Communication-Efficient and Privacy-Preserving Federated Learning via Joint Knowledge Distillation and Differential Privacy in Bandwidth-Constrained Networks

Gad Gad, *Student Member, IEEE*, Eyad Gad, *Graduate Student Member, IEEE*, Zubair Md Fadlullah, *Senior Member, IEEE*, Mostafa M. Fouda, *Senior Member, IEEE*, and Nei Kato, *Fellow, IEEE*

***Abstract*—The development of high-quality deep learning models demands the transfer of user data from edge devices, where it originates, to centralized servers. This central training approach has scalability limitations and poses privacy risks to private data. Federated Learning (FL) is a distributed training framework that empowers physical smart systems devices to collaboratively learn a task without sharing private training data with a central server. However, FL introduces new challenges to Beyond 5G (B5G) networks, such as communication overhead, system heterogeneity, and privacy concerns, as the exchange of model updates may still lead to data leakage. This paper explores the communication overhead and privacy risks facing the implementation of FL and presents an algorithm that encompasses Knowledge Distillation (KD) and Differential Privacy (DP) techniques to address these challenges in FL. We compare the operational flow and network model of model-based and model-agnostic (KD-based) FL algorithms that enable customizing per-client model architecture to accommodate heterogeneous and constrained system resources. Our experiments show that KD-based FL algorithms are able to exceed local accuracy and achieve comparable accuracy to central training. Additionally, we show that applying DP to KD-based FL significantly damages its utility, leading to up to 70% accuracy loss for a privacy budget $\epsilon \leq 10$.**

***Index Terms*—B5G networks, deep learning, differential privacy, federated learning, gradient compression, heterogeneous federated learning, knowledge distillation.**

I. Introduction

In recent years, deep learning has experienced significant growth in various sectors where data collection and processing are feasible. However, the challenge of transferring data from edge devices to central servers while maintaining privacy and security remains a concern. Federated Learning (FL) emerges as an innovative framework enabling collaborative learning among distributed devices without the necessity of explicitly sharing their raw data. This approach holds substantial value across diverse applications [1], [2], especially in scenarios where data are generated at the B5G network edge. Traditional methods involving the transfer of all data to a central server for processing not only under-utilize the computing resources available at the B5G network edge but also entail significant time overheads and privacy risks.

Traditional centralized training requires moving data to a central location for processing, but FL allows distributed devices to train machine learning models on their local data while preserving data privacy. This not only reduces the need to transfer large amounts of data but also improves scalability and communication efficiency while ensuring users' privacy. Fig. 1 depicts the central training approach where data are sent from edge devices to the central server to train a model, which is then broadcasted to all clients. On the other hand, Fig. 2 presents an overview of standard FL [3] where each client trains his local model on local data, and the trained local models are sent to the server to be aggregated. The server broadcasts the aggregated model to clients. These steps are repeated in rounds until convergence or until a termination condition is met. Figs. 1 and 2 also list some of the pros and cons of central deep learning training and standard model-based FL, respectively. As an inherently private distributed learning algorithm, one of the significant benefits of FL is its ability to address privacy concerns. The increasing number of data breaches has raised awareness around data privacy, and regulations such as the European Union's General Data Protection Regulation (GDPR) are becoming increasingly stringent. FL provides an effective solution by keeping data on edge devices and ensuring that it is only used to train models without being shared with central servers.

FL enhances scalability since it enables distributed devices to train models on their local data and send only the model updates to the central server, reducing the need to move large amounts of data. Additionally, FL improves personalization by training

Manuscript received 1 November 2023; revised 24 March 2024; accepted 7 June 2024. Date of publication 31 July 2024; date of current version 7 November 2024. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant RGPIN-2020-06260. The review of this article was coordinated by Dr. Kaigui Bian. *(Corresponding author: Mostafa M. Fouda.)*

Gad Gad, Eyad Gad, and Zubair Md Fadlullah are with the Department of Computer Science, Western University, London, ON N6G 2V4, Canada (e-mail: ggad@uwo.ca; egad@uwo.ca; zubair.fadlullah@uwo.ca).

Mostafa M. Fouda is with the Department of Electrical and Computer Engineering, Idaho State University, Pocatello, ID 83209 USA, and also with the Center for Advanced Energy Studies (CAES), Idaho Falls, ID 83401 USA (e-mail: mfouda@ieee.org).

Nei Kato is with the Graduate School of Information Sciences, Tohoku University, Sendai 980-8577, Japan (e-mail: kato@it.is.tohoku.ac.jp).

Digital Object Identifier 10.1109/TVT.2024.3423718

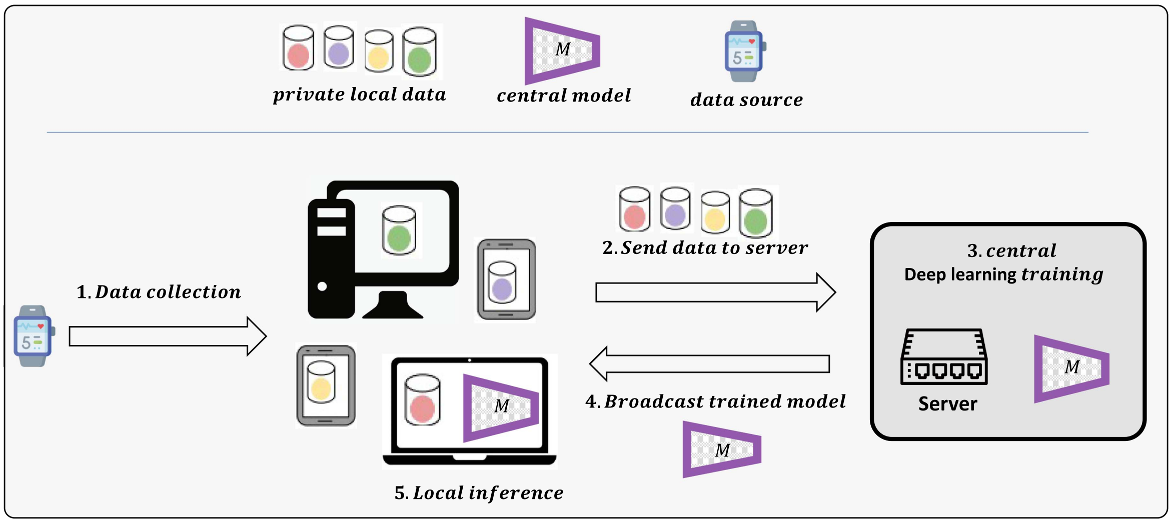


Fig. 1. In central deep learning, data are collected to a central server. This approach yields the best performance in terms of accuracy but poses privacy risks to users' data.

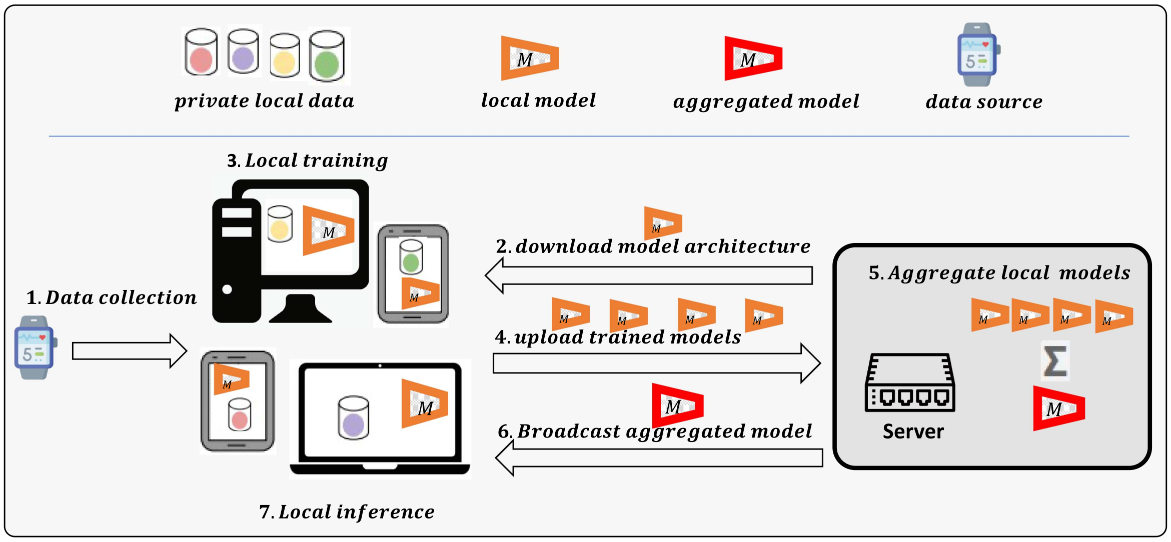


Fig. 2. In standard federated learning (FedAvg), models are trained on users' local data before they are sent to be aggregated at the server.

local models longer on local data while minimizing the global loss of the aggregated model.

FL [3] offers numerous advantages but also poses unique challenges. In this paper, we focus on three of these challenges, namely communication cost, privacy, and heterogeneity. Communication overhead remains a significant concern for applications that rely on distributed communication [4]. Traditional federated algorithms (e.g., FedAvg) exchange model weights whose size depends on the task. Many techniques have been proposed to reduce FL communication overhead without degrading accuracy using compression or by proposing communication-efficient alternatives to exchanging model weights as updates between the clients and the server. While FL prioritizes privacy by not sharing users' data, intercepting the updates exchanged between clients and the server can lead to data leakage. Recent attacks have shown that sensitive data can be revealed not only from model gradients but also from model outputs. To ensure a strong privacy guarantee, additional privacy techniques, such as Differential Privacy (DP), can be employed to quantify and minimize privacy loss by modifying the deep model training pipeline. In a central deep learning setting, the Differentially Private Stochastic Gradient Descent (DP-SGD) was proposed to provide sample-level protection based on the Gaussian Mechanism (GM), which we discuss later in this paper.

Statistical heterogeneity arises in scenarios where Devices generate data in a non-identically distributed manner across the network. Moreover, the number of data points across devices may vary significantly, which is not aligned with distributed learning assumptions. System heterogeneity is another challenge in FL, which refers to the variations in local deep models' architecture used by different devices participating in the same FL process. Low-end devices with limited constraints are often used, and it may be necessary to use different model architectures on different devices. For example, a Raspberry Pi collaborating with an Nvidia Nano board would require different model architectures due to the latter device's GPU and list of optimized models that are too big to efficiently run on the former device. Traditional FL algorithms require all devices collaborating in a task to have the same model architecture, which can limit the use of optimized models. To address this issue, recent FL algorithms employ techniques such as Knowledge Distillation (KD) to enable FL between independently designed models.

II. Background

In this section, we provide a brief overview of recent work in the FL domain. Specifically, we focus on studies that address communication cost, heterogeneity, and privacy challenges in FL.

