Light-weight Federated Learning with Augmented Knowledge Distillation for Human Activity Recognition

by

Gad Gad

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE FACULTY OF GRADUATE STUDIES
OF LAKEHEAD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE (SPECIALIZATION IN ARTIFICIAL INTELLIGENCE)

2023
Lakehead University
Thunder Bay, Ontario, Canada

Supervisors

Dr. Zubair Fadlullah

- Adjunct Professor, Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada
- Associate Professor, Department of Computer Science, Western University, London, Ontario, Canada.

Dr. Mostafa Fouda (co-supervisor)

- Assistant Professor, Department of Electrical and Computer Engineering, College of Science and Engineering, Idaho State University, Pocatello, USA

ABSTRACT

The field of deep learning has experienced significant growth in recent years in various domains where data can be collected and processed. However, as data plays a central role in the deep learning revolution, there are risks associated with moving the data from where it is produced to central servers and data centers for processing. To address this issue, Federated Learning (FL) was introduced as a framework for collaboratively training a global model on distributed data. However, deploying FL comes with several unique challenges, including communication overhead and system and statistical heterogeneity. While FL is inherently private as clients don't share local data, privacy is still a concern in the FL context since sensitive data can be leaked from the exchanged gradients.

To address these challenges, this thesis proposes the incorporation of techniques such as Knowledge Distillation (KD) and Differential Privacy (DP) with FL. Specifically, a model-agnostic FL algorithm based on KD is proposed, called the Federated Learning algorithm based on Knowledge Distillation (FedAKD). FedAKD utilizes a shared dataset as a proxy dataset to calculate and transfer knowledge in the form of soft labels, which are then sent to the server for aggregation and broadcast back to clients to train on them in addition to local training. Additionally, we elaborate on applying Local Differential Privacy (LDP) where clients apply gradient clipping and noise injection according to the Differentially Private Stochastic Gradient Descent (DP-SGD). The FedAKD algorithm is evaluated utilizing Human Activity Recognition (HAR) datasets in terms of accuracy and communication efficiency.

- In chapter 2, various concepts that are employed in the following chapters of this thesis are covered, including HAR, Deep Learning (DL) models, FL, and Empirical Loss Minimization (ELM).
- Chapter 3 introduces FedAKD, a communication-efficient FL method, and evaluates its performance using the Human Activity Recognition (HAR) application with four datasets.
- Chapter 4 discusses the communication efficiency challenge of deploying federated learning on IoT networks and evaluates the communication overhead of FedAKD.
- Chapter 5 focuses on the privacy analysis of FL by presenting a realistic FL threat model and providing privacy guarantees based on DP.

Experimental results demonstrate that FedAKD achieves better performance than other KD-based FL algorithms and comparable performance to model-based FL methods. Furthermore, decreasing the privacy budget causes a slight degradation in performance, as DP clips the gradients during training and adds noise to them to limit per-sample contribution

to model weights, thus protecting privacy. In conclusion, communication-efficient FL algorithms with privacy-preserving techniques present a viable solution for distributed learning while tolerating system heterogeneity and providing formal privacy guarantees, particularly in IoT applications.

ACKNOWLEDGEMENTS

Thanks to many individuals, the success of this thesis is largely dependent on their support and guidance

I am very grateful to the following funding sources for financially supporting my research:

- Lakehead University Faculty of Graduate Studies
- Dr. Zubair Fadlullah
- Vector Institute scholarship in AI

I am grateful to my supervisor, Dr. Zubair Fadlullah, for his guidance and encouragement throughout my academic and research endeavors.

I am also thankful to my research collaborators and co-authors, including Dr. Mostafa Fouda from Idaho State University, for their valuable insights and contributions to my research. Moreover, I extend my appreciation to my fellow lab members from the ACCESS Lab for engaging in research discussions.

Finally, I am sincerely grateful to my family members for their unwavering support.

PUBLICATIONS

Parts of this thesis have been submitted for peer review, published, or accepted for publication:

- Gad, G., Fadlullah, Z. (2022). **Federated Learning via Augmented Knowledge Distillation for Heterogenous Deep Human Activity Recognition Systems**. Sensors, 23(1), 6.(part of Chapter 3)
- Gad Gad, Zubair Md Fadlullah, Khaled Rabie, and Mostafa M. Fouda, **Communication-efficient Privacy-Preserving Federated Learning via Knowledge Distillation for Human Activity Recognition Systems**, Proc. of the 2023 IEEE International Conference on Communications (IEEE ICC'23), Rome, Italy, May 18–June 1, 2023. (part of Chapter 4)

Apart from the manuscripts mentioned above, during my MSc, I also first authored another paper outside the scope of the thesis, on the topic of video analysis. Following is the published paper:

- Gad, G., Gad, E., Cengiz, K., Fadlullah, Z., Mokhtar, B. (2022). **Deep Learning-Based Context-Aware Video Content Analysis on IoT Devices**. Electronics, 11(11), 1785.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The rapid proliferation of wearable devices, such as smartwatches and fitness bands, has led to a paradigm shift in edge applications [1]. These devices have become an integral part of modern life and are widely used for health monitoring, fitness tracking, and human activity recognition (HAR) [2,3]. These edge applications utilize sensory data generated by wearable devices to train machine learning (ML) or deep learning (DL) models to recognize specific user activities.

Machine learning (ML) is a subset of artificial intelligence (AI) that focuses on creating algorithms that can learn from data without being explicitly programmed. One of the most popular types of ML is supervised learning, which involves training a model on a labeled dataset. In contrast, unsupervised learning involves training a model on an unlabeled dataset, relying on the model to discover patterns or relationships in the data.

Deep learning (DL) is a subset of ML that involves training models called artificial neural networks (ANNs) with multiple layers of interconnected nodes. DL has become increasingly popular in recent years because it has achieved remarkable success in various applications, including speech recognition, image classification, and natural language processing.

Despite the potential benefits of ML and DL in HAR systems, a simplistic approach to training these models involves allowing users to share their data with a central server. However, this method has several disadvantages that limit its scalability and effectiveness. Firstly, it requires massive centralized computational power to process and analyze data from a large number of users. Secondly, it incurs high communication costs to move raw data from edge devices to a central server for processing. Thirdly, it poses potential privacy risks since sensitive user data is stored in a central repository.

To overcome these limitations, federated learning (FL) [4] has emerged as a promising solution that allows decentralized learning procedures while keeping each client's data private. This approach enables model updates to be aggregated on the server into global weights, which are then broadcasted to clients for further training without violating user privacy. In

addition to privacy, FL has several other benefits, including reducing communication costs and improving scalability.

Against this backdrop, this thesis aims to explore the application of FL to train distributed HAR systems while preserving user privacy. We also address core FL challenges, such as

1. Communication overhead
2. System heterogeneity
3. Statistical heterogeneity
4. Privacy

While these challenges are generic to any FL scenario, deploying FL on IoT networks, which usually have constrained resources, presents unique challenges. Therefore, we propose a communication-efficient FL algorithm based on Knowledge Distillation (KD) [5] and integrate this algorithm with differential privacy (DP) [6] to overcome these obstacles.

In Federated Learning (FL), sharing weights/gradients is a more communication-efficient approach than sharing raw data and then performing centralized training/analysis in the data center. However, communication in federated networks can be significantly slower than local computation due to the large number of devices involved [7]. Therefore, it is crucial to reduce communication overhead to scale up FL algorithms. Recent methods have employed compression schemes [8] and allowed a flexible number of local updates to achieve a better communication-computation tradeoff.

In federated learning settings, heterogeneity presents itself in various forms. System heterogeneity refers to significant variability in system characteristics across the network due to differences in hardware, network connectivity, and battery power among devices [7]. Approaches such as asynchronous communication [9] and fault tolerance have been proposed to address this issue. Statistical heterogeneity arises when data is not identically and independently distributed (non-IID) across devices, making data modeling more challenging. FedAvg was found to diverge when applied to non-IID data, and FedProx [10] proposes a modified and reparameterized version of FedAvg to ensure convergence.

Lastly, preserving privacy in Federated Learning is not sufficient as gradients can still reveal information about the actual data [11]. Differential Privacy (DP) [6] can provide the necessary guarantees and well-defined bounds to prevent the aggregated model from leaking sensitive information. To keep a model private under the differential privacy framework, noise must be injected either into the gradients during the training procedure or into the final learned model parameters after the procedure concludes [12]. However, injecting too much noise can lead to accuracy deterioration. Developing differentially private machine learning models is, therefore, a non-trivial task.

This thesis proposes a novel approach that tackles the challenges associated with system heterogeneity and communication bandwidth in Federated Learning (FL). Specifically, we leverage Knowledge Distillation (KD) [13] as a communication medium among FL devices instead of model weights or gradients. To this end, we propose a KD-based FL algorithm called FedAKD for training distributed Human Activity Recognition (HAR) systems [14]. FedAKD utilizes a proxy dataset that is shared across the network, and instead of weights or gradients, it calculates and shares soft labels on a Mixup augmented [15] version of the proxy dataset.

The use of FedAKD confers several benefits to clients compared to the standard Federated Averaging (FedAvg) approach. Firstly, clients can design unique model architectures that cater to the specific resource requirements of their devices. Secondly, clients can control their communication bandwidth by choosing the number of samples in the proxy dataset used to calculate soft labels.

To address the privacy concerns inherent in FL, we employ Differential Privacy (DP) as a mechanism for bounding the sensitivity of the learned function (deep model) to private data, thereby providing a rigorous privacy guarantee.

The advantages of our approach are summarized as follows:

1. We use Knowledge Distillation to enable communication across FL devices, thus addressing the challenges of system heterogeneity and communication bandwidth.

2. FedAKD enables clients to design unique model architectures that cater to their devices' resource requirements and control their communication bandwidth.

3. We utilize Differential Privacy to address the privacy concerns of FL, thereby providing a rigorous privacy guarantee.

This approach demonstrates promising results in the evaluation of distributed HAR systems, outperforming other KD-based FL algorithms and achieving comparable performance to model-based FL methods.

1.1 Research Objectives and Significance

This research presents a novel approach to Human Activity Recognition using Federated Learning via Augmented Knowledge Distillation and differential privacy. The proposed approach addresses the challenges of heterogeneity, communication cost, and user privacy. The objectives and significance of this research can be summarized as:

Flexibility in model design: The proposed Federated Learning via Augmented Knowledge Distillation (FedAKD) algorithm enables collaborative training of clients with

independently designed models, addressing the challenge of model heterogeneity in Federated Learning. This allows for more flexibility in designing deep learning models, which can be adapted to different applications and domains.

Effective Human Activity Recognition: The proposed Federated Learning via Augmented Knowledge Distillation (FedAKD) algorithm is evaluated on five different datasets for Human Activity Recognition, one of which is self-collected and covers a variety of modalities. This demonstrates the generalizability of the proposed approach, as it can effectively recognize human activities in different applications and domains. The results also show that the approach is relatively more robust under heterogeneous statistical scenarios, increasing its potential for real-world applications.

Low communication cost: The proposed approach has a low communication cost, which is an important factor in many applications. This allows the proposed approach to be more efficient and scalable, making it more applicable in real-world scenarios.

Privacy Preservation: Differential privacy is employed with the proposed algorithm to limit the contribution of individual data points to preserve privacy. Simulations were performed to investigate the privacy-utility trade-off of heterogeneous and standard Federated Learning with differential privacy. This addresses the privacy concerns associated with the collection and sharing of data, which is a critical issue in many applications.

Robustness under statistical heterogeneity: The proposed approach is shown to be relatively more robust under statistical heterogeneous scenarios. This demonstrates the potential of the proposed approach for Human Activity Recognition in scenarios where the data is not perfectly distributed across clients.

1.2 Contributions

The contributions of this work can be summarized as follows:

- The proposed FedAKD algorithm is a model-agnostic Federated Learning method that incorporates Knowledge Distillation to enable collaborative training of clients with independently designed models. It utilizes a shared dataset to calculate soft labels, which are sent to the server for aggregation and broadcasted back to clients for training in addition to local training

- To address privacy concerns, the proposed algorithm utilizes Differential Privacy by clipping gradients and adding noise to limit the contribution of individual data points to local model training. The trade-off between privacy and utility is explored empirically under different protection levels.

- The proposed algorithm is evaluated using the human activity recognition (HAR) application with four datasets, and the experiments show that it achieves better per-

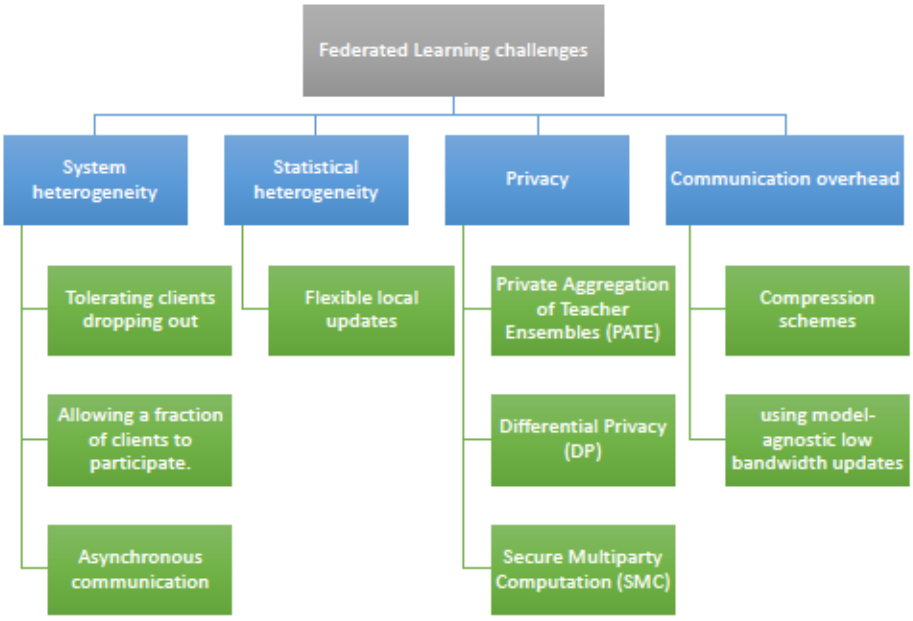

Figure 1.1: The main challenges and directions in federated learning that are discussed in this thesis: Communication overhead, System heterogeneity, Statistical heterogeneity, and Privacy.

formance than other KD-based FL algorithms and comparable performance to model-based FL methods.

1.3 Thesis Organization

Fig. 1.2 outlines the organization and the main concepts discussed in the remaining chapters. The subsequent chapters of this dissertation are structured as follows:

Chapter 2 furnishes an overview of the fundamental concepts that underpin the technologies and methodologies employed in the ensuing chapters of this thesis. These concepts encompass human activity recognition, deep learning, and federated learning.

Chapter 3 introduces FedAKD, a communication-efficient federated learning method based on knowledge distillation, which is evaluated using the human activity recognition (HAR) application as a use case throughout this thesis. Four datasets comprising sensor and image data are utilized to evaluate the performance of FedAKD against other model-based and model-agnostic baseline algorithms.

Chapter 4 of this thesis explores communication efficiency as a significant challenge for the deployment of federated learning (FL) solutions on an Internet of Things (IoT) applications due to low-bandwidth networks. In this chapter, the challenge of communication cost is introduced, and the background works that addressed this challenge are explored.

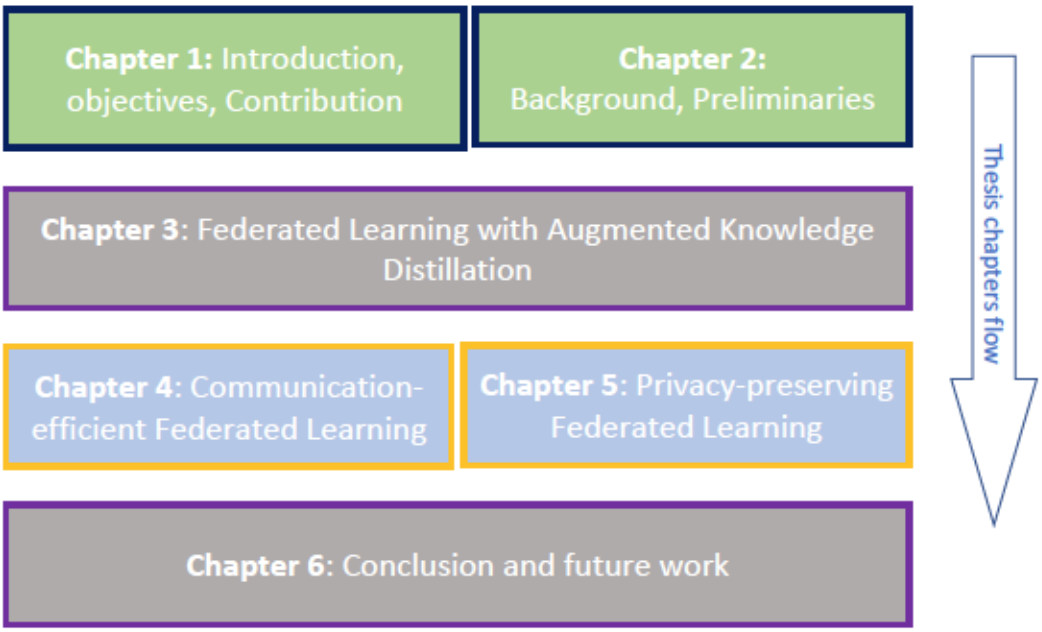

Figure 1.2: Organization of all the chapters of the thesis.

The chapter starts by providing a detailed explanation of FL and its various applications. Then, it delves into the communication overhead involved in FL and how it can hinder its performance. To address this challenge, the communication-efficiency aspects of Knowledge Distillation-based FL algorithms in general, and FedAKD, in particular, are highlighted. Moreover, this chapter presents a compressed version of FedAKD that is even more efficient by introducing some modifications to reduce the size of the soft labels and make them resilient to unreliable/lossy communication protocols that are sometimes used in IoT applications, such as the LoRa communication module. To evaluate the communication overhead of FedAKD and its compressed child algorithm relative to other baseline FL algorithms, Federated learning simulations involving HAR datasets are conducted. The results demonstrate that knowledge distillation-based FL algorithms are more suitable than model-based FL algorithms for IoT applications due to their communication efficiency.

Chapter 5 discusses the application of differential privacy in federated learning algorithms, specifically in the context of model-based and knowledge distillation-based approaches. It covers the mathematical foundations of differential privacy, its key properties like composability and post-processing immunity, as well as the different threat models considered in federated learning. The chapter delves into the implementation of differential privacy in federated learning algorithms, focusing on local differential privacy. Finally, the experiments and results section evaluates the effectiveness of these privacy-preserving techniques on HAR datasets using different protection levels and federated learning algorithms.

Finally, chapter 6 presents the conclusion of the thesis. The chapter visits the main con-

cepts discussed like federated learning knowledge distillation and the techniques proposed like FedAKD and CFedAKD. The chapter discusses differential privacy in deep learning and federated learning contexts. The chapter also presents the main findings of this thesis across three categories: Knowledge Distillation performance analysis, communication overhead comparison, and local differential privacy obtained utility-privacy trade-off under different FL algorithms.

Chapter 2

Background

This chapter depicts an overview of the preliminaries of the concepts and techniques used in this Thesis to lay a foundational understanding of the building blocks that are used throughout this work to build algorithms and conduct analysis.

The Background chapter starts with the topic of Human Activity Recognition (HAR). Since we are interested in addressing communication overhead and system heterogeneity in IoT applications, we consider Wearable-based HAR for evaluating federated learning (FL) algorithms in terms of performance and communication overhead. Machine Learning (ML) and Deep learning (DL) are two related data-driven frameworks comprised of a set of algorithms that can be broadly categorized into supervised and unsupervised learning. In supervised learning, ML/DL models use labeled data for training, then the trained model can be used to infer the prediction, given similar data. On the other hand, unsupervised learning includes tasks like clustering where labeled data is not required. In this thesis, we focus on supervised DL-based models. We explore recent techniques applied in both domains: ML and DL in HAR. The deep learning section ends with a discussion of Knowledge Distillation, a recent DL technique that can be applied in central and federated settings. KD is the technique behind our proposed FL algorithms that we present in chapters 3 and 4.

In this thesis, we focus on supervised DL-based models. We explore recent techniques applied in both domains: ML and DL in HAR. The deep learning section ends with a discussion of Knowledge Distillation, a recent DL technique that was originally proposed for the central setting to train a DL model (called the student) using another pre-trained DL model (called the teacher). KD has also been applied in federated settings [13].

Finally, the background chapter ends with an introduction to federated learning including a formal definition, then it extends this definition to the KD-based FL algorithms and provides a background on recent FL techniques proposed for HAR.

2.1 Human Activity Recognition

Human Activity Recognition (HAR) [16, 17] recognizes human actions or movements from a sensor or vision-based data. The goal of HAR is to develop models that can accurately identify and classify human activity in various applications such as healthcare [18] and human-computer interaction [19]. The advancement in sensor technologies has led to various wearable devices continuously monitoring human activity. However, wearable sensors have limitations such as limited battery life, privacy concerns, and user discomfort. As an alternative, vision-based systems have been proposed for human activity recognition [20]; they use cameras to capture visual data of human movements. These systems are typically used in surveillance, video monitoring, and human-computer interaction systems [21]. The visual data collected from cameras are then used to train models that recognize and classify human activities based on features such as postures and gestures, and facial expressions.

Recently, deep learning-based methods have been widely used in human activity recognition. These methods use neural networks to learn representations of the sensor or vision-based data that can be used to classify human activity [22]. In particular, convolutional neural networks (CNNs) effectively learn spatial and temporal features from data [23]. Additionally, sensor fusion techniques are also used to combine the information from multiple sensors to improve the performance of human activity recognition models [24, 25]. This can help overcome individual sensors' limitations and provide a more robust and accurate human activity recognition. Furthermore, there is a growing interest in developing models for recognizing activities in real-world scenarios, which often involve multiple people, cluttered backgrounds, and variations in lighting and camera viewpoint. To overcome these challenges, recent works have proposed using multi-modal data, such as audio and depth data, to improve recognition performance [26].

2.2 Machine Learning

In this section, we present two of the most popular machine learning algorithms: The K Nearest Neighbor (KNN) and Random Forest (RF). These supervised learning algorithms have shown success in many IoT applications including HAR due to their light weight (relative to their deep learning counterparts) and efficiency [27, 28].

2.2.1 K-Nearest Neighbor

The K Nearest Neighbor (KNN) algorithm [27], is a simplistic, yet powerful machine learning method that can be utilized for classification and regression predictive tasks. While it has been applied in numerous domains, it is primarily utilized for classification problems. The KNN algorithm works by assigning a class to a test instance based on the majority vote of

its K nearest neighbors. The neighbors are determined by calculating the distance between the test instance and all of the training instances. A variety of measures can be employed to compute the distance, including Euclidean distance, Manhattan distance, Minkowski distance, Hamming distance, etc. The value of K is generally a positive integer and is typically small. If K equals one, the object is directly classified as the nearest neighbor's class. However, if K is equal to n, the number of instances in the training data, the algorithm transforms into an eager learning algorithm, as it will explore all of the examples. In order to select the optimal value of K, hyperparameter tuning is traditionally performed. One major drawback of the KNN method is its relatively high prediction time, as it calculates the distance between a single point and all other data points. Some commonly used distance measures, such as Minkowski distance and Euclidean distance, can be expressed as shown in Eq. 2.1 and 2.2, respectively:

$$d_M(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{2.1}$$

$$d_E(x, y) = \sqrt{\sum i = 1^n \left(xi - y_i\right)^2} \tag{2.2}$$

Additionally, the Manhattan distance can be used to measure the distance between two points. The Manhattan distance is calculated as the sum of the absolute differences between the coordinates of two points. The equation for Manhattan distance can be written as follows:

$$d_{Manhattan}(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.3}$$

2.2.2 Random Forest

Random Forest [28] is a popular machine-learning algorithm that uses an ensemble of decision trees to make predictions. It is widely used for classification and regression problems and has shown great success in a variety of domains including finance, healthcare, and image recognition.

The algorithm works by constructing a multitude of decision trees, each of which is trained on a random subset of the training data. The final prediction is then made by combining the predictions of all the individual trees. The result is a robust and accurate model that is less prone to overfitting than a single decision tree.

Each decision tree in the Random Forest is constructed using a subset of the features in the training data. This is done to prevent any one feature from dominating the model and to ensure that the trees are diverse and independent. The features are randomly sampled

at each node in the tree, and the best split is selected based on a criterion that maximizes the information gain.

