Federated Learning in Practice

An introduction to concepts, system architectures, and frameworks

Amir Ali-Pour
Postdoctoral researcher at ETS

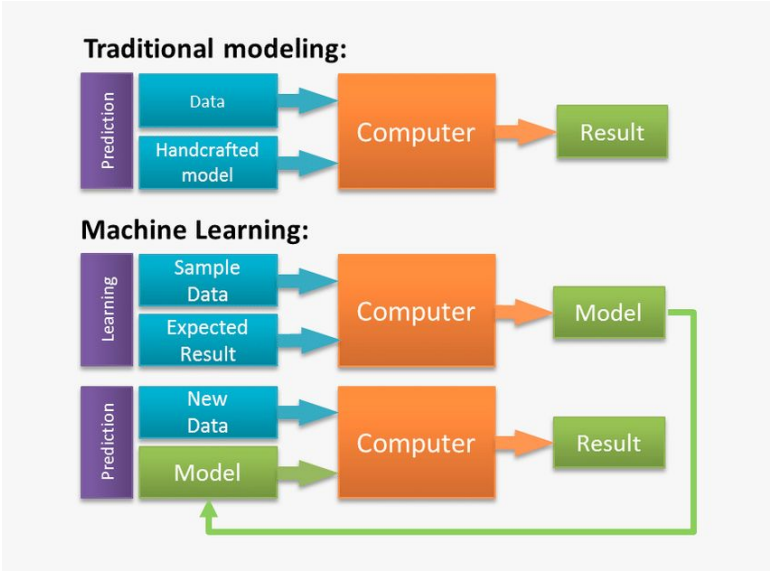
9 April 2025

Outline

1. Brief Introduction to Machine Learning
2. Central ML vs Distributed ML
3. Introduction to Federate Learning
4. FL System Architectures and Topologies
5. FL Frameworks
6. Demo

Introduction: machine learning

- It is a data oriented approach to creating solutions for **sophisticated problems**:
  - Emotion detection
  - System anomaly detection
  - Vehicle steering (self driving vehicles)

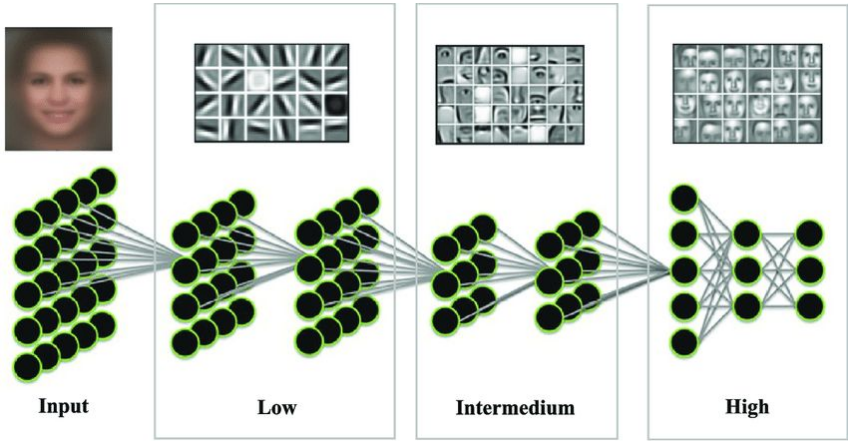


https://www.researchgate.net/figure/Difference-between-Traditional-Programming-and-Machine-Learning-3_fig2_343079524

Introduction: machine learning

- It is a data oriented approach to creating solutions for **sophisticated problems**:
  - Emotion detection
  - System anomaly detection
  - Vehicle steering (self driving vehicles)

- Systems that **learn patterns** from **data** without being explicitly programmed:
  - Emotion patterns
  - Mobile and Stationary object detection



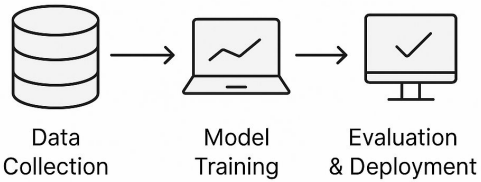
https://www.researchgate.net/figure/A-convolutional-neural-network-for-a-facial-recognition-application_fig10_330373042

Introduction to machine learning

The Classical Approach:

- **Data Collection** → Central server.
- **Model Architecture** → Using frameworks like TensorFlow, PyTorch.
- **Model Training** →Gradient Descent
- **Evaluation & Deployment** → Push to production.

Typical ML Pipeline



Introduction to machine learning

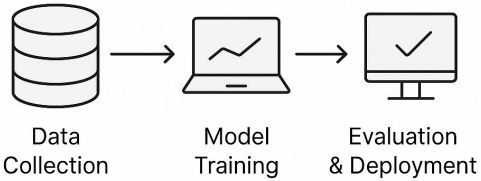
The Classical Approach:

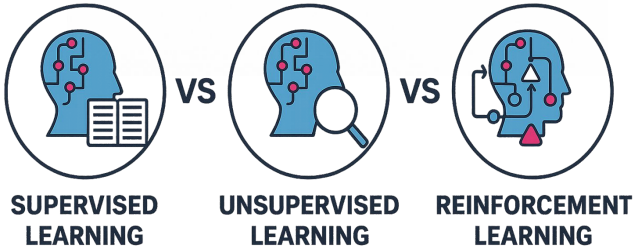
- **Data Collection** → Central server.
- **Model Architecture** → Using frameworks like TensorFlow, PyTorch.
- **Model Training** →Gradient Descent
- **Evaluation & Deployment** → Push to production.

Types of ML:

- **Supervised Learning**
- **Unsupervised Learning**
- **Reinforcement Learning**

