International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 13 Issue: 1
***Article Received:** 25 January 2025 **Revised:** 12 February 2025 **Accepted:** 01 March 2025*

Federated Learning a Collaborative Machine Learning Across Countries with Data Privacy

Federated Learning for Global Collaboration: Securing Citizens Data While Enabling Cross-National AI and ML Development

Narendra Lalkshmana Gowda
Colorado State University Global
Building 555 17th St #1000, Denver
CO 80202, United States
e-mail: narendra.lakshmanagowda@ieee.org

Abstract— With growing importance of data in shaping policies, economic strategies, and healthcare systems, securing citizens data has become a critical issue for national governments. At the same time, the potential benefits of large-scale collaborative machine learning (ML) across countries are undeniable. Federated learning (FL) offers a unique solution to this dilemma by enabling the training of AI models across decentralized data sets without requiring data to be shared. This paper explores how different countries can use federated learning to contribute to collaborative machine learning while ensuring national data security. We examine the privacy-preserving mechanisms in FL, the technical challenges, and propose a framework for cross-country collaboration on a global scale..

Keywords- Federated Learning, Data Security, Cross-National Collaboration, Machine Learning, Privacy-Preserving AI

I. Introduction

In an era where data serves as the base or both economic and social advancement, the need to collaborate on large-scale AI models has never been more pressing. However, with this increased collaboration comes a heightened sensitivity to data privacy, particularly at the national level. Countries are often reluctant to share citizens data due to regulatory concerns and potential risks to individual privacy. Federated learning (FL)[1] provides an innovative approach, allowing multiple entities to contribute to a shared machine learning model without exposing local data to others.
This paper aims to explore how countries can leverage FL to collaboratively build machine learning models while maintaining strict data privacy, focusing on real-world use cases such as healthcare, finance, and smart city infrastructures.

II. Federated Learning: An Overview

Federated Learning is a distributed machine learning approach that trains models across decentralized data sources without transferring the actual data. The main components of FL include:

- Decentralized Model Training: Data remains on local servers, with only model updates shared between participants.
- Privacy Mechanisms: Techniques such as differential privacy [2] and secure multi-party computation are used to safeguard sensitive data during the training process.
- Global Model Aggregation: A central server aggregates the locally trained model parameters without having direct access to any dataset

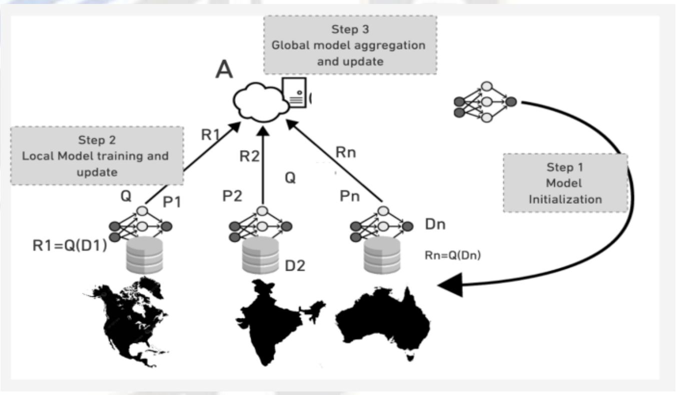

Figure 1: Sequence diagram of FL

The core functionality of federated learning, which involves a centralized Aggregator (A) and multiple parties (Pi), each with its own distinct dataset (Di). The Aggregator sends a query (Q) to all or a subset of participating parties {P1, P2, …, Pn}. This query typically asks the parties to provide information derived from their local datasets, such as updated model parameters after completing several rounds of local training. Upon receiving the query, each party performs the necessary computations on its dataset and generates responses {R1, R2, …, Rn}. For example, each party may conduct a single training epoch and send back the current model parameters. The parties then transmit their responses {R1, R2, …, Rn} to the central Aggregator. The Aggregator aggregates the information received from the parties. Using this aggregated data, the Aggregator updates the global

Article Received: *25 January 2025* ***Revised:*** *12 February 2025* ***Accepted:*** *01 March 2025*

model (M) and initiates the next round by issuing a new query to the parties, who continue with the subsequent training steps

III. BREAKDOWN OF THE FEDERATED LEARNING STEPS

A. Initialization:

- Initialize a global model w0 at the server.
- The server or aggregator defines a query Qt at time t to specify which model parameters or tasks should be updated by the clients.
- Set the total number of clients K.
- Set the number of communication rounds T.
- Set the number of local training epochs E and learning rate η for each device.
- Communication Round $t = 0,1,\dots,T-1$

B. Aggregator/Server:

- Send query Qt and the current global model wt to a subset of clients $St \subseteq 1,2,\dots,K$, indicating which part of the model or specific task to update.

