Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

Adaptive Single-layer Aggregation Framework for Energy-efficient and Privacy-preserving Load Forecasting in Heterogeneous Federated Smart Grids

Habib Ullah Manzoor, Atif Jafri, Ahmed Zoha [*]

James Watt School of Engineering, University of Glasgow, United Kingdom

A R T I C L E I N F O

Keywords:
Resource constraint devices
Load forecasting
Federated learning
Heterogeneity
Quantization
Efficient communication

A B S T R A C T

Federated Learning (FL) enhances predictive accuracy in load forecasting by integrating data from distributed load networks while ensuring data privacy. However, the heterogeneous nature of smart grid load forecasting introduces significant challenges that current methods struggle to address, particularly for resource-constrained devices due to high computational and communication demands. To overcome these challenges, we propose a novel Adaptive Single Layer Aggregation (ASLA) framework tailored for resource-constrained smart grid networks. The ASLA framework mitigates data heterogeneity issues by focusing on local learning and incorporating partial updates from local devices for model aggregation in adaptive manner. It is optimized for resource-constrained environments through the implementation of a stopping criterion during model training and weight quantization. Our evaluation on two distinct datasets demonstrates that quantization results in a minimal loss function degradation of 0.01% for Data 1 and 1.25% for Data 2. Furthermore, local model layer optimization for aggregation achieves substantial communication cost reductions of 829.2-fold for Data 1 and 5522-fold for Data 2. The use of an 8-bit fixed-point representation for neural network weights leads to a 75% reduction in storage/memory requirements and decreases computational costs by replacing complex floating-point units with simpler fixed-point units. By addressing data heterogeneity and reducing storage, computation, and communication overheads, the ASLA framework is well-suited for deployment in resource-constrained smart grid networks.

1. Introduction

Load forecasting is essential for effective energy management and planning, aiding organizations and policymakers in resource allocation, infrastructure development, and energy policy formulation [1,2]. Accurate forecasts help optimize energy generation, distribution, and utilization, addressing the complex demands of data-rich smart grid systems [1]. Energy forecasting encompasses predictions of electricity demand, pricing, and the availability of energy sources such as fossil fuels and renewables [3]. These predictive and energy management systems are based on historical data, economic trends, technological advancements, and environmental factors [4].

Traditional energy systems rely on centralized infrastructures, where large power plants generate electricity for widespread areas through extensive transmission and distribution networks [5]. However, these systems face high costs and numerous operational

* Corresponding author.
E-mail addresses: h.manzoor.1@research.gla.ac.uk (H.U. Manzoor), Atif.Jafri@glasgow.ac.uk (A. Jafri), Ahmed.Zoha@glasgow.ac.uk (A. Zoha).

https://doi.org/10.1016/j.iot.2024.101376
Received 18 June 2024; Received in revised form 9 September 2024; Accepted 17 September 2024
Available online 20 September 2024

challenges. The future energy landscape is shifting towards distributed energy resources (DERs), including renewable technologies like solar photovoltaics and energy storage, which are becoming more efficient, flexible, and cost-effective [5–7].

Further more, the integration of AI in load forecasting has significantly improved prediction accuracy by analyzing vast amounts of complex data to identify intricate patterns [8–10]. Supervised machine learning algorithms are particularly well-suited for using past data to predict future energy events and consumption [11]. The use of deep prediction models for load forecasting may raise privacy concerns, as they require users to provide a large amount of personal electricity consumption data [12]. Developing privacy-preserving techniques is essential to address this challenge [12].

To combat the challenges of distributed load forecasting and privacy issues, a new machine learning method called Federated Learning (FL) was introduced [13]. FL allows various clients, such as energy providers or smart grid devices, to work together to train a shared machine learning model without having to share their individual data [13]. This method helps address the issues of data isolation and privacy concerns associated with traditional centralized machine learning [14]. FL separates model training from the need to store training data centrally, enabling clients to maintain the privacy of their sensitive energy consumption and production data [15]. FL can also enhance the robustness of the load forecasting system without the need for any additional anomaly detection framework against model and false data injection attacks [16,17].

Energy data is often fragmented and spread across various entities, including energy providers, aggregators, and consumers. This fragmentation can hinder the operational efficiency of energy aggregation service providers and obstruct the ability to learn and analyze relevant patterns and insights from the data [18]. Additionally, geographical differences, influenced by regional factors like climate, population density, and local energy policies, lead to diverse data characteristics across different locations within the grid [19,20]. Temporal variability further complicates forecasting, with energy usage fluctuating based on the time of day, season, or specific events [21,22]. These aspects of data heterogeneity pose a significant issue in FL as they can lead to poor forecasting accuracy [23].

