DS 246

Federated Learning

Topics

- Role of Scalability in DL's Evolution
- Distributed training with TensorFlow
- Distributed training with PyTorch
- Challenges and Opportunities
- Summary

“Federated Learning vs. Distributed ML

A Paradigm Shift

The Challenge:

Scaling ML with Traditional Distributed Approaches

Traditional Distributed ML

- Distributes model training across many compute nodes, typically in controlled, centralized environments.

How it Works (Key Pillars)

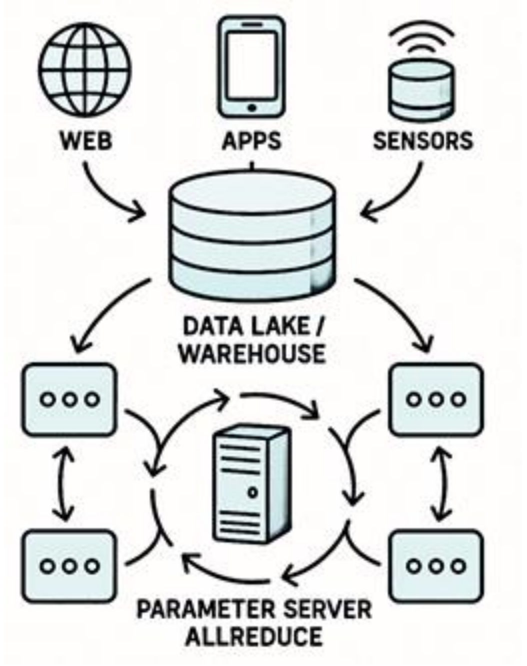
- **Central Hub:** Relies on high-performance compute clusters in data centers.
- **Data Consolidation:** All training data is brought to a central location, then sharded.
- **Rapid Sync:** Frequent, high-speed parameter updates (microseconds/milliseconds).
- **Goal:** Maximize computing speed and hardware efficiency.

The Challenge:

Scaling ML with Traditional Distributed Approaches

Industry Snapshot

- NVIDIA's Selene: Powering massive GPT models.
- Google's TPU Pods: Training giants like BERT & T5.
- Meta's RSC: Developing foundational AI models.



The Shift:

Introducing Federated Learning

Federated Learning: Training Models Where Data Lives

- Trains algorithms across decentralized devices holding local data, without data exchange.

How it Works (Key Pillars)

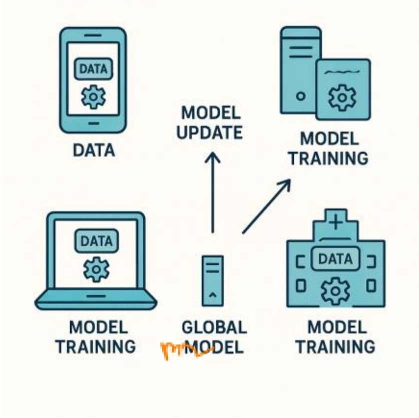
- Decentralized Network: Uses a wide array of devices (phones, IoT, institutions).
- Data Stays: Training data never leaves its original source.
- Efficient Updates: Periodic, asynchronous model updates (not raw data).
- Goal: Prioritize communication efficiency and data privacy.

The Shift:

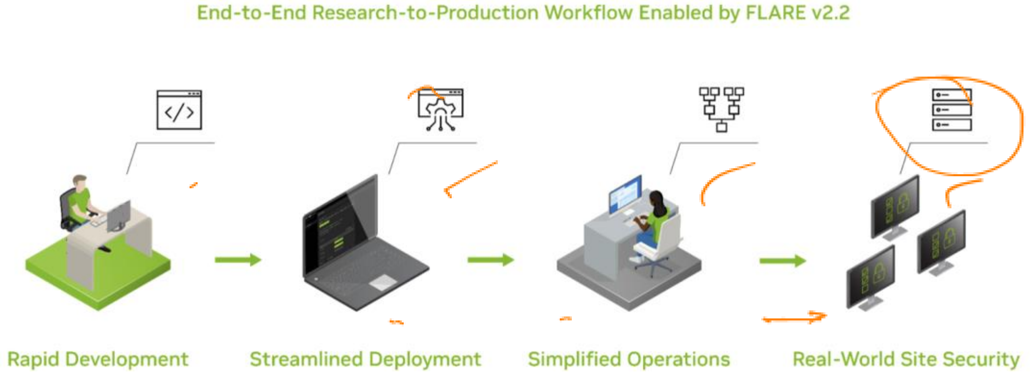
Introducing Federated Learning

Industry Snapshot

- Google Gboard: Better predictions, private typing.
- Apple's Siri: On-device learning for improved voice assistance.
- NVIDIA FLARE: Secure collaboration for healthcare research.



$#@?!?!
COPY
Mobile-friendly - Address: 738 E. 3rd St. Los Angeles, CA 90013. Phone: 323-2...



Comprehensive Comparison

Feature	Traditional Distributed ML	Federated Learning
Data Location	Centralized (Data Lake/Warehouse)	Decentralized (Remains at Source/Edge)
Primary Goal	Computational Scalability & Speed	Privacy, Data Sovereignty, Collaboration
Communication	High Bandwidth, Low Latency (Internal)	Limited Bandwidth, Higher Latency (External)
Data Characteristics	Often IID, Balanced, Controlled	Naturally Non-IID, Unbalanced, Biased
Hardware	Homogeneous, Powerful Data Centers	Heterogeneous, Resource-Constrained Devices
Scale of Participants	10s - 1000s of Nodes	Potentially Millions of Devices/Organizations
Participant Availability	Consistently Available	Intermittent, Availability-Dependent
Trust Model	Single Administrative Domain (High Trust)	Multi-Stakeholder, Zero-Trust Assumptions
Regulatory Context	Internal Data Governance Focus	External Compliance (GDPR, HIPAA, etc.) is Key

Why Federated Learning?

The Driving Forces

Core Drivers:

Regulatory Imperatives (Keeping Data Safe & Legal):

- GDPR: Upholding data minimization, purpose limitation.
- HIPAA: Protecting sensitive health information.
- Sector-Specific Rules: Finance (GLBA), Consumer (CCPA).

