Advanced Topic A05: *Federated Learning*

Data Science with Python – BSc Advanced Lectures

Joerg Osterrieder

10 Minutes

Motivation: Banks Cannot Share Customer Data

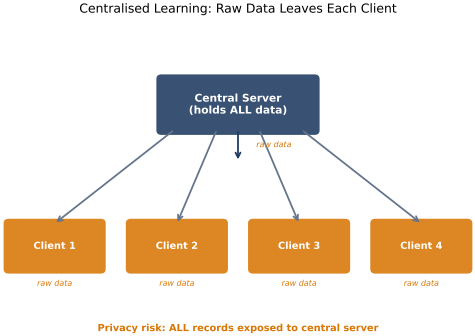
Privacy law makes centralised ML impossible in many finance contexts

- GDPR, CCPA, and banking secrecy laws prohibit sharing raw customer data
- Fraud models trained on one bank's data miss patterns seen at other banks
- Medical, telecom, and financial institutions all face the same dilemma: more data improves models, but data cannot leave the institution
- Federated learning trains on distributed data without centralising it

Federated learning: train the model, not the data – the gradient travels, not the records

Centralised vs Federated: The Key Difference

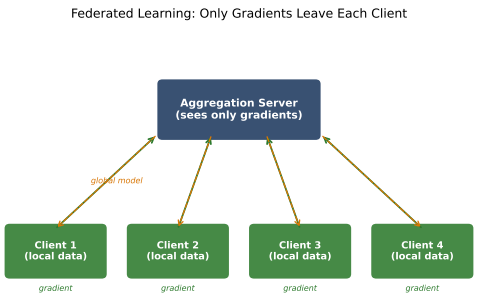
Where does learning happen?



Centralised: raw data moves to the server. Federated: only model updates move.

Federated Architecture

Clients train locally, server aggregates updates



Each client's raw data never leaves its device or institution

Vertical Federated Learning

Different institutions hold different features for the same entities

- **Horizontal FL** (standard): each client holds the same features for different entities; the most common setting
- **Vertical FL**: each client holds different features for a shared set of entities (e.g., a bank holds credit data and a telecom holds usage data for the same customers)
- Training requires privacy-preserving entity alignment (private set intersection) and split-model computation: each party computes partial activations; gradients are exchanged without exposing raw features
- Finance application: bank-telecom joint model for thin-file customers who lack traditional credit history but have rich mobile usage patterns
- Implementation: **FATE** (Federated AI Technology Enabler) and **TensorFlow Federated** both support vertical FL with secure inner-product protocols

Vertical FL: different columns, shared rows; opposite data structure to standard horizontal FL

The FedAvg Algorithm

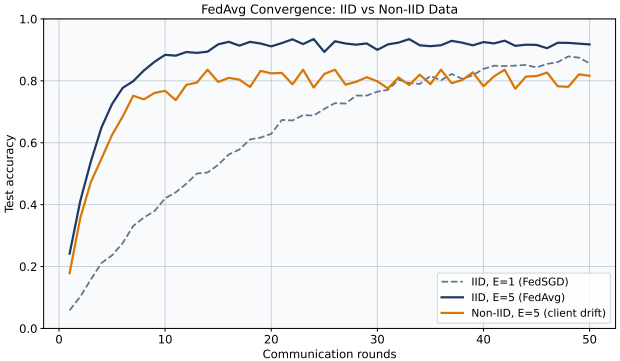
McMahan et al. (2017): Communication-Efficient Learning

- Each round: server broadcasts current global model w_t
- Each client k runs E epochs of SGD on local data: produces w_t^k
- Server aggregates: $w_{t+1} = \sum_k \frac{n_k}{n} w_t^k$ (weighted average by local dataset size n_k)
- Repeat for T rounds
- Key result: $E > 1$ local steps ("FedAvg") greatly reduces communication vs 1 step per round ("FedSGD")

FedAvg is the baseline; most modern FL algorithms improve on it

FedAvg Convergence

Accuracy rises as rounds proceed, even with non-IID data

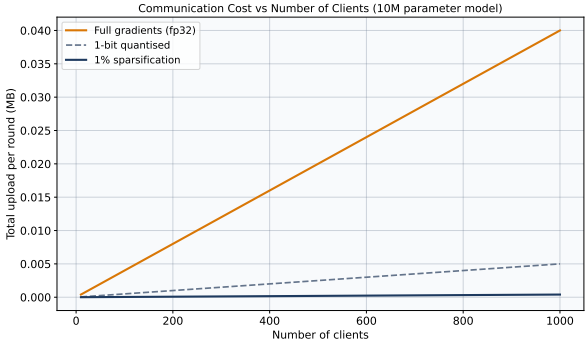


More local steps (higher E) converges faster but diverges with very non-IID data

Communication Cost: The Core Challenge

Gradient transmission is expensive at scale

- GPT-3 has 175B parameters: uploading full gradients over a mobile network is infeasible each round
- Techniques to reduce cost: gradient sparsification, quantisation (1-bit gradients), structured updates, sketching
- FedProx adds a proximal term to keep local updates close to the global model, improving convergence with heterogeneous clients



Communication is often the binding constraint, not computation

Non-IID Data: The Hardest Problem

Federated data is heterogeneous by nature

- IID assumption: each client has data drawn from the same distribution
- Reality: Bank A serves retail clients; Bank B serves corporates; very different transaction patterns
- Non-IID data causes "client drift": local models diverge and aggregation loses accuracy
- Mitigations: FedProx proximal term, SCAFFOLD variance reduction, personalised FL (each client keeps a local head on top of the shared backbone)