The two most commonly used splitting criteria are the Gini impurity and the entropy. Both of these criteria measure the degree of impurity or disorder in a set of samples and aim to split the data in a way that maximizes the homogeneity of the resulting subsets.

The Gini impurity is defined as follows:

$$Gini = \sum_{i=1}^{C} p_i(1 - p_i) \tag{2.4}$$

where C is the number of classes, and p_i is the proportion of samples in class i.

The Gini impurity ranges from 0 to 0.5, where 0 indicates a completely pure node (i.e., all samples belong to the same class), and 0.5 indicates a completely impure node (i.e., samples are evenly distributed across all classes). The goal of the splitting criterion is to minimize the Gini impurity and split the data in a way that maximizes the difference between the impurity of the parent node and the sum of the impurities of the child nodes.

The entropy criterion, on the other hand, is defined as follows:

$$Entropy = -\sum_{i=1}^{C} p_i \log_2 p_i \tag{2.5}$$

where C is the number of classes, and p_i is the proportion of samples in class i.

The entropy ranges from 0 to 1, where 0 indicates a completely pure node, and 1 indicates a completely impure node. The goal of the splitting criterion is to minimize the entropy and split the data in a way that maximizes the difference between the entropy of the parent node and the sum of the entropies of the child nodes.

In general, both Gini impurity and entropy are effective splitting criteria for decision trees, and the choice between them is often a matter of personal preference or empirical performance. Random Forest uses both criteria to construct a diverse set of decision trees and combines their predictions using a weighted average to produce the final output.

In conclusion, Random Forest is a powerful and flexible machine-learning algorithm that uses an ensemble of decision trees that are trained on random subsets of the data and combines their predictions to produce a robust and accurate model. The Gini impurity and entropy are two commonly used splitting criteria that aim to maximize the homogeneity of the resulting subsets and are essential components of the Random Forest algorithm.

2.3 Deep Learning

In this section, we present some of the deep learning models that have shown the superior ability to extract patterns from raw data [29, 30]. These layers are often integrated with

each other and with activation functions as building blocks of more complex Deep Learning models.

The section ends with the topic of Knowledge Distillation (KD) which is not a layer, but a training technique that benefits from a previously trained model (called the teacher) to train a to-be-trained model (called the student).

2.3.1 Multi Layer Perceptron

The Multi Layer Perceptron (MLP) is a type of deep learning algorithm that has gained significant attention in recent years due to its ability to learn complex non-linear relationships between the input and output data and its capability to generalize well to new data. MLPs have been applied to HAR to effectively model the dynamics of human activities [31], which are often characterized by subtle variations and transitions.

The architecture of an MLP typically consists of multiple layers of interconnected neurons, with each layer transforming the input data into a higher-level representation that captures increasingly complex features of the data. The output layer of the MLP produces the predicted activity label, based on the learned relationships between the input data and the corresponding activity.

The training of an MLP involves adjusting the weights and biases of the neurons to minimize a chosen loss function, such as cross-entropy or mean squared error. This is typically achieved using an optimization algorithm, such as stochastic gradient descent.

While MLPs have demonstrated impressive performance in HAR applications, they also have some limitations. One major challenge is that it is only suitable for structured data. Another challenge is the computational complexity of training and testing MLPs, particularly for large datasets.

2.3.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a critical type of neural network that excels in image processing and computer vision tasks [29]. They are particularly effective in detecting features and patterns within images, such as edges, shapes, and textures. CNNs have emerged as one of the most powerful deep learning architectures due to their ability to learn representations automatically from data.

Inspired by the structure and function of the visual cortex in the brain, CNNs consist of multiple layers, each with a specific purpose in processing the image data. The first layer is typically a convolutional layer that applies a set of filters to the input image to extract feature maps. These filters are learned by the network during training, and their values are updated through backpropagation.

The second layer is usually a pooling layer, which downsamples the feature maps obtained from the previous layer to reduce their dimensionality. This helps to prevent overfitting and reduce the number of parameters in the network. Max-pooling and average pooling are the two most commonly used pooling operations in CNNs.

Subsequent layers may include additional convolutional and pooling layers, followed by one or more fully connected layers. The fully connected layers are responsible for classification or regression tasks, depending on the nature of the problem being solved. They take the output from the preceding layers and transform it into a final output that represents the predicted class or value.

CNNs are capable of handling translation invariance and noise, making them robust to real-world conditions. They are widely used in various computer vision applications, including image classification, object detection, segmentation, and recognition. One of their significant advantages is their ability to learn features automatically, without the need for manual feature engineering.

2.3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [30, 32] are a type of neural network commonly used in natural language processing, speech recognition, and other sequence-based tasks. Unlike traditional feedforward neural networks, RNNs can take into account the temporal dependencies between inputs. This allows them to model sequences of arbitrary length, making them a powerful tool for handling time-series data.

At each time step, an RNN takes an input x_t and the output h_{t-1} from the previous time step as input, and produces a new output y_t and hidden state h_t. The hidden state serves as a memory unit that stores information about the previous inputs in the sequence. The equations governing the computation of the output and hidden state are as follows:

$$h_t = f(W_h \cdot h_{t-1} + W_x \cdot x_t + b_h) \tag{2.6}$$

$$y_t = g(W_y \cdot h_t + b_y) \tag{2.7}$$

where f and g are activation functions, W_h, W_x, W_y, b_h, and b_y are learnable weights and biases, and $\cdot$ denotes matrix multiplication.

RNNs have shown great success in many sequence modeling tasks. However, they suffer from the vanishing gradient problem, where the gradients become too small to effectively update the weights during backpropagation. This problem arises due to the repeated multiplication of the same weight matrix in the hidden state computation, which causes the gradient to shrink exponentially over time.

Recent research has explored the use of RNNs for Human Activity Recognition (HAR)

from sensor data, with promising results. For example, [32] proposed an RNN-based model for HAR using smartphone sensor data, achieving high accuracy in identifying activities such as walking, running, and cycling. Other studies have explored the use of RNNs for HAR using wearable sensors such as smartwatches and fitness trackers [33].

2.3.4 Long Short-Term Memory

Long Short-Term Memory (LSTM) [23] is a type of RNN architecture designed to address the vanishing gradient problem. LSTMs use a set of specialized memory cells that allow the network to selectively remember or forget information over time. Each memory cell has three gates: the input gate, the forget gate, and the output gate.

The input gate controls the flow of information into the memory cell, and the forget gate controls the amount of information retained in the cell. The output gate controls the flow of information from the memory cell to the output. The equations governing the computation of the gates and memory cell are as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.8}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.9}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.10}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.11}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{2.12}$$

where σ is the sigmoid activation function, tanh is the hyperbolic tangent function, and $[h_{t-1}, x_t]$ represents the concatenation of the previous hidden state h_{t-1} and the current input x_t.

LSTMs have been shown to be highly effective in modeling long-term dependencies in sequences, and have achieved state-of-the-art performance in a variety of tasks such as speech recognition and machine translation. They have also been widely used in HAR, with many studies reporting improved performance compared to traditional RNNs. For example, [34] proposed an LSTM-based model for HAR using accelerometer data from smartphones, achieving high accuracy in identifying activities such as walking, sitting, and standing. Other studies have explored the use of LSTMs for HAR using wearable sensors such as smartwatches and fitness trackers [23, 34].

Overall, RNNs and LSTMs are powerful tools for modeling sequential data and have shown great promise in the context of HAR. By taking advantage of the temporal dependencies in sensor data, these models can accurately identify human activities, making them valuable in a wide range of applications such as healthcare, sports performance tracking,

and home automation.

2.3.5 Knowlede Distillation

Knowledge Distillation (KD) [5, 35] is a technique used to train a model using a trained model utilizing smoothed predictions from the trained model (called the teacher model). In KD, the student model is trained to match the output of the teacher model by minimizing a loss function that compares the soft labels produced by the teacher model to the predicted probabilities of the student model.

The soft labels are obtained by modifying the temperature parameter T of the softmax function applied to the logits (pre-softmax outputs) of the teacher model. The temperature scaling allows the teacher model to produce a smoother and more informative output, which can help the student model learn better. Alternatively, the soft labels can also be obtained by applying a dense layer on the output of the teacher model.

In the context of Federated Learning (FL), KD can be used [14, 36] to train models on distributed devices, each with its private local data. The knowledge distillation process is divided into two stages: in the first stage, clients calculate the soft labels on a shared dataset using a modified version of their local model; in the second stage, the soft labels are aggregated at the server and used to train the local models of all clients.

The loss functions used to train the student model in KD can vary, but two common ones are the Kullback-Leibler (KL) divergence and the Mean Squared Error (MSE) loss. The KL divergence loss can be defined as:

$$KL(p|q) = \sum_i p_i \log \frac{p_i}{q_i} \tag{2.13}$$

where p and q are probability mass functions, and p_i and q_i are the probabilities of the ith outcome under p and q, respectively.

The MSE loss can be defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2 \tag{2.14}$$

Knowledge Distillation is a powerful technique that can help improve the performance of models by transferring knowledge from a well-trained model to another model. The soft labels produced by the teacher model play a crucial role in the knowledge transfer process, and the loss function used to train the student model depends on the specific application.

2.4 Federated Learning

In this section, we present federated learning, a distributed training paradigm that keeps private data on the users' side, and shared training updates to train a global model on distributed data in a private manner. We formally define FL, and then extend the definition to Knowledge Distillation-based FL where the shared updates are not model weights but soft labels. Finally, we explore recent FL algorithms in HAR that address the challenges of system heterogeneity.

Federated learning [7, 37] has emerged as a promising approach for training machine learning models on distributed data without the need for data centralization. By allowing each participating device or entity to train its own local model using its own data, federated learning can mitigate privacy concerns and enable data-driven applications in various domains such as healthcare, finance, and the Internet of Things (IoT). In federated learning, [38, 39] each participating device or entity has a local dataset D_i, where i is the index of the device or entity. The goal is to train a global model f that can make predictions for any input data point x using the following equation:

$$\hat{y} = f(x) \tag{2.15}$$

To train the global model, each participating device or entity trains a local model f_i using its own local dataset D_i. The local model is updated using the following equation:

$$f_i = \arg\min_{f_i} L(f_i, D_i) \tag{2.16}$$

where $L(f_i, D_i)$ is a loss function that measures the error of the local model on the local dataset.

The local models are then sent to the central server, which aggregates the updates to produce the global model using the following equation:

$$f = \frac{1}{n}\sum_{i=1}^{n} f_i \tag{2.17}$$

where n is the number of participating devices or entities.

The global model is then sent back to the participating devices or entities, which use it to update their local models. This process is repeated until the global model converges to a satisfactory level of accuracy.

2.4.1 Knowledge Distillation-based Federated Learning

Knowledge Distillation (KD) [5] is a technique used to train a model (called the student model) by training them using the output of a trained model (called the teacher model).

The teacher model output (called the soft labels) is a smoothed probability distribution (by modifying the softmax function's temperature parameter), or the output of a dense layer. In the context of Federated Learning (FL), KD can be used to train distributed devices each with its private local data [14, 38]. This is achieved by dividing the knowledge distillation process into two stages. Moreover, KD-based FL methods assume that, in addition to the local dataset possessed by each client, all clients share another unlabeled dataset which is used to transfer knowledge gained by local training among clients. In the first stage of KD-based FL, clients calculate soft labels on the shared dataset, and soft labels are sent to the server. In the second stage, the aggregated soft labels broadcasted back to clients to train them using KD loss functions like Kullback Leibler (KL) loss function L_{KL} or Mean Squared Error (MSE) loss L_{MSE} [35]. A Knowledge Distillation Federated Learning (KD-FL) algorithm can be described as follows:

Each participating device has a deep learning model f_i. The last layer of the model f_i is the softmax layer which converts the input into a probability distribution. To calculate the soft labels S_i^r, we remove the softmax layer from f_i to get g_i, which has a more smoothed output than that produced by f_i, which helps distill knowledge more efficiently. Each client calculates soft labels Z_i^r on a shared public dataset D_p using the model g_i:

$$Z_i^r = g_i(D_p) \tag{2.18}$$

For weighting clients' contributions, we tested uniform and performance-based weighting. In performance-based weighting, clients calculate the performance p_i^r on the test dataset D_t, using f_i.

The local soft labels Z_i^r and performance p_i^r are sent to the central server so that clients' contributions (soft labels) can be weighted proportionally to each client's performance.

At the server, aggregate the local soft labels Z_i^r to produce global soft labels S^r using the following equation:

$$Z^r = \sum_{i=1}^{N_c} \frac{p_i^r Z_i^r}{\sum_{k=0}^{N_c} p_k^r} \tag{2.19}$$

To distill the knowledge aggregated in the global soft labels, MSE loss L_{MSE} to train the local models.

$$g_i = \arg\min_{g_i} L_{MSE}(Z^r, g_i(D_p)) \tag{2.20}$$

Finally, each client performs local training on the local dataset D_i using the learning model f_i and the loss function Categorical Cross Entropy (CCE) loss L_{CCE}. on his local dataset D_i.

$$f_i = \arg\min_{f_i} L_{CCE}(Y_i, f_i(D_i)) \tag{2.21}$$

2.4.2 Federated Learning in Human Activity Recognition

Few federated learning methods were proposed for training local Human Activity Recognition (HAR) models on users' edge devices instead of sharing raw data to train a central model [14, 38, 40]. One example of these algorithms is the method proposed by Xiao et al. in [40]. This method uses a perceptive extraction network (PEN) at each client, composed of a featured network based on CNN blocks for feature extraction and a relation network based on LSTM and attention to mine global patterns in data. Another example is the ClusterFL [38] system, which is a similarity-aware FL system for HAR. This system uses an alternating optimization approach to optimize model weights and a cluster indicator matrix that quantifies the relationship between nodes. The loss function used by ClusterFL includes the sum of empirical losses across nodes, the L2-norm, and the K-means clustering.

Chapter 3

Federated Learning via Augmented Knowledge Distillation

In the initial two chapters, we introduced the concept of federated learning and explicated its utility. Furthermore, we deliberated on the challenges that federated learning encounters, which include performance issues in heterogeneous environments, communication overhead, and privacy concerns. In this chapter, we focus on the heterogeneity challenge from a model architecture perspective. Our key inquiry is whether clients can independently devise their local models. Let us consider two clients: Client A, which has powerful hardware including a Graphical Processing Unit (GPU) that enables faster model execution and larger battery capacity and memory storage, and Client B, which has limited resources. When using a server-controlled model architecture for federated learning between the two clients, employing a common minimum model in terms of size and execution time is necessary to ensure compatibility with both clients. This approach, however, does not leverage the optimized hardware/software capabilities available on Client A, resulting in inefficient and slow training. On the other hand, knowledge distillation enables each client to design their own architecture. Thus, in this chapter, we introduce Federated Learning via Augmented Knowledge Distillation (FedAKD), an algorithm we recently proposed [14] that utilizes Knowledge Distillation (KD) in the context of Federated Learning. In addition to facilitating the use of heterogeneous local models, KD-based FL entails significantly lower communication overhead compared to weight-sharing methods. We will further discuss the communication advantages of FedAKD in the subsequent chapter, Chapter 4.

The Federated Augmented Knowledge Distillation (FedAKD) algorithm, which is the subject of examination in this chapter, was initially introduced in a scholarly article published by MDPI Sensors [14]. The primary authorship of that paper is credited to the author of this current chapter. Certain sections of this chapter incorporate content originally disseminated in the aforementioned publication.

3.1 Introduction

Smartwatches were first introduced in the year 2000 at the IEEE International Solid-State Circuits Conference (ISSCC) and have since seen rapid and widespread adoption [1]. Human Activity Recognition (HAR) [2, 16, 41] is an emerging technology that utilizes the sensors of mobile and wearable devices to detect, track, and analyze activity patterns. HAR can automate data collection using wearables and Internet of Things (IoT) devices and can facilitate remote health monitoring in rural communities. Though localized, distributed computing on such resource-constrained devices in the HAR systems is challenging, it has the potential to revolutionize medical analytics.

3.1.1 Deep Learning-Based HAR Systems

Deep learning models have been successful in many domains such as computer vision [42], smart health [22, 43, 44], and natural language processing (NLP) [45]. Therefore, Deep Learning (DL) has been used in HAR systems as a feature extraction method to improve the classification accuracy of activities using fewer sensors [46–49]. Proposed DL-based methods include using ResNet and BiLSTM to extract spatial features of multidimensional signals and sensor fusion with ConvTransformer to achieve high-performance activity classification [25, 50]. In addition to performance gains, DL-based methods require little domain knowledge as they are able to learn directly from raw signals and fuse multi-sensor modalities. On the other hand, traditional Machine Learning (ML) methods often require expert knowledge and feature engineering, both of which are expensive and unique for a given set of sensors. Furthermore, DL methods' expressive power as universal approximators is superior to that of traditional ML methods [29, 30]. However, deploying HAR DL-based models on edge devices still faces challenges such as memory footprint and power consumption. Additionally, data scarcity is another main obstacle for Deep learning as the availability of large datasets is usually a prerequisite for training high-quality deep learning models.

3.1.2 Sensors Used in Sensor-Based HAR Systems

Sensors are crucial in HAR systems. The diversity and quality of the sensors of a HAR system largely determine the accuracy of that system. Two widely used sensors in smartwatches and fitness bands are the Inertial Measurement Unit (IMU) and the Photoplethysmography (PPG) sensors. IMU is an integrated package that usually consists of an accelerometer, gyroscope, and magnetometer. These sensors measure the linear acceleration, rotation rate, and earth's magnetic fields, respectively. An IMU that has all three sensors is referred to as a nine-axis IMU. Sometimes, an IMU does not have a magnetometer, in which case it is called a six-axis IMU. The frequency of IMU (sampling rate) is manually tuned depend-

ing on the application and therefore ranges from 10 to several hundred Hz. Chung et al. studied the impact of sensor positioning on HAR performance and compared different IMU sampling rates; they found that a low-frequency (10 Hz) IMU signal can be effective for recognizing activities such as eating and driving. Using a higher sampling rate yields data with higher resolution and precision, which leads to more accurate analysis at the cost of higher resource consumption [51].

3.1.3 Chapter Organization

This chapter is structured as follows: The Background section provides an overview of the literature review conducted on Human Activity Recognition (HAR) and explores how federated learning can be used to train HAR systems in a distributed and private manner without exposing users' raw data. The Empirical Risk Minimization and Vicinal Risk Minimization sections introduce the concept of augmentation to enhance dataset density, which ultimately improves the performance of models trained on augmented data. We present the Mixup augmentation technique, proposed by [15], and introduce our proposed federated learning algorithm, Federated Learning via Augmented Knowledge Distillation (FedAKD). The Performance Evaluation section details the FL experiments conducted, including the HAR datasets used for evaluation, the baseline methods, and the heterogeneous model architectures employed. The Results and Discussion section presents the test accuracy obtained by each learning method on the HAR datasets, and we provide interpretations based on the theoretical discussion presented earlier. The Conclusions section summarizes the methodology and findings of this study. Finally, the Limitations and Future Work section identifies potential limitations of our work and suggests possible directions for future research.

3.2 Background

3.2.1 Human Activity Recognition

Research has extensively explored traditional Human Activity Recognition (HAR) systems. Anguita et al. [52] proposed a lightweight HAR system that utilizes a Support Vector Machine (SVM) with fixed-point arithmetic to reduce computational costs. Sun et al. [53] presented another SVM-based HAR system that uses smartphone-based sensors on smartphones. The authors tested fitting a model on various positions of smartphones. Kaghyian et al. [27] developed a smartphone application that detects activities from phone movement by using K-Nearest Neighbor (KNN) and the smartphone-based accelerometer sensor. Machine Learning (ML)-based HAR methods learn shallow features from data leading to low HAR performance.

3.2.2 Sensor Fusion in Human Activity Recognition

HAR systems use sensor fusion to integrate raw sensor measurements into more accurate measurements [54]. Sensor fusion is important in many scenarios to address problems such as limited spatial coverage and imprecision [55]. For instance, Inertial Measurement Unit (IMU) integrates and fuses the readings of the accelerometer and the magnetometer to obtain the accurate orientation of an IMU [56]. At the data processing level, sensor fusion can be divided into three types: data-level fusion, feature-level fusion, and decision-level fusion. In data-level fusion, data from multiple sensors are integrated. For example, the self-collected HARB dataset used in this work integrates heart rate pulses from the PPG sensor with gyroscope sensor readings in one input vector to the HAR system. In feature-level fusion, new features are calculated from the original features to provide a different perspective on the state being measured. For example, to prevent misalignment issues [57], the magnitude of the accelerometer is usually calculated. Utilizing the acceleration forces measured in the three axes: A_x, A_y, and A_z, the orientation invariant acceleration magnitude A_m is calculated as follows:

$$A_m = \sqrt{A_x^2 + A_y^2 + A_z^2} \tag{3.1}$$

Decision-level fusion combines the decision of multiple classifiers into a common decision. Recent work [18] utilized an ensemble of deep learning models, where the predictions of many machine learning models are combined into a more accurate prediction for elderly health monitoring based on smartphone sensors.

Deep Learning (DL) has seen significant growth in HAR due to its superior expressive power [30]. Mekruksavanich et al. [58] used an accelerometer and Photoplethysmographic sensors for recognition of physical activity in a Convolutional Neural Network (CNN)-based model. Zeng et al. [33] used temporal and sensor attention combined with Long Short-Term Memory Units (LSTMs). Temporal attention focuses on important parts of the time series data while sensor attention focuses on important sensor modalities. Doshi et al. [59] proposed a computer vision approach for a Diver Activity Recognition (DAR) system on edge devices using a camera fixed in front of the driver. Reinforcement Learning (RL) approaches were also used in HAR. Bhat et al. [3] proposed an online training RL-based policy gradient HAR system utilizing textile-based stretch and accelerometer sensors.

3.2.3 Federated Learning in Human Activity Recognition

Federated learning (FL) is a machine learning approach where a global model is trained by collaborating with clients. The clients' local models are updated after a few iterations of local training. Sozinov et al. [60] investigated the effect of corrupted labels in local data on federated learning performance for synthetic and real-world datasets. They proposed an

FL algorithm that can detect and reject erroneous clients. They also explored the trade-off between communication costs and model complexity. In [61], the authors proposed an FL algorithm that dynamically aggregates model weights based on the statistical distribution of each client by merging similar clients' models in a layer-wise manner.

Numerous federated learning algorithms have been proposed to train distributed human activity recognition (HAR) systems without sending raw data to a server. Xiao et al. [40] developed an FL method where each client has a perceptive extraction network (PEN). The PEN consists of a feature network based on CNN blocks for feature extraction and a relation network based on LSTM and attention to mine global patterns in data. ClusterFL [38] is a similarity-aware FL system for HAR that leverages intrinsic similarities among users' data. ClusterFL uses an alternating optimization approach to optimize model weights $\mathbf{w}i$, and a cluster indicator matrix F that quantifies the relationship between nodes. The loss function used by ClusterFL is given by:

$$\min_{\mathbf{W},\mathbf{F}} \sum_{i=1}^{M} \frac{1}{N_i} \sum_{r=1}^{N_i} l\left(\mathbf{w}_i^T \mathbf{x}_i^r, y_i^r\right) + \tau \operatorname{tr}\left(\mathbf{W}\mathbf{W}^T\right) - \gamma \operatorname{tr}\left(\mathbf{F}^T \mathbf{W}\mathbf{W}^T \mathbf{F}\right) \tag{3.2}$$

where M, N_i, and $\mathbf{w}i$ represent the number of nodes, the local dataset size of the i-th node, and the local weights of the i-th node, respectively. The first term is the sum of empirical losses across nodes. The second and third terms consist of the L2-norm and the K-means clustering. In this formulation, $F \in \mathbb{R}^{M \times K}$ represents an orthogonal cluster indicator matrix. If node k belongs to the q-th cluster, $Fk, q = \frac{1}{\sqrt{N_q}}$ and $Fk, q = 0$ otherwise, where N_q is the number of nodes in cluster q. τ and γ are hyperparameters, $\tau \geq 0$ and $\gamma > 0$.

Most FL methods use gradient averaging, which assumes participating clients have the same model architecture. However, in real-world scenarios, clients may have different architectures due to privacy concerns or limited computational and storage resources. Knowledge Distillation (KD) [5] is a technique for transferring knowledge from a trained model to a to-be-trained model. KD-based FL [62] offers a model-agnostic alternative for collaboratively training heterogeneous model architectures. In this approach, each client sends its scores on a shared dataset, and the server calculates the consensus by averaging the received scores and broadcasts the consensus scores to clients. Clients train their models on the shared dataset using the consensus scores as labels.