C. Client-Side Response Generation:

- For each client $pk \in St$
- Each client downloads the global model wt and the query Qt.
- Based on the query Qt, the client performs local training on the specified portion of the model or data using its local dataset Dk.
- The client minimizes its local loss function on the query Qt , which could be represented as
$$L_k(w_t, Q_t) = \frac{1}{|\mathcal{D}_k|} \sum_{(x_i,y_i)\in\mathcal{D}_k} \ell(f(x_i; w_t, Q_t), y_i)$$
- Perform **E** epochs of local training to compute the update r_k^{t+1}. where:
$$r_k^{t+1} = w_t - \eta\nabla L_k(w_t, Q_t)$$
This update represents the response from the client to the query, based on the locally available data.

D. Server Aggregation:

- Each client p_k sends its update r_k^{t+1} back to the aggregator.
- The aggregator computes the aggregated response R_t by combining the responses from the selected clients:
$$R_t = \sum_{k\in\mathcal{S}_t} \frac{|\mathcal{D}_k|}{\sum_{j\in\mathcal{S}_t} |\mathcal{D}_j|} r_k^{t+1}$$
- This could represent a weighted sum of the model updates, where the weights are typically proportional to the size of the local datasets |Dk|.

E. Global Model Update:

- The global model is updated based on the aggregated response R_t: $w_{t+1} = w_t + \lambda R_t$ where λ is a step size parameter or aggregation factor that controls the influence of the aggregated responses on the global model.

*F. Repeat: Repeat the steps above for **T** communication rounds, iteratively refining the global model until convergence or achieving satisfactory performance.*

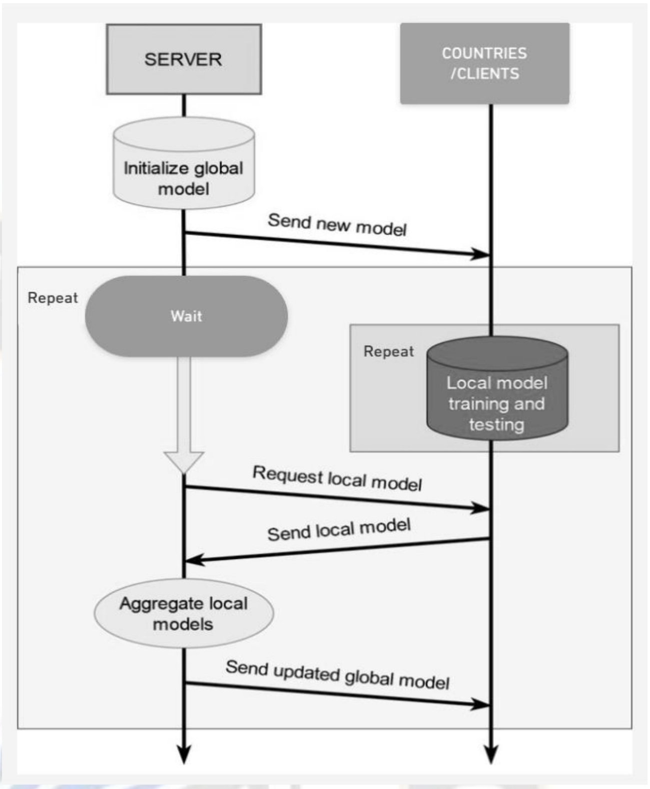

Figure 2: Sequence diagram of FL

This iterative process continues until the final global model (M) is produced and shared with the parties. It's important to note that the data remains with the parties throughout the entire process. However, while this method limits direct data sharing, it may not fully safeguard private information in adversarial scenarios.

