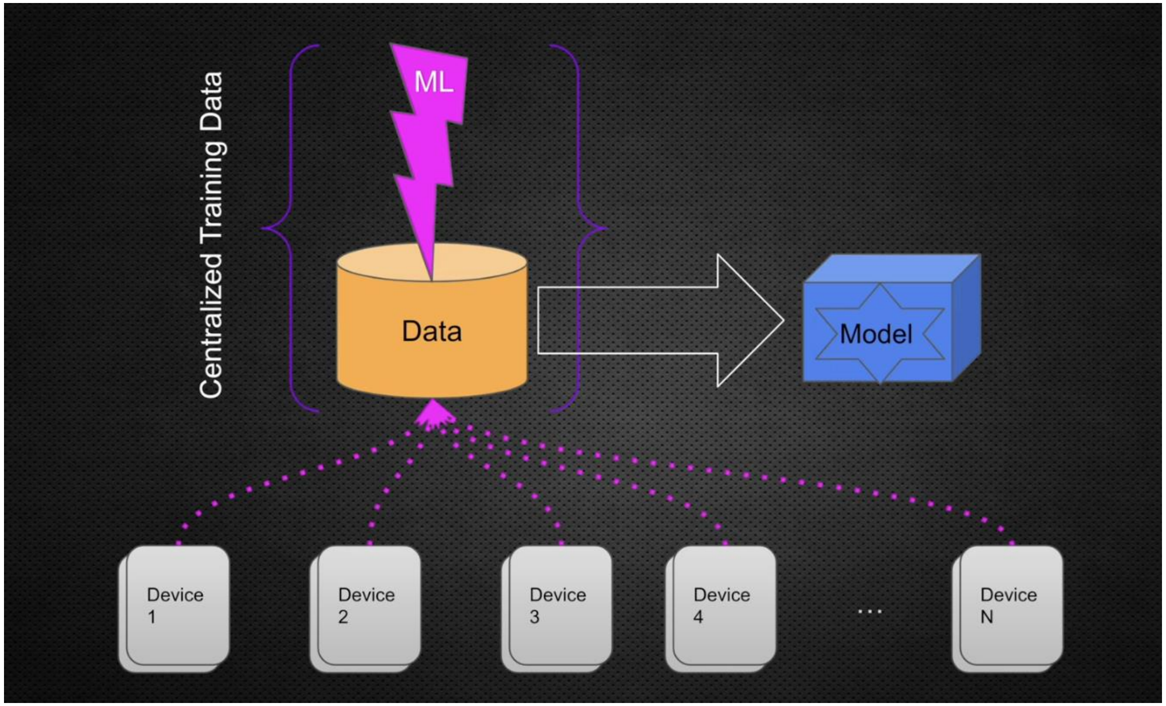
Centralized Machine Learning

- Centralized ML, users' data is collected and stored in a central server
- Privacy concerning: healthcare, finance - data privacy is of utmost priority.



The development of *privacy-preserving AI* methods was in urgent demand in such fields.

Centralized Machine Learning

- **Communication Costs and Latency** - raw data is transmitted to a central server to be processed and used to train ML models
- **Costly and time-consuming** when dealing with large datasets
- Increasing amount of data available - generating vast amounts of data from different sources - **storage and preprocessing** of data

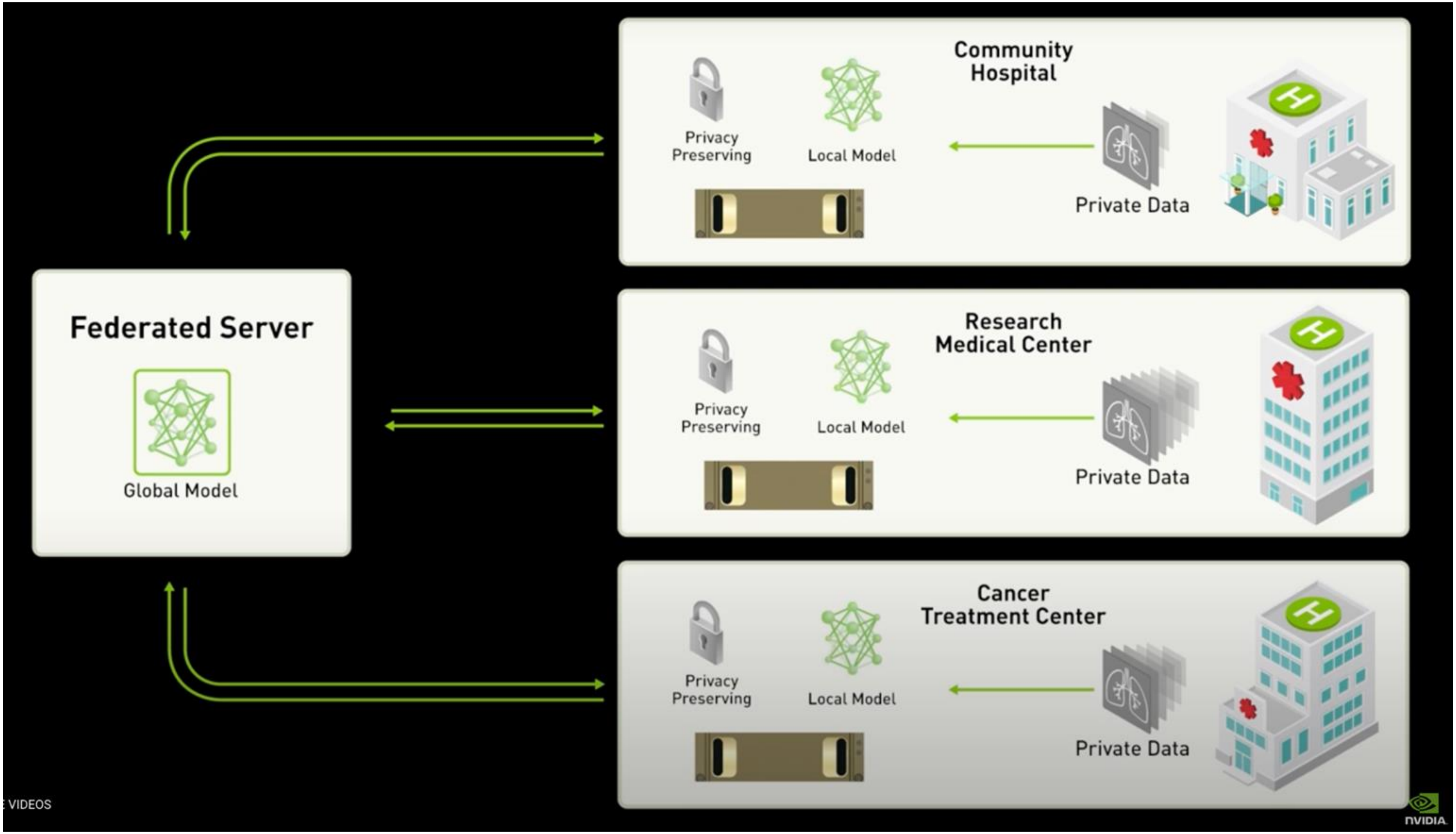
Centralized Machine Learning

- Limitations in data access
- Data can be distributed across different institutions or organizations
- Difficult to access or share data