Technical & Efficiency Gains:

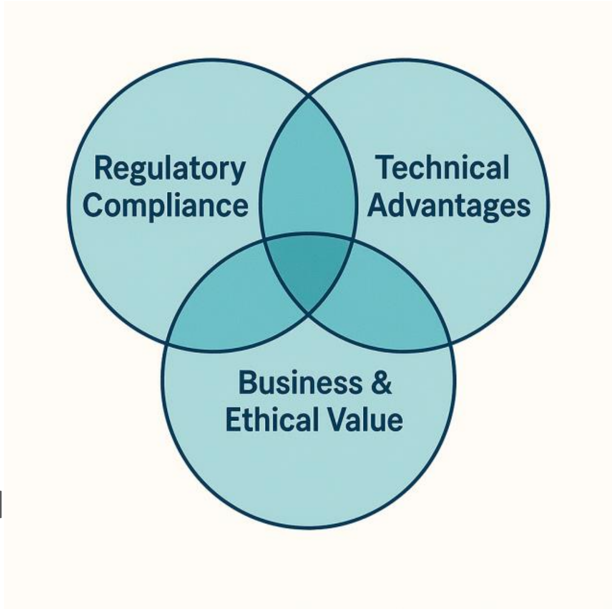
- Bandwidth Savings: Sending small model updates (MBs) vs. vast raw data (GBs/TBs).
- Real-Time Personalization: Models adapt locally to individual user patterns.
- Unlocking Siloed Data: Enables learning from previously inaccessible private datasets.

Why Federated Learning?

The Driving Forces

Business & Ethical

- Building User Trust: Privacy-first features as a competitive differentiator.
- Respecting Data Sovereignty: Enabling collaboration across organizational boundaries.
- Enabling Consortia: Joint research without direct data sharing (e.g., pharma, finance).
- Reducing Data Risk: Minimizing storage of sensitive, potentially liabl information.



The Journey & Horizon

FL's Evolution

Timeline Visual (Key Milestones):

- Pre-2016: Distributed ML reigns for compute efficiency.
- 2016: Google's seminal paper introduces "Communication-Efficient Learning of Deep Networks from Decentralized Data" – Birth of FL.
- 2017-2020: Early wins in mobile (e.g., keyboard prediction, OS features).
- 2021-Present: Expansion to cross-organization use cases, research in robustness, fairness, and standardization efforts (e.g., NVIDIA FLARE).

The Journey & Horizon

FL's Evolution

What's Next? Emerging Frontiers:

- **Federated Fine-Tuning:** Adapting large foundation models with decentralized private data.
- **Hybrid Privacy**: Combining FL with Differential Privacy, Homomorphic Encryption, Secure Multi-Party Computation.
- **Hardware Acceleration**: Chips optimized for efficient on-device training.
- **Standardization**: Developing common protocols for cross-device and cross-silo FL.

Federated Learning Taxonomies

Data Partitioning Taxonomy

&

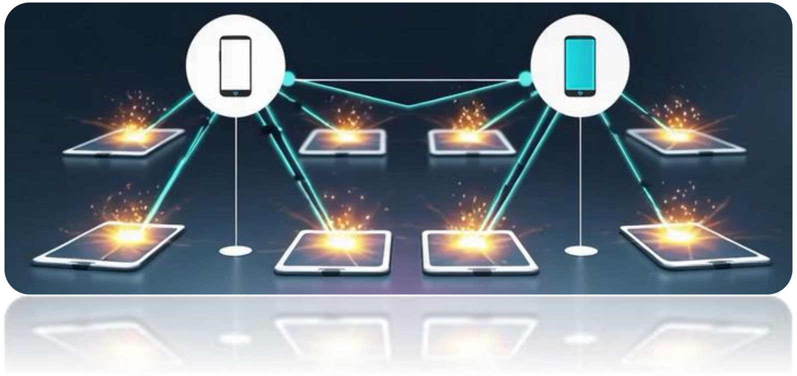
Deployment Architecture Taxonomy

Data Partitioning Taxonomy: Horizontal FL

Focuses on how the data is distributed among participants:

Participants have the same features but different samples (different users/entities)

- Example: Multiple hospitals with patient data having the same medical measurements but for completely different patients
- The data matrices would align along the feature dimension (columns)



Data Partitioning Taxonomy: Vertical FL

Focuses on how the data is distributed among participants:

Participants have the same samples but different features

- Example: A bank and a retailer having some overlapping customers but collecting different information about them (financial data vs. shopping behavior)
- The data matrices would align along the sample dimension (rows)

Data Partitioning Taxonomy: Hybrid FL

Focuses on how the data is distributed among participants:

Some overlap in both samples and features

- Mixed scenarios where organizations have partially overlapping datasets in both dimensions

Data Partitioning taxonomy is primarily concerned with the statistical properties of the data distribution and determines the type of aggregation algorithms needed.

“Deployment Architecture Taxonomy

FL Architectures

Key Architectures :

- Cross-Device FL
- Cross-Silo FL
- Hierarchical FL

FL Architecture: Cross-Device Federated Learning

Training ML models across millions of edge devices (e.g., smartphones, wearables) without centralizing data

Key Features :

- Massive scale, Potentially millions of participating devices, etc

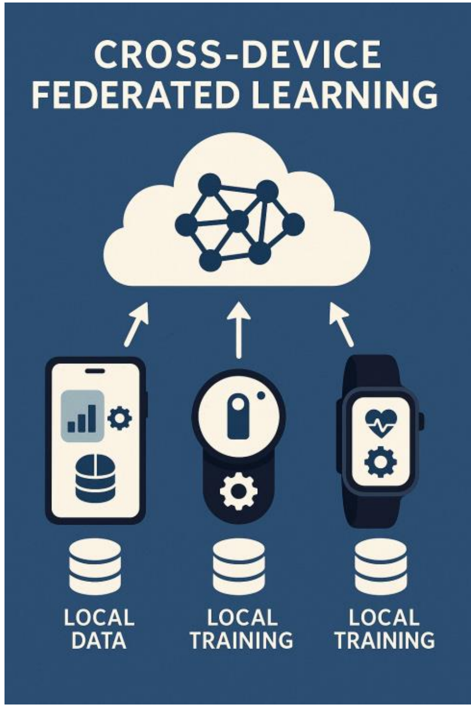
Challenges :

- High churn and drop-off, Device heterogeneity, Extreme communication efficiency required, Personal noisy data, Intermittent bandwidth-limited connections, Limited computation capacity, Unreliable connectivity, Battery constraints, Small local datasets, etc.