A. Communication Overhead in Federated Learning

Unlike in the central training setting, where clients send raw data to be processed at the server, Federated Learning (FL) is relatively computationally and communication-efficient as it requires only sending model updates. However, communication still presents a critical bottleneck in federated networks as communication in the network can be slower than local computation [5]. Recent works aim to improve FL algorithms' adaptability to various computation-communication tradeoffs by reducing the number of global rounds and reducing the size of the exchanged updates.

Many solutions have been proposed to increase the communication efficiency of FL using gradient compression to reduce the message size. The authors in Deep Gradient Compression (DGC) [6] used the magnitude of gradients as an indicator of importance; the algorithm only transmits gradients that are larger than a predefined threshold. Lin et al. [6] also applied gradient clipping as a defense against the gradient explosion problem by setting a threshold on the gradients' L2-norms. Additionally, momentum factor masking and warmup training are used to avoid the staleness problem caused by reduced communication.

Knowledge Distillation (KD) is a technique originally developed for model compression and knowledge transfer in the central deep learning training setting [7], [8] to train one model (called the student) using the predictions generated by a trained model (called the teacher). KD can be used for compression to compress the knowledge of a larger teacher to a student model of a smaller size [9]. KD has also been repurposed in the context of distributed ML and FL by sharing a proxy dataset among clients used to transfer knowledge learned during local training. KD-based FL methods address both the communication and system heterogeneity concerns in FL [10], [11], [12]. In model-based FL (e.g., FedAvg [3], [13]), clients download model weights, perform local training, and upload the updated weights to the server to be aggregated. Since many applications use models with millions of parameters, sharing models can be a communication bottleneck. KD-based FL addresses the communication issue by sharing soft labels instead of model weights. Soft labels are the local models' predictions over a shared dataset. After that, all predictions are aggregated and broadcasted to clients to be used as labels.

B. Heterogeneity in Federated Learning

Heterogeneity in FL refers to the differences that exist between the local datasets owned by individual devices or clients in a federated learning network. These differences can arise due to various factors, such as variations in data collection procedures, hardware capabilities, and network connectivity. The presence of heterogeneity in federated learning can pose significant challenges in terms of model accuracy, convergence speed, and overall performance. There are two main types of heterogeneity in federated learning: statistical heterogeneity and system heterogeneity [5].

Statistical heterogeneity refers to variations in data distribution and characteristics among the local datasets, such as differences in class balance or data quality. In the context of human activity recognition, where users engage in diverse activities, tackling statistical heterogeneity is particularly challenging due to the presence of diverse data distributions. Ouyang et al. [14] proposed a solution to tackle this issue in their paper. They introduced ClusterFL, a similarity-aware FL system that can achieve high system accuracy and low communication overhead, to address the challenges of balancing local personalization and learning from other nodes. The main idea is to exploit the similarity of users' data to simultaneously enhance the model accuracy and communication efficiency of FL. They proposed a clustered multi-task FL formulation problem that captures the intrinsic cluster structure among users' data. The problem aims to minimize the empirical error of activity recognition across all nodes while regularizing the model weights to prevent over-fitting and minimize the overall intracluster distances of model weights. Specifically, the problem is formulated as:

$$\min_F \sum_{i=1}^{M} \frac{1}{N_i} \sum_{r=1}^{N_i} l(w_i^T x_i^r, y_i^r) + \alpha\, \mathrm{tr}(WW^T) - \beta\, \mathrm{tr}(F^T W W^T F) \tag{1}$$

where M is the number of nodes, N_i is the number of training data samples in node i, w_i is the weight vector of the local model, x_i^r and y_i^r are the input and label of the r-th training pair of the i-th node, F is the cluster indicator matrix, and α and β are hyperparameters. The formulation problem consists of the empirical error, the $L2$-norm regularization of model weights, and the K-means clustering to minimize the overall intracluster distances of model weights. Cluster-wise straggler dropout based on the learned cluster structure and correlation-based node selection proves to reduce communication overhead.

The proposed architecture of ClusterFL [14], depicted in Fig. 3, comprises the following steps: 1) nodes transmit their current model weights and training loss updates to the server, 2) the server employs a cluster indicator matrix to quantify the relationship of the model weights and group nodes into clusters with joint data distributions, 3) the server updates the collaborative learning variables based on the learned cluster indicator matrix, discards slower converging stragglers, and removes unrelated nodes within each cluster to minimize communication overhead, 4) the server transmits the learned cluster indicator matrix, collaborative learning variables, and a drop indicator back to

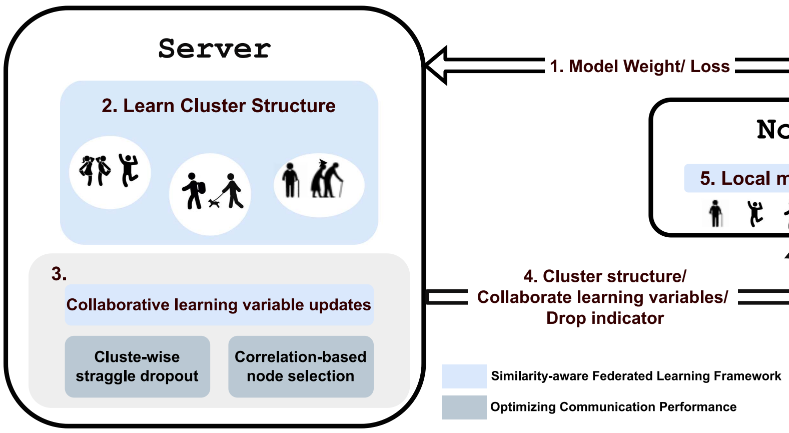

Server
2. Learn Cluster Structure
3.
Collaborative learning variable updates
Cluste-wise straggle dropout
Correlation-based node selection
1. Model Weight/ Loss
Nodes
5. Local model updates
4. Cluster structure/ Collaborate learning variables/ Drop indicator
Similarity-aware Federated Learning Framework
Optimizing Communication Performance

Fig. 3. ClusterFL [14] employs a cluster indicator matrix to group similar nodes. Communication efficiency is improved by discarding stragglers.

each node, and 5) each node updates its model using the received information and determine whether to proceed with the learning process in the subsequent round based on the drop indicator.

System heterogeneity, on the other hand, refers to variations in hardware capabilities, network connectivity, and power among clients, leading to differences in processing power and communication speeds, resulting in only a small fraction of devices being active at once. Consequently, FL algorithms must address challenges related to mitigating stragglers and fault tolerance while being able to anticipate low participation, handle hardware diversity, and cope with dropped devices. Active sampling is a potential approach to this challenge, where participating devices are actively selected based on their system resources or data quality. Nishio et al. [15] delved into this approach by examining innovative device sampling policies that leverage system resources and presenting a new protocol for the efficient implementation of FL in mobile edge computing (MEC) environments. The study highlights that incorporating heterogeneous devices into the FL process without considering their diversity can lead to suboptimal training outcomes. To address this, they introduced a new protocol, named FedCS, which actively manages the resources of heterogeneous clients, thereby enabling the efficient execution of FL.

The main concept of FedCS, depicted in Fig. 4, involves a novel approach to client selection. Rather than choosing clients randomly, the authors introduce a two-step process. In the Resource Request step, random clients are requested to share their resource details, such as wireless channel conditions, computational capabilities, and data size for the current training task, with the MEC operator. The operator then uses this information in the Client Selection step to estimate the time needed for subsequent tasks and determine which clients should participate in those steps. The goal is to maximize the selection of clients that can contribute effectively to the training process. The authors formulate the Client Selection as a maximization problem with respect to a set of selected clients, denoted as S. The objective is to maximize the cardinality of S (i.e., the number of selected clients) subject to a constraint that ensures the total time required for both the Distribution and Scheduled Update step and the Upload step is ensured, along with other factors like client

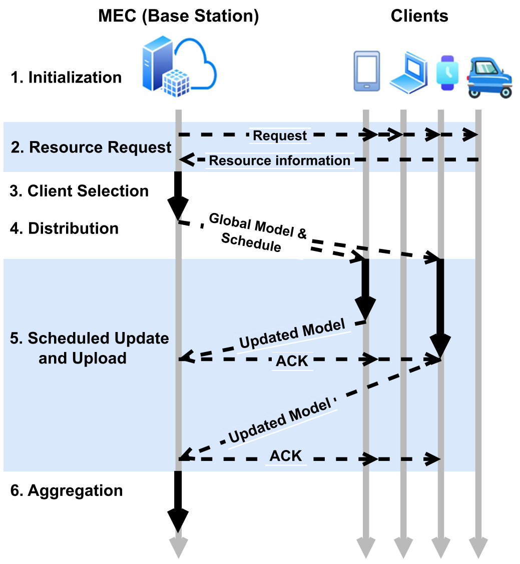


Fig. 4. FedCS framework. Solid black lines denote computation processes, while dashed lines indicate wireless communications.

selection time and aggregation time, does not exceed the round deadline. The formulated problem is represented as follows:

$$\max_{S} |S| \quad \text{s.t.} \quad T_{\text{round}} \geq T_{cs} + T_S^d + \Theta_{|S|} + T_{\text{agg}} \tag{2}$$

Here, S represents the set of selected clients, T_{round} is the deadline for each round, T_{cs} is the time for client selection, T_S^d is the time for distribution, $\Theta_{|S|}$ is the elapsed time of scheduled update and upload, and T_{agg} is the time for aggregation. This maximization problem is solved by a proposed heuristic algorithm based on the greedy algorithm with a knapsack constraint.

In the Distribution step, a global model is shared with the selected clients using a multicast approach. This approach is bandwidth-efficient, allowing simultaneous transmission of the same content to multiple clients. In the Scheduled Update and Upload step, the selected clients perform updates to the model in parallel and upload the updated parameters to the server using the resource blocks allocated by the operator. The server then aggregates the client updates and evaluates the performance of the model. These steps are iterated until the model achieves a desired performance level or until the final deadline arrives. This iterative process allows for collaborative learning and continuous improvement of the model through the contributions of selected clients.

Additionally, Kang et al. [16] proposed an approach to encourage mobile devices with high-quality data to engage in fl. They focused on a universal mobile network that comprises wireless communication infrastructures and a collection of mobile devices. The widely distributed communication infrastructures, e.g., base stations, are serving as task publishers for FL tasks, while mobile devices are data owners for the learning tasks. They have identified information asymmetry issues between the task publisher and mobile devices, such as a lack of knowledge about available computation resources, data sizes, and local data quality. This information asymmetry can lead to high costs for the task publisher while providing incentives to mobile devices. To address this challenge, they introduced a solution based on contract theory that maps the contributed resources into suitable rewards.

