ECAI-2025
October 25–30, 2025
Bologna, Italy

UNIVERSITÀ
DI TORINO

Experimenting with Federated Learning: implementing state-of-the-art methods with **fluke**

Bruno Casella, Samuele Fonio, Mirko Polato
Department of Computer Science, University of Turin, Italy
name.surname@unito.it

SCAN ME

28th European Conference on Artificial Intelligence - ECAI2025, Bologna

Outline

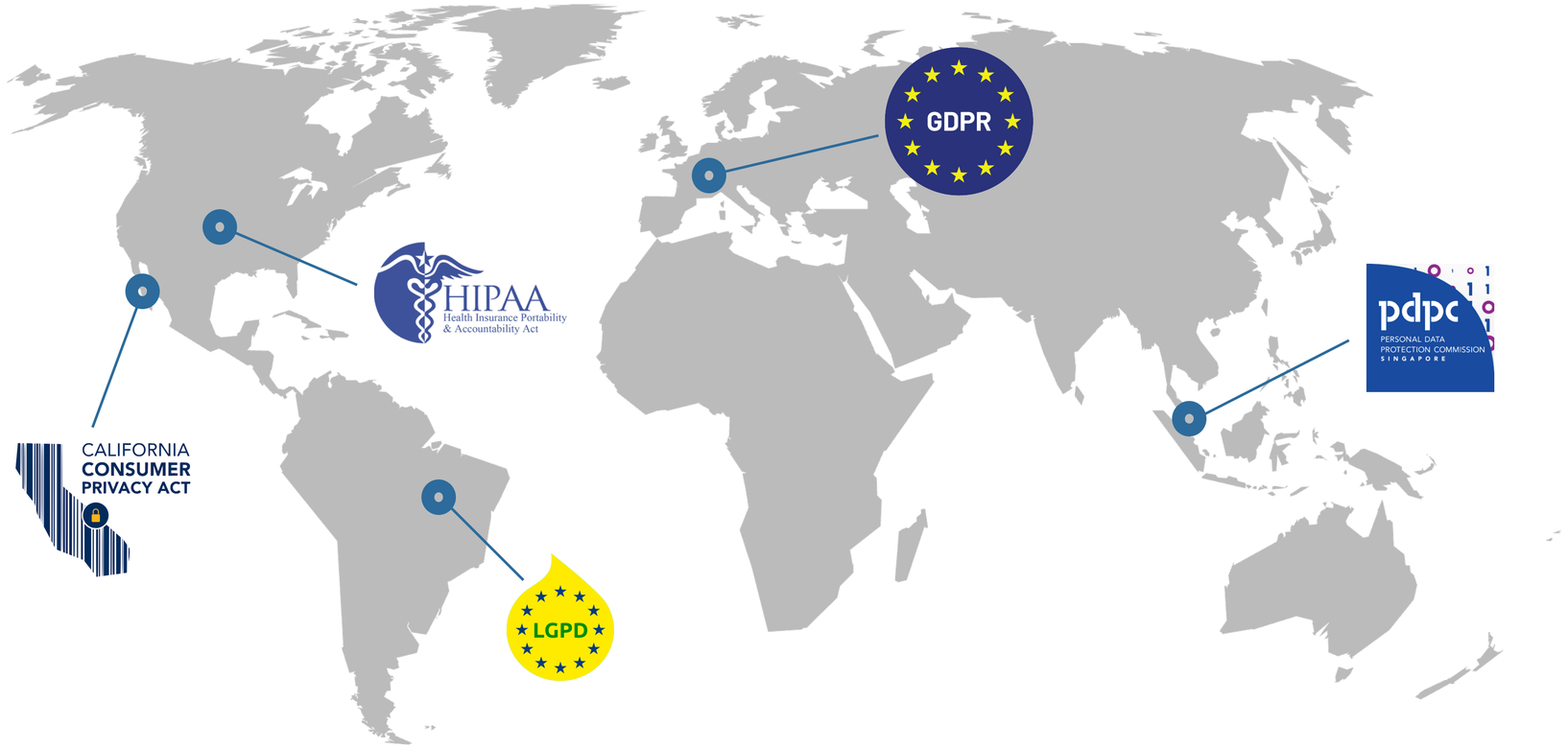
- **Introduction to Federated Learning**: challenges, applications, algorithms
- **Introduction to fluke**: motivation, goals, design & functionalities
- **Hands-on-session**: setup, run & use cases

Introduction to
Federated Learning

Federated Learning

Motivation: Data Sovereignty and Compliance

Recent **privacy regulations** (e.g., GDPR) restrict how data can move across borders or be stored.

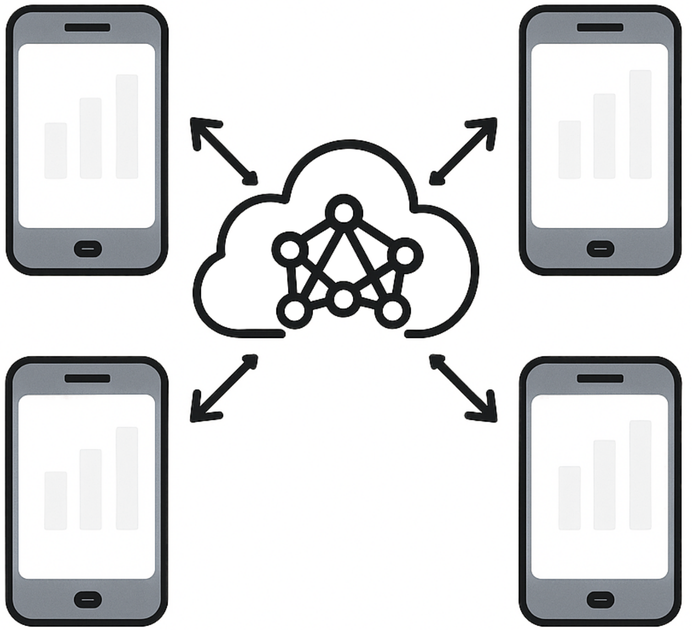


Federated Learning

Motivation: Security, privacy & leveraging distributed data

- In standard ML, the need of centralizing the data **may exposes sensitive data** (e.g., e.g., personal messages, medical records, financial data).

- Many real-world datasets are **naturally distributed** (e.g., smartphones, IoT sensors, hospitals).



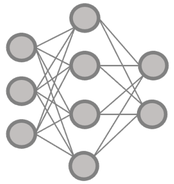
Federated Learning

Privacy-preserving distributed machine learning approach

Federated Learning (FL) is a machine learning setting where **multiple entities (clients) collaborate** in solving a machine learning problem, under the coordination of a central server. **Each client's raw data is stored locally and not exchanged or transferred**; instead, focused updates intended for immediate aggregation are used to achieve the learning objective.

Federated Learning

Parties involved and general idea



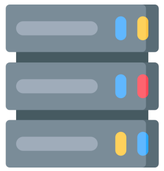
The model is what is exchanged

The model is what is transferred between the parties

Iteratively updated

The learning procedure is divided into “rounds” where a new version of the model is computed

Machine Learning model



Initialisation

The Server initialise the learning process

Orchestration

The Server orchestrates the learning process and acts as a centralisation point

Aggregation

The Server updates the model aggregating the clients’ models

Server (aggregator)

Privacy

The clients’ data is only used by the owner and never transferred to anyone

Local training

The local data is used to perform part of the learning locally

Model sharing

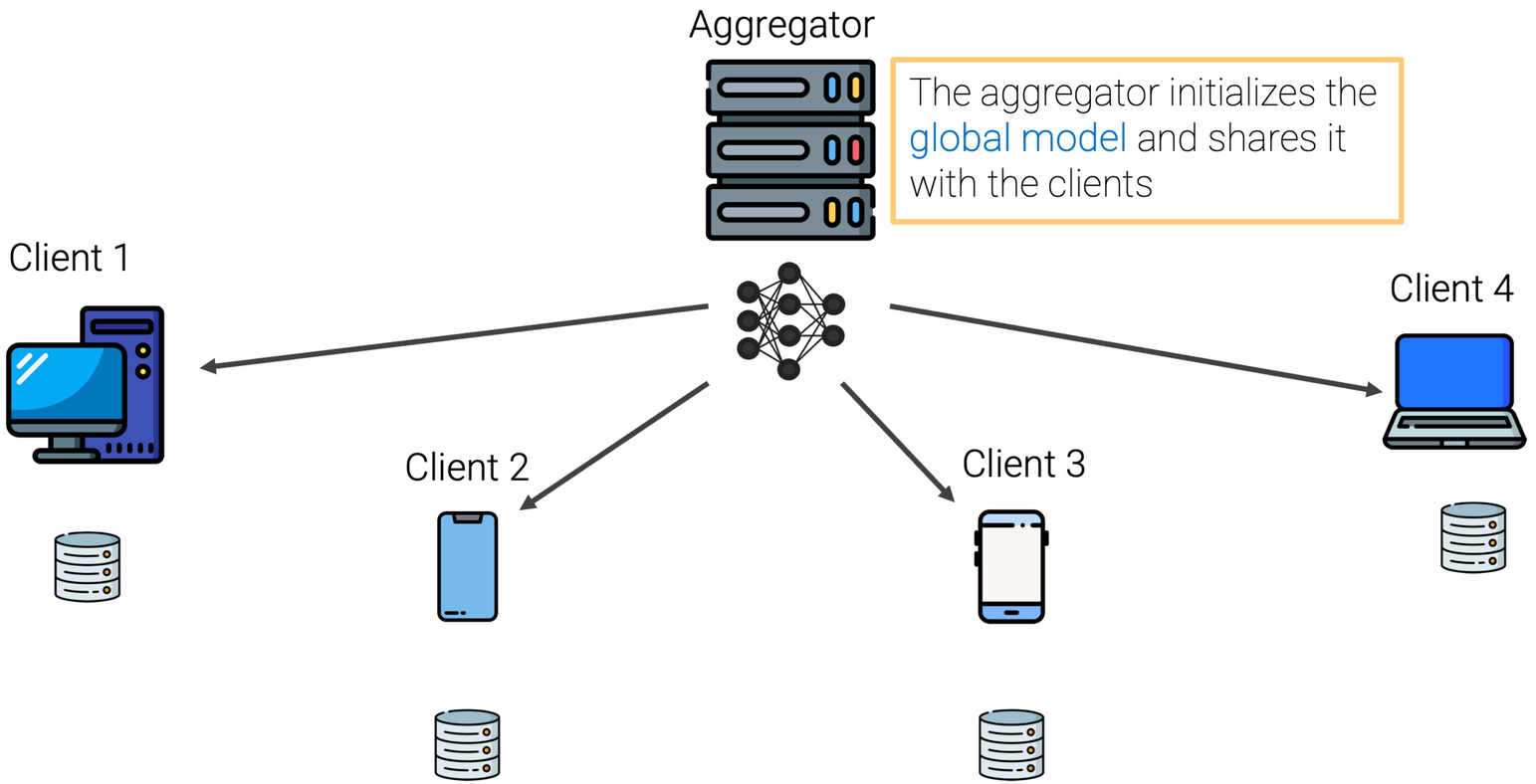
After the local training, the client send its updated model to the Server

Clients (Collaborators)