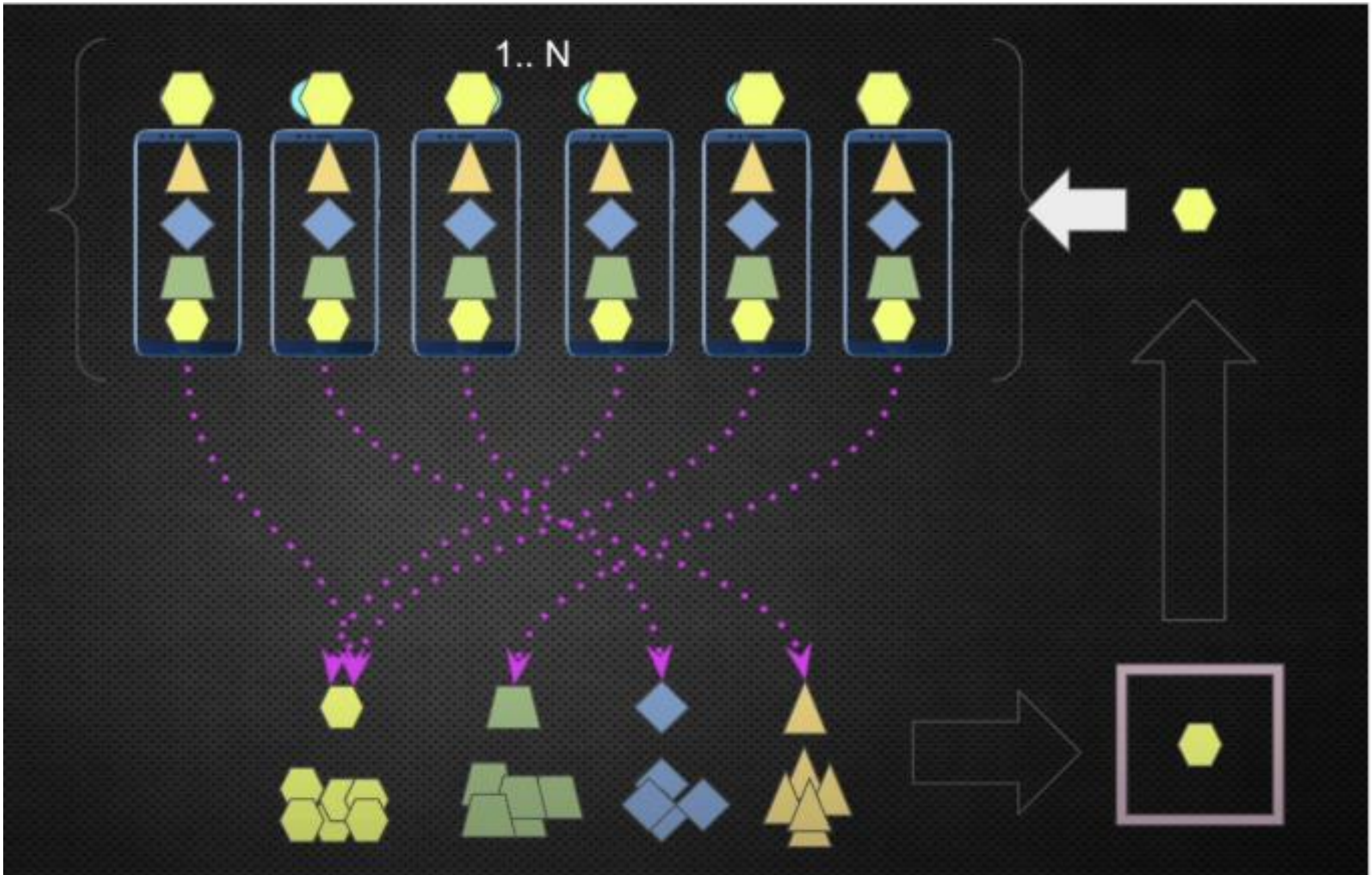
Federated Learning

- A **distributed** ML paradigm
- A ML model without explicitly sharing any data between any of the participants
- A network of clients or data owners
  - trains a local learning model on its data
  - shares the learning model information instead of their training data
  - the trained local models are aggregated to create a trained global learning model
- Inference phase: the trained global model is applied to new data instances

Federated Learning Nvidia



Federated Learning Google



FL - Applications Domains

Training ML models without centralizing sensitive data

Healthcare

Financial Services

Mobile Applications

IoT & Edge

FL - Applications Domains

Healthcare & Medical Imaging

- Patient data is highly sensitive
- Hospitals cannot easily share raw medical records or images

Brain Tumor Segmentation: multiple hospitals collaborate to train MRI analysis models without sharing patient scans. Each hospital keeps data locally while contributing to a global model

COVID-19 Detection: chest X-ray analysis models trained across international medical centers, enabling rapid diagnosis while preserving patient privacy.

Rare Disease Research: combining data from multiple institutions to study rare conditions without centralizing sensitive patient information

FL - Applications Domains

Financial Services & Fraud Detection

- Financial institutions - strict regulations and competitive concerns
- Transaction data is extremely sensitive and sharing it poses security risks

Cross-Bank Fraud Detection: Banks collaborate to identify fraud patterns across institutions without sharing customer transaction details.

Credit Risk Modeling: Financial institutions build better credit scoring models by learning from distributed data while maintaining customer confidentiality.

FL - Applications Domains

Mobile Keyboard Prediction (Google Gboard)

User typing data contains personal messages, passwords, and sensitive information. Sending this to servers would be a major privacy violation.

Next-Word Prediction: a phone learns from typing patterns locally. Model updates (not actual text) are sent to Google to improve the global model.

Personalization Without Privacy Loss: The keyboard adapts to the vocabulary, writing style while personal messages never leave the device.

- Millions of devices train local models
- Improved global model distributed
- Keyboard gets smarter while data stays private.

FL - Applications Domains

IoT & Edge Computing

IoT devices generate massive amounts of data. Sending everything to the cloud is expensive, slow, and raises privacy concerns.

Smart Home Devices: Voice assistants and smart cameras learn from usage patterns across millions of homes without uploading video/audio footage.

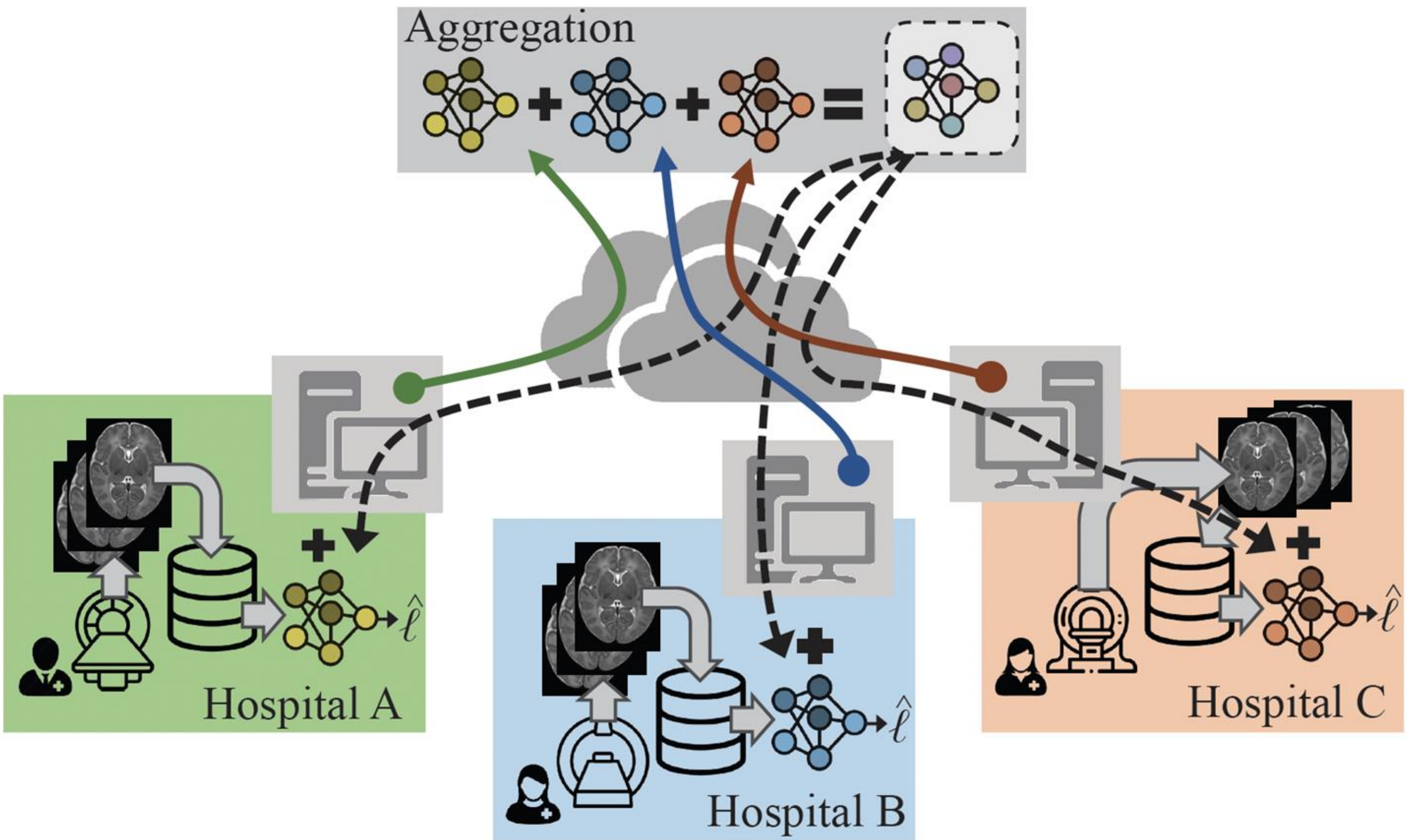
Autonomous Vehicles: Tesla uses FL to improve driving models. Each vehicle learns from local driving conditions and shares insights without transmitting raw data.