The authors formulated the problem through the development of two models. Firstly, a computation model was developed for FL, treating it as a monopoly market with a monopolist operator (task publisher) and a set of mobile devices. The authors analyzed the computation resources required for local model training, measuring CPU cycle frequency from the data owner. They also considered the computation time and energy consumption required for local model training to determine the contribution of each data owner toward the local model training process. Secondly, a communication model was developed to enable all participating data owners to collaborate in training a global model. This was achieved by an iterative method with several communication rounds (i.e., global iterations). The authors used contract theory, a design mechanism that encourages data owners to participate in the training process by rewarding them for their contributions. The mechanism involves a profit function for the task publisher and a reward function for the data owners. The task publisher offers a contract $(R_n(f_n), f_n)$ with bundles to various data owners who possess different computation resources, where f_n represents the computation resource of type n and $R_n(f_n)$ is the corresponding reward. The more computation resources a data owner possesses, the faster the training and the higher the reward. In the context of FL tasks, the primary goal of the task publisher is to maximize its profit, which is expressed as follows:

$$\max_{(R_n,f_n)} U_{TP} = \sum_{n=1}^{N} N p_n w \ln\left[T_{\max} - \left(\frac{\sigma_B}{B\ln(1+\frac{p_n h_n}{N_0})} + \frac{\psi}{\theta_n}\cdot\frac{c_n s_n}{f_n}\right)\right] - lR_n \tag{3}$$

On the other hand, data owners aim to maximize their utility, which can be expressed as:

$$\max_{(R_n,f_n)} U_D = R_n - \mu\left(\frac{\psi}{\theta_n}\zeta c_n s_n f_n^2 + \frac{\sigma\rho_n}{B\ln(1+\frac{\rho_n h_n}{N_0})}\right) \tag{4}$$

Here, R_n represents the reward for data owner n, and μ is a parameter that determines the importance of the computation resource and the communication resource in the data owner's utility. The first term in the utility function represents the reward, while the second term represents the cost of resources. ζ, θ_n, c_n, s_n, and ρ_n represent the computation and communication efficiency, the computation and communication sizes, and the channel gain of data owner n, respectively. To ensure the feasibility of contract design in FL tasks, two constraints are considered: Individual Rationality, which ensures that each data owner participates only if its utility is non-negative, and Incentive Compatibility, which maximizes utility by ensuring that each data owner chooses the contract designed for itself. Data owners choose and sign contracts to finish the task, but failure or misbehavior leads to blacklisting and no payment.

C. Attacks Against Federated Learning

There are many types of attacks in federated networks that can be conducted by insiders (malicious clients or servers) or outsiders (eavesdroppers). Two of the widely researched attacks are poisoning and inference attacks. Poisoning attacks can be random, where the adversary tries to reduce the accuracy of the model, or target, where the attacker's aim is to induce the model to predict a particular output. At a high level, both forms of poisoning attacks strive to alter the behavior of the target model in an unfavorable manner. Data poisoning can be categorized into clean data poisoning [17], where the attacker can't change data labels, and dirty-label poisoning, where the adversary changes the label of the training data to alter the behavior of the model. Label-flipping [18] is one instance of dirty-label poisoning attacks. Previous works have shown that private training data can be revealed from the exchanged gradients [19], [20].

Inference attacks refer to attacks that aim to exploit the memorization aspect of the deep learning model. Attackers in inference attacks aim to infer sensitive information about the training data. For example, in Membership Inference Attack (MIA), given a data point, the attacker aims to determine if it was used to train the model [21]. For instance, an attacker may deduce whether a particular patient profile was used in training a disease-related classifier or infer whether devices used a given text by inspecting the embeddings layer [22].

Numerous defense mechanisms have been proposed to counteract these attacks. Differential Privacy (DP) is a privacy analysis framework that adjusts the Stochastic Gradient Descent (SGD) procedure by incorporating noise to minimize data leakage from the exchanged gradients [23], [24]. Experiments were conducted to use gradient compression to defend against privacy attacks. Previous work [6], [25] proves that exchanged gradients can be reduced by over 300x using sparsification techniques, which suggests that gradient compression can be a good practical approach to avoid data leakage. We also discuss Differential Privacy (DP) as a privacy-preserving mechanism that can be integrated with FL algorithms to allow clients to inject carefully calibrated noise into their gradients to protect against privacy attacks, like MIA, while maintaining accuracy.

III. Preliminaries

A. Federated Learning

Federated Learning (FL) is a distributed machine learning paradigm designed for the decentralized training of models on distributed datasets across a network of clients. The clients, denoted by $\mathbf{C} = C_1, C_2, \ldots, C_{N_c}$, each client C_k possess a local private dataset D_k and a local model f_k with weights θ_k. The goal of federated learning is to train the local models f_k on the private datasets D_i without sharing the data explicitly. This is achieved by initializing a global model f with a set of weights θ_0. Through iterative rounds, the global model is continually updated by aggregating the local models' updates. The models are trained in a distributed manner to push the global model performance beyond local effort and approach central training on the combined private dataset $D = \sum_{k=1}^{N_c}\{D_k\}$. In order to aggregate clients' models, a global architecture f is imposed on all clients. Knowledge Distillation-based FL offers an alternative model-agnostic approach in which all clients share a public dataset D_p on which they calculate local Soft Labels (SL_k) [26], [27]. These local Soft Labels are then transmitted to the server to be aggregated into SL^r, which is broadcasted to all clients to train on it in addition to the local training. This approach allows each client to design its own model architecture f_k independently to suit its resources and hardware capacity. Additionally, the communication overhead incurred by sharing soft labels is often less than that in Standard FL (e.g. FedAvg). Table I lists all the symbols used in this article with their corresponding descriptions.

B. Knowledge Distillation

In Knowledge Distillation (KD), a trained teacher model generates soft labels that are used to train a student model on an unlabeled dataset in the centralized setting. In the federated learning setting, KD is applied by utilizing a proxy dataset D_p shared among all clients to calculate their soft labels. These local soft labels are then sent to the server to aggregate into global soft labels, which are subsequently transmitted back to the clients to train their models on the labeled dataset (D_p, S^r). This process helps improve the performance of the federated learning models while maintaining communication efficiency. In the federated learning setting with Knowledge Distillation, each client has their own private dataset D_i, a shared public dataset D_p, and an independently designed model f_i. The knowledge transfer happens through the shared dataset D_p.

C. Differential Privacy

Differential Privacy (DP) [28] is a mathematical framework dedicated to quantifying and managing the privacy risks associated with the dissemination of statistical data. DP provides strong privacy guarantees by ensuring that the addition or removal of a single data point in a dataset does not significantly impact the output of a statistical analysis.

Formally defined, we delineate a randomized mechanism $M : X \to R$ that possesses a domain X and range R. M satisfies (ϵ, δ)-differential privacy for all sets $S \subseteq R$ and adjacent datasets $D, D' \subseteq X$ if:

$$\Pr[M(D) \in S] \leq e^{\epsilon} \Pr[M(D') \in S] + \delta \tag{5}$$

The privacy budget ϵ controls the amount of privacy leakage, while δ is a small positive value that controls the probability of deviation from the guarantee. The lower the values of both ϵ and δ, the stronger the privacy guarantee.

A widely used mechanism for applying differential privacy is the Gaussian Mechanism (GM). The GM is designed to approximate a function s while protecting the privacy of the

TABLE I
Mathematical Terms and Notations

Symbol	Description
TFT	Total Federated Learning Time
FL	Federated Learning
η	Learning rate
B	FL update size
N_t	Number of time slots for FL
Δ	duration of a time slot (in seconds)
$T_{k,UL}$	Time of FL Upload phase for user k
$T_{k,LD}$	Time of Local Distillation phase for user k
$T_{k,LT}$	Time of Local Training for user k
$T_{k,DL}$	Time of FL Download phase for user k
$N_{k,UL}$	Number of time slots of FL Upload phase for user k
$N_{k,LD}$	Number of time slots of Local Distillation phase for user k
$N_{k,LT}$	Number of time slots of Local Training for user k
$N_{k,DL}$	Number of time slots of FL Download phase for user k
N_r	Number of FL rounds
N_c	Number of FL clients
$\mathbf{C}$	The set of all FL clients
D_p	Public dataset used for KD
D_k	Private dataset at user k
θ^r	Global weights at round r
θ_k^r	Local weights of user k at round r before local training
$\theta_k^{r'}$	Trained weights for client k at round r after local training
f_k	Model architecture of client k
g_k	Model of client k without the Softmax layer
$\mathcal{L}_{\mathcal{CCE}}$	Categorical Cross Entropy loss
$\mathcal{L}_{\mathcal{KL}}$	Kullback-Leibler distance loss
$\nabla\mathcal{L}_{\mathcal{CCE}}$	Derivative of the Categorical Cross Entropy loss
DP	Differential Privacy
KD	Knowledge Distillation
R_d	Data rate
k	Fraction of clients participating in FL at any round
P_k	Transmission Power for the k^{th} antenna
N_a	Number of server antennas
P_t	Total transmission power
$\mathbf{p}_k$	Precoder of signal s_k
$\mathbf{D}$	The power matrix
$\mathbf{h}_k$	Channel response of user k
$\mathbf{x}_k$	Transmitted signal to user k
$\mathbf{y}_k$	Received signal at user k
n_k	Additive Gaussian noise
σ_k^2	Variance of noise at user k
γ_k	SINR of the received signal at user k
s_k	Encoded FL message to user k
D_t	Test dataset
$\mathcal{N}(0, \sigma^2)$	Gaussian noise with variance σ^2
E	Number of epochs for model-based FL
E_{LT}	Number of epochs for local training (for KD-based FL)
E_{LD}	Number of epochs for local distillation (for KD-based FL)
ϵ	Differential Privacy protection level
δ	Differential Privacy guarantee failure probability
SL^r	Aggregated soft labels at round r
SL_k^r	Local soft labels of client k at round r
$\|D_k\|$	Size of the private dataset of client k

samples in D by adding noise that is calibrated to the function's sensitivity Δs of s given by

$$\Delta s = \max_{D,D'} ||s(D) - s(D')|| \tag{6}$$

As shown in [28], GM can approximate a function s to satisfy (ϵ, δ)-differential privacy by injecting calibrated Gaussian noise $n \sim N(0, \sigma^2)$, onto the function s. The noise standard deviation is given by $\sigma \geq c\Delta s/\epsilon$ and the constant $c \geq \sqrt{2\ln(1.25/\delta)}$.

D. Relationship Between DP and KD in the Context of FL

In heterogeneous FL environments where each participant client possesses unique hardware properties and requirements, imposing a unified model architecture does not utilize system resources effectively. For example, the standard FL algorithm (FedAvg) will not use a large model architecture if it can not be hosted on any of the clients' devices that have smaller memory. The requirement of a unified architecture across clients in standard FL is necessary due to the adopted architecture-dependent aggregation scheme, which is model averaging in FedAvg and gradient averaging in FedSGD [3].

