Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.048370

ARTICLE

WebFLex: A Framework for Web Browsers-Based Peer-to-Peer Federated Learning Systems Using WebRTC

Mai Alzamel[1,*], Hamza Ali Rizvi[2], Najwa Altwaijry[1] and Isra Al-Turaiki[1]

[1]Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh, Kingdom of Saudi Arabia

[2]Department of Computer Science and Engineering, Punjab Engineering College, Chandigarh, India

*Corresponding Author: Mai Alzamel. Email: malzamel@ksu.edu.sa

Received: 06 December 2023 Accepted: 05 February 2024 Published: 26 March 2024

ABSTRACT

Scalability and information personal privacy are vital for training and deploying large-scale deep learning models. Federated learning trains models on exclusive information by aggregating weights from various devices and taking advantage of the device-agnostic environment of web browsers. Nevertheless, relying on a main central server for internet browser-based federated systems can prohibit scalability and interfere with the training process as a result of growing client numbers. Additionally, information relating to the training dataset can possibly be extracted from the distributed weights, potentially reducing the privacy of the local data used for training. In this research paper, we aim to investigate the challenges of scalability and data privacy to increase the efficiency of distributed training models. As a result, we propose a web-federated learning exchange (WebFLex) framework, which intends to improve the decentralization of the federated learning process. WebFLex is additionally developed to secure distributed and scalable federated learning systems that operate in web browsers across heterogeneous devices. Furthermore, WebFLex utilizes peer-to-peer interactions and secure weight exchanges utilizing browser-to-browser web real-time communication (WebRTC), efficiently preventing the need for a main central server. WebFLex has actually been measured in various setups using the MNIST dataset. Experimental results show WebFLex's ability to improve the scalability of federated learning systems, allowing a smooth increase in the number of participating devices without central data aggregation. In addition, WebFLex can maintain a durable federated learning procedure even when faced with device disconnections and network variability. Additionally, it improves data privacy by utilizing artificial noise, which accomplishes an appropriate balance between accuracy and privacy preservation.

KEYWORDS

Federated learning; web browser; privacy; deep learning

1 Introduction

Federated learning is a novel technique for training deep learning models that has been developed primarily in response to increasing concerns surrounding data privacy. Traditionally, deep learning models are trained by collecting data from various sources into a centralized location. However, this approach presents significant risks to data privacy because the data from each source becomes

visible during the aggregation process. Therefore, federated learning introduces a paradigm shift in the training methodology to overcome these concerns [1]. Instead of centralizing data, this technique emphasizes the training of models directly at their local data sources. Subsequently, only the model weights, which are a product of local training, are transmitted to a central server. These weights from all participating local models are aggregated at the server to produce a comprehensive global model. This methodology ensures an enhanced level of data protection since it restricts any external entity from gaining direct insight into or access to the local datasets [2].

Federated learning in browser-based peer-to-peer settings offers an important change in machine learning, decentralizing the process and allowing various devices to train a model while preserving data privacy and security. This approach incorporates a wide variety of devices, which improves accessibility and inclusivity [3]. However, it faces challenges such as device heterogeneity, leading to variable computational power and network issues, and necessitates strong security measures to safeguard the integrity of the decentralized model. Additionally, achieving compatibility across different browsers and managing asynchronous client interactions presents implementation complexities. Despite these challenges, this form of federated learning is a significant step forward, challenging traditional centralized models and fostering collaborative, privacy-centric machine learning advancements [4].

Federated learning has been gaining power across various domains due to its ability to train on decentralized data sources while ensuring data privacy. For instance, in healthcare, federated learning can be employed to develop predictive models by training on patient data across different hospitals without directly sharing the sensitive patient information, thus maintaining data confidentiality [5]. Similarly, in the field of mobile devices, companies such as Google have utilized federated learning to improve their predictive text functionalities without extracting the raw data from users' devices. Additionally, in the financial sector, this approach allows banks and financial institutions to collaborate on fraud detection models without demonstrating individual transaction details. Through these applications, federated learning shows its potential to sectors by encouraging collaboration without affecting data security [6].

Federated learning, although an innovative and promising approach to distributed deep learning, is accompanied by several inherent challenges [7]. Initially, the central server plays a crucial role in facilitating the coordination and aggregation of model updates. However, a central server failure or unavailability can disrupt the entire training process and hinder timely update aggregation. Furthermore, the server's ability to manage multiple clients concurrently establishes limitations on the number of active participants during any specific learning session [8]. Additionally, the essential structure of federated learning introduces significant communication overheads. This is attributed to the frequent need for data exchange between the central server and client devices which can lead to pressure on the network bandwidth, consequently decreasing the overall training progression [6]. Lastly, client heterogeneity poses significant challenges, particularly in terms of hardware configurations and compilation environments. Each client device, equipped with varied components such as central processing units (CPUs), graphics processing units (GPUs), and tensor processing units (TPUs), requires specific driver installations for optimal performance. The compilation environment, comprising software tools and libraries, must be compatible with the device's hardware. Ensuring compatibility and proper configuration becomes complicated due to the variety of requirements and variations of different devices [9].

In this paper, we intend to propose an innovative methodology to overcome existing challenges in federated learning by employing the consistency and availability of web browsers. These web browsers present remarkable consistency across a variety of edge devices, including smartphones, laptops, and

personal computers. Particularly, the proposed framework, WebFLex, utilizes web browsers to train a convolutional neural network (CNN) on the MNIST dataset. This training process is executed within a web browser running on heterogeneous devices supported by a virtual reality (VR) headset, which incorporates the web graphics library (WebGL) to enhance both 2D and 3D graphics rendering. The framework incorporates WebGL to facilitate graphics acceleration and render complex visualizations efficiently using the device's GPU. Therefore, this research presents the main contributions in the following aspects:

1. We propose a novel framework called WebFLex, which facilitates the construction of federated learning systems. This framework can train and aggregate models locally on peer nodes within the federation. As a result, the primary function of the central server is restricted to coordinating peer-to-peer connections among the peer participants. After the establishment of these connections, the need for a centralized server is minimized, subsequently reducing the dependence on a centralized architecture. Therefore, the WebFLex framework enables decentralized federated learning within web browsers, allowing devices to participate in the learning process even if they disconnect during training. Moreover, it incorporates local differential privacy and artificial noise to enhance privacy preservation.
2. A comprehensive performance evaluation of the WebFLex framework is performed across a range of settings. The analysis of performance accuracy highlights the trade-off between data privacy and model performance when employing local differential privacy and adding artificial noise. Additionally, the effectiveness of the WebFLex framework is demonstrated in training a deep CNN on the widely used MNIST dataset across various devices.

The remainder of this research paper is structured as follows: Section 2 reviews related work, while Section 3 presents the preliminary concepts to understand the research. The WebFLex framework is detailed in Section 4, outlining its key features and design principles. Section 5 discusses the experimental results obtained from evaluating WebFLex's performance. Section 6 provides a comprehensive discussion, evaluating the advantages and disadvantages of the framework. Section 7 presents the practical implementation challenges of WebFLex. Lastly, Sections 8 and 9 provide future directions and conclude this research work, respectively.

2 Related Work

In recent years, federated learning has attracted significant advances and attention, leading to numerous related works that investigate its various aspects to enhance its effectiveness, scalability, and robustness. This section presents an overview of literature related to the field of federated learning.

Federated learning has witnessed many recent publications on different applications and challenges across various domains, showing its versatility and potential impact [5,9–11]. Yang et al. [12] use federated learning to train a model for the detection of credit card fraud. Federated learning with differential privacy has been used to train an in-hospital mortality prediction model [13]. More recently, Moshawrab et al. [9] examine federated learning for the prediction of diseases, including cardiovascular diseases, diabetes, and cancer.

Federated learning presents two distinct settings: The traditional centralized setting and the decentralized setting. In the traditional centralized setting, the communication is asymmetric, where a server aggregates local models and shares the training results with all parties. Federate averaging (FedAvg) [14] is the standard federated learning algorithm, and it is based on deep learning. In FedAvg, local stochastic gradient descent (SGD) on each client is combined with a central server that performs

model averaging. This algorithm considerably reduces the cost of communication between the server and participating devices. However, McMahan et al. [14] suggest that the central server in this setting may experience communication and computational overhead as a result of the numerous connected participants. Furthermore, model training may be affected in the case of a central server failure. Thus, a decentralized federated learning architecture has been proposed as an alternative.

On the other hand, the decentralized setting does not have a central server, and all parties can communicate directly and update the global model. Lalitha et al. [15] propose a framework for a fully decentralized federated learning system in which users update their weights by exchanging information with their one-hop neighbors. They theoretically investigate how participating devices collaboratively train a model over a network without the need for a central server. Despite the potential benefits of this approach, an empirical assessment of the proposed framework needs to be demonstrated. Sun et al. [16] extend the FedAvg algorithm to the decentralized setting, where participants connect to each other to form an undirected graph.

There are many issues related to the implementation of federated learning in real-world applications. One important aspect is related to data and system heterogeneity [17,18]. Li et al. [19] propose decentralized federated learning that incorporates mutual knowledge transfer. This is introduced to address the degradation of learning performance caused by client-drift in the presence of heterogeneous data. Chen et al. [20] introduce a peer-to-peer framework to address the challenges posed by systems and data heterogeneity in federated learning. The proposed framework involves local clients iteratively selecting learning pairs for model exchange to optimize multiple learning objectives and promote fairness while avoiding small-group dominance. Ramanan et al. [21] develop a framework that utilizes smart contracts on the Ethereum blockchain network to manage model aggregation, round delineation, and local update tasks. By eliminating the need for a central aggregator, the proposed framework achieves high scalability while utilizing the decentralized nature of blockchain technology and operating at a lesser cost as opposed to the centralized alternative on the same network while achieving similar accuracy. WebFed [22] is a centralized federated learning framework that addresses the issue of system heterogeneity. It allows for multiple devices to engage in a federated learning session through web browsers. While local updates occur within the browser, weights are aggregated at a centralized parameter server and distributed to all clients in the federation. To preserve local differential privacy, each client adds artificial noise to its local training weights before they are aggregated at the semi-trusted parameter server. Although WebFed is suited for heterogeneous participants in terms of configuration requirements, the central parameter server may become a potential point of failure if it is stressed by requests from multiple clients, potentially disrupting the federated learning session.

Table 1 provides a comprehensive comparison of various existing federated learning frameworks. It systemically outlines their key features and characteristics. This includes the communication architecture of each framework, assessing their compatibility with different systems, scalability potential, and the privacy methods employed. It also offers insights into significant contributions, datasets utilized, and their overall performance.