3.3 Empirical Risk Minimization

In supervised learning, the goal is to find a function f that belongs to a set F, and describes the relationship between a feature vector X and a target vector Y, based on a joint probability distribution $P(X, Y)$. This function f is selected by minimizing a loss function $\mathcal{L}$ that calculates the discrepancy between predicted outputs $f(x)$ and actual targets y for

any given examples (x, y) sampled from P. The average of the loss function over the entire data distribution P, is referred to as the expected risk and is represented mathematically by this equation:

$$R(f) = \int \mathcal{L}(f(x), y) dP(x, y) \tag{3.3}$$

3.3.1 Limitations of Empirical Risk Minimization

Unfortunately, the distribution P is unknown in most practical situations. Instead, we usually have access to a set of training data $D = \{(x_i, y_i)\}_{i=1}^n$, where $(x_i, y_i) \sim P$ for all i. To approximate the true distribution P, we use the empirical distribution:

$$P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta(x = x_i, y = y_i) \tag{3.4}$$

where $\delta(x = x_i, y = y_i)$ is a Dirac mass centered at (x_i, y_i). Using the empirical distribution P_δ, we can approximate the expected risk by the empirical risk:

$$R_\delta(f) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i), y_i) \tag{3.5}$$

Learning the function f by minimizing the empirical risk $R_\delta(f)$ is known as the Empirical Risk Minimization (ERM) principle. However, this method has its limitation. For example, it can lead to memorization of the training data and poor performance on unseen examples.

3.4 Vicinal Risk Minimization

Vicinal Risk Minimization (VRM) is an alternative to ERM that aims to overcome the limitations of memorization of the training data and poor performance on unseen examples.

In VRM, the true distribution P is approximated by the vicinal distribution $P_{nu}(x, y)$:

$$P_\nu(x, y) = \frac{1}{n} \sum_{i=1}^{n} \nu(x, y | x_i, y_i) \tag{3.6}$$

where $\nu(x, y|x_i, y_i)$ is a vicinity distribution that measures the probability of finding the virtual feature-target pair (x, y) in the vicinity of the training feature-target pair (x_i, y_i).

3.4.1 Gaussian Vicinities

One example of a vicinity distribution used in VRM is the Gaussian vicinities $\nu(x, y|x_i, y_i) = \mathcal{N}(x - x_i, \sigma^2)\delta(y = y_i)$, where $\mathcal{N}(x - x_i, \sigma^2)$ is the probability density function of a Gaussian

distribution with mean x_i and variance σ^2, and $\delta(y = y_i)$ is the Dirac mass centered at y_i. This is equivalent to augmenting the training data with additive Gaussian noise.

For example, consider a dataset of 100 points (x_i, y_i) where x_i represents the age of a person and y_i represents their salary. The true distribution $P(X,Y)$ is unknown, but we can approximate it using the empirical distribution $P_\delta(X,Y)$. Using this distribution, we can calculate the expected risk $R(f)$ and minimize it to find the function f that best describes the relationship between age and salary. However, this method, known as Empirical Risk Minimization (ERM), can lead to overfitting and poor performance on unseen examples.

On the other hand, using VRM, we can construct a vicinal distribution $P_\nu(X,Y)$ by augmenting the training data with additive Gaussian noise. Instead of only considering the 100 points (x_i, y_i), we sample from the vicinal distribution to construct a new dataset of m points (x'_i, y'_i), where each point is a slightly perturbed version of a point in the original dataset. We then minimize the empirical vicinal risk $R_\nu(f)$ using this new dataset. By using a smooth vicinal distribution, VRM tries to capture the underlying density of the data, providing a better approximation of the true risk.

3.4.2 Vicinal Risk

To learn using VRM, we sample the vicinal distribution to construct a dataset $D_{nu} := (x'_i, y'_i)_{i=1}^m$, and minimize the empirical vicinal risk:

$$R_\nu(f) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(f(x'_i), y'_i) \tag{3.7}$$

3.5 Mixup Augmentation

Mixup augmentation is a technique that creates a vicinal distribution by interpolating between two feature vectors (x_i, x_j) randomly drawn from the training data. The interpolation is controlled by a hyperparameter α and is defined as:

$$\mu(\ x|x_i) = \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}_\lambda[\delta(\ x = \lambda \cdot x_i + (1-\lambda)\cdot x_j)] \tag{3.8}$$

where $\lambda \sim Beta(\alpha, \alpha)$, for $\alpha \in (0, \infty)$.

The equation states that the mixup vicinal distribution is calculated as the average of the expectation of the Dirac delta function over all possible j pairs of feature vectors in the training data, where n is the number of samples in the training data.

The Dirac delta function, denoted by δ, is defined as:

$$\delta(\ x = \lambda \cdot x_i + (1-\lambda)\cdot x_j) \tag{3.9}$$

This function returns 1 if the x value is equal to the interpolated value of the feature vectors (x_i, x_j), and 0 otherwise.

The expectation, denoted by $\mathbb{E}$, is calculated over the interpolation coefficient λ, where $\lambda \sim Beta(\alpha, \alpha)$, for $\alpha \in (0, \infty)$. Essentially, this equation defines how the mixing of the two feature vectors is done and the mixing is controlled by the α , The smaller the α the more similar the interpolated points are to the original points.

3.5.1 Using Mixup Augmentation for Knowledge Distillation

To implement mixup augmentation for knowledge distillation, one possible method is to first generate a permutation of the training set and use mixup augmentation to merge the original and permuted datasets. This can be done by creating a new set of synthetic input features $(\tilde{x}, \tilde{y})$ using the mixup interpolation, where λ is a sample from a Beta distribution with hyperparameter α.

The new loss function is defined as:

$$\mathcal{L}_{distill} = (1 - \gamma) \cdot L_{CCE}(f(x), y) + \gamma \cdot L(f(\tilde{x}), T(\tilde{x})) \tag{3.10}$$

, where $f(\tilde{x})$ is the student model's predictions for the synthetic input features, $\tilde{y}$ is the corresponding hard labels, and $T(\tilde{x})$ is the output probability distribution of the teacher model for the synthetic input features, obtained from a dense layer or softmax with a tuned temperature (soft labels), L_{CCE} is the cross-entropy loss, and L is the distance loss function like KL or MSE. The parameter γ is a hyperparameter that controls the trade-off between the two loss terms.

To create synthetic input features $(\tilde{x}, \tilde{y})$ using mixup interpolation, we use the permuted dataset and the original dataset:

$$\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x'_i, \quad \tilde{y} = \lambda \cdot T(x_i) + (1 - \lambda) \cdot T(x'_i) \tag{3.11}$$

where $\lambda \sim Beta(\alpha, \alpha)$ with α being the mixup hyperparameter and (x_i, x'_i) are a pair of examples selected from the original and permuted datasets respectively and $(T(x_i), T(x'_i))$ are the corresponding soft labels from the teacher model.

3.6 Proposed Federated Learning via Augmented Knowledge Distillation (FedAKD)

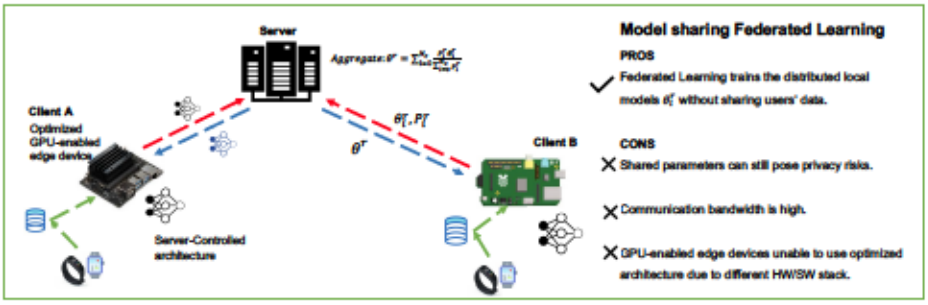

Figure 3.1: Standard Model-based Federated Learning.

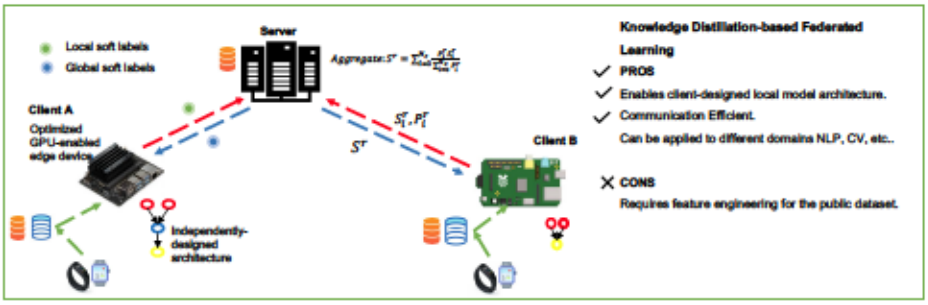

Figure 3.2: Knowledge Distillation-based Federated Learning.

Standard federated learning aims to improve the performance of a global model by combining distributed models trained on decentralized data across different clients. However, in heterogeneous federated learning, each client has an independently designed model, making it impossible to combine weights of different architectures to create a global model. The goal of heterogeneous federated learning is to enhance the performance of each client beyond their local efforts. Knowledge distillation-based federated learning was presented as a model-agnostic distributed training approach to achieve this goal. However, the utility of such methods may be compromised when the shared data used for knowledge transfer is not representative or sufficient for the other local datasets to be learned. To address this issue, we propose an augmented knowledge distillation federated learning algorithm that provides competitive performance to model-based federated learning with significantly less communication overhead. This is achieved by using vicinal methods to generate synthetic public samples that increase the density of data, thus creating a richer feature-to-soft label space that efficiently distills knowledge from limited data. The overview of Model-based and Knowledge Distillation-based Federated learning is presented in figures 3.1 and 3.2,

respectively.

Algorithm 1 FedAKD Algorithm

Input: Public dataset: D_p, Test dataset: D_t, Local dataset of client i: D_i, Independently designed local model of client i: f_i, Number of communication rounds: R, Number of epochs for local training: E_l, Number of epochs for KD training: E_{KD}, Loss function for local training: L_l, Loss function for KD training: L_{KD}, Total number of participating clients: N_c, Fraction of clients participating at any given round: K
Output: Collaboratively trained local model f_i
Initialize f_i
for round $r = 1$ to R **do**
    $N_k = K \cdot N_c$
    Select N_k clients randomly
    $\rho^r, \alpha^r \leftarrow$ server randomly generated
    Broadcast ρ^r, α^r
    for client $i = 1$ to N_k **do**
        $D_d^r \leftarrow \text{permute}(D_p, \rho^r)$
        $D_{Aug}^r \leftarrow \text{mixup}(D_p, D_d^r, \alpha^r)$
        $S_i^r \leftarrow$ calculate soft labels on D_{Aug}^r
        $P_i^r \leftarrow$ calculate accuracy on D_t
        Send S_i^r, P_i^r to Server
    end for
    $S^r \leftarrow$ aggregate S_i^r weighted by P_i^r
    Broadcast S^r
    for client $i = 1$ to N_k **do**
        for epoch $e = 1$ to E_{KD} **do**
            Compute gradients $g_{KD} = \nabla L_{KD}(f_i, D_{Aug}^r, S^r)$
            Update f_i: $f_i \leftarrow f_i - \eta \cdot g_{KD}$
        end for
        for epoch $e = 1$ to E_l **do**
            compute the gradients $g_{CCE} = \nabla L_{CCE}(f_i, D_i, Y_i)$
            Update f_i: $f_i \leftarrow f_i - \eta \cdot g_{CCE}$
        end for
    end for
end for

The FedAKD algorithm 1 includes the following steps:

1. The server sends ρ^r and α^r to clients, where ρ^r is used to generate a permuted version

of the public dataset D_p called D_d^r, and α^r is used for mixup augmentation.

2. Clients use ρ^r and α^r to calculate the augmented dataset D_{Aug}^r from the permuted public dataset D_p.
3. Clients calculate soft labels S_i^r and performance P_i^r on the test dataset D_t.
4. Clients send S_i^r and P_i^r to the server.
5. The server aggregates the soft labels S_i^r from all clients into consensus soft labels S^r, weighted by P_i^r.
6. The server sends the consensus soft labels S^r to all clients.
7. Clients use S^r as labels for the augmented dataset D_{Aug}^r during knowledge distillation training for local epochs E_{KD}.
8. Clients train on their local datasets D_i using Categorical Cross Entropy (CCE) loss for local epochs E_L.

ρ^r and α^r are hyperparameters that control the permutation and mixup degree respectively. D_p is the public dataset and D_d^r is the permuted version of it at global round r using

$$D_d^r = \text{permute}(D_p, r) \tag{3.12}$$

Mixup augmentation is applied to D_p and D_d^r as follows to create D_{Aug}^r.

$$(x_i, y_i) \sim D_p, \quad (x_i', y_i') \sim D_d^r \tag{3.13}$$

$$\lambda^r \sim \text{Beta}(\alpha^r, \alpha^r), \quad \tilde{x} = \lambda^r \cdot x_i + (1 - \lambda^r) \cdot x_i', \tag{3.14}$$

Note that we do not need to generate $\tilde{y}$ because in our case the labels that clients use to train on $\tilde{x}$ are the global soft labels, not $\tilde{y}$.

3.7 Performance Evaluation

3.7.1 HARB Dataset

We evaluate the proposed system, FedAKD, using the HARB dataset, which is a self-collected sensor-based time-series HAR dataset. It uses Gyroscope and Photoplethysmography (PPG) sensors to classify three activities: walking, studying, and sleeping. The dataset was collected using the Mi band 4 fitness band, a commercial wearable device. We extracted Gyroscope readings and heart rate pulses calculated by the band based on the

PPG sensor. To collect the data, volunteers wore the band and powered on the Raspberry Pi (RPI) computing board via Bluetooth Low Energy (BLE), which was connected to the Mi band 4 fitness band. They then performed the target activity for a period of time, which we refer to as a data collection session. The session length was determined by the volunteer and ranged between 20–400 minutes.

As shown in Figure 3.4, the data collection equipment consisted of a battery-powered Raspberry Pi 3B and a Mi band 4. During data collection, volunteers put the RPI in their pockets and wore the band to perform the target activity easily. Once the RPI booted, a Python script was triggered to extract IMU and heart rate measurements using the predefined MAC address and authentication key of the Mi band 4. IMU has a frequency (i.e., sampling rate) of about 10–15 Hz, while the heart rate has a lower frequency of 2–3 Hz, as PPG raw data are processed by the wearable device before the calculated heart rate is produced. Each data collection session generated a new file in a designated data folder for that volunteer, and the data were appended to it line by line. Since the frequency is inconsistent between the two data types, each line may contain either IMU measurements or heart rate values, in addition to the timestamp.

To address the frequency inconsistency between the two data streams, heart rate readings were interpolated to increase their frequency to match that of the Gyroscope sensor. Another challenge we encountered was noisy heart rate measurements, where the fitness band sometimes returned negative heart rate readings. As both Gyroscope and heart rate measurements are integers, while the timestamp is a float value, the parser ignored negative heart rate values as noise.

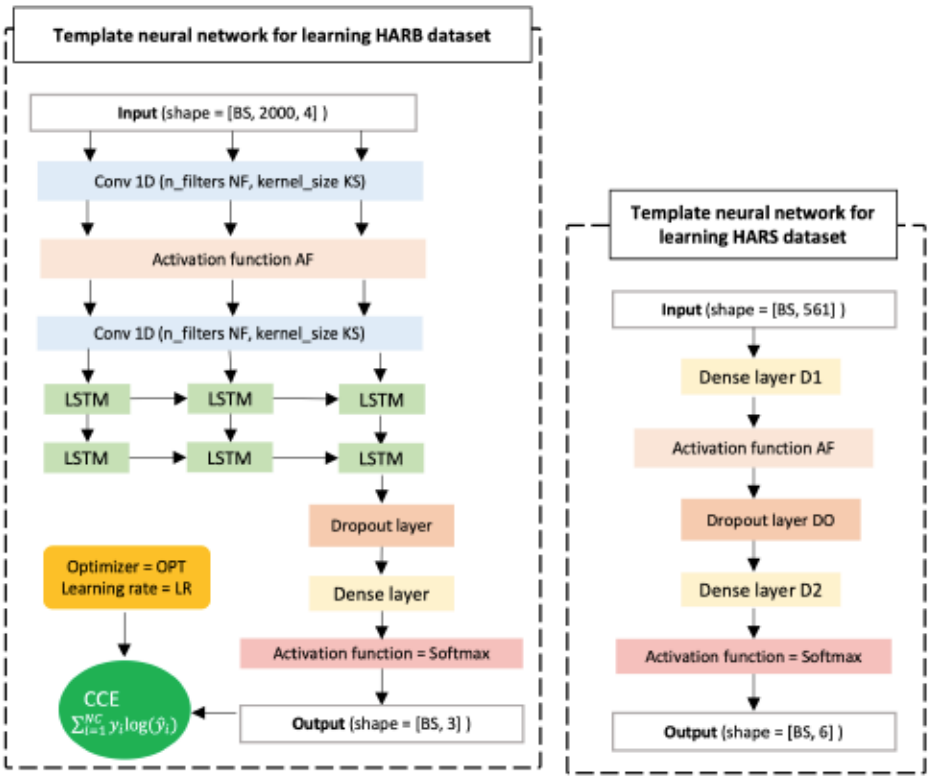

Figure 3.3: (**Left**) A template deep learning model used to derive heterogeneous models with various learning capacities for learning the HARB dataset in both centralized and federated learning settings; (**Right**) A template deep learning model is used to derive variant models to train on the HARS dataset in centralized and federated settings.

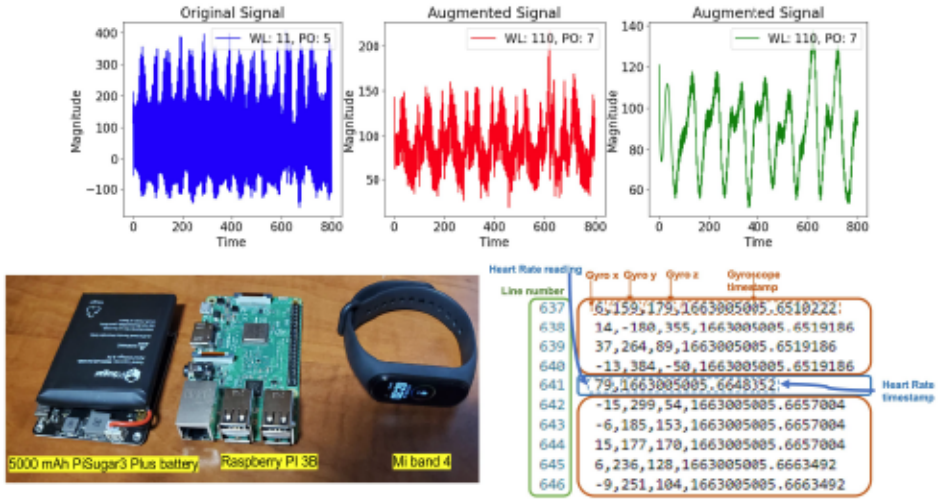

Figure 3.4: (**Top**) From left to right: Raw signal from the self-collected HARB dataset. An augmented version of the raw signal using a Sav–Gol filter using a window length of 11 and a polynomial order of 5. An augmented version of the raw signal using a Sav–Gol filter with a window length of 110 and a polynomial order of 7. The goal of augmentation is to balance the dataset. (**Bottom**) Right: Human Activity Recognition with fitness Band (HARB) dataset sample file format; (**Bottom**) Left Data collection equipment.

3.7.2 HARS Dataset

The HARS dataset is a Human Activity Recognition (HAR) dataset that maps smartphone-embedded inertial sensors to six activities: Walk, Walk up-stairs, Walk down-stairs, Stand, Sit, and Lay. This dataset is tabular and consists of a 561-feature vector calculated using time and frequency domain variables from fixed-width sliding windows of 128 readings with an offset of 64 readings, after applying noise filters to the raw signal [63]. The dataset is publicly available [63]. Table 3.2 summarizes the details of the HARS dataset. The dataset employs 50 Hz sensors, and it is distinct from the HARB dataset in terms of data modality and sensors used [63]. The HARB dataset is a time-series dataset, which relies on a device that uses low-frequency sensors (2–15 Hz), the Gyroscope, and Photoplethysmography (PPG) sensors [64]. Table 3.1 and Table 3.2 present a comparison between the two datasets.

Table 3.1: HARB and HARS datasets sizes

Dataset	**HARS**	**HARB**
Train set size	6616 samples	3000 samples
Test set size	2947 samples	2000 samples
Local dataset size (per client): i.i.d	$20 \times 6 = 120$ samples	$20 \times 3 = 60$ samples
Local dataset size (per client): non-i.i.d	$20\times$ classes (≤ 6)	$20\times$ classes (≤ 3)

Table 3.2: HARB and HARS datasets characteristics.

Dataset	HARS	HARB
Availability	Public dataset	Self-collected.
Source	Waist-mounted	Wrist-mounted
Sensors	Inertial sensors	Photoplethysmography and Gyroscope
Sensors frequency	50 Hz	Hear rate: 2 Hz, Gyroscope: 15 Hz
Data modality	Tabular	Time-series
Number of Activities	3	6

3.7.3 Dataset Preprocessing

Figure 4.3 presents an overview of the federated learning (FL) experiment conducted on the HARB dataset. The experiment involved sampling the raw time-series data of each volunteer, labeling the data, and splitting it subject-wise into train and test sets. To prevent an information leak that causes high test accuracy but poor performance on new subjects, all subjects were included in either the train or test sets, but not both. This issue was highlighted in [65]. Table 3.1 displays the sizes of the train and test sets for both datasets. To balance the HARB dataset, we used the Savitzky-Golay filter [66] to double the size of the *walk* data before centralized training. We trained 10 models for 20 epochs with a batch size of 32 for centralized training of the HARB FL experiment. Similarly, the deep models for the HARS FL experiment were trained with the same number of epochs and batch size in a centralized manner. Since both datasets are balanced, accuracy was used to evaluate the performance of the models during both centralized and FL training. For unbalanced datasets, metrics such as macro-F1 score and balanced accuracy should be used to reflect the model's performance on minority classes [67]. Figure 3.6 provides a closer look at the proposed heterogeneous FL algorithm.

The accuracy of each model during centralized training on the HARB and HARS datasets, along with the models' hyperparameters, is reported in tables 3.3 and 3.4, respectively. After centralized training, the FL experiment began, with each of the 10 clients using only 20 samples per class as their local dataset. However, the entire test set shown in table 3.1 was used as the test set to calculate the model's test accuracy during the FL experiment. Figure 3.7 shows the non-i.i.d distribution of the HARB and HARS datasets, with some classes being dropped from the local datasets due to their non-i.i.d nature. For example, the local dataset of the second client/party (P1) contained only four classes, resulting in a total of 80 samples in their local dataset, assuming 20 samples per class.

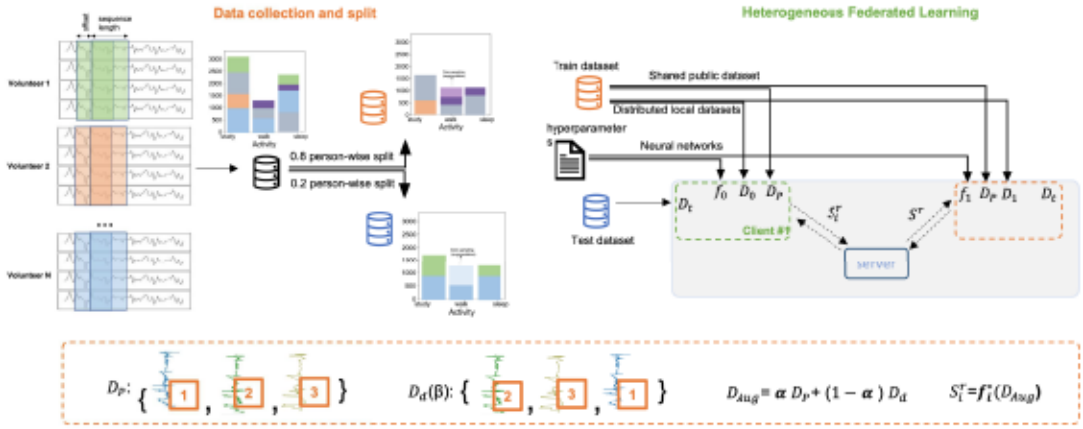

Figure 3.5: An overview of data preprocessing and splitting of the self-collected HARB dataset, and how each component is being utilized in the proposed heterogeneous Federated Learning system. The dotted box in the bottom explains the augmentation mechanism used in this work which is based on mixup augmentation and permutation. Signal colors and the associated number represent the index of each sample.