Federated Learning

Training procedure

ROUND 1

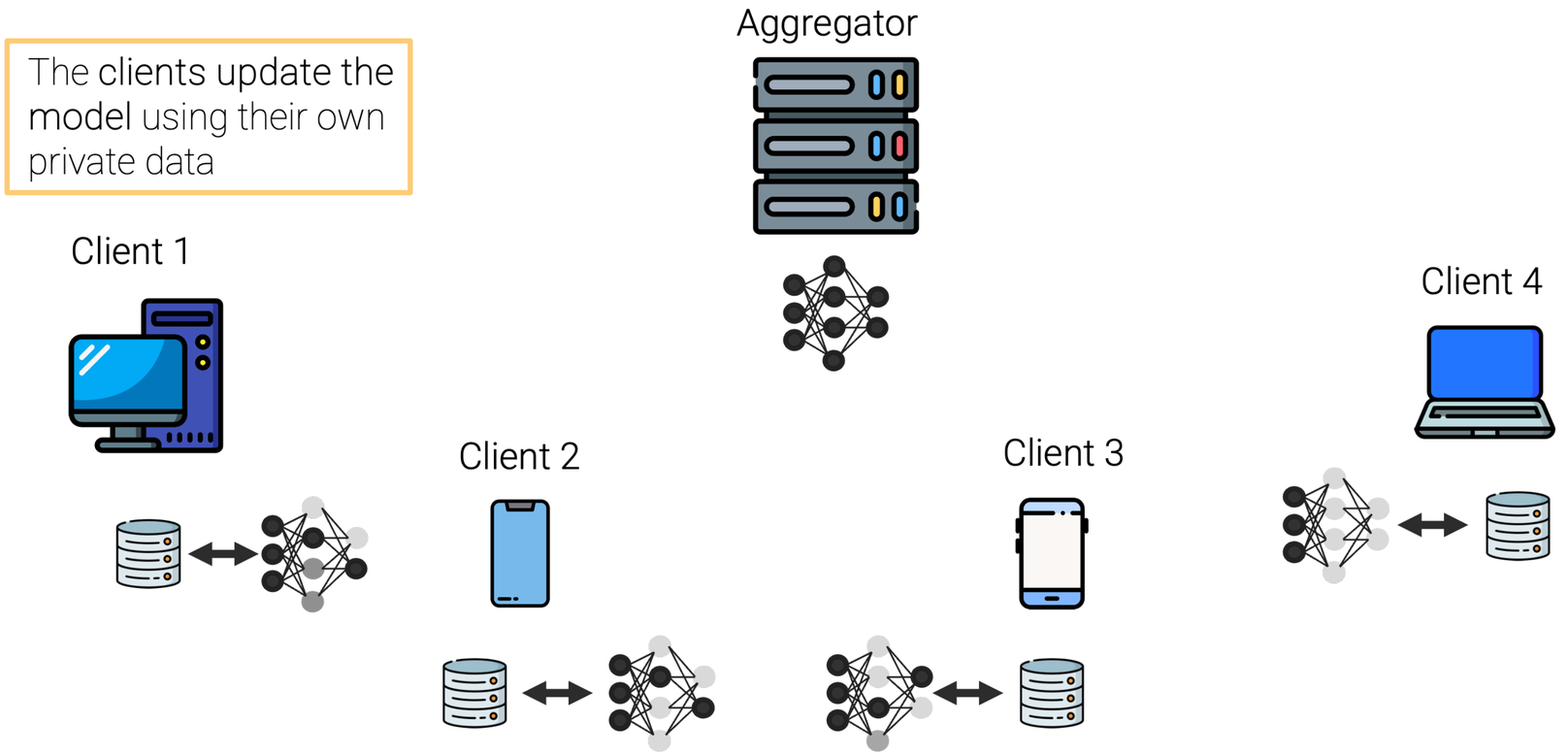


Federated Learning

Training procedure

ROUND 1

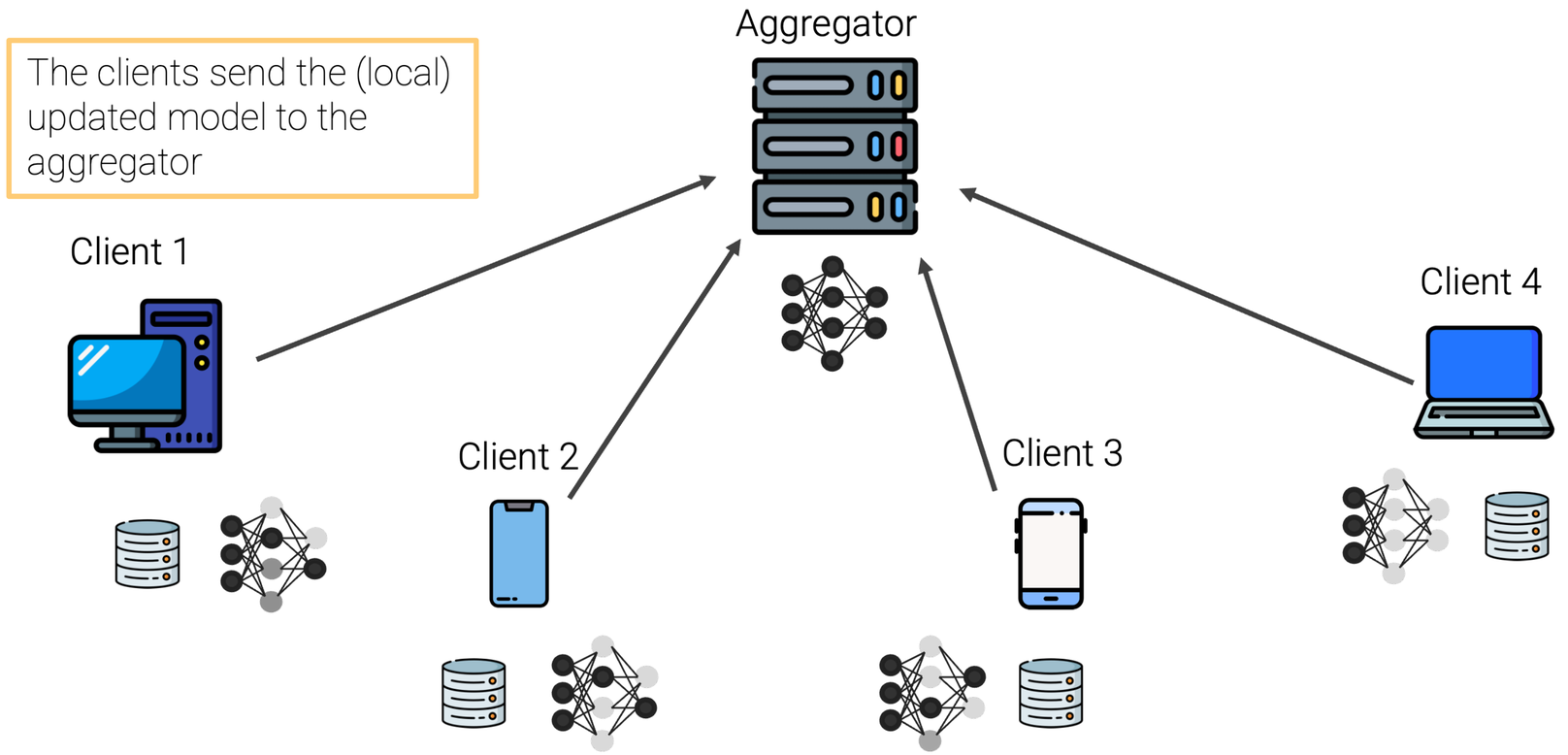
The clients update the model using their own private data



