


Article

Edge Federated Optimization for Heterogeneous Data

Hsin-Tung Lin and Chih-Yu Wen * 

Department of Electrical Engineering, National Chung Hsing University, Taichung 40227, Taiwan; g110064764@mail.nchu.edu.tw

* Correspondence: cwen@dragon.nchu.edu.tw; Tel.: +886-04-2285-1549

Abstract: This study focuses on optimizing federated learning in heterogeneous data environments. We implement the FedProx and a baseline algorithm (i.e., the FedAvg) with advanced optimization strategies to tackle non-IID data issues in distributed learning. Model freezing and pruning techniques are explored to showcase the effective operations of deep learning models on resource-constrained edge devices. Experimental results show that at a pruning rate of 10%, the FedProx with structured pruning in the MIT-BIH and ST databases achieved the best F1 scores, reaching 96.01% and 77.81%, respectively, which achieves a good balance between system efficiency and model accuracy compared to those of the FedProx with the original configuration, reaching F1 scores of 66.12% and 89.90%, respectively. Similarly, with layer freezing technique, unstructured pruning method, and a pruning rate of 20%, the FedAvg algorithm effectively balances classification performance and degradation of pruned model accuracy, achieving F1 scores of 88.75% and 72.75%, respectively, compared to those of the FedAvg with the original configuration, reaching 56.82% and 85.80%, respectively. By adopting model optimization strategies, a practical solution is developed for deploying complex models in edge federated learning, vital for its efficient implementation.

Keywords: federated learning; deep learning; embedded systems; heterogeneous; pruning



Citation: Lin, H.-T.; Wen, C.-Y. Edge Federated Optimization for Heterogeneous Data. *Future Internet* **2024**, *16*, 142. <https://doi.org/10.3390/fi16040142>

Academic Editors: Luis Javier García Villalba and Zheng Li

Received: 14 March 2024

Revised: 17 April 2024

Accepted: 19 April 2024

Published: 22 April 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the rapid advancements in data science and artificial intelligence technologies, machine learning has found widespread applications in various fields. For instance, in the healthcare domain, machine learning techniques are gradually being integrated into the healthcare diagnostic workflow, particularly in the analysis of medical images and clinical data to assist physicians in making more precise diagnoses. However, with the exponential growth of data, traditional centralized machine learning approaches face significant challenges in handling sensitive medical data, especially in terms of privacy protection and data storage. Given the abundance of personal health information in medical data, ensuring patient privacy becomes of paramount importance.

In this context, federated learning, as an emerging distributed machine learning approach, demonstrates its unique advantages in handling sensitive data. By conducting model training locally and sharing only the model updates rather than raw data, federated learning can simultaneously ensure data privacy and enable effective learning. This approach is particularly suitable for applications in the healthcare domain, as it allows learning from medical institutions distributed across various locations without the need for centralized storage of sensitive data.

In the implementation of federated learning, it is generally expected that the datasets among nodes are IID. However, the training data collected by nodes often exhibits non-IID characteristics, posing challenges to traditional federated learning approaches. To address this, federated learning has been further subdivided into Horizontal Federated Learning (HFL) (Figure 1a) and Vertical Federated Learning (VFL) (Figure 1b).

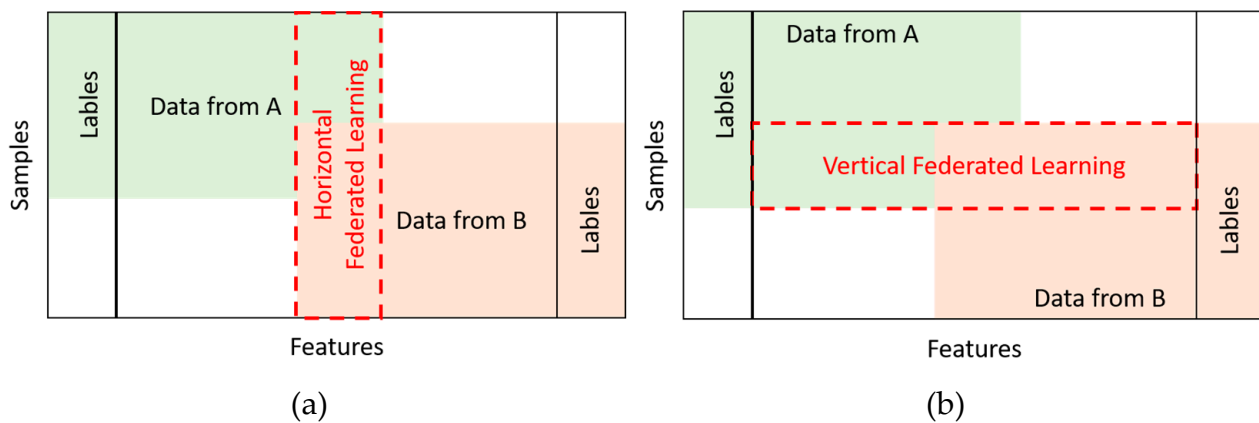


Figure 1. The comparison of (a) horizontal and (b) vertical federated learning (modified and reproduced from [1]).

In HFL, there is a high overlap in the feature space, but the sample identities are different. An example can be seen in different hospitals having patients with similar health data features. HFL allows these hospitals to share the knowledge of the learned model while protecting individual privacy, facilitating cross-institutional collaborative learning. On the other hand, VFL is applicable in scenarios where the same sample ID space is shared but the feature spaces are different, such as the common customer groups among different banks in the same region. This enables banks to jointly build comprehensive credit assessment models without directly sharing detailed customer information [1].

Nevertheless, federated learning still faces a series of challenges in practical applications, especially the issue of data heterogeneity. Heterogeneous data distributions, such as the non-independently and identically distributed (non-IID) phenomenon, can have adverse effects on the accuracy and stability of models. Referring to [2], the main approaches to dealing with non-IID problems in HFL can be categorized from data, algorithm, and system perspectives. With data-based approaches, this issue can be addressed by modifying the distributions (e.g., data sharing [3] and augmentation [4]), which significantly improves the learning performance. Nonetheless, most of these techniques can only be implemented with the help of the aforementioned data sharing, which may increase the risk of data privacy leakage. With algorithm-based approaches, personalization approaches may be applied to adjust the model according to the local tasks, including performing local fine-tuning [5–7] (e.g., personalization via regularization and interpolation, or meta learning), having personalized layers in the neural network models [8], multi-task learning [9–11], and knowledge distillation [12–15]. Note that although personalization layers are able to enhance the learning performance on non-IID data and reduce communication costs by only sharing the base layers between the server and clients, each client may need to permanently store the personalization layers without releasing them. With system-based approaches, client clustering is proposed to construct a multi-center framework by grouping those clients with similar local training data into the same cluster, introducing the concept of secure data similarity evaluation with respect to the loss value [16–19] and model weights [11,20,21]. However, these methods may consume additional computation and communication resources for both model training and testing.

This work applies an algorithm-based approach and focuses on exploring how to address the problem of data heterogeneity in federated learning by leveraging the FedProx algorithm [22] along with advanced optimization techniques. The FedProx algorithm effectively handles heterogeneous datasets by introducing additional local update steps, thereby enhancing the model’s generalization ability across different data distributions. The research also focuses on the feasibility of implementing federated learning models in resource-constrained environments, utilizing the Nvidia Jetson Nano as the experimental platform. Its balanced performance and computational capabilities make it an ideal choice

for edge computing applications. Leveraging the Flower federated learning framework, this study effectively deploys and manages federated learning tasks on the Jetson Nano. Flower, being a flexible and lightweight open-source federated learning framework, is well-suited for running on embedded systems like the Jetson Nano. It enhances the efficiency of coordinating model training across multiple devices while reducing communication costs and computational resource consumption.

Additionally, this study delves into model optimization strategies, particularly the application of freezing and pruning techniques. Model freezing involves fixing certain layers in a deep learning model, reducing the computational burden significantly, which is especially important for devices with limited computational capabilities. Concurrently, pruning techniques, by removing unimportant weights or neurons from the model, help reduce model size and improve operational efficiency. These techniques not only enhance model performance on devices like the Jetson Nano but also boost the efficiency and scalability of the entire federated learning system.

The research in this work covers several key areas:

- Exploration of sensible practices for implementing federated learning on the Jetson Nano.
- Evaluation of the performance of the FedProx algorithm in handling heterogeneous data.
- Model optimization involves pruning and layer freezing techniques for improving accuracy and efficiency.

This research aims to make contributions to the field of federated learning by providing empirical studies on how to effectively utilize FedProx in a heterogeneous data environment. It also explores and demonstrates the feasibility and efficiency of deploying complex machine learning models on resource-constrained edge computing devices.

The structure of this work is as follows: Section 2 reviews recent works related to federated learning in the context of the Internet of Things (IoT). Section 3 discusses the key technologies used in this study. Section 4 describes the system architecture and optimization methods employed. Section 5 comprehensively discusses the experimental results. Finally, Section 6 summarizes the findings of this research and explores future research directions.

2. Related Works

With the development of big data and IoT technologies, the healthcare sector has begun to extensively utilize various sensors, wearable devices, mobile applications, and remote monitoring equipment for the collection of health and medical data. These innovative technologies enable real-time monitoring of patients' health status, provide personalized health recommendations, and rapidly respond in emergency situations. However, with the proliferation of these technologies, data privacy and security issues have become increasingly significant. Therefore, federated learning has emerged as a promising solution for handling data in the medical IoT context. This concept, initially proposed by Google [23], aims to address key challenges in distributed data processing, including communication costs, data privacy, and regulatory compliance. Federated learning allows multiple devices or data centers to collaboratively train a shared model without sharing the raw data, thereby effectively protecting individual privacy. This approach has seen further research and development in various areas, demonstrating its vast potential applications in fields ranging from healthcare to financial services, the Internet of Vehicles (IoV), and personalized services.

In recent years, conducting federated learning on edge devices has become a popular research trend in both academia and industry. Pioneering research in this field is continuously propelling advancements and practical applications of this technology.