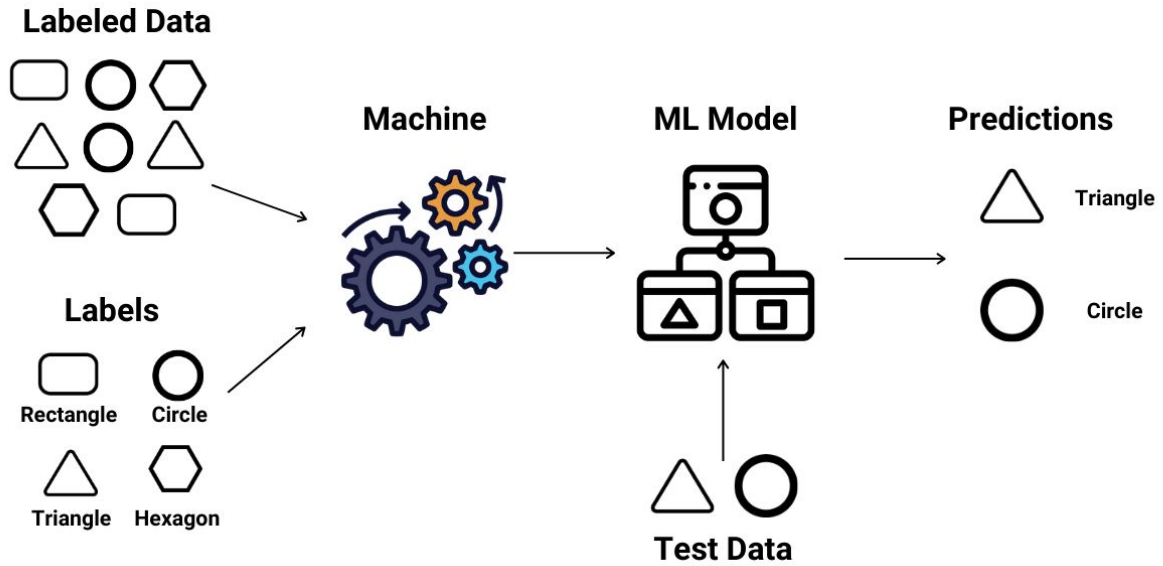
Typical ML Pipeline





Supervised ML

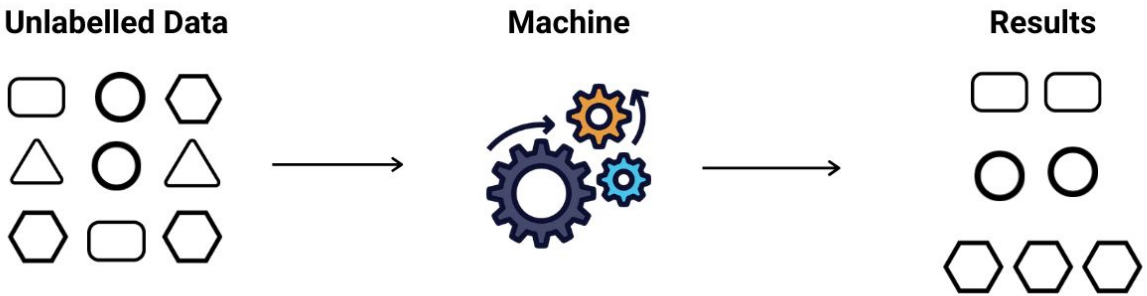
Supervised Learning: Learn from labeled examples (e.g., spam detection).



https://medium.com/@dhara732002/supervised-machine-learning-a-beginners-guide-9ac0b07eccbb

Unsupervised ML

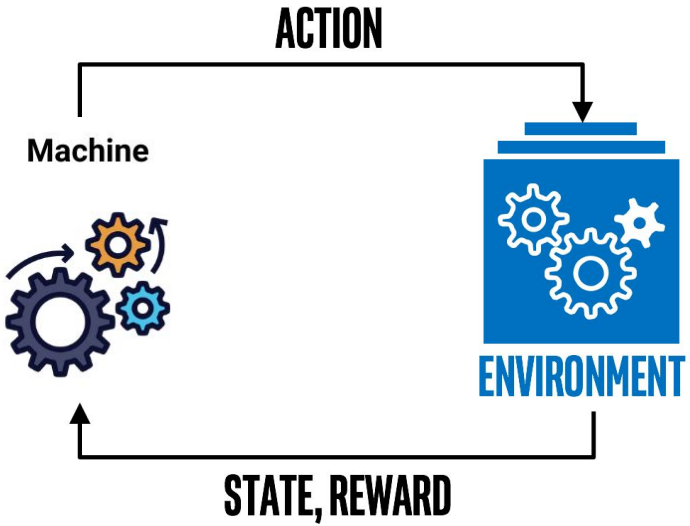
Unsupervised Learning: Find patterns in unlabeled data (e.g., customer segmentation).



https://www.linkedin.com/pulse/unveiling-top-5-unsupervised-machine-learning-algorithms-yadav-0yikc

RL

Reinforcement Learning: Learn from interactions (e.g., training a robot).



https://medium.com/ai%C2%B3-theory-practice-business/reinforcement-learning-part-1-a-brief-introduction-a53a849771cf

Central ML vs Distributed ML

Centralized ML: Collect all data in one place

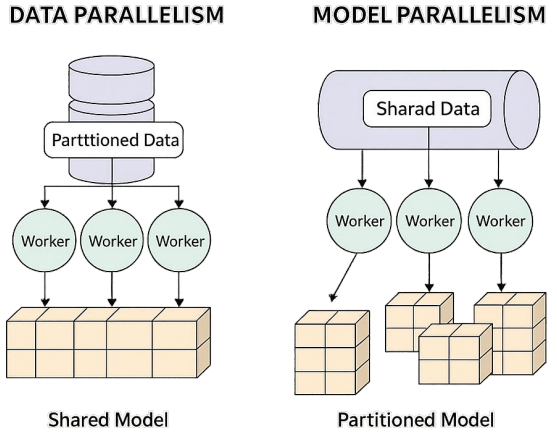
- Easy to **deploy** and **train**
- **Costly** due to big data processing.
- Can violate **data privacy**.

Distributed ML: Data/computation is distributed across machines.

Distributed ML

Why Distributed ML

- Need for a central HPC is elevated
- Compliant towards Privacy regulations (GDPR, HIPAA) limit data sharing.
- Lesser Bandwidth occupation

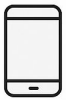


Distributed ML Models:

- Data parallelism
- Model parallelism

Federated Learning

Training a global ML model across many devices **without sharing raw data**.



- Each device:
  - Trains model locally.
  - Sends **model updates** (not data!) to the server.

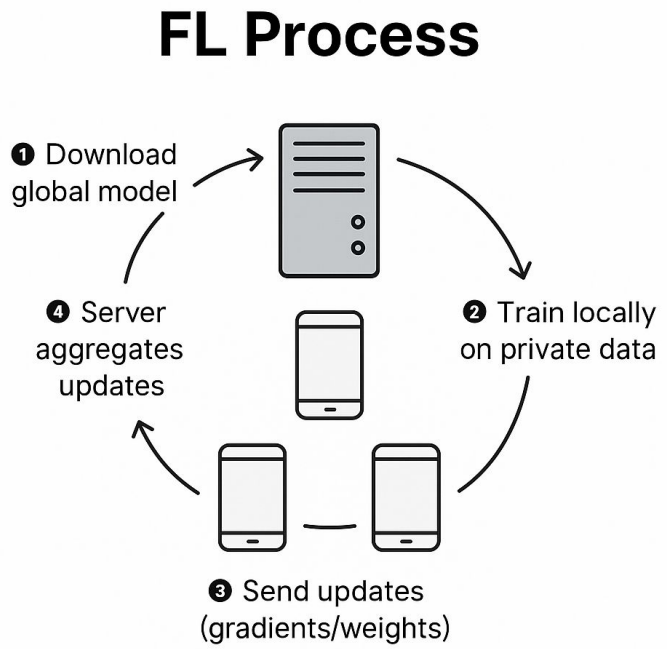
- Server (aggregation site):
  - Aggregates updates to improve the global model.
  - Sends the new global model back to devices.

Federated Learning

Training a global ML model across many devices **without sharing raw data**.

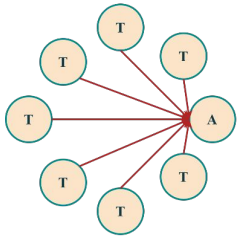
- Each device:
  - Trains model locally.
  - Sends **model updates** (not data!) to the server.

- Server (aggregation site):
  - Aggregates updates to improve the global mode
  - Sends the new global model back to devices.



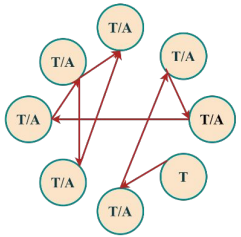
FL Topologies

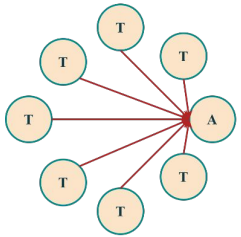
- **Centralized FL (CFL):** All clients send updates to one central server.