Frameworks :

- TensorFlow Federated, FedML

https://www.tensorflow.org/federated/tutorials/tutorials_overview



FL Architecture: Cross-Silo Federated Learning

Training ML models across a small number of reliable institutions with sensitive and structured datasets

Key Features :

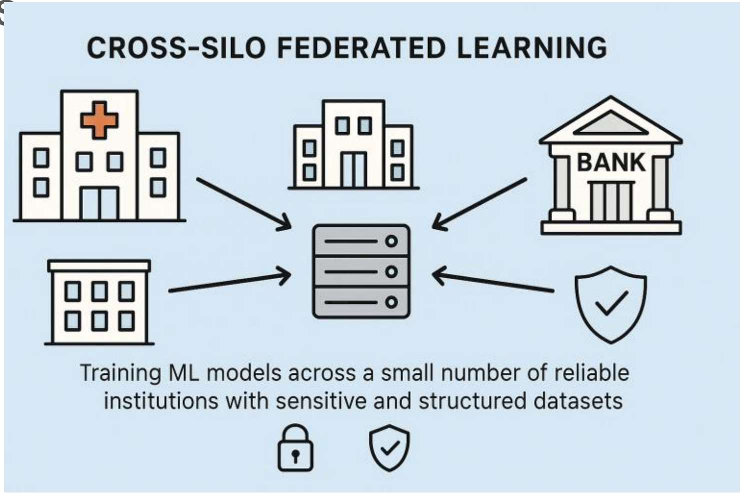
- Reliable connectivity, Higher compute and storage, Strong data governance, etc

Challenges :

- Compliance and standardization, Inter-institutional trust, Data schema alignment, etc.

Use cases :

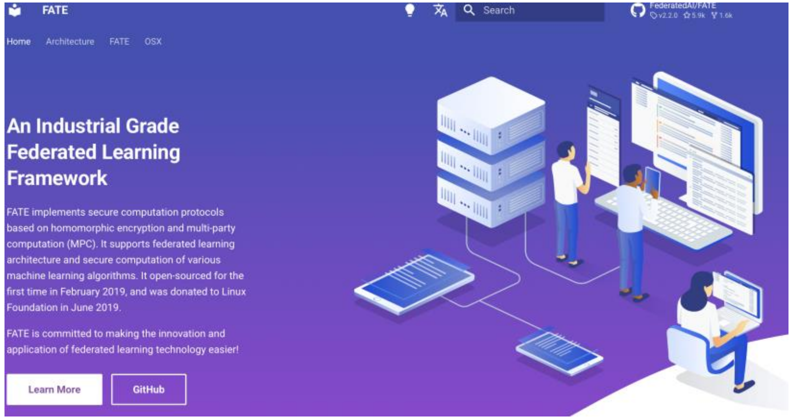
- Medical research across hospitals, Joint fraud detection in banking, Industrial IoT analytics, etc.


CROSS-SILO FEDERATED LEARNING
BANK
Training ML models across a small number of reliable institutions with sensitive and structured datasets

FL Architecture: Cross-Silo Federated Learning

Frameworks :

- FATE, PySyft


FATE

Search

FederatedAI/FATE v2.2.0 5.9k 1.6k

Home Architecture FATE OSX

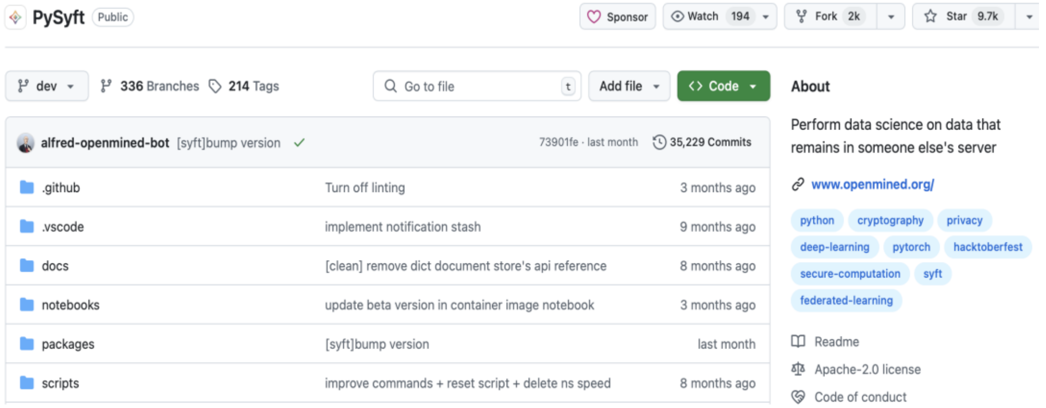
An Industrial Grade Federated Learning Framework

FATE implements secure computation protocols based on homomorphic encryption and multi-party computation (MPC). It supports federated learning architecture and secure computation of various machine learning algorithms. It open-sourced for the first time in February 2019, and was donated to Linux Foundation in June 2019.

FATE is committed to making the innovation and application of federated learning technology easier!

Learn More GitHub

https://fate.readthedocs.io/en/latest/


PySyft Public

Sponsor | Watch 194 | Fork 2k | Star 9.7k

dev | 336 Branches | 214 Tags | Go to file | Add file | Code

alfred-openmined-bot [syft]bump version ✓ 73901fe · last month 35,229 Commits

.github	Turn off linting	3 months ago
.vscode	implement notification stash	9 months ago
docs	[clean] remove dict document store's api reference	8 months ago
notebooks	update beta version in container image notebook	3 months ago
packages	[syft]bump version	last month
scripts	improve commands + reset script + delete ns speed	8 months ago

About

Perform data science on data that remains in someone else's server

www.openmined.org/

python cryptography privacy deep-learning pytorch hacktoberfest secure-computation syft federated-learning

Readme

Apache-2.0 license

Code of conduct

https://github.com/OpenMined/PySyft

FL Architecture: Hierarchical Federated Learning

FL in multi-tiered structures, where intermediate aggregators reduce communication loads and synchronize updates regionally before sending to the central server.