Industrial IoT: Factory sensors and manufacturing equipment collaborate to detect anomalies and optimize processes while keeping proprietary operational data..

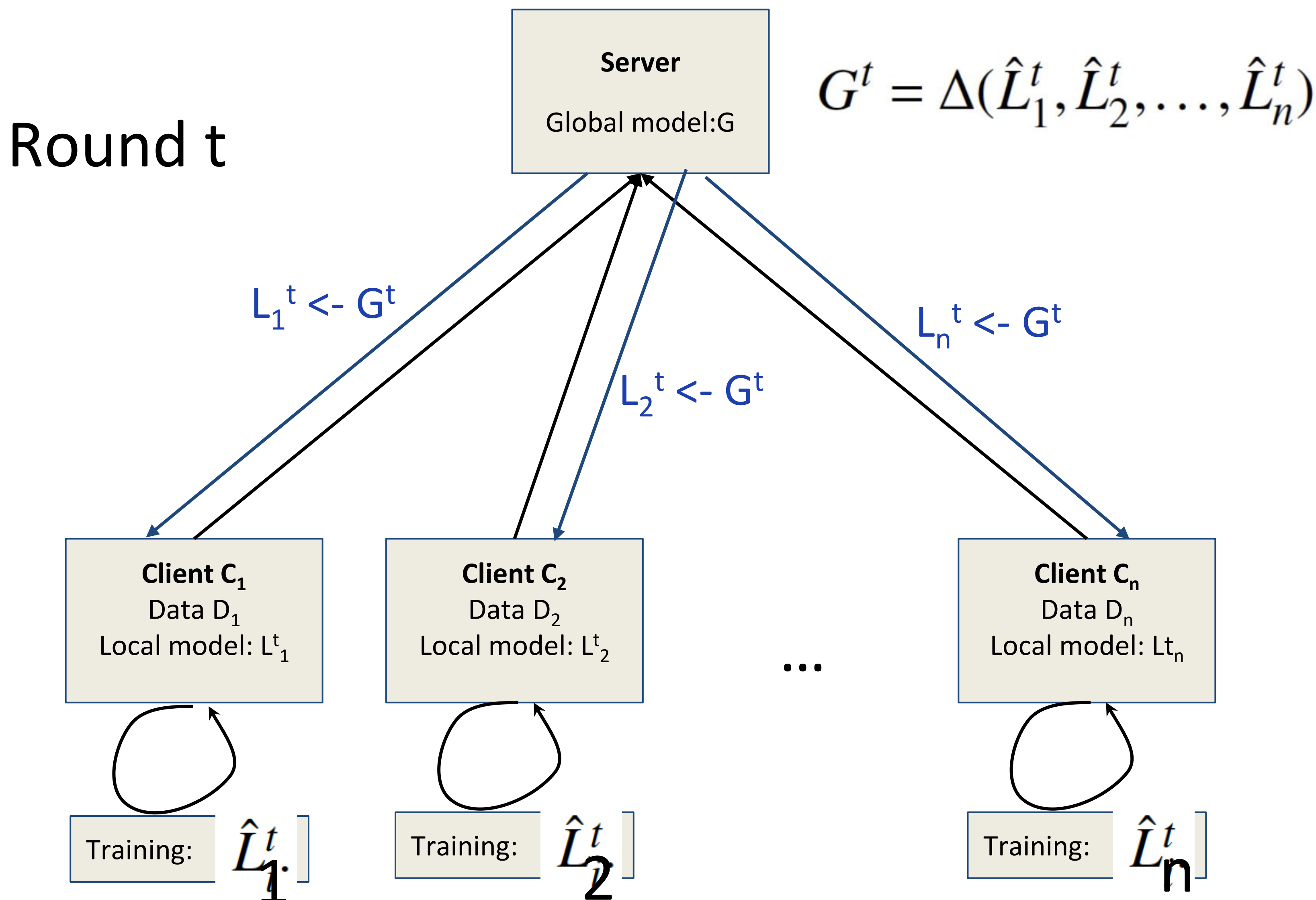
Smart Cities; Traffic cameras and sensors improve traffic flow prediction without centralizing surveillance data, preserving citizen privacy.

Federated Learning

- Synchronous or asynchronous - depending on the data availability of nodes and the trained model



Federated Learning



Federated Learning

- A set of clients or data owners $\{C_1, \ldots, C_n\}$
- With their local training data $\{D_1, .. D_n\}$
- Each of these clients C_i owns a local learning model L_i expressed as the parameters $\{L_1, .. L_n\}$
- Learn a global learning model G using data across clients through an iterative learning process known as **round of learning**
- In each learning round **t** each client trains its local model over their local training data D_i -> update of the local parameters L_i^t to $\hat{L}_i^t$.

Federated Learning

- Global parameters G^t are computed by aggregating the trained local parameters $\{\hat{L}_1^t, \ldots, \hat{L}_n^t\}$ using:
  - a fixed federated aggregation operator Δ
  - the local learning models are updated with the aggregated parameters

$$G^t = \Delta(\hat{L}_1^t, \hat{L}_2^t, \ldots, \hat{L}_n^t)$$

$$L_i^{t+1} \leftarrow G^t, \quad \forall i \in \{1, \ldots, n\}.$$

- Updates among the clients and the server are repeated for the learning process until a given stop criteria is met.
- G will sum up the knowledge modelled in the clients.