FL Topologies

- **Centralized FL (CFL):** All clients send updates to one central server.
- **Decentralized FL (DFL):** Clients communicate with each other (peer-to-peer).

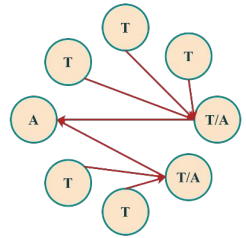


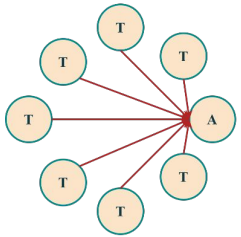


a) CFL

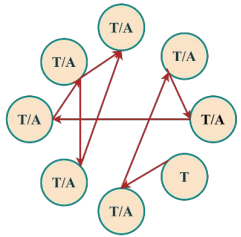
FL Topologies

- **Centralized FL (CFL):** All clients send updates to one central server.
- **Decentralized FL (DFL):** Clients communicate with each other (peer-to-peer).
- **Semi-Decentralized FL (SDFL):** Clients grouped into sites, sites coordinate updates.





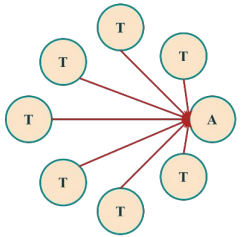
a) CFL



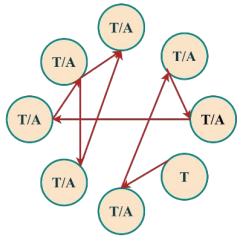
b) DFL

FL Topologies

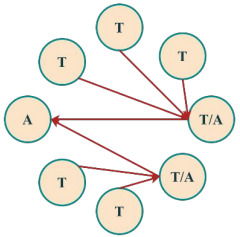
- **Centralized FL (CFL):** All clients send updates to one central server.
- **Decentralized FL (DFL):** Clients communicate with each other (peer-to-peer).
- **Semi-Decentralized FL (SDFL):** Clients grouped into sites, sites coordinate updates.
- **Hierarchical FL (HFL):** Clients → Edge servers → Central server.



