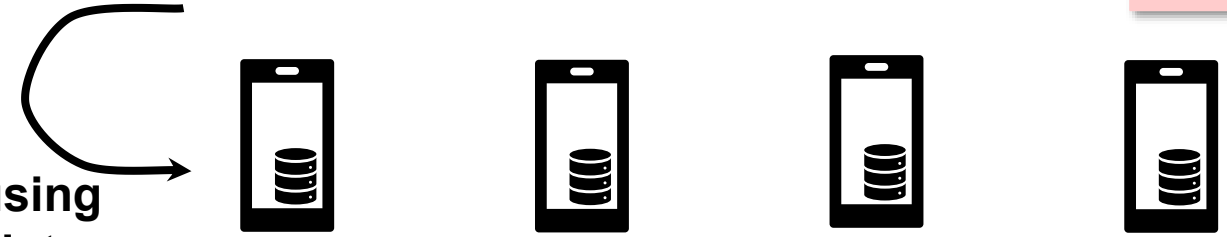
In each iteration

Download global model from server

Send updated model back to server for aggregation

Train using local data

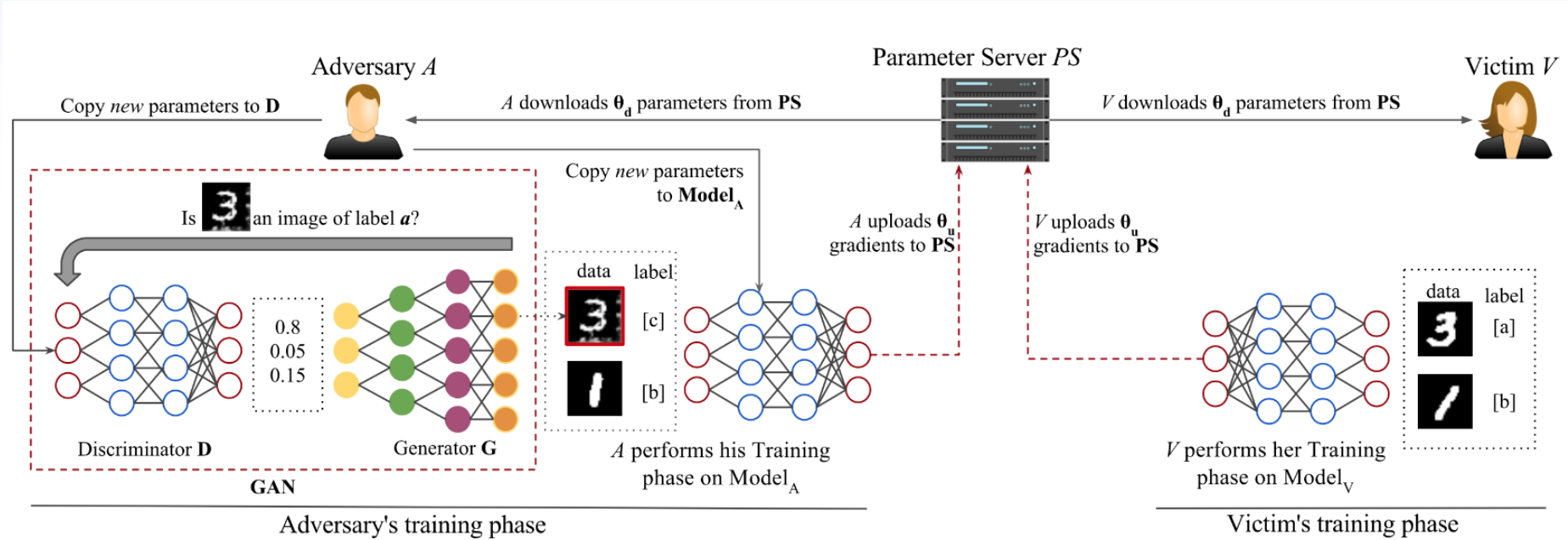




Multiple clients want to collaboratively train a model

Sharing updates rather than data...
Is privacy guaranteed?