KD-based FL provides an alternative to architecture-dependent aggregation that relies on the teacher-student learning scheme, which is used in central deep learning training to augment training to improve performance [7] or to compress the knowledge learned by a larger model into a smaller model [9]. The performance yielded by sharing and aggregating soft labels, which are predictions calculated by the clients on a shared public dataset, is often inferior to that achieved using architecture-dependent aggregation methods. KD-based FL is more flexible in the sense that it allows each client device to tailor its model architecture to fit its software and hardware constraints and requirements. For example, a GPU-enabled device could benefit from GPU-optimized models provided by deep learning frameworks while collaborating with another CPU-only device that uses a different model architecture. These two different devices can participate in the same FL process under KD-based by sharing predictions calculated on a shared public/proxy dataset.

Both model and gradient sharing FL methods pose privacy risks [29], KD-based FL methods with recursive model exchange are also susceptible to privacy leakage [30]. To address privacy concerns in the central Knowledge Distillation (KD) scenario, PATE uses a non-overlapping proxy dataset and restricts access to the teacher's knowledge [31].

In this work, we employ local DP along with KD-based FL, combining flexibility and data privacy. Each client independently designs its model architecture: It follows general guidelines regarding the task (e.g. data point shape, number of classes) but controls details related to the architecture type (e.g. Multi-Layer Perceptron (MLP), Long Short Term Memory (LSTM), and Convolutional Neural Network (CNN)) and the architecture size (e.g. number of hidden layers, number of neurons in each layer, etc). This ensures each client's efficient resource utilization. During training, the amount of noise needed to achieve a predefined DP protection level - parameterized by ϵ and δ - is calculated locally. Before the noise is added, gradients are clipped to limit the contribution per sample. The noise is added on the client side, and the model is updated. Unlike the parameter-based FL methods, where clients send the updated models to the server to be aggregated and broadcasted back to clients, we assume a public dataset is shared across clients on which local soft labels are calculated. Local soft labels are then sent with the server to be aggregated into global soft labels which are broadcasted back to clients. Finally, clients perform distillation training, which utilizes a distance loss function such as Mean Squared Error (MSE) to minimize the distance between the local soft labels produced by each of the clients and the downloaded global soft labels.

IV. System Model

In this section, we first describe the wireless system model upon which we aim to design a privacy-preserving federated learning model. Then, we describe the preliminaries of model-based and knowledge-distillation-based federated learning frameworks, respectively.

A. System Model of Bandwidth-Constrained Communication Networks

We consider a wireless network comprising one BS serving as a server that coordinates federated learning among K users. In this section, we present the signal model of the download phase, and the same model can be extended to the upload phase of data transmission. In the download phase, the server downloads the global FL update. In the case of model-based FL, the FL update is the global model weights, denoted as θ^r. On the other hand, Knowledge Distillation-based FL methods download the global soft labels, denoted as SL^r, where r is the current FL round. In the following sections, we provide the traffic model for both paradigms. The FL update for the kth user C_k is encoded into s_k. The overall message to be transmitted by the server is given by

$$\mathbf{s^r} = \begin{bmatrix} s_1^r \\ \vdots \\ s_K^r \end{bmatrix}^T. \tag{7}$$

The overall message is linearly precoded via a precoder

$$\mathbf{P} = \begin{bmatrix} p_1 \\ \vdots \\ p_K \end{bmatrix}, \tag{8}$$

where $\mathbf{P} \in \mathbb{C}^{N_a \times K}$, $\|p_k\|^2 = 1$, $k = 1, 2, \ldots, K$.

The transmitted signal vector is given by

$$\mathbf{x} = \mathbf{PDs} = \sum_{k=1}^{K} \sqrt{P_k} p_k s_k, \tag{9}$$

where the power matrix is denoted as $\mathbf{D} = \mathrm{diag}(\sqrt{P_1}, \ldots, \sqrt{P_K})$. The total transmit power is written as $P_t = \sum_{k=1}^{K} P_k$. Specifically, the received signal at C_k is given by

$$y_k = \mathbf{h}_k^T \mathbf{x} + n_k, \quad k = \{1, 2, \ldots, K\}, \tag{10}$$

where $h_k \in \mathbb{C}^{N_a \times 1}, k = 1, 2, \ldots, K$ is the channel response between the server and client C_k. We assume i.i.d. additive Gaussian noise with variance σ^2 at the receivers. The SINR of decoding the private message s_k at C_k is given by:

$$\gamma_k = \frac{P_k|\mathbf{h}_k^H \mathbf{p}_k|^2}{\sum_{\substack{j=1 \\ j\neq k}}^{K} P_j|\mathbf{h}_k^H \mathbf{p}_j|^2 + \sigma^2}. \quad (11)$$

The data rate of the transmission of the private message s_k is given by

$$R_k = log_2(1 + \gamma_k). \quad (12)$$

B. Model-Based Federated Learning Model

The FL process [3], [32] is conducted over several rounds/iterations. In each round, selected clients perform three operations: Downloading the global model (DL), Learning (L), and Uploading the locally trained model for aggregation (UL).

Let $T_{k,UL}$ and $T_{k,DL}$ denote the time the kth user takes to complete the upload and download phases, respectively, in any round. We divide these time durations into $N_{k,UL}$ and $N_{k,DL}$ time slots, respectively, where each time slot duration is δ. Therefore $T_{k,UL}$ and $T_{k,DL}$ are given by:

$$T_{k,UL} = N_{k,\mathrm{DL}} \cdot \Delta \quad \forall k \in K \quad , \quad (13)$$

$$T_{k,DL} = N_{k,\mathrm{UL}} \cdot \Delta \quad \forall k \in K \quad . \quad (14)$$

During the download/upload phases, each client downloads/uploads its packets as:

$$s_k = \sum_{i=0}^{N_{k,\mathrm{DL}}} \Delta \cdot R_k \quad \forall k \in K \quad , \quad (15)$$

$$s_k = \sum_{i=0}^{N_{k,\mathrm{UL}}} \Delta \cdot R_k \quad \forall k \in K \quad , \quad (16)$$

respectively.

The Total Federated Learning Time (TFT) is characterized by the following equation:

$$TFT = N_r(T_{DL} + T_L + T_{UL}), \quad (17)$$

where N_r is the number of rounds. T_{DL}, T_L, and T_{UL} are the maximum times across clients for downloading, learning, and uploading phases, respectively, given by:

$$T_{DL} = \min(T_{1,DL}, T_{2,DL}, \ldots, T_{k,DL}), \quad (18)$$

$$T_L = \min(T_{1,L}, T_{1,L}, \ldots, T_{1,L}), \quad (19)$$

$$T_{UL} = \min(T_{1,UL}, T_{2,UL}, \ldots, T_{k,UL}). \quad (20)$$

In model-based FL, $T_L = T_{LT}$ since each client should perform only local training. In KD-based FL, $T_L = T_{LT} + T_{DL}$ since each client performs local training and local distillation. This also means that while KD-based FL is significantly communication efficient compared to model-based FL, the computational resources and time required for KD-based FL are more than their counterpart that are required for model-based FL as KD-based FL performs more epochs per round. where $N_{k,\mathrm{DL}}$, $N_{k,\mathrm{UL}}$, and $N_{k,\mathrm{L}}$ are the time slots for downlink, uplink, and learning phases, respectively. $R_{i,k}$ and Δ are the instantaneous data rate for time slot i of the node k and the duration of a single time slot, respectively.

During the DL phase, nodes download the weights/parameters, θ^r, of the server-controlled global model f, where r represents the round. In the L phase, clients perform local training where they update their weights using:

$$\theta_k^r \leftarrow \theta_k^{r-1} - \eta \frac{1}{|D_k|} \sum_{(x,y)\in D_k} \nabla \mathcal{L}_{\mathcal{CCE}}(f(\theta_k^{r-1}, x), y). \quad (21)$$

Here, η is the learning rate and $\mathcal{L}_{\mathcal{CCE}}$ denotes the Categorical Cross Entropy (CCE) loss function.

Finally, in the UL phase, clients sent their locally trained weights to the server to be aggregated as:

$$\theta^{r+1} \leftarrow \frac{1}{N_k} \sum_{k=1}^{N_k} \theta_k^r. \quad (22)$$

This process continues until convergence is reached or a predetermined number of rounds have elapsed.

The uploading and downloading of model weights incur large communication costs and network bandwidth. In collaborative learning applications, bandwidth is often a concern, therefore we discuss Knowledge Distillation-based FL which proposes to use soft labels instead of model weights/gradients for communication.

C. Knowledge Distillation-Based Federated Learning Model

As Knowledge Distillation was first proposed for the central training setting [7], [8], modifications to this technique were proposed to adapt KD for the Federated Learning (FL) scenario [10], [11]. The authors in [10] employed a proxy dataset D_p, which is accessible to all clients for soft labels. Each client calculates his soft labels SL using the locally trained model on the shared dataset. Subsequently, the local soft labels are transmitted to a server for aggregation into global soft labels, which are then returned to the clients for training on the labeled dataset (D_p, SL^r).

As shown in Fig. 5, one notable advantage of KD-based FL in comparison to model-based FL is that the former addresses the system heterogeneity concern for clients in FL as it gives each client the freedom to customize his own local model architecture. On the other hand, FedaAvg [3] imposes a server-controlled architecture to enable model aggregation.

Each global round in KD-based FL includes four phases on the client side: Local training phase (L), Upload Soft Labels phase (USL), Download Soft Labels (DSL), and Knowledge Distillation Learning phase (KDL).