Federated Learning

Training procedure

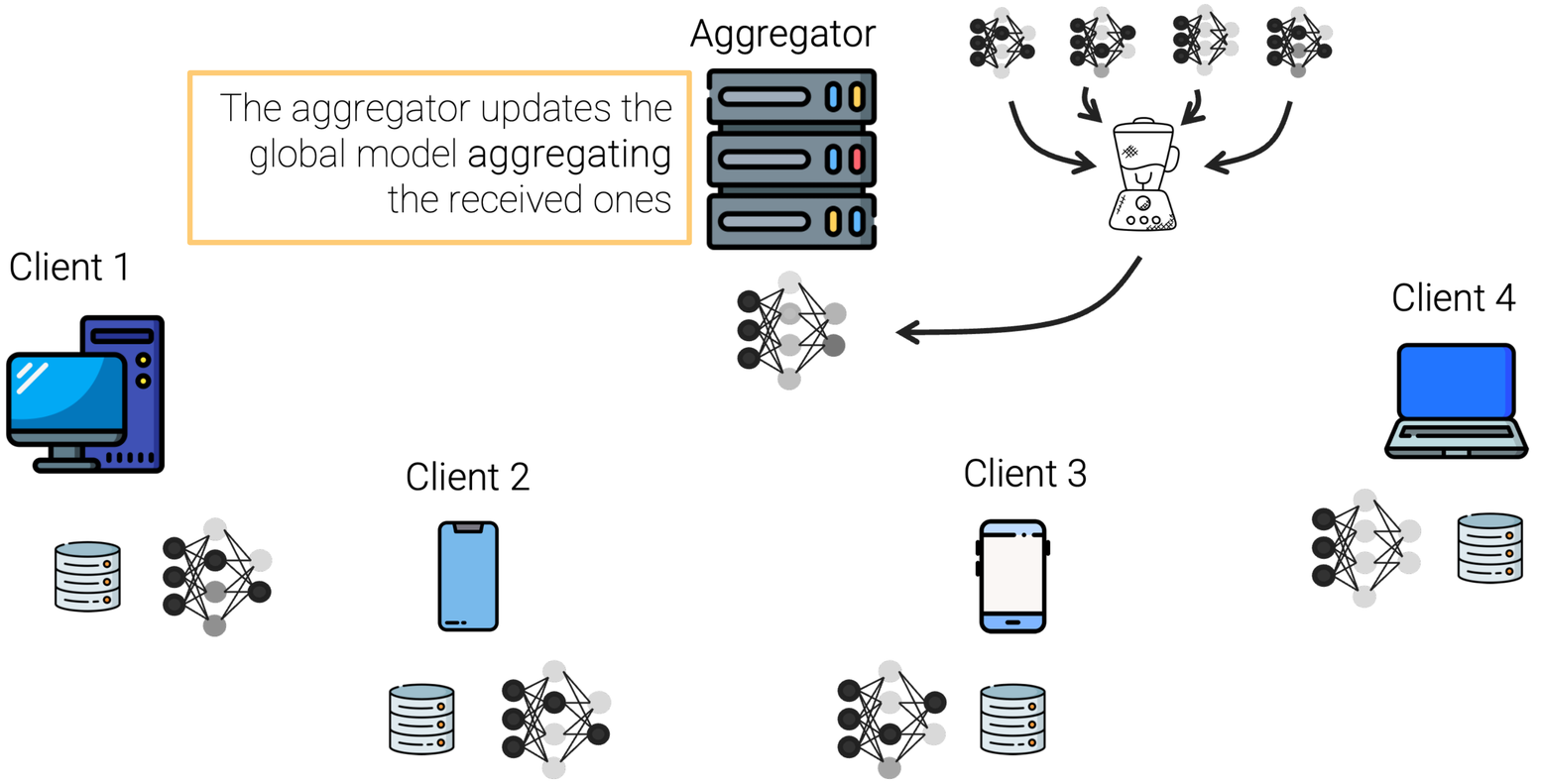
ROUND 1



Federated Learning

Training procedure

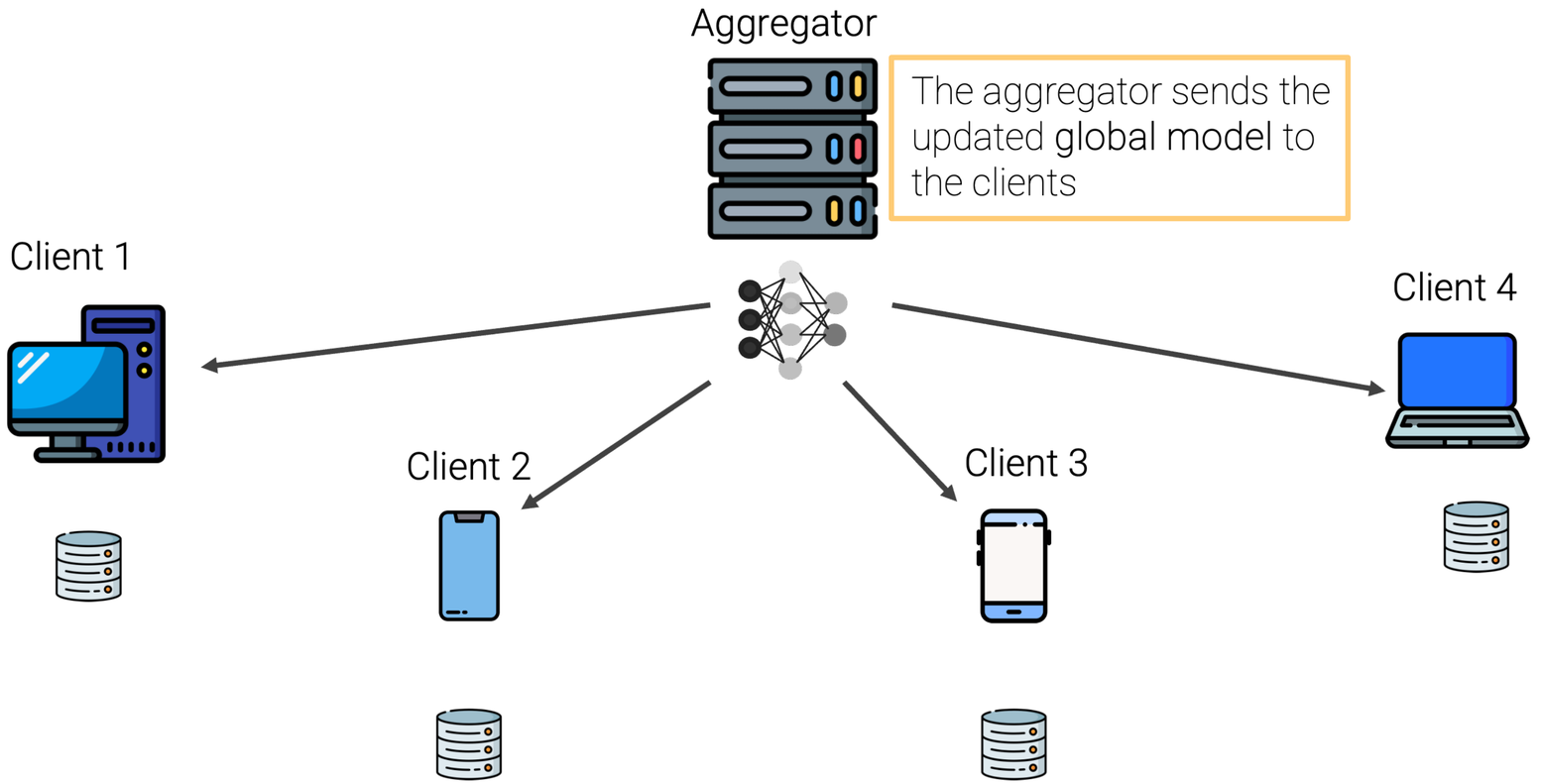
ROUND 1



Federated Learning

Training procedure

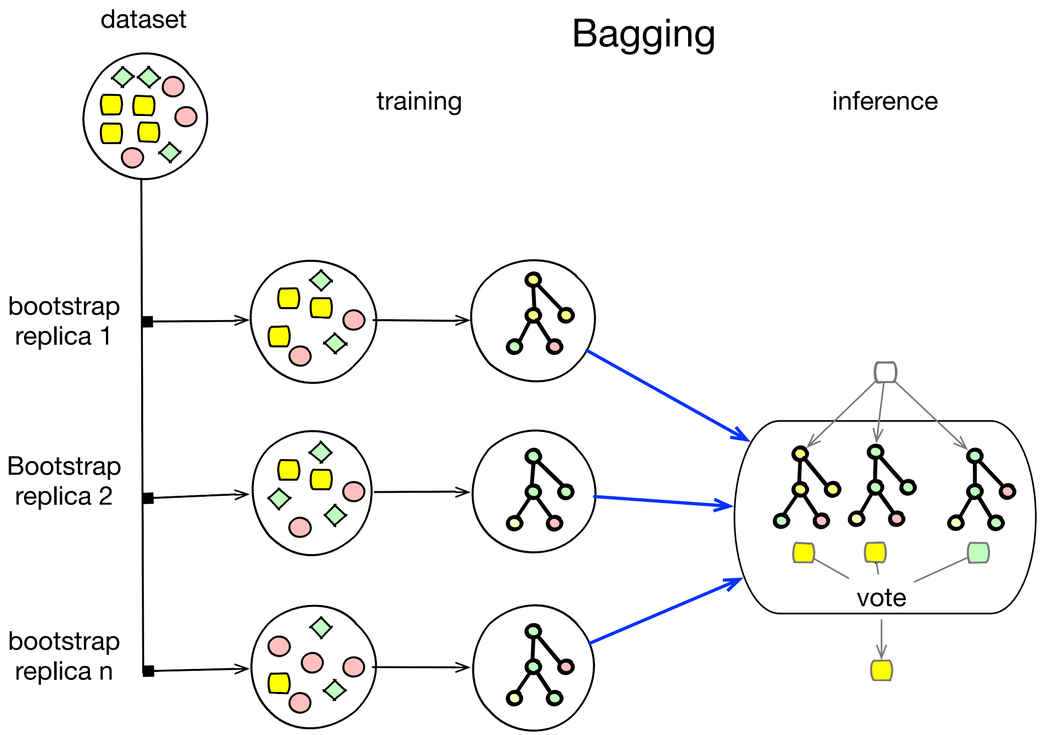
ROUND 2…



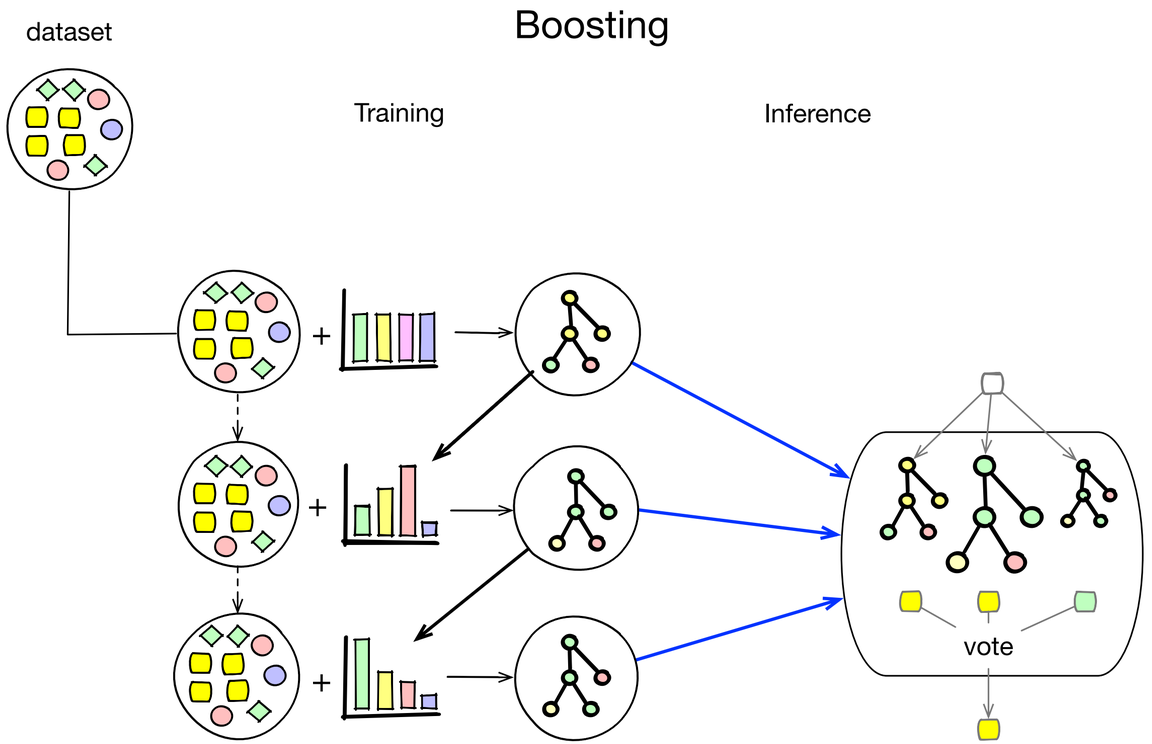
Federated Learning

Not only neural networks

Parallel



Sequential



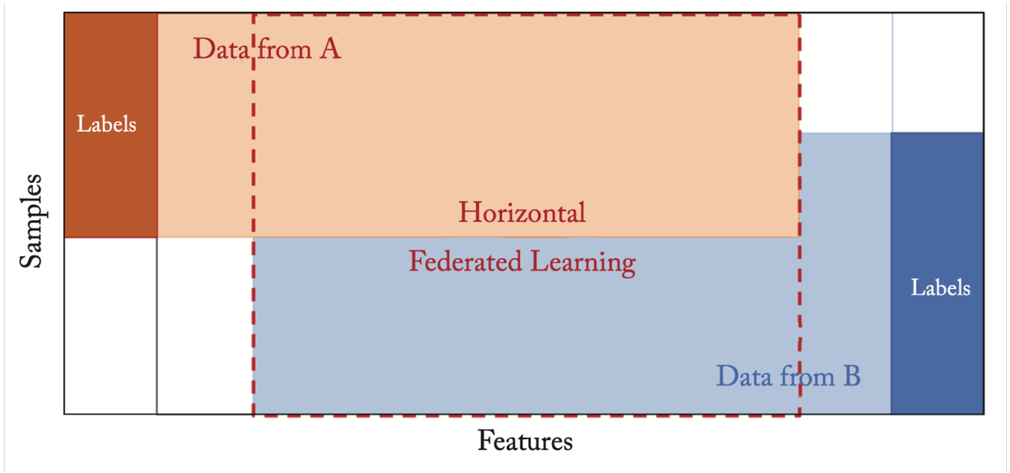
Mirko Polato, Roberto Esposito, Marco Aldinucci. Boosting the federation: Cross-silo federated learning without gradient descent. IJCNN 2022

Federated Learning

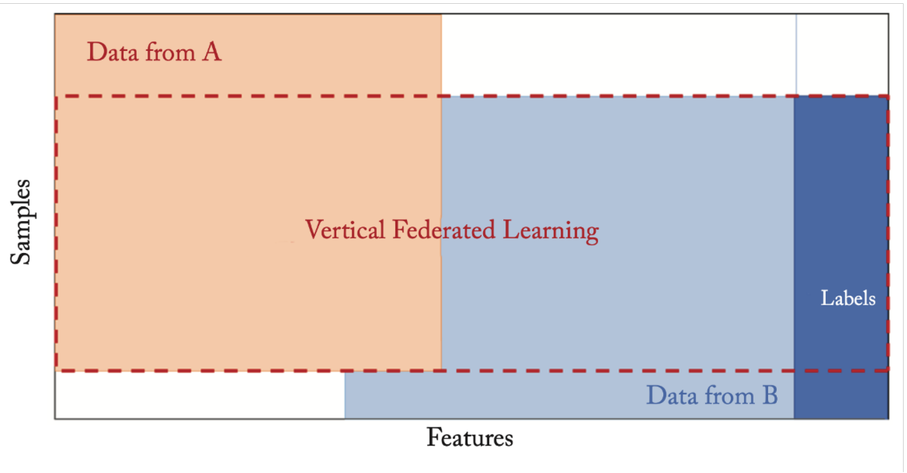
Data Categories

- In **Horizontal FL**, clients share the same feature space, while they differ in the sample space.
- In **Vertical FL**, clients share the same sample space, while they differ in the feature space.

Horizontal



Vertical



Federated Learning

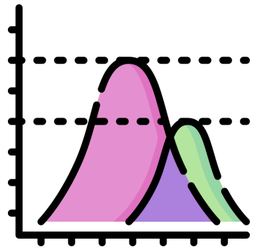
Deployment scenarios

- FL can be divided into cross-silo and cross-device.
  - **Cross-silo** FL involves a small number of organizations (silos) like corporations or medical institutions. Few clients with large local datasets.
  - **Cross-device** FL involves large scale and often low-power devices like smartphones. Many clients with relatively small local datasets

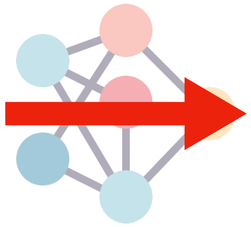
Federated Learning Challenges

Security & Privacy

Hyper-parameters tuning



Non-iidness



Beyond neural networks

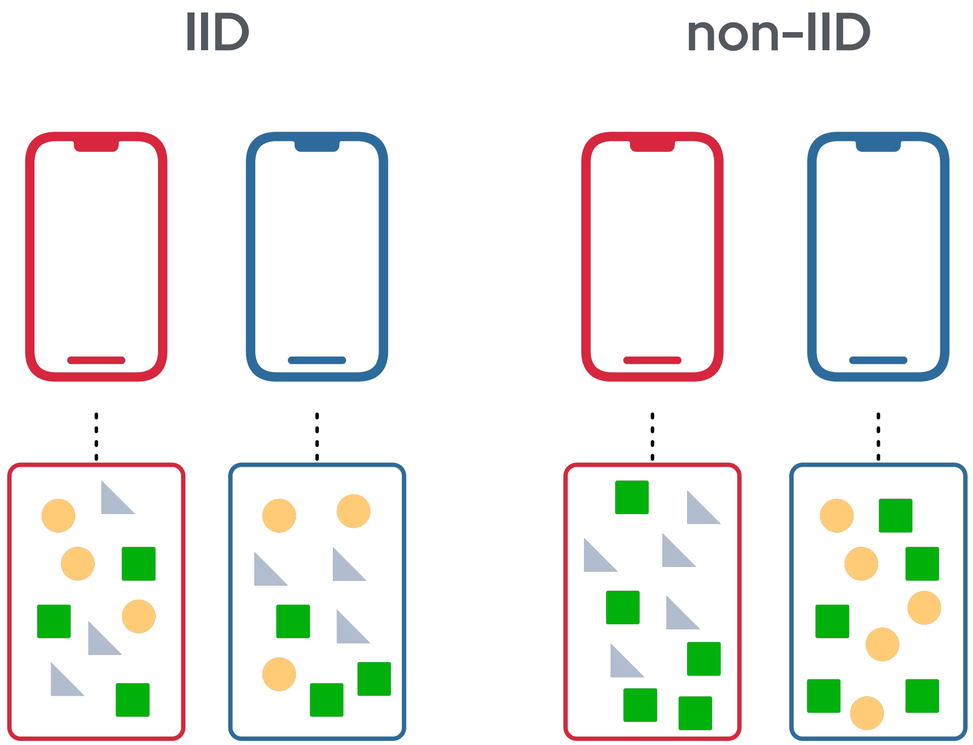
Coordination

Fairness

Federated Learning

Data Distribution

- Data is considered **IID** when each data point is drawn **identically and independently** from the same probability distribution.
- Each sample is statistically independent of the others.
- All samples follow the same underlying distribution.
- However, in **real-world scenarios, the IID assumption does not hold.**
- **Non-IID** data come from different probability distributions (i.e., one per client).