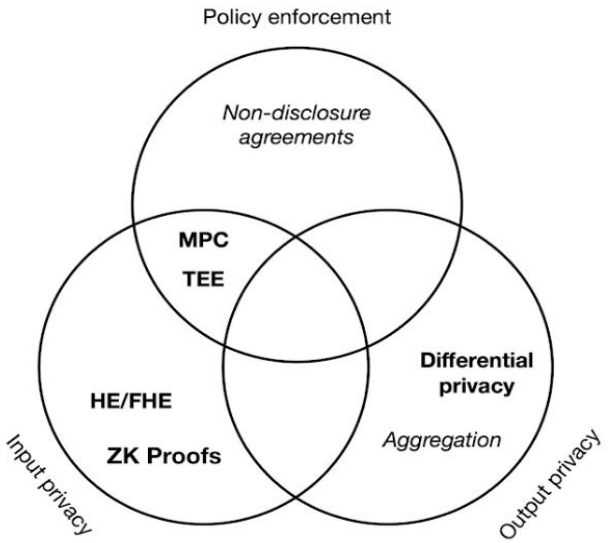
Information Leakage*



*B. Hitaj, G. Ateniese & F. Perez-Cruz. **Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning**.

Privacy-preservation in FedML

- Input privacy simplified since data stays locally (handled according to local policies)
- Output privacy - depends on the algorithm, how easy it is to invert the model etc.
- What can be learned from the coordination of computation?



UN Handbook for Privacy-Preserving Techniques:
https://docs.google.com/document/d/1GYu6UJI81jR8LgooXVDsYk1s6FlM-SbOvo3oLHglFhY/mobilebasic

Enhancing Privacy in FedML

Apart from “standard security” (data at rest and in transit), a number of techniques can be used to enhance privacy:

- Differential privacy (add noise to data)
- Homomorphic encryption (compute directly on encrypted data)
- Secure multiparty computation (emulate a trusted third party)
- Secure enclaves (a hardware solution to private computations)

Differential Privacy can protect against inference attacks

- Rigorous statistical technique to measuring and minimizing the privacy leakage from a statistical database.
- Add controlled noise to function we want to compute (e.g. Laplace mechanism).
- An interesting tradeoff between accuracy and the number of allowed queries to the model given epsilon.
- Related to the sensitivity of the function

Explored for FedML by e.g. Papernot et. al. https://arxiv.org/abs/1802.08908

Homomorphic Encryption

Computations directly on encrypted data producing encrypted results.

- Outsourced secure computations.
- “Secure pooling of data”
- Still not feasible for real world ML tasks.

- In FedML we do not need to outsource computations, except for parts such as secure aggregation of model weights / scores etc. For those parts of the algorithm, HE can be a viable option.

- SEAL (Microsoft): https://github.com/Microsoft/SEAL
- HELib (IBM): https://github.com/shaih/HElib
- PALISADE: https://git.njit.edu/palisade/PALISADE

Secure Multiparty Computation
(Secure computation, MPC, privacy preserving computation)

Parties P_1 .. P_N each with private data x_1,..x_N want to compute y = f(x_1, .., X_N)

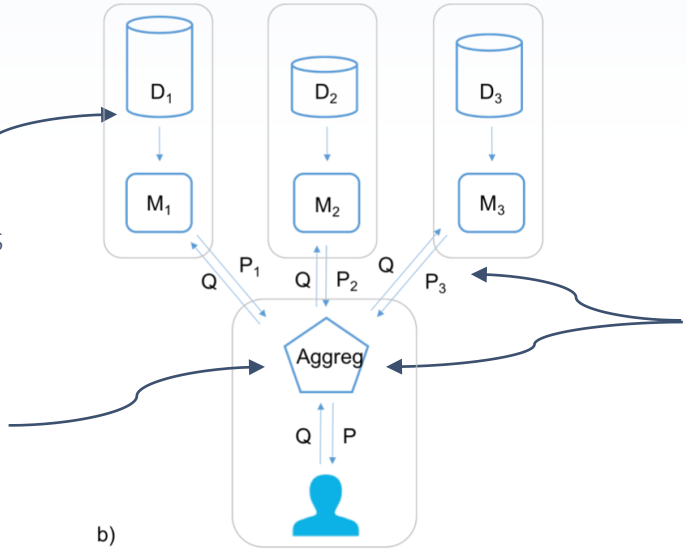
- No trust amongst parties P
- Do not want to trust a third party to compute f
- MPC deals with protocols to emulate a trusted third party.
- Highly active area of research, hard problem for large N and large fraction of dishonest members.