Finally, researchers have explored various aspects of federated learning, including communication efficiency, data heterogeneity, security, and privacy [14,22–25]. The advancements in these areas have allowed for the building of more robust, scalable, and privacy-preserving federated learning systems [2,16,20]. However, several challenges, such as model aggregation techniques, optimization strategies, and standardization, still remain open research questions, providing exciting opportunities for future investigations [6,10,11,17].

Table 1: Features and characteristics of existing federated learning frameworks

Ref.	Communication architecture	Device compatibility	Scalability	Data Privacy/Method	Contribution	Dataset	Performance
[12]	Decentralized with a central server to coordinate updates	Moderate to high	High	High (uses encryption for communication)	Addresses data insufficiency	European credit card (ECC)	Outperforms traditional models
[13]	Centralized and decentralized	Suitable for many clients	High	High (differential privacy)	Enhances privacy and bandwidth efficiency	Electronic health records (EHR)	Reduces bandwidth consumption
[14]	Decentralized	Likely high	Very high (over 500,000 clients in one experiment)	High (differential privacy)	Efficient communication and privacy preservation	MNIST, CIFAR-10, SHAKESPEARE	Reduces communication rounds
[15]	Decentralized	High	Likely high	High (Bayesian-like approach)	Novel decentralized approach	Generalized dataset	Collaborative learning, reduced control
[16]	Decentralized (communicate with their neighbors only)	Likely high	High	High	Improves communication efficiency and privacy	MNIST, CIFAR-10, SHAKESPEARE	Reduces communication cost
[19]	Peer-to-peer decentralized	High	High	High	Enhances learning performance	MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100	Outperforms baseline methods
[20]	Peer-to-peer decentralized	High	High	High	Addresses learning fairness and device heterogeneity	E-MNIST, CIFAR-10	Improves fairness in learning distribution
[21]	Decentralized (uses blockchain technology to avoid a centralized aggregator)	High	High	High	Novel model using blockchain technology	NYC taxi dataset	Reduces operational costs and enhances privacy
[22]	Centralized (involves a parameter server)	High (browser-based approach)	High (browser-based approach)	High (local differential privacy)	A novel browser-based approach with enhanced privacy	MNIST	Close to conventional frameworks with added benefits of privacy and ease of use

3 Preliminaries

Federated learning incorporates a decentralized approach to train models across multiple data sources without the need to share raw data. This approach presents unique challenges and considerations, particularly in terms of privacy and communication efficiency [2,13,20]. This section presents a set of preliminary concepts and background information, which helps to understand this research paper as follows:

1) Diffie-Hellman Key Exchange (DHKE):
DHKE is used to facilitate secure communication between peers in federated learning. This cryptographic method allows two peers to establish a shared secret key over an insecure channel without prior knowledge of each other's presence. By utilizing this method, peers can securely exchange encryption keys and establish a secure communication channel. In federated learning, this ensures that data or model parameters transferred between peers remain confidential, protecting against potential attackers [23]. The key exchange process can be mathematically represented as:

$$K = g^{a.b} \; mod \; p \tag{1}$$

where K represents the shared secret key, g is a public base value, p is a public large prime number, and a and b are the private keys of the two communicating peers.

2) Brokering Server
The brokering server plays an important role as a centralized moderator, coordinating the collaboration of distributed devices or peers without direct access to their localized data [25]. This server supports peers to communicate with each other using their unique peer IDs. Therefore, the primary purpose of the brokering server is to address the challenge of peers lacking knowledge about each other's presence. For more information on its operational processes, when two individual peers seek to establish a connection, they initially lack any knowledge regarding the presence or attributes of the other. This information gap is filled by the brokering server. Each peer connects to this server, which recognizes the presence of both peers. Subsequently, the brokering server initiates the exchange of metadata, including network location information, between the two peers. This metadata exchange, known as signaling, enables peers to discover and establish a direct connection for data exchange. Peers can disconnect from the brokering server once the signaling process is complete if they do not intend to establish connections with other peers [24]. This process can be represented as:

$$M_{ij} = B_s(P_i, P_j) \tag{2}$$

where B_s is the brokering server that acts as a centralized coordinator for peer connections, P_i and P_j are two peers, and M_{ij} is the metadata that the server facilitates to exchange between P_i and P_j.

3) Global Model
The global model represents the aggregated weights accumulated from all participating peers in the federated learning system. After each communication round, local model updates generated by individual peers are aggregated in a weighted manner to update and refine the global model. This process involves integrating the contributions of each peer's localized model. By integrating these updated weights, the global model enhances to reflect the collective understanding and knowledge acquired from the distributed peers. Once the global model is updated, it is transmitted back to the peers, enabling the subsequent rounds of local training to start based on the updated global model. This iterative procedure guarantees that the

global model continually improves and matches the collective knowledge and expertise of its distributed peers [18]. The mathematical representation of global model aggregation from local models in federated learning can be expressed as:

$$G = f(L_1, L_2, \ldots\ldots, L_n) \tag{3}$$

where G is the global model, L_i is the local model update of the i^{th} peer, f is the aggregation function, and n is the number of peers.

4) Local Model
The local model denotes a duplicated version of the global model that is maintained by each participating peer. This model represents the same architectural design and initial random weight configuration as the global model. Moreover, the local model enables individual peers to conduct computations and updates according to their own local data. By incorporating local data insights, the local models capture the variations and patterns specific to each peer dataset. Each peer transmits these updates, rather than the raw data, to the brokering server or other peers. This decentralized approach enables peers to independently learn from their local data while collectively contributing to the refinement and accuracy of the overall model [20]. The update process of local model can be represented as follows:

$$L_i = G + \Delta(D_i) \tag{4}$$

where G is the global model, L_i is the local model update of the i^{th} peer, D_i is the local dataset, and Δ is the function that applies the learning from local data.

5) Communication Round
The communication round is an iterative process to enable the synchronization and collaboration of models across distributed peers. It involves the exchange of information and data between peers. During each communication round, peers transmit their local model updates to the brokering server or other peers to generate an updated global model [18].

4 Framework Design: WebFLex

In this work, we propose a novel framework, WebFLex, for building decentralized federated learning systems within web browsers. The proposed WebFLex framework is designed to prioritize privacy and scalability. Moreover, it is adeptly capable of handling scenarios in which devices may disconnect from the federated learning session at any stage of the training process. The WebFlex framework consists of two main stages: Creating and joining a federation and training models in a federated learning session. The following subsections present the initialization of peer attributes, the participation of peers in pair federations, the training models in a federated learning session, and the overall workflow of the WebFLex framework, which enables training models on web browsers in a peer-to-peer setting.

4.1 Initialization of Peer Attributes

In the WebFLex framework, the first stage involves creating a federation where multiple peers can join to participate in the same federated learning task. There are three attributes for each peer that are initialized before a peer can join or create a federation. These attributes are used as inputs for facilitating the peer-to-peer federated learning process. The list of peer attributes and their descriptions are explained in Table 2.

Table 2: List of peer attributes and descriptions

Attributes	Description
Peer unique ID	A unique ID is automatically generated from the peer's MAC address.
Address book	A set contains the unique IDs of all peers who are a part of the federation.
Keychain	A directory of cryptographic keys [23] that a peer can use to communicate with other peers in the federation.

4.2 Participation of Peers in Pair Federations

Once the federation is established, the second stage is peer-pair federation participation; each peer node is automatically considered a member of its own federation. For clarification, let us refer to the first peer as Peer A and the second peer as Peer B. To facilitate the participation of the pair in the federation, a series of actions are undertaken. Firstly, Peer B establishes a connection with a brokering server to enable Peer B to join the Peer A federation. Secondly, an important aspect of peer pair federation participation requires the establishment of a direct connection between the two peers, specifically Peer A and Peer B. This connection plays an important role in facilitating direct communication and interaction between the two peers within the federated network. Through this connection, Peer A and Peer B acquire the capability to exchange vital information, share models, and actively participate in collaborative learning processes. Peer A and Peer B respect the attributes of their own federations, as outlined in Table 2. In the case of Peer B desiring to join Peer A's federation, it must first establish a connection with a brokering server. This connection serves as a critical prerequisite for Peer B to gain access to Peer A's federation and participate in its collaborative activities. Through this connection, each peer is enabled to both receive and initiate direct peer-to-peer connections with other peers within the federated network, as shown in Fig. 1.

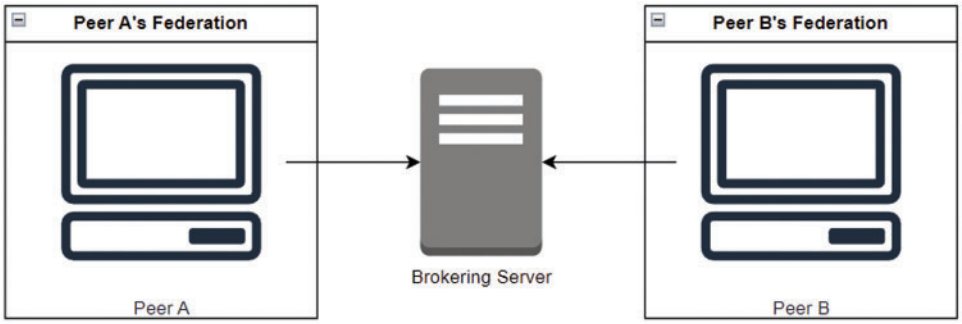

Figure 1: Peers connected to a brokering server

After establishing a connection between peers, both peers can disconnect from the connection of the brokering server to avoid receiving or initiating other connections; this ability is optional for each peer. The DHKE algorithm is used to ensure communication security as shown in Eq. (1). DHKE plays a crucial role in enhancing privacy and security in decentralized federated learning. It allows each peer to create a unique key pair, exchange public keys, and independently generate a shared secret key that is not transmitted, minimizing interception risks. This shared key encrypts peer-to-peer communication, preserving the privacy of model updates due to its ability to maintain confidentiality and support forward secrecy. Once the key exchange process is complete, both Peer A and Peer B

will add each other's ID and the corresponding encryption key to their respective address books. Consequently, both peers will be considered members of each other's federation. Fig. 2 illustrates the mutual membership of Peer A and Peer B within their respective federations.

