2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT)

2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT) | 979-8-3315-1857-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/CE2CT64011.2025.10939611

A Survey of Federated Learning Orchestration Using Kubeflow: Challenges, Advances, and Future Directions

Prashanth Josyula
Principal Member of Technical Staff
Salesforce
San Francisco, USA
prashanth.16@gmail.com

Sethuraman Ulaganathan
IEEE Senior Member
San Francisco, USA
ulaganathan.sethuraman@gmail.com

Sai Kumar Arava
Machine Learning Engineer
Adobe
San Francisco, USA
aravasai@gmail.com

***Abstract*—In distributed machine learning, Federated Learning (FL) has become a game-changing paradigm that allows for cooperative model training while protecting data privacy and taking regulatory compliance into account and with a focus on edge computing integration and privacy protection techniques. This paper offers a systematic assessment of federated learning orchestration using Kubeflow, looking at architectural approaches and implementation issues in Kubernetes-native contexts and includes thorough technical evaluation and code-driven solutions, identifies a number of crucial implementation challenges for Federated Learning processes, such as edge gateway service design, dynamic device management, and adaptive privacy methods. We explore exciting research avenues like context-aware federation protocols, adaptive privacy mechanisms, and dynamic privacy preservation. We also point out important research gaps in existing implementations, particularly with regard to standardized performance metrics, sophisticated fault tolerance mechanisms, and machine learning-optimized orchestration strategies. For researchers and practitioners working on Kubeflow-based federated learning deployments, this paper offers conceptual examples for managing heterogeneous device capabilities, secure model exchange, and adaptive training configurations in Kubeflow-based FL systems highlighting important areas that need further research for enterprise-grade implementations.**

***Index Terms*—Federated Learning, Kubeflow, Kubernetes, Edge Computing, Privacy Preservation, Distributed Machine Learning, Kubernetes, Artificial Intelligence, ML Ops, Scalable Computing**

I. Introduction

The two main goals of this survey article are to synthesize possible architectural solutions based on our thorough literature review and to systematically review present approaches to federated learning orchestration using Kubeflow and we have included code samples and implementation patterns, that are conceptual frameworks based on the results of our study rather than solutions that have been empirically verified. Our objective is to close the gap between theoretical knowledge and real-world application by offering conceptual examples of potential solutions for identified challenges.

With the advent of edge devices and distributed computing paradigms, the field of machine learning has experienced a radical change, and this development has sped up the deployment of federated learning in a variety of industries, including telecommunications and healthcare [1]. Federated learning has emerged as a compelling solution that allows machine learning while keeping sensitive data at its source, as enterprises struggle with increasingly complicated data, privacy requirements and the necessity for collaborative model training. The needs for machine learning deployments have changed significantly because of the spread of edge devices and the exponential rise in distributed data generation. When handling privacy-sensitive data across organizational boundaries, traditional centralized systems have significant limitations. These issues are resolved by federated learning, which makes it possible to train models on decentralized data sources without raw data exchange [2].

For coordinating these complex federated learning workflows, Kubeflow, a Kubernetes-native framework for machine learning operations emerged as a promising solution and its natural interface with Kubernetes and containerized approach offers a strong basis for managing distributed training processes [3]. Kubeflow and federated learning combine to form a novel technological intersection that necessitates careful research, especially as businesses look for production-ready solutions that can scale across various computing environments while protecting data privacy and adhering to legal requirements.

Our comprehensive review thoroughly examines current approaches for implementing Federated Learning (FL) using Kubeflow, looking at implementation techniques, architectural patterns, and operational difficulties that arise in real world scenarios and additionally, it looks at and evaluates established patterns and best practices in implementation strategies, focusing on solutions that effectively mitigate typical FL orchestration roadblocks. Current approaches to privacy, security, and scalability concerns are also evaluated critically in this paper, providing an in-depth analysis of their advantages and disadvantages in practical deployments and lastly, it identifies important research gaps and suggests exciting avenues for future study that could advance FL orchestration using Kubeflow.