In FedML, see e.g. the PySyft project, MPC in PyTorch:
https://github.com/OpenMined/PySyft

Differential Privacy & Homomorphic encryption in FedML

Differential privacy

- add noise to data (protects against inference attacks)



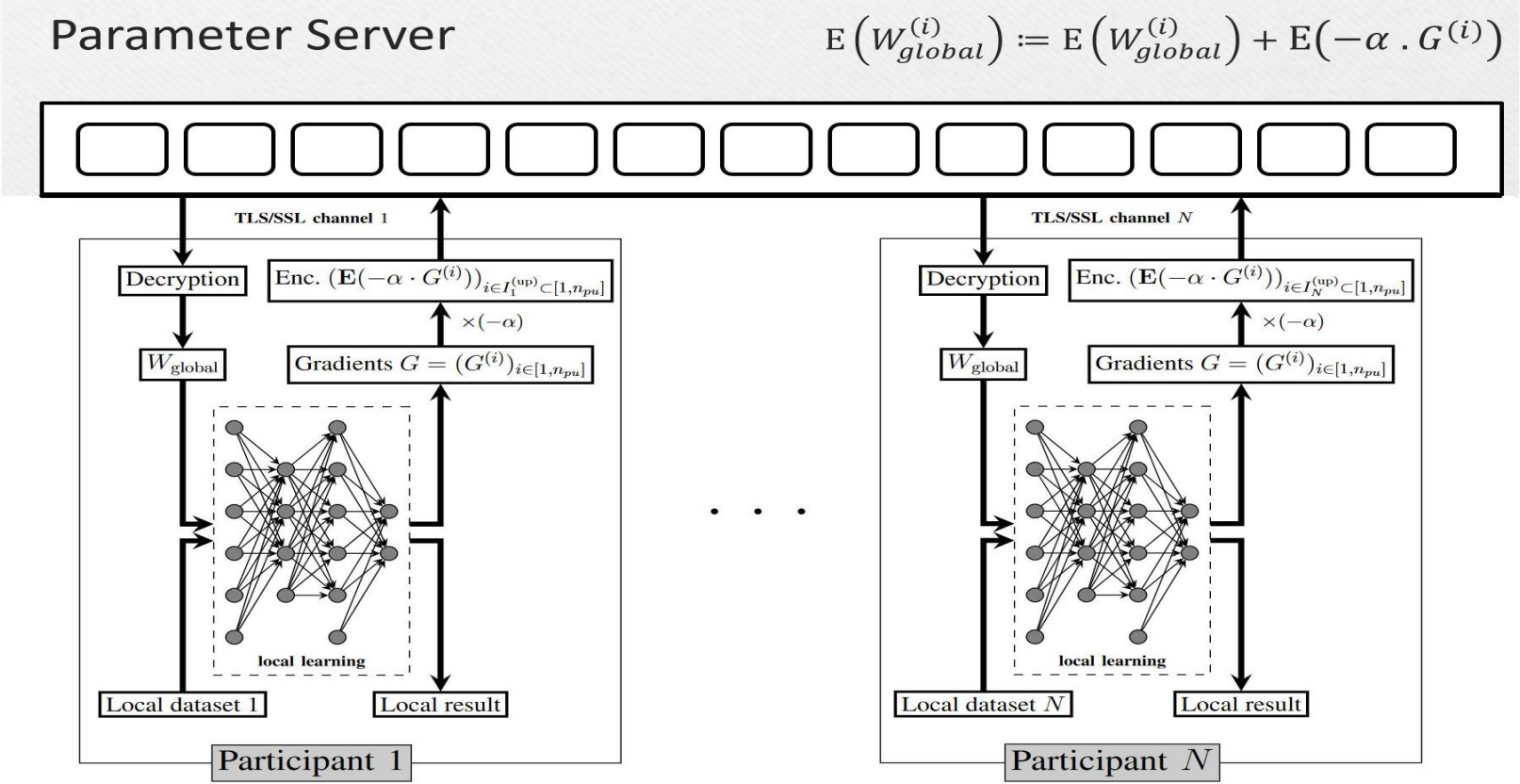
Homomorphic encryption

- Methods work on encrypted data

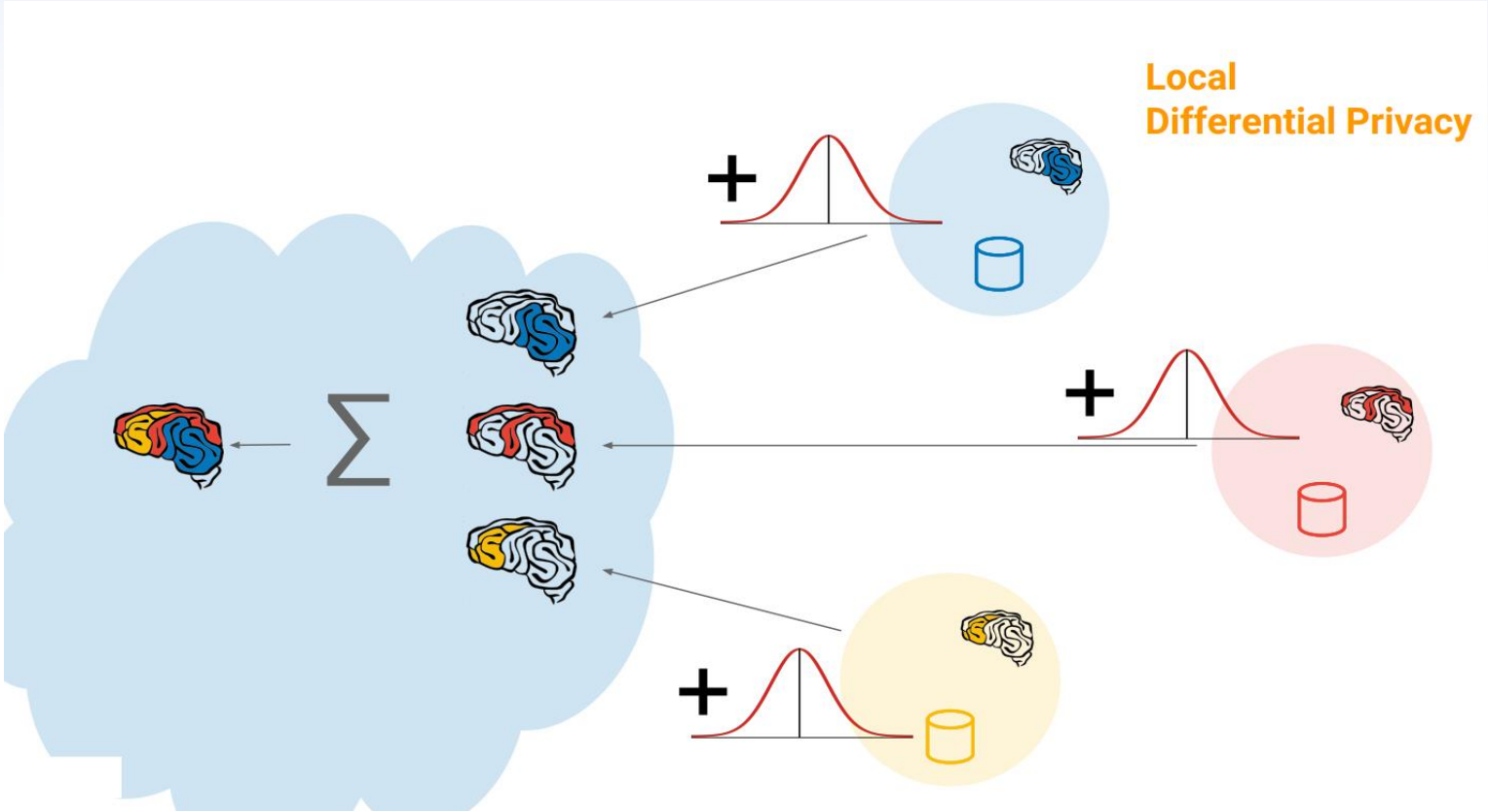
Secure multiparty computation

- Aggregate/compute without a third party trust provider/server.