Similar to model-based FL, the L phase in KD-based FL includes training the locally designed model f_k on the private local dataset using Cross Entropy Loss (CCE) as:

$$\theta_k^{r'} \leftarrow \theta_k^r - \eta \cdot \frac{1}{|D_k|} \sum_{(x,y)\in D_k} \nabla \mathcal{L}_{CCE}(\theta f_k, f_k(x), y). \quad (23)$$

After the L phase, each client calculates his local soft labels on the shared dataset as $SL_k = g_k(D_p)$, where g_k is a model

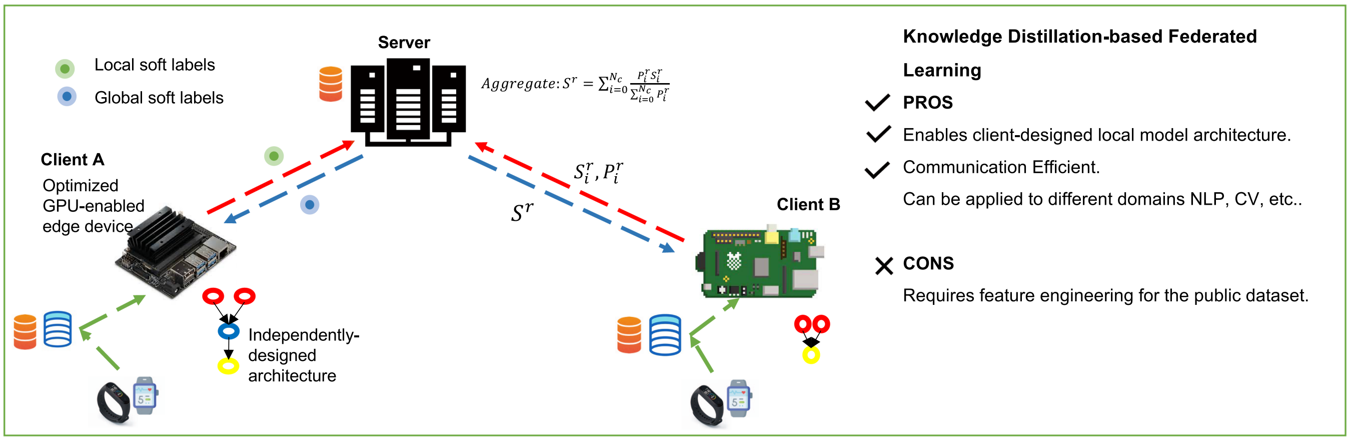

Local soft labels
Global soft labels
Server
$Aggregate: S^r = \sum_{i=0}^{N_c} \frac{P_i^r S_i^r}{\sum_{i=0}^{N_c} P_i^r}$
Client A
Optimized GPU-enabled edge device
S_i^r, P_i^r
S^r
Client B
Independently-designed architecture
Knowledge Distillation-based Federated Learning
PROS
Enables client-designed local model architecture.
Communication Efficient.
Can be applied to different domains NLP, CV, etc..
CONS
Requires feature engineering for the public dataset.

Fig. 5. Knowledge distillation-based federated learning incurs significantly less communication overhead, leveraging a shared dataset for knowledge distillation.

obtained from f_k by removal of the final Softmax layer [10]. Another approach to build $g_k(x)$ is to use a temperature variable to smooth the probability outputs produced by the Softmax layer [8]. The USL phase involves uploading soft labels from clients to the server. After that, the server aggregates these soft labels. Finally, in the DSL phase, global soft labels are downloaded to clients.

$$SL^r \leftarrow \sum_{i=1}^{N_k} \frac{P_i^r \cdot SL_k^r}{\sum_{k=0}^{N_k} P_k^r}. \tag{24}$$

The global soft labels reflect the aggregated knowledge learned by all clients on their distributed datasets. During the KDL phase, distance functions such as Mean Squared Error (MSE) or Kullback-Leibler divergence loss (KL) are employed to train f_i on the public unlabeled dataset, utilizing the global soft labels.

$$\theta^{r+1} \leftarrow \theta_k^{r'} - \eta \cdot \frac{1}{|D_p|} \sum_{(x,y)\in(D_p,SL^r)} \nabla\mathcal{L}_{KL}(\theta_k^{r'}, g_k(x), y). \tag{25}$$

Here, $\theta_k^{r'}$ and θ_k^{r+1} denote the model weights following local dataset training and model weights post Local Distillation (LD) training, respectively. Since the KD-based FL paradigm is model agnostic, the model weights learned by client k after LD training, θ_k^{r+1}, are also the model weights that client k starts with at the next round r+1.

In the following section, we discuss our envisioned privacy-preserving federated learning by seamlessly integrating knowledge distillation and differential privacy techniques.

V. Envisioned Privacy-Preserving Federated Learning Technique

The authors in [23] proposed an algorithm to protect the privacy of the training data during deep learning training called Differentially Private Stochastic Gradient Descent (DP-SGD). DP-SGD achieves sample-level protection by clipping the gradients generated in the backpropagation cycle of the SGD algorithm. By clipping the gradient of each sample, the contribution of each sample to the output of the model is bounded. After that, a batch of gradients is averaged, and noise is added according to the Gaussian Mechanism (GM). The steps of the DP-SGD algorithm are depicted in Fig. 6.

To apply DP in the context of FL, where there are many clients, each poses private training data that is used to train a local model whose weights or gradients are then shared with the server in rounds as described earlier, one can use one of two approaches:

1) Central Differential Privacy (CDP). CDP provides participant-level protection. This setting assumes an honest server that is trusted by clients to inject noise.
2) Local Differential Privacy (LDP). LDP provides sample-level protection. Each client is responsible for applying DP-SGD in local training to produce a (ϵ, δ)-DP model that guarantees protection to each sample with probability bounded by ϵ.

As seen in Algorithm 1, Local Differential Privacy on FedAvg is an exemplification of leveraging differential privacy in FL, ensuring that the collaborative global model, trained over several rounds of communication, preserves individual data privacy. This algorithm implements Local Differential Privacy (LDP) by integrating noise via the OpacusPrivacyEngine [33], yielding a collaboratively trained global model with a robust privacy protection level denoted by ϵ and δ. Fig. 7 depicts the stages of local differential privacy in model-based FL.

On the other hand, the Local Differential Privacy for KD-based FL, as illustrated in Algorithm 2, extends the application of privacy to Knowledge Distillation (KD)-based FL. This algorithm initializes with independently designed local models, with each client applying differential privacy on their local models. The server subsequently aggregates soft labels received from clients and broadcasts them.

Furthermore, the Differential Privacy Stochastic Gradient Descent (DP-SGD), portrayed in Algorithm 3, serves as a foundational stone for both aforementioned algorithms leveraging gradient clipping and noise addition to the gradients during each epoch of training, ensuring the resultant model is (ϵ, δ)-Differentially Private.

VI. Experiment Setup

In this section, we describe the experiments conducted to compare the performance of central vs federated learning, evaluate

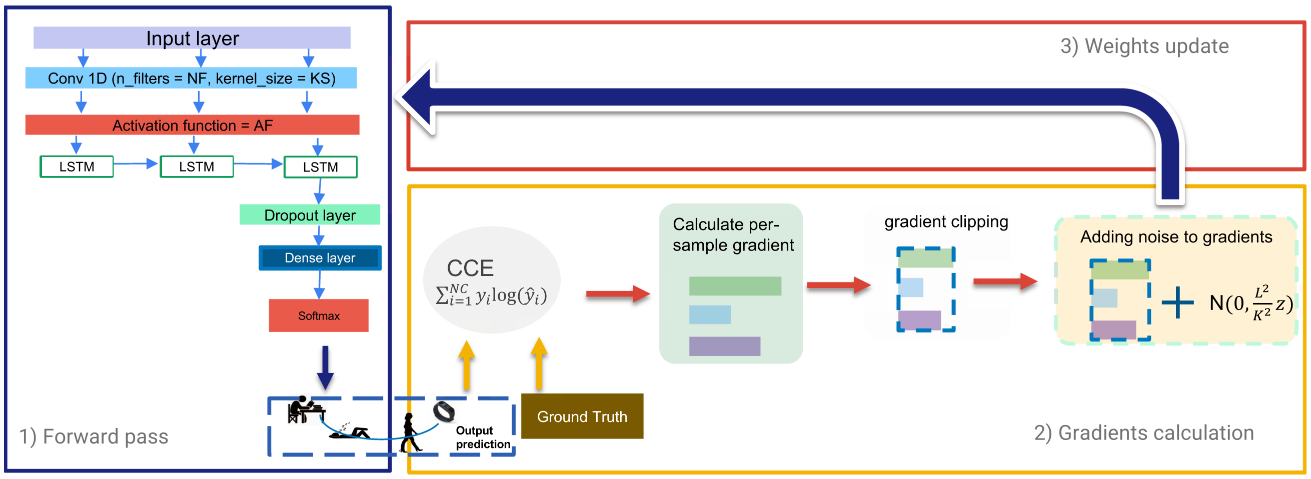


Fig. 6. Differentially private stochastic gradient descent (DP-SGD). While in SGD, gradients are calculated per batch, in DP-SGD, gradients are clipped per sample to limit the contribution of each data point on training. Protection is further enhanced by adding noise which is proportional to the desired protection level (ϵ).

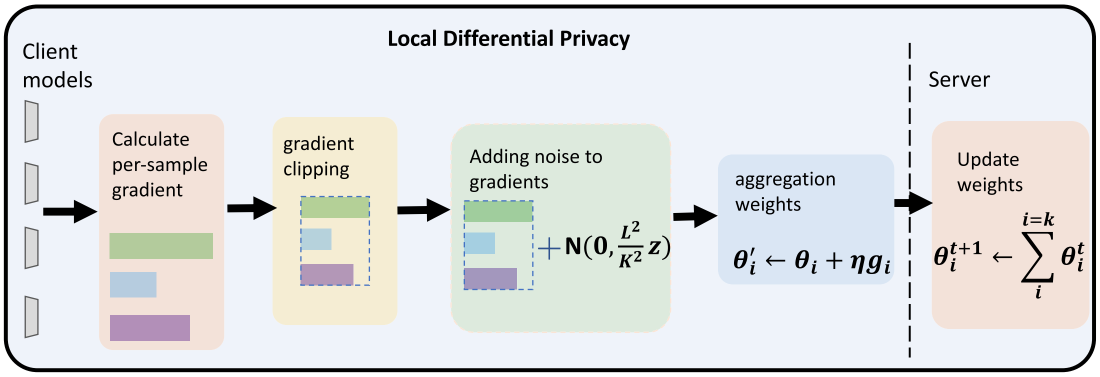

Local Differential Privacy
Client models
Calculate per-sample gradient
gradient clipping
Adding noise to gradients
$+N(0, \frac{L^2}{K^2} z)$
aggregation weights
$\theta_i' \leftarrow \theta_i + \eta g_i$
Server
Update weights
$\theta_i^{t+1} \leftarrow \sum_{i}^{i=k} \theta_i^t$

Fig. 7. In local differential privacy, each client keeps his private data. The client calculates the gradients of the loss function for its private data. The gradients of each data point are clipped to limit its contribution. In this approach, calibrated noise proportional to the target epsilon is added on the client side before the gradients are applied to model weights. The server is only responsible for aggregating client models and downloading the global model to clients.

the utility of Knowledge Distillation (KD)-based FL, and measure the accuracy loss incurred by applying Local Differential Privacy (LDP) to KD-based FL by introducing calibrated noise during each client's local training using DP-SGD.