The rest of the paper is structured as follows. Section II lays

the theoretical groundwork by providing essential background knowledge on Kubeflow architecture and federated learning. A thorough explanation of our review approach, including search strategies and analysis frameworks, is provided in Section III. The main literature review, which looks at architectural approaches, privacy concerns, and performance optimization techniques, is presented in Section IV. Our findings are summarized in Section V, together with their implications for researchers and practitioners. Section VI wraps up the survey by highlighting significant contributions and suggesting exciting avenues for further study.

II. Background and Preliminaries

This section introduces the fundamental ideas of federated learning orchestration in Kubernetes-native systems, focusing on Kubeflow's role in managing distributed training workflows. We look at the convergence between federated learning principles with containerized orchestration platforms, offering important context for understanding implementation issues and architectural concerns.

A. Federated Learning Fundamzentals

When it comes to distributed machine learning, federated learning is a game-changer. It allows model training while protecting data privacy and satisfying regulatory standards. Unlike previous centralized systems, FL enables enterprises to collaborate on model training without relocating sensitive data from its original location, addressing major difficulties in regulated industries and scenarios with stringent privacy regulations [4].

Federated learning systems are composed of three major components that work together: a federation coordinator, which manages training rounds and model aggregation, edge participants, who perform local training on private datasets, and a secure communication layer, which ensures that model updates are safely transmitted. These components communicate via sophisticated protocols that guarantee data sovereignty while enabling successful collaborative learning [5].

Beyond the traditional FederatedAveraging algorithm, modern FL implementations employ a variety of approaches including FedProx for handling device heterogeneity, FedMA for managing diverse model architectures, and hierarchical federated learning for improved scalability. These algorithms can be implemented in different topologies, from simple star configurations to more complex ring and hierarchical arrangements, each offering distinct trade-offs between communication efficiency, convergence speed, and system complexity [6].

B. Kubeflow Architecture

Kubeflow offers a complete Kubernetes-native platform made specifically for orchestrating machine learning processes at scale. At its core, the platform uses Kubeflow Pipelines to run complex federated learning processes as directed acyclic graphs. Specialized tasks like model distribution and aggregation across participating nodes are handled by custom operators.

Scalable model serving among distributed participants is made possible by the platform's advanced model management features, which are based on KServe while maintaining version control and artifact management this infrastructure is crucial for maintaining reproducibility throughout complex distributed training processes and monitoring model evolution throughout federation rounds. The metadata management and monitoring tools offered by Kubeflow offer vital insight into the federated learning processes. The platform monitors training progress, participant performance, and system health using ML Metadata and custom metrics and this allows administrators to effectively oversee remote training operations and guarantee optimal resource use throughout the federation [7].

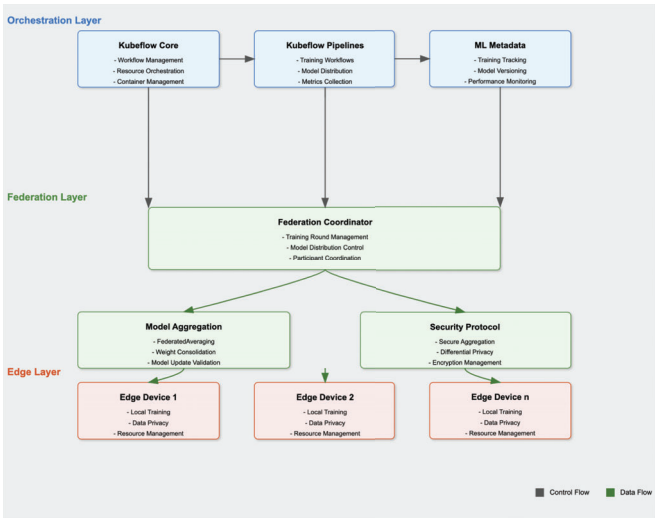

Fig. 1. High-level architecture of Kubeflow integrated with federated learning components, showing the interaction between Kubeflow Pipelines, KServe for model serving, and ML Metadata for tracking distributed training operations across federated participants

C. Kubeflow with Federated Learning

When Kubeflow is integrated with federated learning, specific architectural patterns emerge that require meticulous design and execution and because it manages the primary tasks including participant registration, capability discovery, and secure model update transmission, the edge gateway service is a crucial component of this integration. It also controls communication between edge participants and Kubernetes clusters. Sophisticated scheduling and allocation techniques are necessary for resource management in federated learning environments, especially when complex topologies are being implemented. To maintain optimal training performance and system stability, the system must dynamically manage federation rounds, adjust to different participant capabilities, and ensure equal distribution of resources within the federation.