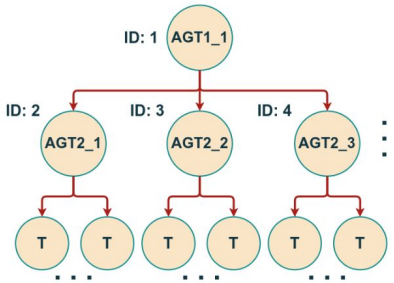
a) CFL



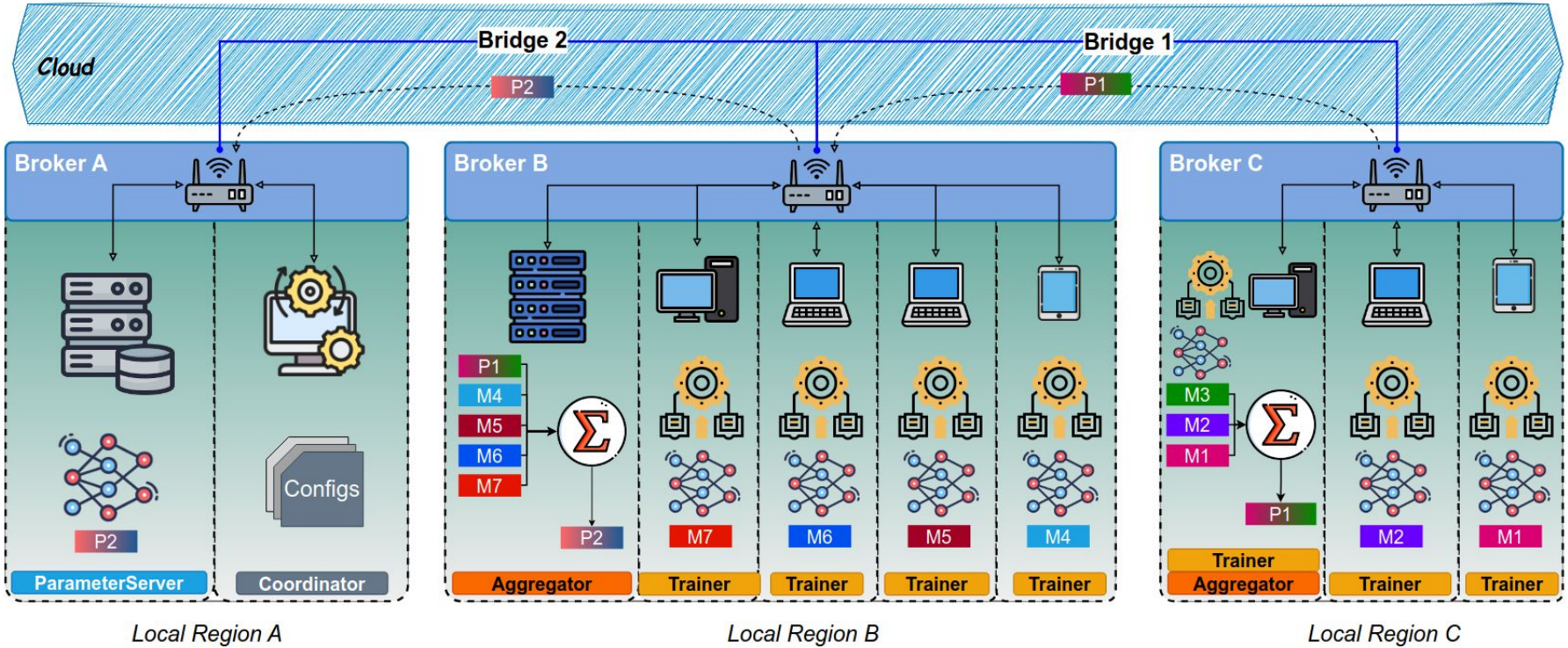
b) DFL



c) SDFL

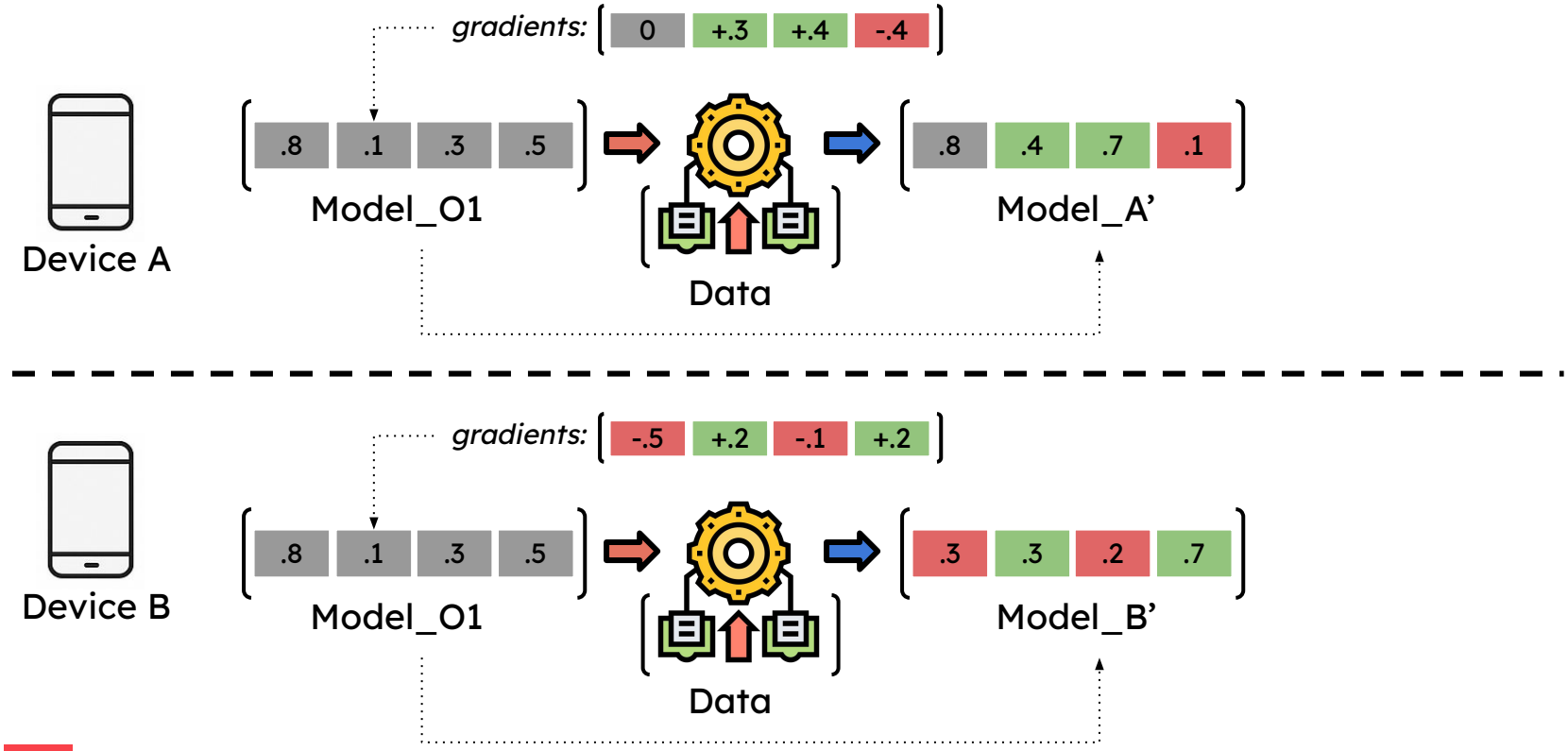


FL Topologies: An SDFL Example

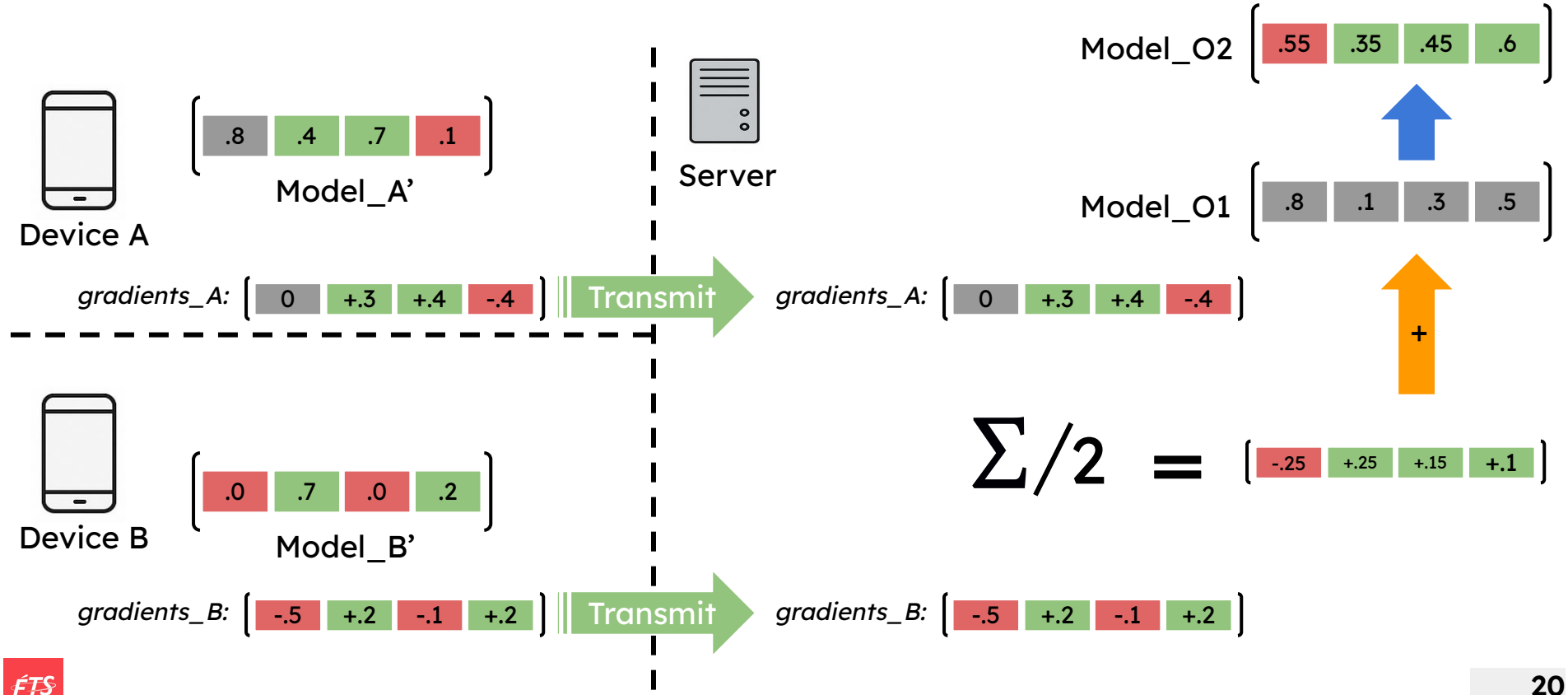


https://arxiv.org/abs/2503.13624

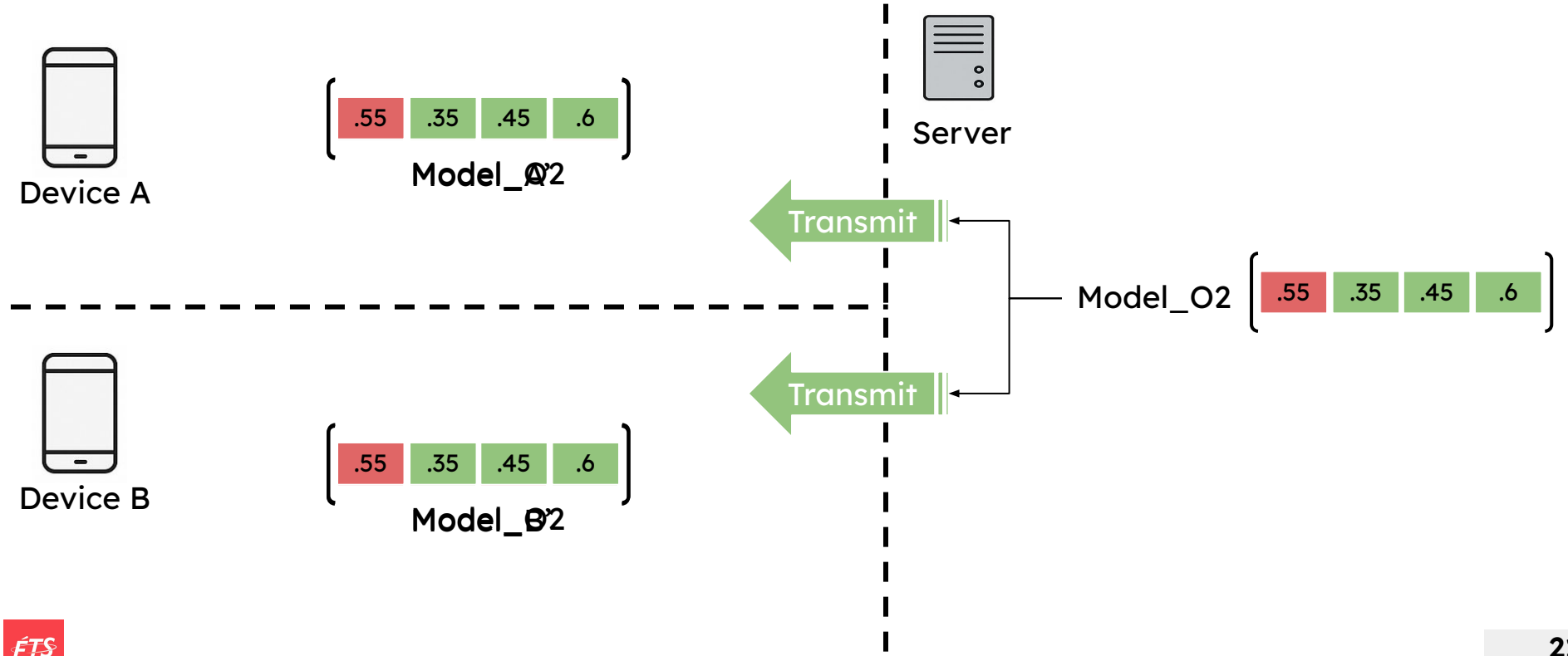