The first experiment utilizes a public dataset for human authentication based on Wi-Fi sensing: CAUTION [34]. It includes 546 training samples and 294 testing samples. The input size of each data point is $3 \times 114 \times 500$, representing the Channel State Information (CSI) of a Wi-Fi signal collected while a subject was walking between a router and a connected device. CAUTION is a classification dataset that aims to classify the input CSI signal into one of 14 classes corresponding to 14 subjects. We used the CAUTION dataset to compare the performance achieved by local, central, and federated learning. The deep learning architecture employed for the time-series dataset was a Multi-Layer Perceptron (MLP) architecture. To evaluate local performance, we assume there are 20 clients over which the data are distributed and divide the training samples randomly into 20 sets, where each set represents the local private data of one of 20 clients. We train the model using only one of these sets and call the model trained using this approach the locally trained model. Then, we perform model-based federated learning where each client possesses a model trained on his private data for five epochs before all local models are aggregated into one global model. The global model is broadcast to all clients. This process continues for 50 rounds, where each round starts with randomly selecting half the clients (10 out of 20) to perform local training and ends with downloading the aggregated model from the server to all clients. After 50 rounds, the aggregated global model is stored at each client for inference; this model is called the federated learning model. Finally, to evaluate central deep learning, we collect the 20 sets of training data in a single training set and train a central model for 30 epochs, and we call the model trained under this approach the centrally trained model. We use the same model architecture (i.e. MLP) for the three learning methods. Additionally, all the test samples are used to evaluate the test accuracy of the three methods. Fig. 12 shows the test accuracy achieved by each of these three methods. FedAvg achieves a test accuracy of 88% exceeding both the central performance (79%) and the local training performance (42%).

In the second experiment, we used a distributed version of the MNIST dataset as the private data distributed over all clients and FEMNIST as the shared public dataset. In the third experiment,

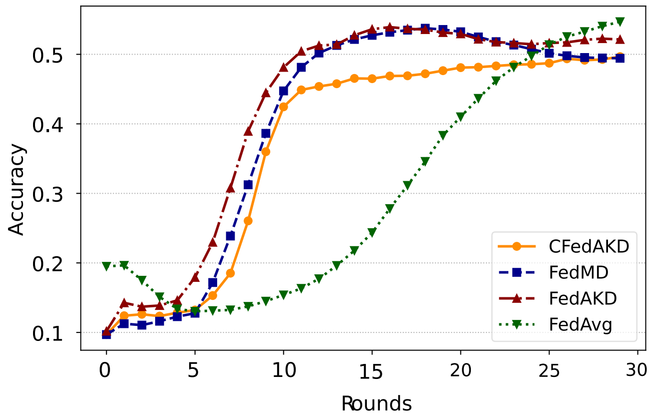


Fig. 8. Average per-round test accuracy achieved by clients on MNIST dataset under the considered federated learning algorithms. Note that the obtained accuracy surpasses local accuracy, proving the effectiveness of federated learning algorithms, though further improvements are possible with hyper-parameter tuning and additional experimentation.

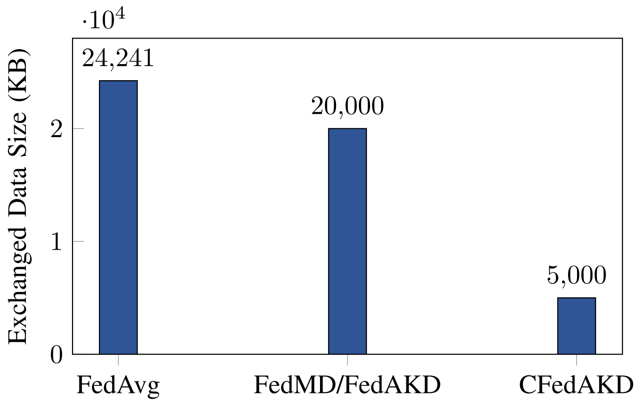


Fig. 9. Communication overhead of different considered FL algorithms on CIFAR100 dataset. We can observe that model-based FL use significantly more bandwidth than KD-based FL, as the former exchanges model weights while the latter exchanges logits calculated over the shared public dataset.

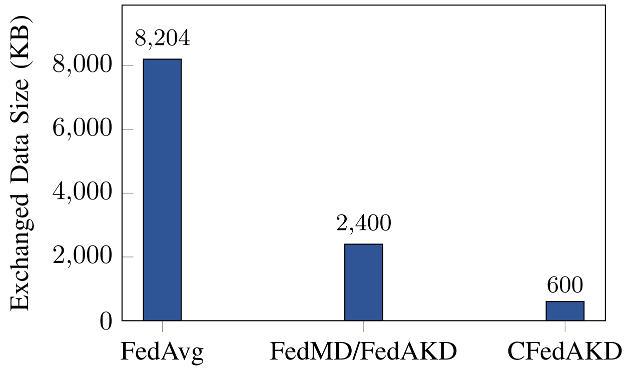


Fig. 10. The communication overhead of various federated learning (FL) algorithms on MNIST dataset. Notably, model-based FL requires substantially more bandwidth compared to KD-based FL. In the former, model weights are exchanged, whereas in the latter, logits calculated over the shared public dataset are exchanged.

Algorithm 1: Local Differential Privacy on FedAvg.

```
1: function MAIN
2:   Input: Client $k$ owns a local dataset: $D_k$, Initial global
     model weights: $\theta_0$, Number of communication rounds:
     $N_r$, Number of epochs: $E$, Total number of
     participating clients: $N_k$, DP parameters $\epsilon$ and $\delta$,
     Gradient Clipping norm $S$
3:   Output: Collaboratively trained global model $\theta^R$
4:   Initialize model $\theta_0$
5:   for round $r = 1, 2, \ldots, N_r$ do
6:     $K^r \leftarrow$ randomly select $K$ participants
7:     for each participant $k \in K^r$ do
8:       $W \leftarrow E \cdot N_r$                ▷ Number of steps
9:       $z \leftarrow OpacusPrivacyEngine(S, W)$▷ noise
         scale
10:      $\theta_k^r \leftarrow$ DP-SGD$(z, S, E)$▷ Clients train in parallel
11:    end for
12:    $\theta^{r+1} \leftarrow \sum_{k \in K^r} \theta_k^r$
13:  end for
14:  return $\theta^R$
15: end function
```

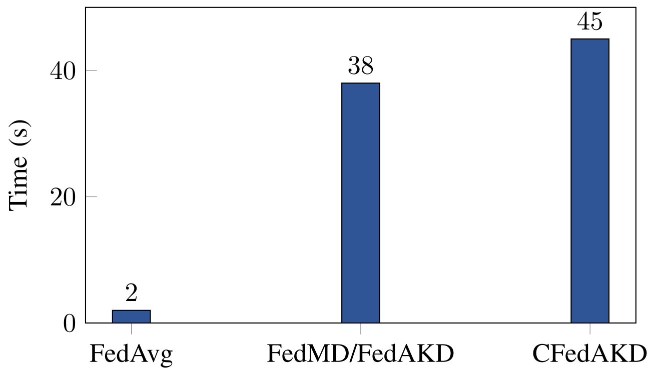


Fig. 11. Average per-round time in seconds for the considered FL algorithms on the CIFAR100 dataset.

we used CIFAR100 as the distributed private dataset and CIFAR10 as the public dataset. The goal of FL is to train a global model on a private dataset distributed over many devices. In KD-based FL, we assume all clients share an additional public dataset whose features are similar to the private dataset.

For the second and third datasets, we considered the following FL algorithms:

1) Federated Learning with model Averaging (FedAvg) [3]. The standard FL algorithm in which a server imposes a model architecture in which all clients download and train on local data. Then, updated model weights are aggregated by the server.
2) Federated Learning with Model Distillation (FedMD) [10]. A KD-based FL algorithm in which each client designs his local model, which he trains on local data. Additionally, clients share a public dataset. After local training, logits are calculated using the local models on the public dataset and shared with the server. The server collects and aggregates the local logits from clients, and

Algorithm 2: Local Differential Privacy for KD-Based FL.

```
Input: Public dataset: $D_p$, Test dataset: $D_t$, Local dataset of client k: $D_k$, Independently designed local model of client k: $f_k$, Number of communication rounds: $N_r$, Number of epochs for local training: $E_{LT}$, Number of epochs for local distillation training: $E_{LD}$, Local training Loss function: $\mathcal{L}_{\mathcal{CCE}}$, Local distillation loss function: $\mathcal{L}_{\mathcal{CCE}}$, Total number of participating clients: $N_k$
Output: Collaboratively trained local model $f_k$
$C_k$ designs $f_k$ and randomly initializes it weights $\theta_k$
for round $r = 1, 2, \ldots N_r$ do
  Client:
  for client $k = 1$ to $N_c$ do
    $W \leftarrow E_{LT} \cdot N_r$ ▷ Number of steps
    $z \leftarrow OpacusPrivacyEngine(S, W)$ ▷ noise scale
    $SL_k^r \leftarrow g_k(D_p)$
    $P_k^r \leftarrow$ calculate accuracy on $D_t$
    $C_k$ sends $SL_k^r$ and $P_k^r$ to the Server
    $\theta_k^{r'} \leftarrow$ DP-SGD$(z, S, E)$
    for epoch $e = 1$ to $E_{KD}$ do
      $\theta_k^{r+1} \leftarrow \theta_k^{r'} - \eta \frac{1}{|D_p|} \sum_{(x,y)\in(D_p,SL^r)} \nabla \mathcal{L}_{LD}(\theta_k^{r'}, g_k(x), y)$
    end for
  end for
end for
Server:
$SL^r \leftarrow \sum_{k=1}^{N_k} \frac{P_k^r SL_k^r}{\sum_{k=0}^{N_k} P_k^r}$
Broadcast $SL^r$
```

Algorithm 3: Differential Privacy Stochastic Gradient Descent (DP-SGD).

```
1: function DP-SGD$(z, S, E)$
2:   $\sigma \leftarrow z \cdot \frac{S}{q}$
3:   Initialize model $f$ with weights $\theta_0$
4:   for epoch $i$ from 1 to $E$ do
5:     $b_k \leftarrow$ randomly select a batch from dataset $D_k$ with probability q
6:     for $(x, y) \in b_k$ do
7:       $g_i = \nabla_\theta L(\theta_i, (x, y))$
8:     end for
9:     $g_{DP} = \frac{1}{|b_k|} \sum_{i\in b_k} g_i \min(1, \frac{S}{\|g_i\|_2}) + \mathcal{N}(0, \sigma^2)$
10:    $\theta_{i+1} = \theta_i - \eta(g_{DP})$
11:  end for
12:  return $\theta_E$
13: end function
```

broadcasts the aggregated logits back to clients to align their local models using these global logits by training to reduce the distance between local and global logits.

3) Federated Learning with Augmented Knowledge Distillation (FedAKD) [11]. A KD-based method is similar to FedMD with an enriched public dataset. Each round, the server broadcasts the parameters α and β, which the clients use to augment their public datasets using Mixup augmentation [35]. This way, clients calculate their local logits on different yet synchronized data each round, which helps distill knowledge more efficiently.

4) Compress Federated Learning with Augmented Knowledge Distillation (CFedAKD) [36]. A KD-based Fl method in which local and global logits are compressed by normalizing and converting them to integers instead of floats.