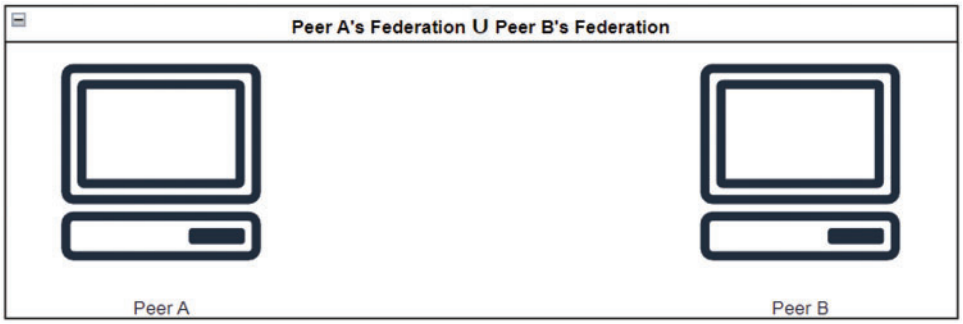

Figure 2: Mutual federation membership between Peer A and Peer B

4.3 Training Models in a Federated Learning Session

After a federation is created and member peers join, the process of deep learning training starts. Many attributes and weights are employed during the training session. Table 3 shows the global and local model attributes, including S, R, and weights queue, accompanied by a detailed description. The distinction between global and local model attributes lies in their scope and impact on the training process. Global model attributes are typically related to general aspects of the training, while local model attributes are specific to individual peers or participants in the federated learning setup. These parameters offer insight into the architecture and operational dynamics of the federated training session.

Table 3: Global and local model attributes for federated learning

Attributes	Description
R	It is the number of communication rounds; in each round, S peers will be selected randomly to perform the training on their local data.
S	It determines the number of peers whose weights are aggregated in each communication round, where the maximum number of peers is the address book size.
Weights queue	It serves as a structured storage mechanism, organizing model weight updates from various nodes. It ensures orderly processing and aggregation of these updates, facilitating efficient global model.

To illustrate the utilization of attributes and weights, we consider the practical scenario of Peer A's federation, which comprises Peer B, Peer C, and Peer A itself. In this scenario, Peer A initiates the federated learning session by sharing its model's weights with the other peers listed in its address book, as shown in Fig. 3. This step ensures that all peers have a common starting point for model training.

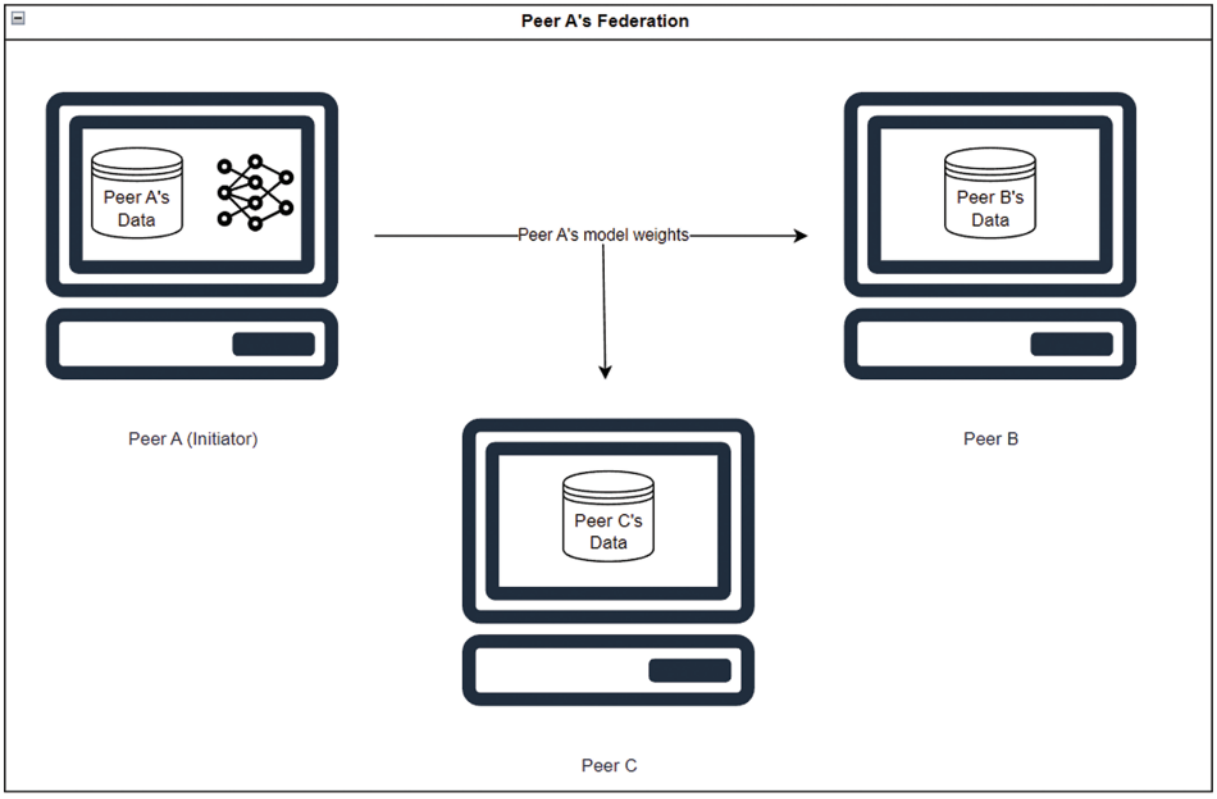

Figure 3: Distribution of Peer A's weights to the other federation peers

In federated learning, upon receiving the weights from Peer A, each participating peer, such as Peer B or Peer C, loads these weights into their respective local models. Subsequently, each peer conducts training using its own locally available data, as illustrated in Fig. 4. This decentralized training approach allows for privacy preservation and avoids the need for sharing raw data. The local model training process typically employs deep learning algorithms that are well-suited for iterative improvement. Algorithms based on iterative improvement, such as gradient descent, naturally align with the local update step in the federated learning framework. Each device independently computes the gradient based on its local data, and subsequently, these gradients are aggregated to update the global model. The gradient descent process at each peer node can be represented by the following equation:

$$W_p^{'} = W_p - \eta.\nabla L(W_p, D_p) \tag{5}$$

where η is the learning rate, ∇L is the gradient of the loss function L, W_p and $W_p^{'}$ are the local model weights of peer p before and after training, respectively, and D_p is the local dataset of peer p.

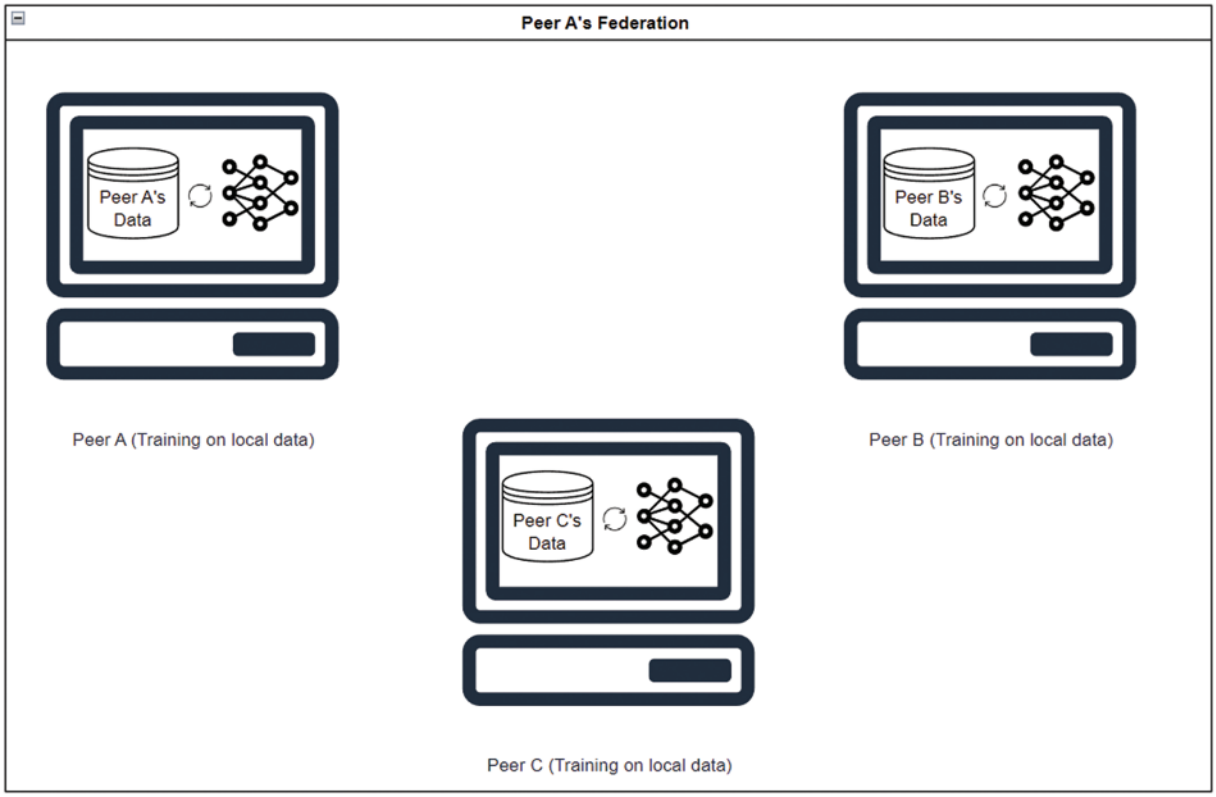

Figure 4: Model training of Peers B and C on local data

Upon completion of the local training loop, an essential step is to address privacy concerns related to potential information leakage about the local datasets used for training. To reduce this risk, artificial Gaussian noise is introduced to the local model weights before they are transmitted for aggregation, as defined by:

$$W_p^{''} = W_p^{'} + N(0, \sigma^2) \tag{6}$$

where $N(0, \sigma^2)$ represents Gaussian noise with mean 0 and variance σ^2, and $W_p^{''}$ is the local model weights of peer p after training and the addition of Gaussian noise. Moreover, the addition of Gaussian noise helps to hide patterns specific to individual datasets, thereby enhancing privacy protection [26,27]. Then, the locally updated model weights are sent back to the initiating node (in this case, Peer A) for the aggregation process, as shown in Fig. 5. Aggregation involves combining the weights received from participating peers to obtain an updated global model as the following:

$$W_{global}^{'} = \frac{1}{S} \sum_{p=1}^{S} W_p^{''} \tag{7}$$

where $W_{global}^{'}$ is the updated global model weight, S is the number of selected peers, and $W_p^{''}$ are the local model weights of peer p. Once the aggregation is complete, the updated weights are distributed to all other peers in the federation to ensure uniformity and synchronization, as shown in Fig. 6. This distribution of weights enables the peers to stay aligned and maintain consistency in the subsequent iterations of the federated learning session.