IV. ASYNCHRONOUS FEDERATED LEARNING

Distributed systems are an effective way to achieve parallel and efficient computation [3], offering excellent scalability. In asynchronous federated learning, multiple clients (such as institutions or countries) independently and simultaneously train their local models on their respective datasets. Each client processes its data in parallel and periodically sends updates, such as model parameters, to a central Aggregator that builds and updates the global model. This asynchronous nature means that clients do not need to wait for each other to complete their local training [4], allowing the global model to be updated faster as more clients join the learning process. As a result, the global model benefits from quicker iteration cycles, which improves its accuracy and convergence speed. The efficiency of this method is particularly evident in scenarios where clients have varying computational power and data availability. For instance, as more clients contribute updates at their own pace, the global model is continuously refined, accelerating the learning process even when some clients are slower than others [5]. The following table provides data points showing how asynchronous federated

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 13 Issue: 1
***Article Received:** 25 January 2025 **Revised:** 12 February 2025 **Accepted:** 01 March 2025*

learning leads to faster model convergence as the number of clients increases.

V. Ensuring Data Security in Federated Learning

Federated learning offers a way to develop AI models for enterprise clients without directly accessing their data, making it a useful approach for addressing privacy regulations such as GDPR. It also enables joint training efforts across multiple organizations. While this method may work for some cases, many enterprise clients require robust privacy and security assurances when sharing model parameters with third parties to meet regulatory requirements. In particular, if an enterprise partner participates in a federated learning consortium, it often involves data that is sensitive to its business operations and clients. Competitors and external malicious actors are motivated to gain access to this data, making it crucial for any federated learning platform to safeguard against colluding or malicious agents.

There are several privacy risks in the basic federated learning framework, including:

- The potential for inferring private data during model training.
- The risk of data leakage through the deployment of the final predictive model.

Additionally, we identify two types of potential malicious actors:

- Internal attackers: where one or more parties may collude to extract data from other participants.
- External attackers: who may attempt to extract private information from the final model using techniques such as membership inference attacks.

Studies such as have shown that these attacks are possible unless proactive steps are taken. To mitigate these risks, approaches like multiparty computation are used to secure individual responses, and local differential privacy [5] is applied to defend against membership inference attacks and prevent data leakage in the final model. While differential privacy introduces noise to responses sent to the aggregator, reducing the chances of data leakage, this noise can also degrade model performance, especially in distributed settings. Therefore, achieving the right balance between maximizing privacy and maintaining model accuracy is a complex challenge.

A. Addressing the Gap in Federated Learning Privacy

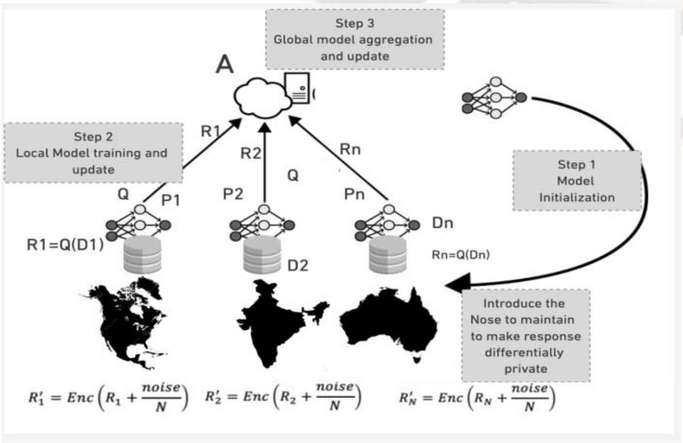
Step 3
Global model aggregation and update
A
Step 2
Local Model training and update
R1
R2
Rn
Q
Step 1
Model Initialization
Q
P1
P2
Pn
Dn
R1=Q(D1)
D2
Rn=Q(Dn)
Introduce the Nose to maintain to make response differentially private
$R_1' = Enc\left(R_1 + \frac{noise}{N}\right)$
$R_2' = Enc\left(R_2 + \frac{noise}{N}\right)$
$R_N' = Enc\left(R_N + \frac{noise}{N}\right)$

Figure 3: Federated learning with Noise injection

The AI Security and Privacy Solutions team at IBM Research has been developing privacy-preserving techniques that enable users to collaboratively train accurate machine learning models using federated learning while safeguarding data owners' privacy. To enhance this, the team has designed a new federated learning framework, as illustrated in Figure 2, which integrates multiparty computation and differential privacy. In this hybrid approach, not only does the data remain local, but it is also protected against malicious actors through encryption and differential privacy. This ensures that all participants in the training process benefit from mathematically proven privacy guarantees without compromising model performance.