The architecture as shown in Fig. 1 incorporates robust privacy and security infrastructure with comprehensive fault tolerance mechanisms. This involves incorporating differential privacy techniques into training workflows, putting secure aggregation protocols into practice as Kubernetes services, and

making sure that participant failures during training rounds are handled automatically. These factors are very much essential in regulated sectors where system dependability and data protection are critical.

III. Review Methodology

We used Kubeflow to perform a multi-stage search across IEEE Xplore, ACM Digital Library, arXiv, Google Scholar, and Science Direct to systematically evaluate federated learning orchestration and the search framework included secondary phrases (”privacy preservation”, ”orchestration”, ”edge computing”, ”Kubernetes native”), tertiary refinement terms (”implementation”, ”deployment”, ”production environment”), and main terms (”federated learning”, ”Kubeflow”, ”distributed machine learning”). Only papers scoring above 70 points were included in the final analysis to guarantee focus on high-quality implementations with precise architectural details, validation metrics, and thorough documentation. Papers were evaluated for quality based on technical implementation (40 points), validation methodology (30 points), and documentation quality (30 points).

System architectures (component diagrams, interface specifications, deployment configurations), implementation details (privacy mechanisms, resource optimization, fault tolerance strategies), and performance attributes (model convergence approaches, communication patterns, resource management) were all analyzed through our comprehensive literature review. Through a comparative review of previous research, we were able to identify emerging patterns in architectural solutions (system design patterns, scaling strategies), optimization techniques (communication optimization, resource allocation), and security improvements (secure aggregation protocols, privacy preservation mechanisms) [2]. This review of the state of the art laid the groundwork for future research directions in this domain and helped in identifying important gaps in Kubeflow-based federated learning orchestration solutions.

IV. Literature Review

A. Architectural Strategies for Federated Learning Orchestration

The FederatedAveraging method, which forms the basis of many modern implementations, was proposed by McMahan et al. [8], who made the fundamental contribution to federated learning architecture. Their research demonstrated that it is feasible to learn from decentralized data while maintaining anonymity. Important insights into the structural needs of federated learning systems are provided by this architectural framework.

Building on this foundation, one of the first large-scale concepts for federated learning systems was presented by Bonawitz et al. [9]. Their research highlights crucial architectural elements needed for federated learning’s widespread deployment, emphasizing the value of robust orchestration systems for managing distributed training operations.

B. Communication Efficacy and Resource Allocation

One major issue with federated learning coordination is communication efficiency, Konečný et al. [10] provided a number of techniques to improve the effectiveness of communication in federated learning systems and their research emphasizes how important it is for participating nodes to optimize their communication protocols, which is a critical component of any federated learning implementation.

In the field of resource management, Wang et al. [11] looked into adaptive federated learning within resource-constrained edge computing systems. Their research demonstrates how resource management can be optimized in federated contexts, and their findings are particularly relevant to coordinated deployments where resource allocation is a key concern [3].

C. Considerations for Privacy and Security

Lyu et al. [12] conducted a thorough analysis of the security landscape of federated learning, including a detailed evaluation of the threats to these systems. Their research tackles a number of important security concerns that must be addressed in any federated learning orchestration architecture, including data privacy, model poisoning, and secure aggregation procedures. Backdoor attacks in federated learning were one of the security issues that Sun et al. [6] further investigated, exposing vulnerabilities and possible countermeasures and their findings highlight the need for comprehensive security measures in orchestration platforms and have significant implications for safeguarding federated learning systems.