Non-IID FL is an active research problem; no complete solution exists yet

Personalised Federated Learning

One global model does not fit all clients

- A single global model is suboptimal when clients have very different data distributions
- **Local fine-tuning**: use the global model as initialisation; each client fine-tunes on its own data for a few steps (FedMA, Per-FedAvg)
- **Model personalisation with a local head**: shared backbone (global) plus a client-specific output layer; the backbone captures shared patterns, the head captures client-specific preferences
- **MAML-based meta-FL**: train the global model to be easily fine-tuneable; each client adapts with 1–5 gradient steps
- Finance: a retail bank and a corporate bank share a fraud backbone but use separate scoring heads calibrated to their own customer risk profiles

Personalised FL: share what is generalisable, keep what is client-specific

Privacy Guarantees: Differential Privacy

Gradients can leak private information; DP adds noise to prevent this

- Gradient inversion attacks (Zhu et al. 2019) can reconstruct training images from shared gradients
- Differential privacy: add calibrated Gaussian noise to gradients before sharing
- Cost: privacy budget ϵ controls the privacy-accuracy trade-off; smaller ϵ = more noise = more privacy = less accuracy
- Secure aggregation: cryptographic protocol where server learns only the sum of gradients, not individual client updates

FL without DP is not truly private; gradient inversion is a real attack

Secure Aggregation Protocols

Cryptographic guarantees that the server sees only the aggregated update

- **Secure aggregation** (Bonawitz et al. 2017): each client masks its gradient with random pairwise noise; masks cancel in the sum so the server decrypts only $\sum_k \Delta w_k$, never individual Δw_k
- Construction: client i adds $+r_{ij}$ for each pair (i, j); client j subtracts r_{ij}; the server's sum is noise-free but individual contributions are cryptographically hidden
- Handles client dropout: if a client disconnects before the round completes, threshold secret sharing allows the protocol to recover the partial sum
- **Secure multi-party computation (MPC)**: generalises secure aggregation to arbitrary functions; computationally expensive at scale but provides stronger guarantees
- Regulatory argument: with secure aggregation, the server never processes individual client data – consistent with GDPR's data minimisation principle

Secure aggregation: the server learns only the gradient sum; individual updates are cryptographically hidden

Limitations

Federated learning is not a silver bullet

- Honest-but-curious server can still perform membership inference attacks
- Byzantine-robust aggregation: a malicious client can poison the global model by sending crafted gradients
- Stragglers: slow clients delay each round; synchronous FL is bottle-necked by the slowest participant
- Regulatory grey area: sharing gradients may still constitute data processing under some interpretations of GDPR

FL reduces privacy risk; it does not eliminate it

Tools and Frameworks

Production-ready FL libraries

- **Flower (flwr):** framework-agnostic, supports PyTorch, TensorFlow, scikit-learn; easy local simulation
- **PySyft:** privacy-preserving ML with differential privacy hooks
- **NVIDIA FLARE:** enterprise-grade FL for medical imaging and finance
- **TensorFlow Federated:** Google's official FL research platform

`pip install flwr`: **Flower runs full FL simulation on a laptop in minutes**

Code Pointer

Federated simulation with Flower in 30 lines

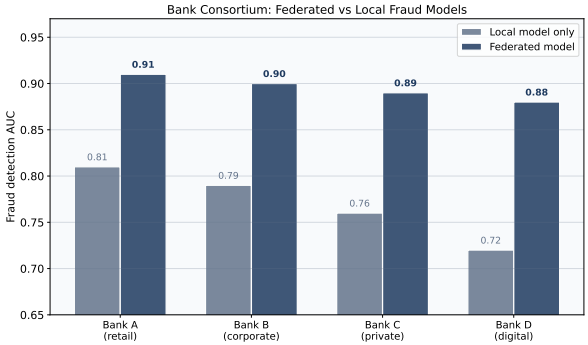
- Define a `FlowerClient` class with `fit()` and `evaluate()`
- Each client wraps a standard PyTorch/sklearn model
- Start simulation: `fl.simulation.start_simulation(client_fn, num_clients, config)`
- No infrastructure needed: all clients run in separate processes on one machine
- Add DP: wrap client with `fl.client.dp.DifferentialPrivacyClientWrapper`

Flower's simulation mode makes FL accessible without a distributed cluster

Finance Application: Bank Consortium Fraud Detection

Multiple banks train a shared fraud model without sharing customer records

- Each bank trains on its own transaction data and shares only model updates
- The consortium model catches fraud patterns seen at any member bank
- NVIDIA FLARE has deployed this pattern in production with several European banks
- Regulatory alignment: data stays within each bank's jurisdiction



Consortium FL for fraud: each bank contributes to the global model without exposing records

Practical Takeaway

When to use federated learning

- **Use FL when:** data cannot be centralised due to regulation, latency, or commercial sensitivity; and when the model quality gain from pooling outweighs the communication cost
- **Do not use FL when:** data can be centralised; or when a simple privacy agreement and pseudonymisation is sufficient
- Always pair FL with differential privacy if the gradient privacy guarantees matter
- Simulate locally with Flower before committing to distributed infrastructure

Start with the simplest solution: FL adds engineering complexity

Summary and Open Questions

Three things to remember

- FedAvg: local SGD on each client, weighted averaging on the server; multiple local steps reduce communication rounds significantly
- Non-IID data and privacy leakage via gradient inversion are the two main unsolved challenges
- Finance: bank consortia for fraud detection are the canonical use case; NVIDIA FLARE and Flower are production-ready frameworks

Open questions:

1. Can FL match centralised training under severe non-IID conditions?
2. What is the minimum privacy budget ϵ regulators will accept?
3. Can vertical FL (different features, same entities) scale to production?

Further reading: McMahan et al. (2017) Communication-Efficient FL; Kairouz et al. (2021) Advances and Open Problems