B. A Hybrid Approach to Privacy-Preserving Federated Learning

Using threshold homomorphic encryption based on the Paillier cryptosystem, our framework minimizes the amount of noise that each participant must introduce while still achieving differential privacy guarantees (for a detailed mathematical explanation, refer to our paper). Here's an overview of how our private federated learning framework operates:

- The Aggregator sends a query (Q) to each party {P1, P2, …, Pn}, and based on their respective datasets, each party computes replies {R1, R2, …, Rn}.
- Unlike standard federated learning, each party adds noise to their responses based on the number of queried parties to ensure differential privacy. Before sending these noisy replies to the Aggregator, each party encrypts their responses using the specified encryption scheme.
- The Aggregator collects the encrypted replies from all parties.
- The encrypted aggregated result is then sent back to each party, who use their partial keys to perform partial decryption and send the results back to the Aggregator.
- The Aggregator combines the partially decrypted results from each party to obtain a plaintext version of the aggregated noisy replies (e.g., the average of the noisy replies) and issues the next query for the following training step.

This iterative process continues until the final model (M) is created and shared with all participants.

C. Key Enhancements for Privacy and Security

We introduced several crucial elements to strengthen the privacy and security of this federated learning approach against malicious agents:

- Differential Privacy: An algorithm is considered differentially private if the inclusion of a single data instance results in only statistically negligible changes to its output. By limiting the influence of any individual data point on the final model, we reduce the ability of adversaries to infer membership in the dataset.
- Additive Homomorphic Encryption [6]: This encryption technique allows the Aggregator to perform calculations on encrypted data without needing to decrypt it [8]. In our framework, an additively homomorphic encryption scheme ensures privacy by enabling the secure aggregation of parties' responses

Crucially, by aggregating noisy models before decryption, we can reduce the amount of noise needed per dataset to maintain differential privacy. Our goal is to ensure that the final decrypted result is differentially private, preventing data leakage, while minimizing the noise added during local training by each party. In fact, the amount of noise can be reduced in proportion to the number of participating parties.

D. Balancing Privacy and Model Performance

Additive homomorphic encryption allows us to lower the noise required per party while ensuring that no single reply is decrypted until all responses are combined. As more data parties contribute, the overall amount of noise injected remains consistent, allowing for the training of machine learning models with high accuracy.

VI. Challenges and Solutions for International Federated Learning

While the benefits of federated learning are clear, several challenges must be addressed for successful cross-country collaboration:

- Data Heterogeneity: Different countries generate data in different formats and structures, which can affect the performance of a global model. Solutions such as domain adaptation and data harmonization techniques can be implemented to standardize the contributions of each participant.
- Communication Overhead: Federated learning requires frequent communication between local and central servers. Optimizing communication protocols through techniques like federated averaging (FedAvg) [7] can reduce the overhead and ensure the scalability of FL systems.
- Trust and Governance: For federated learning to be successful on an international scale, there must be agreements in place regarding data usage, model ownership, and intellectual property. Creating a global governance framework for federated learning can provide clarity on these issues.