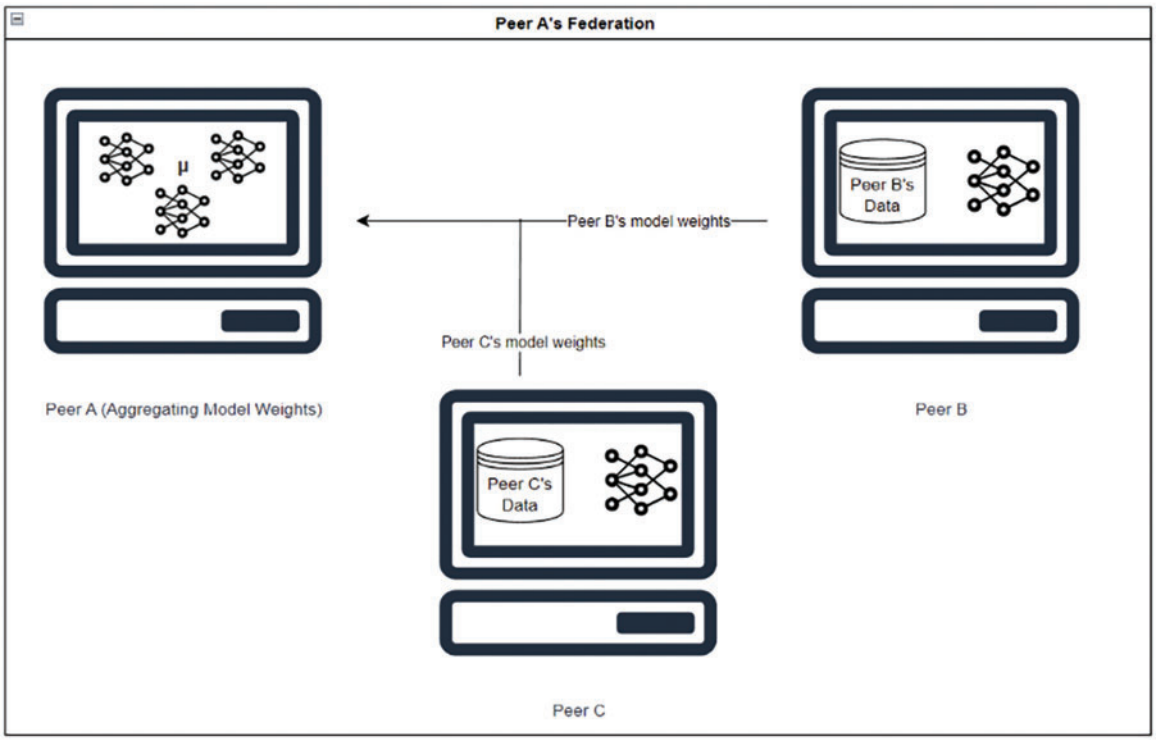

Figure 5: Weights transmission from Peers B and C to Peer A for aggregation

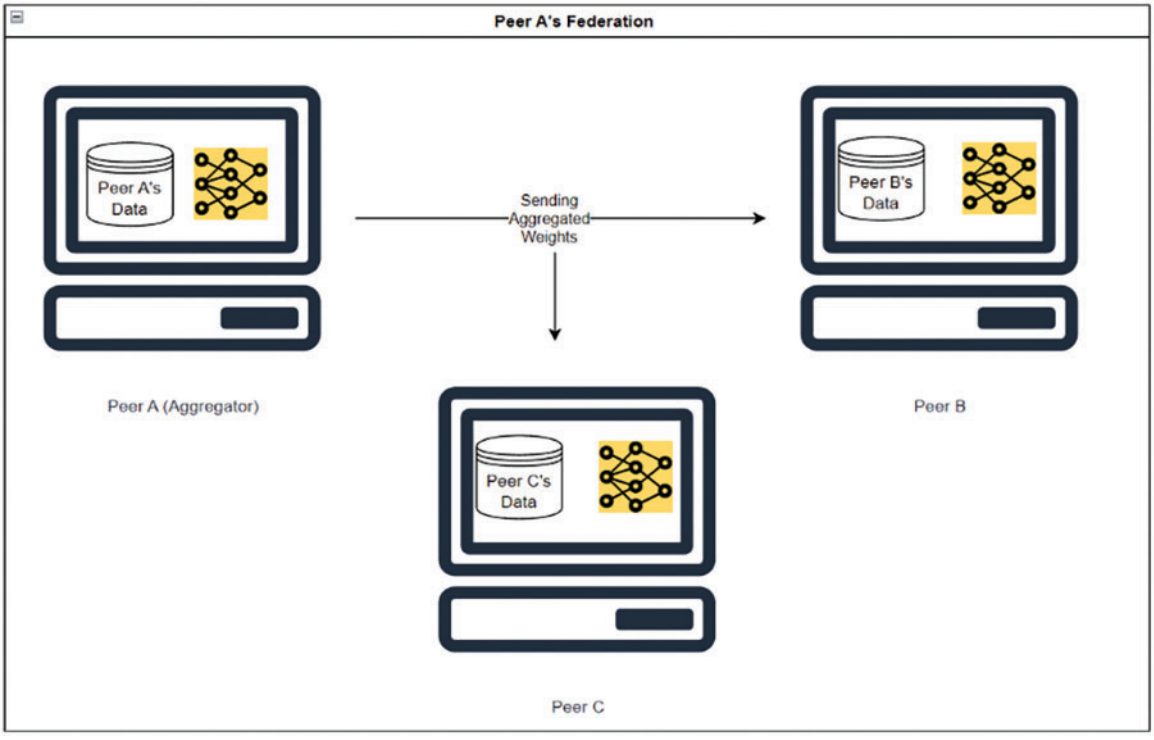

Figure 6: Transmission of aggregated weights from Peer A to Peers B and C

4.4 Overall WebFLex Workflow

In light of the details presented above, the proposed WebFLex framework is illustrated step-by-step in Algorithm 1. Moreover, this subsection offers a concise summary of the WebFLex workflow, outlining its various stages and steps in a detailed manner as follows.

In WebFLex, the initiator peer establishes connections with all peers listed in its address book. Subsequently, it establishes the parameters such as the total number of communication rounds (R), participating peers in each round (S), and epochs for each local training phase, and the process enters the communication loop. During each communication round, the initiator sends its current model weights to a random subset of peers. Each of these peers employs and loads the received weights into their respective local training model using their unique datasets. Then, there is a loop for each epoch of local training where the gradient is computed with the local data and the model is updated via gradient descent. After local training, to ensure privacy, Gaussian noise is integrated into their model weights before sending them back to the initiator. At the end of each round, the initiator aggregates the received weights from all the participating peers. The aggregated weights are then distributed to every peer listed in the address book. This process iteratively continues until all communication rounds are completed.

Algorithm 1: WebFLex

Input:
- AddressBook: Set of all peer IDs in the federation
- R: Number of communication rounds
- S: Number of peers sampled per round
- E: Number of epochs for local training per round

Procedure:

Initialize at the initiator peer:
Connect to peers in AddressBook
Set R, S, E
for r = 1 to R **do**
    SampledPeers = Randomly select S peers from AddressBook
    for each peer p in SampledPeers **do**
        Send current global model weights W_{global} to peer p
        At peer p:
            Load received weights W_{global} into local model
            for e = 1 to E **do**
                Compute gradient ∇W_p using local data D_p: $\nabla W_p = \eta.\nabla L(W_p, D_p)$
                Update local model using gradient descent: $W_p^{'} = W_p - \nabla W_p$
            end for
            Add Gaussian noise $N(0, \sigma^2)$ to local model weights $W_p^{'}$: $W_p^{"} = W_p^{'} + N(0, \sigma^2)$
            Send local model weights $W_p^{"}$ back to the initiator peer
        end for
    Back at the initiator peer:
    Aggregate weights from all sampled peers: $W_{global}^{'} = \frac{1}{S}\sum_{p=1}^{S} W_p^{"}$
    Update global model: $W_{global} = W_{global}^{'}$
    Distribute W_{global} to all peers in AddressBook
end for

5 Experimental Results

In this section, a series of experiments are conducted to evaluate the performance and effectiveness of training models based on the WebFLex framework. The experiments analyze two primary factors: Initially, the impact of integrating artificial Gaussian noise into the weights of local models, and subsequently, the performance evaluation of a trained CNN on the MNIST dataset [28]. Furthermore, this section provides extensive details about the experimental setup, including client device settings, dataset, and software libraries that are used in the training process, while also offering a comprehensive analysis of the successfully achieved results.

5.1 Experimental Setup

A wide variety of devices are illustrated to evaluate the training performance of a CNN. The client devices consist of two laptops, a smartphone, and a VR headset, which are employed to conduct the experiments. The hardware and software specifications of these devices, essential for ensuring accurate evaluation and analysis, are presented in Table 4. The selection of different devices demonstrates the ability of WebFLex to operate across heterogeneous devices with different types of GPUs, operating systems, and web browsers. Importantly, WebFLex allows for smooth deployment using the same codebase, eliminating complex driver configurations. Furthermore, a VR headset is incorporated into a group of devices to demonstrate the capabilities and versatility of the WebFLex framework. This choice may initially seem nontraditional because VR headset is usually associated with interactive gaming and experiences rather than computational tasks such as deep learning. Nevertheless, motivation stems from the presence of web browsers on VR headset, similar to numerous other devices. This not only illustrates WebFLex's compatibility with a wide range of devices, but it also demonstrates the rapidly changing environment of the Internet of Things (IoT). This flexibility opens the way for distributed learning in scenarios where data privacy is crucial and traditional data collection methods are impossible or expensive.

Table 4: Hardware and software specifications of the client nodes used for training

Type	Name	OS	Hardware	Browser
PC	Lenovo Ideapad	Windows 11	NVIDIA RTX 3050	Google chrome
PC	Lenovo Thinkpad	Windows 10	AMD Radeon Vega 6	Brave browser
Smartphone	Oneplus Nord	Oxygen OS	Qualcomm Adreno 620	Google chrome
VR Headset	Meta Quest 2	Android	Qualcomm Snapdragon XR2	Meta quest browser

The MNIST dataset is currently available to the public and has gained a common benchmark to evaluate the performance of different deep learning methods, in particular those employing CNNs. The MNIST database is comprised of a collection of 70,000 grayscale images of handwritten digit, subdivided into 60,000 images for training set and 10,000 images for testing set. This grayscale dataset offers standardized 28×28 pixel images representing digits from 0 to 9. The MNIST dataset is a popular choice for educational purposes in the field of computer vision and deep learning.

This study employs the Tensorflow.js and Peer.js JavaScript libraries to implement a federated neural network. Tensorflow.js is widely known for its capabilities in the creation, training, and deployment of deep learning models within web browser environments. On the other hand, Peer.js facilitates the establishment of peer-to-peer connections, enabling decentralized communication and coordination between devices by using WebRTC. These libraries enable the development of a federated

neural network system that facilitates the distributed training process across multiple devices while ensuring privacy and secure communication.