Formal Definition of Federated Learning

Network of clients $\{C_1, C_2, ..., C_n\}$

Local datasets $\{D_1, D_2, ..., D_n\}$

Local models $\{L_1, L_2, ..., L_n\}$

Global model G

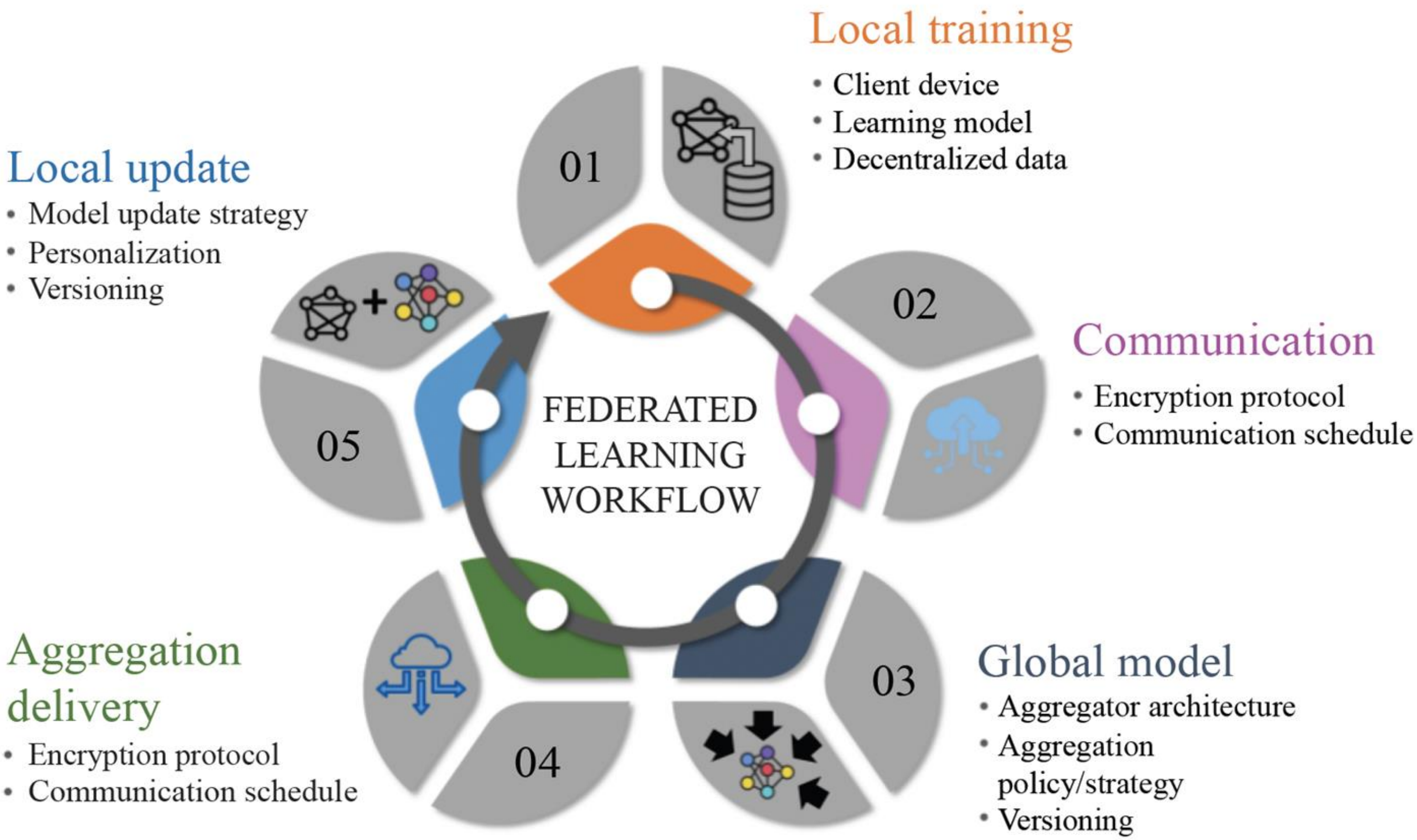
For each round t:

1. Each client C_i trains on local data D_i
2. Updates local parameters: $L_i^t \rightarrow \hat{L}_i^t$
3. Server aggregates: $G^t = \Delta(\hat{L}_1^t, \hat{L}_2^t, ..., \hat{L}_n^t)$
4. Distribute updated model: $L_i^{t+1} \leftarrow G^t$

Why Federated Learning?

- **Data Privacy**: Sensitive data remains on local devices
- **Communication Efficiency**: Only model updates shared
- **Data Access**: Enables collaboration across institutions

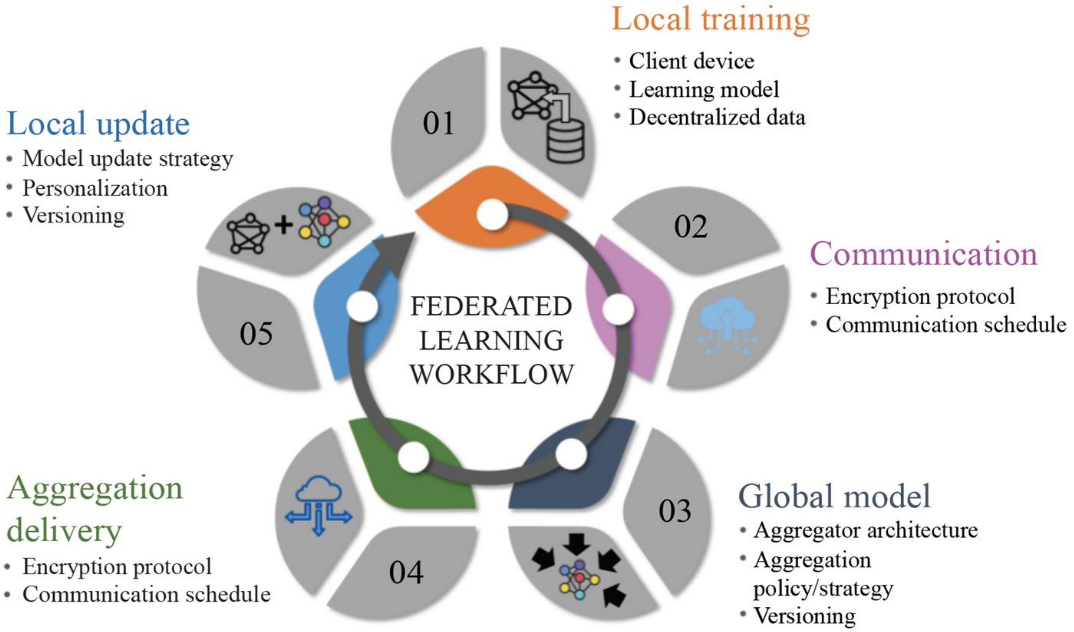
Steps of Federated Learning



Steps of Federated Learning

Local training

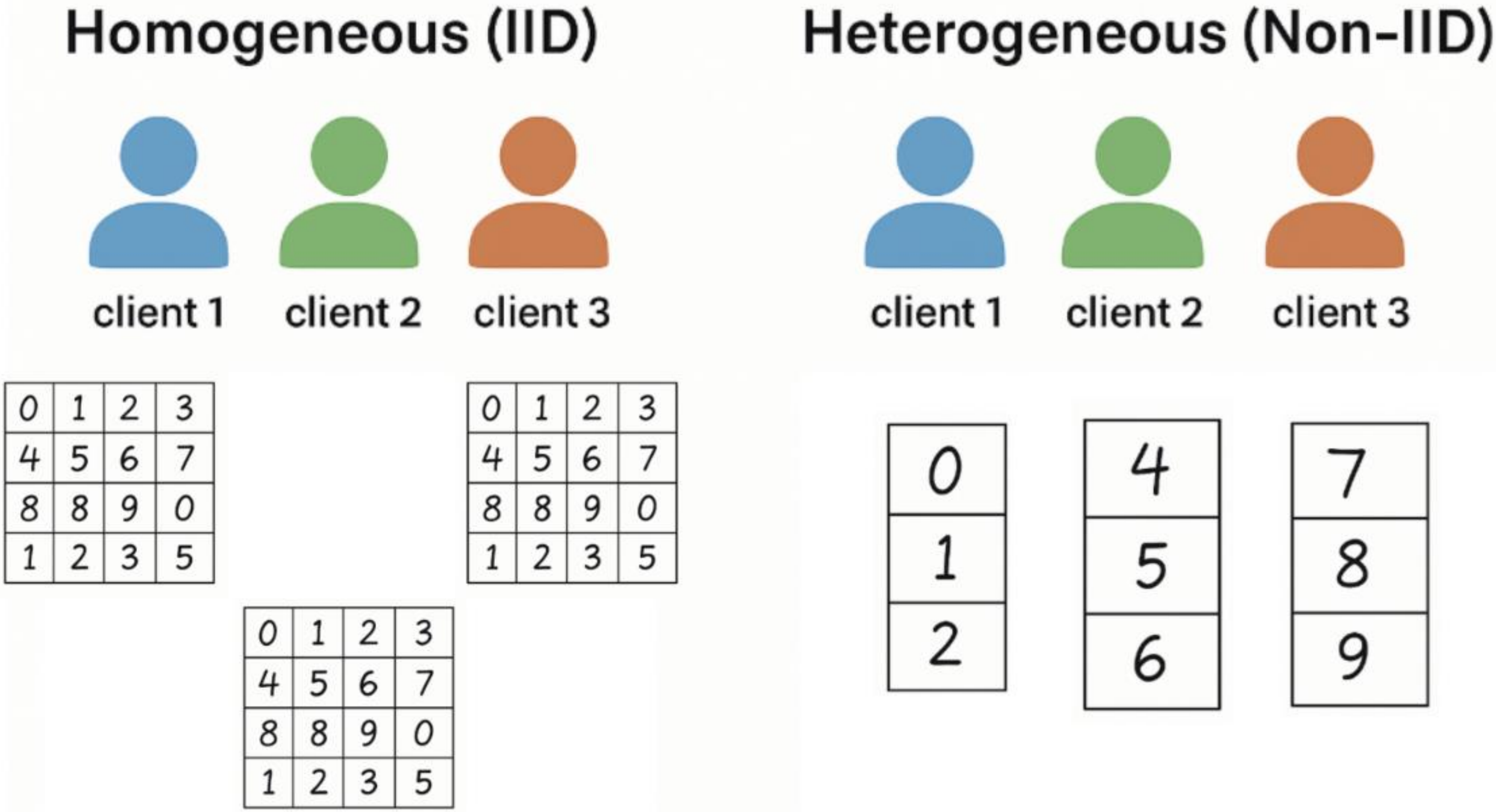
- starts with the local training of each of the local ML models by each of the data owner nodes
- all these locally trained learning models shared all the aspects concerning training
- hyperparameters (number of epochs, batch size, learning rate) may differ among clients