Key Features :

- Reduces central server bottlenecks
- Balances scalability and communication efficiency
- Useful in edge-cloud continuum scenarios

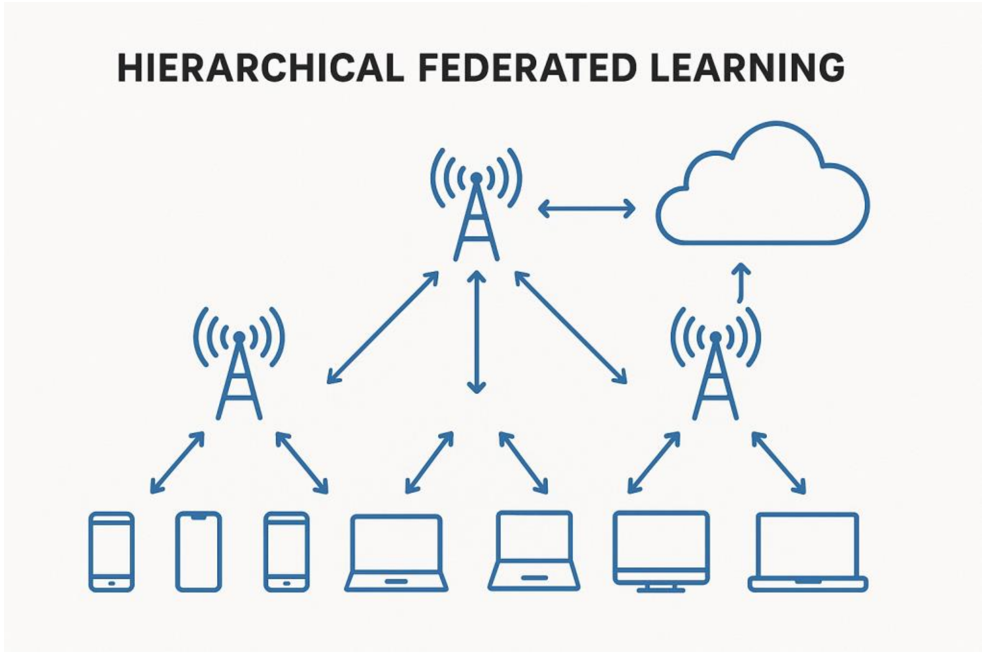
Challenges :

- Coordination across tiers
- Potential single point of failure in intermediates
- Latency in multi-hop aggregation

FL Architecture: Hierarchical Federated Learning

Use cases :

- Telecom edge nodes aggregating from devices
- Federated analytics in smart city infrastructure
- Content recommendation across distributed CDNs



FL Architecture: Comparison

Feature	Cross-Device FL	Cross-Silo FL	Hierarchical FL
Participants	Millions of devices	Dozens of institutions	Nested device clusters
Data Type	Personal, user data	Structured, sensitive	Mixed (edge + enterprise)
Connectivity	Intermittent	Stable	Tiered (local → global)
Compute Power	Low	High	Varies across layers
Aggregation Level	Global	Global	Local + global
Use Cases	Gboard, personalization	Hospitals, banks	Smart cities, telecom, CDN

FL Architecture: Key Takeaways

- **Cross-Device FL** is best for mass personalization.
- **Cross-Silo FL** supports institutional collaboration with privacy compliance.
- **Hierarchical FL** offers a scalable middle-ground, ideal for edge-to-cloud orchestration.

“Technical Workflow of Federated Learning

Technical Workflow of Federated Learning

System Architecture

Client-Server Architecture:

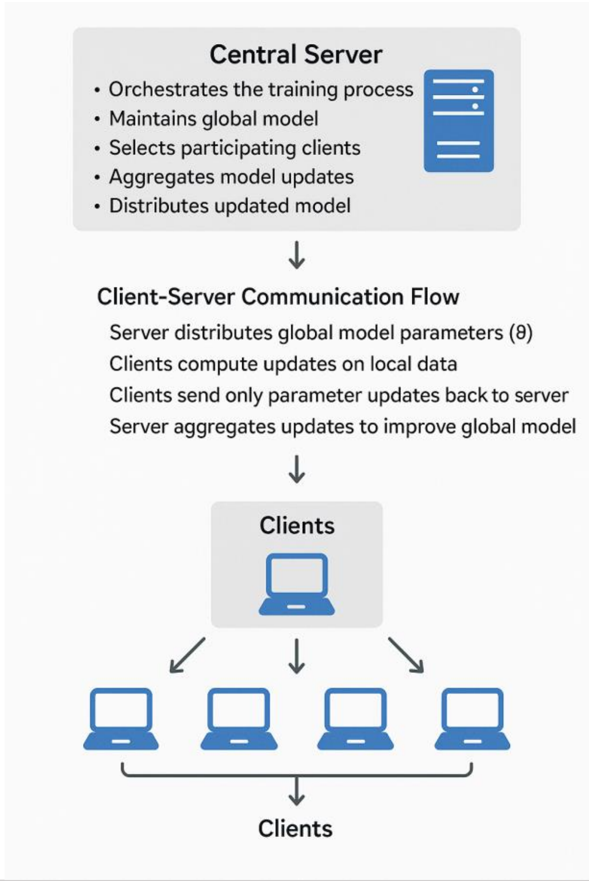
- **Central server:**
  - Orchestrates the training process
  - Maintains global model
  - Selects participating clients
  - Aggregates model updates
  - Distributes updated model
- **Clients**:
  - Individual data sources
  - Store private data locally
  - Perform local model training
  - Send model updates (not data) to server

Technical Workflow of Federated Learning

Client-Server Communication Flow

Key Steps:

- Server distributes global model parameters (θ)
- Clients compute updates on local data
- Clients send only parameter updates back to server
- Server aggregates updates to improve global model


Central Server
• Orchestrates the training process
• Maintains global model
• Selects participating clients
• Aggregates model updates
• Distributes updated model
Client-Server Communication Flow
Server distributes global model parameters (θ)
Clients compute updates on local data
Clients send only parameter updates back to server
Server aggregates updates to improve global model
Clients
Clients

Technical Workflow of Federated Learning

Client-Server Communication Flow

Advantages:

- Simple to implement
- Centralized monitoring and control
- Easier to maintain model consistency

Challenges:

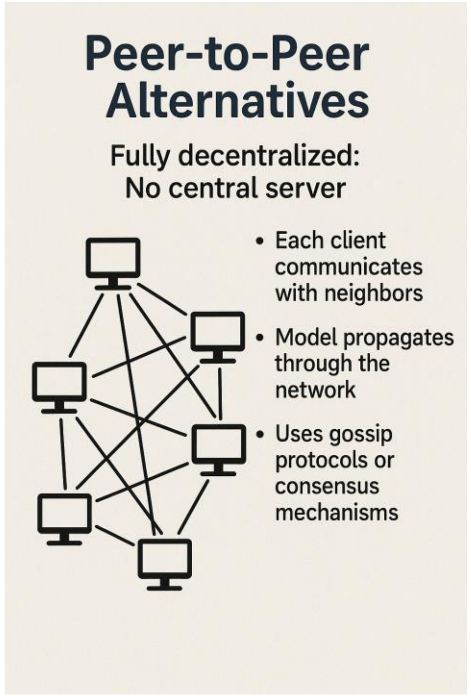
- Single point of failure
- Potential bottleneck at server
- Trust requirements for central entity

Technical Workflow of Federated Learning

System Architecture

Peer-to-Peer Architecture:

- **Fully decentralized:**
  - Each client communicates with neighbours
  - Model propagates through the network
  - Uses gossip protocols or consensus mechanisms


Peer-to-Peer Alternatives
Fully decentralized: No central server
Each client communicates with neighbors
Model propagates through the network
Uses gossip protocols or consensus mechanisms

Technical Workflow of Federated Learning

Peer-to-Peer Architecture

Advantages:

- No central point of failure
- More democratic structure
- Potentially better privacy

Challenges:

- Complex consensus mechanisms
- Higher communication overhead
- Harder to monitor and debug

Technical Workflow of Federated Learning

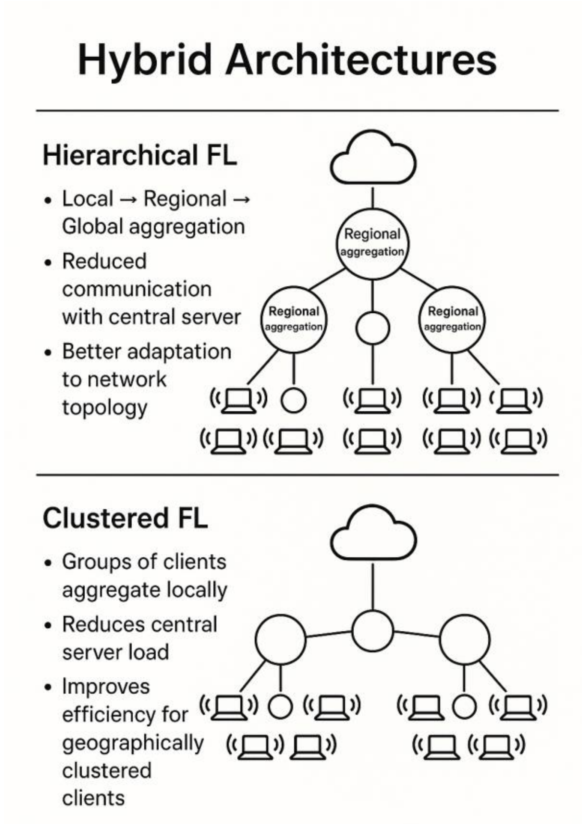
Hybrid Architectures

Hierarchical FL: Multi-level aggregation

- Local → Regional → Global aggregation
- Reduced communication with central server
- Better adaptation to network topology

Clustered FL: Groups of clients aggregate locally

- Reduces central server load
- Improves efficiency for geographically clustered clients



Hybrid Architectures

Hierarchical FL

- Local → Regional → Global aggregation
- Reduced communication with central server
- Better adaptation to network topology

Clustered FL

- Groups of clients aggregate locally
- Reduces central server load
- Improves efficiency for geographically clustered clients

“FL Process Flow

FL Process Flow

Step 1: Client Selection

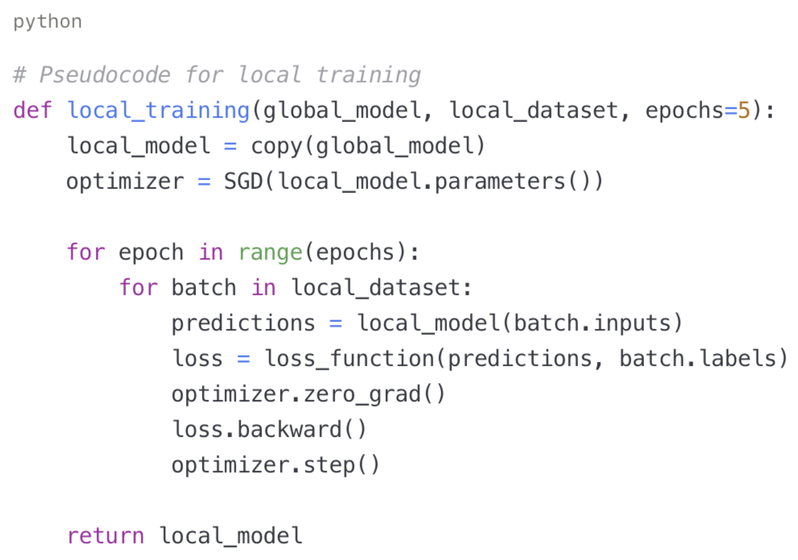
- **Challenge**:
  - Not all clients available/suitable for each round
- **Selection criteria:**
  - Device status (charging, idle, connectivity)
  - Resource availability (memory, CPU, battery)
  - Data quality and quantity
  - Client history (reliability, contribution)
- **Selection strategies:**
  - Random sampling
  - Importance sampling (data quality-weighted)
  - Round-robin
  - Performance-based
- **Practical consideration:**
  - Must handle clients dropping out

FL Process Flow

Step 2: Local Training

Process on each selected client:

- Receive global model parameters θ(t) from server
- Load local private dataset D
- Compute local objective function F(θ)
- Run mini-batch SGD for E local epochs
- Produce updated local model parameters θ



python

```python
# Pseudocode for local training
def local_training(global_model, local_dataset, epochs=5):
    local_model = copy(global_model)
    optimizer = SGD(local_model.parameters())

    for epoch in range(epochs):
        for batch in local_dataset:
            predictions = local_model(batch.inputs)
            loss = loss_function(predictions, batch.labels)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

    return local_model
```

FL Process Flow

Step 3: Secure Aggregation

Challenge: Server should not see individual updates:

Goals:

- Aggregate model updates from multiple clients
- Preserve privacy of individual updates
- Ensure integrity of aggregation pro

Methods:

- Secure Multi-Party Computation (MPC)
- Homomorphic Encryption
- Secret Sharing
- Differential Privacy

Simplified secure aggregation:

1. Each client adds random noise: θ_i + noise
2. Noise cancels out during aggregation
3. Server learns only the sum, not individual values

(often involving pairs of devices sharing secret random numbers that add up to zero globally).