VII. Proposed best practices in the Framework for Cross-National Federated Learning

- Standardized Protocols: Establish a global protocol for federated learning that can be adopted by countries with diverse regulations and technical capabilities.
- Decentralized Governance: Set up an international body to oversee federated learning collaborations, ensuring compliance with local laws and promoting transparency.
- Incentivization Models: Develop incentive structures to encourage countries to contribute to global models while ensuring their national interests are protected

VIII. System Design for Training a Global Model using Federated Learning

The system for federated learning enables global model training through continuous, decentralized client from different country participates, each client asynchronously pushes local model updates. The system should handle high client utilization, asynchronous client participation, and fast model aggregation to scale efficiently. The design is structured into several key components that handle the training process from client selection to global model updates

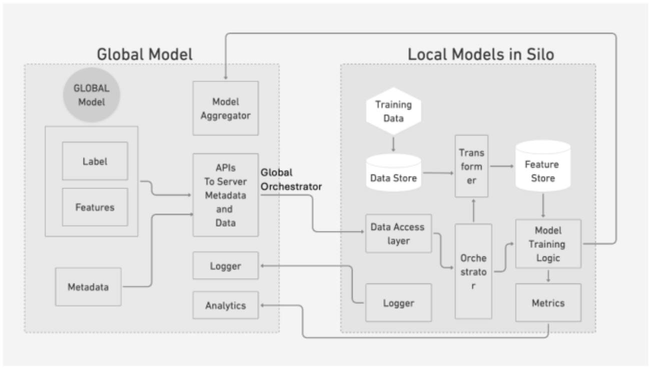

Figure 4: Federated learning System Design

A. Architecture Overview

The system is composed of several main components:

- Clients: Distributed devices or nodes that download the global model, train it locally on their own data, and send updates to the server.
- Global Model delivery endpoint: A network for distributing the global model, configuration files, and other resources to clients, to increase the reliability these services can be deployed using Hybrid deployment model [10].
- Model Aggregators: Responsible for collecting and aggregating local model updates from clients and updating the global model. Stateful components that aggregate client updates and handle individual task-related model updates
- Global orchestrator/ coordinator: Manages task assignments, client selection, and task redistribution. To increase the performance for parallel local model updates the orchestrator should be written using highly async programming languages [11].
- Local Orchestrator: Agents that help assign clients to tasks by interacting with the Global orchestrator

B. Client-Server Interaction in Asynchronous Federated Learning

- Client Selection and Task Assignment
  Each client attempts to connect to the system by requesting global model metadata and request to train using local data. The coordinator sends metadata for client based on the model to be trained using local data and existing global model metadata. The Selector forwards the client to an Aggregator based on client demand, where demand is tracked for each task to ensure proper task allocation.
- Local Training on Clients
  Upon being assigned to a task, clients download the

global model, configuration, and training instructions from the Model deliver APIs. Clients perform local training on their datasets asynchronously, , independent of other clients, and report the status once training is complete. After training, clients upload their local model updates to the server, typically after compressing or masking the model.

- Model Aggregation Process
  To efficiently handle the frequent model updates generated by clients asynchronously, the system employs the following mechanisms:

Parallel Model Aggregation: Once a client uploads the local model, the update is serialized and placed into an in-memory queue. A separate set of threads processes the updates, deserializing them and performing intermediate aggregations. Aggregation across multiple cores is parallelized to speed up the process. The thread responsible for intermediate aggregation is assigned via a hashing algorithm to reduce contention.

Aggregation Strategy:As updates are received, they are aggregated into a global model. Aggregation can be done using a weighted average, with each client's contribution potentially weighted by factors such as the amount of data used for training or the client's overall performance. The system performs incremental updates, where local updates are integrated into the global model as soon as they are received.

Final Global Model Generation: Once the cumulative number of aggregated updates reaches the system's aggregation goal, the final model aggregation is performed, generating a new global model. The global model is then redistributed back to clients for further rounds of training.

Asynchronous Update Frequency: Asynchronous model learning model updates proceeds as soon as client updates are received, leading to a higher frequency of global model updates. This asynchronous nature allows the global model to be updated up to 30 times more frequently than in synchronous systems