D. Performance Optimization and Management of Non-IID Data

The important problem of handling non-IID (non-independent and identically distributed) data in federated learning environments was addressed by Sattler et al. [13]. Their research on communication-efficient and resilient federated learning provides significant insights for the implementation of efficient orchestration systems. Alongside this study, Li et al. [14] offered an in-depth analysis of the issues, approaches, and future directions in federated learning. Although many of the challenges they have identified have a substantial impact on orchestration strategies and execution techniques, their research tackles broader aspects of federated learning.

E. The Function of Kubeflow in Orchestrating Federated Learning

The review of the literature revealed a significant research gap in comprehensive Kubeflow-based federated learning implementations. We can evaluate how well Kubeflow’s features match the requirements of federated learning by applying the system design concepts described in recent research. According to Bonawitz et al. [9], Kubeflow’s built-in interface with Kubernetes offers robust support for handling distributed training, making it easier to scale the orchestration of federated learning participants across several nodes and clusters.

Kubeflow's resource scheduling and allocation techniques improve computational resource usage and support the adaptive needs of federated learning systems by successfully addressing the resource management challenges identified by Wang et al. [11]. By optimizing complex workflows that include model distribution, aggregation, and validation, Kubeflow's pipeline automation tools successfully address the operational issues identified by Li et al. [14]. Collectively, these capabilities highlight Kubeflow's potential to improve federated learning orchestration, notwithstanding the present absence of comprehensive implementation studies.

F. Challenges and Opportunities in Integration

Combining Kubeflow and federated learning has several problems and opportunities, underscoring critical domains that require further research and development and although Kubeflow offers good orchestration capabilities, it must be improved to effectively address key federated learning components. Based on the security concerns described by Lyu et al. [12], privacy-preserving practices are an important area of research because it is crucial to clarify how Kubeflow may implement secure aggregation and other privacy-preserving techniques in federated learning environments [1]. Additionally, utilizing the efficiency solutions proposed by Konečný et al. [10], communication optimization is a crucial area that requires enhancement to improve Kubeflow's network communication patterns. The lack of standardization in Kubeflow's federated learning process execution emphasizes the need to develop best practices for model distribution, aggregation, and validation to streamline and enhance its deployment in real-world scenarios.

V. Discussion

A. Synthesis of Findings

Our comprehensive analysis of Kubeflow-based federated learning orchestration identifies multiple levels of architectural challenges and improvements, and at the basic level, we faced the inherent conflict between the distributed nature of edge computing and Kubernetes cluster-centric architecture. This architectural mismatch calls for a sophisticated bridging mechanism that preserves Kubernetes orchestration's resilience while considering the unique characteristics of edge devices [3]. The edge gateway service whose protocol is defined in Algorithm 1 emerged as an essential part of the architecture which acts as more than just a communication bridge. This service also owns the configuration required to decide the overall federation strategy as defined in Algorithm 2. Based on the review of our literature, we suggest that an effective gateway implementation could serve as an advanced orchestration layer. We offer a conceptual implementations throughout the paper that shows how the key challenges identified by our survey could be resolved.

Most importantly, we discovered that treating edge devices as a homogeneous group results in suboptimal performance and resource consumption, and that a dynamic multi-tiered approach to device management is necessary for successful edge federation [1]. The code samples shows how the theoretical ideas and challenges identified in the literature could be converted into practical implementations and act as architectural blueprints based on the results of our survey. Rather than serving as solutions that are ready for production, these examples are intended to serve as a roadmap for future research and development, as they fill in the gaps seen in existing implementations while combining best practices and patterns found across various research.

Algorithm 1 Edge Gateway Service Protocol

Require: Device request D, Capability set C, Registry R
Ensure: Registration status and secure configuration
Protocol for managing federated learning communication between Kubernetes clusters and edge devices, handling participant registration, capability discovery, and secure model updates
function InitializeGatewayService(R)
$registry \leftarrow$ InitializeRegistry(R)
$security \leftarrow$ ConfigureSecureCommunication()
$monitor \leftarrow$ InitializeMonitoring()
return $registry$
end function
function RegisterDevice(D, C)
$device\_id \leftarrow D.id$
$capabilities \leftarrow$ DiscoverCapabilities(C)
$config \leftarrow$ GenerateSecureConfig($capabilities$)
$status \leftarrow$ CreateKubernetesCRD($device\_id$, $config$)
return $status$
end function
function SecureModelUpdate($device\_id$, $model$)
$encrypted\_model \leftarrow$ EncryptModel($model$)
$signature \leftarrow$ SignUpdate($encrypted\_model$)
$status \leftarrow$ TransmitSecurely($device\_id$, $encrypted\_model$, $signature$)
return $status$
end function