FL Process Flow

Step 4: Global Update

- Server computes weighted average of client models
- Updates global model parameters
- Evaluates convergence and model quality
- Prepares for next round of training

python

```python
# Pseudocode for global update
def global_update(client_models, client_sizes):
    total_size = sum(client_sizes)
    global_model = ModelArchitecture()

    for param_name, param in global_model.named_parameters():
        weighted_sum = 0
        for client_idx, model in enumerate(client_models):
            weight = client_sizes[client_idx] / total_size
            weighted_sum += weight * model.state_dict()[param_name]

        global_model.state_dict()[param_name].copy_(weighted_sum)

    return global_model
```

“FedAvg Algorithm

FedAvg Algorithm

- Most widely used FL aggregation algorithm
- Weighted average of locally trained models
- Simple but surprisingly effective approach

FedAvg Algorithm

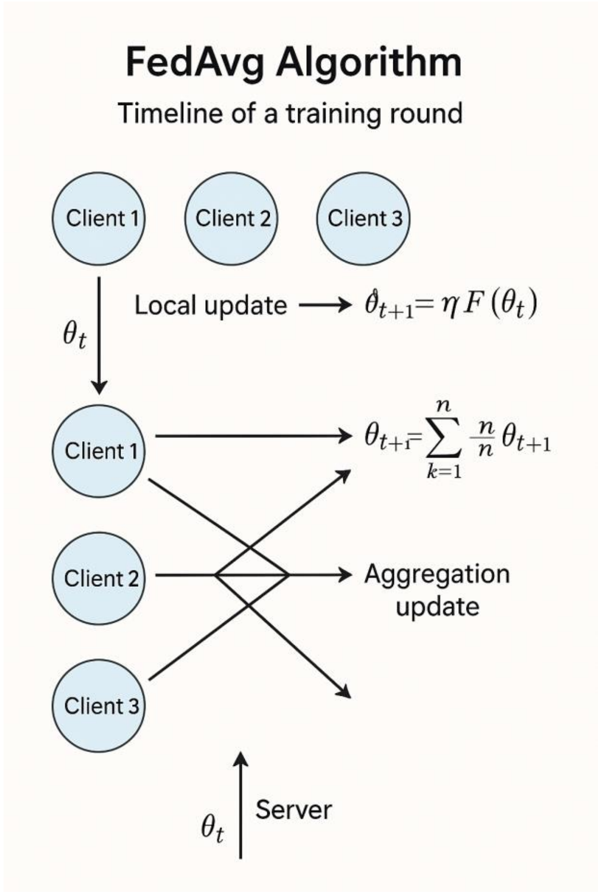
Mathematical Intuition

- Each client updates its local model via:

$$\theta_{t+1}^{(k)} = \theta_t - \eta \nabla F_k(\theta_t)$$

- Global aggregation (weighted by data size):

$$\theta_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \theta_{t+1}^{(k)}$$


FedAvg Algorithm
Timeline of a training round
Client 1
Client 2
Client 3
θ_t
Local update
$\theta_{t+1} = \eta F(\theta_t)$
$\theta_{t+1} = \sum_{k=1}^{n} \frac{n}{n} \theta_{t+1}$
Aggregation update
Server
θ_t

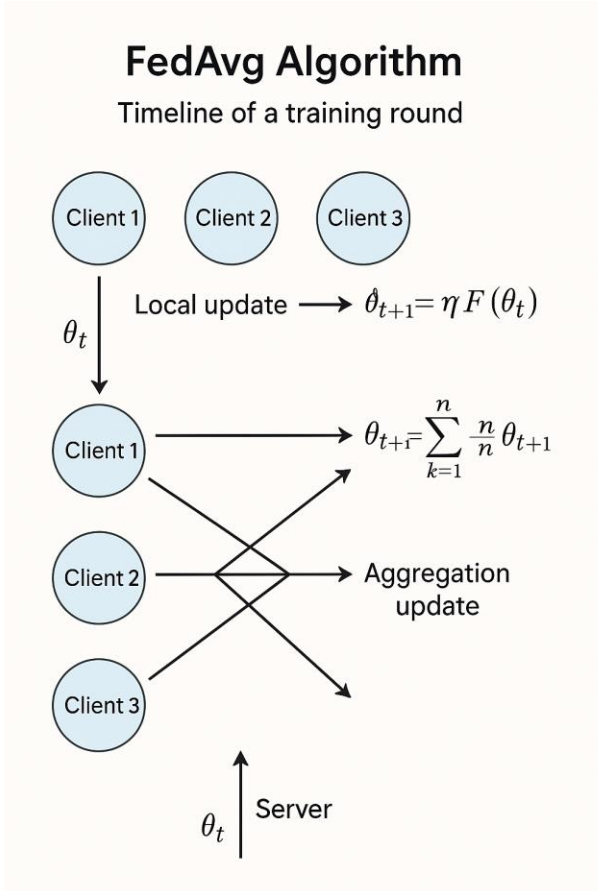
FedAvg Algorithm

FedAvg Hyperparameters

- **C:** Fraction of clients selected per round
- **B:** Local batch size
- **E:** Number of local epochs
- **η:** Learning rate

Impact of parameters:

- **Higher E:** More local computation, fewer communication rounds
- **Lower C:** Less communication per round, potentially slower convergence
- **Tradeoff:** Communication efficiency vs. convergence speed



FL Process Flow: FedAvg Variants

Why Variants?

- FedAvg fails to converge well under:
  - Non-IID (Independent and Identically Distributed) data
  - Unbalanced participation
  - Unequal local computation

In Machine Learning: When we say training data is IID, it implies that each data point used to train the model is a fair, unbiased sample from the true underlying data distribution that we want the model to learn. This is a common and often simplifying assumption in classical machine learning.