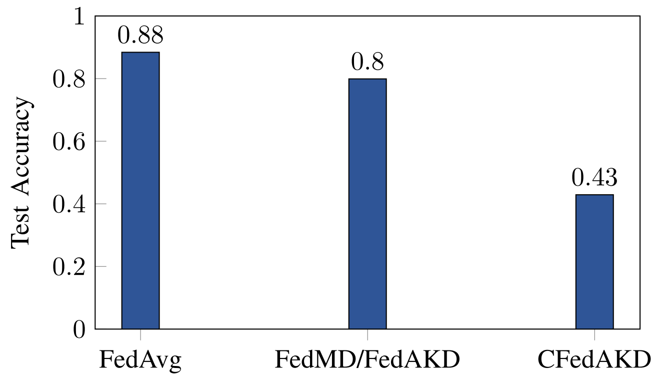


Fig. 12. Test accuracy achieved by central, local, and model-based FL (FedAvg), respectively, on the NTU-HumanID dataset.

For the CIFAR experiment, we set the number of clients to 20 and the number of rounds to 50. 5000 randomly drawn samples from CIFAR10 were shared across clients to be used as the public dataset. In the FEMNIST experiment, the number of clients and rounds were set to 10 and 30, respectively. The size of the public dataset in the MNIST experiment is set to 10,000. For both experiments, each client possesses five samples out of each class. For example, CIFAR100 has 100 classes, therefore each client has $100 \times 5 = 500$ samples as his private dataset. We use the Adam optimizer [37] and a learning rate of $1e^{-4}$ and $1e^{-3}$ for MNIST and CIFAR experiments, respectively.

For the KD-based FL algorithms, in each round, each client trains for two epochs on his local private dataset using Categorical Cross Entropy (CCE) as the loss function (this is called local training) and trains for one epoch on the public dataset using the Mean Squared Error (MSE) loss function (we call this KD-training).

To investigate the accuracy-privacy tradeoff, we apply LDP, where each client applies DP during training on the local private dataset using DP-SGD instead of SGD. In this setup, we first split CIFAR10 into two non-overlapping datasets, the public and private datasets. We then split the private dataset into ten i.i.d sampled datasets resembling the ten clients' private datasets. Similar to the previous experiment, we run FedAKD for 30 rounds. This time, both local training and KD training perform one epoch. Due to the utility loss incurred by adding noise under DP, DP-training (using DP-SGD) often uses larger learning rates compared to non-DP-training (using SGD). We use the SGD optimizer with a learning rate of 0.1. We fix the privacy parameters, gradient clipping norm, and the delta to 1.5 and $1e^{-}5$, respectively, while changing the $epsilon$ values from 0.1 to 1000.

TABLE II
LOCAL AND CENTRAL ACCURACIES ON MNIST AND CIFAR-100 DATASETS USING HETEROGENEOUS MODELS LISTED IN TABLE III

Client	MNIST	CIFAR-100
1	0.548	0.011
2	0.542	0.013
3	0.626	0.010
4	0.460	0.013
5	0.581	0.013
6	0.639	0.011
7	0.631	0.012
8	0.566	0.012
9	0.517	0.010
10	0.503	0.015
Average local Acc	0.574	0.0125
Central Acc	**0.7**	**0.15**

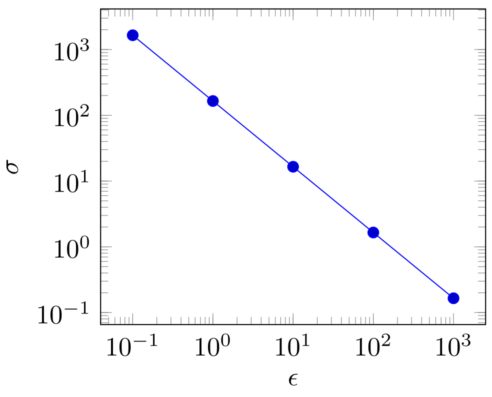


Fig. 13. The amount of noise added by local differential privacy under FedAKD.

VII. RESULTS AND DISCUSSION

Table II shows the local accuracies achieved by clients vs the central accuracy on the CIFAR-100 and MNIST datasets, respectively. There is a big discrepancy of 20 percent between clients' accuracies especially for the local accuracies of the MNIST dataset, this is because each client possesses different model architectures and each has a different learning capacity. FL aims to train a global model on distributed data to exceed the test accuracy achieved by individual users and approach central deep learning accuracy in which data are aggregated to a central server where a model is trained and broadcasted to clients/users.

In the next part of the results, we show results on four aspects of the studied FL algorithms: test accuracy, communication overhead, enabling system heterogeneity, round computation time, and the utility-privacy tradeoff.

A. Test Accuracy

Starting with the accuracy results, Fig. 8 shows the test accuracy attained by the considered FL algorithms on the MNIST dataset. Table IV also provides a summary of the test accuracy achieved by the considered FL paradigms on both datasets: CIFAR-100 and MNIST. The table also shows how the collaborative accuracies compare to local and central performance. For the CIFAR-100 dataset, KD-based FL algorithms performed better than FedAvg [3]. Specifically, FedMD accuracy approached 12 percent while FedAvg achieved below 8 percent. This is a huge improvement over the 1-2 percent accuracy obtained by local training but less than the central accuracy of 14 percent obtained by training on the collected clients' data. On the other hand, FedAvg performed better than all other KD-based algorithms on the MNIST dataset, where it achieved a test accuracy of 54.6 percent.

B. Communication Overhead

In KD-based FL, such as FedAKD and FedMD, clients share the soft labels they calculate over the shared public dataset. This approach has two advantages: first, the size of soft labels is usually less than the size of raw data or model weights; therefore, by sending soft labels, less data are being shared each round, which makes KD-based suitable for wireless networks with limited bandwidth. Figs. 9 and 10 show the communication overhead incurred by the considered FL algorithms on CIFAR-100 and MNIST datasets, respectively. Note that adding noise through differential privacy does not change the communication overhead of clients. It is clear that on both datasets, KD-based FL algorithms exchange less data than FedAvg. Specifically, CFedAKD [36] exchanges almost 5x and 13x less data than FedAvg for CIFAR-100 and MNIST, respectively. Secondly, unlike standard FL (i.e., FedAvg) which aggregates model weights and thus enforces a single architecture on all clients, in KD-based FL, each client can design his own model architecture to meet his hardware requirements. Table III lists the ten model architectures used by the ten clients participating in the MNIST experiment. As can be seen, each client has a different model architecture. This is made possible because only soft labels are shared in KD-based FL algorithms. However, when we run FedAvg, all clients have to have the same model architecture to be able to aggregate their model weights after each round.

C. Differential Privacy in KD-Based FL

We described in Section V how DP is used to perturb gradients during the training process: DP-SGD. The protection budget that characterizes a DP-trained model is controlled by the parameters ϵ. The amount of noise added depends on the value of ϵ we choose with a lower value, which means a stronger privacy guarantee and, therefore, adding more noise. Fig. 13 shows the amount of noise σ corresponding to different ϵ values. Adding noise, however, harms the utility of the model. To investigate this utility-privacy tradeoff, we run LDP while training FedAKD on CIFAR-10 under different protection budgets from $\epsilon = 10^{-1}$ to 10^3.

Fig. 14(a) illustrates the accuracy loss from adding noise during FedAvg training compared to non-private training. The accuracy loss, expressed as a percentage of the non-private training accuracy, is computed using the following equation:

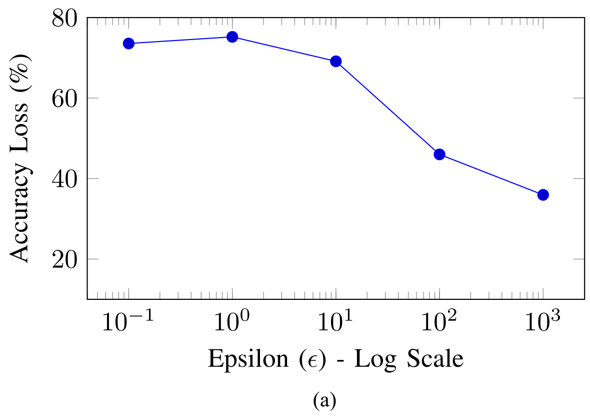
$$\text{accuracy\_loss} = \left(\frac{\text{nondp\_acc} - \text{acc}}{\text{nondp\_acc}}\right) \times 100. \qquad (26)$$

TABLE III
SPECIFICATIONS OF HETEROGENEOUS CNN MODELS USED BY KD-BASED FL CLIENTS

Model Number	1st Conv Layer Filters	2nd Conv Layer Filters	3rd Conv Layer Filters	Dropout Rate
1	128	256	-	0.2
2	128	384	-	0.2
3	128	512	-	0.2
4	256	256	-	0.3
5	256	512	-	0.4
6	64	128	256	0.2
7	64	128	192	0.2
8	128	192	256	0.2
9	128	128	128	0.3
10	128	128	192	0.3

TABLE IV
TEST ACCURACY OF FL ALGORITHMS ON MNIST AND CIFAR DATASETS

Algorithm	MNIST	CIFAR-100
CFedAKD	0.4967	0.089
FedMD	0.49	0.115
FedAKD	0.52	0.089
FedAvg	0.54	0.075
Average Local	0.56	0.01
Central Accuracy	**0.7**	**0.14**



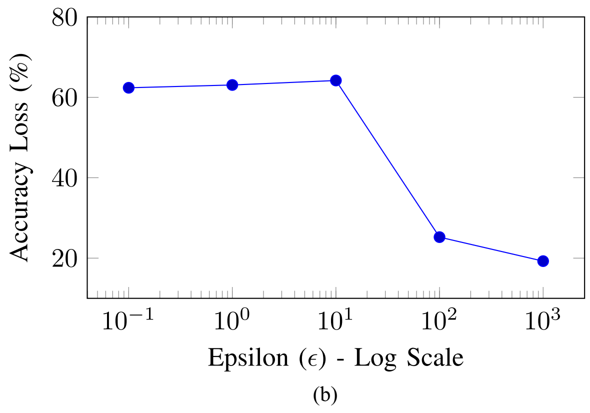


Fig. 14. Accuracy losses for different protection levels when local differential privacy is applied to FedAvg and FedAKD. (a) FedAvg accuracy losses for various protection levels (ϵ values) under the effect of local differential privacy. (b) FedAKD accuracy losses for various protection levels (ϵ values) under the effect of local differential privacy.

Similarly, Fig. 14(b) shows the accuracy loss resulting from adding noise during FedAKD training compared to non-private training. In our DP experiments, we used the advanced composition variant [28].

As can be seen, about 70% of the non-DP accuracy is lost due to adding noise under DP for $\epsilon \leq 10$. As ϵ increases to 100 and 1000, the amount of noise to be added (σ) decreases (as shown in Fig. 13), and the accuracy loss incurred decreases to below 40%.