5.2 Results and Analysis

5.2.1 The Effect of Adding Noise to the Local Model Weights

One of the primary motivations for integrating artificial noise into local model weights is to achieve local differential privacy. This ensures that the source of the weights, when used for aggregation at the aggregator client, remains ambiguous, thereby offering plausible deniability. This modification aids in preventing membership inference attacks, where information about the training dataset can be extracted from the weights, potentially affecting the privacy of the local data used for training. To comprehensively analyze the influence of the noise amount added to weights on the performance of the model, we have conducted multiple training sessions, each characterized by different amounts of noise. The results of these training sessions are shown in Fig. 7.

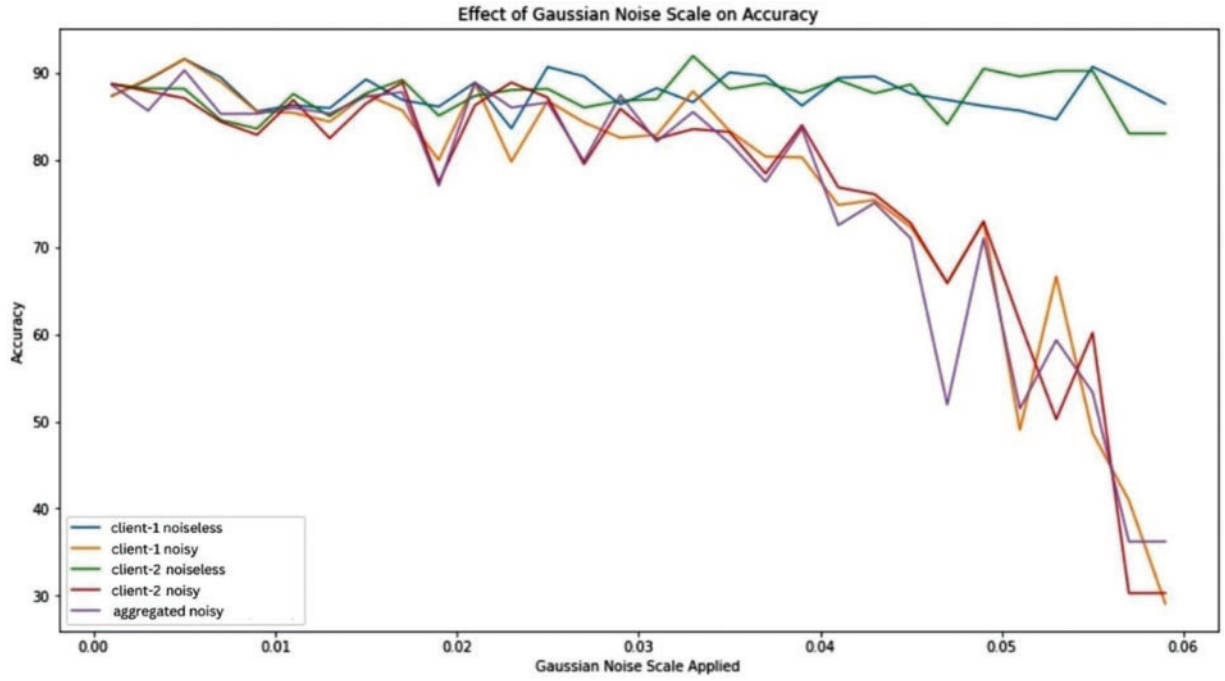

Figure 7: Effect of Gaussian noise scale on model accuracy

Fig. 7 presents the complex relationship between the scale of Gaussian noise added to model weights ranging from 0.00 to 0.06 and the subsequent impact on model accuracy. It represents various training conditions and shows unique behavioral trends. The accuracy of the noiseless client changes moderately around the upper accuracy levels, suggesting that the model performs consistently well in the absence of external noise. In contrast, the accuracy of the noisy client demonstrates more distinct variations as the noise scale increases, indicating the actual impacts of different noise levels on model performance. The aggregated impact of noise on model accuracy is clearly demonstrated during data aggregation from different sources. Finally, at lower Gaussian noise scales of approximately 0.00 to 0.03, the accuracy of all models seems to change moderately. While the noise scale increases, especially between 0.04 and 0.06, there is a clear decrease in the accuracy of the various models. Therefore, while

the addition of noise can improve data privacy and protect against potential inference attacks, it comes at the cost of decreased model accuracy and performance.

Furthermore, adding artificial noise to model weights affects the convergence of deep learning models as well as its impact on the performance of the aggregated model. To explore this aspect, we train two distinct models on the same dataset: One without any noise and the other with noise. The comparative results of these models are shown in Fig. 8.

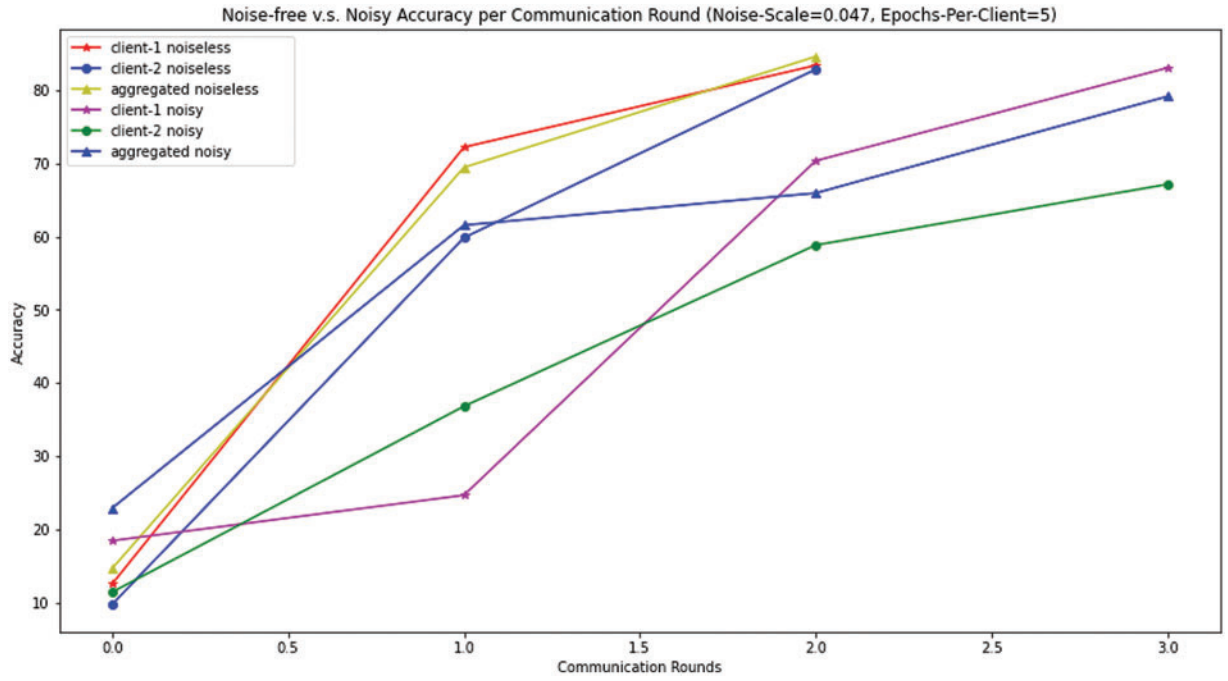

Figure 8: Noise-free *vs.* noisy accuracy per communication round (noise scale = 0.047, epochs per client = 5)

In Fig. 8a, comparative analysis of noiseless *vs.* noisy accuracy across communication rounds is presented, utilizing a noise scale of 0.047 and a fixed 5 epochs per client for both scenarios. Beginning with the noiseless conditions, the accuracy of both clients shows a significant upward trend as the communication rounds progress. Therefore, the aggregated performance of the noiseless model also follows a similar slope, indicating efficient aggregated learning from both clients. In contrast to the noisy conditions, the accuracy of both clients illustrates a slight increase in slope and appears to slow behind their noiseless equivalents. The impact of noise on individual client models indicates slower convergence or the need for additional rounds to achieve comparable accuracy. Therefore, the aggregated noisy accuracy follows a direction slightly parallel to the aggregated noiseless accuracy, but it remains consistently lower throughout the observed communication rounds. Finally, starting at round 0.0, noiseless and noisy models show low accuracy in the initial training stages. As rounds progress, noiseless models outperform noisy ones, illustrating the impact of noise on convergence. The accuracy of the aggregated noiseless model also increases significantly and outperforms the accuracy of the aggregated noisy model. After round 2.5, all models display stabilized accuracy growth, but the gap between noiseless and noisy models remains clear. Therefore, adding noise to local models can initially reduce model accuracy, but convergence is achievable after a longer training time. Thus, adjusting the noise amount is essential to align privacy with optimal performance.

5.2.2 Performance Evaluation of the CNN

In this research, we conduct experiments to evaluate the performance of a CNN trained using the WebFLex framework on the MNIST dataset. The experimental setup involves four devices to represent a communication network architecture for training the model, centralized around a brokering server as shown in Fig. 9. Four distinct devices, namely Meta Quest 2, Oneplus Nord, Lenovo Ideapad, and Lenovo Thinkpad, are interconnected through this server. Meta Quest 2 symbolizes VR technology, while Oneplus Nord represents a mobile platform, and the two Lenovos denote conventional computing devices. The primary role of the brokering server in this network indicates its significance in coordinating tasks, while the direct interconnections or peer-to-peer communication between certain devices also demonstrate the flexibility and versatility of the system for data sharing or load distribution.

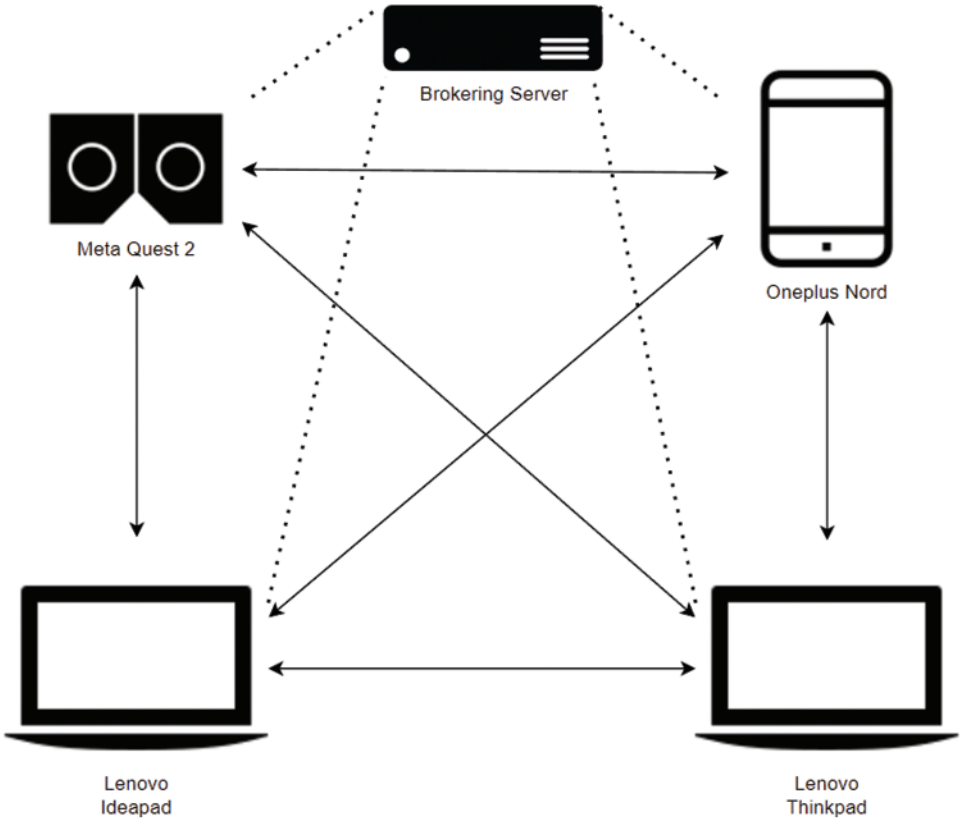

Figure 9: Distribution of devices in the WebFLex training process

To ensure effective training, we utilize a sampling rate of 0.5, meaning that during each communication round, two out of the four devices are randomly selected for training the CNN on a local subset of the MNIST dataset stored on each device. We set the number of epochs per communication round to 10, allowing for iterative improvement of the model parameters over 5 communication rounds. The results of this training session are analyzed and presented in Fig. 10 and Table 4.