Fig. 2 illustrates our comprehensive architectural vision for integrating Kubeflow with federated learning. This three-tiered architecture demonstrates how Kubeflow's native capabilities can be extended to support distributed model training across edge devices. The top layer leverages Kubeflow pipelines for orchestrating the training workflow and hyperparameter optimization, while the middle layer implements a specialized edge gateway service that manages device coordination and secure communications. The bottom layer represents the edge devices themselves, where local model training occurs before updates are aggregated into the global model. This architecture addresses key challenges in federated learning deployment, including device heterogeneity, secure model exchange, and efficient resource utilization across the federation.

B. Research Gaps

Our survey uncovered several critical gaps in current implementations that warrant immediate attention [2]. Beyond the previously identified lack of standardization in device management, we discovered significant limitations in how current systems handle the dynamic nature of edge environments. The absence of sophisticated fault tolerance mechanisms represents

Algorithm 2 Federation Strategy Configuration Protocol

Require: Device capabilities C, Training parameters T, Privacy settings P
Ensure: Dynamic federation configuration for Kubeflow-based FL system
Protocol for dynamically managing device capabilities, training parameters, and privacy settings in Kubeflow-based federated learning system

```
function ConfigureFederationStrategy(C, T, P)
    for each device in C do
        if device.capability = HIGH then
            config ← {
              local_epochs: 5
              batch_size: 64
              update_frequency: "every_round"
              aggregation_role: "intermediate_aggregator"
              model_compression: "none"
              early_stopping: false}
        else if device.capability = MEDIUM then
            config ← {
              local_epochs: 3
              batch_size: 32
              update_frequency: "every_other_round"
              aggregation_role: "participant"
              model_compression: "quantization"
              early_stopping: false}
        else
            config ← {
              local_epochs: 1
              batch_size: 16
              update_frequency: "adaptive"
              aggregation_role: "participant"
              model_compression: "heavy_quantization"
              early_stopping: true}
        end if
        strategy[device] ← config
    end for
    return strategy
end function
function UpdateFederationConfig(device, metrics)
    performance ← EvaluateDevicePerformance(metrics)
    config ← strategy[device]
    if performance.degrading then
        config.model_compression ← IncrementCompression()
        config.early_stopping ← true
    end if
    return config
end function
```

a major gap, and current implementations often fail to adequately handle scenarios where devices drop out mid-training or experience significant performance degradation, and our implementation attempts to address this through a basic retry mechanism, as defined in Algorithm 3.

The absence of standardized performance metrics for edge-based federated learning systems is another notable gap. Traditional machine learning systems have established benchmarks; however, federated learning deployments require consideration of additional parameters such as system reliability, energy consumption, and communication efficiency to measure their effectiveness [7].

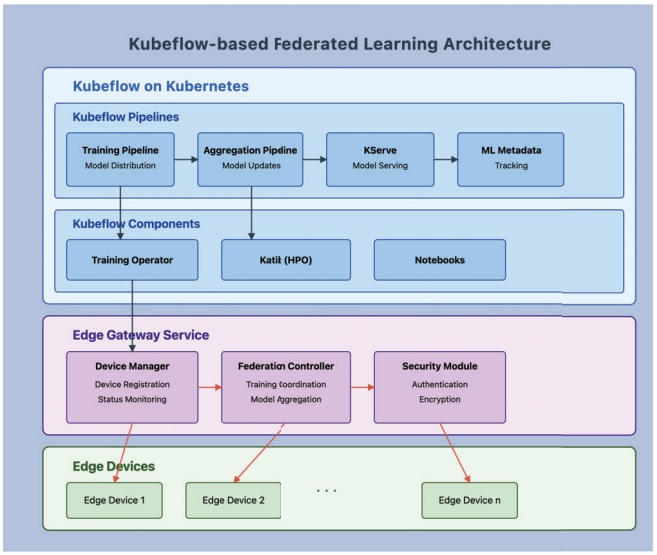

Fig. 2. This architecture consists of three layers: (1) Kubeflow pipelines and components for training, hyperparameter optimization, and model serving; (2) an edge gateway service for device management, training coordination, and secure communication; and (3) edge devices that perform local model training and share updates for global aggregation.