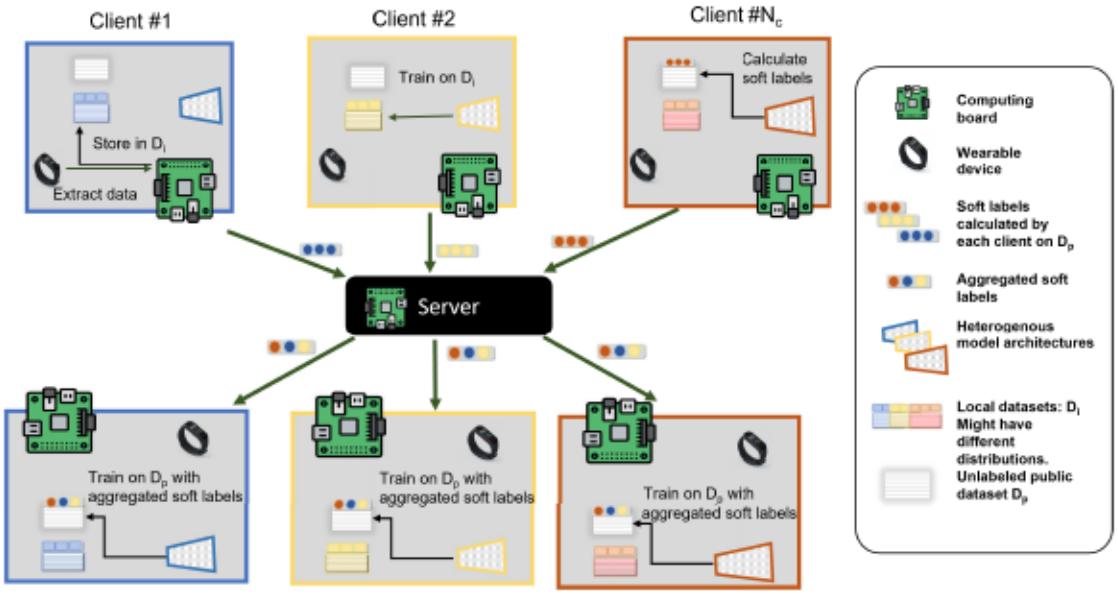

Figure 3.6: An overview of the proposed heterogeneous federated learning with knowledge distillation architecture. Each client owns a local dataset, an independently designed model, and a shared dataset. By utilizing knowledge distillation, clients use the shared dataset to transfer the knowledge they learned from local datasets by communicating their soft labels on the shared dataset with all clients. We propose an additional step where the shared dataset D_p is augmented to be D^r_{Aug} in order to enhance performance.

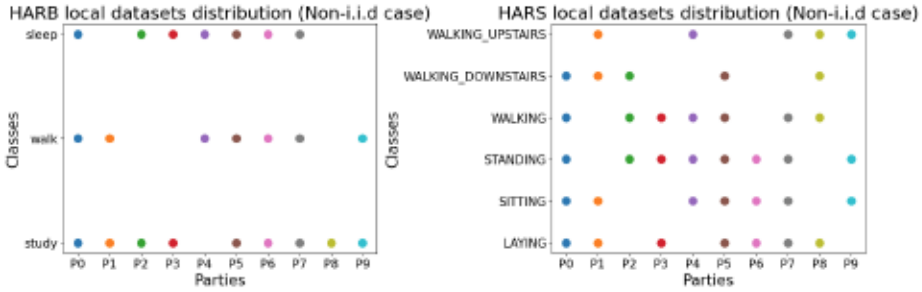

Figure 3.7: From left to right, the figures show the non-i.i.d distribution in the federated learning experiments of HARB and HARS datasets, respectively. Points show whether a party/client possesses a particular class in his local dataset D_i or not. For example, in the HARS dataset, the party $P2$ has three classes out of six. All clients are tested against all classes (The test dataset D_t is the same for all parties). Each color in the scatter plot refers to a client.

To distill the knowledge between clients, we need a shared dataset that has a distribution similar to the local datasets' distributions that the clients are trying to collaboratively learn. In both datasets, 100 sample points from the train set were employed as a public dataset where they are shared between clients and used to calculate soft labels for knowledge distillation. In a real-world scenario, the public dataset is shared and broadcasted at the beginning of the FL processing by the server (e.g., fitness band company) as a medium for communication.

3.7.4 Model Architecture Selection

There are various reasons to apply Federated Learning (FL) on heterogeneous model architectures. For instance, a smartwatch manufacturer may release a new device annually with more resources, and thus, it can run more expensive deep Neural Networks (NN). To make use of the data generated on users' devices while protecting their privacy, this manufacturer may want to apply FL among these devices. We tested our FL system, FedAKD, on a group of heterogeneous NNs with a varying number of parameters for two datasets, HARB and HARS [68]. We build ten heterogeneous models using different sequence processing units such as Long Short Term Memory (LSTM) units [69] and one-dimensional Convolutional Neural Network (1DCNN) layers [70] with different types of activation functions including Relu, Sigmoid, and Tanh, and a dropout layer [71] to prevent overfitting. A distinct feature of FedAKD is that it allows clients to choose the optimizer, which is usually controlled by the server in gradients/weights-based FL methods. We set the optimizer in the model variants to be built to be one of the following three optimizers: Stochastic Gradient Descent (SGD) [72], Adam [73], and RMSprop [74].

The models participating in the HARB experiment were selected based on a custom

template deep learning model for each dataset. One hundred variant models of that template were generated, and we used hyperparameter tuning to sample ten models that cover the range of performance and learning capacity of the whole group. A correlation can be observed between the number of parameters and the models' performance. Other factors that impact the performance of the model in addition to the number of parameters and the model architecture include the used learning rate and the optimizer. Table 3.3 shows the architecture details of the HARB models.

In model selection for the HARS experiment, we built a model with random hyperparameters based on the template model and manually tuned its hyperparameters until we obtained good performance on the HARS dataset. The other models were derived from the initial model by randomly changing numerical values such as the number of units in dense layers and the drop rate, and randomly selecting other categorical hyperparameters such as the optimizer and the activation functions. Table 3.4 shows the architecture details of the HARS models.

The goal of the model selection step is to obtain ten models that have distinct learning capacities to evaluate FedAKD's ability to collaboratively boost the performance of these models. Instead of tuning model hyperparameters to find a good balance between size and performance, another approach called feature selection trains a particular model architecture on different feature sets. Feature selection is also used in centralized training of DL models; however, this is out of the scope of this work as we are interested in evaluating FedAKD in the FL setting [68].

3.8 Results and Discussion

In this section, we evaluate the proposed Federated Averaging Knowledge Distillation (FedAKD) algorithm on two Human Activity Recognition (HAR) datasets, namely HARS and HARB. The template models employed for the HARS and HARB datasets are shown in Figure 3.3, with the HARS template model on the right and the HARB template model on the left. Ten variant models with various sizes and hyperparameters are derived from these template models to evaluate FedAKD on both datasets using heterogeneous model architectures.

First, models are trained in a centralized manner on their respective datasets to assess their learning capacity. The training data used in centralized training is distributed, and each client receives 20 samples per class as their local dataset. It is important to note that the number of classes available to each client is different in the non-i.i.d case. Each model is trained in a centralized manner on its own local dataset and trained on the collected local datasets. Models' performance under these two training settings forms the lower and upper bound for our Federated Learning (FL) experiment.

The goal of FL is to push the performance of each model beyond its local effort and towards the performance that would be achieved if all local datasets were combined and made available for training. These lower and upper bounds are shown as horizontal dashed lines to the left and right, respectively, on the plots of the figure 3.8.

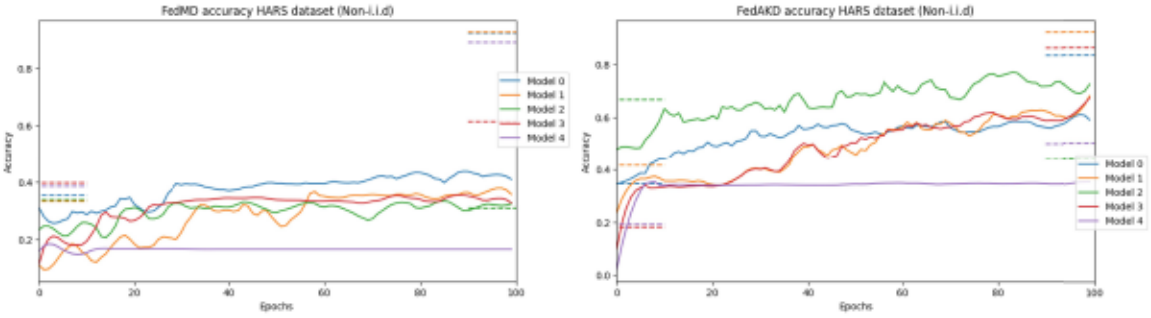

Figure 3.8: Ten heterogeneous models are trained in an FL setting using FedAKD and FedMD [13]. From left to right, the plots show five models (out of ten) trained collaboratively using FedAKD and FedMD, respectively, under the non-i.i.d case. It can be observed that models achieve better using our method. The five models shown here are the first five models in table 3.4. The dashed line to the left and the right of each graph represents models' performance on their local private dataset and models' performance on all the local datasets combined, respectively. In all experiments, every client has 20 samples only per activity (as shown in Figure 3.7) as his D_i, and $|D_p|= 100$. The lines are smoothed using the Sav–Gol filter to show the trend more clearly.

The tables 3.3 and 3.4 show the model architectures, sizes, and centralized training performance (accuracy) for the HARB and HARS datasets, respectively. The best-performing model on the HARS dataset achieved 95.4% accuracy, while the least-performing model achieved 34.5%. These two models had 17K and 242K parameters, respectively.

Table 3.5 summarizes the numerical results of our proposed FL system FedAKD and FedMD [13] on both datasets. FedAKD achieves better average accuracy gains than FedMD, particularly under the non-i.i.d case. On the HARS dataset, FedAKD obtained 25.4% and 27.5% under the i.i.d and non-i.i.d cases, respectively. On the other hand, FedMD achieved 24.5% and 7.2% under the i.i.d and non-i.i.d cases, respectively. For the HARB dataset, FedAKD obtained 12.7% and 0.4% under the i.i.d and non-i.i.d cases, respectively, while FedMD obtained 11.5% and -2.7% under the i.i.d and non-i.i.d cases, respectively. FedAKD outperforms FedMD on the HARB dataset by achieving a positive average accuracy gain under the non-i.i.d case.

The bar plot 3.9 shows a bar plot comparing test accuracy between two KD-based FL algorithms: FedMD and FedAKD on the loss functions: Mean Squared Error (MSE) and Kullback-Leibler (KL). It can be observed that using MSE as the loss function for the Knowledge Distillation (KD) mechanism produced the highest average accuracy gains

compared to using Kullback-Leibler divergence loss. While FedMD also uses MSE for KD, our method FedAKD applies mixup augmentation to the shared dataset to increase the variance of the shared soft labels resulting in higher accuracy than FedMD. Overall, MSE loss was found to be a better choice than KL loss for KD, which is also reported by [35].

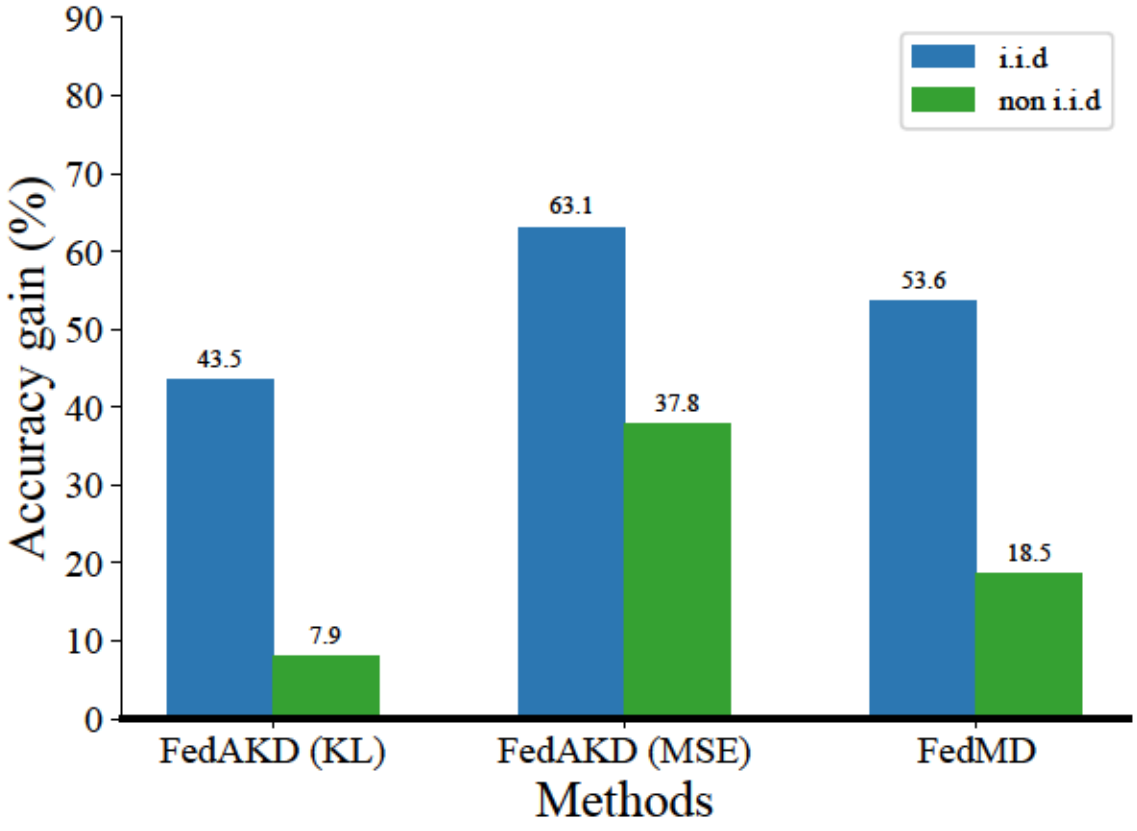

Figure 3.9: The performance of FedAKD vs FedMD using two loss functions: Kullback–Leibler (KL) divergence loss and Mean Squared Error (MSE)

Our experiments show that both FedAKD and FedMD outperform FedMD [13] under non-i.i.d scenarios. Figure 3.8 shows the test accuracy performance of the first five heterogeneous models in table 3.4 using FedMD [13] (to the left), and our proposed method FedKD (to the right). The bar plots in Figure 3.10 show the accuracy gains of each of the ten models in the FL experiment on the HARS dataset. The performance of individual models under FedAKD is better than their performance under FedMD, especially under the non-i.i.d case (left).

Table 3.3: Architecture details of the deep learning models participating in the HARB dataset FL experiment

Model ID	NF	KS	NCL	NLL	AF	OPT	LR	Size	Accuracy (%)	Accuracy gain (%) i.i.d		Accuracy gain (%) Non-i.i.d	
										FedMD	FedAKD	FedMD	FedAKD
Model 0	20	5	3	2	Relu	Adam	1e-4	28016	58.6	0	20	-6	-3
Model 1	20	5	1	1	Sigmoid	Adam	7e-5	7064	67.8	22	38	-8	-5
Model 2	20	9	2	1	Relu	Adam	4e-5	11004	60	-11	13	-12	-9
Model 3	10	9	2	2	Relu	RMSprop	1e-5	23556	60.9	-1	6	9	12
Model 4	20	9	2	2	Sigmoid	RMSprop	7e-5	5344	63.1	8	-7	2	6
Model 5	5	9	3	3	Tanh	Adam	1e-4	30601	58.9	42	27	2	5
Model 6	20	9	3	1	Relu	RMSprop	1e-5	8744	68	18	20	-20	-17
Model 7	10	18	2	3	Sigmoid	Adam	1e-5	3544	59.9	14	2	-14	-11
Model 8	5	9	1	3	Sigmoid	SGD	4e-5	12189	61.2	22	23	5	8
Model 9	20	9	1	3	Sigmoid	SGD	4e-5	1944	57.5	1	-15	15	18

Table 3.4: Architecture details of the deep learning models participating in the HARS dataset FL experiment

Model ID	D1	AF1	DO	D2	OPT	LR	Size	Accuracy (%)	**Accuracy gain per model (%)** **i.i.d**		**Non-i.i.d**	
									FedMD	FedAKD	FedMD	FedAKD
Model 0	290	relu	0.1	340	Adam	1e-3	291k	85.1	-7	45	4	25
Model 1	240	elu	0.25	300	Adam	1e-4	242k	34.5	-11	0	6	19
Model 2	200	selu	0.15	270	Adam	1e-5	207k	72.4	47	7	6	-7
Model 3	93	relu	0.2	200	RMSprop	1e-5	131k	87.1	55	61	0	42
Model 4	99	tanh	0.1	170	RMSprop	1e-4	113k	94.4	9	13	1	41
Model 5	90	elu	0.15	120	Adam	1e-3	78k	94.9	-5	-16	-16	22
Model 6	20	relu	0.25	70	RMSprop	1e-3	40k	86.9	-2	-1	21	43
Model 7	7	selu	0.1	30	Adam	1e-4	17k	95.4	52	18	13	28
Model 8	5	tanh	0.15	10	SGD	1e-3	5.5k	39.1	50	64	14	56
Model 9	5	tanh	0.25	8	SGD	1e-5	4.5k	87.4	54	63	23	8

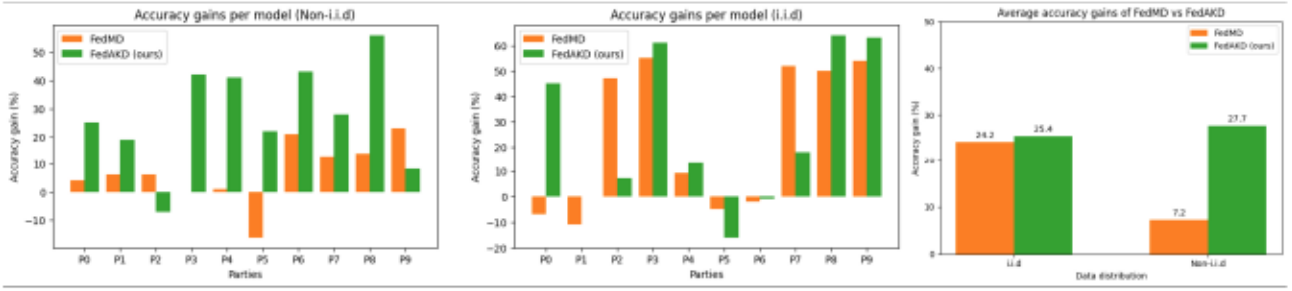

Figure 3.10: Comparison of individual models' accuracy gains achieved by our proposed method: FedAKD and FedMD [13] under i.i.d (**middle**) and non-i.i.d (**left**) cases on the HARS dataset. The bar plot to the (**right**) shows the average accuracy gains (across models) under both statistical conditions. FedAKD performs significantly better than FedMD in the non-i.i.d case

Table 3.5: Summary of the numerical results of the FL experiments on both the HARB and HARS datasets. Our proposed FL algorithm FedAKD outperforms FedMD on both datasets under i.i.d and non-i.i.d statistical scenarios.

Average accuracy gains of Federated Learning experiments (%)					
Dataset		HARS		HARB	
Data distribution		i.i.d	Non-i.i.d	i.i.d	Non-i.i.d
Method	FedMD	24.5	7.2	11.5	-2.7
	FedAKD	25.4	27.5	12.7	0.4

3.9 Conclusions

In this chapter, we introduce FedAKD, a federated learning algorithm that employs knowledge distillation to collaboratively train heterogeneous deep learning models. We evaluate FedAKD on two human activity recognition datasets: HARS, a tabular dataset extracted from smartphone-embedded inertial sensors, and HARB, a self-collected time-series dataset extracted from the gyroscope and photoplethysmography sensors of a fitness band. The FL experiments employ heterogeneous deep learning models with sizes ranging from 1.9k to 30k parameters for the HARB dataset, and from 4k to 291k parameters for the HARS dataset. FedAKD is also evaluated under extreme statistical heterogeneity, in which some clients are tested on activities/labels whose corresponding samples are not found in their local datasets; therefore, the knowledge needed to classify these labels has to be distilled from the other clients.

Compared to FedAvg [39], our proposed FL algorithm has much lower communication costs. In the FL experiments on the HARS dataset, FedAKD is shown to be 200 times more communication-efficient than FedAvg; FedAKD devices communicate a total of 8.8 KB vs. 1.8 MB on average if devices were to use FedAvg.

Compared to other knowledge distillation-based FL algorithms [13] that enable FL of heterogeneous models, our proposed algorithm FedAKD achieves higher accuracy gains for most participating models and significantly higher average accuracy gain across models on both datasets under i.i.d and non-i.i.d conditions. Specifically, for the HARS dataset, FedAKD obtains 25.4% and 27.5% under the i.i.d and non-i.i.d cases, respectively, while FedMD achieves 24.5% and 7.2% under the same statistical scenarios. Thus, FedAKD achieves an additional 20% of average accuracy gains compared with FedMD. This performance boost is attributed to the fact that FedAKD uses augmentation to generate a new variant of the public dataset in each communication round, which helps distill knowledge more efficiently.

3.10 Limitations and Future Work

In our FL experiments, the public dataset D_p was taken from the training set of the respective dataset. A better approach would be to select D_p from a different dataset that has a similar distribution to the local dataset. For example, in [13], the authors used MNIST as a public dataset to train heterogeneous models on the local dataset FEMINIST. In another experiment, when training models on CIFAR100, they used CIFAR10 as the public dataset. In our approach, we assumed that the public dataset (which contains only 100 samples) is made available to clients by the server at the beginning of FedAKD. In a real-world scenario, a company would collect some data and store them on its devices to be used as a public dataset during FL. This way, the company can protect users' data (by not using part of these data as a public dataset), and at the same time, the stored public dataset will have a distribution that is similar to the distribution of the local data that will be collected by users (since they are both collected using the same sensors).

In future work, we plan to integrate privacy-preserving techniques such as Differential Privacy (DP) with our Augmented Knowledge Distillation (AKD) algorithm. Additionally, we will conduct more analysis on class-level performance under this knowledge-distillation FL paradigm. Finally, we will evaluate FedAKD on other data modalities

Chapter 4

Enhancing Communication Efficiency in Federated Learning: Challenges and Approaches

Chapter 4 expands upon the concepts introduced in the previous chapter on Federated Learning through Augmented Knowledge Distillation (FedAKD) by focusing on communication efficiency in FL. This challenge is particularly relevant in IoT applications, where low-bandwidth networks can lead to significant communication overhead and slow, inefficient training. We examine prior work addressing this issue and explore how Knowledge Distillation-based FL algorithms can offer a communication-efficient solution. The chapter starts with an overview of FL and its applications, followed by an explanation of the communication overhead in FL. We then discuss the communication efficiency aspects of Knowledge Distillation-based FL algorithms, including FedAKD, and introduce a more efficient compressed version of FedAKD. To evaluate these algorithms, we conduct federated learning simulations using HAR datasets, demonstrating that Knowledge Distillation-based FL algorithms are more suitable for IoT applications due to their communication efficiency.

4.1 Introduction

In federated networks, consisting of potentially massive numbers of devices, communication is often orders of magnitude slower than local computation. Therefore, minimizing communication overhead is crucial for scaling up FL algorithms. Communication-efficient FL methods strive to optimize two key aspects of FL communication:

1. Total number of communication rounds
2. Size of client/server exchanged updates

In the background section, a review of the literature on communication efficiency in federated learning, comparing model-based federated learning algorithms with Knowledge Distillation (KD) FL algorithms in terms of communication overhead. After that, we introduce an efficient FL algorithm, Compressed Federated Learning with Augmented Knowledge Distillation (CFedAKD), which employs compression techniques on soft labels, similar to FedAKD. Then, we discuss a use case resembling a low-bandwidth drone-assisted network scenario, utilizing LoRa modules for exchanging federated learning updates between the server (drone) and clients (houses). In the performance evaluation section, a detailed description of the FL experiments conducted, including the HAR datasets used for evaluation, the baseline methods, and the heterogeneous model architectures employed.

The Results and Discussion section provides a Presentation and analysis of the test accuracy and communication cost for each learning method on the HAR datasets, taking into account soft label compression.

Finally, in the conclusion section, a summary of the chapter's methodology and findings, emphasizing the benefits of Knowledge Distillation-based FL algorithms for IoT applications.