FL Process Flow: FedAvg Variants

FedProx – Proximal Regularization

- **Problem it solves:** Statistical heterogeneity (divergent local updates)
- **Solution:** Adds penalty to keep local updates close to global model
- **Objective:**

$$\theta_{t+1}^{(k)} = \arg\min_{\theta} \left[F_k(\theta) + \frac{\mu}{2} \|\theta - \theta_t\|^2 \right]$$

- **Interpretation:** "Don’t drift too far from the server”

FL Process Flow: FedAvg Variants

SCAFFOLD – Control Variate Correction

- **Problem it solves:** Client drift caused by non-IID local optima
- **Solution:** Each client and server keep control variates to correct gradient directions
- **Key Idea:**

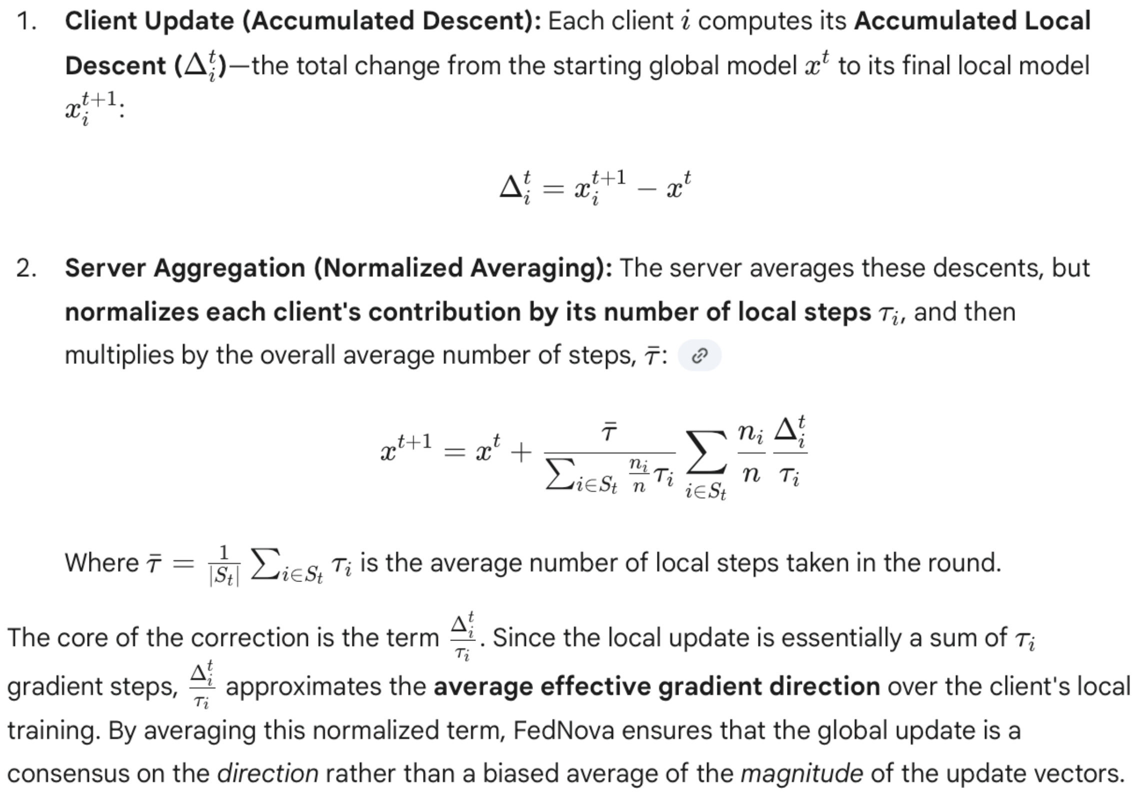
$$\Delta_k = \nabla F_k(\theta_t) - c_k + c$$

  where c_k is client's control variate and c is server's average control variate
- **Interpretation:** "Don't drift too far from the server"

FL Process Flow: FedAvg Variants

FedNova – Normalized Averaging

- **Problem it solves:** Clients perform different numbers of steps/epochs
- **Solution:** Normalize updates based on the total amount of work



1. **Client Update (Accumulated Descent):** Each client i computes its **Accumulated Local Descent (Δ_i^t)**—the total change from the starting global model x^t to its final local model x_i^{t+1}:

$$\Delta_i^t = x_i^{t+1} - x^t$$

2. **Server Aggregation (Normalized Averaging):** The server averages these descents, but **normalizes each client's contribution by its number of local steps τ_i**, and then multiplies by the overall average number of steps, $\bar{\tau}$:

$$x^{t+1} = x^t + \frac{\bar{\tau}}{\sum_{i \in S_t} \frac{n_i}{n} \tau_i} \sum_{i \in S_t} \frac{n_i}{n} \frac{\Delta_i^t}{\tau_i}$$

Where $\bar{\tau} = \frac{1}{|S_t|} \sum_{i \in S_t} \tau_i$ is the average number of local steps taken in the round.

The core of the correction is the term $\frac{\Delta_i^t}{\tau_i}$. Since the local update is essentially a sum of τ_i gradient steps, $\frac{\Delta_i^t}{\tau_i}$ approximates the **average effective gradient direction** over the client's local training. By averaging this normalized term, FedNova ensures that the global update is a consensus on the *direction* rather than a biased average of the *magnitude* of the update vectors.

FL Process Flow

Key Takeaways

- Federated learning follows a cyclical process of client selection, local training, secure aggregation, and global update
- System architecture choices (client-server vs. peer-to-peer) impact privacy, reliability, and efficiency
- FedAvg provides a simple but effective foundation, with variants addressing specific challenges
- Convergence is complicated by statistical heterogeneity, system heterogeneity, and communication constraints
- Real-world deployments require careful tuning of hyperparameters to balance communication efficiency and model quality

“Federated Learning: Challenges & Solutions

Federated Learning: Challenges & Solutions

Overview of FL Challenges

- Statistical Challenges
- Communication Bottlenecks
- Robust Aggregation
- Privacy Concerns

Federated Learning: **Statistical Challenges**

The Non-IID Problem

- **IID assumption:** Independent and Identically Distributed data
  - Foundation of most ML convergence proofs
  - Rarely true in federated settings
- **Sources of Non-IID data in FL:**
  - **Label skew:** Different label distributions across clients
  - **Feature skew:** Different feature distributions for same labels
  - **Quantity skew:** Varying amounts of data per client
  - **Temporal skew:** Data collected at different times