2.1. Microcontroller Unit

In [24–26], the feasibility of implementing federated learning on microcontrollers (MCUs) using MCUs as client devices is demonstrated. Wulfert et al. [24] introduce TinyFL, a federated learning approach using only MCUs for communication and aggregation, which demonstrates feasible federated learning communication and aggregation on MCUs via integrated circuits (I²C), showing speed advantages over centralized training. It validates its effectiveness with a gesture recognition case, achieving comparable accuracy to centralized training even with non-IID data. Kopparapu et al. [25] merge federated learning and transfer learning for resource-constrained IoT devices with less than 1 MB of memory. It features federated transfer learning for binary image classification on MCUs, utilizing pre-trained CNNs for feature extraction. Llisterra Giménez et al. [26] employ federated learning on three Arduino Nano 33 BLE Sense boards to train a fully connected neural network (FCNN) for keyword spotting (KWS). The study notes communication bandwidth limitations between MCUs and the server due to serial port usage, which highlights the potential of federated learning on low-cost, low-power devices but acknowledges performance and capability limitations, particularly with complex models or large datasets, and accuracy decreases with more device participation.

2.2. Main Approaches to Handling Non-IID Data

The current main approaches to dealing with non-IID problems in HFL can be categorized from data, algorithm, and system perspectives.

2.2.1. Data-Driven Approach

There are two main data-driven approaches: data sharing and data augmentation. Data sharing [27] is an effective way to deal with non-IID data in HFL. The global model is built up by training a globally shared dataset. For the client model, the connected clients download a random percentage of the shared global data, such that the client model is updated by both local training data and the shared global data. Data augmentation [4] techniques handle the imbalance issue of the local data, which can be further categorized as follows: the vanilla method [28], the mixup method [29], and the generative adversarial network (GAN) [30]-based method. Note that data augmentation techniques, incorporating data sharing, can effectively improve the model's learning performance with non-IID data. However, these data processing techniques may increase the risk of data privacy leakage [2].

2.2.2. Algorithm-Driven Approach

This subsection describes several major types of personalization methods, including conducting local fine-tuning, multi-task learning, and knowledge distillation [13]. The clients perform local fine-tuning to adjust the local models based on the global model from the server [7]. Note that FedAvg [23] is the basic form of local fine-tuning, which includes the procedures of finding a suitable initial shared model and combining local and global information. With meta-learning methods, a high-quality initial global model is developed with fine-tuning (e.g., Personalized FedAvg (Per-FedAvg) [5]). This method [5] achieves first-order optimality with convergence guarantees and better performance on heterogeneous data than FedAvg, but the approximate gradient of Per-FedAvg will significantly affect the results. An alternative to solving the personalization problem is to treat it as a multi-task learning problem [9], considering issues of communication cost, stragglers, and fault tolerance [10].

Besides local fine-tuning and multi-task learning, knowledge distillation [12,14,15] is also a promising method for personalized federated learning. Hinton et al. [15] extend the concept of information transfer from large models to small ones [12] to knowledge distillation techniques. To deal with non-IID data, two types of strategies may be applied, namely federated transfer learning and domain adaptation. For federated transfer learning, Chang et al. [31] propose a collaborative, robust learning method by uploading learned

features instead of local models to implement local personalization. Li et al. [32] propose a framework called decentralized federated learning via mutual knowledge transfer (Def-KT), in which local clients exchange messages directly in a peer-to-peer manner without the participation of the cloud server. Another knowledge distillation strategy is domain adaptation, which emphasizes eliminating the differences between data shards between clients. Peng et al. [33] propose a federated adversarial domain adaptation (FADA) algorithm that handles the domain shift problem with adversarial adaptation techniques. Li et al. [34] present the FedMD algorithm to transfer knowledge from a public dataset and enable clients to train their unique models on local data without privacy leakage risk.

2.2.3. System-Driven Approach

For system-driven approaches, two main kinds of secure data similarity evaluation methods are introduced in the literature, namely evaluating the similarity of the loss value and the similarity of model weights. With the first approach, the authors in [16–19] execute the similarity evaluation by comparing the loss values of different cluster models, where multiple global models are constructed in clusters. Each connected client receives the cluster models for local empirical loss computation. Accordingly, for cluster model aggregation, each client refurbishes the smallest loss-value cluster model, and the server collects these updated models. Instead of considering the loss value, the second approach evaluates the local data similarity and does clustering based on the local model weights. In [11,20], before clustering the clients, the standard FedAvg algorithm is applied to train the global model. Afterwards, the clients receive the warmed-up global model, perform local updates, and return these local models to the server for calculating the similarity scores of model weights. Thus, based on the received local updated models and the calculated similarity scores, the server can group the clients into clusters.

Note that client clustering may be applied to handle the issues concerning negative knowledge transfer and performance degradation due to local model aggregation with considerably different training data. Furthermore, generating multiple global models provides a sensible way to enhance the scalability and flexibility of federated learning systems, considering the specific task requirements, at the cost of more communication and computational resource utilization.

2.3. Mode Optimization

Malan et al. [35] introduced a federated learning with gradual layer freezing (FedGLF) strategy. This approach, validated in two image classification tasks under varying data distributions, has demonstrated its effectiveness by significantly reducing communication volumes during training while maintaining or even enhancing model accuracy, which is particularly crucial for resource-limited edge devices. Lee et al. [36] proposed a grouping method based on data distribution and the physical locations of nodes, coupled with the FedAvg-IC optimization algorithm. This approach enhanced the accuracy and communication efficiency of federated learning.

2.4. Sparse Learning and Pruning

Derrmers et al. [37] demonstrated the feasibility of what is termed sparse learning introducing an innovative sparse momentum algorithm. This algorithm employs exponentially smoothed gradients (momentum) for precise identification and optimization of layers and weights that significantly reduce error rates, thereby enhancing the efficiency and accuracy of federated learning applications on edge devices. Ullah et al. [38] presented a federated learning approach for embedded edge computing using sparse-adaptive model selection. This method leverages sparse learning for local model training and smart model selection and aggregation on a central server. On the client side, stronger nodes are frozen, while weaker nodes are retrained to better capture features relevant to new categories, enhancing overall model adaptability and efficiency.

Refs. [39–41] explored how pruning techniques can enhance the efficiency and effectiveness of federated learning in handling non-IID data and edge devices. Jiang et al. [39] introduced a federated learning method called “PruneFL”, which reduces model complexity through pruning techniques, making it more suitable for running on edge devices with limited computational capabilities. Vahidian et al. [40] proposed a novel strategy that combines structured and unstructured pruning to improve model performance under non-IID data distributions, enabling it to adapt more effectively to diverse client data. Yu et al. [41] introduced an adaptive dynamic pruning technique that incorporates global gradient control variables to optimize federated learning. This method specifically addresses non-IID data issues on edge devices by dynamically pruning the model to reduce complexity while maintaining high accuracy.

2.5. Comparative Summary

Most of the research integrating federated learning with embedded devices primarily aims at conducting model training in constrained environments or implementing local data processing and model training in edge environments for accelerated data processing and real-time responses. These innovative studies are gradually overcoming challenges in the processing capabilities and communication efficiency of edge devices, invigorating the development in fields such as the IoT and smart devices, thereby demonstrating the potential and possibilities in this domain. Table 1 lists various embedded devices used for different federated learning training tasks.

It is worth mentioning that finding a proper target density for pruning is nontrivial. Adaptive pruning models have been designed in [39,41], while [40] combines structured and unstructured pruning. In [39], training was conducted separately on different datasets under both IID and non-IID settings. In the non-IID setting, a single dataset was trained in a non-IID manner by classifying it differently. Note that the above three works use different pruning methods to integrate their proposed algorithms. In contrast, our study employed two entirely distinct datasets for non-IID training with the partial node training strategy proposed in [38], which focuses on retraining weaker nodes in the network. However, when applying this method to our data model, it did not yield the expected results due to the non-IID nature of our dataset, which significantly differs from the independent and IID datasets used in [38]. Despite these challenges, we found the iterative structured pruning strategy mentioned in [40] to be effective during the training process. Therefore, we chose to use a structured pruning method, specifically employing the sparse optimization framework provided by [42], pruning 20% of the nodes, and focusing on training the remaining 80%. Table 2 compares the experimental methods and data formats in the literature.

Table 1. The comparison of different sensors used in FL.

Works	Node	Features	Data Type	Dataset
Llisterra Giménez et al. [26]	Arduino Nano 33	Implementing a KWS application through federated learning training using the Arduino Nano 33 BLE Sense development board.	non-IID	Self-recorded voice samples [43]
Zhang et al. [44]	Raspberry Pi	In an IoT environment, employing federated learning for anomaly detection using the FedDetect algorithm and an adaptive optimizer for efficient local training.	IID	N-BaIoT/LANDER
Ullah et al. [38]	Nvidia Jetson Nano	Utilizing sparse learning techniques to freeze significant nodes during local training phases and retrain weaker nodes.	IID	MNIST/ CIFAR-10/ OCSLab/ UCI-HAR/EC10
Mathur et al. [45]	Android smartphones	Investigating the implementation of federated learning on various smartphones and embedded devices using the Flower framework.	IID	Office-31
	Nvidia Jetson devices + Raspberry Pi			CIFAR-10
Jiang et al. [39]	Raspberry Pi	Introduces a method called “PruneFL”, which reduces model complexity through pruning techniques, enabling its efficient execution on edge devices.	IID	CIFAR-10/ ImageNet-100
			non-IID	FEMNIST/CelebA

Table 2. The comparison of experimental methods and data formats in the literature.

Works	Structured	Unstructured	IID	Non-IID	Node
Ours	✓	✓	✓	✓	Jetson Nano
Jiang et al. [39]	Combining adaptive and distributed parameter pruning method.		✓	✓	Raspberry Pi
Vahidian et al. [40]	✓	✓		✓	✗
Yu et al. [41]	✓			✓	✗
Ullah et al. [38]			✓		Jetson Nano
Wulfert et al. [24]				✓	STM32F446
Kopparapu et al. [25]			✓		Arduino
Llisterri Giménez et al. [26]				✓	Arduino

3. Methodology

This section elucidates the data preprocessing and implementation methods used in this work, which explicates the architecture of federated learning, the algorithms associated with deep learning, and the pertinent formulas and techniques utilized in this study.