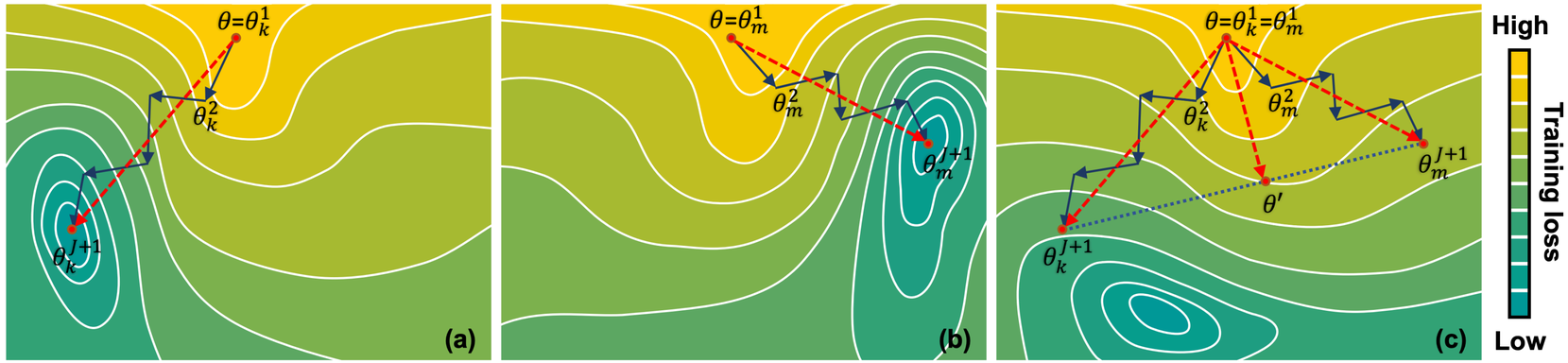


Federated Learning

Data Distribution

- Local data may be really different in terms of distribution, features, and classes.
- Non-IID data cause unstable and **inefficient training**.

- **Quantity skew**: different clients hold different amounts of data.
- **Prior shift**: the distribution of labels varies across clients.
- **Covariate shift**: the input features distribution changes across clients.

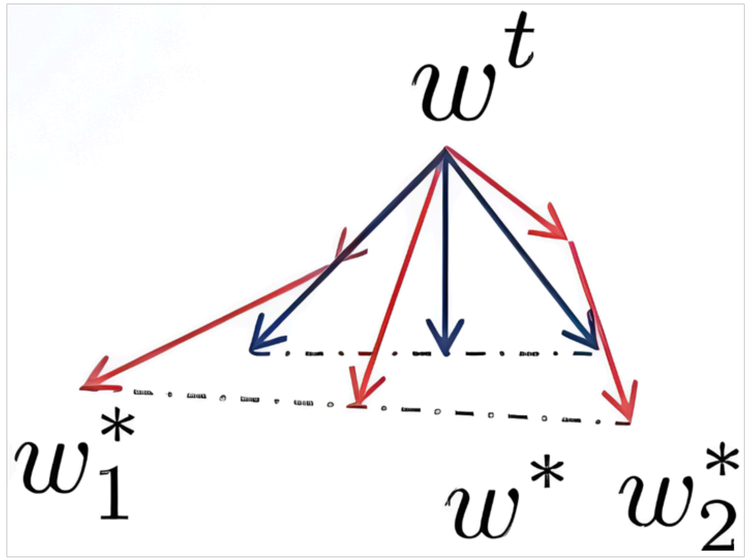


Dealing with non-IID distributions

FedProx

FedProx is a FedAvg variant for heterogeneous clients

$$\min_{w_t^k} \mathcal{L}(w_t^k; \mathcal{D}_k) + \frac{\mu}{2} \|w_t^k - w_{t-1}\|_2^2$$



- **Proximal term** limits the impact of heterogeneous local updates
- Allows to avoid the "device drop" by incorporating partial updates from slow devices
- Theoretical guarantees on convergence

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, Virginia Smith. "Federated Optimization in Heterogeneous Networks". MLSys 2020.

Federated Learning

Privacy risks

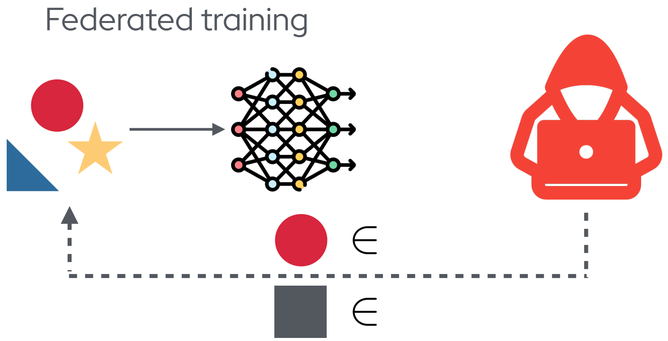
Server — **Honest-but-curious**: a party that adheres strictly to the prescribed FL protocol without any malicious alterations, but harboring intentions to infer private information.

Clients — **Malicious**: actively seeks to undermine the privacy and integrity of the FL process.