An FL Example: Training



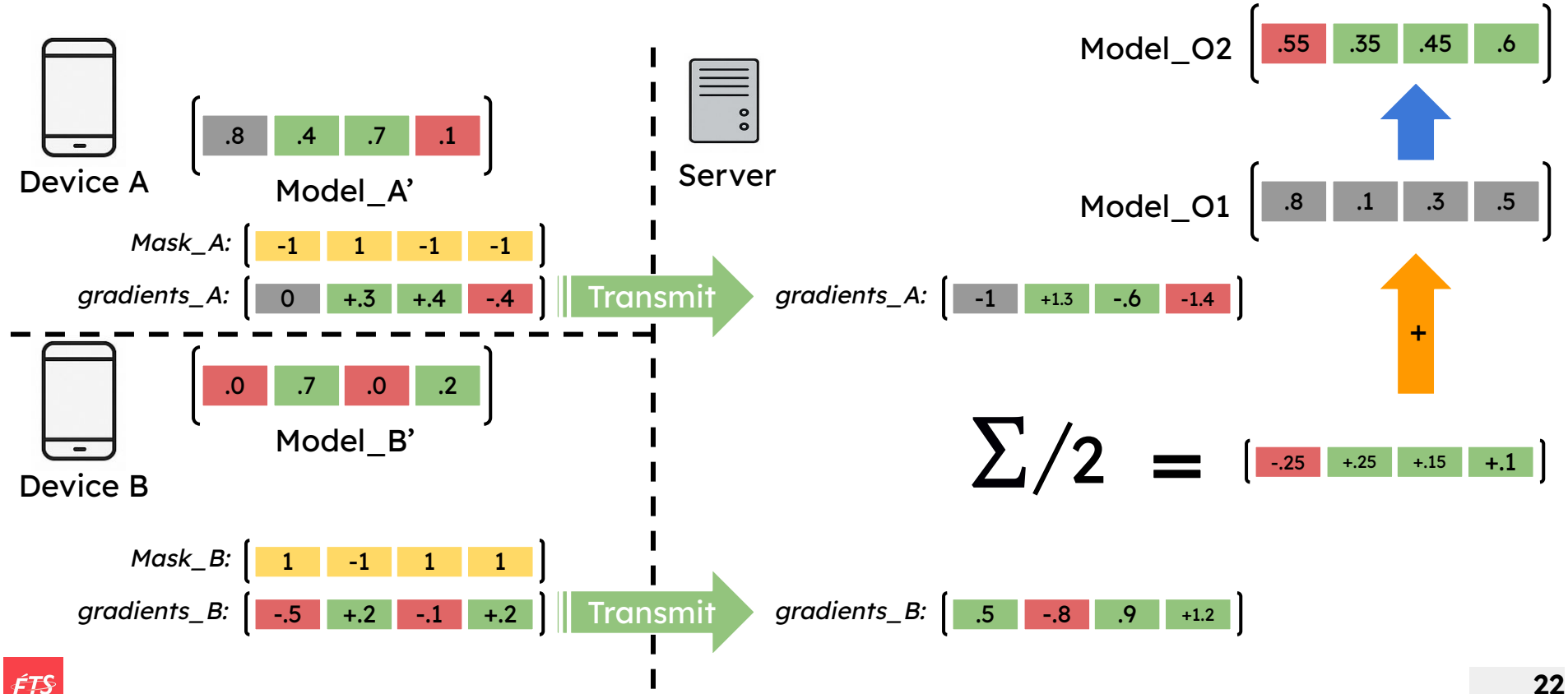
An FL Example: Aggregation



An FL Example: Global update

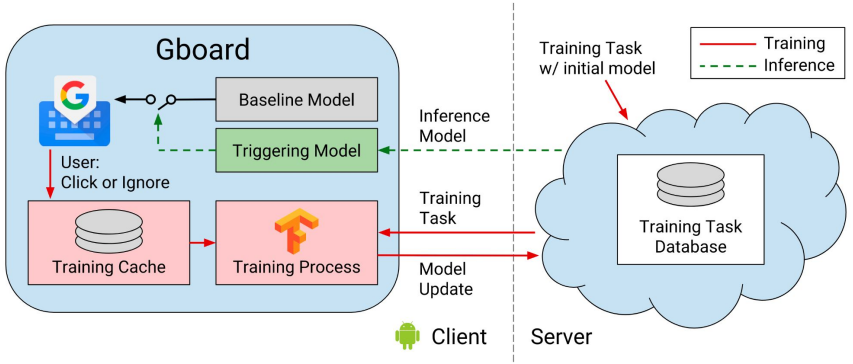


An FL Example: Secure Aggregation



Real-world FL Applications

- **Gboard (Google Keyboard):** Learn next-word prediction privately.