4.2 Background

4.2.1 Communication Efficiency in Federated Learning

Effective communication is a fundamental aspect when designing methods for federated networks. In this section, we examine various contemporary works addressing this challenge, which can be broadly categorized into two primary areas: 1) Adaptable local updating, and 2) Compression techniques.

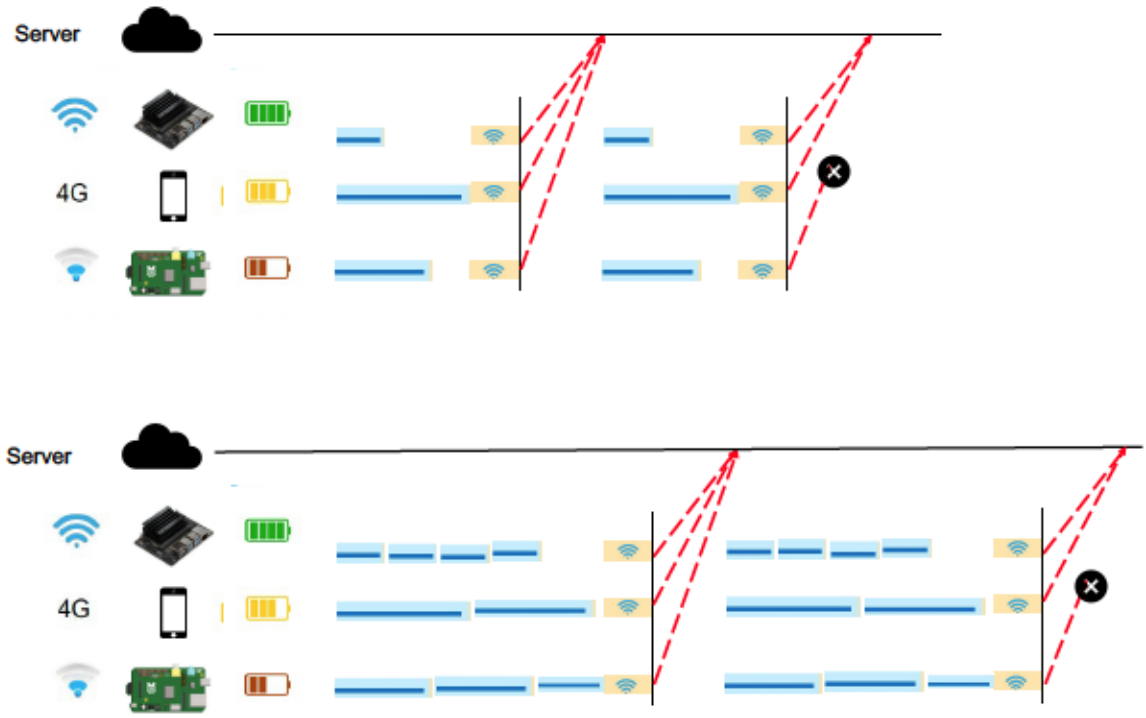

Figure 4.1: System heterogeneity and communication challenges in federated learning.

Mini-batch optimization methods process multiple data points simultaneously [75, 76], necessitating a fixed batch size to maintain a balance between computation and communication [77]. On the other hand, local-updating methods permit a varying number of local updates on each machine, offering a more adaptable balance between computation and communication, as depicted in Figure 4.1.

In the context of federated learning, where data is dispersed across numerous devices, FedAvg is a prevalent method that enables local updates to be computed on each client device and subsequently averaged to produce a global update. However, FedAvg may diverge when data is heterogeneous, meaning that data distribution across client devices differs significantly. To tackle this issue, various methods have been proposed, such as FedProx [78], which introduces a regularization term to the objective function to promote similarity between local models on each client device. In general, these optimization methods are vital for achieving efficient and effective machine learning in distributed environments.

While local updating methods can reduce the total number of communication rounds, model compression techniques like sparsification, subsampling, and quantization can considerably decrease the size of messages exchanged during each round [8, 79].

Table 4.1: Communication overhead between model-based and model-agnostic FL methods. Here Z, S, θ refers to FedMD soft labels, FedAKD, soft labels, and model weights, respectively.

Method	Weighting	Communication overhead
FedMD	UW	$\lvert Z_i^r\rvert + \lvert Z^r\rvert$
	PW	$\lvert Z_i^r\rvert + \lvert P_i^r\rvert + \lvert Z^r\rvert$
FedAKD	UW	$\lvert S_i^r\rvert + \lvert S^r\rvert$
	PW	$\lvert S_i^r\rvert + \lvert P_i^r\rvert + \lvert S^r\rvert$
FedAvg	UW	$\lvert \theta^r\rvert + \lvert \theta^r\rvert$
	PW	$\lvert \theta^r\rvert + \lvert P_i^r\rvert + \lvert \theta^r\rvert$

4.2.2 Model-based Federated Learning and Knowledge Distillation-based Federated Learning

Federated learning is a technique for training machine learning models on decentralized datasets where data is distributed across a network of clients. Given a set of clients $\mathbf{C} = C_1, C_2, \ldots, C_{N_c}$, where N_c represents the number of clients and C_i denotes the i^{th} client, each client C_i possesses a local dataset D_i, where $D = D_i i = 1^{N_c}$, and a local model f_i parameterized by weights θ_{f_i}. The global model f is initialized with a set of weights θ_0.

In federated learning with Knowledge Distillation, the clients each have their private datasets, D_i, and a shared public dataset, D_p, that is used to transfer knowledge. The clients also have independently designed models, f_i. The goal of federated learning is to train the models f_i on the private datasets D_i without explicitly sharing the data, in order to achieve the performance that would be obtained if the models were trained on the combined private dataset $D = D_{i_{i=1}}^{N_c}$.

In centralized Knowledge Distillation (KD), given an unlabeled dataset and a trained teacher model, the aim is to use the soft labels generated by the teacher model to train a student model. To apply KD to the FL context, we utilize a proxy dataset D_p shared among all clients to calculate their soft labels. Then, these local soft labels are transmitted to the server for aggregation into global soft labels, which are subsequently sent back to the clients to train on the labeled dataset (D_p, S^r). This process helps improve the performance of the federated learning models while maintaining communication efficiency.

Table 4.1 compares the communication overhead of various Federated Learning (FL) algorithms, emphasizing the importance of communication efficiency in FL. Specifically, it juxtaposes two model-agnostic FL algorithms, FedMD and FedAKD, known as Knowledge Distillation-based FL algorithms, with one model-based FL algorithm, FedAvg.

The table presents the communication overhead for two weighting schemes: uniform weighting (UW) and performance-based weighting (PW). In the PW scheme, the perfor-

Table 4.2: Train/test splitting and local sets partitioning of the HAR datasets used in this work.

Dataset	HARS	Depth	Harbox	IMU
Total train set size	10,800	3,544	22,657	683
Total test set size	2,947	-	-	-
Local train size (per client):	200 x 6 = 1,200	418-696	80-737	133-146
Local test size (per client):	2,947	895-902	55-317	58-63
Public/Shared set size:	736	449	185	136

Table 4.3: The characteristics of the four HAR datasets used to evaluate the FL algorithms considered in this work.

Dataset	HARS	Depth	Harbox	IMU
Num of activities	6	5	5	3
Activities	Walk, Walk up-stairs, Walk down-stairs, Sit, Stand, and Lay	Good, Ok, Victory, Stop, and Fist	Walking, Hopping, Phone calls, Waving, and Typing	Walking in corridor, Walking upstairs, and Walking downstairs
Sensors	Smartphone Intertial Sensors	Depth camera	9-axis IMU	IMU
Data modality	Tabular	Image	Tabular	Tabular
Data dimension	(561)	(30,30,1)	(900)	(900)
Num of clients	9	8	115	6

mance of each client (P_i^r) in round r is utilized to determine the weight assigned to that client.

For FedMD and FedAKD, the communication overhead is measured in terms of the size of the soft labels (Z or S) that need to be transferred between the server and clients. The overhead is smaller in comparison to FedAvg, which measures communication overhead in terms of the size of the model weights (θ) that need to be transferred. This highlights how Knowledge Distillation-based FL algorithms can contribute to more efficient communication in federated learning scenarios.

In the case of FedMD and FedAKD with PW, the communication overhead is the sum of the sizes of the soft labels for each client (Z_i^r), the personalized weight for each client (P_i^r), and the global soft labels (Z^r). By reducing the size of the data exchanged during each communication round, these algorithms can substantially enhance communication efficiency, particularly in low-bandwidth networks or IoT applications.

4.2.3 Federated Learning with Augmented Knowledge Distillation

The authors in [14] proposed a KD-based FL algorithm that employs mixup augmentation [15] to generate a dataset D^r_{Aug} during each global round r. This algorithm demonstrates the potential of Knowledge Distillation-based FL algorithms to improve communication efficiency by leveraging smaller data transfers between the server and clients.

By applying Mixup augmentation to D_p and D^r_d, the synthesized dataset D^r_{Aug} contributes to reducing communication overhead while maintaining consistency across all clients. Moreover, the utilization of mixup augmentation [15] and the permutation of the public dataset not only helps prevent overfitting but also enhances generalization, thereby improving the efficiency and performance of FL algorithms in distributed settings.

In summary, Knowledge Distillation-based FL algorithms, such as FedMD and FedAKD, can offer a communication-efficient solution for federated learning, especially in IoT applications or other scenarios with low-bandwidth networks. By minimizing communication overhead and optimizing the balance between computation and communication, these algorithms can greatly improve the scalability and performance of FL systems.

4.3 Proposed Compressed Federated Learning with Augmented knowledge distillation

Algorithm 2 Compressed FedAKD Algorithm

Input: Public dataset: D_p, Test dataset: D_t, Local dataset of client i: D_i, Independently designed local model of client i: f_i, Number of communication rounds: R, Number of epochs for local training: E_l, Number of epochs for KD training: E_{KD}, Loss function for local training: L_l, Loss function for KD training: L_{KD}, Total number of participating clients: N_c, Fraction of clients participating at any given round: K
Output: Collaboratively trained local model f_i
Client i designs f_i and initializes θ_i
for round $r = 1$ to R **do**
    $N_k = K \cdot N_c$
    Select N_k clients randomly
    $\rho^r, \alpha^r \leftarrow$ server randomly generated
    Broadcast ρ^r, α^r
    for client $i = 1$ to N_k **do**
        $D_d^r \leftarrow \text{permute}(D_p, \rho^r)$
        $D_{Aug}^r \leftarrow \text{mixup}(D_p, D_d^r, \alpha^r)$
        $S_i^r \leftarrow$ calculates soft labels on D_{Aug}^r
        $CS_i^r \leftarrow Quantize(S_i^r)$
        $P_i^r \leftarrow$ calculates accuracy on D_t
        Client i sends CS_i^r and P_i^r to the Server
    end for
    $CS^r \leftarrow \sum_{i=1}^{N_k} \frac{P_i^r CS_i^r}{\sum_{k=0}^{N_k} P_k^r}$
    Broadcast CS^r
    for client $i = 1$ to N_k **do**
        for epoch $e = 1$ to E_{KD} **do**
            $\theta'_{f_i} \leftarrow \theta_{f_i} - \eta \cdot \frac{1}{|D_{Aug}^r|} \sum_{(x,y) \in (D_{Aug}^r, CS^r)} \nabla \mathcal{L}_{KD}(\theta_{f_i}, f_i(x), y)$
        end for
        for epoch $e = 1$ to E_L **do**
            $\theta'_{f_i} \leftarrow \theta_{f_i} - \eta \cdot \frac{1}{|D_i|} \sum_{(x,y) \in D_i} \nabla \mathcal{L}_{CCE}(\theta_{f_i}, f_i(x), y)$
        end for
    end for
end for

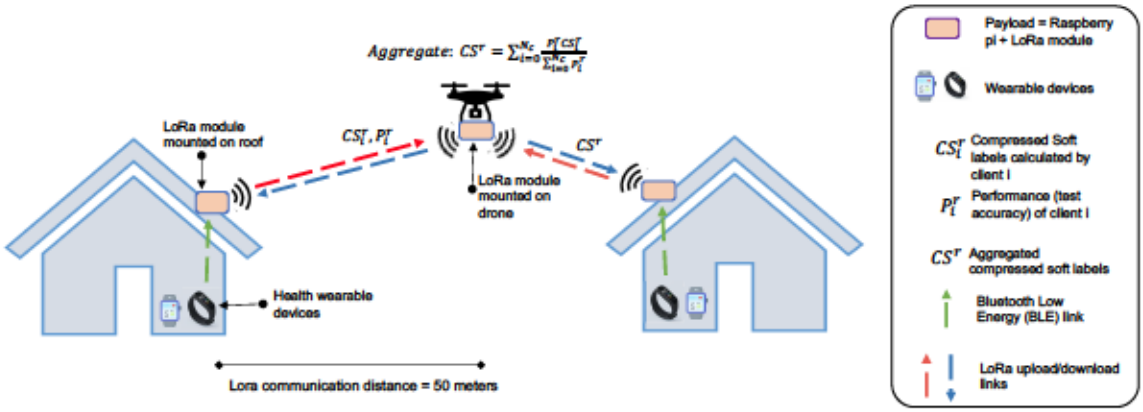

Figure 4.2: An overview of the proposed Compressed Federated Learning via Augmented Knowledge Distillation algorithm (CFedAKD) on a drone-aided LoRa network.

In this section, we present the Compressed Federated Learning with an Augmented knowledge distillation algorithm (CFedAKD), which is built on top of FedAKD and uses a compression scheme to reduce the size of soft labels. While FedAKD uses soft labels S_i^r as the basis for communication between the clients and the server, CFedAKD applies quantization to S_i^r. This is done by normalizing S_i^r using Min-Max normalization, multiplying by 255, and then casting type from floating point (46 bits) to unsigned integer (8 bytes). The compressed local soft labels, CS_i^r, are then sent to the server to be aggregated as:

$$CS^r \leftarrow \sum_{i=1}^{N_k} \frac{P_i^r \cdot CS_i^r}{\sum_{k=0}^{N_k} P_k^r} \tag{4.1}$$

The server broadcasts CS^r to all clients to train on it. Before that, CS^r is normalized and cast back to float. After receiving the global soft labels, each client undergoes two training processes. The first training process for E_{KD} epochs uses the KD loss to digest the distilled knowledge CS^r, we use MSE loss for KD. The second training process for E_l epochs is the standard local training using the Categorical Cross Entropy (CCE) loss to fine-tune the model on the local dataset D_i. The complete algorithm for CFedAKD is depicted in algorithm 2.

4.4 Use Case: Implementing CFedAKD on Low-Bandwidth LoRa Networks

Drone-assisted LoRa networks have found numerous applications in the Internet of Things (IoT), particularly in monitoring rural areas. Examples of such applications include localization [80] and forestry monitoring [81]. To overcome the challenges of limited internet connectivity in rural health monitoring, we propose a Drone-assisted LoRa network (DORA)

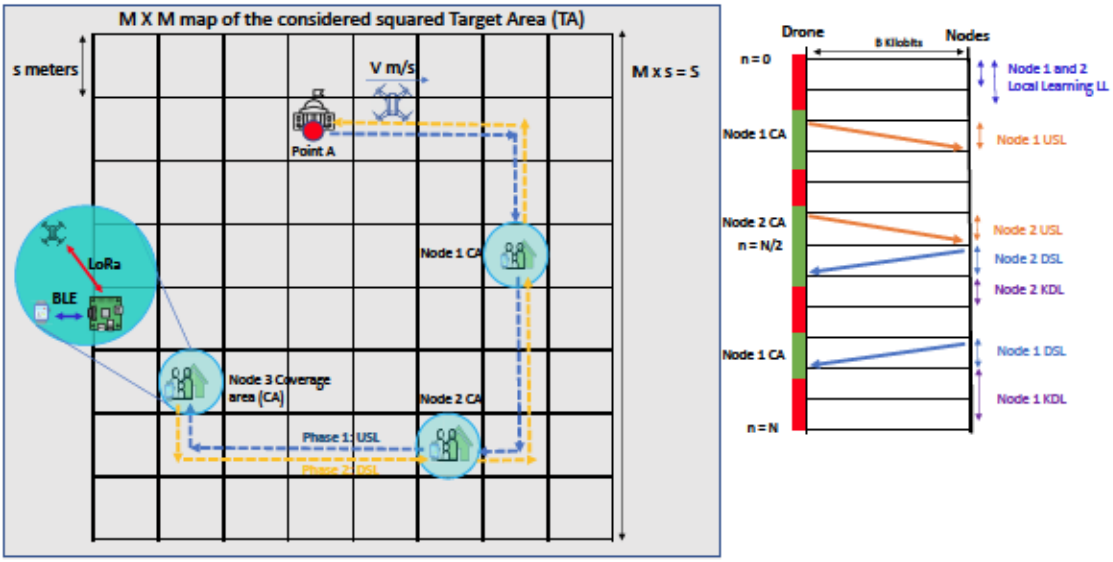

Figure 4.3: Left: The drone traverses an optimized path while performing KD-based FL over two passes across the considered Target Area (TA). The drone communicates lightweight messages via LoRa link only if it is inside the Coverage Area (CA). Right: Timeline of the FL communication phases for a map of 1 drone and 2 nodes.

and evaluate the performance of CFedAKD for training distributed deep learning models on edge devices. The proposed Drone-aided LoRa network is designed to enable smart health applications utilizing wearable devices and lightweight FL.

We consider a target area (TA) represented by an $M \times M$ map, containing N_k houses/nodes, each denoted by $K \in K = 1, 2, \ldots, N_k$, with house k having spatial coordinates $\mathbf{p_k} = (x_k, y_k)$. The map is divided into M^2 blocks, each with a side length of s meters, resulting in a total side length $S = M \cdot s$. Each house has an access point AP_k, which consists of a Raspberry Pi board and a LoRa Module. Each AP_k connects to one or more wearable devices via BLE to extract sensory data and train a local model using the proposed communication-efficient FL algorithm.

A drone equipped with an access point AP_d travels across the TA starting from point A at $\mathbf{p_A} = (x_A, y_A)$ and returning to the same point. The drone flight consists of two phases: The upload Soft Labels (USL) phase and the Download Soft Labels (DSL) phase, as shown in figure 4.3. In the USL phase, the drone visits each node $p_k, \forall k \in K$ to receive FL updates in the form of Soft Labels (SL) via LoRa communication. Once the drone has collected updates from the last node, the DSL phase begins in which the drone returns along the USL path, stopping at each node to download the aggregated SL via LoRa communication before returning to the starting point A. Let T_{USL} and T_{DSL} denote two equal time durations in which the drone completes the USL and DSL phases, respectively,

while traversing linear distances of D_{USL} and D_{DSL}.

$$T = T_{USL} + T_{DSL}, \quad D = D_{USL} + D_{DSL} \tag{4.2}$$

The drone completes one global FL round per journey, divided between two identical paths for uploading and downloading FL messages between AP_k (the server) and $AP_k \forall k \in K$ (the clients). The total time T is divided into N_t equal time slots $n \in \{1, 2, \ldots, N_t\}$, each lasting t seconds, such that $T = N_t \cdot t$ and $T_{USL} = T_{DSL} = \frac{N}{2} \cdot t$. The drone's speed V and height VD are fixed, and it follows a trajectory $\mathbf{P_d} = \{\mathbf{p_{d,n}}\} \in \mathbb{R}^{N_t \times 2}, \forall n \in \{1, 2, \ldots, N_t\}$, a sequence of coordinates throughout its path. $\mathbf{p}_n^d$ represents the drone's coordinates at time n. The first and last points have the same coordinates: $\mathbf{p_A} = \mathbf{p_{d,0}} = \mathbf{p_{d,N_t}} = (x_A, y_A)$. And the horizontal distance between the drone and node k is given by:

$$d(k, \mathbf{p_{d,n}}) = \sqrt{(x_{d,n} - x_k)^2 + (y_{d,n} - y_k)^2} \tag{4.3}$$

We define a circular Coverage Area CA_k with radius $r = \frac{s}{2}$ centered at $\mathbf{p_k} \forall k \in K$. The drone access point AP_d can communicate with an access point AP_d only if the distance is below the threshold distance r:

$$dr(k, \mathbf{p_{d,n}}) = \begin{cases} R_b, & \text{if } d(k, \mathbf{p_{d,n}}) \leq r \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

Table 4.4: The used Lora, distance, and Reliable Data Transfer (RDT) parameters in our experiments.

Parameter	Value
Spread Factor	12
Bandwidth	125 KHz
Carrier frequency	915 MHz
Coding rate	4/5
Programmed Preamble	7
Lora engine	SX1276
CA radius	50 meters
VD	8 meters
Delay	3 seconds
Window size	3 packets
Timeout	3 seconds

Here, $dr(k, \mathbf{p_{d,n}})$ represents the data rate for client k at time n, and R_b is the nominal data rate. Because of the low bandwidth of LoRa and its high drop rate, we define a coverage area (CA) with radius r in which the signal quality is above the sensitivity of the receiver antenna and therefore is acceptable for sending FL messages. This ensures that the

drone access point AP_d communicates with an access point AP_k only when the distance is below the threshold distance r as shown in equation 4.4. We set the radius of CA $r = 50$ meters based on our drone-loRA distance test shown using the parameters specified in table 4.4.

Due to the limited resources of the drone, it is essential to minimize the total trip time T. To address this optimization problem, we tackle the minimization of the total linear distance D covered by the drone and the message size MS as two distinct problems.

For the first problem, we formulate the goal of minimizing the distance D covered by the drone as a path optimization problem in which the drone aims to traverse a graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_{N_v}\}$ is the set of $N_v = K+1$ vertices (home access points $AP_i, \forall i \in I$ and the starting point A which has location $\{x_A, y_A\}$); $E = \{e_{ij}\}, \forall i, j \in V, i \neq j$ is the set of edges; and w_{ij} is the weight (distance) of edge e_{ij} between vertex v_i and vertex v_j.

In the USL phase, the drone starts from point A and follows a trajectory $\mathbf{P_d}$ which passes through and stops to send and receive updates at, all house locations. In the DSL phase, the drone starts from the last house visited in the USL,the drone access point AP_d aggregates the received local soft labels $SL_k, \forall k \in K$ and travels back its USL path, this time downloading the aggregated soft labels SL, to $AP_k, \forall k \in K$ until it reaches the starting point.

We realize this whole trip which lasts time T over distance D and includes both phases, is similar to the Traveling Salesman Problem (TSP) with modification and can be approached in two steps: First, formalizing the drone path optimization as a TSP problem and finding a Hamiltonian cycle H. Second, we apply post-processing steps, described later in this sub-section, to derive G_d from H where G_d is the path (order of vertices) the drone should follow to perform the two FL phases USL and DSL with minimum D.

The Hamiltonian cycle H is a permutation of vertices forming a closed loop that visits each path once. To ensure that this permutation minimizes the distance D covered by the drone, TSP minimizes the sum of distances masked by modeling the edges included in H using a binary decision variable x_{ij}, corresponding to each edge $e_{ij} \in E$, where

$$x_{ij} = \begin{cases} 1, & \text{if } e_{ij} \text{ is included in } H \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

The Traveling Salesman Problem (TSP) is formally defined as

$$\min_{H} \sum_{e(ij) \in H} w_{ij} = \min_{x_{ij}} \sum_{i=1}^{N_v} \sum_{j=1, j \neq i}^{N_v} w_{ij} x_{ij} \tag{4.6}$$

under the constraints:

1. Each node is visited exactly once:

$$\sum_{j=1, j\neq i}^{N_v} x_{ij} = 1, \quad \forall i \in V \tag{4.7}$$

2. No mini-cycles (sub-tours) are allowed, meaning any cycle in the solution must include all nodes:

$$\sum_{i,j\in S} x_{ij} \leq |S|-1, \quad \forall S \subset V, 2 \leq |S| \leq N_v - 1 \tag{4.8}$$

Notice that the first constraint states each node is visited once, however in our case, each node is visited twice (one time in the USL phase and another in the DSL phase). To address this, We manually remove the last edge connecting the starting vertex to the last vertex (v_{N_v}) in the solved H, we then append the vertices from the second-to-last vertex (v_{N_V-1}) to the starting vertex following the same path in reverse order. This will produce the drone path G_d which can be used to produce the drone trajectory $\mathbf{P_d}$.

With regards to the second goal of minimizing MS, we utilize a KD-based FL algorithm that leverages soft labels like FedAKD or CFedAKD for sending updates with significantly smaller sizes compared to model weights used in standard FL. Figure 4.2 demonstrates the implementation of CFedAKD over the proposed Drone-LoRa system (DORA).