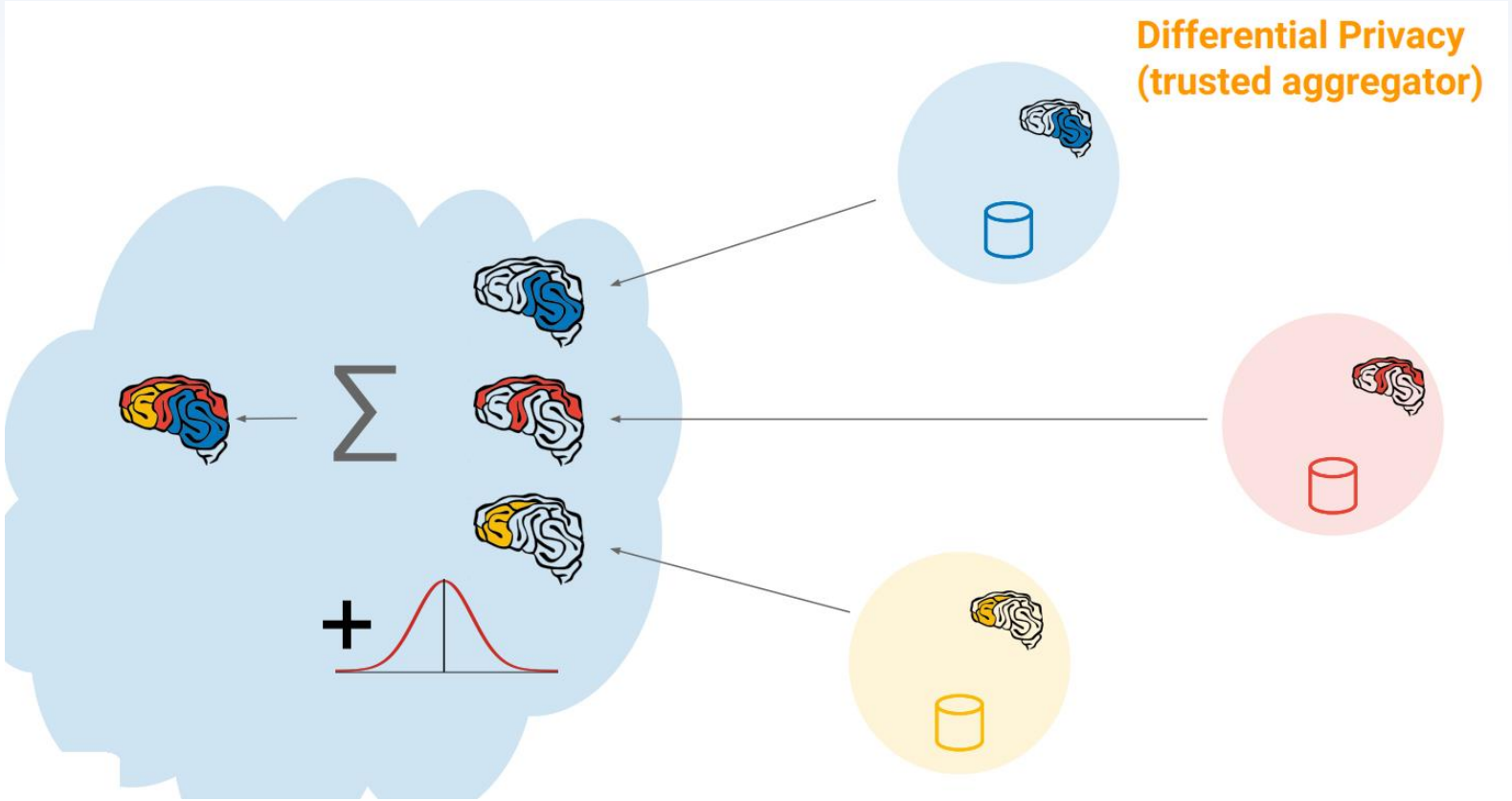
Homomorphic Encryption in FedML



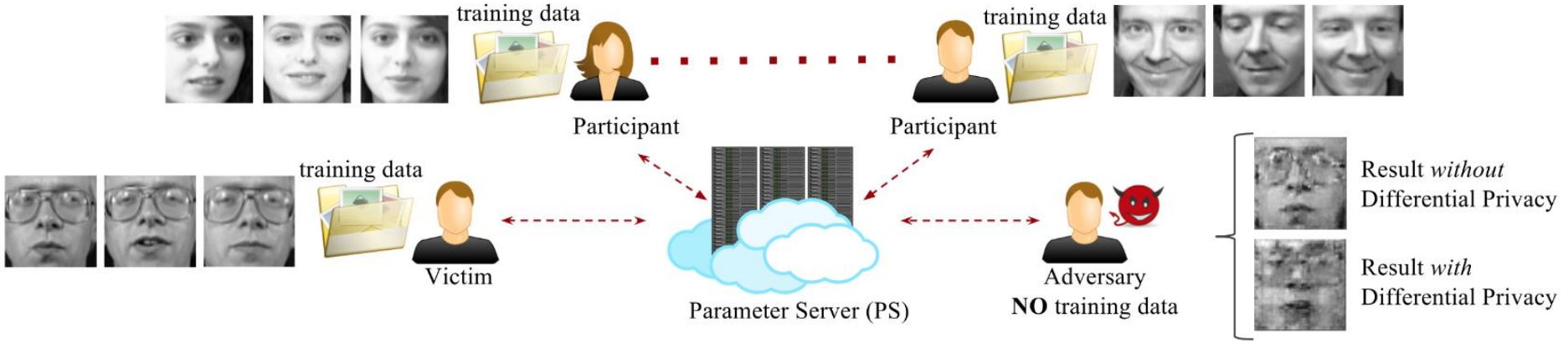
Differential Privacy in FedML



Differential Privacy in FedML



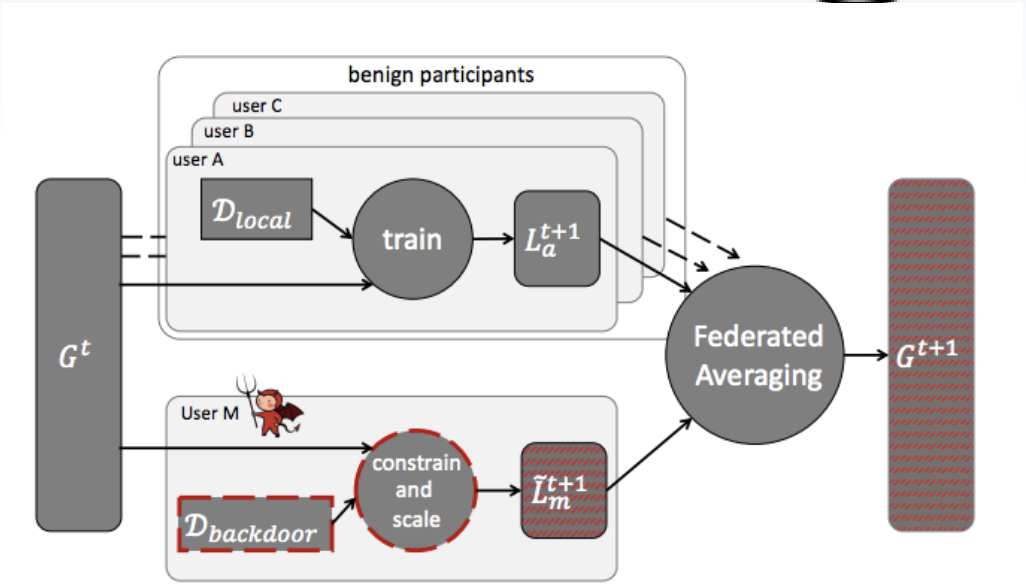
Information Leakage with DP



*B. Hitaj, G. Ateniese & F. Perez-Cruz. ***Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning***.

Security Challenges in FedML

- Big threat to a FedML comes from within the alliance / from compromized members.