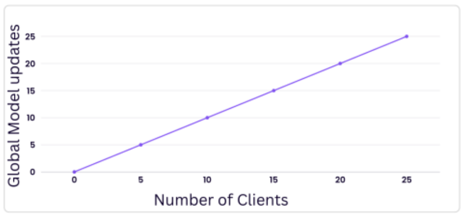

Figure 5: model updates performance with clients

IX. Incentivizing Federated Learning Across Countries/clients

Federated learning, where participants from different countries contribute to building a shared machine learning model without sharing raw data, provides a unique opportunity to drive global AI collaboration. However, one key challenge is how to incentivize participants, especially when some countries or institutions may contribute more than others. One approach to incentivizing participants is by creating a reward system based on their contributions to the overall model's performance. This can be done by tracking the value added by local models to the global model. A simple formula to measure contribution could be based on the improvement in model performance, such as:

$$C_i = \frac{\Delta P_i}{\sum_{j=1}^{n} \Delta P_j}$$

Where Ci is the contribution of participant ii, ΔP_i is the improvement in performance (e.g., accuracy, precision) brought by the local model of participant ii, and nn is the total number of participants. Countries that contribute more to improving the global model would thus receive higher rewards, creating an incentive for more active participation.

A. *Contribution-Based Reward Models*

Another approach is to implement a contribution-based reward model [11] that gives countries or participants a direct incentive based on their level of participation. For instance, countries contributing more data or computational resources can receive a proportional share of model ownership or future model usage rights. This can be implemented using a reward function:

$$R_i = \alpha \cdot D_i + \beta \cdot C_i$$

Where Ri is the total reward for participant ii, Di represents the amount of data contributed, Ci represents the contribution to model performance, and α, β are weights assigned to data and model improvement respectively. Countries with more resources and higher quality datasets can therefore be incentivized to participate more, as they would receive larger rewards. These rewards can be monetary or in the form of preferential access to the trained global model.

B. *Performance-Based Model Allocation*

A performance-based model allocation system [12] could also be implemented, where countries that contribute more get access to a better version of the global model. For example, instead of all participants receiving the same version of the trained model, those who have contributed more could receive a version of the model that is fine-tuned to their local data. The performance score Pi of each country's local model could be factored into the allocation:

$$M_i = M + \gamma \cdot P_i$$

Where Mi is the allocated model for country i, M is the global model, and γ is a tuning parameter that adjusts the model based on the country's local performance Pi. This incentivizes countries to invest in improving their local models, as their reward will be a more tailored version of the global model.

C. *Collaborative Research Incentives*

Lastly, collaboration between countries can be incentivized by providing joint ownership or intellectual property (IP) rights to the model developed through federated learning. Countries that contribute more to the development of the global model could

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 13 Issue: 1
***Article Received:** 25 January 2025 **Revised:** 12 February 2025 **Accepted:** 01 March 2025*

receive a higher share of the IP or profit generated from the deployment of the model. A simple collaborative research incentive [13] can be designed as:

$$I_i = \frac{C_i \cdot S}{\sum_{j=1}^{n} Cj}$$

Where I_i is the incentive (e.g., IP rights, profits) for participant i, Ci is the contribution score, and S is the total IP or profits generated. This ensures that countries with higher contributions receive a greater share of the rewards, encouraging them to actively participate and contribute to global AI research efforts.

X. Off the Shelf popular frameworks to Implement federated learning

A. TensorFlow Federated

Developed by Google, TensorFlow [14] Federated (TFF) is a Python-based, open-source framework for training machine learning models on decentralized data. This framework has been pioneering the experimentation with federated learning as an approach.TFF performs in two main API layers:
Federated Learning (FL) API offers high-level interfaces that enable developers to plug existing machine learning models to TFF without the need to dive deeply into how federated learning works. Federated Core (FC) API offers low-level interfaces that provide opportunities to build novel federated algorithms.

B. OpenFL

Developed by Intel, OpenFL[15] (Open Federated Learning) is an open-source framework that leverages the data-private federated learning paradigm for training ML algorithms. The framework comes with a command-line interface and a Python API. Open FL can work with training pipelines built with PyTorch and TensorFlow while it can go beyond and work with other frameworks.

C. IBM Federated Learning

IBM Federated Learning [16] is a framework that promises data scientists and machine learning engineers an easy integration of federated learning workflows within the enterprise environment. This federated learning framework supports a variety of algorithms, topologies, and protocols out-of-the-box, including Linear classifiers/regressions, Deep Reinforcement Learning algorithms, Naïve Bayes, Decision Tree, and Models written in Keras, PyTorch, and TensorFlow, to name a few.
Not to mention that researchers in the field of federated learning can use the existing functionality of the framework to fit the specific needs of their organization or the application domain.