Steps of Federated Learning

Local training

- **Learning Model**: each device or node trains its model
- **Clients**: nodes store data and train models
- **Decentralized Data**:
  - Data is distributed among different devices
  - Data is inaccessible and not shared with any third-party
  - Data distribution:
    - **Homogeneous or independent and identically distributed (IID):** the data of each client follows the same data distribution
    - **Heterogeneous or non independent and identically distributed (non-IID)**: the data of each client follows a different data distribution
      - the feature space of the clients' data are different, but they share the same goal


Homogeneous (IID)
Heterogeneous (Non-IID)
client 1
client 2
client 3
client 1
client 2
client 3

Independent and Identically Distributed

Scenario	Description	Why it's IID
1. MNIST digits split randomly	MNIST dataset (digits 0–9). Randomly shuffle all 60,000 images and split them evenly among 10 clients.	Each client has roughly the same proportion of all digits, so their label and feature distributions are identical.
2. Federated image classification with balanced sampling	Each client receives 5,000 randomly chosen CIFAR-10 images.	The class and feature proportions per client mirror the global dataset.
3. Sensor networks measuring the same phenomenon	Multiple temperature sensors measuring data in a controlled environment, each collecting readings from the same distribution (same model, same conditions).	Data are independent and identically distributed across sensors.
4. Healthcare FL with standardized sampling	A central authority ensures each hospital gets a randomized and balanced subset of global medical images.	Same underlying data distribution across hospitals.

Non Independent and Identically Distributed

Type	Description	Example scenario
1. Label distribution skew	Clients have data from different classes.	MNIST: - Client 1 has digits 0–1 - Client 2 has digits 2–3 - Client 3 has digits 4–9
2. Feature distribution skew	Clients have similar labels, but input features differ.	Hospitals in different regions have different patient demographics (age, ethnicity, equipment types).
3. Quantity imbalance	Clients have very different amounts of data.	- Client 1 has 10,000 samples - Client 2 has 200 samples - Client 3 has 30 samples
4. Temporal non-IID	Data distributions shift over time.	- Clients are IoT devices collecting data at different times of day (morning vs night).

Steps of Federated Learning

Communication

- Enables the coordination and aggregation of model updates generated by the participating nodes
- Protection of the privacy and security of the data when paired with data security techniques
- Communication schedule: *communication can be both synchronous and asynchronous:*
  - *a central server that handles the collection of all local models,*
  - *distributed across multiple nodes in the network*
- Privacy Protocols: no training data is shared during FL communications
  - the information shared is susceptible to privacy leaks
  - communications are one of the weak points of FL regarding susceptibility to attacks

Steps of Federated Learning

Aggregation

- The local model updates generated by each node are combined by means of a specific aggregation operator
- The result is incorporated to update and create a trained global learning model.
- The aggregation mechanism: depends on the task addressed:
  - Federated Averaging (FedAvg) - the most common one - when the ML model can be expressed as a vector of weights

Steps of Federated Learning

Local update

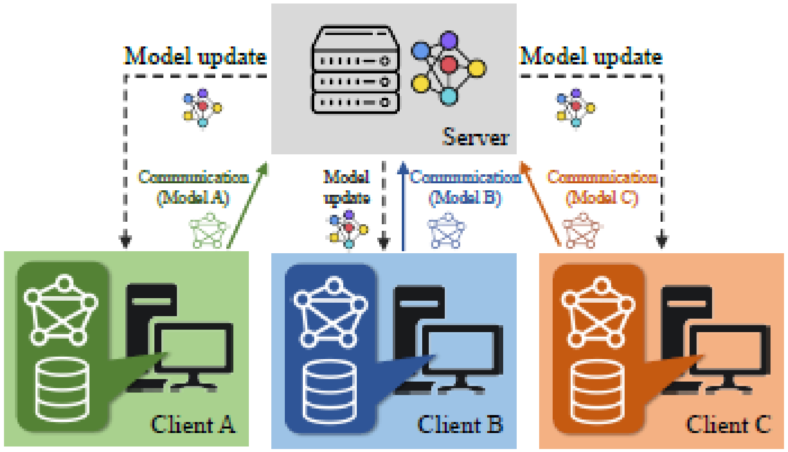
- Updating the local models stored in the different nodes with the new global model

- Other update strategies: combining the local and global models - personalization of the clients to their local data.

Architectures of FL

Client-server architecture

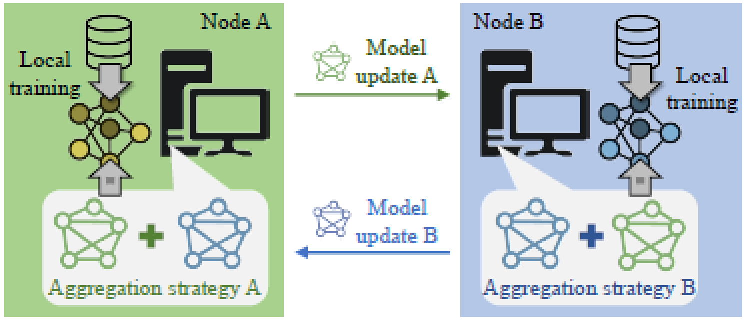
- Server: A manager node responsible for the coordination and aggregation of model updates
- Clients: nodes which own data and are responsible for training their local models
- Requires a high level of trust in the server
- Vulnerable to attacks



Architectures of FL

Decentralized (Peer-to-Peer) Architecture

- No central server
- Clients communicate directly with each other
- Clients form a network (e.g., ring, mesh, or random graph)
- Clients own both the training data and aggregate model updates directly with neighbors
- No fixed coordinator of the learning process.