- Large alliances can be expected to be relatively robust to data poisoning attacks.
- Bagdasaryan et al. shows how their proposed approach of model replacement can efficiently introduce backdoors in a global model.
- *Secure aggregation/MPC makes it impossible to detect a malicious model update, and who submitted it!*

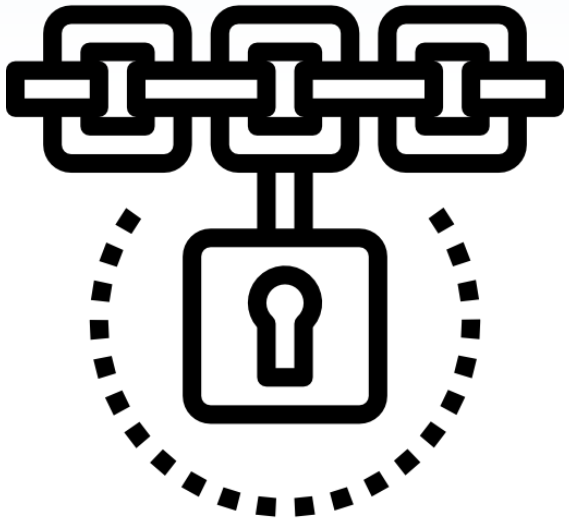


Bagdasaryan et al. *How to backdoor federated learning (2019)*
https://arxiv.org/pdf/1807.00459.pdf

Trust building mechanisms for FedML

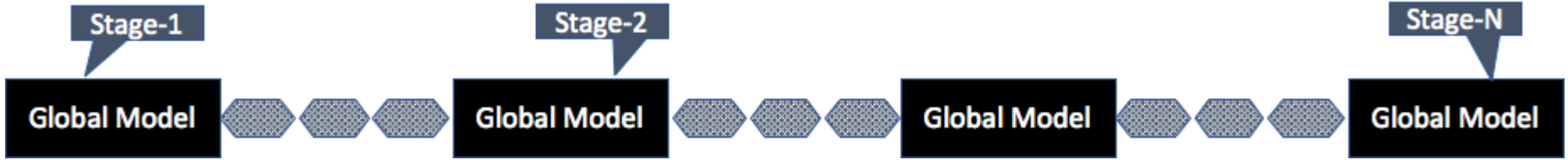
- FedML is inherently a distributed system with full control over the local environment
- Less or zero control over distributed datasets
- Contributions from different federated members can make or break the global model
- A transparent and efficient FedML framework allows different parties to work together

Using Blockchain Technology?