Algorithm 3 Federated Client Retry Protocol

Require: Model update M, Maximum retries $MAX\_RETRIES$
Ensure: Successful model update submission or local storage
Protocol for handling device failures and network interruptions during training rounds to ensure system resilience

```
function SubmitModelUpdate(M)
    retry_count ← 0
    while retry_count < MAX_RETRIES do
        compressed_update ← CompressUpdate(M)
        signature ← GenerateSignature(compressed_update)
        metadata ← CollectTrainingMetadata()
        request ← CreateSubmissionRequest(
          device_id, compressed_update,
          signature, metadata)
        response ← SubmitUpdate(request)
        if response.status = SUCCESS then
            ClearLocalUpdate()
            return SUCCESS
        end if
      grpc.NoRetryError
        retry_count ← retry_count + 1
        Wait(Min(2^retry_count, 30))
    end while
    StoreUpdateLocally(compressed_update)
    ScheduleDelayedRetry()
    return FAILURE
end function
function ScheduleDelayedRetry
    backoff ← CalculateExponentialBackoff()
    QueueRetryTask(backoff)
end function
```

C. Future Research Directions

Building on our results, we suggest several prominent areas of research that have the potential to make major strides in the field. First, we anticipate the creation of "context-

aware federation protocols" that consider the larger operational context in addition to basic device capabilities [1] as defined in Algorithm 4

As illustrated in Fig. 3, these research directions form an interconnected framework for advancing federated learning orchestration, with each area contributing to the overall goal of more robust and efficient systems.

Algorithm 4 Context-Aware Federation Protocol

Require: Device ID D, System context S
Ensure: Context-optimized federation parameters
Protocol for dynamically adjusting federation parameters based on device capabilities, network conditions, and operational context for optimized training performance
function CONTEXTAWAREPROTOCOL(D, S)
$context\_analyzer \leftarrow$ ContextAnalyzer()
$strategy\_optimizer \leftarrow$ StrategyOptimizer()
end function
function ANALYZECONTEXT($device\_id$)
$context \leftarrow \emptyset$
$context.capabilities \leftarrow$ GetDeviceCapabilities($device\_id$)
$context.network \leftarrow$ GetNetworkMetrics($device\_id$)
$context.behavior \leftarrow$ GetUsagePatterns($device\_id$)
$context.energy \leftarrow$ GetEnergyStatus($device\_id$)
$context.app\_requirements \leftarrow$ GetAppReq($device\_id$)
return $context$
end function
function OPTIMIZESTRATEGY($context$)
$constraints \leftarrow$ GetSystemConstraints()
$objectives \leftarrow$ GetOptimizationObjectives()
$strategy \leftarrow$ GenerateStrategy($context$,$constraints$,$objectives$)
return $strategy$
end function
function GENERATESTRATEGY($context$,$constraints$,$objectives$)
if $context.capabilities.compute\_power = HIGH$ **then**
return GenerateHighPerformanceStrategy()
else if $context.network.quality = POOR$ **then**
return GenerateNetworkEfficientStrategy()
else
return GenerateBalancedStrategy()
end if
end function

We also propose research into "adaptive privacy mechanisms" that can dynamically adjust the level of privacy preservation based on device capabilities and data sensitivity [2]. Algorithm 5 shows the adaptive privacy mechanism that we envision.

Finally, we suggest exploring the integration of machine learning techniques to optimize the orchestration itself. This might entail dynamically altering federation parameters through reinforcement learning and together with continuous developments in edge computing and 5G technologies, these research avenues offer promising prospects for enhancing the usefulness and effectiveness of federated learning in real-world applications. If these areas are successful, federated learning's applicability could be significantly expanded beyond its existing constraints, opening new application classes while still meeting efficiency and privacy concerns.