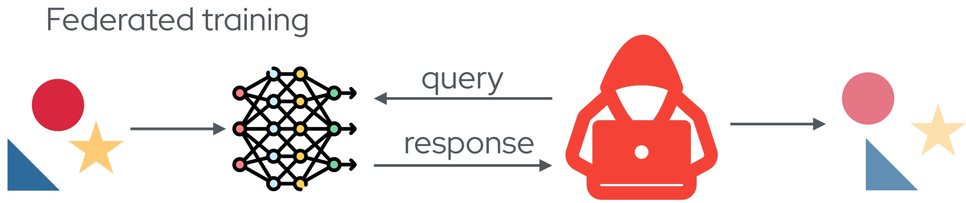
Federated Learning

Security & Privacy - Attacks

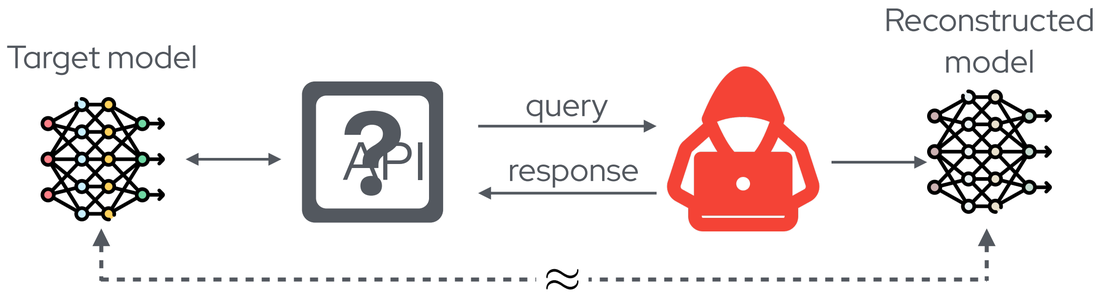
Membership inference



Model Inversion



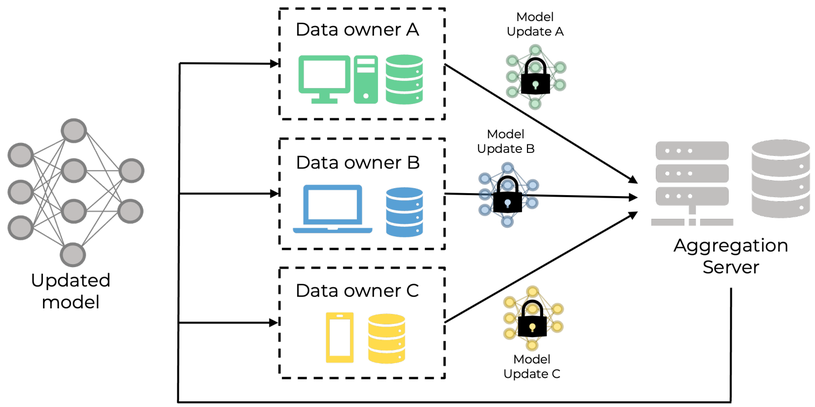
Model Extraction



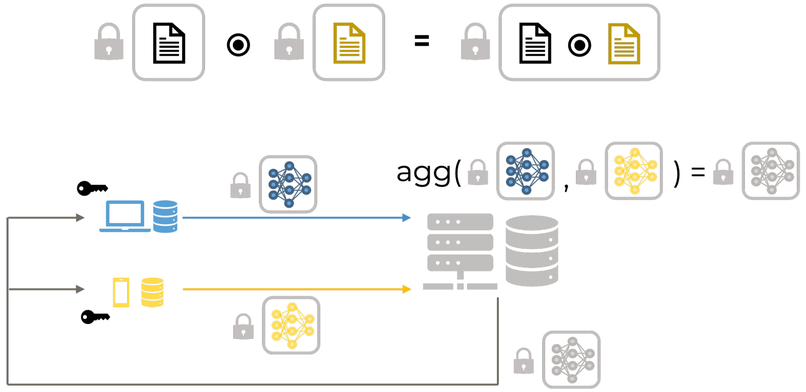
Federated Learning - Privacy

Security & Privacy - Defences

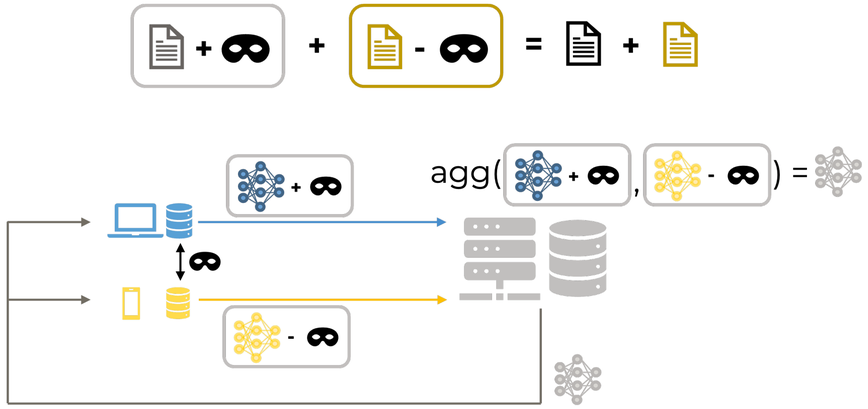
Differential Privacy



Homomorphic encryption



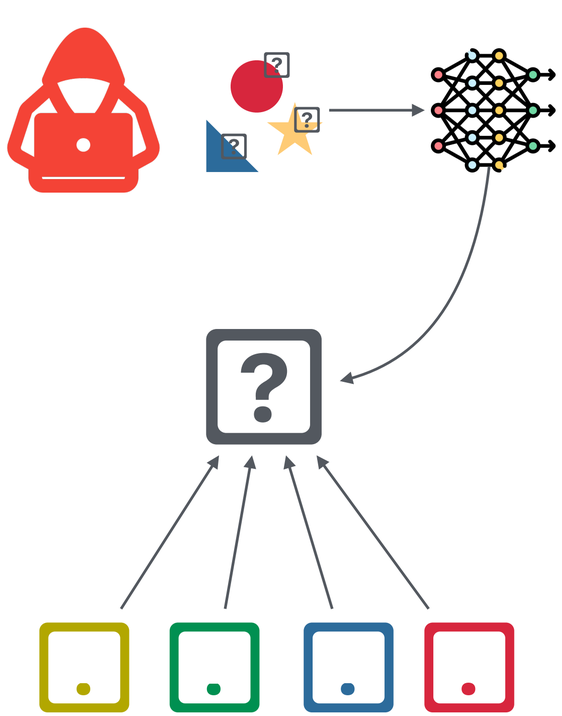
Secure aggregation



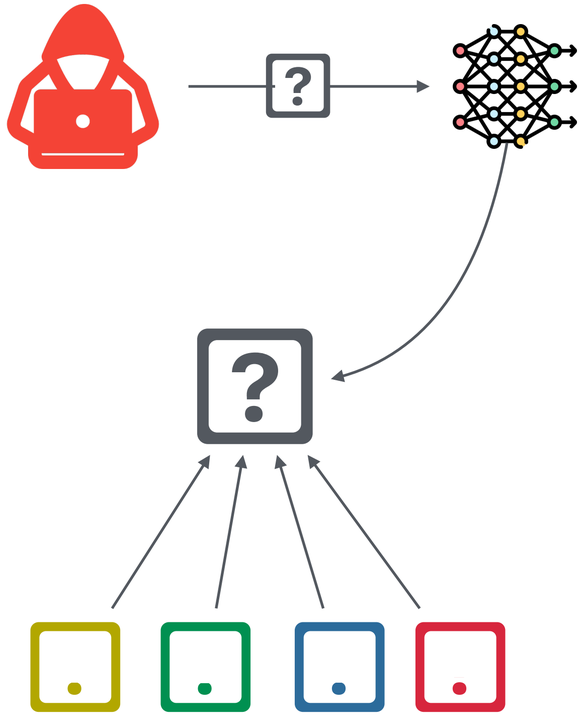
Federated Learning

Robustness

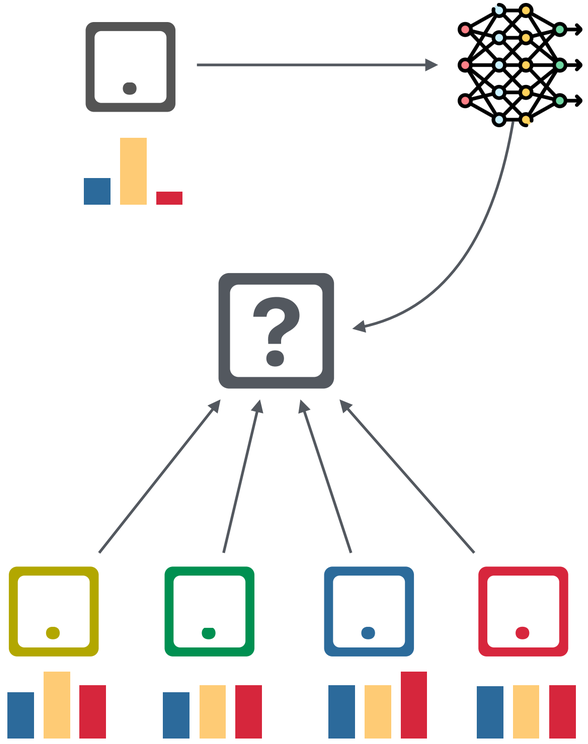
Adversarial Data



Adversarial Model



Noisy updates



Federated Learning

Robustness - possible approaches

Server-side

- Client selection
- Aggregation approach modification
- Aggregation Hyperparameter Tuning

Clients-side

- Objective regularization
- Ad-hoc optimizer
- Data Selection
- Manifold

Federated Learning

Robust aggregation

The **mean** (as in FedAvg) **is not robust** to outliers! A malicious client can deviate the model in a consistent way

Possible alternatives

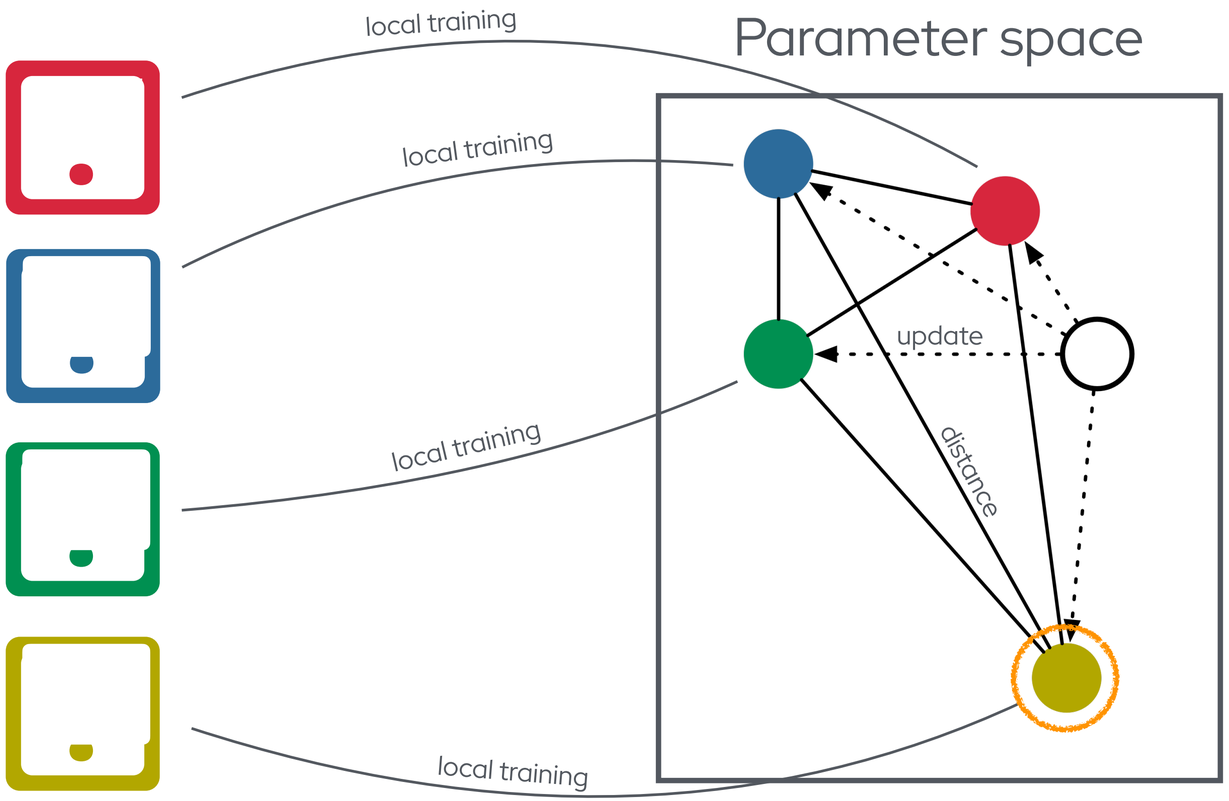
- **Geometric median**[1]
- **Coordinate-wise median**[2]
- **Trimmed-mean**[2]

[1]Pillutla, Krishna, Sham M. Kakade, and Zaid Harchaoui. "Robust aggregation for federated learning." IEEE Transactions on Signal Processing 70 (2022): 1142-1154.

[2]Yin, Dong, et al. "Byzantine-robust distributed learning: Towards optimal statistical rates." International conference on machine learning. Pmlr, 2018.

Federated Learning

Robust aggregation: KRUM



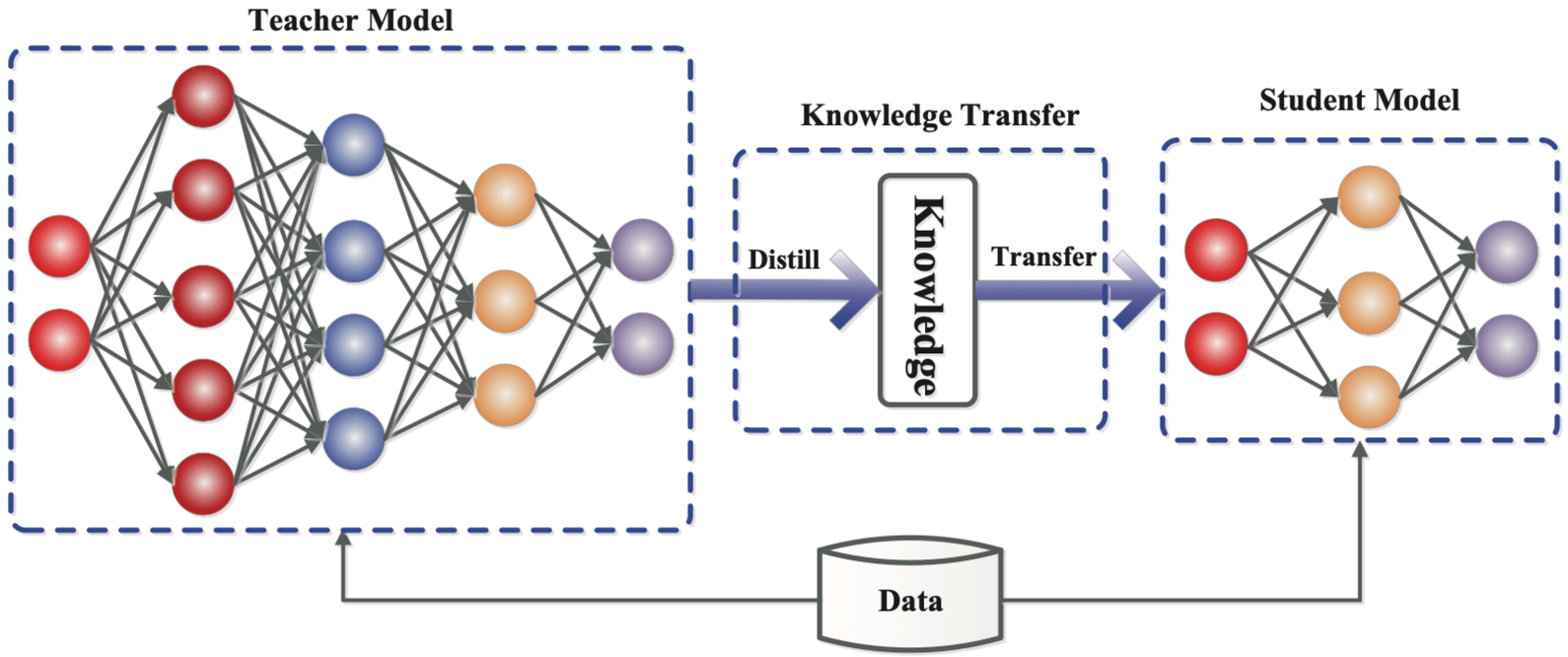
KRUM: Identifies the outliers as the models that are one average more distant than the others

Blanchard, Peva, et al. "Machine learning with adversaries: Byzantine tolerant gradient descent." Advances in neural information processing systems 30 (2017).

Federated Learning

Communication

A simple technique to reduce the communication is to leverage **knowledge distillation**

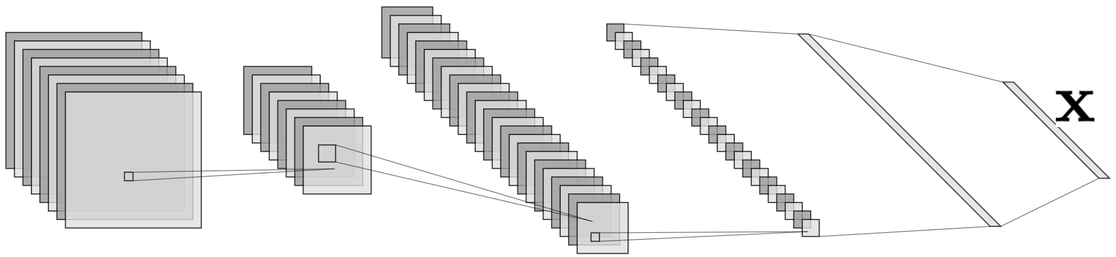


Gou, Jianping, et al. "Knowledge distillation: A survey." International journal of computer vision 129.6 (2021): 1789-1819.