D. Computation Time

The paradigm of KD-based FL enables model heterogeneity and reduces the communication requirements at the cost of increased computation overhead, presenting a critical consideration in scenarios where resources are constrained. Consequently, the deployment of such algorithms may not always be feasible on devices with limited resources, even though these algorithms are notably communication-efficient. Moreover, while this class of solutions enhances the performance of non-collaborative training, they typically fall short in global model accuracy when compared to FedAvg-based baselines. [12], [38] A per-round computation time comparison between FedAvg [3], FedMD [10], and FedAKD [11] is depicted in Fig. 11 for the CIFAR-100 dataset. KD-based FL methods, specifically FedAKD and FedMD, exhibit computation times that are twenty times greater compared to model-based FL such as FedAvg. This difference in computational time is due to the operational framework of these methods. In model-based FL, the local computation time T_L aligns with the local training time T_{LT} as each client is only engaged in local training. However, in KD-based FL, the local computation time T_L encompasses both local training time T_{LT} and local distillation time T_{DL}. Thus, it emerges that KD-based FL, despite its significant communication efficiency in comparison to model-based FL, demands a more substantial allocation of computational resources and time. This increased demand is attributed to KD-based FL undergoing more epochs per round.

VIII. CONCLUSION

This paper investigated communication, heterogeneity, and privacy challenges that face the deployment of FL on hardware-constrained devices and B5G wireless networks. We perform a comparative analysis of recent solutions to these challenges, such as Knowledge Distillation (KD) and Differential Privacy (DP), and propose a system that integrates both techniques to enable system heterogeneity between devices and greatly reduces the communication cost of standard FL (e.g. Fedavg), and risks of privacy attacks such as Membership Inference Attacks (MIA). Experiments show that KD-based FL algorithms boost the performance of clients' models beyond local training efforts while incurring significantly less communication overhead compared to standard FL. On the other hand, the per-round computation time for KD-based FL is more than that of model-based FL methods, as KD-based FL clients need to perform local distillation in addition to local training. We study the utility-privacy tradeoff of adding calibrated noise under DP in KD-based FL using different privacy protection budgets ϵ. Additionally, KD-based FL algorithms enable clients to have control over the design of their local models, which is a crucial requirement for hardware-constrained devices. Finally, our experiments show that applying DP significantly damages the utility of KD-based FL algorithms; setting the privacy budget $\epsilon \leq 10$ resulted in 70% accuracy loss compared to the non-DP trained model. Our investigation reveals the tradeoff between privacy strength and model accuracy in the federated learning setup when a differential privacy mechanism is applied.

References

[1] I. Zhou et al., "Internet of Things 2.0: Concepts, applications, and future directions," *IEEE Access*, vol. 9, pp. 70961–71012, 2021, doi: 10.1109/ACCESS.2021.3078549.

[2] R. Saleem, W. Ni, M. Ikram, and A. Jamalipour, "Deep-reinforcement-learning-driven secrecy design for intelligent-reflecting-surface-based 6G-IoT networks," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 8812–8824, May 2023, doi: 10.1109/JIOT.2022.3232360.

[3] B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artif. Intell. Statist.*, 2017.

[4] P. Ramezani and A. Jamalipour, "Toward the evolution of wireless powered communication networks for the future Internet of Things," *IEEE Netw.*, vol. 31, no. 6, pp. 62–69, Nov./Dec. 2017, doi: 10.1109/MNET.2017.1700006.

[5] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.

[6] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2017, *arXiv:1712.01887*.

[7] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, 2021, doi: 10.1007/s11263-021-01453-z.

[8] T. Kim, J. Oh, N. Y. Kim, S. Cho, and S.-Y. Yun, "Comparing Kullback-Leibler divergence and mean squared error loss in knowledge distillation," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021, pp. 2628–2635, doi: 10.24963/ijcai.2021/362.

[9] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for BERT model compression," in *Proc. 2019 Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 4323–4332, doi: 10.18653/v1/D19-1441.

[10] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," 2019, *arXiv:1910.03581*.

[11] G. Gad and Z. Fadlullah, "Federated learning via augmented knowledge distillation for heterogenous deep human activity recognition systems," *Sensors*, vol. 23, no. 1, 2022, Art. no. 6, doi: 10.3390/s23010006.

[12] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021.

[13] A. Asad, M. M. Fouda, Z. M. Fadlullah, M. I. Ibrahem, and N. Nasser, "Moreau envelopes-based personalized asynchronous federated learning: Improving practicality in network edge intelligence," in *Proc. 2023-2023 IEEE Glob. Commun. Conf.*, 2023, pp. 2033–2038, doi: 10.1109/GLOBECOM54140.2023.10437327.

[14] X. Ouyang, Z. Xie, J. Zhou, G. Xing, and J. Huang, "ClusterFL: A clustering-based federated learning system for human activity recognition," *ACM Trans. Sensor Netw.*, vol. 19, no. 1, 2022, Art. no. 17, doi: 10.1145/3554980.

[15] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. 2019 IEEE Int. Conf. Commun.,* 2019, pp. 1–7, doi: 10.1109/ICC.2019.8761315.

[16] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," in *2019 IEEE VTS Asia Pacific Wireless Commun. Symp.*, 2019, pp. 1–5, doi: 10.1109/VTS-APWCS.2019.8851649.

[17] A. Shafahi et al., "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 7564–7575, doi: 10.5555/3327345.3327509.

[18] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," 2018, *arXiv:1808.04866*.

[19] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018, doi: 10.1109/TIFS.2017.2787987.

[20] L. Su and J. Xu, "Securing distributed gradient descent in high dimensional statistical learning," in *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 1, 2019, Art. no. 12, doi: 10.1145/3322205.3311083.

[21] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symp. Secur. Privacy*, 2017, pp. 3–18, doi: 10.1109/SP.2017.41.

[22] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symp. Secur. Privacy*, 2019, pp. 691–706, doi: 10.1109/SP.2019.00029.

[23] M. Abadi et al., "Deep learning with differential privacy," in *Proc. 2016 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318, doi: 10.1145/2976749.2978318.

[24] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," in *NIPS'18: Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 6106–6116, doi: 10.5555/3327757.3327856.

[25] Y. Tsuzuku, H. Imachi, and T. Akiba, "Variance-based gradient compression for efficient distributed deep learning," 2018, *arXiv:1802.06058*.

[26] G. Gad, Z. M. Fadlullah, M. M. Fouda, M. I. Ibrahem, and N. Nasser, "Joint knowledge distillation and local differential privacy for communication-efficient federated learning in heterogeneous systems," in *Proc. 2023 IEEE Glob. Commun. Conf.*, 2023, pp. 2051–2056, doi: 10.1109/GLOBECOM54140.2023.10437358.

[27] G. Gad, A. Farrag, A. Aboulfotouh, K. Bedda, Z. M. Fadlullah, and M. M. Fouda, "Joint self-organizing maps and knowledge distillation-based communication-efficient federated learning for resource-constrained UAV-IoT systems," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15504–15522, May 2024, doi: 10.1109/JIOT.2023.3349295.

[28] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations Trends Theor. Comput. Sci.*, vol. 9, no. 3/4, pp. 211–407, 2014, doi: 10.1561/0400000042.

[29] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?," in *NIPS'20: Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 16937–16947, doi: 10.5555/3495724.3497145.

[30] X. Gong et al., "Preserving privacy in federated learning with ensemble cross-domain knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, pp. 11891–11899, doi: 10.1609/aaai.v36i11.21446.

[31] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," 2016, *arXiv:1610.05755*.

[32] I. Donevski, N. Babu, J. J. Nielsen, P. Popovski, and W. Saad, "Federated learning with a drone orchestrator: Path planning for minimized staleness," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 1000–1014, 2021, doi: 10.1109/OJCOMS.2021.3072003.

[33] A. Yousefpour et al., "Opacus: User-friendly differential privacy library in pytorch," 2021, *arXiv:2109.12298*.

[34] D. Wang, J. Yang, W. Cui, L. Xie, and S. Sun, "CAUTION: A robust WiFi-based human authentication system via few-shot open-set recognition," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17323–17333, Sep. 2022, doi: 10.1109/JIOT.2022.3156099.

[35] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–13. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[36] G. Gad, "Light-weight federated learning with augmented knowledge distillation for human activity recognition," Ph.D. dissertation, Lakehead Univ., Thunder Bay, ON, Canada, 2023. [Online]. Available: https://knowledgecommons.lakeheadu.ca/handle/2453/5175

[37] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," 2014, *arXiv.1412.6980*.

[38] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data," 2018, *arXiv:1811.11479*.

Gad Gad (Student Member, IEEE) received the bachelor's degree in computer engineering from Nile University, Egypt, in 2021, and the master's degree in computer science from Lakehead University, Thunder Bay, ON, Canada, in 2023. He is currently working toward the Ph.D. degree in computer science with Western University, London, ON.. His main research interests include deep learning, federated learning, and differential privacy. He was the recipient of Vector Institute AI scholarship, and Best Poster Award at the NRSC 2020 conference.

Eyad Gad (Graduate Student Member, IEEE) received the bachelor's degree in computer engineering from Nile University, Egypt, in 2023. He is currently working toward the master's degree in computer science with Western University, London, ON, Canada. His research interests include artificial intelligence and its applications in wireless communications and network systems.

Zubair Md Fadlullah (Senior Member, IEEE) is currently an Associate Professor with Western University, London, ON, Canada. He was an Associate Professor with Lakehead University, Thunder Bay, ON, Canada, and Tohoku University, Sendai, Japan. He was a highly cited Researcher in computer science in 2021 and 2022. His research interests include data collection, communication, and computing of cyber-physical systems with a focus on performance quality, reliability, and security.

Mostafa M. Fouda (Senior Member, IEEE) is currently an Associate Professor with the Department of Electrical and Computer Engineering at Idaho State University, ID, USA. He also holds the position of a Full Professor with Benha University, Banha, Egypt. He has coauthored more than 260 technical publications. His current research intersts include cybersecurity, communication networks, signal processing, wireless mobile communications, smart healthcare, smart grids, AI, and IoT. He has guest-edited a number of special issues covering various emerging topics in communications, networking, and health analytics. He currently serves on the Editorial Board of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY (TVT), IEEE INTERNET OF THINGS JOURNAL (IoT-J), and IEEE ACCESS. He has received several research grants including NSF Japan-U.S. Network Opportunity 3 (JUNO3).

Nei Kato (Fellow, IEEE) is currently a Full Professor and the Dean with the Graduate School of Information Sciences. He has authored or coauthored more than 400 papers in prestigious peer-reviewed journals and conferences. He has been engaged in research on computer networking, wireless mobile communications, satellite communications, ad hoc and sensor and mesh networks, UAV networks, smart grid, AI, IoT, Big Data, and pattern recognition. He has been acclaimed with many best paper awards and a long list of other awards and recognition. He was the Vice-President (Member and Global Activities) of IEEE Communications Society from 2018 to 2021, the Editor-in-Chief of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY from 2017 to 2021 and *IEEE Network Magazine* from 2015 to 2017. He is a Fellow of The Engineering Academy of Japan and IEICE.