- Complex to implement
- The communication costs increase
- Elevated level of security and data privacy

Architectures of FL

Clustered or Personalized Federated Learning

- Clients form clusters or get individualized models
- Group similar clients or personalize global models

- Handles non-IID data better, improves accuracy for diverse users
- More complex coordination and model management

Categories of FL

- Most relevant FL categories:
  - Data Feature, Label and Sample Space:

- Based on the dimension in which the data is partitioned across clients, there are different categories:
  - the feature space
  - the label space
  - the sample space

Horizontal FL

- **Same Features, Different Samples**
- Participants share the **same feature** space but have **different sample sets**
- No user overlap

Hospital A

Features: Age, Blood Pressure, Heart Rate

Patients: 1, 2, 3, 4, 5

Hospital B

Features: Age, Blood Pressure, Heart Rate

Patients: 6, 7, 8, 9, 10

Vertical FL

- **Different Features, Same Sample**
- Participants have different features for the same set of users/entities
- User overlap

Bank

Features: Credit Score, Income, Loan History

Users: Alice, Bob, Charlie, Diana

E-commerce

Features: Purchase Frequency, Cart Value, Product Category

Users: Alice, Bob, Charlie, Diana

Federated Transfer Learning FL

- **Different Features and Different Samples**
- Participants have both different features and different samples - minimal or no overlap
- No overlap needed
- Transfer knowledge across domains through shared representations, even when data is completely different

Hospital A (Radiology)

Features: X-ray images, CT scans

Patients: 1, 2, 3, 4, 5

Hospital B (Pathology)

Features: Tissue slides, Lab results

Patients: 6, 7, 8, 9, 10

Categories of FL

Type	What differs between hospitals	Analogy
HFL	Different patients, same data schema	Hospitals record the same variables for different people.
VFL	Same (some) patients, different data types	One hospital has the lab data, the other has the imaging data for the same patients.
TFL	Different patients *and* different data types	Hospitals in different domains collaborate — one has clinical data, the other has genetic data — and transfer knowledge.

Horizontal FL

- Same Features, Different Samples
- Participants share the **same feature** space but have **different sample sets**
- No user overlap

Hospital A

Features: Age, Blood Pressure, Heart Rate

Patients: 1, 2, 3, 4, 5

Hospital B

Features: Age, Blood Pressure, Heart Rate

Patients: 6, 7, 8, 9, 10

Horizontal FL

- Few organizations (2-100) with large, high-quality datasets
- Balancing collaboration with competition and compliance

Scale

- 2 to 100 organizations
- Each has substantial data
- All participate each round

Infrastructure

- Reliable data centers
- Stable high-speed networks
- Powerful compute resources

Examples

- Hospital collaborations
- Bank fraud detection
- Multi-factory optimization

Concerns

- Legal agreements (contracts)
- Competitive sensitivity
- Regulatory compliance

Horizontal FL

FedAvg Algorithm - Federated Averaging

1. **Server Initialization:** Server initializes global model w^0 and broadcasts to participants
2. **Client Selection:** Server randomly selects fraction C of clients (e.g., C=0.1 for 10%)
3. **Local Training:** Each selected client k trains on local data D_k for E epochs using SGD: update w_k^{t+1} (based on w_k^t)

1. **Upload Updates:** Clients send updated weights w_k^{t+1} to server

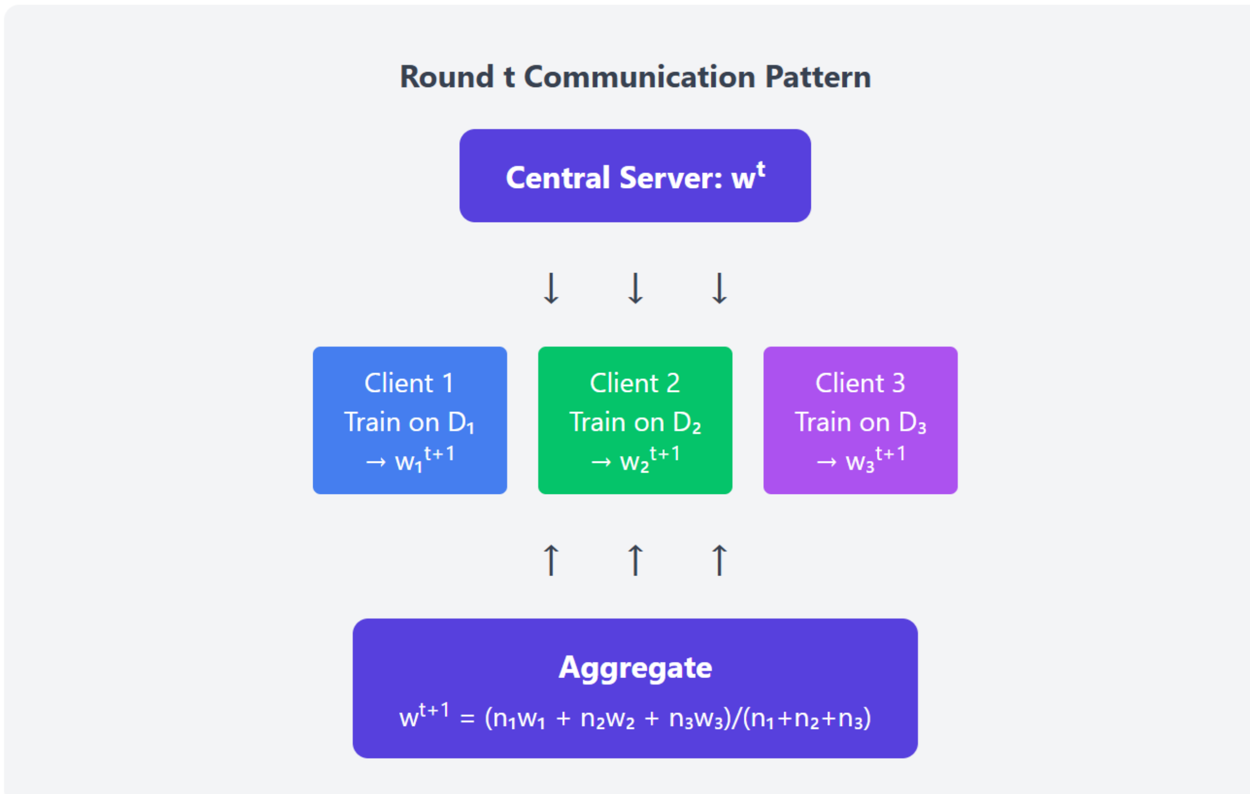
1. **Aggregation:** Server computes weighted average of client models

$$w^{t+1} \leftarrow \Sigma_k \ (n_k/n) \cdot w_k^{t+1}$$

1. **Broadcast & Repeat:** Server broadcasts w^{t+1} to all clients. Repeat steps 2-6 until convergence

Horizontal FL

FedAvg Algorithm



Client 1: n_1=100
Weight: 100/300 = 0.33

Client 2: n_2=150
Weight: 150/300 = 0.50

Client 3: n_3=50
Weight: 50/300 = 0.17

Horizontal FL

- Real-world federated data is rarely Independent and Identically Distributed (IID)
- Convergence issues and poor performance

1. Feature Distribution Skew: Different feature distributions across clients

- Healthcare:
  - Elderly care hospital (avg age 75, BP 145/90)
  - Pediatric hospital (avg age 8, BP 95/60)
- Finance:
  - Luxury bank branch (high-income clients, large transactions)
  - Community credit union (modest incomes, small transactions)
- IoT:
  - Sensors in Arctic (temperature -30°C)
  - Tropical region (temperature +35°C)

Horizontal FL

- Real-world federated data is rarely Independent and Identically Distributed (IID)

2. Label Distribution Skew: Different class distributions across clients

- Mobile Keyboard:
  - China (90% Chinese characters)
  - France (95% French words)
  - USA (80% English, 15% Spanish)
- Medical Imaging:
  - Cancer center (80% malignant cases)
  - General hospital (95% benign cases)
- Fraud Detection:
  - Tourist area bank (high fraud rate 5%)
  - Residential area bank (low fraud 0.1%)

Horizontal FL

3. Quantity Skew: Highly imbalanced data sizes (some clients have 10x more data)

- Healthcare:
  - Mayo Clinic (500K patients)
  - Rural clinic (2K patients)
  - 250x difference
- Mobile Users:
  - Power users (10K text messages/month)
  - Casual users (50 messages/month)
  - 200x difference
- Banking:
  - JPMorgan (50M customers)
  - Local credit union (5K customers)
  - 10,000x difference

Horizontal FL

4. Temporal Skew: Data distribution changes over time differently per client

- Retail:
  - Ski resort (peak winter, dead summer)
  - Beach resort (peak summer, dead winter)
  - opposite seasonal patterns
- Hospitals:
  - Flu clinic (surge Nov-Feb)
  - Allergy clinic (surge Mar-May)
  - different temporal patterns
- Traffic:
  - Business district (peak weekdays 9am, 5pm)
  - Entertainment district (peak weekends, evenings)