Algorithm 5 Adaptive Privacy Mechanism Protocol

Require: Device capabilities C, Data sensitivity S, Performance goals G
Ensure: Privacy configuration adapted to device capabilities and data sensitivity
Protocol for dynamically adjusting privacy preservation levels based on device capabilities and data sensitivity requirements in federated learning environments
function CONFIGUREPRIVACY(C, S, G)
if $S > 0.8$ **then**
return ConfigureHighPrivacy(C)
else if $S > 0.4$ **then**
return ConfigureMediumPrivacy(C)
else
return ConfigureBasicPrivacy(C)
end if
end function
function CONFIGUREHIGHPRIVACY($capabilities$)
if $capabilities.compute\_power = HIGH$ **then**
return {
differential_privacy: TRUE,
noise_multiplier: 0.9,
secure_aggregation: TRUE,
encryption_strength: "AES-256"}
else
return {
differential_privacy: TRUE,
noise_multiplier: 0.8,
secure_aggregation: TRUE,
encryption_strength: "AES-128"}
end if
end function
function CONFIGUREMEDIUMPRIVACY($capabilities$)
return {
differential_privacy: TRUE,
noise_multiplier: 0.5,
secure_aggregation: capabilities.supports_encryption}
end function

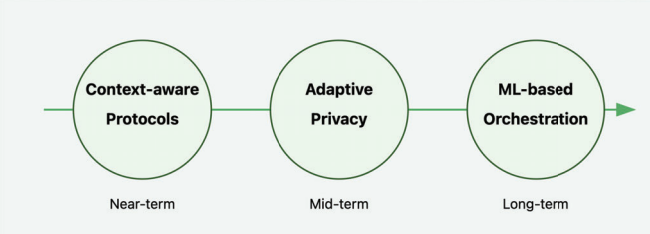

Fig. 3. Key future research directions for federated learning, emphasizing ML-optimized orchestration and advanced privacy techniques"

VI. CONCLUSION

The comprehensive survey of the complex landscape of federated learning orchestration using Kubeflow has shown significant advancements and critical obstacles in this rapidly evolving area [3]. We have given researchers and practitioners a fundamental framework for comprehending and putting into practice federated learning systems in Kubernetes-native environments by means of our systematic review of the existing literature and synthesis of potential solutions. While proposing conceptual implementation patterns that close the gap between

theoretical knowledge and real-world deployment, our analysis highlights the critical role of edge gateway services and the significance of dynamic device management [1]. The lack of standardized performance metrics for edge-based federated learning systems, the need for more advanced fault tolerance mechanisms, and the difficulty of implementing dynamic privacy preservation are just a few of the important research gaps that our investigation has revealed and that demand immediate attention [2]. Context-aware federation protocols and adaptive privacy methods are two of the exciting research directions we have suggested that could greatly improve the ability of the field to manage practical deployment scenarios. Despite lacking empirical validation, these suggested methods are useful models for researchers and practitioners creating their own implementations. We hope to have aided in the continuous development of more reliable, effective, and secure federated learning implementations that may satisfy the growing demands of contemporary distributed machine learning applications by offering a systematic assessment of the field and combining viable solutions.

REFERENCES

[1] M. R. Uddin, R. Akhtar, A. Gupta, and A. Chaudhuri, "Evolving topics in federated learning: Trends and emerging directions for IS research," in *Proceedings of the International Conference on Information Systems (ICIS 2024)*, 2024.

[2] N. Truong, "Privacy preservation in federated learning: An insightful survey from the GDPR perspective," 2021.

[3] M. Hassan, M. H. Rehmani, and J. Chen, "FedEdge: Federated learning with Docker and Kubernetes," 2023.

[4] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, and J. Roselander, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems*, vol. 1, 2019, pp. 374–388.

[5] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[6] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" in *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[7] F. Sattler, S. Wiedemann, K. R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[8] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[9] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, and J. Roselander, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems*, vol. 1, 2019, pp. 374–388.

[10] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[11] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[12] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[13] F. Sattler, S. Wiedemann, K. R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[14] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.