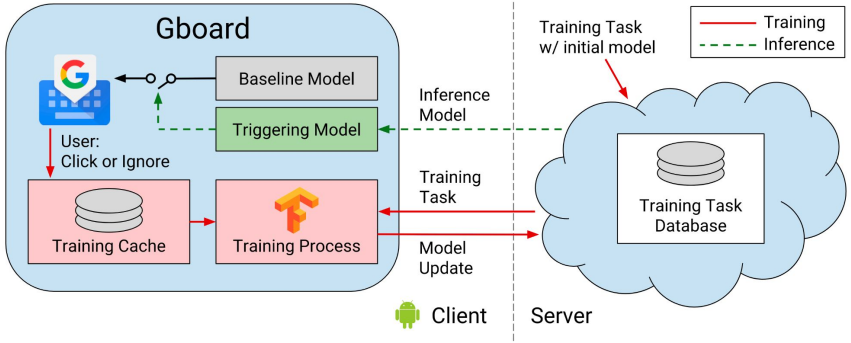
https://ar5iv.labs.arxiv.org/html/1812.02903

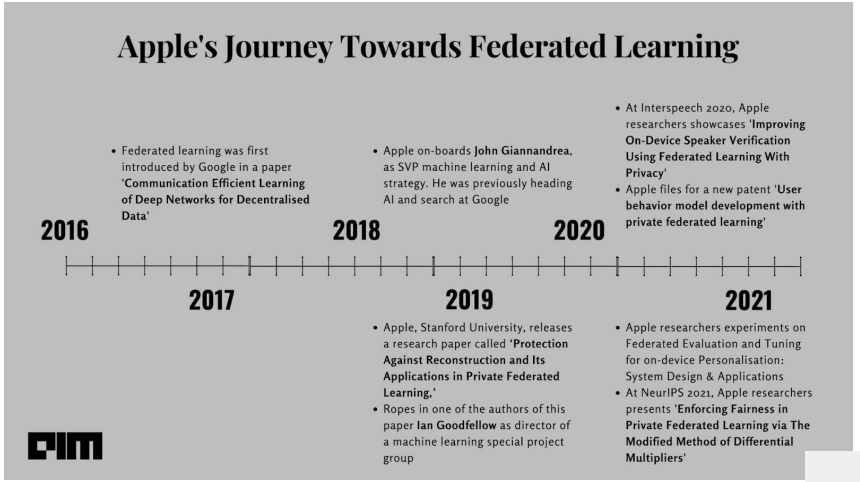
Real-world FL Applications

- **Gboard (Google Keyboard):** Learn next-word prediction privately.

- **Apple Siri:** Personalize voice recognition.

https://ar5iv.labs.arxiv.org/html/1812.02903

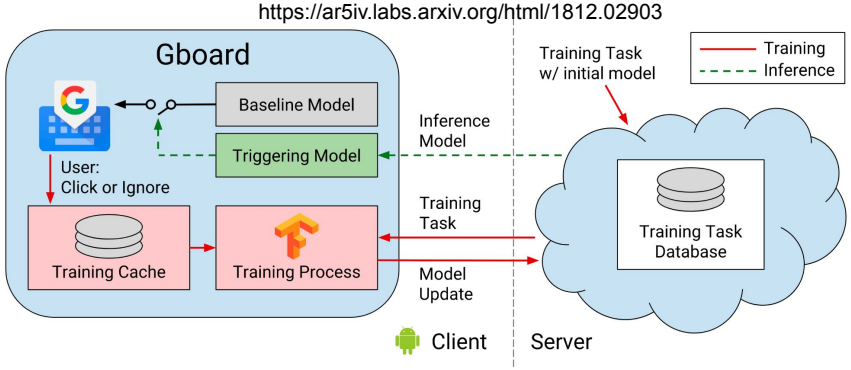




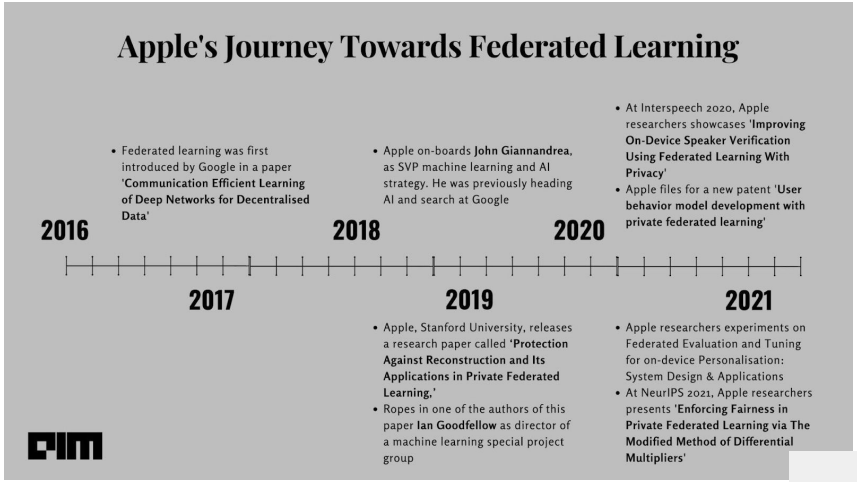
https://analyticsindiamag.com/ai-features/for-the-sake-of-privacy-apples-federated-learning-ap

Real-world FL Applications

https://ar5iv.labs.arxiv.org/html/1812.02903



- **Gboard (Google Keyboard):** Learn next-word prediction privately.
- **Apple Siri:** Personalize voice recognition.
- **Healthcare:** Collaborate across hospitals without sharing patient data.



https://analyticsindiamag.com/ai-features/for-the-sake-of-privacy-apples-federated-learning-ap

FL Advantages and Challenges

- **Advantages**
  a. Core Data Privacy is preserved
  b. Distributed Learning/Inference load
- **Challenges**
  a. **Communication overhead**: Sending models repeatedly is heavy.
  b. **System heterogeneity**: Clients differ in compute power, connectivity.
  c. **Data heterogeneity:** Non-IID data, adversarial data, noisy data
  d. **Model Privacy and security**: Even model updates can leak information.
     i. Membership inference
     ii. Gradient inversion
     iii. Sybil attack

FL Frameworks

- **TensorFlow Federated (TFF):** Research-oriented, Python.
- **PySyft:** Privacy-preserving federated learning, deep integration with PyTorch.
- **FedStellar (Nebula FL):** End-to-end FL platform for research and deployment.
- **Flower:** Lightweight and flexible framework for FL.
- **FedML:** End-to-end FL platform for research and deployment.
- **SDFLMQ:** A lightweight Modular framework with support for Pytorch, uses MQTT for communication

TFF Features

✅ **Rich Abstractions:**
TFF separates model logic from federated computation logic, making experiments modular.

✅ **Custom Algorithm Design:**
Fine-grained control for designing *novel federated learning* strategies.

✅ **Open-source and extensible:**
Well-documented and extensible for new federated algorithms and research.

❌ **No Native Multi-device Deployment (yet):**
TFF is mostly for *simulation* — **not** designed for *production at scale* across mobile devices or real-world networks (yet).

PySyft Features

✅ **Privacy by Design:**
Supports **Federated Learning**, **Secure Multiparty Computation**, **Differential Privacy**, and **Encrypted Computation** natively.

✅ **Real Deployment Support:**
Unlike TFF, PySyft is designed to run across *real distributed networks* (e.g., devices, servers).

✅ **Integration with PyTorch:**
Seamless use of standard PyTorch models and APIs with privacy enhancements.

✅ **Cross-device, cross-network:**
Supports WebRTC, Websockets, HTTP protocols to operate across real networks.

❌ **Still Evolving:**
APIs and structures have changed a lot over the years — breaking backward compatibility in some versions.

❌ **Deployment Overhead:**
Setting up real secure federated systems (with PyGrid, Duet, encryption) requires significant system engineering skills.

FedStellar (Nebula FL) Features

✅ **Decentralization by Design:**
No need for a trusted centralized server — good for environments where no single party should be in control.