Horizontal FL

- Slow convergence or divergence
- Poor generalization to global distribution
- Weight divergence across clients
- Unfair performance across subgroups

Horizontal FL

Data level solutions: modify the data to reduce heterogeneity

- **Data sharing:** (public dataset) Share a small public dataset across all clients to act as a "common ground"
  - Each client trains on their private data + shared public data (e.g., 10% of training)
  - Example: Medical imaging - use publicly available ImageNet or ChestX-ray dataset alongside private hospital data
  - Benefits: Helps align feature representations, acts as regularization
  - Drawback: Requires finding relevant public data (not always available)

- **Data augmentation:** Generate synthetic samples to balance local distribution
  - Apply transformations (rotation, cropping, noise) or use GANs to create diverse samples
  - Example: Client with only elderly patients generates synthetic data for younger age groups
  - Benefits: Increases diversity without privacy concerns
  - Drawback: Synthetic data may not perfectly represent real distribution

- **Resampling & Reweighting:** Oversample minority classes or reweight samples during training
  - If client has 90% class A, 10% class B → oversample class B or assign higher loss weight
  - Example: Fraud detection - bank with 0.1% fraud oversamples fraud cases 10x
  - Benefits: Simple to implement
  - Drawback: Can lead to overfitting on minority classes

Horizontal FL

Algorithm-Level Solutions: modify training algorithm to handle heterogeneity

FedProx (Federated Proximal)

- Use μ - proximal term to prevent local models from drifting too far from global model

  $$\text{Loss} = \text{Original\_Loss} + (\mu/2)||w - w\_global||^2$$

- Penalty term keeps local weights close to global weights, reducing divergence
- Example: Hospital with unique patient distribution won't overfit to local data, stays connected to global knowledge
- Benefits: Simple modification
- Parameter μ: Controls regularization strength (typical: 0.01-1.0)

Horizontal FL

Addressing Non-IID Challenges - Personalization Solutions

Instead of forcing one model for all - each client have personalized model

1. Multi-Task Learning / Split Architecture

- Share some layers globally, keep other layers client-specific
- Architecture: Base layers (shared) + Local layers (personalized)
- Learn common features globally, adapt final layers to local data
- Example: Medical imaging - shared: general anatomy features, local: disease patterns specific to demographics
- Benefits: Best of both worlds - leverage global knowledge + local adaptation

Horizontal FL

Addressing Non-IID Challenges - Personalization Solutions

Instead of forcing one model for all, let each client have personalized model

2. Local Fine-Tuning / Interpolation

- Use global model as starting point, fine-tune locally
- Full fine-tuning: Train global model, then each client fine-tunes on local data
- Example: Global model predicts with 85% accuracy → fine-tune on your data → 95% accuracy for you
- Benefits: Extremely simple, no algorithm changes needed
- Drawback: Requires sufficient local data for meaningful fine-tuning

Horizontal FL

Addressing Non-IID Challenges - Client Selection Solutions - select which clients participate

1. Clustered Federated Learning

- Group similar clients together and train separate models per cluster
- Identify clusters based on data distribution, train one model per cluster

- Example: Banking - Cluster 1: High-income clients, Cluster 2: Students, Cluster 3: Retirees
- Benefits: Each cluster gets specialized model, better than one-size-fits-all
- Process: Start with FedAvg → detect clusters from model updates → split into separate federations

Horizontal FL

Addressing Non-IID Challenges - Client Selection Solutions - select which clients participate

2. Importance Sampling / Biased Selection

- Select clients based on data importance, not uniformly at random
- Clients with diverse data are more valuable for training
- Compute importance score for each client, sample proportionally

- Example: Hospital with rare disease cases selected more often than hospital with common cases
- Benefits: Faster convergence, better coverage of data distribution
- Challenge: Requires estimating importance without seeing data

Horizontal FL

Addressing Non-IID Challenges

Data-Level

- Data sharing: Share small public dataset
- Data augmentation: Synthetic samples
- Resampling: Balance local distributions

Algorithm-Level

- FedProx: Add proximal term to loss

Personalization

- Multi-task learning: Client-specific layers
    - Fine-tuning: Local personalization

Client Selection

- Clustered FL: Group similar clients
- Importance sampling: Smart selection
- Active learning: Select informative clients

No single solution fits all cases - often need combination of techniques

Vertical FL

- Different Features, Same Sample
- Participants have different features for the same set of users/entities
- User overlap

Bank

Features: Credit Score, Income, Loan History

Users: Alice, Bob, Charlie, Diana

E-commerce

Features: Purchase Frequency, Cart Value, Product Category

Users: Alice, Bob, Charlie, Diana

VFL vs HFL

VFL enables cross-industry collaboration where organizations share users but have different data about them

Horizontal FL (HFL)

Data Partition
Same features, different samples

Example
Hospital A: Patients 1-100
Hospital B: Patients 101-200

Aggregation
Average model weights

Challenge
Non-IID data distributions

Vertical FL (VFL)

Data Partition
Different features, same samples

Example
Bank: Credit data for Users 1-100
E-commerce: Shopping data for Users 1-100

Aggregation
Combine feature embeddings

Challenge
User alignment

Vertical FL

- The Challenge: Finding Common Users Without Revealing Identities
- Before VFL training, participants must identify overlapping users without revealing who their users are

The Problem

Bank has: Users Alice, Bob, Charlie, Diana, Eve

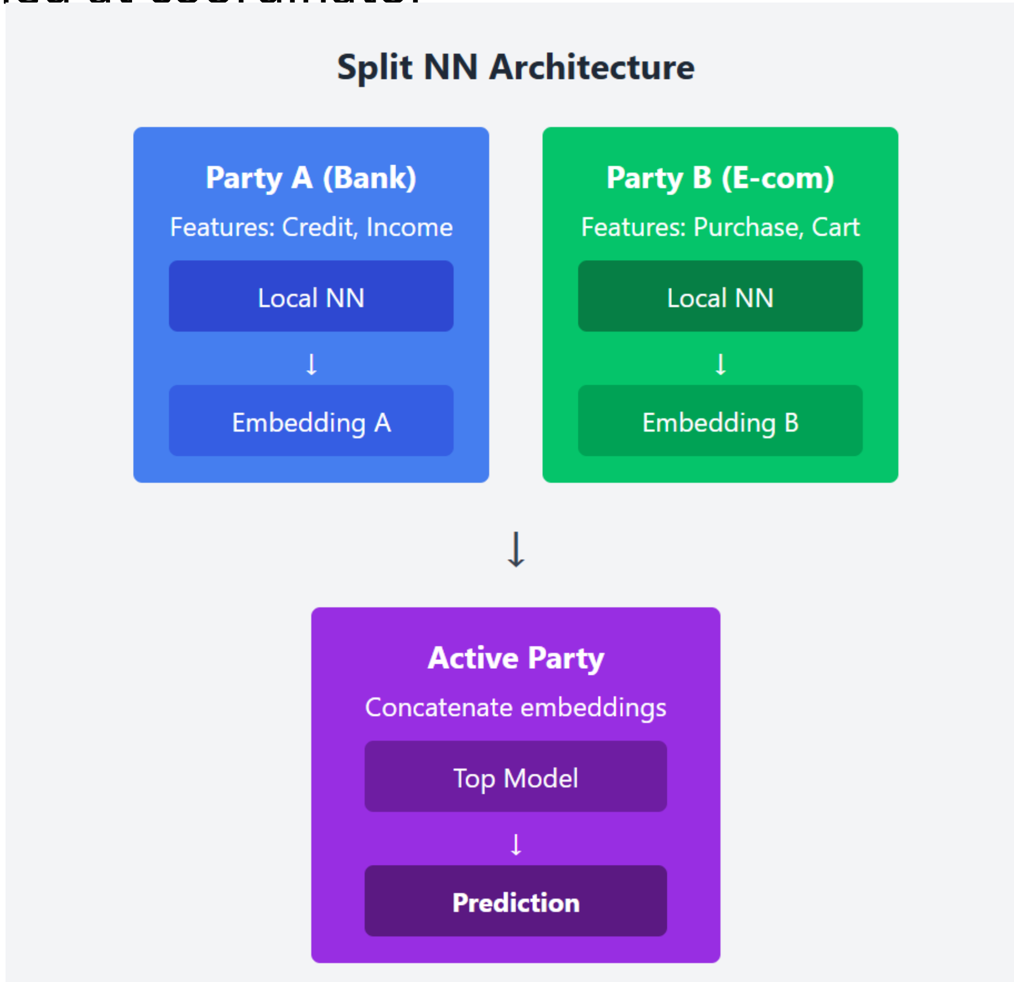
E-commerce has: Users Alice, Bob, Frank, George