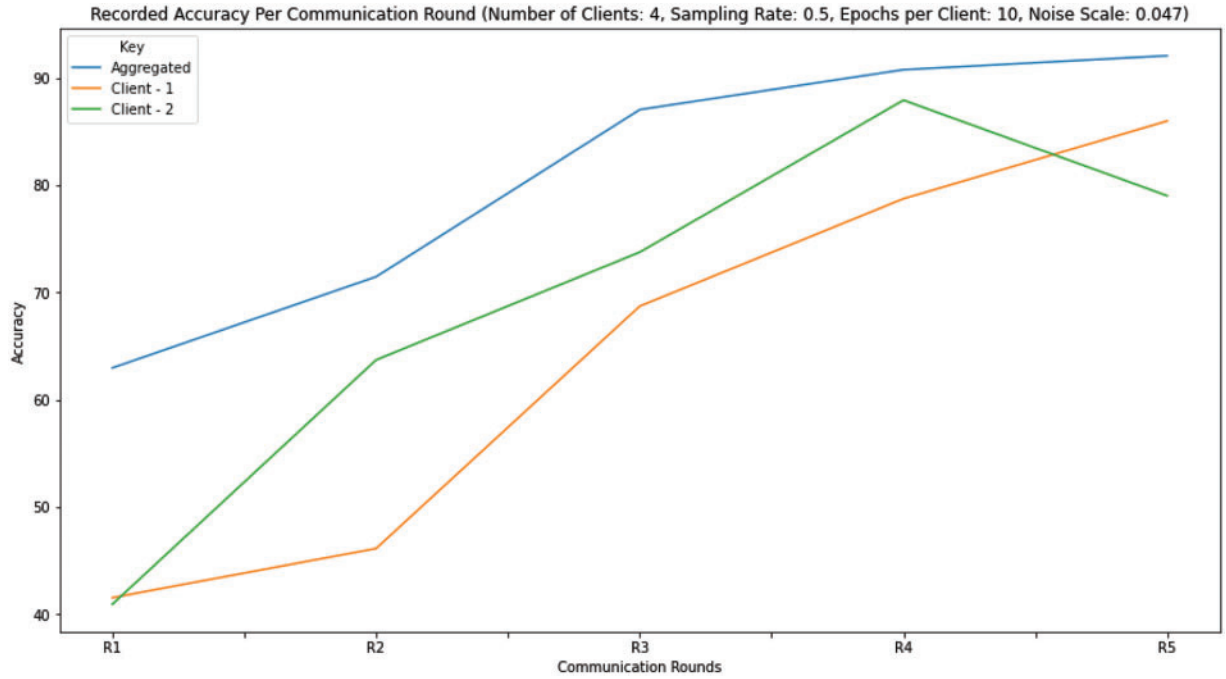

Figure 10: Accuracy rate in each communication round

Fig. 10 defines the accuracy rate over five communication rounds, from R1 to R5. The first client shows a sharp and significant initial increase in accuracy until it reaches its peak at R3, achieving an accuracy of just over 70%. However, between R3 and R5, the accuracy rises very slowly, achieving an accuracy of just over 80% by R5. This can be due to model overfitting, changes in the nature of the data, or perhaps effects resulting from the addition of noise. While the second client starts at just over 40% accuracy, model performance increases steadily, exceeding 85% at the R4 peak. However, it is interesting that the model sees a drop from R4 to R5 until the accuracy reaches 80%, indicating a possible challenge in data quality or model adaptation in the final stages. In contrast, the aggregated accuracy starts just above 60% and shows a consistent increase across the communication rounds, with a minor inflection between R3 and R4, reaching close to 90% by R5. This demonstrates the power of aggregating individual client models to improve performance.

Table 5 offers the interactions observed over five distinct communication rounds within a federated learning framework. It illustrates the specific clients selected for each round, identifies their corresponding training durations, and provides individual accuracies, which subsequently lead to the aggregated model accuracy. Firstly, the frequent selection of the Lenovo Ideapad can be observed during the first two communication rounds. This device registers a training time of 35 s in the first round and shows a small decrease to 28 s in the second round. Concurrently, Lenovo Thinkpad becomes the second client, with training times of 71 and 52 s in the first and second rounds, respectively. The accuracy of the first client remains relatively stable at around 40%, while the second client shows a significant improvement from 40.96% to 63.72%, but its training time is relatively longer. Furthermore, the aggregated model accuracy in the first two rounds improves from 62.98% to 71.47%, indicating an effective aggregation of individual client performances.

Table 5: Summary of accuracy rate in each communication round

Communication round	Selected client 1	Client 1's training time (s)	Selected client 2	Client 2's training time (s)	Client 1's accuracy (%)	Client 2's accuracy (%)	Aggregated model accuracy (%)
1	Lenovo Ideapad	35	Lenovo Thinkpad	71	41.57	40.96	62.98
2	Lenovo Ideapad	28	Lenovo Thinkpad	52	46.15	63.72	71.47
3	Meta Quest 2	56	Oneplus Nord	173	68.73	73.76	87.04
4	Meta Quest 2	67	Oneplus Nord	158	78.74	87.92	90.76
5	Meta Quest 2	65	Lenovo Ideapad	26	85.98	79.02	92.06

In the third and fourth communication rounds, there is a clear shift in the client setting with the use of Meta Quest 2 and Oneplus Nord. Meta Quest 2 demonstrates a relatively consistent training duration of 56 to 67 s. In contrast, the Oneplus Nord presents a significantly longer training time, peaking at 173 s in the third round. Despite the increased training duration, the Oneplus Nord demonstrates extremely high accuracy, rising from 73.76% in the third round to 87.92% in the fourth round. The aggregated model accuracy reflects this upward trend, registering a jump from 87.04% to 90.76% across these rounds.

The fifth and final round introduces an interesting contrast: Meta Quest 2 remains consistent in its role, while Lenovo Ideapad reappears as the second client. The Lenovo Ideapad requires just 26 s for training, aligning with its initial efficiency. Accuracies for both clients remain around the high 70 s to mid-80 s, with the aggregated model further combining its robustness at 92.06%.

Therefore, the results of Table 5 demonstrate the difficulties related to federated learning, highlighting the interconnected relationship between device capabilities, training durations, and resultant accuracies. The variety of clients, including laptops, VR devices, and smartphones, confirms the heterogeneous nature of real-world distributed training environments. Moreover, the consistent increase in aggregated model accuracy, even between the different individual client performances, provides evidence of the robustness and possibility of collaborative model training within federated systems.

5.2.3 Comparative Analysis

According to the comprehensive experimental results presented and related works [5–7,11,16,18,22], the WebFLex framework can be critically compared to the general characteristics of traditional centralized and decentralized federated learning frameworks. The comparison focuses on key aspects such as communication architecture, deployment, compatibility, scalability, privacy, accuracy, disconnectivity, and convergence speed. Additionally, Table 6 provides a comprehensive comparison of the features and characteristics of each framework to facilitate a detailed analysis.

Table 6: WebFLex *vs.* traditional federated learning frameworks

Feature/Aspect	WebFLex framework	Centralized framework	Decentralized framework
Communication architecture	**Decentralized** Hybrid peer-to-peer and central server for coordination	**Centralized** All communication goes through a central server	**Decentralized** Peer-to-peer communication without a central server
Deployment ease	**High** Due to utilizing web browsers	**Low** Due to server and maintenance	**High** Due to peer-to-peer communication setup
Device compatibility	**High** Uses heterogeneous devices via web browsers	**Low** Requires specific infrastructure	**High** Adaptable to various devices
Scalability	**High** Due to decentralized and absence of central server	**Low** Due to resource limited by server capacity	**High** Due to peer-to-peer architecture
Privacy preservation	**High** Due to noise addition and peer-to-peer connection	**Low** Due to central server	**High** Due to peer-to-peer connection
Model accuracy	**Medium to high** Initially reduced due to noise, increases over time	**High** Beneficial from centralized and consistent data	**Medium to high** Depending on network structure and device capabilities
Robustness to disconnection	**High** Maintains session continuity despite peer disconnections	**Medium** May face challenges if the central server is hacked	**High** Robust to individual peer disconnections
Model convergence speed	**Slow** Due to noise, requires more rounds for convergence	**Fast** Due to centralized control	**Slow** Due to decentralized nature, more rounds may be needed

The WebFLex framework presents a competitive approach in the domain of federated learning, demonstrating notable strengths and some limitations when compared to traditional centralized and decentralized frameworks. The communication architecture is decentralized, incorporating both peer-to-peer communications and a central server for task coordination. This architecture offers a balance between the flexibility of decentralized frameworks and the structured oversight of centralized

frameworks. This hybrid framework ensures effective communication and collaboration across heterogeneous nodes while maintaining a level of centralized coordination. The ease of deployment is a significant advantage of WebFLex, which uses the accessibility of web browsers for cross-platform compatibility. The WebFLex framework contrasts sharply with the traditional centralized framework, which requires specific server setups and ongoing maintenance. Decentralized frameworks, similar to WebFLex, also benefit from a high degree of deployment ease due to their reliance on peer-to-peer communication setups.