In the case of CFedAKD, the local updates, denoted as CS^r, are compressed soft labels. The time required to exchange updates with each home access point under CFedAKD is measured as 4.8 minutes for sending the compressed soft labels, as per Table 4.5, which means a total of 2x4.8 = 9.2 minutes for sending and receiving updates. To minimize waiting time, the appropriate drone model should be chosen based on the time needed to exchange updates with all homes. Our proposed CFedAKD approach demands less waiting time compared to FedMD/FedAKD and FedAvg, which require 41.2 minutes and over 30 hours, respectively, due to the relatively large size of the model file and the low bandwidth of LoRa.

The traffic models for both the model-based and communication-efficient KD-based FL are presented in the following sections.

4.4.1 Model-based Federated Learning traffic model

Each global round in model-based FL [82] consists of three ordered phases: The download phase (DL), the Learning phase (L), and the Upload phase (UL).

In the DL phase, each node downloads the weights of the server-controlled global model f denoted as θ^r, where r is the current global round. In the L phase, each client updates his local copy of weights using

$$\theta_k^r \leftarrow \theta^r - \eta \frac{1}{|D_i|} \sum_{(x,y)\in D_i} \nabla \mathcal{L}(\theta_{t-1}, f(\theta^r, x), y) \tag{4.9}$$

where η is the learning rate and $\mathcal{L}$ is the loss function. In the case of a classification problem, a Categorical Cross Entropy function is usually used $\mathcal{L} = \mathcal{L}_{CC\mathcal{E}}$.

Finally, in the UL phase, clients upload the locally trained weights to be aggregated at the server as

$$\theta^{r+1} \leftarrow \frac{1}{N_k} \sum_{i=1}^{N_k} \theta_k^r \tag{4.10}$$

This process is repeated until convergence or until a predetermined number of rounds have been completed.

Uploading and downloading model weights requires high bandwidth communication. In our drone-based system, the drone flight time is limited, therefore, we use a different FL approach that is based on the concept of KD.

4.4.2 Knowledge Distillation-based Federated Learning traffic model

In centralized settings, Knowledge Distillation (KD) is a technique used to train a student model using a trained teacher model, i.e. given an unlabeled dataset and a trained teacher model, the goal is to use the soft labels produced by the teacher model to train a student model. To apply KD to the FL context, we use a proxy dataset D_p shared with all clients to calculate their soft labels on it. Then these local soft labels are sent to the server to be aggregated into global soft labels and sent back to clients to train on the (D_p, SL^r) labeled dataset.

In FL with KD, each node has its private dataset, D_k, and another public dataset, D_p, that is shared with all other nodes and is used to transfer knowledge. A major advantage of KD-based FL over model-based FL, which was presented in the previous section, is that the former gives nodes the freedom to design their own model architecture f_k. On the other hand, model-based FL [39, 82] assumes a server-controlled model architecture.

The global cycle in KD-based FL consists of four ordered phases: Local Learning phase (LL), Upload Soft Labels phase (USL), Download Soft Labels (DSL), and Knowledge Distillation Learning phase (KDL).

Both the USL and DSL phases are described earlier. In these phases, soft labels are uploaded from nodes/clients to the server (the drone), then the server aggregates and downloads the global soft labels to nodes, respectively.

In the LL phase, Cross Entropy Loss (CCE) is used to train the locally designed model

f_k on the locally labeled dataset.

$$\theta_k^{r,'} \leftarrow \theta_k^r - \eta \cdot \frac{1}{|D_k|} \sum_{(x,y)\in D_k} \nabla \mathcal{L}_{CCE}(\theta_{f_k}, f_k(x), y) \tag{4.11}$$

After the local clients have trained their models on the local data, each client uploads his local soft labels SL_k^r to the drone in the USL phase. In the DSL phase, the drone traverses back its USL path and downloads the aggregated soft labels SL^r to be used for training in the KDL phase.

$$SL^r \leftarrow \sum_{i=1}^{N_k} \frac{P_i^r \cdot SL_k^r}{\sum_{k=0}^{N_k} P_k^r} \tag{4.12}$$

In the KDL phase, distance functions like the Mean Squared Error (MSE) or Kullback-Leibler divergence loss (KD) are used to train f_i on the public unlabeled dataset leveraging the downloaded soft labels.

$$\theta_k^{r+1} \leftarrow \theta_k^{r,'} - \eta \cdot \frac{1}{|D_p|} \sum_{(x,y)\in(D_p,SL^r)} \nabla \mathcal{L}_{KD}(\theta_k^{r,'}, f_k^*(x), y) \tag{4.13}$$

where $\theta_k^{r,'}, \theta_k^{r,''}, and f_k^*(x)$ represent, the model weights after local dataset training, the model weights after knowledge distillation training, and a similar architecture to f_k constructed by removing the last layer [62] or increase the SoftMax temperature [35] to smooth the distribution of the output vector, which is the soft labels $SL_k = f_k^*(D_p)$.

4.5 Performance Evaluation

4.5.1 Datasets

In this chapter, four Human Activity Recognition (HAR) datasets are used to evaluate the proposed FL algorithms. First, three datasets (HARS, Depth, and IMU) are used to evaluate the performance of the different considered FL methods, and a larger fourth dataset (HARBox) is used to evaluate the performance of KD-based FL methods while scaling the public dataset D_p on which the soft labels are calculated and KD is performed. We already used HARS in the previous chapter to evaluate FedAKD algorithm, in this chapter, more datasets are utilized to test the generalizability of FedAKD and its compressed relative CFedAKD. Table 4.3 shows the characteristics of the four datasets. The datasets cover different modalities: tabular/structured and image-based. Three of these datasets: Depth, IMU, and HARBox are collected by [38] with decentralized training in mind, therefore these datasets are pre-partitioned into clients, as shown in table 4.2. The number of clients of each dataset range from 6 to 115 clients.

1. IMU-based dataset:

   The IMU-based dataset was created using an off-the-shelf Inertial Measurement Unit (IMU) module to record three different walking activities. Seven participants, comprising 4 males and 3 females, were recruited to perform the activities, which included walking in the corridor, walking upstairs, and walking downstairs, in two different buildings. The IMU was set to a sampling rate of 50 Hz, resulting in each frame of data containing 9-axis IMU data. To capture the activity, a time window of 2 seconds was chosen, resulting in each recording being a 900-dimensional vector. This dataset presents a challenge due to its heterogeneity, as it includes data from various subjects and environments.

2. Depth-camera dataset:

   The Depth-camera dataset was created using depth cameras, which are preferred for activity monitoring and gesture control because of their ability to preserve user privacy. The dataset includes records of five types of gestures (good, ok, victory, stop, and fist) performed by two subjects in three different environments (outdoor, dark, and indoor) using a depth camera. To create the dataset, the region of interest (ROI) for each depth gesture was first obtained, and the depth values were then normalized to a range of 0-1. The resulting depth images were resized to 36*36 pixels. This dataset has a large number of records and each record has a relatively high number of dimensions, making activity recognition more challenging.

3. Smartphone-sensors dataset:

   The Smartphone-sensors dataset is a collection of recordings made by 30 individuals performing daily activities while wearing a waist-mounted smartphone with built-in inertial sensors. The goal of the dataset is to classify the recorded activities into one of six categories: WALKING, WALKINGUPSTAIRS, WALKINGDOWNSTAIRS, SITTING, STANDING, and LAYING. The dataset was collected from a group of 30 volunteers with ages ranging from 19 to 48, using a Samsung Galaxy S II smartphone equipped with an accelerometer and gyroscope. Data were collected at a rate of 50 Hz and labeled by watching the recorded video. The dataset was then divided into a training set and a test set, with 70% and 30% respectively. Data was preprocessed by applying noise filters and the sensor signals were divided into windows of 2.56 seconds with 50% overlap. Various time and frequency domain variables were calculated from each window to create a vector of features.

4. HARBox dataset: An Android App named "HARBox" was developed and released in this dataset to collect human activity recognition (HAR) data using users' own smartphones in a crowdsourcing manner. The App collects 9-axis IMU data from

users' smartphones while they perform five activities of daily life (ADL), including walking, hopping, phone calls, waving, and typing. The users label the activities themselves by clicking the "start" and "end" buttons in the App before and after each activity. After removing invalid and repeated data, valid submissions were obtained from 121 users (ranging in age from 17 to 55) with 77 different smartphone models. The original IMU data was resampled at 50Hz and a sliding time window of 2 seconds was used to generate a 900-dimensional feature for each data sample. This dataset is larger and more heterogeneous, making it useful for evaluating the scalability and robustness of different methods.

4.5.2 Baseline Federated Learning algorithms

We compare the performance and communication overhead of several Federated Learning (FL) algorithms, including:

1. FedAvg: A traditional FL approach where the server aggregates client model weights after they have undergone a certain number of local training iterations.

2. FedMD: A model-agnostic FL algorithm where clients share their soft labels Z_i^r, which are calculated on a shared public dataset. This method is well-suited for IoT applications with limited communication resources or applications where clients need to have control over the design of their local model.

3. FedAKD: A KD-based FL algorithm that improves accuracy by augmenting the shared public dataset D_p using Mixup augmentation in each global round. Clients communicate their soft labels S_i^r, which are calculated from the augmented version of the shared public dataset D_{Aug}^r. The augmentation variables are controlled by the server to ensure consistency across clients.

4.5.3 Heterogeneous Local Model Architectures

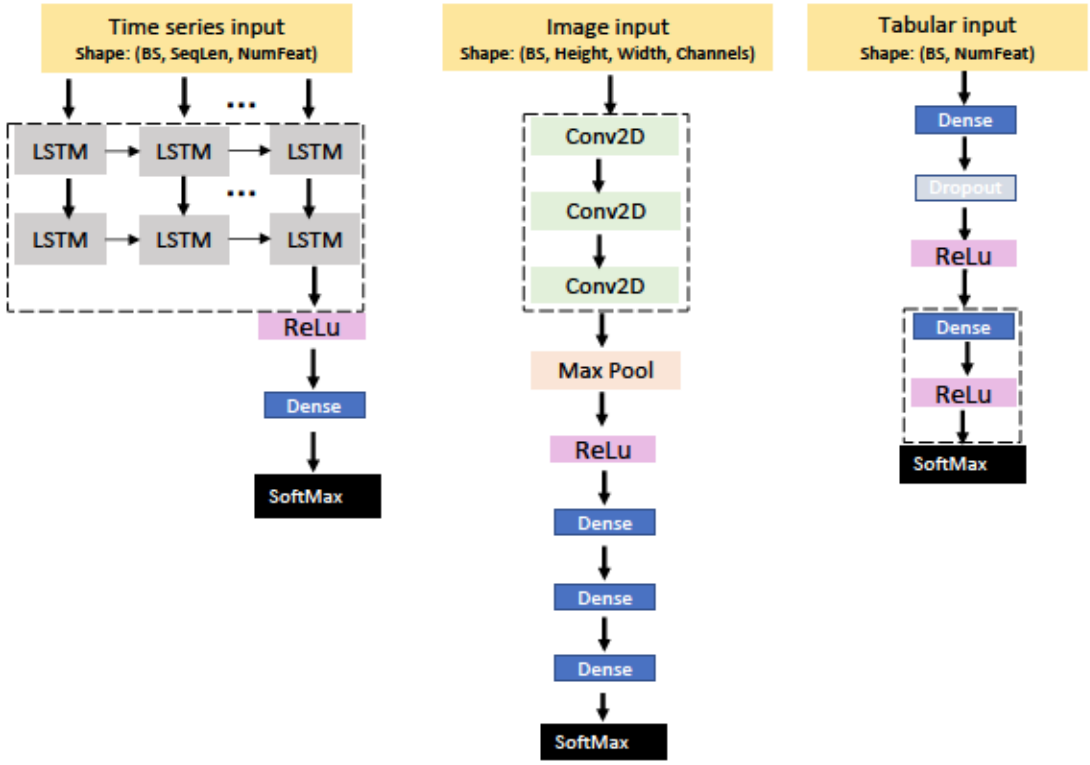

Figure 4.4: Left: Time series data template deep learning model architecture. Middle: Image data template deep learning model architecture. Right: Tabular data template deep learning model architecture. Because these architectures are used as templates for their respective datasets/tasks, the dashed box on each model highlights the variable layers that are modified to construct new variants of template models..

The evaluation of the proposed federated learning algorithms is based on the template deep learning model architectures depicted in Figure 4.4. These architectures, specifically designed for time series, image, and tabular data, are adapted to create new variants of template models by modifying the variable layers within the dashed boxes. By leveraging these template architectures, we can rigorously assess the effectiveness, generalizability, and robustness of the proposed federated learning algorithms in diverse real-world IoT applications.

4.6 Results and Discussion

In this section, we present the results of applying federated learning algorithms on HAR datasets. The considered baseline FL algorithms are shown in section 4.5.2 and the datasets

are shown in section 4.5.1. This section is split into two parts: Performance results and Communication results.

4.6.1 Performance Results

In this part, we present the test accuracy obtained by different FL algorithms on HAR datasets. The two figures 4.5 and 4.6 show the relative accuracy of two model-agnostic FL methods FedAKD and FedMD, and one model-based FL method. The scatter plots show the accuracy of these FL algorithms on four HAR datasets. The weighting schemes considered are uniform weighting and performance-based weighting. In performance-based weighting, each client calculates her test accuracy and sends it to the server to weigh each client's contribution according to her performance. We can observe that model-agnostic achieves comparable performance to model-based FL methods. Additionally, FedAKD and FedAvg outperform FedMD.

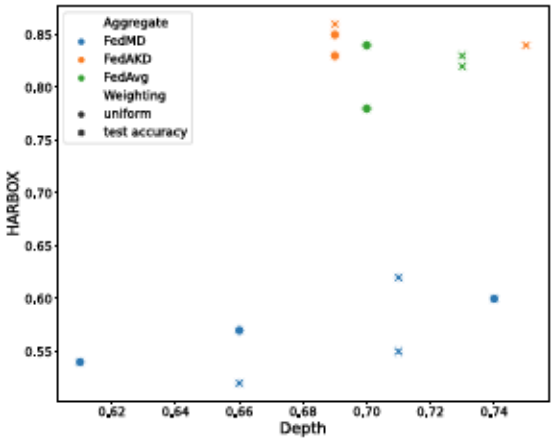

Figure 4.5: Test accuracy of the federated learning methods on HARBOX and Depth datasets. Points are styled according to the weighting scheme used.

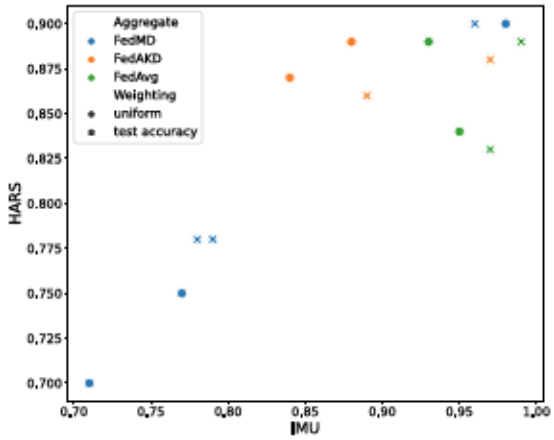

Figure 4.6: Test accuracy of the federated learning methods on HARS and IMU datasets. Points are styled according to the weighting scheme used.

In our explanation of how KD-based FL methods work, we discussed the importance of the public dataset to distill clients' knowledge obtained from training on the distributed local datasets. Figure 4.7 compares the impact of scaling the public dataset on the test accuracy. The performance increases as we increase the public dataset up until a certain point where the performance is constant.

4.6.2 Communication Results

In this section, we present and discuss the communication cost of various baseline FL algorithms. Figure 4.9 shows the sending time of model-agnostic (FedAKD/FedMD/CFedAKD)

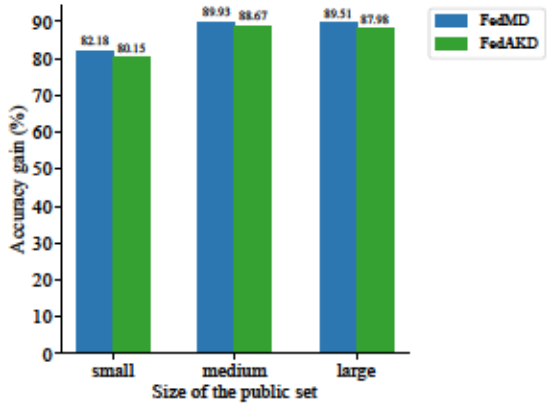

Figure 4.7: The test accuracy of the KD-based federated learning methods on the HARBOX dataset, while scaling up the public dataset (HARBOX) from one to four and eight times the average local dataset size.

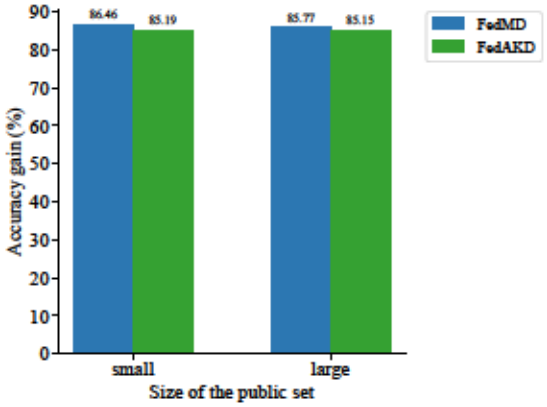

Figure 4.8: The test accuracy of the KD-based federated learning methods on the HARBOX dataset, while utilizing the IMU dataset as a public dataset.

Table 4.5: Sending deep model files vs soft labels using LoRa + Reliable Data Transfer (RDT) communication details. The proposed RDT method avoids losing packets as the LoRa communication protocol is originally unreliable (lossy communication). The file size of FedMD's Soft Labels (Z) and CFedAKD's Compressed Soft Labels (CS) is significantly smaller than the size of model files therefore they are more suitable for federated learning in bandwidth-limited environments.

Federated Learning algorithm	**FedAvg**			**CFedAKD**	**FedMD**
Basis of Communication	**Model 0**	**Model 1**	**Model 2**	**CS**	**Z**
File size (KiloBytes: KB)	256 KB	638 KB	466 KB	3 KB	12 KB
Num packets	12,005	30,066	21,907	49	210
Packets preparation time (second:s)	0.4 s	0.77 s	0.6 s	<1 ms	<1 ms
Sending time (minute:m)	1178 m	2947 m	2147 m	4.8 m	20.6 m
Data rate (bits/second: b/s)	29 b/s	29 b/s	29 b/s	83 b/s	77 b/s
Packet rate (Packet/s)	0.17	0.17	0.17	0.17	0.17

vs model-based (FedAvg) FL methods. And figure 4.10 shows the file sizes of the same FL methods, where the file size represents the size of the message exchanged between the server and the clients each round.

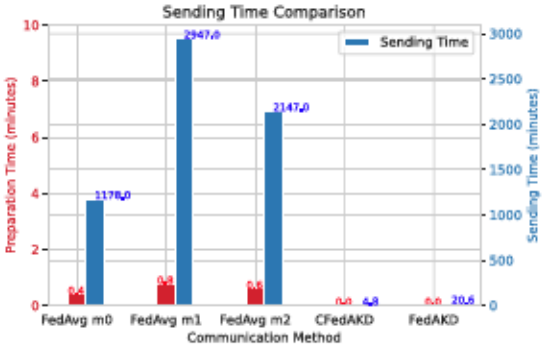

Figure 4.9: Preparation and Sending time of different federated learning methods.

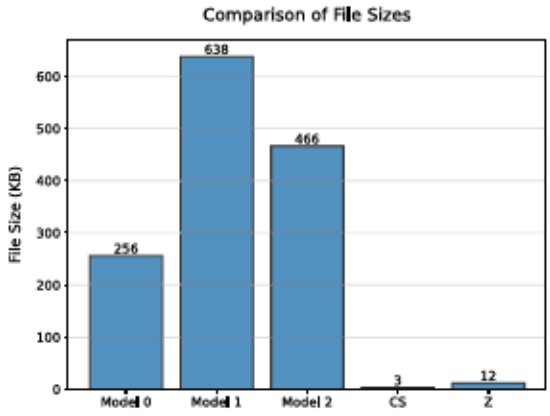

Figure 4.10: Client/Server Update sizes of different federated learning methods .

In this chapter, we proposed a Knowledge Distillation-based FL method that uses compression to reduce the size of soft labels. The scatter plots shown in figures 4.11 and 4.12 show the impact of compression on the test accuracy of FL algorithms. Additionally, figure 4.13 shows the accuracy of model-agnostic FL methods on different HAR datasets and how they compare to the accuracy obtained by FedAvg which is a model-based FL method. Compression is only applied to model-agnostic algorithms that use soft labels as the basis for communication converting FedAKD/FedMD to CFedAKD/CFedMD. The figure suggests that compression does not reduce accuracy. The figure also confirms that KD-based FL algorithms achieve comparable performance to model-based FL methods

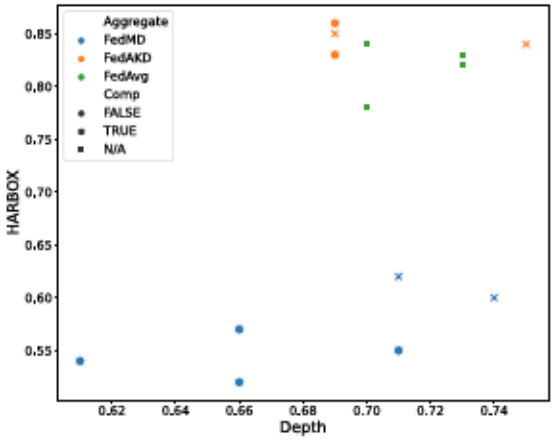

Figure 4.11: Test accuracy of the federated learning methods on HARBOX and Depth datasets. Points are styled according to whether soft label compression is applied.

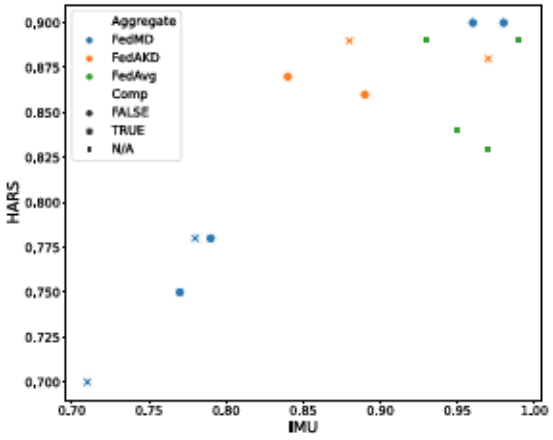

Figure 4.12: Test accuracy of the federated learning methods on HARS and IMU datasets. Points are styled according to whether soft label compression is applied.

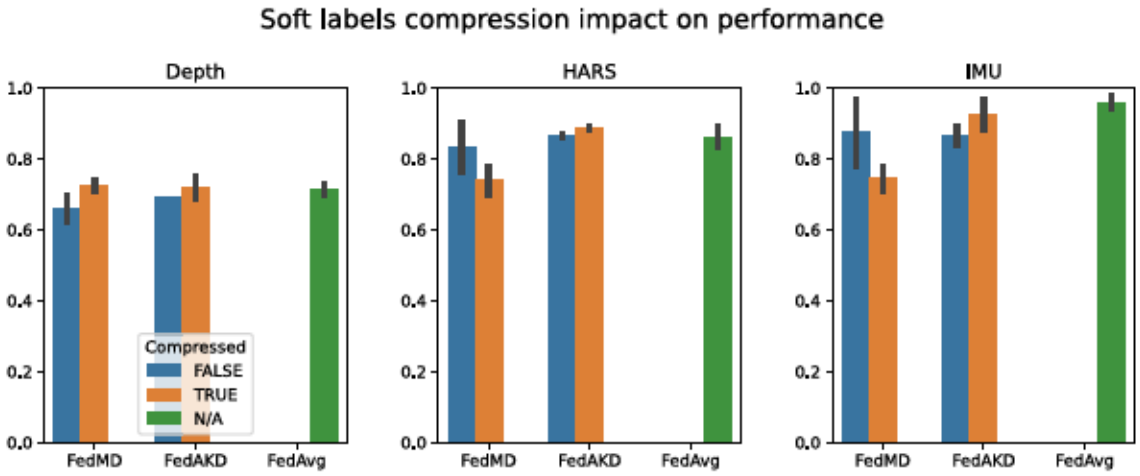

Figure 4.13: Test accuracy of the federated learning methods on Depth, HARS, and IMU datasets. Bars are grouped based on whether soft label compression is applied. For FedAvg, a N/A label is given since it does not use soft labels.