Federated Learning - Communication

Communication - Prototype Learning

Prototype Learning is a well-known approach that relies heavily on the direct representation of the data



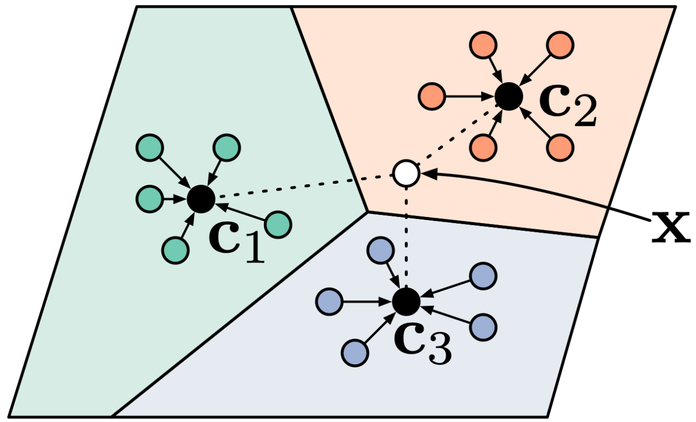
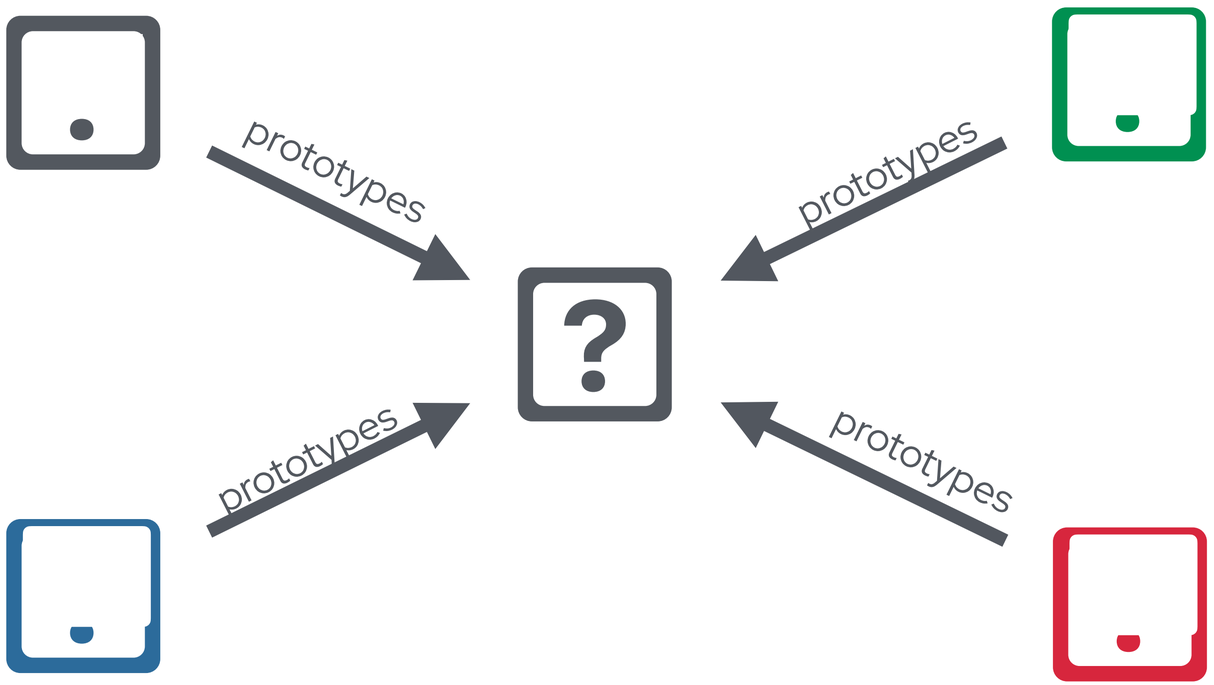


Image from Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." Advances in neural information processing systems 30 (2017).

Federated Prototype Learning

Communication - Prototype Learning

The main idea is to **exchange only the prototypes** thus hugely reduce the communication cost



- Low communication cost
- Heterogenous architectures
- Privacy

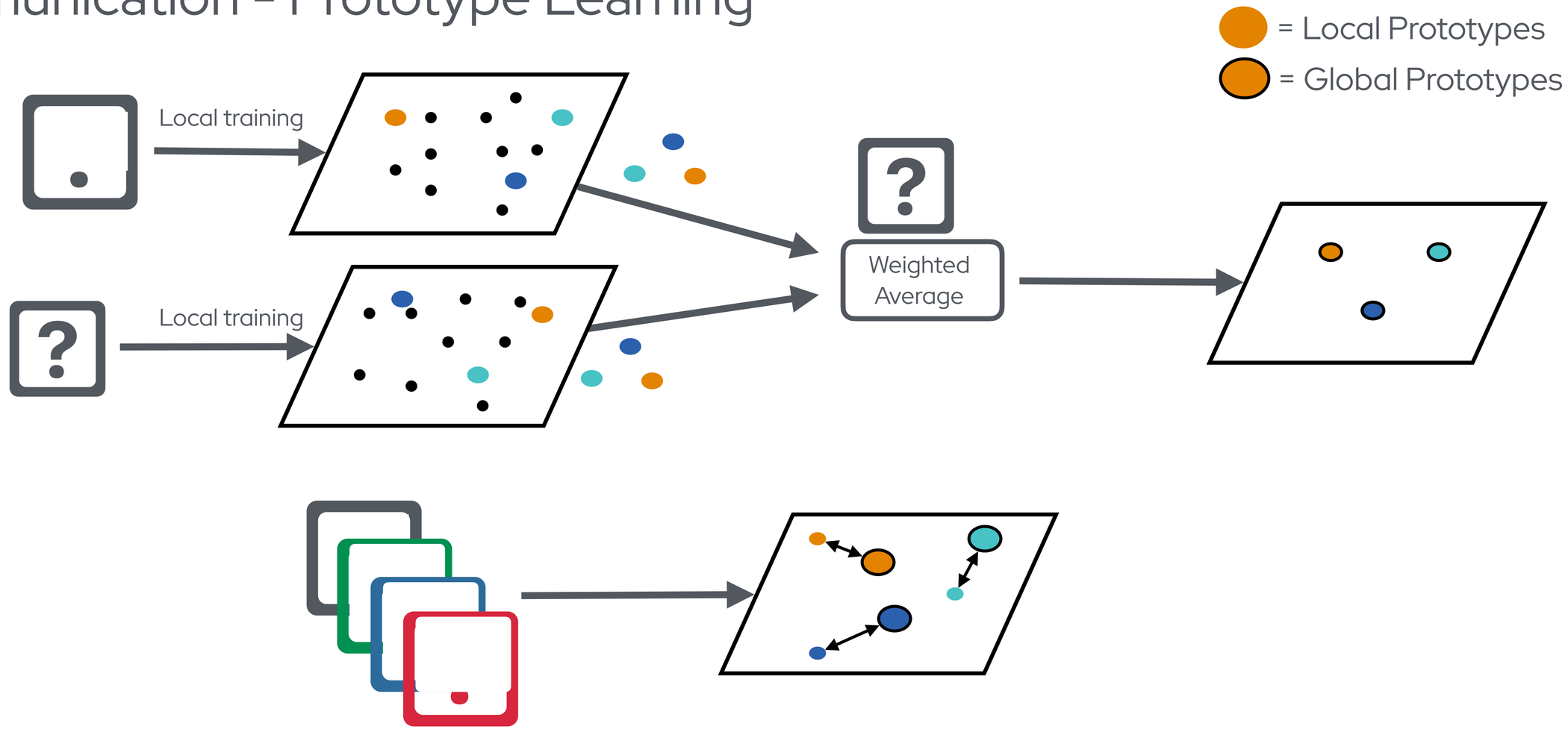
?

Federated Prototype Learning

The naive way

Federated Prototype Learning

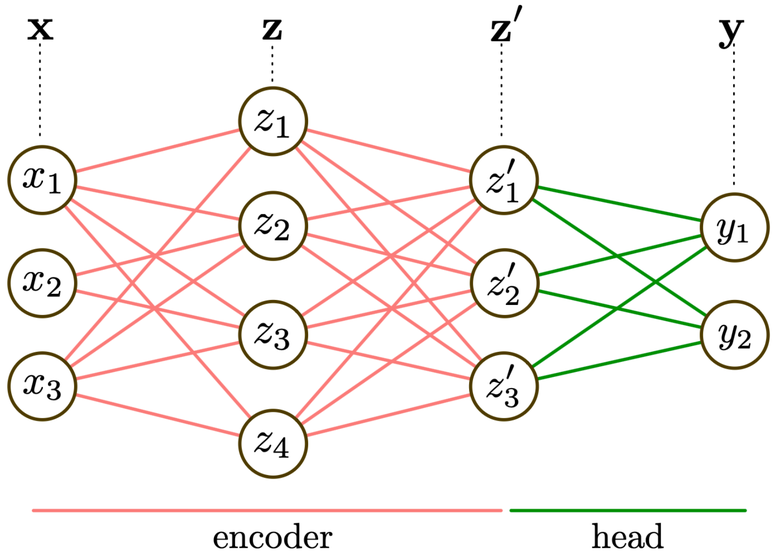
Communication - Prototype Learning

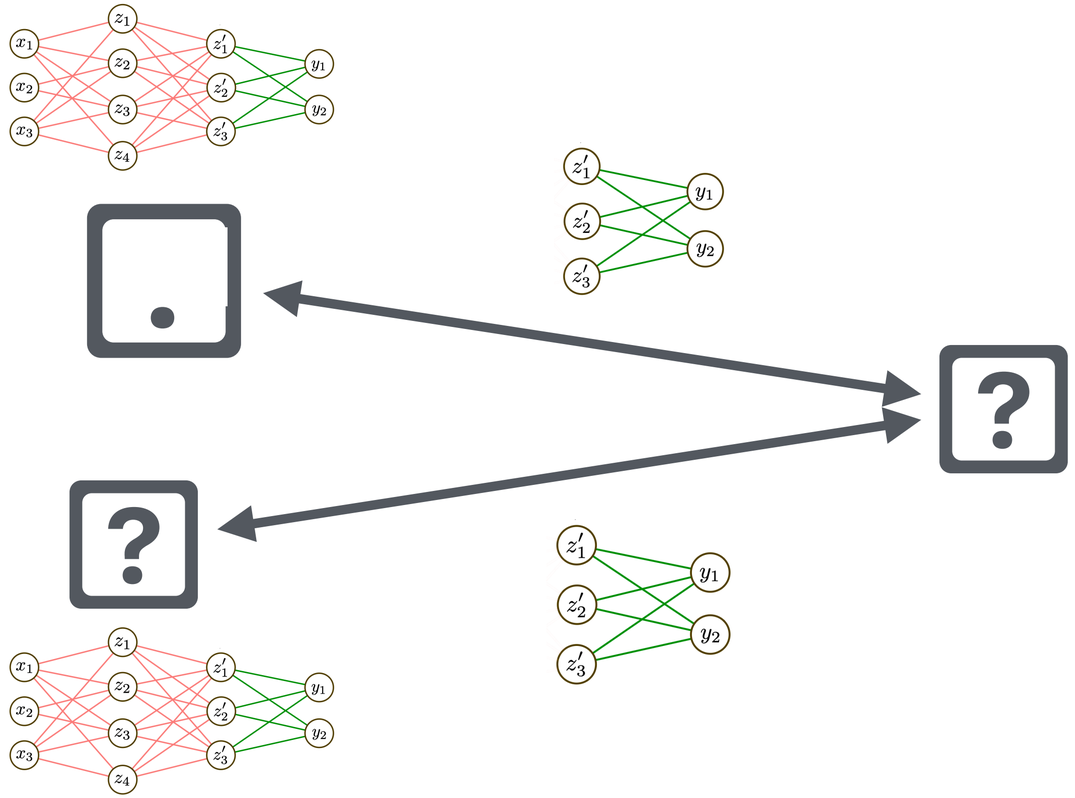


Tan, Yue, et al. "Fedproto: Federated prototype learning across heterogeneous clients." Proceedings of the AAAI conference on artificial intelligence. Vol. 36. No. 8. 2022.

LG-FedAVG

Partial model sharing: shared head and private backbone





Paul Pu Liang*, Terrance Liu*, Liu Ziyin, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think Locally, Act Globally: Federated Learning with Local and Global Representations. NeurIPS 2019 Workshop on Federated Learning

Introduction to fluke

Yet another FL framework?

Not really!

from the idea

through the implementation

to testing

Yet another FL framework?

Not really!

fluke

from the idea

through the implementation

to testing

✔ simulated federation

✘ deploy

Main features of fluke

Designed for fast prototyping and testing

https://github.com/makgyver/fluke

- **Open source**: fluke is an open-source Python package;
- **Easy to use**: fluke is designed to be extremely easy to use out of the box;
- **Easy to extend**: fluke is designed to minimize the overhead of adding new algorithms;
- **Up-to-date**: fluke comes with several (30+) state-of-the-art federated learning algorithms and datasets and it is regularly updated to include the latest affirmed techniques;
- **Easy to read**: the source code of the algorithms is written to mimic as close as possible the description in the reference papers.

SCAN ME

fluke is on PyPi!