3.1. Dataset Description

The datasets utilized in this study are sourced from PhysioNet [46], a publicly available resource for ECG data. We apply two distinct arrhythmia databases: the MIT-BIH Arrhythmia Database (MIT-BIH database) [47] and the St Petersburg INCART 12-lead Arrhythmia Database (St database) [48]. Note that the MIT-BIH database comprises ECG recordings from 48 patients, each lasting 30 min. Each recording includes 2 standard ECG leads, sampled at a rate of 360 Hz to ensure high-quality data collection. The St database, encompassing ECG recordings from 32 patients, comprises 75 half-hour ECG recordings, with each record lasting 30 min. Each recording includes 12 standard ECG leads, sampled at a rate of 257 Hz to ensure high-quality data collection.

The choice of these datasets is based on their wide availability and significance in ECG research. The MIT-BIH database is highly regarded as it provides extensive, long-duration ECG records from multiple patients, making it suitable for a wide range of arrhythmia studies.

3.2. Model Architecture

The architecture of Convolutional Neural Network (CNN) consists of two convolutional operations, two pooling operations, and two fully connected operations. Typically, CNNs add pooling layers after convolutional layers to reduce the dimension of feature maps. Through multiple rounds of convolution and pooling, low-level features gradually combine to form higher-level features. Ultimately, the neural network performs classification based on the extracted high-level features. CNN's convolution operation exhibits characteristics of local connectivity and weight sharing, enabling it to effectively capture local features within the signal, making it particularly advantageous for processing electrocardiogram data with spatial structure.

3.3. Pruning

Structured pruning is a technique employed to prune structural elements of a model, such as neurons or convolutional kernels within a neural network. This pruning method typically adheres to the structure and topology of the model to ensure that the connectivity and hierarchical structure of the network remain unchanged. The structured pruning used in this work involves setting the weights of certain neurons to zero, thereby disabling the associated connections. On the other hand, unstructured pruning involves pruning individual weights within the model, regardless of their location or structure. In this

scenario, pruning is typically achieved by setting weights with smaller values to zero. This may result in some weights within the model being entirely eliminated, thereby reducing the model’s size. Unstructured pruning is often used to further reduce the model’s storage requirements and computational costs, but it may require sophisticated techniques to minimize its impact on model performance. In this work, we utilized the sparse optimization framework provided by DessiLB [14] to transform the original unstructured architecture into a structured one and make it iteratively adapt to our model.

Figure 2a shows the initial network. The unstructured (weight) pruning (Figure 2b) and structured (filter) pruning (Figure 2c) approaches are techniques for reducing the size of deep learning models, improving efficiency, and mitigating overfitting. Structured pruning cuts out full filters and kernel rows, leading to fewer intermediate feature maps, unlike just pruning connections [49]. The primary distinction between them lies in their focus on pruning different types of elements within the model (Figure 3). In [10], it is mentioned that conventional training is applied to specific parameters (weak nodes), while non-specific parameters (strong nodes) are not subjected to training. Similar to iterative structured pruning, pruning is applied to strong nodes while iteratively training weak nodes.

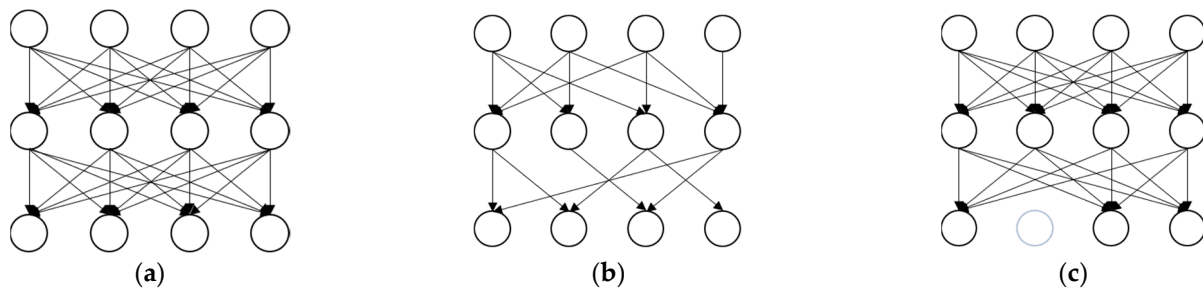


Figure 2. (a) initial network, (b) weight pruning, and (c) filter pruning.

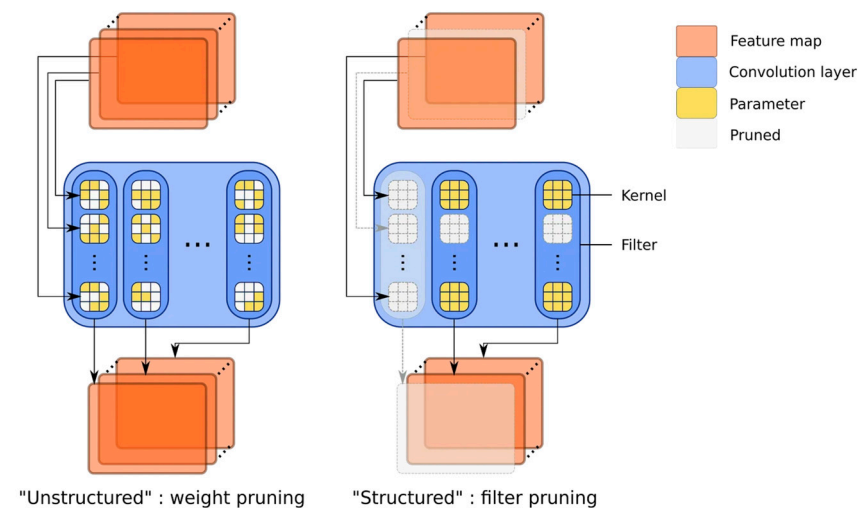


Figure 3. Difference between unstructured (left) and structured (right) pruning (reproduced and modified from [49]).

3.4. Layer Freezing

Layer freezing refers to the practice of setting certain layers or weights in a pre-trained model to an untrainable state. This means that during the fine-tuning stage, the weights of these frozen layers remain unchanged and do not obtain updated. The frozen layers technique is particularly applicable when you already possess a pre-trained model that has been trained on large-scale data relevant to your new task. Such a pre-trained model can provide valuable features and knowledge, and you only need to fine-tune it to adapt it to your specific requirements.

In the field of deep learning, the layer freezing technique is an important strategy for effectively leveraging prior model knowledge. By freezing a subset of the model's layers or weights, we ensure that these layers will not undergo retraining, thus preserving the knowledge they acquired from previous tasks. This helps expedite the model fine-tuning process, reduces reliance on large-scale data, and enhances the model's generalization capabilities. In summary, the layer freezing technique is a powerful tool in deep learning that streamlines transfer learning and the adaptation of models to new tasks, making the process more efficient and feasible.

3.5. Early Stopping

Early stopping is employed to prevent model overfitting and improve the model's generalization ability. Its principle lies in periodically evaluating the model's performance during training, and once the performance reaches a specified criterion, the training process is halted, preventing further training beyond the convergence of the training error. By terminating unnecessary training prematurely, early stopping can save time and computational resources, particularly in the training of large-scale deep learning models.

3.6. Model Evaluation

Model evaluation is the process of using various evaluation metrics to understand the performance strengths, and weaknesses of a machine learning model. This process is crucial for assessing the model's efficacy in the early stages of research. It also plays a vital role in model monitoring, ensuring that the model continues to progress towards its intended goals while preventing unnecessary retraining. After the model completes training, it is essential to evaluate its usability and effectiveness in real-world applications. To assess how well the model performs on new data, various evaluation metrics can be employed. In this work, the F1 score and the confusion matrix are used as the primary evaluation metrics.

4. System Description

This section presents the federated learning system architecture, including a compact embedded computing device (i.e., The Jetson Nano from NVIDIA Corporate, Santa Clara, CA, USA [50]), a framework for federated learning, heterogeneous datasets, federated learning algorithms, and an optimization approach. We utilized the commonly used Linux operating system as the computational platform and integrated it with the Flower federated learning framework along with two Jetson Nano devices to establish a federated learning environment. Furthermore, this section will also describe the optimization methods used in this study.

4.1. Flower FL

Flower [51] is an open-source framework designed specifically for federated learning, aimed at addressing the challenges of machine learning in multi-node environments. The design of the Flower framework is both flexible and scalable, supporting a variety of mainstream machine learning frameworks, such as TensorFlow and PyTorch [52]. This enables developers to experiment with and deploy federated learning in diverse hardware environments and software platforms. Due to its open-source nature, Flower facilitates collaboration between academia and industry, thereby accelerating the innovation and application of federated learning technologies.

Especially in embedded systems, mobile devices, or edge computing environments [45], Flower demonstrates its unique advantages. In scenarios like the one presented in this paper, where data volume can be massive and dispersed, Flower effectively coordinates the computational resources of these devices, facilitating efficient model training and updates across different nodes.

4.2. Data Heterogeneity

In this study, we confront the challenge of statistical heterogeneity. Specifically, this heterogeneity refers to the fact that the data held by different devices is not independently and identically distributed, and there is an imbalance in the data. Due to the isolation of patient data from different hospitals, the data sources used on the two embedded boards are completely different. In this study, the data we collected originates from various sources of arrhythmia data, as detailed in Section 3.1, and the data preprocessing procedures are outlined in Section 3.2. Under a normal scenario, the number of instances representing normal heartbeats significantly outweighs those representing abnormal heartbeats. We categorized the classes into three categories: Normal beat (N), Atrial premature contraction (A), and Premature ventricular contraction (V). In the MIT-BIH database, the ratio between each class is 5.28:1:1.1, while in the St database, the ratio is 5.45:1:1.15. Table 3 depicts the proportion distribution of N, A, and V data.

Table 3. The proportion distribution of N, A, and V data.

	MIT-BIH	St Petersburg INCART
N	16,696	18,568
A	3160	3406
V	3489	3943
	5.28:1:1.1	5.45:1:1.15

4.2.1. FedProx

In traditional federated learning methods such as federated averaging (FedAvg) [23], while they demonstrate effectiveness in distributed data processing, their performance is often limited when dealing with data heterogeneity. To address these limitations of the FedAvg algorithm, Ref. [22] introduced improvements and proposed the federated proximal (FedProx) algorithm, which aims to solve system heterogeneity while providing theoretical guarantees.