Regarding device compatibility and scalability, WebFLex stands out due to its ability to integrate heterogeneous devices through web browser compatibility. This attribute is notably limited in centralized systems due to their reliance on specific infrastructure. The previous feature of WebFLex aligns with the current trend towards the IoT, where various devices with different capabilities are required to be accommodated in learning networks. Scalability is another area where WebFLex succeeds, due to its decentralized nature and the absence of a central server that could become a bottleneck. This aspect is in sharp contrast to centralized frameworks, which often have difficulties scaling due to resource limitations imposed by server capacity. Decentralized frameworks share this advantage with WebFLex, benefiting from their inherent peer-to-peer architecture that naturally supports scalability.

However, the trade-offs in the WebFLex framework become evident when considering aspects such as model accuracy and convergence speed. The integration of noise for privacy preservation, while enhancing the privacy aspect of the framework, initially reduces model accuracy. This accuracy does improve over time with continued training, but it highlights the inherent balance between privacy and performance in such systems. Centralized frameworks generally exhibit higher model accuracy due to the consistency and quality of the centralized data. In contrast, decentralized frameworks show a variable range of model accuracy dependent on network structure and device capabilities. The convergence speed of models in the WebFLex framework is slower due to the added noise and the need for more rounds of communication for convergence, a common characteristic shared with decentralized frameworks. Centralized systems, on the other hand, benefit from faster model convergence owing to their centralized control and uniform data processing. This comparison analysis highlights the significance of carefully considering the specific requirements and constraints of a learning task when choosing between these federated learning frameworks.

A comparison of the WebFLex and TFF (TensorFlow Federated) frameworks is shown in Fig. 11. It shows the comparison with and without artificial noise added over several communication rounds to improve local differential privacy. Initially, it is clear that both noiseless frameworks of WebFLex and TFF show a relatively steady and rapid rise in accuracy as the number of communication rounds increases, with TFF noiseless maintaining a slight advantage over WebFLex noiseless.

However, when noise is introduced, we observe a noticeable difference in the performance of both frameworks. The TFF noisy framework demonstrates a significantly changing accuracy rate, with a significant drop occurring in the middle of the communication rounds. This suggests that the mechanism for integrating noise in TFF may introduce a level of uncertainty or instability in the training process, which impacts the ability to learn consistently over time. In contrast, the WebFLex noisy framework's accuracy, while starting lower than its noiseless counterpart, shows a more gradual and stable increase in accuracy without significant variations. This can indicate a more effective strategy within WebFLex for dealing with the noise or a more robust architecture that reduces the effects of noise on performance and further improves privacy.

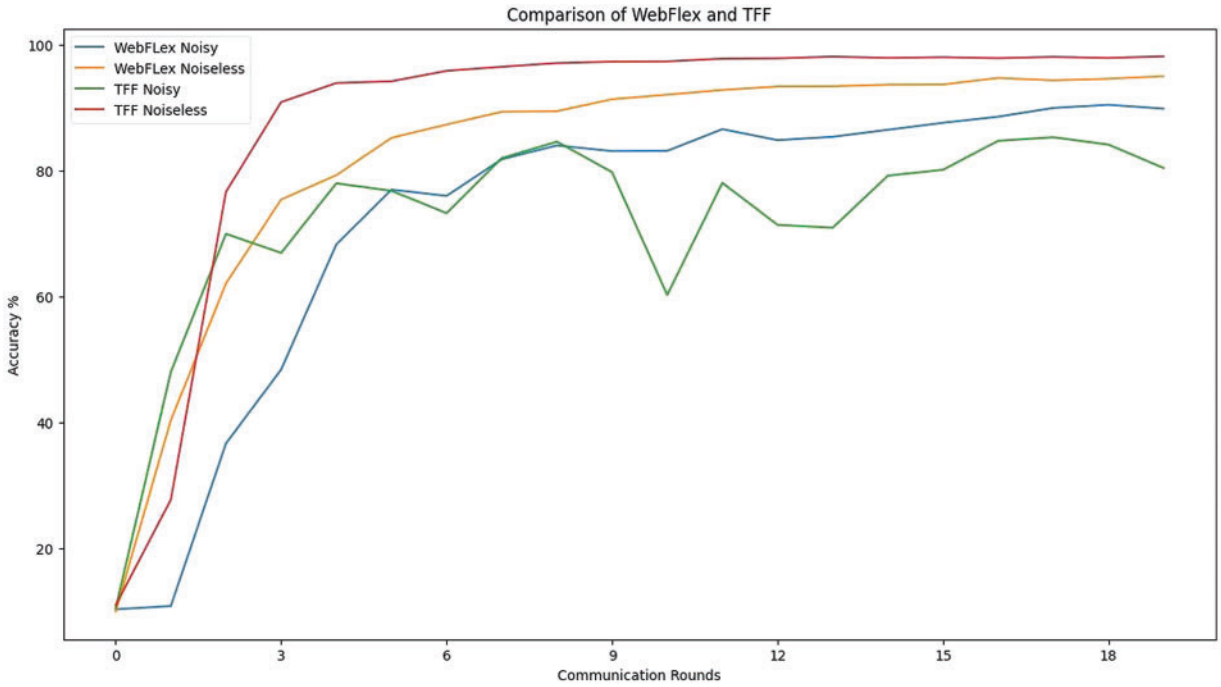

Figure 11: Comparative analysis of noisy and noiseless Federated Learning frameworks

Finally, as the number of communication rounds increases, the noisy frameworks begin to converge towards higher accuracy levels. The convergence is more clear in the WebFLex framework compared to TFF, indicating a potential for the WebFLex framework to better adapt to the noise over time. Therefore, WebFLex demonstrates a desirable balance between privacy and performance, with a promising convergence pattern that suggests effective adaptation to privacy constraints over time.

6 Discussion

In this section, we discuss a number of significant findings that offer valuable insights into the performance and efficacy of the WebFLex framework. The comprehensive analysis of performance accuracy enables us to evaluate the influence of the WebFLex framework, facilitating a deeper understanding of the framework's effectiveness in training models using decentralized data sources.

On the one hand, the results of this study provide implications for adding artificial noise to enhance privacy through local differential privacy. It is evident that while the addition of noise brings advantages in terms of privacy preservation, it negatively affects the performance of the model. Although the noiseless model demonstrates higher accuracy in the short term, the model employing local differential privacy tends to converge over time, even if it takes a longer training period. Therefore, it is critical to carefully adjust the amount of noise when using WebFLex to understand and balance the trade-off between data privacy and model performance.

On the other hand, this study demonstrates the effectiveness of utilizing the WebFLex framework for training deep CNNs across a different range of devices in a peer-to-peer manner. The results indicate that the computational power of each device significantly influences the efficiency of the training process. It is worth noting that when a smartphone is randomly selected as a training node, it takes longer training times to complete 10 epochs in each communication round. This variation in

computational power acts as a bottleneck, negatively impacting the overall performance and training time of the session. The longer training times may result in increased latency and slower convergence rates. Consequently, it becomes essential to carefully manage the distribution of training tasks among devices to optimize the training process and minimize the impact of computational variation.

The proposed WebFLex framework offers several advantages for constructing federated learning systems. These advantages are detailed as follows:

1) Cross-Platform Compatibility and Ease of Setup:
   WebFLex utilizes the inherent cross-platform compatibility of web browsers, eliminating the need to deal with compatibility issues across different devices. Furthermore, the framework utilizes WebGL, which allows the utilization of a device's graphics processing card without demanding complex driver setup processes. This feature not only simplifies the deployment of the framework but also enhances its performance by using the computational capabilities of the device's graphics card.
2) Robustness to Peer Disconnection:
   WebFLex framework ensures the continuity of federated learning sessions even though some participating peers go offline during the session. This is achieved through the implementation of event-driven gossip communication, which enables reliable correspondence between peers in cases where direct communication cannot be guaranteed. By allowing peers to continue their participation even in the absence of certain peers, WebFLex enhances the robustness and flexibility of the federated learning process.
3) Decentralized Communication and Scalability:
   The decentralized communication architecture is another advantage of the WebFLex framework. While a central server is employed for establishing initial connections between peers over a network, it is not involved in aggregating model weights from multiple clients or acting as a relay point for communication between clients. This decentralization enables easy scalability without concerns about overloading the central server. It allows efficient communication between many clients, facilitating the scaling of federated learning systems to accommodate a growing number of participants.
4) Simplified Distribution and Participation:
   WebFLex further simplifies the distribution of the federated learning framework. Participants only need to download the required HTML and JavaScript files from a content delivery network to join a federation. This simplified distribution process reduces the limitations on entry and enables broader participation in federated learning. It enhances accessibility and facilitates the adoption of the WebFLex framework across various devices and platforms.

Although there are several advantages that support the performance and efficiency of the WebFLex framework, it also presents some limitations and challenges, which are discussed as follows:

1) Vulnerability to Model-Inversion Attacks:
   One notable disadvantage of the WebFLex system is its dependence on trusted peers to prevent vulnerabilities to model-inversion attacks. Due to the nature of the framework, where any peer can initiate a federated learning session, there is a potential risk of exposing model weights generated by other peers during the local update step. To reduce this risk, it is necessary to ensure the integrity and reliability of participating peers to prevent unauthorized access and the potential limitation of sensitive information.

2) Limitations of Network Address Translation (NAT) Firewalls:
   While WebRTC enables serverless peer-to-peer connections between browsers within the same local network, it encounters limitations when attempting to establish connections between devices on different networks over the internet. NAT firewalls commonly found in routers restrict connections from external devices. To overcome this limitation, the use of session traversal utilities for NAT (STUN) or traversal using relays around NAT (TURN) servers becomes necessary. This dependence on external servers introduces additional complexity and potential points of failure in the WebFLex system.
3) Susceptibility to Model Degradation:
   Peers in the WebFLex framework are susceptible to model degradation if they are not consistently online or active participants in the federated learning process. As the WebFLex framework relies on collaborative updates from multiple peers, the absence of certain peers for extended periods can lead to a degradation in model performance over time.
4) Latency Issues:
   A primary limitation of WebFLex is latency, especially in real-time applications. Federated learning is decentralized, which can delay data transmission and model updates, reducing learning efficiency.
5) Network Robustness and Reliability:
   The dependence on network connectivity can be a bottleneck. Interruptions or inconsistencies in network quality can adversely affect the learning process.

The broader impact of the WebFLex framework extends significantly into various real-world scenarios, particularly where data privacy and decentralized processing are extremely important. In the healthcare sector, for instance, WebFLex can facilitate collaborative research and data analysis while maintaining patient confidentiality. By enabling different healthcare providers to train models on local datasets without sharing sensitive patient information, WebFLex ensures a high level of data privacy. This approach could revolutionize medical research, allowing for the development of more accurate diagnostic tools and personalized treatment plans based on a diverse range of data sources. Furthermore, in the world of smart cities and IoT environments, WebFLex could be instrumental in processing vast amounts of data generated by various devices. By distributing the computation load across numerous nodes, the framework can efficiently process and analyze data for urban planning, environmental monitoring, and resource management, while ensuring that individual data sources retain their privacy. Similarly, in the finance and retail sectors, WebFLex could enable businesses to harness decentralized data sources for predictive analytics without compromising user confidentiality. The WebFLex framework's ability to operate across different devices and platforms also holds significant promise for applications in remote education and collaborative research, where participants can contribute to collective learning endeavors irrespective of their hardware capabilities.