XI. Conclusion

Federated learning presents a powerful opportunity for countries to collaborate on machine learning projects while preserving the privacy and security of their citizens' data. By implementing privacy-preserving mechanisms and establishing international protocols, we can foster cross-national collaborations in critical sectors such as healthcare, finance, and smart cities. Future research should focus on refining the technical aspects of federated learning and developing governance models that promote trust and cooperation on a global scale.

References

[1] Xu, J., Glicksberg, B. S., Su, C., Walker, P., Bian, J., & Wang, F. (2020). Federated Learning for Healthcare Informatics. *Journal of Healthcare Informatics Research*, *5*(1), 1–19. https://doi.org/10.1007/s41666-020-00082-4

[2] Jain, P., Gyanchandani, M., & Khare, N. (2018). Differential privacy: its technological prescriptive using big data. *Journal of Big Data*, *5*(1). https://doi.org/10.1186/s40537-018-0124-9

[3] Tech Dummies Narendra L. "System Design Basics: Real-Time Data Processing." *YouTube*, 30 June 2019, www.youtube.com/watch?v=dZ3swmtR1As. Accessed 19 Jan. 2025.

[4] Tech Dummies "System Design Basics: When to Use Distributed Computing | How Distributed Computing Works." *YouTube*, 29 June 2019, www.youtube.com/watch?v=3zCK_5U69A8. Accessed 19 Jan. 2025

[5] Gowda, N. L. (2024, May 4). *Bridging classical conditioning and deep reinforcement learning: advancements, challenges, and strategies for autonomous systems*. https://journal.esrgroups.org/jes/article/view/4121

[6] *Practical Additive Homomorphic Encryption for Statistical Analysis over Encrypted Data*. (2016, February 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/7456817

[7] Sun, T., Li, D., & Wang, B. (2021, April 23). *Decentralized federated averaging*. arXiv.org. https://arxiv.org/abs/2104.11375

[8] Wood, A., Altman, M., Bembenek, A., Bun, M., Gaboardi, M., Honaker, J., Nissim, K., O'Brien, D. R., Steinke, T., & Vadhan, S. (n.d.). *Differential Privacy: a primer for a Non-Technical audience*. Scholarship@Vanderbilt Law. https://scholarship.law.vanderbilt.edu/jetlaw/vol21/iss1/4/

[9] *Rewarding contribution | Commentary and insights | Tools | HR & Compliance Centre.co.uk*. (n.d.). https://hrcentre.uk.brightmine.com/commentary-and-insights/rewarding-contribution/17441/#:~:text=Contribution%2Drelated%20pay%20is%20essentially%20a%20mix%20of%20the%20two,competence)%20are%20crucial%20to%20performance.

[10] Hybrid cloud deployments for distributed systems. (2025). *International Journal of Computer Applications Technology and Research*. https://doi.org/10.7753/ijcatr1401.1008

[11] Evolution of programming languages: From punch cards to AI-Powered LLMs. (2025). *International Journal of Computer Applications Technology and Research*. https://doi.org/10.7753/ijcatr1401.1003

[12] IFAD. (n.d.). *IFAD's performance-based allocation system: Frequently asked questions*. https://webapps.ifad.org/members/wgpbas/docs/IFADs-performance-based-allocation-system-Frequently-asked-questions-e.pdf

[13] Nandi, A., Ngan, T. "., Singh, A., Druschel, P., & Wallach, D. S. (2005). Scrivener: Providing incentives in cooperative content distribution systems. In *Lecture notes in computer science* (pp. 270–291). https://doi.org/10.1007/11587552_14

[14] *TensorFlow federated*. (n.d.). TensorFlow. https://www.tensorflow.org/federated

[15] *OpenFL - Creative expression for desktop, mobile, web and console platforms*. (n.d.). https://www.openfl.org/

[16] *IBM watsonx.ai and watsonx.governance 2.0.x*. (n.d.). https://www.ibm.com/docs/en/watsonx/w-and-w/2.0.x?topic=models-federated-learning