The FedProx algorithm is an enhanced version of FedAvg, with a key innovation being the introduction of a proximal term during the local update process. The introduction of this regularization term is primarily aimed at mitigating model bias arising from data heterogeneity and plays a crucial role in reconciling differences in model updates across different devices. Through this approach, FedProx not only enhances the stability of the model in data-heterogeneous environments but also accelerates convergence, effectively reducing the adverse effects of data heterogeneity. The local model update formulas of FedProx are described as follows:

$$\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2} \|w - w^t\|^2 \quad (1)$$

where $\min_w h_k(w; w^t)$ means γ_k^t -inexact solution and γ is a variable for different devices and at different iterations. $F_k(w)$ means the local objective function for device k , μ represents the regularization parameter for the proximal term, controlling the extent of the proximal term's impact. $\frac{1}{2} \|w - w^t\|^2$ means the proximal term itself, which is half the squared norm of the difference between the current model parameters w and the model parameters w^t after the last global update. The purpose of this term is to prevent the model parameters updated locally from deviating too far from the global model parameters w^t .

In this system, each participating node (such as Jetson Nano) performs model training locally and then uploads the model updates to the central server. After collecting all the model update data from the nodes, the FedProx algorithm is applied to construct a more accurate and updated global model on the server. This design not only improves the overall model accuracy but also ensures the privacy and security of user data, as the data does not need to leave the local device to complete the training process.

4.3. Model Optimization

Figure 4 illustrates a workflow diagram for federated learning, providing a detailed depiction of the complete process, from global model download to local training, and then from local training to ultimate model aggregation. The specific steps are as follows:

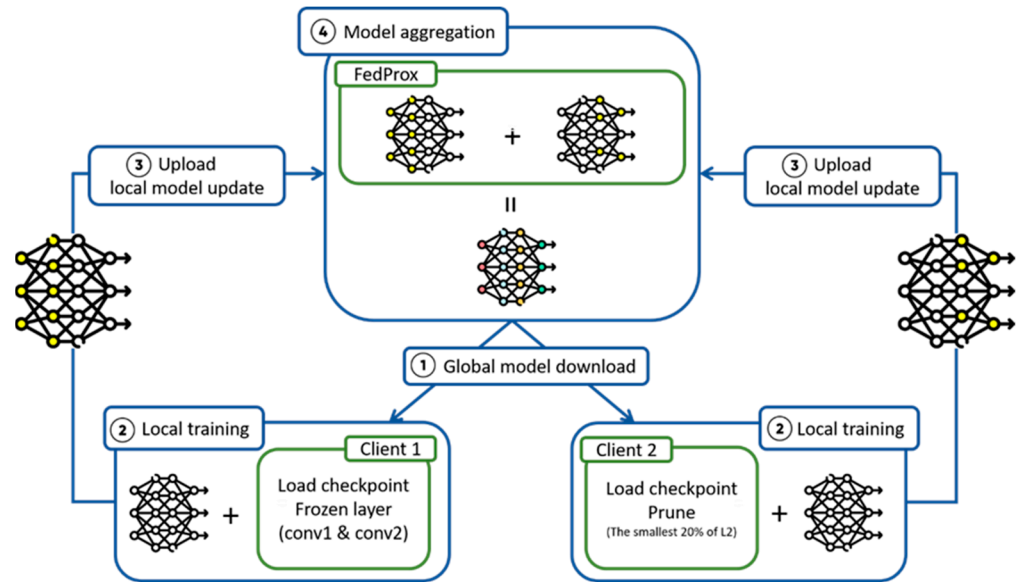


Figure 4. System architecture of federated learning in experiments.

- Step 1: Global Model Download

In this work, we employed a deep learning model trained on the MIT-BIH database with an impressive accuracy of up to 99% on the PC platform. This CNN model was initially trained on the MIT-BIH database and subsequently tested on an embedded system within the context of federated learning. The CNN architecture of the model is depicted in the accompanying Figure 5 and is categorized into three classes. Note that the numbers 300/150/75/38/3 represent the input feature size for each layer, the numbers 1/4/16/32/64 represent the filter size, and the conceptual widths with grey and cadet grey are related to the size of individual layers. This phase involves the download of the current global model from the central server, which is implemented using the Flower framework (Section 4.1). This marks the commencement of the federated learning process, involving clients downloading the global model from the central server.

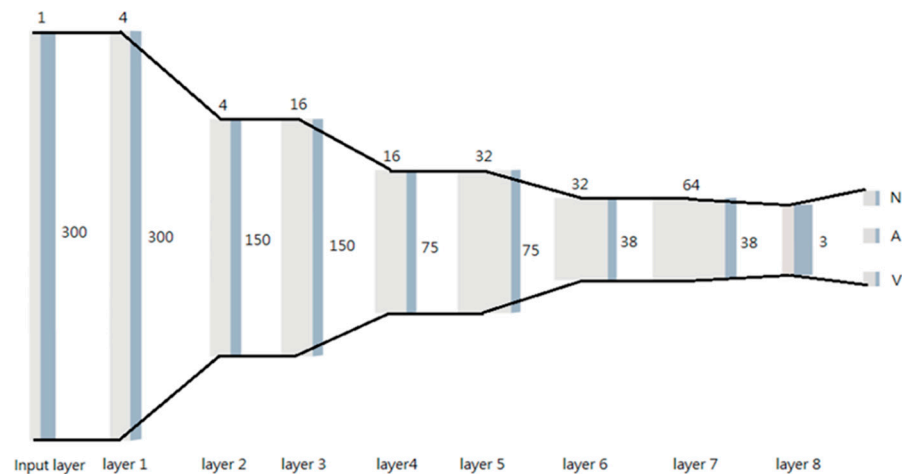


Figure 5. CNN network architecture.

- Step 2: Local Training

In this stage, each client locally trains the downloaded global model on its local data to update model parameters. To address system failures, computational resource constraints, and external disturbances, we save model checkpoints every two training cycles, facilitating rapid recovery in case of interruptions and thereby minimizing time and resource waste. Additionally, we incorporate the use of early stopping (Section 3.5) to prevent overfitting. Referring to Figure 6, Client 1 focuses on enhancing local training by freezing the first two convolutional layers, which adopts a strategy of freezing the first two convolutional layers of the model (Section 3.4). The primary objective of this approach is to maintain accuracy in recognizing existing classes while strengthening the learning capacity of the remaining layers for new tasks. Freezing the first two layers reduces the demand for computational resources, which is particularly important for devices with limited computing capabilities, such as embedded systems. Moreover, it aids in rapid adaptation to new datasets, as this portion of the model has stabilized during previous training and performs well on diverse datasets.

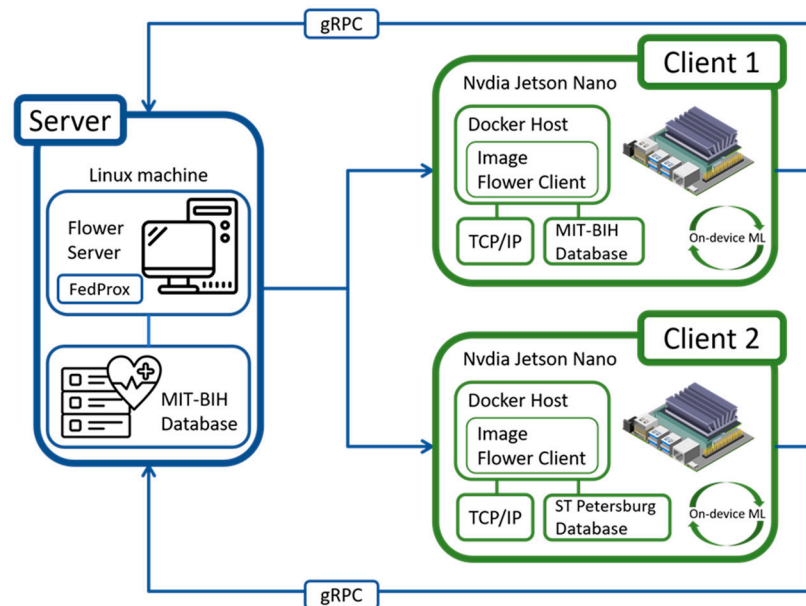


Figure 6. Experimental system architecture [53].

For Client 2, it employs L2 regularization to prune internal nodes within fully connected layers [42], where the choice is made to implement pruning techniques (Section 3.3) rather than Sparse Learning, based on several considerations. Firstly, pruning, as a post-training model optimization technique, allows for more precise model adjustments after training. By removing weights or neurons with minimal contribution to predictions, pruning reduces model complexity and computational burden, which is critical for devices with limited computational capabilities, such as embedded devices. Additionally, compared to sparse learning, pruning provides greater flexibility and adaptability. Sparse learning fixes the model structure during training and may be constrained when handling vastly different datasets. In contrast, pruning allows for fine-tuning after training based on actual performance and requirements, reducing the risk of overfitting while maintaining high accuracy.

In summary, the application of the above two steps in the local training phase contributes to enhancing the model’s generalization ability when dealing with data from different sources. It also maintains efficiency and stability in embedded systems and federated learning environments.

- Step 3: Uploading a Local Model Update

Upon completing local training, clients upload their locally trained model updates to the central server. This process ensures that all clients participate in the overall model update and optimization, promoting the efficiency and effectiveness of the collective learning process.

- Step 4: Model Aggregation

After receiving model updates from all clients, the server aggregates the models using the FedProx algorithm (Section 4.2.1). In this step, updates from each client are combined to form a new global model. This iterative process is repeated to create a more refined and efficient global model. This stage is at the core of federated learning, allowing different clients to collaborate while preserving the privacy of their respective data.

These four steps collectively constitute a complete federated learning process, from global model download to local training, model update upload, and aggregation, ultimately resulting in an efficient, stable, and highly generalized deep learning model.

5. Experimental Results

This section aims to compare and evaluate the deep learning model optimization techniques applied in this study for the heartbeat classification task. After multiple rounds of experimental data, this research identifies the most suitable optimization methods for handling data heterogeneity. In addition to presenting the specific results of each experiment, this section will delve into the performance and characteristics of the model as a whole.