WebFLex could significantly influence the development of decentralized learning systems. Its approach to handling heterogeneous devices and its robustness against connectivity issues make it a template for creating flexible and scalable learning networks. Moreover, the emphasis on privacy and security in the WebFLex framework aligns with the growing global concern for data protection, potentially setting a new standard for privacy-preserving distributed computing. In essence, WebFLex stands as a harbinger for a new era in federated learning, where the balance between data utility and privacy is meticulously maintained, inspiring future frameworks to adopt a similarly nuanced approach to decentralized learning.

7 Practical Implementation Challenges of WebFLex

The practical implementation of the WebFLex framework presents several challenges that are critical to address for its successful deployment. These challenges primarily stem from the complexities inherent in a web-based, decentralized learning environment, and the diverse nature of the devices and platforms involved, as follows:

1) Browser Compatibility: A fundamental challenge lies in ensuring consistent functionality across different web browsers. Given the diversity in browser architectures and their interpretation of web standards, particularly for complex JavaScript operations, WebFLex must maintain compatibility without affecting performance. This necessitates rigorous testing and adaptation to various browser behaviors, especially when using advanced libraries like TensorFlow.js and Peer.js.
2) Device Heterogeneity: The operational versatility of the WebFLex framework across diverse devices, such as laptops, smartphones, and VR headsets, introduces the complexity of device heterogeneity. This variety, although beneficial for inclusivity and accessibility, leads to differences in computational power and processing speeds. These differences can lead to inconsistent training times and possible bottlenecks, especially when less powerful devices are utilized in the federated learning network.
3) Network Reliability and Connectivity: It is essential to guarantee stable and secure peer-to-peer connections, particularly when dealing with varying network conditions. The framework must be robust against changes in internet connectivity and capable of handling data transmissions efficiently across potentially unstable or slow connections.
4) Security and Privacy Concerns: While WebFLex aims to enhance data privacy through decentralized learning, it still faces challenges in safeguarding against potential security breaches, such as model-inversion attacks. Protecting sensitive data during transmission and ensuring the integrity of participating peers are ongoing critical concerns.
5) Scalability and Performance Optimization: Ensuring scalability while preserving performance is an important challenge as the network expands. Balancing the load across diverse devices, optimizing training times, and ensuring efficient aggregation of learning models are key aspects that need continuous refinement.

8 Suggestions for the Future

This section provides some recommendations for innovative and impactful research in the field of federated learning. It is essential to address latency issues, especially in real-time applications, where delays in data transmission and model updates may limit process efficiency. Improving data compression techniques and optimizing network protocols are key areas of focus. Enhancing network robustness and reliability is also vital, as the learning process is sensitive to network quality changes. Developing more resilient network architectures and fault-tolerant systems is necessary.

Scalability remains a challenge in managing large and diverse networks. Future studies should investigate scalable architectures and algorithms that can handle synchronization, resource allocation, and model aggregation effectively. Additionally, the diversity and varying quality of the data pose challenges for consistent model training. Advanced data preprocessing and innovative training techniques need exploration to accommodate this heterogeneity.

Security and privacy are paramount, with risks of data leakage and breaches present. Research should investigate advanced cryptographic techniques and secure computation methods to strengthen these aspects. Energy efficiency is another concern, especially in mobile and IoT contexts. Developing

energy-efficient learning algorithms and system optimizations is critical for reducing energy consumption.

9 Conclusion

This paper introduces WebFLex, a novel framework for training deep neural networks that utilizes browser-to-browser peer-to-peer connections to enhance the decentralization of the federated training process. WebFLex allows for increased scalability and flexibility in the training process because it does not rely on the need for an external server beyond establishing the initial peer connection. Moreover, the WebFLex framework depends on direct connections between peers, allowing for easy distribution and compatibility across different platforms. The study highlights the trade-off between data privacy, achieved through artificial noise via local differential privacy, and model performance within the WebFLex framework. Despite initial superior accuracy from noiseless models, those incorporating differential privacy converge over long periods. Device computational power distinctly influences training efficiency, with variations causing bottlenecks, longer training times, and latency. Thus, careful task distribution across devices is essential to improve training and reduce computational variances.

Acknowledgement: The authors extend their appreciation to King Saud University for funding this research work.

Funding Statement: This work has been funded by King Saud University, Riyadh, Saudi Arabia, through Researchers Supporting Project Number (RSPD2024R857).

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Mai Alzamel, Isra Al-Turaiki; data collection: Hamza Ali Rizvi, Najwa Altwaijry; analysis and interpretation of results: Mai Alzamel, Isra Al-Turaiki, Hamza Ali Rizvi; draft manuscript preparation: Mai Alzamel, Najwa Altwaijry. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, M. A., upon reasonable request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Kletz, M. Bertini, and M. Lux, "Open source column: Deep learning in the browser: TensorFlow JS," *ACM SIGMulti. Records*, vol. 11, no. 1, pp. 1, 2021.

[2] C. Huang, G. Xu, S. Chen, W. Zhou, E. Y. Ng and V. H. C. de Albuquerque, "An improved federated learning approach enhanced internet of health things framework for private decentralized distributed data," *Inf. Sci.*, vol. 614, no. 15, pp. 138–152, 2022. doi: 10.1016/j.ins.2022.10.011.

[3] M. Gheisari *et al.*, "Deep learning: Applications, architectures, models, tools, and frameworks: A comprehensive survey," *CAAI Trans. Intell. Technol.*, vol. 8, no. 3, pp. 581–606, 2023. doi: 10.1049/cit2.12180.

[4] R. N. Bhimanpallewar, S. I. Khan, K. B. Raj, K. Gulati, N. Bhasin and R. Raj, "Federate learning on Web browsing data with statically and machine learning technique," *Int. J. Pervas. Comput. Commun.*, vol. 18, no. 1, pp. 1416, 2022. doi: 10.1108/IJPCC-05-2022-0184.

[5] Q. Li *et al.*, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3347–3366, 2023. doi: 10.1109/TKDE.2021.3124599.
[6] K. J. Rahman *et al.*, "Challenges, applications and design aspects of federated learning: A survey," *IEEE Access*, vol. 9, pp. 124682–124700, 2021. doi: 10.1109/ACCESS.2021.3111118.
[7] P. M. Mammen, "Federated learning: Opportunities and challenges," 2021. arXiv preprint arXiv:2101.05428.
[8] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019. doi: 10.1145/3339474.
[9] M. Moshawrab, M. Adda, A. Bouzouane, H. Ibrahim, and A. Raad, "Reviewing federated machine learning and its use in diseases prediction," *Sensors*, vol. 23, no. 4, pp. 2112, 2023. doi: 10.3390/s23042112.
[10] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021. doi: 10.1561/2200000083.
[11] B. Liu, N. Lv, Y. Guo, and Y. Li, "Recent advances on federated learning: A systematic survey," 2023. arXiv preprint arXiv:2301.01299.
[12] W. Yang, Y. Zhang, K. Ye, L. Li, and C. Z. Xu, "FFD: A federated learning based method for credit card fraud detection," in *Big Data-BigData 2019: 8th Int. Congr., Held as Part of the Services Conf. Federation, SCF 2019*, San Diego, CA, USA, Springer, 2019, pp. 18–32.
[13] R. Kerkouche, G. Acs, C. Castelluccia, and P. Genevès, "Privacy-preserving and bandwidth-efficient federated learning: An application to in-hospital mortality prediction," in *Proc. Conf. Health, Inference, and Learning*, USA, 2021, pp. 25–35.
[14] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artif. Intell. Stat.*, Fort Lauderdale, Florida, USA, vol. 54, 2017, pp. 1273–1282.
[15] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Third Workshop on Bayesian Deep Learning (NeurIPS)*, Montréal, Canada, vol. 2, 2018, pp. 1–9.
[16] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4289–4301, 2022. doi: 10.1109/TPAMI.2022.3196503.
[17] 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, 2020. doi: 10.1109/MSP.2020.2975749.
[18] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, no. 1, pp. 106775, 2021. doi: 10.1016/j.knosys.2021.106775.
[19] C. Li, G. Li, and P. K. Varshney, "Decentralized federated learning via mutual knowledge transfer," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1136–1147, 2021. doi: 10.1109/JIOT.2021.3078543.
[20] Z. Chen, W. Liao, P. Tian, Q. Wang, and W. Yu, "A fairness-aware peer-to-peer decentralized learning framework with heterogeneous devices," *Future Int.*, vol. 14, no. 5, pp. 138, 2022. doi: 10.3390/fi14050138.
[21] P. Ramanan and K. Nakayama, "Baffle: Blockchain based aggregator free federated learning," in *IEEE Int. Conf. Blockchain (Blockchain)*, Rhodes Island, Greece, 2020, pp. 72–81.
[22] Z. Lian, Q. Yang, Q. Zeng, and C. Su, "Webfed: Cross-platform federated learning framework based on web browser with local differential privacy," in *ICC 2022-IEEE Int. Conf. Commun.*, Seoul, Republic of Korea. IEEE, 2022, pp. 2071–2076.
[23] V. Mugunthan, A. Peraire-Bueno, and L. Kagal, "Privacyfl: A simulator for privacy-preserving and secure federated learning," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.*, Ireland, 2020, pp. 3085–3092.
[24] A. Yousafzai, L. U. Khan, U. Majeed, O. Hakeem, and C. S. Hong, "FedMarket: A cryptocurrency driven marketplace for mobile federated learning services," *IEEE Access*, vol. 10, pp. 87602–87616, 2022. doi: 10.1109/ACCESS.2022.3199369.
[25] J. Shin *et al.*, "FedTherapist: Mental health monitoring with user-generated linguistic expressions on smartphones via federated learning," in *Proc. 2023 Conf. Emp. Methods Natural Lang. Process.*, Singapore. Association for Computational Linguistics, 2023, pp. 11971–11988. doi: 10.18653/v1/2023.emnlp-main.734.

[26] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," 2016. arXiv preprint arXiv:1610.05755.
[27] Q. Zhang, J. Ma, J. Lou, L. Xiong, and X. Jiang, "Private semi-supervised knowledge transfer for deep learning from noisy labels," 2022. arXiv preprint arXiv:2211.01628.
[28] L. Deng, "The MNIST database of handwritten digit images for machine learning research [Best of the Web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012. doi: 10.1109/MSP.2012.2211477.