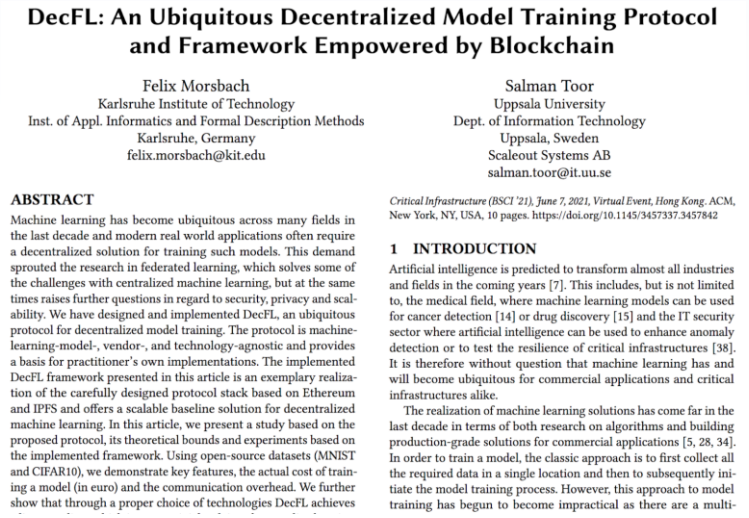
Blockchain and FedML

- Ongoing work to design a platform for FedML that will hold features of the Blockchain technology
- The aim will be to provide security, auditability and checkpointing for global model training
- Should allow different stakeholders to jointly train models in a more transparent and secure manner



DecFL: An Ubiquitous Decentralized Model Training Protocol and Framework Empowered by Blockchain

https://dl.acm.org/doi/abs/10.1145/3457337.3457842?casa_token=W48W949L3owAAAAA:EW9q9eJbocNsmJypdIILJa7nNZ7wk1JLvUm9DdbKU6mRPx4T_hfxOHXFOoH06ZwhXdMhT_m5jLK7aQ



DecFL: An Ubiquitous Decentralized Model Training Protocol and Framework Empowered by Blockchain

Felix Morsbach
Karlsruhe Institute of Technology
Inst. of Appl. Informatics and Formal Description Methods
Karlsruhe, Germany
felix.morsbach@kit.edu

Salman Toor
Uppsala University
Dept. of Information Technology
Uppsala, Sweden
Scaleout Systems AB
salman.toor@it.uu.se

ABSTRACT

Machine learning has become ubiquitous across many fields in the last decade and modern real world applications often require a decentralized solution for training such models. This demand sprouted the research in federated learning, which solves some of the challenges with centralized machine learning, but at the same times raises further questions in regard to security, privacy and scalability. We have designed and implemented DecFL, an ubiquitous protocol for decentralized model training. The protocol is machine-learning-model-, vendor-, and technology-agnostic and provides a basis for practitioner's own implementations. The implemented DecFL framework presented in this article is an exemplary realization of the carefully designed protocol stack based on Ethereum and IPFS and offers a scalable baseline solution for decentralized machine learning. In this article, we present a study based on the proposed protocol, its theoretical bounds and experiments based on the implemented framework. Using open-source datasets (MNIST and CIFAR10), we demonstrate key features, the actual cost of training a model (in euro) and the communication overhead. We further show that through a proper choice of technologies DecFL achieves

Critical Infrastructure (BSCI '21), June 7, 2021, Virtual Event, Hong Kong. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3457337.3457842

1 INTRODUCTION

Artificial intelligence is predicted to transform almost all industries and fields in the coming years [7]. This includes, but is not limited to, the medical field, where machine learning models can be used for cancer detection [14] or drug discovery [15] and the IT security sector where artificial intelligence can be used to enhance anomaly detection or to test the resilience of critical infrastructures [38]. It is therefore without question that machine learning has and will become ubiquitous for commercial applications and critical infrastructures alike.

The realization of machine learning solutions has come far in the last decade in terms of both research on algorithms and building production-grade solutions for commercial applications [5, 28, 34]. In order to train a model, the classic approach is to first collect all the required data in a single location and then to subsequently initiate the model training process. However, this approach to model training has begun to become impractical as there are a multi-

What does it take to build a production federated machine learning system ?

- Decentralized computing / fog computing
- Information security/systems security expertise
- Trust provider (third-party or decentralized protocol)
- Machine learning algorithms adapted to the decentralized case
- Protection against adversarial ML
  - Data poisoning
  - Inference attacks
  - …

A considerable increase in system and developer complexity compared to the standard paradigm!

Thank you for listening!