5.1. Experimental Environment

To construct the federated learning environment, we combined a local PC and two ReComputer Jetson Nano J1010 devices from NVIDIA Corporate, Santa Clara, CA, USA. The local PC is used to create a virtual machine (VM) for controlling and coordinating the entire system's operation. The hardware configuration includes an Intel(R) Core (TM) i9-10900KF CPU @ 3.70GHz, 128GB RAM, and a Nvidia RTX 3080 graphics card. The embedded device, the ReComputer Jetson Nano J1010 device, is equipped with a quad-core ARM[®] Cortex[®]-A57 MPCore processor designed for edge computing to support efficient data processing and machine learning tasks at the device's edge. Due to the requirements of the Flower development environment, we configured a PyTorch software environment compatible with Python 3.8 on this device. Note that, due to the support limitations of Nvidia JetPack 4.6.1, Python versions above 3.6 are not supported for GPU acceleration. Therefore, in our experimental setup, although the Jetson Nano device has GPU computing capabilities, we could only utilize the CPU for computations in this environment.

Figure 6 shows the federated learning architecture employed in this experiment, which includes a central server and two clients. Communication between each client and the server is facilitated through gRPC [54], which is a high-performance, open-source, and general-purpose RPC framework led by Google, allowing for remote procedure calls. The following are the steps in the operation process:

(1) Server:

On the server side, we have deployed the FedProx algorithm, which is a federated learning algorithm designed to handle heterogeneous data distributions from different clients. The primary responsibility of the server is to coordinate the entire learning process, including receiving model updates from clients, aggregating them, and distributing the updated global model back to the clients.

(2) Client 1 and Client 2:

Each client (such as Client 1 and Client 2 in Figure 6) has its own dataset for local model training. Clients utilize their hardware to perform local computations. Once a client completes local model training, it sends model updates back to the server via gRPC.

(3) Model Update and Aggregation:

As the server receives updates and performs aggregation, the global model gradually evolves. The aggregated model is then redistributed to all clients, which subsequently commence the next round of local training using the new parameters. This iterative learning cycle enhances the model’s generalization capabilities and enables it to better handle data from different sources.

To evaluate the system performance, a baseline algorithm (i.e., the FedAvg) and the proposed FedProx algorithm are examined with the following model optimization strategies for Clients 1 and 2:

- Strategy 1: Both Client 1 and Client 2 have an initial configuration.
- Strategy 2: Client 1 (Frozen layer); Client 2 (Initial configuration)
- Strategy 3: Both Client 1 and Client 2 are Frozen layer.
- Strategy 4: Client 1 (Frozen layer with unstructured pruning); Client 2 (Initial configuration with unstructured pruning)
- Strategy 5: Client 1 (Frozen layer with structured pruning); Client 2 (Initial configuration with structured pruning)

5.2. Performance Evaluation

This subsection compares and contrasts the FedAvg and FedProx algorithms, considering the five above-mentioned model optimization strategies.

5.2.1. FedAvg/FedProx with Strategy 1

The confusion matrices presented in Figure 7 reflect the classification results obtained by applying the FedAvg and FedProx algorithms with initial configurations to two different datasets. Based on the experimental data in this study, we examine the F1 score and confusion matrix for a comprehensive performance evaluation. The F1 score reveals that the classification results are superior when employing the FedProx algorithm compared to FedAvg. Such comparative results indicate the superiority of FedProx over FedAvg in model performance when handling heterogeneous datasets.

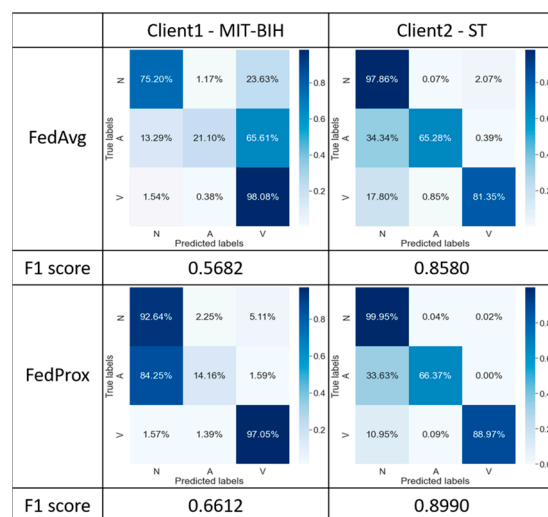


Figure 7. Confusion matrix for comparison between FedAvg and FedProx algorithms.

5.2.2. FedAvg/FedProx with Strategy 2

To enhance the stability of the model on the MIT-BIH database, we introduced the freeze layer technique. To ensure controlled variables in the experiments, we retained the original configuration of the model on the ST database. As depicted in Figure 8, the experimental results indicate that for the application of freeze layers in the MIT-BIH database, the FedProx algorithm demonstrates excellent performance with an F1 score of

97.31%, compared to that of the FedAvg with an F1 score of 79.76%. In the ST database without the use of the freeze layer technique, category A was not accurately classified with the FedProx algorithm (i.e., yet an overall F1 score of 62.15% was still obtained). In contrast, comparing the classification results of the FedAvg with Strategy 1 (Figure 7), the FedAvg algorithm with Strategy 2 achieves significant performance improvements of 40.37% and 9.46% in the MIT-BIH and ST databases, respectively.

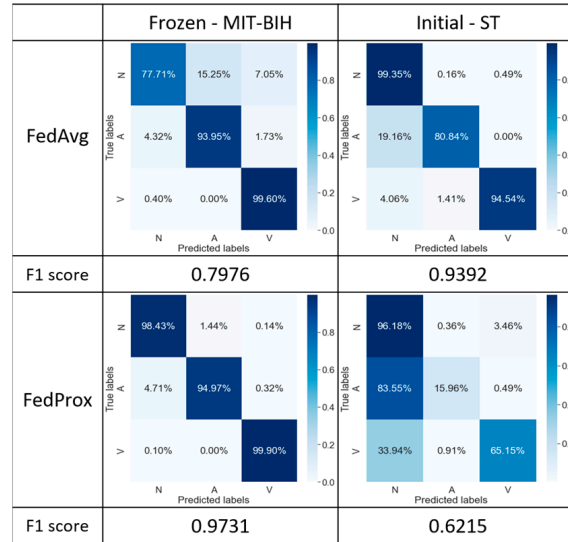


Figure 8. The confusion matrix is frozen in the MIT-BIH database only.

5.2.3. FedAvg/FedProx with Strategy 3

In Figure 9, we applied freezing to two layers (conv1 and conv2) of the model on both clients and evaluated the impact of this operation on the model’s performance. While theoretically, freezing the initial layers might contribute to maintaining the stability of feature extraction, experimental results of both the FedAvg and FedProx algorithms reveal that the models failed to accurately extract features distinguishing different categories, thereby affecting their classification accuracy.

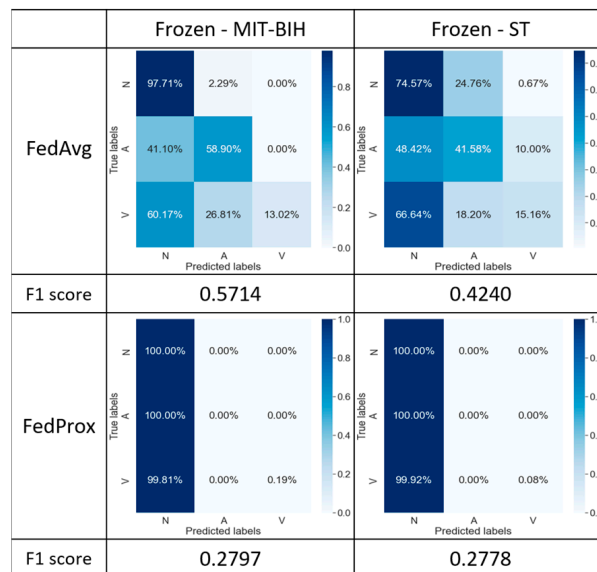


Figure 9. Confusion matrix with both frozen in the MIT-BIH and St databases.

5.2.4. FedAvg/FedProx with Strategy 4

In specific federated learning contexts, we hypothesize that freezing the initial layers may be disadvantageous for the model to capture sufficient discriminative power to handle more fine-grained classification tasks. In order to enhance the adaptability and generalization capabilities of the model across different datasets, we introduced the open-source tool DessiLB for implementing unstructured pruning in this study. Figure 10 presents the corresponding confusion matrix, following the approach outlined in [38] with tests conducted at three different pruning percentages (i.e., 10% (Figure 10a), 20% (Figure 10b), and 80% (Figure 10c) pruning percentages).