4.7 Conclusion

In this chapter, we propose to use compression on top of Knowledge Distillation to save communication bandwidth in federated learning. Instead of calculating soft labels and sending them as floats, we propose to first normalize and convert the soft labels to unsigned int resulting in 8 times less size. The proposed algorithm: Compressed Federated Learning with Augmented Knowledge Distillation (CFedAKD) modifies the FedAKD algorithm presented in the previous chapter. We evaluated the performance of CFedAKD on four Human Activity Recognition (HAR) datasets with different modalities against model-based and model-agnostic FL methods and found that the communication overhead of CFedAKD is significantly less than other FL methods, especially model-based algorithms while maintaining accuracy.

Chapter 5

Privacy in Federated Learning

5.1 Introduction

In this thesis, we study Federated Learning (FL) and the challenges associated with the development and deployment of FL algorithms such as communication overhead; statistical and system heterogeneity; and privacy concerns. We present knowledge distillation-based FL algorithms such as FedMD/FedAKD as an alternative paradigm to standard federated learning (fedAvg). While in FedAvg clients send locally trained model weights and receive aggregated model weights, clients in KD-based FL share local soft labels calculated by independently designed models trained on private local datasets. The server in FedMD/FedAKD aggregates local soft labels and broadcasts the global soft labels to clients to train their local models for a few epochs called knowledge distillation training using distance loss functions like Kullback-Leibler (KL) or Mean Square Error (MSE). In the previous chapters, we presented two versions of Knowledge Distillation-based Federated Learning: 1) FedAKD and 2) CFedAKD where both algorithms employ a shared public dataset to calculate soft labels which are then sent to the server, aggregated, and the consensus soft labels are broadcasted back. In FedAKD, we apply an augmentation technique called Mixup augmentation on the public dataset with server-controlled parameters to generate a new version of the public dataset at each global round and thus help distill knowledge more efficiently. We discuss the performance and communication characteristics of the proposed algorithm. In the previous chapter, we take another step in our discussion of communication-efficient FL algorithms and propose CFedAKD, which employs compression techniques to cast the unit of soft labels from float (8 Byte) to Unsigned Int (1 Byte).

In this chapter, we discuss privacy concerns associated with training deep learning models in the central and the FL contexts. We start by laying out some of the recent work that presents privacy attacks and counter-defense mechanisms. Differential privacy is a privacy analysis framework that protects against numerous attacks by injecting calibrated

noise. We discuss the proposed mechanisms to construct DP training pipelines for deep learning models. The Gaussian mechanism to apply noise for DP is introduced as well as relevant DP properties such as the composition property and the post-processing immunity property. The Differentially Private Stochastic Gradient Descent (DP-SGD) algorithm [83] to train deep learning models with differential privacy using a noise-adding mechanism is presented in figure 5.3.

We then focus more on the difference between applying DP in the contest of model-based FL and KD-based FL algorithms. The experiments performed in this thesis to train DP models under different FL algorithms and privacy protection levels are detailed, and the implementation details are described. Finally, we present the results obtained from running the experiments which include reporting and plotting test accuracy of training different FL baselines with different noise scales resembling privacy protection levels.

5.2 Background

There is a need for Deep Learning to ensure the privacy of training data because deep learning models learn by memorizing patterns and data [84]. Differential Privacy (DP) [85] has become the de facto framework for privacy guarantee and privacy analysis in Deep Learning and FL to protect against data leaks and privacy attacks. In particular, differential privacy has been shown to effectively protect against Membership Inference Attacks (MIA) in which an adversary is interested to know whether a given sample was used to train a neural network [86]. Recent works have also demonstrated that by exploiting implicit memorization, sensitive data can be revealed not only from the model parameters but also from the model output. For example, Fredrikson et al [87] retrieved training samples (individual faces) by exploiting the output probabilities of a computer-vision classifier. The authors in [88] evaluate the efficiency of DP to protect against membership inference attacks as well as backdoor attacks, in which rogue clients inject a backdoor task in the client. It was found that applying Local Differential Privacy (LDP) can protect against backdoor attacks (e.g., data and label poisoning attacks).

Because the ML training process is an iterative process, with each iteration a batch of samples is fed into the model, gradients are calculated, and model weights are updated, the risk of memorizing data is higher for each iteration.

Differential Privacy introduces concepts to estimate the privacy loss as training progresses and employs noise addition mechanisms that link noise magnitude to the estimated privacy loss to achieve a given Privacy level. In FL settings, DP properties like composition are necessary to analyze the privacy risk imposed by training distributed functions on sensitive distribute datasets. According to the assumed threat model, DP can be realized in different ways. For example, in local differential privacy (LDP), the server is assumed to

be untrusted. While in central DP (CDP, the server is trusted and responsible for adding noise to the aggregated clients' updates. While implementing LDP and analyzing the global guarantee might be harder than in CDP, LDP-trained models achieve better privacy-utility tradeoffs which is always the biggest concern in applying Differential Privacy.

5.3 Differential Privacy

Differential Privacy (DP) [85] is a mathematical framework for quantifying and managing the privacy risks associated with the release or sharing of statistical data. It provides strong privacy guarantees by ensuring that the addition or removal of a single data point in a dataset does not significantly impact the output of a statistical analysis.

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

$$\Pr[M(D) \in S] \leq e^{\epsilon} \Pr[M(D') \in S] + \delta \tag{5.1}$$

where ϵ is the privacy budget controlling the amount of privacy leakage and δ is a small positive value that controls the probability of deviation from the guarantee. The smaller the value of ϵ and δ, the stronger the privacy guarantee.

A commonly used method for applying differential privacy via injecting noise is the Gaussian Mechanism (GM). The GM is designed to approximate a function s while protecting the privacy of the samples in D by adding noise that is calibrated to the function's sensitivity Δs of s given by

$$\Delta s = \max_{D,D'} ||s(D) - s(D')|| \tag{5.2}$$

To satisfy (ϵ, δ)-differential privacy, properly calibrated Gaussian noise $n \sim N(0, \sigma^2)$ can be added to a real-valued function s by setting the noise standard deviation $\sigma \geq c\Delta s/\epsilon$ and the constant $c \geq \sqrt{2\ln(1.25/\delta)}$ for $\epsilon \in (0, 1)$ [85].

5.3.1 DP Properties

In this section, we discuss two key DP properties: composability and post-processing immunity. The composition properties allow privacy guarantees to be maintained even when combining multiple mechanisms or queries. We discuss two important composition scenarios: sequential and parallel compositions.

- Sequential Composition. Sequential composition refers to the case where we have k mechanisms, M, each providing (ϵ_i, δ_i)-differential privacy, and we want to use the outputs from the first mechanism as inputs to the second, and so on, without sacrificing privacy too much. If, for $i \in 1, 2, \ldots, k$, we let $M_i(d)$ be an (ϵ_i, δ_i)-differentially

private mechanism executed on database d, then the function composition F of these mechanisms, $F = (M_1 \circ M_2 \circ \ldots \circ M_k)$, is $(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i)$-differentially private.

- Parallel Composition. The serial composition discussed earlier assumes that the outputs are correlated, resulting in a more pessimistic total privacy budget value ϵ and a higher probability of failure δ. Parallel composition considers the situation where we have a single database d partitioned into k disjoint subsets d_i. If we compute mechanisms $M_1, M_2, \ldots, M_k$ on these disjoint subsets (i.e., $M_i(d_i)$), with privacy guarantees $\epsilon_1, \epsilon_2, \ldots, \epsilon_k$ and $\delta_1, \delta_2, \ldots, \delta_k$ respectively, then any function composition F of these mechanisms, $F = (M_1 \circ M_2 \circ \ldots \circ M_k)$, is $(\max_{i=1}^{k} \epsilon_i, \max_{i=1}^{k} \delta_i)$-differentially private.

- Immunity to Post-Processing. The post-processing property of DP states that the output a (ϵ_i, δ_i)-DP mechanism is also (ϵ_i, δ_i)-DP private.

  We denote $M : N^{|X|} \to R$ as a randomized algorithm that satisfies (ϵ, δ)-differentially private. Additionally, $f : R \to R_0$ represents another function. Then, $f \circ M : N^{|X|} \to R_0$ also satisfies (ϵ, δ)-DP.

  Proof. For a deterministic function $f : R \to R_0$.

  For any neighboring databases x, y with $||x - y||_1 \leq 1$, and event $S \subseteq R_0$. Let $T = \{r \in R : f(r) \in S\}$. We then have:

$$\begin{aligned} \Pr[f(M(x)) \in S] &= \Pr[M(x) \in T] \\ &\leq \exp(\epsilon) \Pr[M(y) \in T] + \delta \\ &= \exp(\epsilon) \Pr[f(M(y)) \in S] + \delta \end{aligned}$$

  □

  This result demonstrates that the probability $\Pr[f(M(x)) \in S]$ is bounded by

$$\exp(\epsilon) \Pr[f(M(y)) \in S] + \delta \tag{5.3}$$

  , showing that applying the deterministic function f to the output of the (ϵ, δ)-differentially private algorithm M maintains the (ϵ, δ)-differential privacy guarantees. The result follows for randomized mapping as any randomized mapping can be decomposed into a convex combination of deterministic functions.

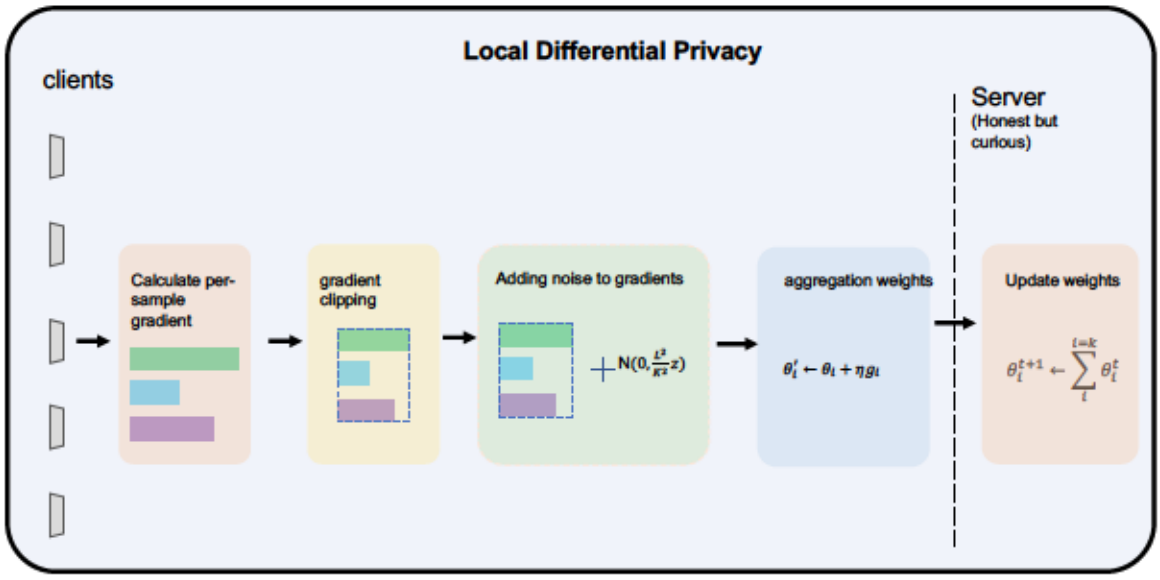

Figure 5.1: Local Differential Privacy.

5.4 Federated Learning Threat Model

In the context of FL, there are different threat models each of them having different potential privacy risks associated with the learning process [12]. Two of the practical and widely considered FL threat models are :

1. Honest-but-curious server: In this scenario, the central server follows the prescribed FL protocol but may attempt to infer information about clients' data from the received model updates.

2. Malicious server: The central server deviates from the prescribed protocol and actively attempts to compromise the privacy of the clients' data.

Based on the threat model adopted, Differential Privacy can be applied by different actors to guarantee privacy in the context FL process.

In case the server is assumed to be a trusted entity, Central Differential Privacy [88] could be applied where the server is responsible for adding noise to the aggregated model updates, ensuring that the global model's output does not reveal information about individual clients' data.

In the Malicious server threat model, privacy protection is applied at the client level. Local Differential Privacy (LDP) is applied where each client adds noise to their local updates before sharing them with the server, making it difficult for the server or other clients to infer information about their local data. In LDP [89], each client controls the privacy-utility trade-off according to its own privacy requirements, without relying on a centralized

authority. For example, a client that has more sensitive data may choose to add more noise to its data to ensure higher privacy, while a client with less sensitive data may choose to add less noise to preserve more utility. This adaptability makes LDP particularly useful in scenarios where data owners have different privacy needs, such as in FL or in healthcare applications where different patients may have different levels of privacy sensitivity.

Algorithm 3 Local Differential Privacy using PD-SGD in FedAvg

function MAIN
  Input: Client k owns a local dataset: D_k, Initial global model weights: θ_0, Number of communication rounds: R, Number of epochs: E, Total number of participating clients: N_k, Fraction of clients participating each round: q, DP parameters ϵ and δ, Gradient Clipping norm S
  Output: Collaboratively trained global model θ^R
  Initialize model θ_0
  $W \leftarrow E \cdot R$ ▷ Number of steps
  $z \leftarrow OpacusPrivacyEngine(S, W)$ ▷ noise scale
  $\sigma \leftarrow z \cdot \frac{S}{q}$
  for round $r = 1, 2, \ldots, R$ **do**
    $K^r \leftarrow$ randomly select K participants with probability q
    for each participant $k \in K^r$ **do**
      $\theta_k^r \leftarrow$ DP-SGD(S, σ) ▷ Clients train in parallel
    end for
    $\theta^r \leftarrow \sum_{k \in K^r} \frac{n_k}{n} \theta_k^r$ ▷ n_k is the size of k's dataset
  end for
  return θ^R
end function

function DP-SGD(S, σ)
  Initialize model f with weights θ_0
  for epoch i from 1 to E **do**
    for $(x, y) \in$ in each batch from dataset D_k **do**
      $g_i = \nabla_\theta L(\theta_i, (x, y))$
    end for
    $g_{DP} = \frac{1}{|D_k|} \sum_{i \in \text{batch}} g_i \min\left(1, \frac{S}{\|g_i\|_2}\right) + \mathcal{N}(0, \sigma^2)$
    $\theta_{i+1} = \theta_i - \eta(g_{DP})$
  end for
  return θ_E
end function

Figure 5.1 represents an overview of the Local Differential Privacy algorithm shown in algorithm 3. In this setting, noise is applied at the client's side since we assume an honest-but-curious server, i.e. we don't trust the curator to add the noise. Each client calculates per-example gradients and clips them using a clipping norm specified by the user. After clipping, we use the Gaussian Mechanism to add noise with a standard deviation σ calibrated by the chosen clipping norm S.

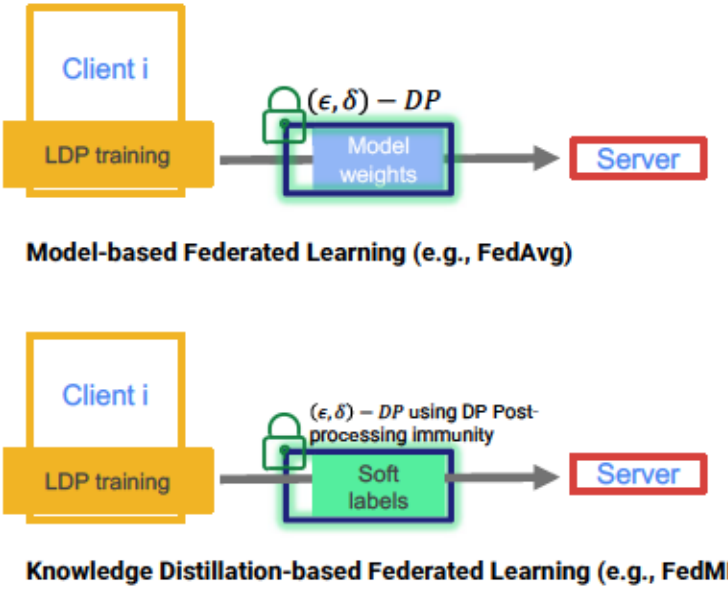

Figure 5.2: The privacy guarantee applied by a client with DP-SGD is preserved by the soft labels generated by that client.

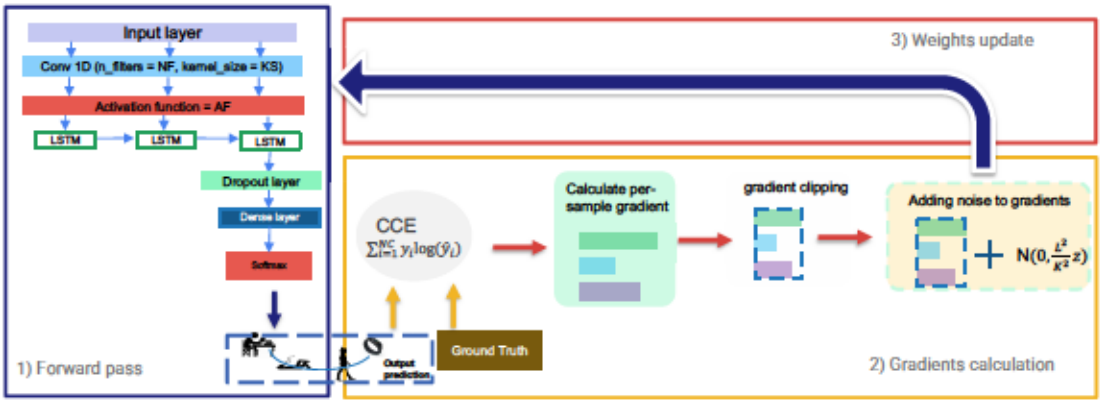

Figure 5.3: Differential Private Stochastic Gradient Descent (DP-SGD).

5.5 Knowledge Distillation Federated Learning Differential Privacy

In KD-based FL algorithms [13, 14], instead of weights, clients share knowledge in the form of soft labels. Adversaries in this case become interested in extracting private data from shared model predictions. As the shared soft labels are used for knowledge distillation training of clients, it is hard to maintain an acceptable utility-privacy balance with noise injection mechanisms that add noise to these predictions. An alternative is to use the DP property of postprocessing immunity which was discussed earlier. The privacy guarantee which is attained by LDP where noise is injected into the gradients by each client during local training is inherited by any other algorithm applied to the output of these LDP-trained models, as shown in figure 5.2.

5.6 Differential Privacy Implementation

We implemented Local Differential Privacy (LDP) where each client applies DP to its calculated gradients. LDP has many advantages over Central Differential Privacy (CDP) like adapting the amount of noise added at each client to achieve a customized protection level per client depending on the sensitivity of his data and the desired utility-privacy balance. Furthermore, LDP is more realistic as it assumes an honest-but-curious server threat model, whereas CDP assumes a trusted curator.

We use the Opacus library [90] which provides a Differential Privacy training pipeline for Pytorch models. We investigate how changing the privacy protection level affects the performance of FL algorithms. The considered FL algorithms are available in section 4.5.2. These baseline FL methods are evaluated on the Human Activity Recognition (HAR) datasets which are presented in the section 4.5.1. Opacus provides functions to calculate the amount of noise that needs to be applied over a specific number of epochs to achieve a certain privacy protection level (ϵ, δ)-differential privacy. We try three ϵ values: inf (no Differential Privacy), 20, 5. Where 5 is the strongest protection level and inf means normal training without DP. We set δ to 1e-4.

5.7 Experiments and Results

In this section, the accuracy obtained by the FL algorithms FedAKD [14], FedMD [13], and FedAvg [39] when trained under different privacy protection levels. We consider three Differential Privacy (DP) protection levels: No DP, (20, 1e-5)-DP, and (5, 1e-5)-DP. We utilized the same four HAR datasets as in the previous chapter.

Figure 5.4 shows three bar plots. From left to right, the accuracies obtained under the three privacy protection levels: No DP, (20, 1e-5)-DP, and (5, 1e-5)-DP are shown, respectively. For the Depth dataset in left most bar plot, we can see Knowledge Distillation-based FL methods FedAKD/FedMD outperform FedAvg by over than 15 percentage points. Applying differential privacy resulted in significant accuracy loss in all algorithms until they reach about 20 % accuracy. For the other two datasets, HARS and IMU, balancing utility-privacy is easier than in the Depth dataset, under both protection levels ($\epsilon = 20$ and $\epsilon = 5$). The accuracy was less severe for HARS and IMU datasets than the accuracy drop observed in the Depth dataset for all FL algorithms. One of the reasons for this is the large dimension size and complexity of the Depth dataset relative to the two other datasets.

The figures 5.6 and 5.5 show the test accuracy obtained by model-agnostic (FedMD/FedAKD) and model-based (FedAvg) FL algorithms on Three HAR datasets. The first figure 5.6 reports the accuracy while applying LDP. On the other hand, 5.5 shows the accuracy without applying DP. The first bar plot of the two figures from the left depicts the performance on the image dataset: Depth. We can observe that the drop in accuracy of FedAvg was significant compared to the two other model-agnostic FL methods. However, applying DP resulted in an accuracy drop across datasets with varying severity due to the differences in data complexity, model architecture, training parameters, etc.

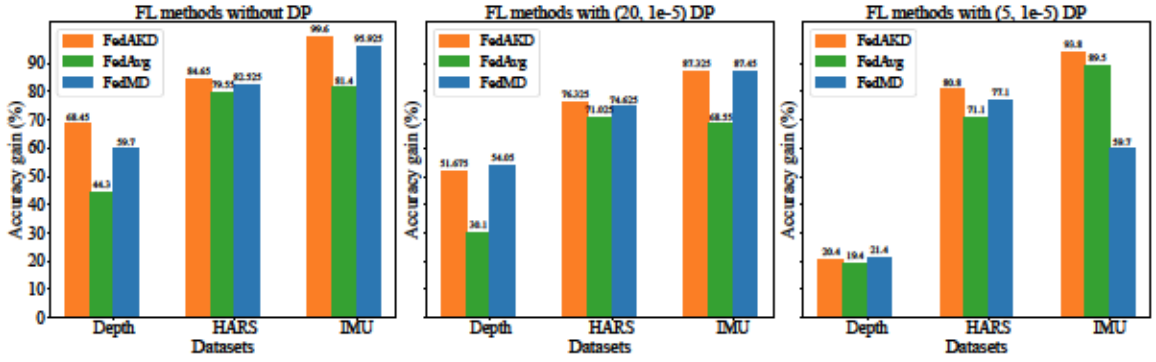

Figure 5.4: Test accuracy of the FL methods on Depth, HARS, and IMU datasets with different differential privacy protection levels.

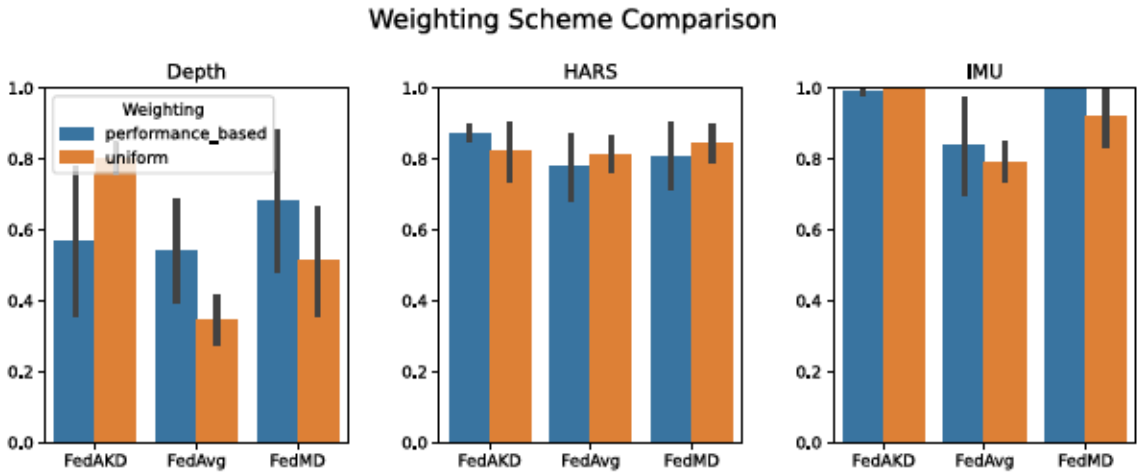

Figure 5.5: Test accuracy of the FL methods on Depth, HARS, and IMU datasets. Bars are grouped based on whether the weighting scheme employed: uniform vs Performance-based weighting while not applying DP.