Install it using a single command

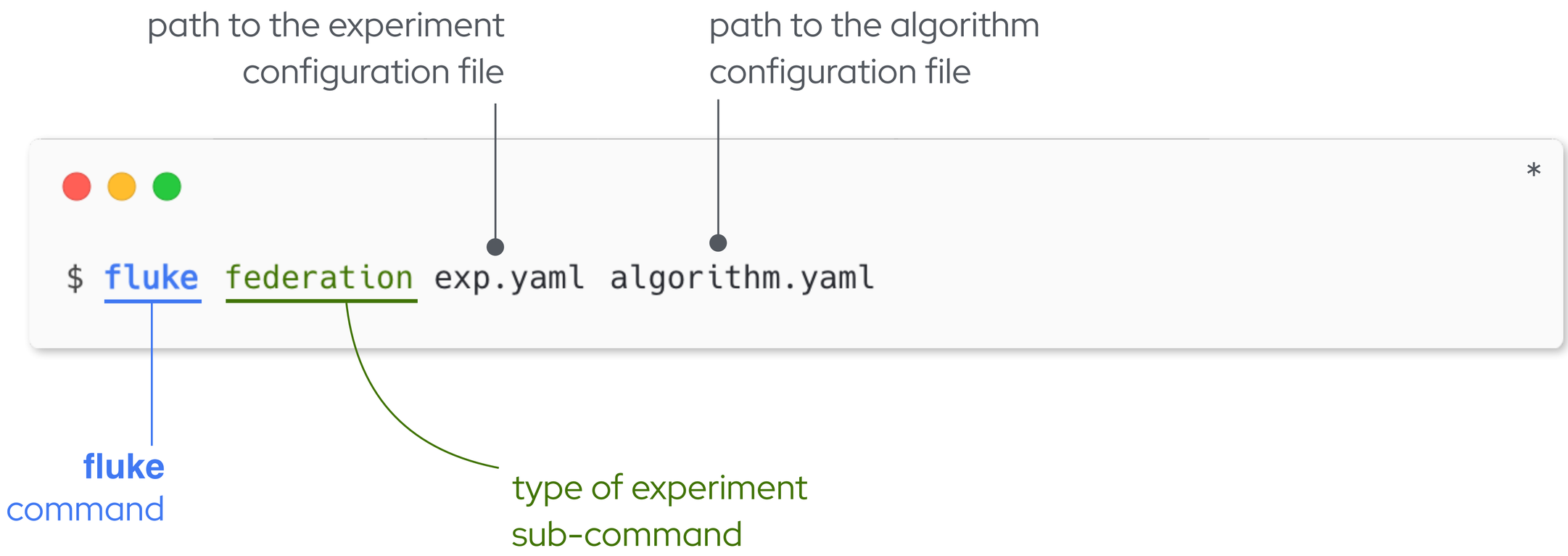

```
$ pip install fluke-fl
```

... or by cloning the repo

```
$ git clone https://github.com/makgyver/fluke.git
$ cd fluke
$ pip install -r requirements.txt
```

fluke CLI

You can run your experiment outright with a single command!



(*) If you cloned the repo, the command (launched from the fluke folder) is $ python -m fluke.run federation exp.yaml algorithm.yaml

Configuration files

Two YAML files for everything you want to configure

exp.yaml

```yaml
data:
  dataset:
    name: mnist
    path: ./data
  distribution:
    name: iid
  ...

exp:
  device: cpu
  seed: 42

eval:
  ...

logger:
  name: Log

protocol:
  eligible_perc: 0.2
  n_clients: 10
  n_rounds: 10
```

dataset and data distribution configuration

general settings

evaluation settings

logger setting

federated protocol configuration

client-side hyper-parameters

server-side hyper-parameters

model to be learned

algorithm name

algorithm.yaml

```yaml
hyperparameters:

  client:
    batch_size: 32
    local_epochs: 5
    loss: CrossEntropyLoss
    optimizer:
      lr: 0.1
    scheduler:
      gamma: 1
      step_size: 1

  server:
    weighted: true

  model: MyNeuralNet

name: fluke.algorithms.fedavg.FedAVG
```

Default configuration files can be downloaded using the command: `fluke-get`

Configuration files

Two YAML files for everything you want to configure

exp.yaml

```yaml
data:
  dataset:
    name: mnist
    path: ./data
  distribution:
    name: iid
  ...

exp:
  device: cpu
  seed: 42

eval:
  ...

logger:
  name: Log

protocol:
  eligible_perc: 0.2
  n_clients: 10
  n_rounds: 10
```

dataset and data distribution configuration

general settings

evaluation settings

logger setting

federated protocol configuration

algorithm.yaml

```yaml
hyperparameters:

  client:
    batch_size: 32
    local_epochs: 5
    loss: CrossEntropyLoss
    optimizer:
      lr: 0.1
    scheduler:
      gamma: 1
      step_size: 1

  server:
    weighted: true

  model: MyNeuralNet

  name: fluke.algorithms.fedavg.FedAVG
```

client-side hyper-parameters

server-side hyper-parameters

model to be learned

algorithm name

Default configuration files can be downloaded using the command: **fluke-get**

fluke CLI – not only federation

You can run the "same" experiment without the federation for comparison

same experiment but without the federation –
the number of epochs client-side are calculated*
as n_rounds * eligible_perc * local_epochs

```
$ fluke clients-only exp.yaml algorithm.yaml
```

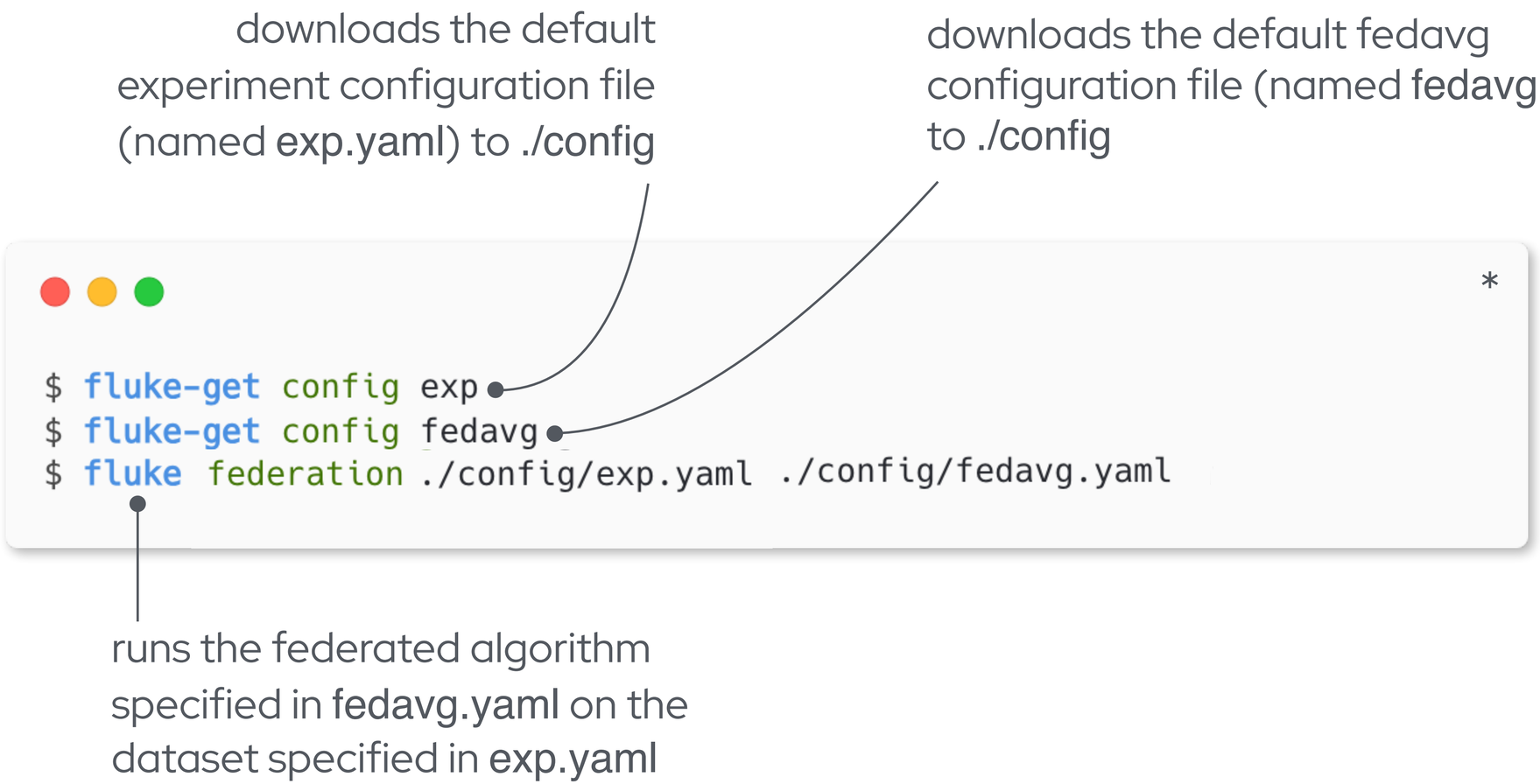
```
$ fluke centralized exp.yaml algorithm.yaml
```

same type of experiment but with all the dataset centralised

(*) The number of epochs can be set by the user via an option of the command, e.g., `—epochs=100`

Example: FedAVG on MNIST

Fluke comes with many downloadable configuration files ready to be used/modified

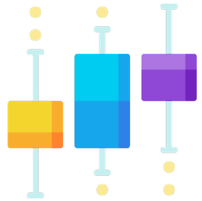


(*) If you want to know all the available default configuration files use the command $ fluke-get list

fluke logging

Performance can be logged on your preferred tool

communication cost*



classification performance**
(e.g., accuracy, precision, recall, F1 - marco and micro)

system performance***

TensorBoard

W&B

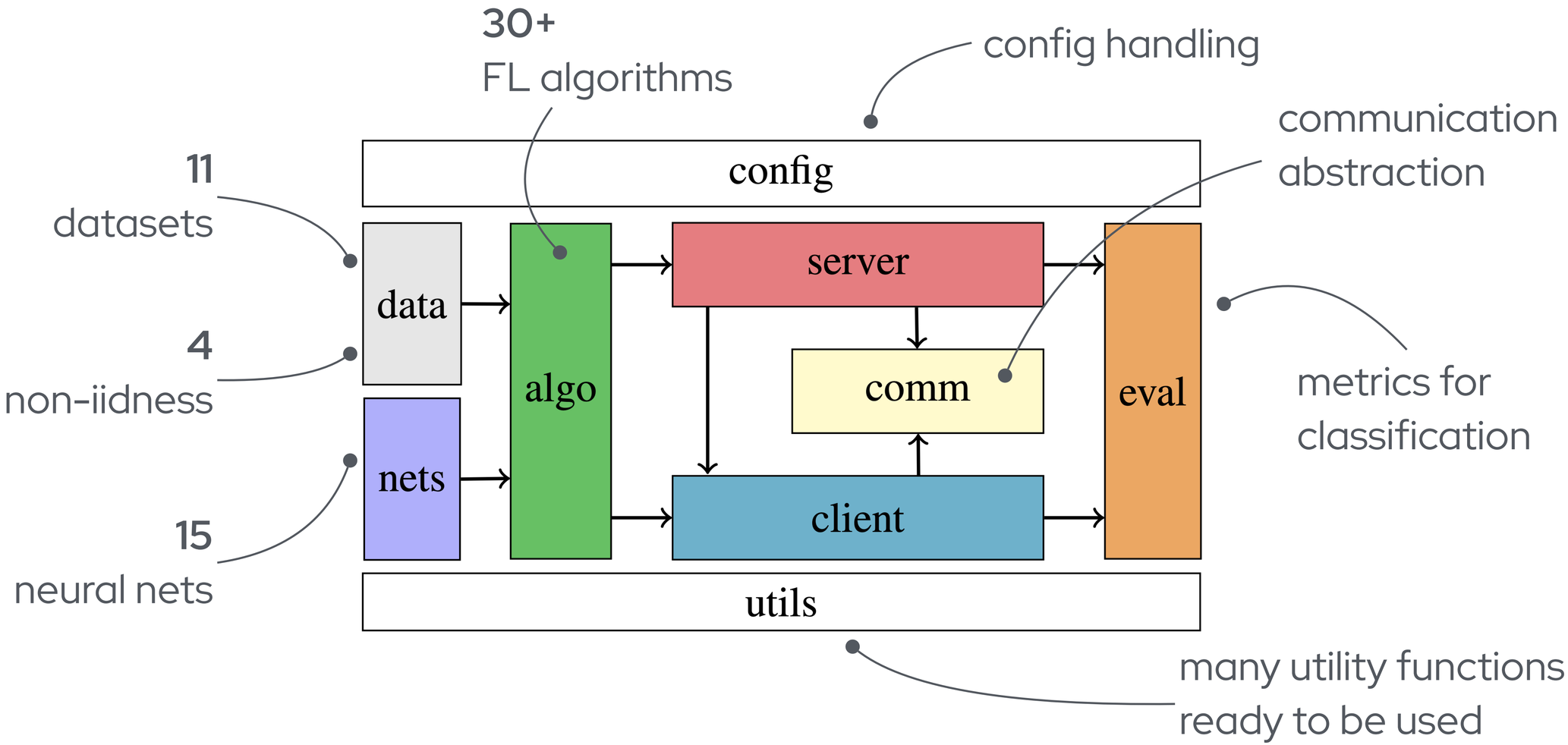
(*) The communication cost is estimated as the number fo floating points numbers exchanged by the entire federation.
(**) Currently, **fluke** supports only classification but it is straightforward to extend to other tasks.
(***) System performance are automatically logged by W&B and ClearML.

fluke python API

https://github.com/makgyver/fluke

SCAN ME

The python API offers all you need to implement and test your FL ideas



fluke server

Code readability is a key feature of fluke

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**

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

  $m_t \leftarrow \sum_{k \in S_t} n_k$

  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

Server *

```python
def fit(self, n_rounds: int, eligible_perc: float) -> None:

    for round in range(n_rounds):
        eligible = self.get_eligible_clients(eligible_perc)
        self.broadcast_model(eligible)

        for c, client in enumerate(eligible):
            client.local_update(round + 1)

        self.aggregate(eligible)
```

McMahan et al. Communication-Efficient Learning of Deep Networks from Decentralized Data. AISTATS 2017.