✅ **Research and Simulation-Friendly:**
Good for studying decentralized P2P learning and designing resilient FL systems.

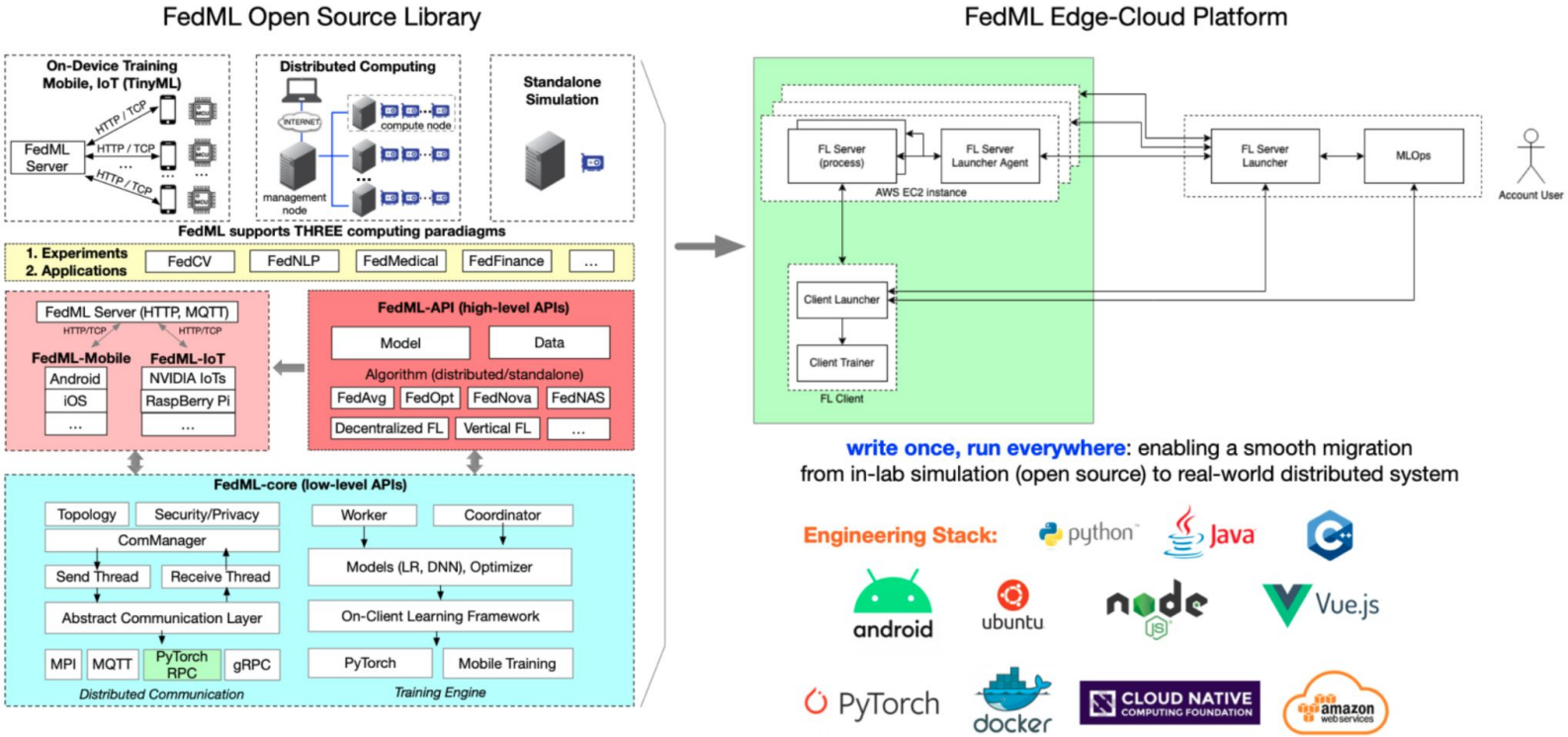
❌ **Early Stage Project:**
Compared to Flower, FedML, etc., it's newer and less mature.

❌ **Higher Communication Overhead (Peer-to-Peer):**
Direct peer-to-peer communication can become heavy, especially at scale, without careful optimization.

❌ **Limited Production Deployments:**
Primarily used for **research** and **proof-of-concept** projects — not widely deployed in industry yet.

❌ **Less Tooling:**
No full MLOps integration, monitoring, or deployment pipelines like FedML yet.

FedML Architecture



https://docs.tensoropera.ai/federate/getting_started

FedML Features

✅ **End-to-End Platform:**
From data partitioning to training, aggregation, deployment, and monitoring.

✅ **Rich Algorithm Support:**
FedAvg, FedProx, FedNova, personalized FL, split learning, decentralized FL, etc.

✅ **Distributed Communication Optimization:**
Handles bandwidth-aware communication, client dropouts, stragglers.

✅ **Federated MLOps Tools:**
Tools for monitoring, managing experiments, versioning, etc.

✅ **Benchmarking and Simulations:**
Includes datasets and benchmarks for FL research at scale.

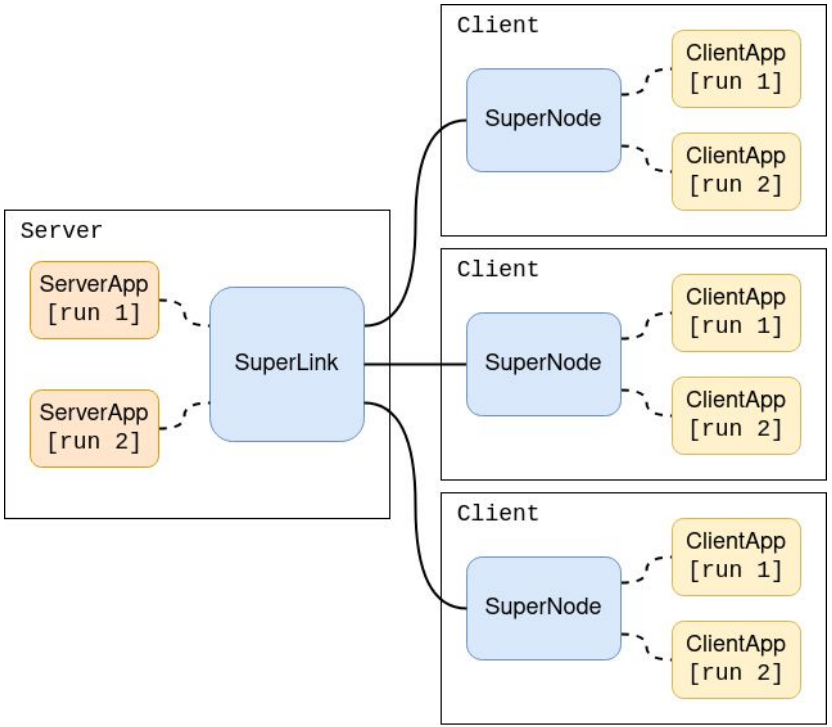
❌ **Steeper Learning Curve:**
Setting up a FedML project is **more complex** than Flower (but easier than building from scratch).

❌ **Heavier Infrastructure Needs:**
Running cross-device experiments with FedML often needs **GPU clusters** or **special setup** (e.g., Docker, Kubernetes).

❌ **More Configuration Overhead:**
Needs careful tuning of config files, orchestration, and sometimes lower-level system management compared to Flower.