Goal: Find intersection Alice, Bob without revealing other users

Privacy requirement: Bank should not learn Frank/George exist

Vertical FL

- Split Neural Networks
- Architecture: Split Model Across Participants
- Each party trains a local model on their features, embeddings combined at coordinator



Use Case: Bank + E-commerce

Credit Risk Assessment

Bank improves

Bank (Active)

Features
Credit score, Income, Loans, Payment history

Labels
Loan default (Yes/No)

Users
500K customers

E-commerce (Passive)

Features
Spending, Frequency, Categories, Returns

Labels
No labels (passive)

Users
800K shoppers (300K overlap)

Business Case

- 70% accuracy with financial data alone
- Shopping behavior correlates with responsibility
- VFL enables collaboration without data sharing

Result: Accuracy improved from 70% to 85%

VFL unlocks cross-organization collaboration for mutual benefit

Federated Transfer Learning FL

- Different Features AND Different Samples
- Participants have BOTH different features AND different samples - minimal or no overlap

Hospital A (Radiology)

Features: X-ray images, CT scans

Patients: 1, 2, 3, 4, 5

Hospital B (Pathology)

Features: Tissue slides, Lab results

Patients: 6, 7, 8, 9, 10

- Transfer knowledge across domains through shared representations, even when data is completely different

FTL vs HFL vs VFL

HFL

Features
✓ Same

Samples
✗ Different

Example
Same disease, different hospitals

VFL

Features
✗ Different

Samples
✓ Same

Example
Same users, different companies

FTL

Features
✗ Different

Samples
✗ Different

Example
Different domains, transfer knowledge

- Learn shared representations that work across domains
- Pre-train on one domain, fine-tune on another (federated style)

Transfer Learning Fundamentals

Leverage knowledge learned from one task/domain to improve learning in another task/domain

Traditional Transfer Learning

Step 1 - Pre-training: Train model on large source dataset (e.g., ImageNet with 1M images)

Step 2 - Fine-tuning: Adapt model to target task with small dataset (e.g., medical images with 1K samples)

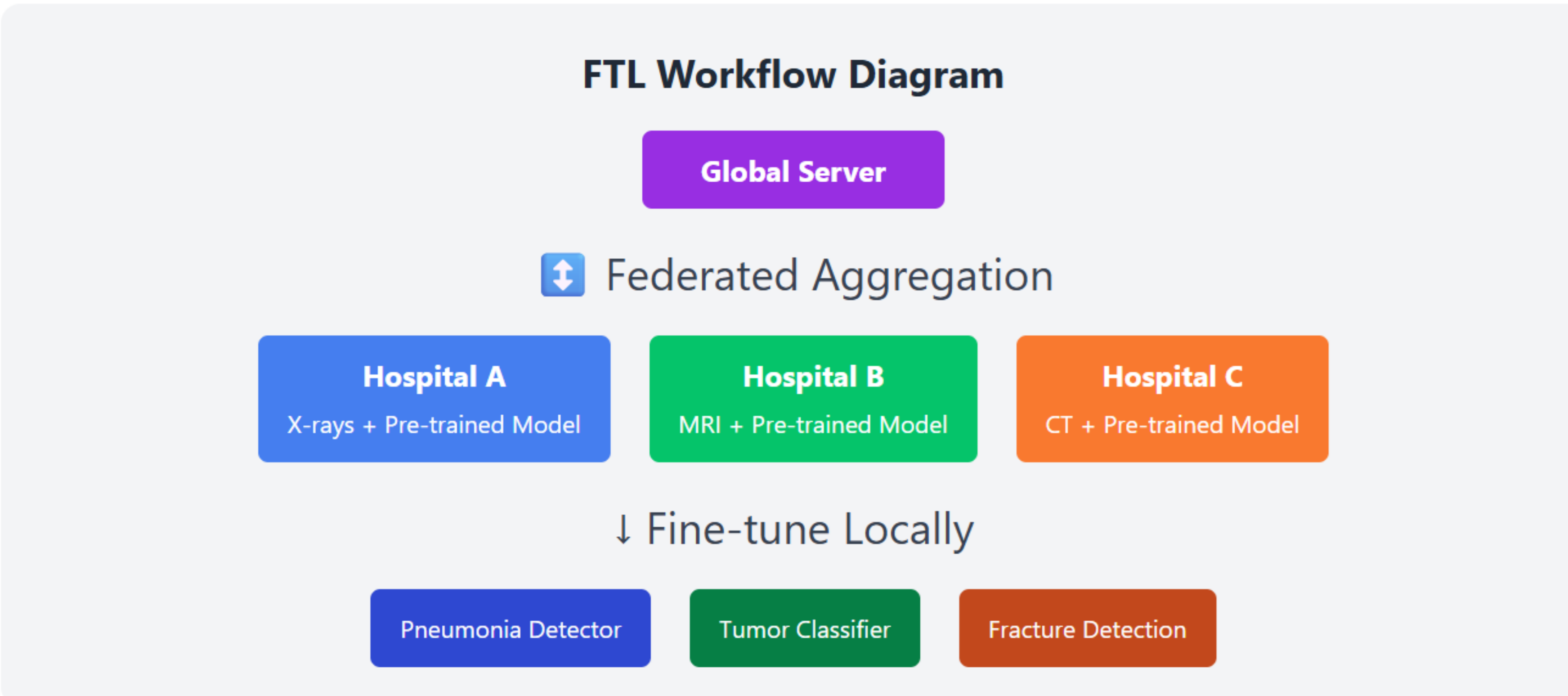
Key insight: Low-level features (edges, textures) transfer across domains!

Why Transfer Learning Works

Hierarchical representations: Neural networks learn progressively abstract features

- **Layer 1:** Edges, colors (universal)
- **Layer 2:** Textures, patterns (mostly universal)
- **Layer 3:** Parts, shapes (somewhat specific)
- **Layer 4:** Objects, concepts (task-specific)

Federated Transfer Architecture



Federated Transfer Architecture

Phase 1: Federated Pre-training

Goal: Learn general-purpose representations across all participants

Process:

1. Initialize global model with random weights
2. Each participant trains on their local data (even if different domains)
3. Aggregate models to create universal feature extractor
4. Repeat until convergence

Output: Pre-trained base model that captures cross-domain patterns

Phase 2: Local Fine-tuning

Goal: Adapt pre-trained model to each participant's specific task

Process:

1. Download pre-trained global model
2. Add task-specific layers (classification head, regression layer, etc.)
3. Fine-tune locally on participant's data
4. Deploy personalized model

Output: Specialized model for each participant's domain/task

Cross-Domain Knowledge Transfer

Transferring Knowledge Across Different Domains

Shared Representation Learning

Core idea: Find common feature space that works for all domains

- Hospital A (X-rays) learns edge detection
- Hospital B (MRI) learns texture patterns
- Hospital C (CT scans) learns shape features

→ Combined: Universal medical image features!

The global model learns to extract domain-invariant features

FTL Challenges

1. Negative Transfer: Source knowledge hurts target performance

Solution: Selective transfer

2. Catastrophic Forgetting: Fine-tuning erases pre-trained knowledge

Solution: Regularization

3. Domain Shift: Large distribution gap between domains

Solution: Domain adaptation

4. Communication Efficiency: Large models expensive to transmit

Solution: Model compression