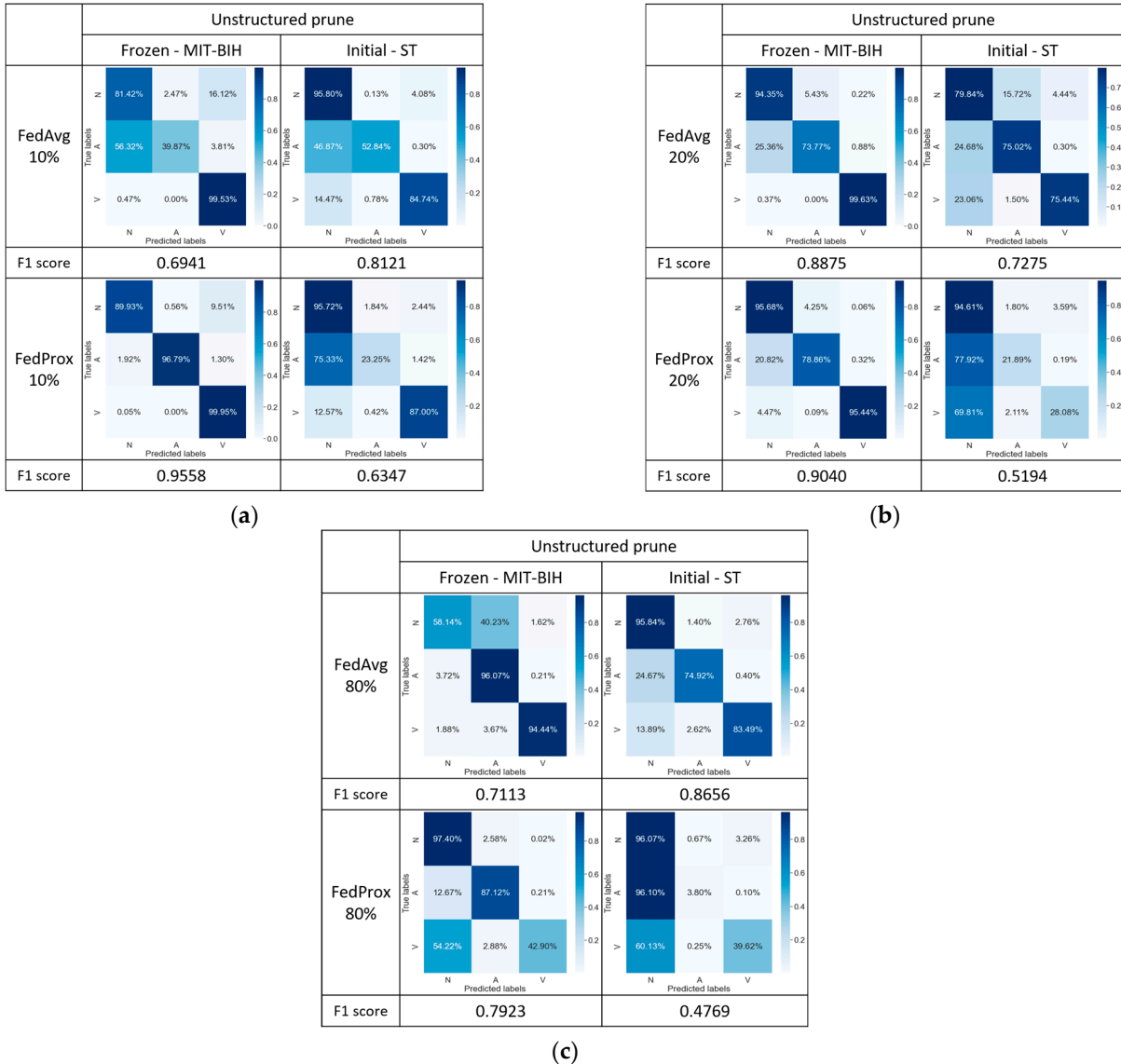


Figure 10. Confusion matrix comparison of model performance: (a) at 10%, (b) at 20%, and (c) at 80% unstructured pruning rates.

For the FedProx algorithm, experimental results indicate that, across various pruning percentages, 10% pruning (95.58% and 63.47%) achieves the most competitive performance, compared to the results in Figure 8 (97.31% and 62.15%). However, even with a 10% pruning percentage, the pruning strategy still falls short of accurately classifying category A. Despite the simplicity and flexibility inherent in unstructured pruning and its ability to rapidly compress models in certain contexts, this pruning strategy did not significantly improve the model’s classification performance in this experiment.

Nonetheless, for the FedAvg algorithm, Strategy 4 with 20% and 80% pruning percentages achieves better classification performance, respectively, with (88.75% and 72.75%) and (71.13% and 86.56%), compared to the results of the FedProx algorithm respectively with (90.40% and 51.94%) and (79.23% and 47.69%), which implies that compared with those of the FedAvg with Strategy 1 (Figure 7) and the FedProx, implementing optimization Strategy 4 (i.e., unstructured pruning and layer freezing) may achieve effective balance between model complexity and classification accuracy of the FedAvg algorithm.

5.2.5. FedAvg/FedProx with Strategy 5

To further enhance the model’s performance, we subsequently modified the open-source code DessiLB to implement a structured pruning strategy, conducting experiments with multiple pruning percentages (i.e., 10% (Figure 11a), 20% (Figure 11b), 40% (Figure 11c), and 80% (Figure 11d) pruning percentages). Based on the experimental results in Figure 11, we observed that the FedAvg with structured pruning has performance deterioration compared to the classification performance of the FedAvg with unstructured pruning.

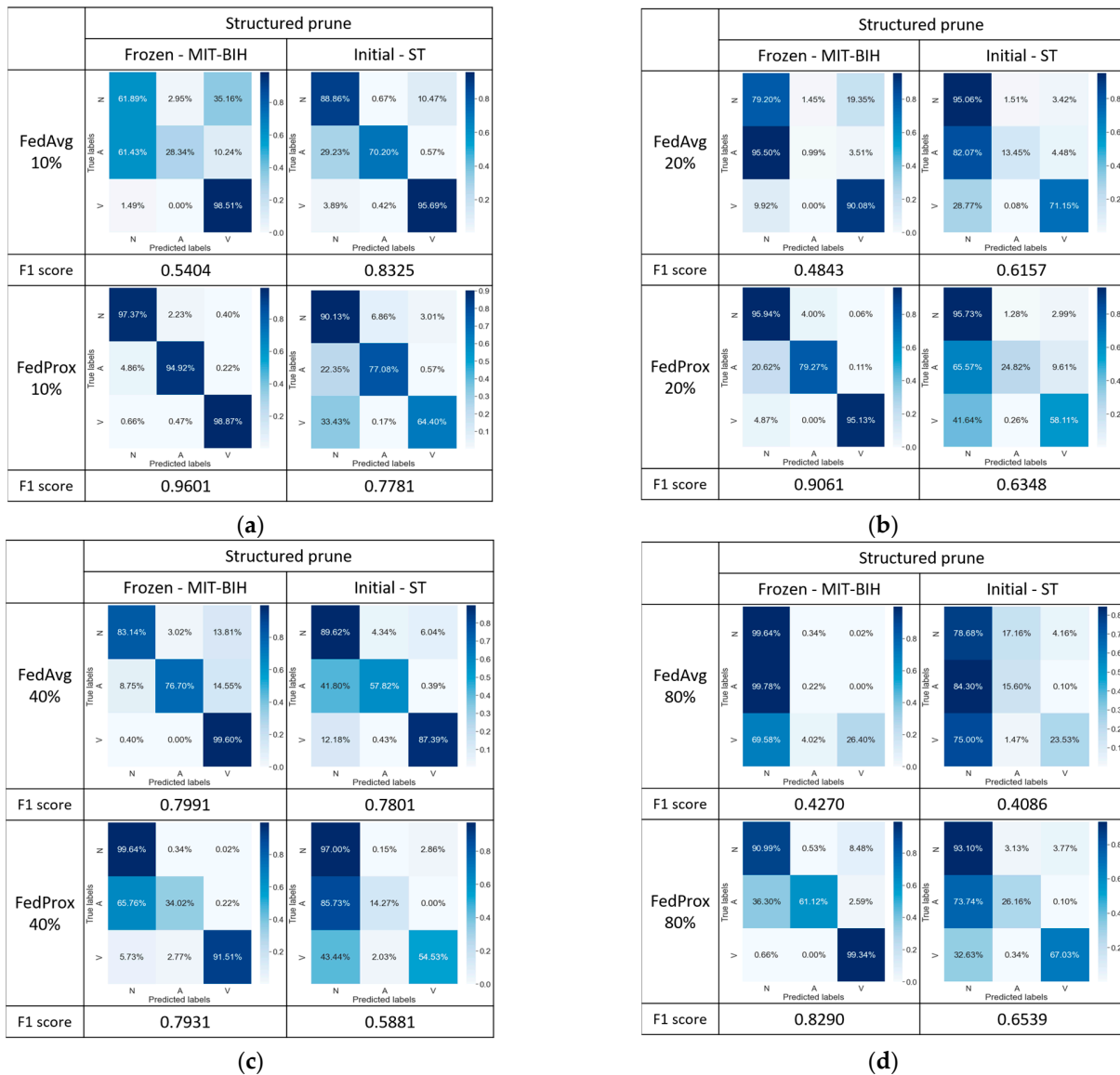


Figure 11. Confusion matrix comparison of model performance: (a) at 10%, (b) at 20%, (c) at 40%, and (d) at 80% structured pruning rates.

On the other hand, at a pruning rate of 10%, the FedProx with structured pruning achieved the best F1 scores, reaching 96.01% and 77.81% (Figure 11a), respectively, which is superior to those of the FedProx with unstructured pruning, reaching 95.58% and 63.47% (Figure 10a), respectively. However, with an increasing pruning percentage, the F1 scores exhibited a decreasing trend, reaching a minimum at the 40% pruning rate, followed by a subsequent rising trend. This phenomenon may be attributed to the substantial pruning of nodes, leading the model to undergo a near-retraining process, thereby highlighting the impact of structured pruning on model performance within specific contexts.

To assess the training efficiency, Figure 12 depicts the F1 score with respect to the number of epochs using the FedProx algorithm with Strategy 5 at a pruning rate of 10% and an early stopping mechanism. Note that in this typical run, with the pruning operation, the model size and parameters are reduced by removing connections, leading to improved generalization. Moreover, given that the training process consists of three rounds with 10 epochs at each round. Observe that Figure 12 provides an example where the training process with the proposed approach stops 12 epochs earlier than that without early stopping (which stops with a 30 epoch delay). Accordingly, the training efficiency of F1 has improved. The series of experimental results underscores the sensitivity of structured pruning to model performance at different pruning levels, providing valuable insights for further exploration of optimized pruning strategies.

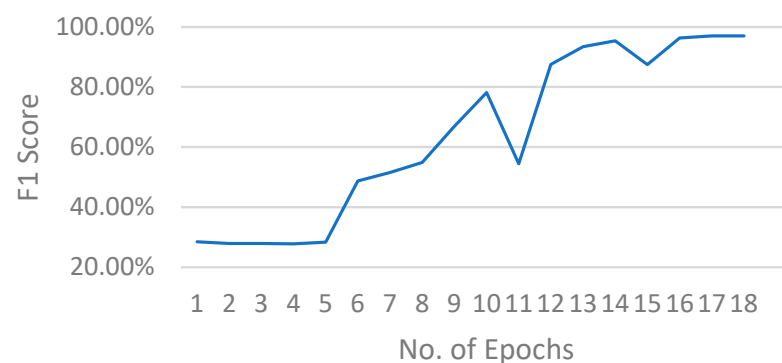


Figure 12. The F1 score with respect to the number of epochs on the ST dataset using the FedProx algorithm at a pruning rate of 10%.