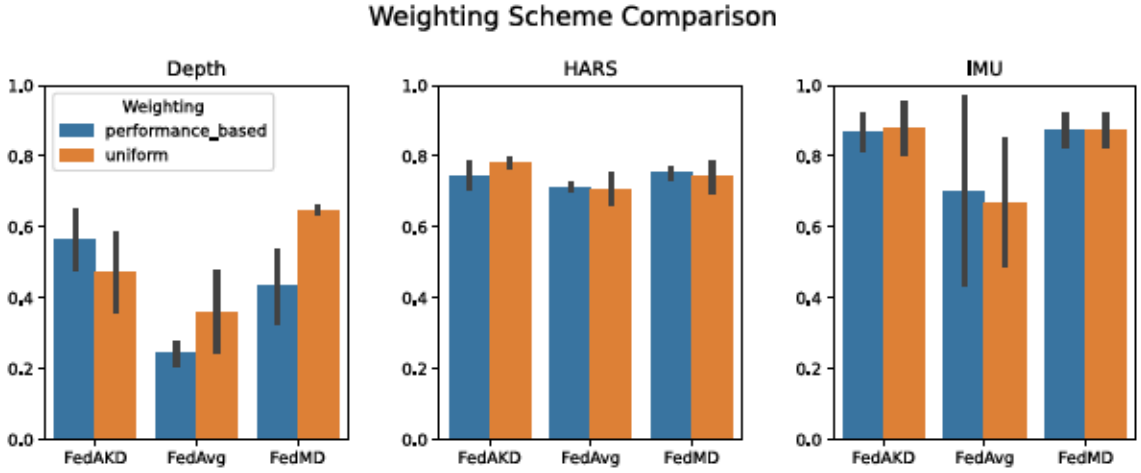

Figure 5.6: Test accuracy of the FL methods on Depth, HARS, and IMU datasets. Bars are grouped based on whether the weighting scheme employed: uniform vs Performance-based weighting while applying DP.

5.8 Conclusion

In this chapter, we first provide a brief background on privacy challenges and solutions in the federated setting. Then, we focused on Differential Privacy as a privacy analysis tool by providing a formal definition for DP and discussing some of its important properties that make DP appealing to work with in machine learning. After that, we studied Local Differential Privacy (LDP) and discussed the threat model in which LDP is preferred

over Central Differential Privacy (CDP). Privacy in Knowledge Distillation-based federated learning is also discussed in light of DP. To implement DP, we employ the Opacus python library and consider two DP protection levels. The simulations were run considering model-based and model-agnostic FL algorithms on the datasets listed in 4.5.1. Results show that for the same protection level, the accuracy obtained by FL algorithms on datasets with higher input dimensionality/complexity is more severely impacted than on datasets with lower complexity.

Chapter 6

Conclusion

In the previous two chapters of this thesis, we studied Federated Learning (FL) as a distributed learning paradigm and focused on addressing three key limitations of FL. These challenges include system heterogeneity, communication overhead, and privacy concerns. The first challenge is concerned with the heterogeneous nature of the participants of FL which stems from their different computational capabilities and requirements. The second challenge studies the communication cost incurred by different FL approaches during uploading local updates and broadcasting global updates to them. The third challenge focuses on protecting users' privacy from attacks such as the Membership Inference Attack (MIA) in deep learning and FL.

We introduce a novel FL approach based on the Knowledge Distillation (KD) technique and Mixup augmentation. The proposed FL algorithm, called Federated Learning via Augmented Knowledge Distillation (FedAKD), employs knowledge distillation for the collaborative training of heterogeneous deep learning models. The algorithm was evaluated on human activity recognition datasets HARS and HARB, with results showing significant communication efficiency and accuracy gains compared to other algorithms like FedAvg and FedMD. This improved performance is due to FedAKD's use of augmentation for generating new variants of the public dataset in each communication round, facilitating more effective knowledge distillation.

To support the deployment of FL algorithms on low-bandwidth IoT networks. A drone-aided LoRa (DORA) network is proposed to perform lightweight FL across a network of scattered houses. Each house resembles an FL client with the local dataset. The communication is performed using a LoRa low-bandwidth link that can't support model-based FL where model weights are sent each round from clients (houses) to the server (the drone) and vice versa. We propose a two-fold solution. First, we train a Self-Organizing Map (SOM) to optimize the drone path to minimize distance and save energy. Second, the traffic model for KD-based FL algorithms (e.g., FedMD, FedAKD) and the traffic model of model-based FL

methods (e.g., FedAvg) are presented and we conduct experiments to show the significant communication advantage of KD-based Fl algorithms over the model-based counterparts. The performance achieved by FedAKD was also comparable to that obtained by FedAvg. KD-based FL algorithm leverages a shared public dataset to distill knowledge learned by clients on the local private datasets.

Moreover, we proposed Compressed Federated Learning via Augmented Knowledge Distillation (FedAKD) CFedAKD, a compressed version of FedAKD, which reduces communication overhead by normalizing and converting soft labels into unsigned integers. Our evaluation demonstrated that CFedAKD maintains accuracy while significantly decreasing communication overhead, especially when compared to model-based algorithms.

Privacy challenges in FL were addressed using Differential Privacy (DP) which is the De facto statistical framework for privacy analysis. We explored the implementation of DP in the FL setting by applying Local Differential Privacy. We also present the key DP properties like sequential composition and post-processing immunity. The latter property also extends the DP protection level achieved by LDP to soft labels in knowledge distillation-based FL. Finally, we conduct simulations to assess how applying different protection levels impacts the accuracy obtained on datasets of various modalities using KD-based FL algorithms vs model-based Fl algorithms.

6.1 Future Work

There are several directions for future work, including:

1. Enhancing the selection of the public dataset D_p by choosing a dataset with a similar distribution to the local datasets, rather than using the training set of the respective dataset. This approach would better preserve users' privacy and ensure the public dataset has a distribution akin to local data.

2. Investigating the class-level performance achieved by the KD-based FL paradigm to better understand the characteristics of KD and the potential improvements of its efficiency on a class level.

In this work, we focus on the Human Activity Recognition (HAR) domain and evaluate FedAKD only on HAR datasets. We aim to evaluate FedAKD on other data sources and modalities to validate its versatility and effectiveness across a wider range of applications.

Overall, this work contributes to the development of FL algorithms that achieve high accuracy and communication efficiency while addressing privacy concerns. As more data is generated and shared across devices, these advances become increasingly vital in realizing the full potential of FL.

Appendix A

List of Abbreviations

FL	Federated Learning
HAR	Human Activity Recognition
i.i.d	Independent and identically distributed
KD	Knowledge Distillation
E	The number of local epochs
R	The number of global communication rounds
β^r	An integer used to seed the permutation of D_p at round r
α^r	constant $\in [0, 1]$ used to calculate D^r_{Aug}
C	a set of clients/devices participating in FL
C_i	The i^{th} client
f_i	Independently designed deep model of the i^{th} client
f_i^*	f_i minus the last softmax activation layer
D_i	The i^{th} local dataset of the i^{th} client
$\{X^{[j]}, Y^{[j]}\}$	The j^{th} sample pair in a local dataset
D_t	test dataset (shared)
D_p	the public dataset (shared)
$\{X_P^{[j]}\}$	The j^{th} sample in D_p
D^r_d	D_p permuted using the seed β^r
D^r_{Aug}	augmented public dataset. D_p and D^r_d weighted by α^r.
P^r_i	is the accuracy of f_i on D_t at global round r
S^r_i	soft labels of i^{th} client on D^r_{Aug} at global round r
S^r	soft labels aggregated by the server at global round r

Appendix B

Examiner's Comments and Responses

Examiner Comment: One key objective of the work seems to be giving the clients better control. The issue, in this case, is that if the clients are given better control of the mechanism, there are chances of collusion among the rogue clients. Even if a single one's input is ignored, there could be a team/group of clients and those clients can collude to influence the data or training mechanism or the overall outcome. How are the clients authenticated properly to participate in the system? This should be clarified. Again, are not the clients overloaded with data and calculation tasks? Is it a good trade-off for this case compared to the other traditional methods and models? Why? Some cogent rationale is needed. Memory and computational requirements may be analyzed a bit more, if possible. If that adds more work, at least you can address the issues with some good discussion.

Student Response:

Thank you for your question.

First, I would like to clarify that KD-based FL algorithms like the one proposed in this thesis (FedAKD) have two main characteristics: They give clients control over the design of their local models, and clients share soft labels predicted on a public dataset instead of sharing model weights.

It is worth mentioning that Differential Privacy (DP) addresses privacy attacks and not attacks that aim to harm the performance of the model. Therefore, even if several clients colluded (the adversaries), they would try to exploit the local updates released by some client (the victim), which according to the Local Differential Privacy (LDP) algorithm, will be protected via calibrated noise injection. Therefore, Independently designing the local model doesn't give colluding rogue clients an advantage to extract sensitive data from local updates. Furthermore, since local updates are in the form of locally trained model weights in the case of FedAvg and soft labels in the case of FedMD/FedAKD, KD-based

FL algorithms are less vulnerable to privacy attacks than standard Fl algorithms. While there are studies that show that DP has a limited advantage when it comes to protecting against attacks that target model performance [88], DP is known to be effective against inference attacks and will be less effective against data poisoning attacks. The reason is that DP ensures privacy by injecting noise which generally harms utility but is useful in hiding the individual contribution of each training sample. While injecting noise to achieve a certain $(\epsilon, \delta) - DP$ will not prevent corrupt weighted from malicious users from harming the overall performance when included in the aggregation process, one direction that is worth exploring in this regard is to treat potentially malicious users as valuable assets and inject more noise into their model to mitigate the negative impact that these corrupt weights have on performance if aggregated as they were received.

Examiner Comment 2: Your work focuses on Knowledge Distillation (KD). What are the challenges that this approach addresses? and what is the motivation for integrating this technology in the FL context?

Student Response: KD was originally developed for central training and knowledge transfer from a trained model (the teacher) to another to-be-trained model (the student). KD defines the internal representation that a model learns when trained for a specific task as knowledge and provides a mechanism to transfer this knowledge from one model to another. If the KD process is efficient, the student model can be much smaller than the teacher model and still achieve comparable performance to the teacher [?]. In the FL context, KD is used to transfer local model representations (knowledge) usually by incorporating a public dataset [14,62] making the KD-based FL methods model-agnostic. This gives KD-based FL two main advantages over model-based FL algorithms. First, clients under a KD-based FL algorithm can design their local model unlike model-based methods (e.g., FedAvg) on which a server-controlled architecture is imposed. Second, the communication cost of KD-based FL methods which communicate soft labels [62] is generally less than that of model-based FL algorithms which communicate local/global model weights [39, 78].

Bibliography

[1] "About one-in-five Americans use a smart watch or fitness tracker." [Online]. Available: https://www.pewresearch.org/fact-tank/2020/01/09/about-one-in-five-americans-use-a-smart-watch-or-fitness-tracker/

[2] C. Jobanputra, J. Bavishi, and N. Doshi, "Human activity recognition: A survey," *Procedia Computer Science*, vol. 155, pp. 698–703.

[3] G. Bhat, R. Deb, V. V. Chaurasia, H. Shill, and U. Y. Ogras, "Online human activity recognition using low-power wearable devices," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, pp. 1–8.

[4] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207.

[5] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819.

[6] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407. [Online]. Available: https://dx.doi.org/10.1561/0400000042

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60.

[8] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep Gradient Compression: Reducing the communication bandwidth for distributed training," in *The International Conference on Learning Representations*, 2018.

[9] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: A Semi-Asynchronous Protocol for Fast Federated Learning With Low Overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668.

[10] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450.

[11] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting Gradients - How easy is it to break privacy in federated learning?" vol. 33. Curran Associates, Inc., pp. 16 937–16 947. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html

[12] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. Vincent Poor, "Federated Learning With Differential Privacy: Algorithms and Performance Analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469.

[13] D. Li and J. Wang. FedMD: Heterogenous Federated Learning via Model Distillation. [Online]. Available: http://arxiv.org/abs/1910.03581

[14] G. Gad and Z. Fadlullah, "Federated Learning via Augmented Knowledge Distillation for Heterogenous Deep Human Activity Recognition Systems," *Sensors*, vol. 23, no. 1. [Online]. Available: https://www.mdpi.com/1424-8220/23/1/6

[15] D. L.-P. Y. N. D. Hongyi Zhang, Moustapha Cisse, "Mixup: Beyond empirical risk minimization," *International Conference on Learning Representations*. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[16] F. Demrozi, G. Pravadelli, A. Bihorac, and P. Rashidi, "Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey," *IEEE access : practical innovations, open solutions*, vol. 8, pp. 210 816–210 836.

[17] O. D. Lara, A. J. Pérez, M. A. Labrador, and J. D. Posada, "Centinela: A human activity recognition system based on acceleration and vital sign data," *Pervasive and mobile computing*, vol. 8, no. 5, pp. 717–729.

[18] D. Bhattacharya, D. Sharma, W. Kim, M. F. Ijaz, and P. K. Singh, "Ensem-HAR: An ensemble deep learning model for smartphone sensor-based human activity recognition for measurement of elderly health monitoring," *Biosensors*, vol. 12, no. 6, p. 393.

[19] S. Love, *Understanding mobile human-computer interaction*. Elsevier, 2005.

[20] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[21] Z. Lv, F. Poiesi, Q. Dong, J. Lloret, and H. Song, "Deep learning for intelligent human–computer interaction," *Applied Sciences*, vol. 12, no. 22, p. 11457, 2022.

[22] D. A. Wood, S. Kafiabadi, A. Al Busaidi, E. L. Guilhem, J. Lynch, M. K. Townend, A. Montvila, M. Kiik, J. Siddiqui, N. Gadapa *et al.*, "Deep learning to automate the labelling of head MRI datasets for computer vision applications," *European Radiology*, vol. 32, no. 1, pp. 725–736.

[23] Y. Chen, K. Zhong, J. Zhang, Q. Sun, and X. Zhao, "Lstm networks for mobile human activity recognition," in *2016 International conference on artificial intelligence: technologies and applications*. Atlantis Press, 2016, pp. 50–53.

[24] W. Elmenreich, "An introduction to sensor fusion," *Vienna University of Technology, Austria*, vol. 502, pp. 1–28.

[25] Y. Zhang, L. Wang, H. Chen, A. Tian, S. Zhou, and Y. Guo, "IF-ConvTransformer: A framework for human activity recognition using IMU fusion and ConvTransformer," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 6, no. 2, pp. 1–26.

[26] D. R. Beddiar, B. Nini, M. Sabokrou, and A. Hadid, "Vision-based human activity recognition: a survey," *Multimedia Tools and Applications*, vol. 79, pp. 30 509–30 555, 2020.

[27] S. Kaghyan and H. Sarukhanyan, "Activity recognition using k-nearest neighbor algorithm on smartphone with tri-axial accelerometer," *International Journal of Informatics Models and Analysis (IJIMA), ITHEA International Scientific Society, Bulgaria*, vol. 1, pp. 146–156.

[28] L. Xu, W. Yang, Y. Cao, and Q. Li, "Human activity recognition based on random forests," in *2017 13th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD)*. IEEE, 2017, pp. 548–553.

[29] D.-X. Zhou, "Universality of deep convolutional neural networks," *Applied and computational harmonic analysis*, vol. 48, no. 2, pp. 787–794.

[30] A. M. Schäfer and H. G. Zimmermann, "Recurrent neural networks are universal approximators," in *International Conference on Artificial Neural Networks*. Springer, pp. 632–640.

[31] J. Talukdar and B. Mehta, "Human action recognition system using good features and multilayer perceptron network," in *2017 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2017, pp. 0317–0323.

[32] A. Murad and J.-Y. Pyun, "Deep recurrent neural networks for human activity recognition," *Sensors*, vol. 17, no. 11, p. 2556, 2017.

[33] M. Zeng, H. Gao, T. Yu, O. J. Mengshoel, H. Langseth, I. Lane, and X. Liu, "Understanding and improving recurrent networks for human activity recognition by continuous attention," in *Proceedings of the 2018 ACM International Symposium on Wearable Computers*, pp. 56–63.

[34] F. Hernández, L. F. Suárez, J. Villamizar, and M. Altuve, "Human activity recognition on smartphones using a bidirectional lstm network," in *2019 XXII symposium on image, signal processing and artificial vision (STSIVA)*. IEEE, 2019, pp. 1–5.

[35] T. Kim, J. Oh, N. Y. Kim, S. Cho, and S.-Y. Yun, "Comparing Kullback-Leibler Divergence and Mean Squared Error Loss in Knowledge Distillation," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, pp. 2628–2635. [Online]. Available: https://www.ijcai.org/proceedings/2021/362

[36] A. Afonin and S. P. Karimireddy, "Towards model agnostic federated learning using knowledge distillation," *arXiv preprint arXiv:2110.15210*, 2021.

[37] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10 713–10 722.

[38] X. Ouyang, Z. Xie, J. Zhou, G. Xing, and J. Huang, "ClusterFL: A clustering-based federated learning system for human activity recognition," *ACM Transactions on Sensor Networks (TOSN)*.

[39] B. McMahan, E. Moore, D. Ramage, S. Hampson, and p. u. family=Arcas, given=Blaise Aguera, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, pp. 1273–1282.

[40] Z. Xiao, X. Xu, H. Xing, F. Song, X. Wang, and B. Zhao, "A federated learning system with enhanced feature extraction for human activity recognition," *Knowledge-Based Systems*, vol. 229, p. 107338.

[41] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, "Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–40.

[42] K. Bjerge, H. M. Mann, and T. T. Høye, "Real-time insect tracking and monitoring with computer vision and deep learning," *Remote Sensing in Ecology and Conservation*, vol. 8, no. 3, pp. 315–327.

[43] L. K. Ramasamy, F. Khan, M. Shah, B. V. V. S. Prasad, C. Iwendi, and C. Biamba, "Secure smart wearable computing through artificial intelligence-enabled internet of things and cyber-physical systems for health monitoring," *Sensors*, vol. 22, no. 3, p. 1076.

[44] A. Bouguettaya, H. Zarzour, A. M. Taberkit, and A. Kechida, "A review on early wildfire detection from unmanned aerial vehicles using deep learning-based computer vision algorithms," *Signal Processing*, vol. 190, p. 108309.

[45] I. Lauriola, A. Lavelli, and F. Aiolli, "An introduction to deep learning in natural language processing: Models, techniques, and tools," *Neurocomputing*, vol. 470, pp. 443–456.

[46] F. Kulsoom, S. Narejo, Z. Mehmood, H. N. Chaudhry, A. K. Bashir *et al.*, "A review of machine learning-based human activity recognition for diverse applications," *Neural Computing and Applications*, pp. 1–36.

[47] S. Zhang, Y. Li, S. Zhang, F. Shahabi, S. Xia, Y. Deng, and N. Alshurafa, "Deep learning in human activity recognition with wearable sensors: A review on advances," *Sensors*, vol. 22, no. 4, p. 1476.

[48] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE communications surveys & tutorials*, vol. 15, no. 3, pp. 1192–1209.

[49] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, pp. 1–33.

[50] Y. Li and L. Wang, "Human activity recognition based on residual network and BiLSTM," *Sensors*, vol. 22, no. 2, p. 635.

[51] S. Chung, J. Lim, K. J. Noh, G. G. Kim, and H. T. Jeong, "Sensor positioning and data acquisition for activity recognition using deep learning," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, pp. 154–159.

[52] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *International Workshop on Ambient Assisted Living*. Springer, pp. 216–223.

[53] L. Sun, D. Zhang, B. Li, B. Guo, and S. Li, "Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations," in *International Conference on Ubiquitous Intelligence and Computing*. Springer, pp. 548–562.

[54] S. Díaz, J. B. Stephenson, and M. A. Labrador, "Use of wearable sensor technology in gait, balance, and range of motion analysis," *Applied Sciences*, vol. 10, no. 1, p. 234.

[55] O. Dehzangi, M. Taherisadr, and R. ChangalVala, "Imu-based gait recognition using convolutional neural networks and multi-sensor fusion," *Sensors*, vol. 17, no. 12, p. 2735, 2017.

[56] D. Laidig, T. Schauer, and T. Seel, "Exploiting kinematic constraints to compensate magnetic disturbances when calculating joint angles of approximate hinge joints from orientation estimates of inertial sensors," in *2017 International Conference on Rehabilitation Robotics (ICORR)*. IEEE, pp. 971–976.

[57] C. Zampieri, A. Salarian, P. Carlson-Kuhta, K. Aminian, J. G. Nutt, and F. B. Horak, "The instrumented timed up and go test: Potential outcome measure for disease modifying therapies in Parkinson's disease," *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 81, no. 2, pp. 171–176.

[58] S. Mekruksavanich and A. Jitpattanakul, "CNN-Based deep learning network for human activity recognition during physical exercise from accelerometer and photoplethysmographic sensors," in *Computer Networks, Big Data and IoT*. Springer, pp. 531–542.

[59] K. Doshi and Y. Yilmaz, "Federated learning-based driver activity recognition for edge devices," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3338–3346.

[60] K. Sozinov, V. Vlassov, and S. Girdzijauskas, "Human activity recognition using federated learning," in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. IEEE, pp. 1103–1111.

[61] L. Tu, X. Ouyang, J. Zhou, Y. He, and G. Xing, "Feddl: Federated learning via dynamic layer sharing for human activity recognition," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pp. 15–28.

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

[63] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, and J. L. Reyes Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 437–442.

[64] M. Boukhechba, L. Cai, C. Wu, and L. E. Barnes, "ActiPPG: Using deep neural networks for activity recognition from wrist-worn photoplethysmography (PPG) sensors," *Smart Health*, vol. 14, p. 100082.

[65] Z. D. Tekler, R. Low, Y. Zhou, C. Yuen, L. Blessing, and C. Spanos, "Near-real-time plug load identification using low-frequency power data in office spaces: Experiments and applications," *Applied Energy*, vol. 275, p. 115391.

[66] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639.

[67] R. Low, L. Cheah, and L. You, "Commercial vehicle activity prediction with imbalanced class distribution using a hybrid sampling and gradient boosting approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1401–1410.

[68] Z. D. Tekler and A. Chong, "Occupancy prediction using deep learning approaches across multiple space types: A minimum sensing strategy," *Building and Environment*, vol. 226, p. 109689.

[69] A. Graves, *Long Short-Term Memory*. Springer, 2012.

[70] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, "1-D convolutional neural networks for signal processing applications," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 8360–8364.

[71] P. Baldi and P. J. Sadowski, "Understanding dropout," *Advances in neural information processing systems*, vol. 26, 2013.

[72] P. Netrapalli, "Stochastic gradient descent and its variants in machine learning," *Journal of the Indian Institute of Science*, vol. 99, no. 2, pp. 201–213.

[73] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. Ieee, pp. 1–2.

[74] M. C. Mukkamala and M. Hein, "Variants of rmsprop and adagrad with logarithmic regret bounds," in *International Conference on Machine Learning*. PMLR, pp. 2545–2553.

[75] Z. Qu, P. Richtárik, and T. Zhang, "Quartz: Randomized dual coordinate ascent with arbitrary sampling," *Advances in neural information processing systems*, vol. 28.

[76] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches." *Journal of Machine Learning Research*, vol. 13, no. 1.

[77] V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi, "CoCoA: A general framework for communication-efficient distributed optimization," *Journal of Machine Learning Research*, vol. 18, p. 230.

[78] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, pp. 429–450. [Online]. Available: https://proceedings.mlsys.org/paper/2020/file/38af86134b65d0f10fe33d30dd76442e-Paper.pdf

[79] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," *Advances in Neural Information Processing Systems*, vol. 31.

[80] V. Delafontaine, F. Schiano, G. Cocco, A. Rusu, and D. Floreano, "Drone-aided localization in lora iot networks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 286–292.

[81] S. Park, S. Yun, H. Kim, R. Kwon, and J. Ganser, "Forestry monitoring system using lora and drone," in *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, 2018, pp. 1–8.

[82] I. Donevski, N. Babu, J. J. Nielsen, P. Popovski, and W. Saad, "Federated learning with a drone orchestrator: Path planning for minimized staleness," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1000–1014, 2021.

[83] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[84] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 233–242.

[85] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407. [Online]. Available: https://www.nowpublishers.com/article/Details/TCS-042

[86] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[87] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[88] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," *arXiv preprint arXiv:2009.03561*, 2020.

[89] L. Sun and L. Lyu, "Federated Model Distillation with Noise-Free Differential Privacy," vol. 2, pp. 1563–1570. [Online]. Available: https://www.ijcai.org/proceedings/2021/216

[90] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao *et al.*, "Opacus: User-friendly differential privacy library in pytorch," *arXiv preprint arXiv:2109.12298*, 2021.