FLOWER Architecture

- Hub-and-spoke topology (one server, multiple clients).
- Server is the coordinator as well.
- **SuperLink**: a long-running process that forwards task instructions to clients (SuperNodes) and receives task results back
- **SuperNode**: a long-running process that connects to the SuperLink, asks for tasks, executes tasks (for example, “train this model on your local data”) and returns task results back to the SuperLink.



https://flower.ai/docs/framework/explanation-flower-architecture.html

FLOWER Features

✅ **ML Framework Agnostic:**
Use any ML library — PyTorch, TensorFlow, even scikit-learn — inside clients and servers.

✅ **Production-ready:**
Good for real-world FL deployments across different devices, clouds, or hybrid systems.

✅ **Easy to Learn and Use:**
The simplest API among major FL frameworks. Setting up a basic FL system can take **<50 lines of code**.

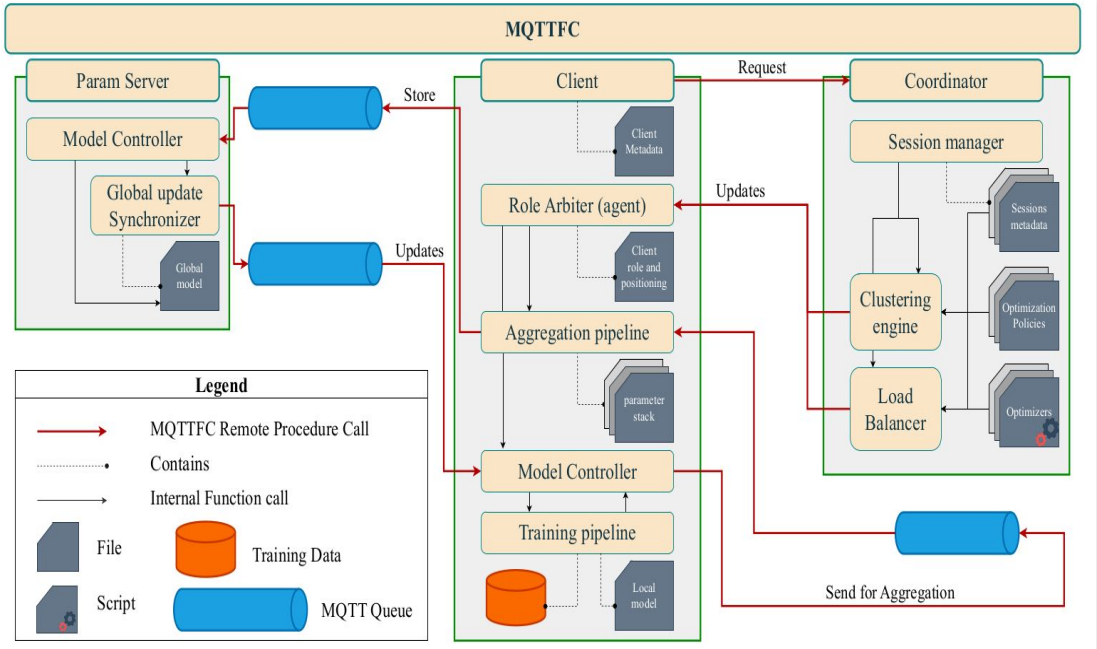
✅ **High Customization:**
Customize client behavior, server aggregation logic, training rounds, client sampling, etc.

❌ **Security Features are Limited (by default):**
No built-in Differential Privacy or Secure Aggregation (you must integrate them manually).

❌ **Not Built-in Specialized Federated Optimizers:**
Some optimizers for FL (like FedProx, FedNova) are available, but you may have to customize advanced optimization strategies.

SDFLMQ: Architecture

- Topic-based role association
- Dynamic role-management
- Modular design



https://arxiv.org/abs/2503.13624

SDFLMQ: Features

✅ **Support for Constrained IoT systems**
No heavy implementation for orchestration, and it is also based on a lightweight communication protocol
No need for a server system.

✅ **Simple to Integrate**
Client configuration is simple, and contribution is done via a few lines of code.

✅ **Support for various FL Topologies**
While still based on Pub/Sub, it can mimic CFL, DFL, and SDFL via topic-based client role configuration

✅ **Client Anonymity supported**
Clients submit their contribution through associated topics. Only the coordinator knows the client IPs

❌ **Security features are still very primitive**
Since it is a very new framework, the security features are not well implemented compared to competitors

❌ **Broker in the middle**
Existence of a central broker demands minimum two extra hops for model

Demo

- Central local machine learning
- Semi-Decentralized FL with SDFLMQ

Demo: Central ML with Pytorch

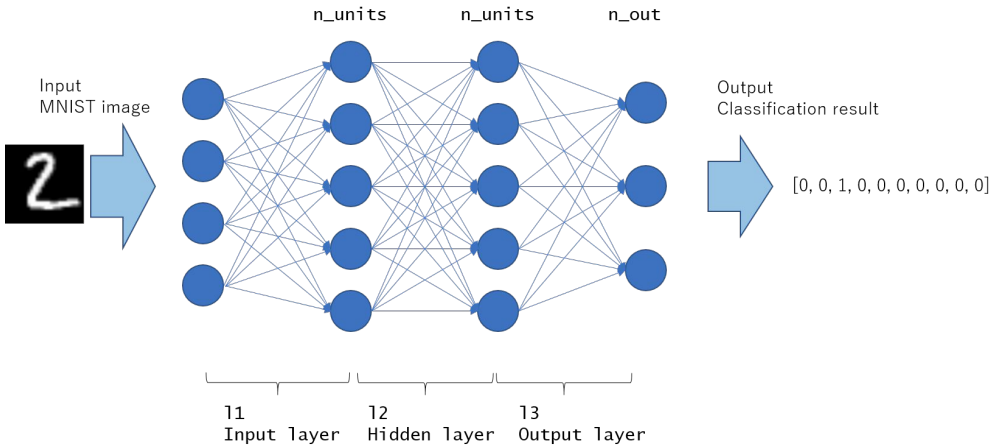
Handwritten digit detection with Multi–Layer Perceptron and MNIST dataset

Model specification:

- Num layers: 2
- Input: 784 * 256
- H-Layers : (256 * 128) (128 * 10)
- Num classifiers: 10
- Optimizer: Adam Optimizer

Dataset specification:

- Training dataset: 60,000 samples
- Test dataset: 10:000 samples
- Image size: 28 x 28 (784 pixels)
- Num classes: 10 (1-hot)



https://corochann.com/mnist-training-with-multi-layer-perceptron-536/

Demo: SDFL with SDFLMQ

- Handwritten digit detection with Multi–Layer Perceptron and MNIST dataset

Model specification:

- Num layers: 2
- Num classifiers: 10
- Optimizer: Adam Optimizer

Dataset specification per client:

- Training dataset:6000
- Test dataset: 10,000

FL specifications:

- Num clients: 10
- Aggregation function: FedAVG

Takeaway

- Distributed Learning **brings learning closer** to data sites, but data can still be moved.
- Federated Learning **mandates stationary data**, and moves model updates instead.
- CFL is suitable for big organizations, but costly due to need of a server and stable network
- DFL and SDFL is suitable for IoT and ecosystems with constrained systems
- There existing frameworks to Practice FL, but need further development before being standardized.