5.2.6. Comparative Analysis

As mentioned in [22], while FedAvg has demonstrated empirical success in heterogeneous settings, it does not fully address the underlying challenges when learning over data from non-IID data distributions (i.e., statistical heterogeneity). Accordingly, to tackle heterogeneity in federated networks with the FedAvg algorithm, this work applies the layer freezing technique and the unstructured pruning method to effectively strike the right balance between model complexity and classification performance, compared with the performance of the FedAvg with the optimization strategies proposed in Section 5.1.

For the FedProx algorithm, this series of research results strongly demonstrates the significant advantages of structured pruning in improving the overall performance of the FedProx model, particularly in enhancing the model's capability to recognize specific categories. By selectively removing entire channels or filters, structured pruning systematically reduces the complexity of the model, showcasing better adaptability to heterogeneous datasets while effectively improving computational efficiency. This holds particular significance for resource-constrained application environments. Therefore, in this study, structured pruning technology is identified as a superior model optimization method. In summary, structured pruning successfully enhances the model's performance on heterogeneous datasets, confirming its feasibility in maintaining model accuracy and opening new possibilities for resource-efficient optimization.

By combining the FedProx algorithm, layer freezing technique, and structured pruning method, this study successfully demonstrates the practical effectiveness of these methods in optimizing deep learning models in a federated learning environment. Importantly, federated learning aims to enhance the overall network performance rather than focusing solely on the accuracy of individual nodes. This discovery points to new research directions and methodological pathways for future exploration and application in the field of federated learning. Table 4 summarizes the classification performance of the FedAvg and FedProx algorithms with optimized operations.

Table 4. Comparison of model optimization for MIT-BIH and ST databases.

Figure	Database	Initial	Frozen	Unstructured Prune	Structured Prune	FedAvg F1 Score	FedProx F1 Score
Figure 7	MIT-BIH	✓				0.5682	0.6612
	ST	✓				0.8580	0.8990
Figure 8	MIT-BIH		✓			0.7976	0.9731
	ST	✓				0.9392	0.6215
Figure 9	MIT-BIH		✓			0.5714	0.2797
	ST		✓			0.4240	0.2778
Figure 10a	MIT-BIH		✓			0.6941	0.9558
	ST			✓		0.8121	0.6347
Figure 10b	MIT-BIH		✓			0.8875	0.9040
	ST			✓		0.7275	0.5194
Figure 10c	MIT-BIH		✓			0.7113	0.7923
	ST			✓		0.8656	0.4769
Figure 11a	MIT-BIH		✓			0.5404	0.9601
	ST				✓	0.8325	0.7781
Figure 11b	MIT-BIH		✓			0.4843	0.9061
	ST				✓	0.6157	0.6348
Figure 11c	MIT-BIH		✓			0.7991	0.7931
	ST				✓	0.7801	0.5881
Figure 11d	MIT-BIH		✓			0.4270	0.8290
	ST				✓	0.4086	0.6539

5.2.7. With Imbalanced Class Distribution

To further assess the robustness of the pruned models, here we explore system performance with respect to imbalanced class distribution by increasing the distribution variation in the datasets (i.e., changing the ratio among each class 5.28:1:1.1 (the MIT-BIH database) and 5.45:1:1.15 (the St database) in Table 3 to the ratio 4.5:1:1.1 and 7.5:1:1.01, as shown in Table 5). Referring to the experimental results with a less imbalanced class distribution (e.g., Table 3), the FedProx with structured pruning at a pruning rate of 10% achieves the best F1 scores, reaching 96.01% and 77.81% (Figure 11a), respectively. In contrast, Figure 13 depicts that with a more imbalanced class distribution (e.g., Table 5), at a pruning rate of 20%, the FedProx with structured pruning achieved the best F1 scores, reaching 98.46% and 64.14%, respectively. Thus, considering the scenario of an imbalanced class distribution, the results suggest that we may adaptively adjust the pruning ratio to suppress the degradation of pruned model accuracy. Table 6 presents a comparison of model accuracy for MIT-BIH and ST with the FedProx given an imbalanced class distribution.

Table 5. The proportion distribution of N, A, and V data with an imbalanced dataset.

	MIT-BIH	St Petersburg INCART
N	26,861	29,852
A	5907	3955
V	6808	4009
	4.5:1:1.15	7.5:1:1.01

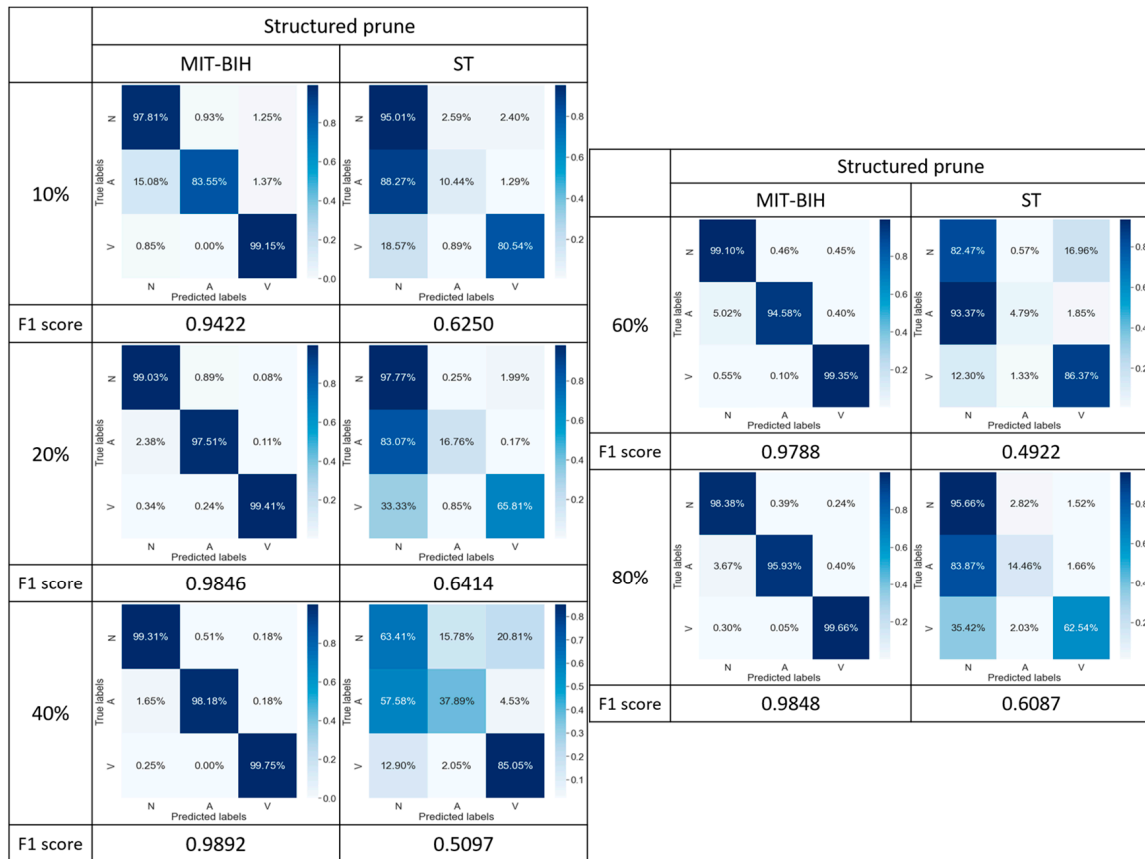


Figure 13. Confusion matrix comparison of model performance at 10%, 20%, 40%, 60%, and 80% structured pruning rates with an imbalanced dataset.

Table 6. Comparison of model optimization for MIT-BIH and ST with an imbalanced dataset.

Prune Percent	Database	Initial	Frozen	Unstructured Prune	Structured Prune	FedProx F1 Score
10%	MIT-BIH		✓			0.9422
	ST				✓	0.6250
20%	MIT-BIH		✓			0.9846
	ST				✓	0.6414
40%	MIT-BIH		✓			0.9892
	ST				✓	0.5097
60%	MIT-BIH		✓			0.9788
	ST				✓	0.4922
80%	MIT-BIH		✓			0.9848
	ST				✓	0.6087

6. Conclusions

This work employs the Flower federated learning framework and Jetson Nano devices to deeply explore and optimize federated learning for heterogeneous data at the edge. It improves the FedProx algorithm to effectively address non-IID data distribution challenges and enhances training efficiency through layer freezing and data pruning techniques. The research demonstrated the viability of federated learning across various datasets, offering a new solution for privacy protection in sensitive areas like healthcare. By parallel processing across multiple nodes, the study optimized medical data distributed across different healthcare institutions and devices, devising strategies for resource-constrained devices like the Nvidia Jetson Nano to boost the practicality of federated learning in edge computing environments. This study establishes a solid foundation for the application of federated learning technology in healthcare and other data-sensitive domains, which provides crucial technical support for the development of big data and artificial intelligence technologies in the future, allowing more people to benefit from intelligent healthcare systems. These advancements not only drive technological innovation but also offer new possibilities for improving the quality of global healthcare.

Based on the system architecture and experimental results of the proposed framework, the future directions of research include (1) introductions of classic datasets (e.g., MNIST, Cifar10/100, and Tiny-ImageNet) and system heterogeneity (e.g., considering the systems characteristics on heterogeneous devices in the network), weight adjustments or the utilization of more advanced methods for handling class imbalance, aiming to enhance the model's performance on rare classes; (2) data security assessment and model applicability (e.g., introducing more advanced encryption techniques and privacy protection mechanisms to further safeguard the data of participating nodes); (3) balancing model complexity and performance (e.g., improving computational efficiency while maintaining model performance with limited network resources). Through the comprehensive application of these approaches, federated learning models will become more suitable for edge computing environments, making a greater contribution to future healthcare systems.

Author Contributions: Conceptualization, H.-T.L. and C.-Y.W.; Formal analysis, H.-T.L.; Funding acquisition, C.-Y.W.; Methodology, H.-T.L. and C.-Y.W.; Supervision, C.-Y.W.; Visualization, H.-T.L.; Writing—original draft, H.-T.L.; Writing—review and editing, C.-Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: The Smart Sustainable New Agriculture Research Center (SMARTer) at the NSTC of Taiwan: 112-2634-F-005-002.

Data Availability Statement: The data presented in this study are available in this article.