(*) This is a "polished" version of the fluke source code.

fluke client

Code readability is a key feature of fluke

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

Client *

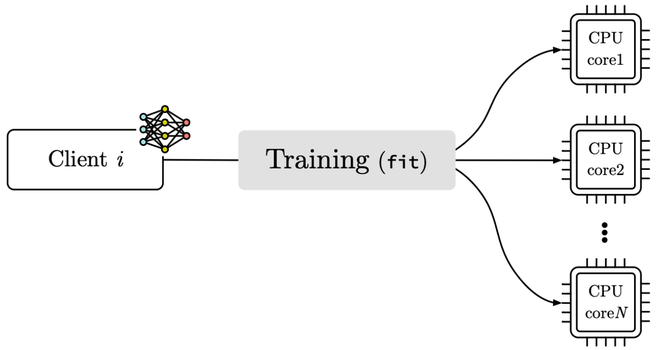
```python
def local_update(self, current_round: int):
    self.receive_model()
    self.fit()
    self.send_model()

def fit(self, override_local_epochs: int = 0):
    self.model.train()

    if self.optimizer is None:
        self.optimizer, self.scheduler = self.optimizer_cfg(self.model)

    for _ in range(epochs):
        for _, (X, y) in enumerate(self.train_set):
            X, y = X.to(self.device), y.to(self.device)
            self.optimizer.zero_grad()
            y_hat = self.model(X)
            loss = self.hyper_params.loss_fn(y_hat, y)
            loss.backward()
            self.optimizer.step()
        self.scheduler.step()

def receive_model(self):
    msg = self.channel.receive(self, self.server, msg_type="model")
    self.model.load_state_dict(msg.payload.state_dict())

def send_model(self):
    self.channel.send(Message(self.model, "model", self), self.server)
```

McMahan et al. Communication-Efficient Learning of Deep Networks from Decentralized Data. AISTATS 2017.

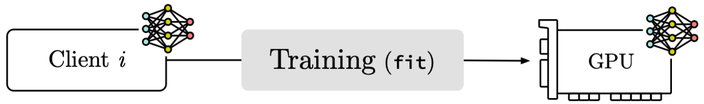
(*) This is a slightly "polished" version of the fluke source code.

Parallelization in fluke

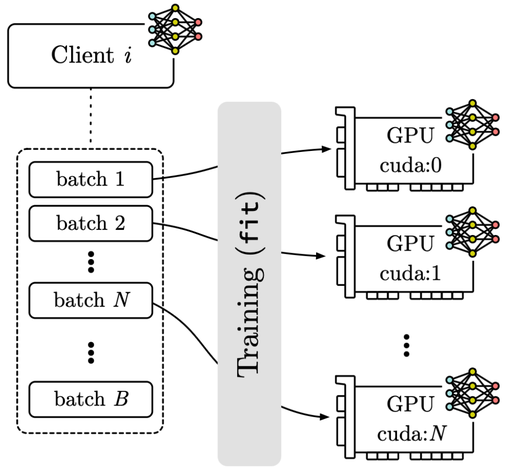
fluke *can easily handle multi-cpus/gpus*



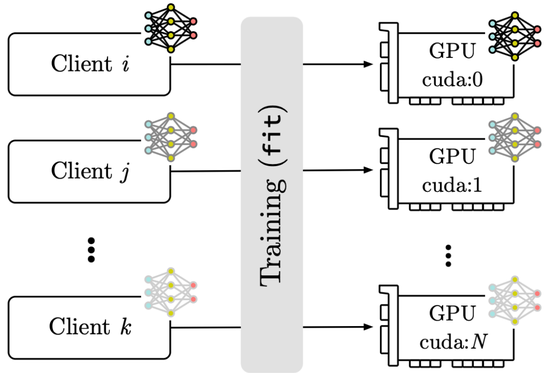
(a) Single client training on multiple CPU cores.



(b) Single client training on a single GPU.



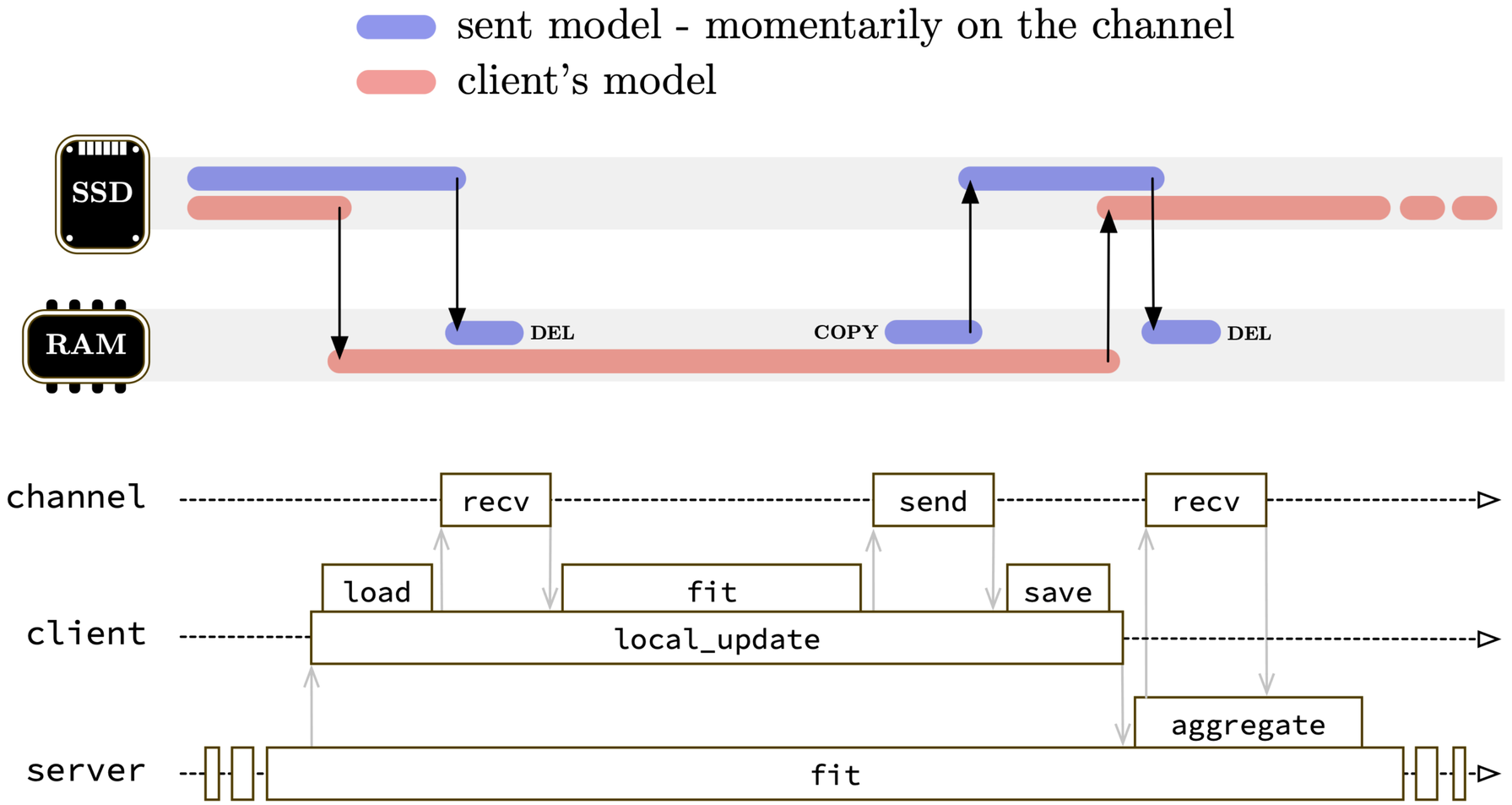
(c) Single client training on many GPUs.



(d) N clients training on N GPUs (one client per each GPU).

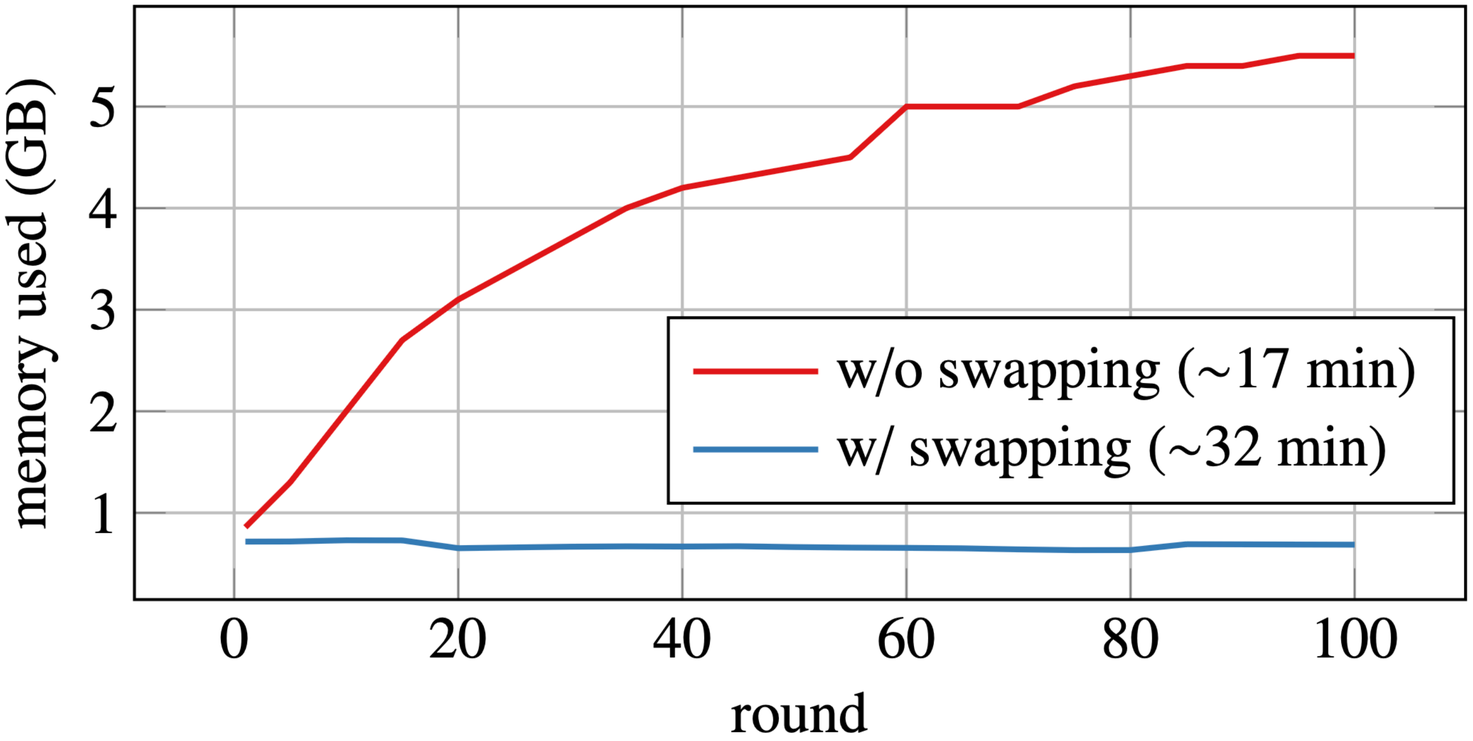
Memory management in fluke

fluke can optimize the memory usage



Memory management in fluke

The swapping mechanism guarantee (almost) constant memory consumption



Memory consumption in fluke with and without swapping on MNIST with 1000 clients and a CNN model architecture.

Give it a try, you won't regret it!

fluke 0.7.9 (current release) - fluke 0.7.10 available soon!

30+
FL algorithms

11
datasets

15
neural networks

propose your FL algorithm
to be included in **fluke**!

SCAN ME

https://github.com/makgyver/fluke

https://makgyver.github.io/fluke/

Hands-on session

https://github.com/CasellaJr/Fluke-Tutorial-ECAI25

Most of the icons have been downloaded from www.flaticon.com