Federated Learning: **Statistical Challenges**

Non-IID Solutions: Data Approaches

- **FedProx:**
  - Adds proximal term to client objective
  - Limits drift from global model Local objective: $F_k(\theta) + (\mu/2)||\theta - \theta^t||^2$
- **SCAFFOLD**:
  - Uses control variates to correct drift
  - Tracks variance between local and global updates
  - Faster convergence for non-IID data
- **FedNova:**
  - Normalizes client updates by local steps
  - Reduces bias from varying local work
  - Especially effective for heterogeneous clients

Federated Learning: **Communication Efficiency**

The Communication Bottleneck

- **Why it matters:**
  - Communication = largest time cost in FL
  - Limited bandwidth in cross-device settings
  - Energy consumption on mobile devices
- **Scale of the problem:**
  - Modern CNN/Transformer: 100MB-1GB model size
  - Thousands to millions of clients

- **Key metrics:**
  - Total communication cost
  - Per-round communication cost
  - Client-side upload vs. downloa

Federated Learning: **Communication Efficiency**

Strategies

- **Full-precision → reduced precision:**
  - 32-bit float → 16/8/4/2/1-bit representation
  - Reduces communication by 2-32x
- **Client Sampling Insight: Not all clients need to participate in every round**
  - Uniform sampling: Random subset of clients
  - Importance sampling: Based on data distribution
  - Diversity sampling: Maximize client diversity
  - Power-of-choice: Sample multiple subsets, pick best

Federated Learning: **Communication Efficiency**
Strategies

- **Hierarchical Communication Insight: Leverage network topology for efficiency**
    - Tree-structured aggregation: Tiered aggregation
    - Peer-to-peer exchanging: Clients share updates locally
    - Gossip learning: Information propagation among neighbors

Federated Learning: **Robust Aggregation**

The Need for Robust Aggregation

- **Challenges:**
  - Unreliable clients (stragglers, dropouts)
  - Noisy updates (poor data quality)
  - Byzantine attacks (adversarial clients)
- **Goals of robust aggregation:**
  - Resilience to faulty/malicious clients
  - Stability despite heterogeneity
  - Consistent convergence properties

Federated Learning: **Robust Aggregation**

The Need for Robust Aggregation

- **Threat models:**
    - Label-flipping attacks
    - Gradient poisoning
    - Model replacement

Federated Learning: **Robust Aggregation**

Strategies

- **FedProx Motivation:**
  - Reduce impact of statistical heterogeneity
- **Trimmed Mean Aggregation Approach:**
  - Sort parameter values across clients
  - Remove β largest and β smallest values
  - Average the remaining values
- Other variants: Median-Based Aggregation, Krum Aggregation
- Advanced Robust Aggregation: Bulyan, RFA (Robust Federated Aggregation), AFA (Adaptive Federated Averaging)

Federated Learning: **Privacy Mechanisms**

Privacy Threats in FL

- **Inference attacks:**
    - Membership inference: Was sample in training data?
    - Property inference: Learn statistical properties
    - Model inversion: Reconstruct training examples
- Attack vectors:
    - Observing model updates
    - Analyzing global model behavior
    - Participating as malicious client
- Real-world implications:
    - Health data exposure, Financial information leakage, User behavior revelation etc

Federated Learning: **Privacy Mechanisms**

Solutions

- **Differential Privacy (DP):**
  - Algorithm output shouldn't reveal if a specific sample was used
  - Provides plausible deniability for individuals
- Implementation in FL:
  - Client-side: Add noise to local updates
  - Server-side: Add noise during aggregation

Federated Learning: **Privacy Mechanisms**

Solutions

- **DP-SGD for Federated Learning**
  - Compute gradient g
  - Clip to max L2 norm C: $\bar{g} = g/\max(1, ||g||_2/C)$
  - Add Gaussian noise: $\tilde{g} = \bar{g} + N(0, \sigma$
  - Update model with noisy gradient $\tilde{g}$

python

```python
# DP-SGD pseudocode
def dp_sgd_update(model, data, epsilon, delta, C=1.0):
    gradients = compute_gradients(model, data)

    # Clip gradients
    norms = np.linalg.norm(gradients, axis=1)
    scaling = np.minimum(1, C / norms)
    clipped_grads = gradients * scaling[:, np.newaxis]

    # Add noise (calibrated to epsilon, delta)
    sensitivity = C
    noise_scale = sensitivity * compute_sigma(epsilon, delta)
    noisy_grads = clipped_grads + np.random.normal(0, noise_scale,

    return noisy_grads.mean(axis=0)
```

Federated Learning: **Privacy Mechanisms**

Solutions

- **Privacy-Utility Tradeoff**
    - Lower ε = stronger privacy guarantees
    - Privacy guarantees compose across rounds
- Effects on model quality:
    - More privacy → more noise → lower accuracy
    - Typical accuracy drop: 1-5% with moderate DP
    - Larger models often more resilient to DP noise

Federated Learning: **Privacy Mechanisms**

Solutions

- **Secure Aggregation Goal:** Server learns only aggregate update, not individual updates
  - Secure Multi-Party Computation (MPC):
    - Allow computation on encrypted/masked data
    - No party learns others' inputs
  - Practical implementations:
    - Google's SecAgg protocol
    - Handling dropouts with secret sharing
    - Communication cost: $O(n^2)$ or $O(n \log n)$

Frameworks & Implementation

Frameworks & Implementation

- TensorFlow Federated (TFF): Google's production-ready framework
- Flower: Lightweight, language-agnostic FL framework
- PySyft/PyGrid: Privacy-first ML ecosystem
- Code Walkthrough: Simple FL implementation using Flower

Federated Learning

Key Takeaways

- Statistical heterogeneity requires specialized algorithms like FedProx, FedNova, or personalization techniques
- Communication efficiency is critical for practical deployment - use compression, quantization, and smart client sampling
- Robust aggregation protects against unreliable clients and attacks through methods like Krum, Trimmed Mean, and Median
- Privacy mechanisms like DP, Secure Aggregation, and HE offer different tradeoffs between privacy, utility, and efficiency
- Real-world systems often combine multiple solutions to address the full spectrum of challenges