Acknowledgments: During the preparation of this work, the authors used a Grammar Checker for grammar checking and English language enhancement. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Xu, C.; Qu, Y.; Xiang, Y.; Gao, L. Asynchronous federated learning on heterogeneous devices: A survey. *Comput. Sci. Rev.* **2023**, *50*, 100595. [[CrossRef](#)]
2. Zhu, H.; Xu, J.; Liu, S.; Jin, Y. Federated learning on non-IID data: A survey. *Neurocomputing* **2021**, *465*, 371–390. [[CrossRef](#)]
3. Chen, T.; Jin, X.; Sun, Y.; Yin, W. VAFL: A method of vertical asynchronous federated learning. *arXiv* **2020**, arXiv:2007.06081.
4. Tanner, M.A.; Wong, W.H. The calculation of posterior distributions by data augmentation. *J. Am. Stat. Assoc.* **1987**, *82*, 528–540. [[CrossRef](#)]
5. Fallah, A.; Mokhtari, A.; Ozdaglar, A.E. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *Advances in Neural Information Processing Systems 33, Proceedings of the Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Online, 6–12 December 2020*; Curran Associates, Inc.: New York, NY, USA, 2020.

6. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017; PMLR: Sydney, NSW, Australia, 2017; pp. 1126–1135.
7. Wang, K.; Mathews, R.; Kiddon, C.; Eichner, H.; Beaufays, F.; Ramage, D. Federated evaluation of on-device personalization. *arXiv* **2019**, arXiv:1910.10252.
8. Arivazhagan, M.G.; Aggarwal, V.; Singh, A.K.; Choudhary, S. Federated learning with personalization layers. *arXiv* **2019**, arXiv:1912.00818.
9. Caruana, R. Multitask learning. *Mach. Learn.* **1997**, *28*, 41–75. [[CrossRef](#)]
10. Smith, V.; Chiang, C.; Sanjabi, M.; Talwalkar, A.S. Federated multi-task learning. In *Advances in Neural Information Processing Systems 30, Proceedings of the Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017, Long Beach, CA, USA, 4–9 December 2017*; Curran Associates, Inc.: New York, NY, USA, 2017; pp. 4424–4434.
11. Sattler, F.; Müller, K.R.; Samek, W. Clustered federated learning: Model agnostic distributed multitask optimization under privacy constraints. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 3710–3722. [[CrossRef](#)] [[PubMed](#)]
12. Bucilua, C.; Caruana, R.; Niculescu-Mizil, A. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2006; pp. 535–541.
13. Deng, Y.; Kamani, M.M.; Mahdavi, M. Adaptive personalized federated learning. *arXiv* **2020**, arXiv:2003.13461.
14. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vision* **2021**, *129*, 1789–1819. [[CrossRef](#)]
15. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
16. Mansour, Y.; Mohri, M.; Ro, J.; Suresh, A.T. Three approaches for personalization with applications to federated learning. *arXiv* **2020**, arXiv:2002.10619.
17. Koppurapu, K.; Lin, E. Fedfmc: Sequential efficient federated learning on noniid data. *arXiv* **2020**, arXiv:2006.10937.
18. Ghosh, A.; Hong, J.; Yin, D.; Ramchandran, K. Robust federated learning in a heterogeneous environment. *arXiv* **2019**, arXiv:1906.06629.
19. Ghosh, A.; Chung, J.; Yin, D.; Ramchandran, K. An efficient framework for clustered federated learning. In *Advances in Neural Information Processing Systems 33, Proceedings of the Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Online, 6–12 December 2020*; Curran Associates, Inc.: New York, NY, USA, 2020.
20. Briggs, C.; Fan, Z.; Andras, P. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–9.
21. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc. Ser. B (Methodol.)* **1977**, *39*, 1–22. [[CrossRef](#)]
22. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. *Proc. Mach. Learn. Res.* **2020**, *2*, 429–450.
23. McMahan, H.B.; Ramage, E.M.D.; Hampson, S.; Arcas, B.A.Y. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, FL, USA, 20–22 April 2017; PMLR, 2017.
24. Wulfert, L.; Wiede, C.; Grabmaier, A. Tinyfl: On-device training, communication and aggregation on a microcontroller for federated learning. In Proceedings of the 2023 21st IEEE Interregional NEWCAS Conference (NEWCAS), Edinburgh, UK, 26–28 June 2023; IEEE: Piscataway, NJ, USA, 2023.
25. Koppurapu, K.; Lin, E.; Breslin, J.G.; Sudharsan, B. Tinyfedtl: Federated transfer learning on ubiquitous tiny iot devices. In Proceedings of the 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), Pisa, Italy, 21–25 March 2022; IEEE: Piscataway, NJ, USA, 2022.
26. Giménez, N.L.; Grau, M.M.; Centelles, R.P.; Freitag, F. On-device training of machine learning models on microcontrollers with federated learning. *Electronics* **2022**, *11*, 573. [[CrossRef](#)]
27. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated learning with non-iid data. *arXiv* **2018**, arXiv:1806.00582. [[CrossRef](#)]
28. Duan, M.; Liu, D.; Chen, X.; Tan, Y.; Ren, J.; Qiao, L.; Liang, L. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. In Proceedings of the 2019 IEEE 37th International Conference on Computer Design (ICCD), Abu Dhabi, United Arab Emirate, 17–20 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 246–254.
29. Zhang, H.; Cissé, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond empirical risk minimization. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018. Available online: <https://openreview.net/pdf?id=r1Ddp1-Rb> (accessed on 21 April 2024).
30. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, C.; Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27, Proceedings of the Annual Conference on Neural Information Processing Systems 2014, NeurIPS 2014, Montreal, QC, Canada, 8–13 December 2014*; Curran Associates, Inc.: New York, NY, USA, 2014; pp. 2672–2680.
31. Chang, H.; Shejwalkar, V.; Shokri, R.; Houmansadr, A. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv* **2019**, arXiv:1912.11279.

32. Li, C.; Li, G.; Varshney, P.K. Decentralized federated learning via mutual knowledge transfer. *arXiv* **2020**, arXiv:2012.13063. [[CrossRef](#)]
33. Peng, X.; Huang, Z.; Zhu, Y.; Saenko, K. Federated adversarial domain adaptation. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. Available online: https://openreview.net/attachment?id=HJezF3VYPB&name=original_pdf (accessed on 21 April 2024).
34. Li, D.; Wang, J. Fedmd: Heterogenous federated learning via model distillation. *arXiv* **2019**, arXiv:1910.03581.
35. Malan, E.; Peluso, V.; Calimera, A.; Macii, E. Communication-Efficient Federated Learning with Gradual Layer Freezing. *IEEE Embed. Syst. Lett.* **2022**, *15*, 25–28. [[CrossRef](#)]
36. Lee, J.-W.; Oh, J.; Shin, Y.; Lee, J.-G.; Yoon, S.-Y. Accurate and fast federated learning via IID and communication-aware grouping. *arXiv* **2020**, arXiv:2012.04857.
37. Dettmers, T.; Zettlemoyer, L. Sparse networks from scratch: Faster training without losing performance. *arXiv* **2019**, arXiv:1907.04840.
38. Ullah, S.; Kim, D.-H. Federated learning using sparse-adaptive model selection for embedded edge computing. *IEEE Access* **2021**, *9*, 167868–167879. [[CrossRef](#)]
39. Jiang, Y.; Wang, S.; Valls, V.; Ko, B.J.; Lee, W.-H.; Leung, K.K.; Tassiulas, L. Model pruning enables efficient federated learning on edge devices. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 10374–10386. [[CrossRef](#)] [[PubMed](#)]
40. Vahidian, S.; Morafah, M.; Lin, B. Personalized federated learning by structured and unstructured pruning under data heterogeneity. In Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW), Washington, DC, USA, 7–10 July 2021; IEEE: Piscataway, NJ, USA, 2021.
41. Yu, S.; Nguyen, P.; Anwar, A.; Jannesari, A. Adaptive dynamic pruning for non-iid federated learning. *arXiv* **2021**, arXiv:2106.06921.
42. Fu, Y.; Liu, C.; Li, D.; Zhong, Z.; Sun, X.; Zeng, J.; Yao, Y. Exploring structural sparsity of deep networks via inverse scale spaces. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 1749–1765. [[CrossRef](#)] [[PubMed](#)]
43. NilLlisterri. Available online: <https://github.com/NilLlisterri/TinyML-FederatedLearning/tree/master/datasets> (accessed on 5 March 2024).
44. Zhang, T.; He, C.; Ma, T.; Gao, L.; Ma, M.; Avestimehr, S. Federated learning for internet of things. In Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, Coimbra, Portugal, 15–17 November 2021.
45. Mathur, A.; Beutel, D.J.; de Gusmão, P.P.B.; Fernandez-Marques, J.; Topal, T.; Qiu, X.; Parcollet, T.; Gao, Y.; Lane, N.D. On-device federated learning with flower. *arXiv* **2021**, arXiv:2104.03042.
46. Lxysl/Mit-Bih_Ecg_Recognition. Available online: https://github.com/lxysl/mit-bih_ecg_recognition (accessed on 5 March 2024).
47. MIT-BIH Arrhythmia Database. Available online: <https://www.physionet.org/content/mitdb/1.0.0/> (accessed on 5 March 2024).
48. St Petersburg INCART 12-Lead Arrhythmia Database. Available online: <https://physionet.org/content/incartdb/1.0.0/> (accessed on 5 March 2024).
49. Tessier, H. *Neural Network Pruning 101: All You Need to Know Not to Get Lost*; Towards Data Science: Toronto, ON, Canada, 2021.
50. Jetson Nano Developer Kit. Available online: <https://developer.nvidia.com/embedded/jetson-nano-developer-kit> (accessed on 5 March 2024).
51. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; et al. Flower: A friendly federated learning framework. *arXiv* **2022**, arXiv:2007.14390.
52. Flower. Available online: <https://github.com/adap/flower> (accessed on 5 March 2024).
53. Jetson-Nano-Isometric. Available online: <https://developer-blogs.nvidia.com/wp-content/uploads/2019/03/Jetson-Nano-isometric.png> (accessed on 5 March 2024).
54. gRPC. Available online: <https://grpc.io/> (accessed on 5 March 2024).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.