Another key challenge is the high communication overhead involved in training the model across numerous distributed devices [24]. Frequent exchanges of model updates between devices and the central server can consume significant network bandwidth, particularly in resource-constrained settings [24]. User behavior and usage patterns introduce further variability, making uniform forecasting models challenging to develop. Therefore, there is a need for a technique that can effectively address data heterogeneity in distributed load forecasting while being computationally inexpensive.

In this paper, we introduce the Adaptive Single Layer Aggregation (ASLA) framework specifically designed for heterogeneous smart grid energy forecasting. ASLA aggregates only a single layer of the neural network at the server, introduces effective stopping criteria for FL training, and employs weight quantization to minimize communication and computational overhead. Our framework aims to efficiently manage data heterogeneity in energy forecasting while simultaneously enhancing both communication and computational efficiency, thus making it suitable for resource-constrained devices within smart grids.

The following items describe the main contributions of this work:

(1) Designed two separate deep neural networks for two different datasets of smart grids for short-term load forecasting (STLF) in both centralized and FL environments.
(2) Proposed a lightweight framework for distributed load forecasting, called the Adaptive Single Layer Aggregation (ASLA) framework. The framework uses a single layer of a neural network for aggregation in adaptive manner, to be optimized for resource constraint devices.
(3) Defined an early stopping criterion during the training of the global model to improve communication efficiency.
(4) Analyzed the effects of weight quantization on memory usage, as well as communication and computational overheads, to assess the suitability of the proposed ASLA framework for resource-constrained devices.

The rest of the paper is structured as follows: Section 2 describes the related work. Section 3 provides an overview of load forecasting in FL-assisted networks and introduces our proposed model. Section 4 presents the details of our proposed model. Section 5 discusses the simulation setup and the results of both the baseline and proposed frameworks. Section 6 covers the findings and insights derived from the proposed work and the results obtained. Section 7 offers a discussion on the results, including a comparison with state-of-the-art frameworks. Finally, Section 8 concludes the paper.

2. Background and related work

In FL-enabled smart grids, two primary challenges are data heterogeneity and communication/computational efficiency. Data from various devices and sensors within smart grids vary in format, volume, and quality, creating significant hurdles for FL models [25,26]. Integrating and harmonizing these diverse data sources while maintaining privacy and security is complex. FL also relies on effective communication between edge devices and a central server for model updates. In smart grids with many distributed devices, this can lead to bottlenecks, delays, limited bandwidth, memory and computational constraints, and increased energy consumption [27]. Optimizing communication and computational protocols is essential for efficient and reliable data transmission while minimizing processing costs in smart grid environments.

The state-of-the-art related work is divided into two sections: Data Heterogeneity in Energy Networks and Communication Efficiency.

2.1. Data heterogeneity in load forecasting

Load forecasting plays a crucial role in smart grid applications, where data from various sources, such as wind farms and solar plants, need to be combined to improve predictions [1]. Addressing the heterogeneity in this data can lead to more accurate and reliable load forecasts [1]. Different types of heterogeneity in FL are discussed in [28], including data, communication, device, model, task, and privacy heterogeneity. In this work, we focus on data heterogeneity, characterized by clients often being spatially distant from one another, which can impede communication. The distributions of data acquired by clients may exhibit skewness due to the possibility of each client gathering data from distinct sources [29]. Data heterogeneity can be further categorized into Distribution Skew [30], Label Skew [31], Feature Skew [32,33], Quality Skew [34,35], and Quantity Skew [36,37]. In this paper, our focus is on Distribution Skew, which refers to the imbalance in the distribution of data held by different clients. Specifically, this skew occurs when the distribution of features or the joint distribution of data across different clients diverges from the overall data distribution. Skewed data can cause forecasts to be biased and less accurate, as the model may find it challenging to correctly capture the true pattern of the demand distribution [38]. For instance, energy usage can vary significantly based on factors such as occupancy levels, building characteristics, and appliance ownership [39].

Data heterogeneity in FL poses challenges in achieving model convergence and maintaining consistency due to variations in local datasets. In our previous work, we explored the challenges associated with the convergence of local models due to the highly diverse data in smart grids [40]. We proposed a zero knowledge-based clustering framework to identify the diverse clients and group them together to make multiple mini-global models. Some other clustering frameworks such as FedCluster [41], FedSeq [42], FedLabCluster [43], FedLC [44], FedDDB [45], IFCA [46], Clustered Federated Learning [47], generative adversarial network-based clustering [48], Cluster-driven Graph Federated Learning (FedCG) [49], ClusterFL [50], FedSoft [51], socially-aware-clustering-enabled dispersed FL [52], Clustered Sampling [53], Clustered Vehicular Federated Learning [54], FedGroup [55], FlexCFL [56], pp-CFL [57], WSCC [58], One-Shot Federated Clustering [59], DQRE-SCnet [60], ClusterGrad [61], AutoCFL [62], FLIS [63], ACFL [64], ACS-FL [65], FedCE [66], FedCO [67], Federated FCM [68], ZeKoC [69], Genetic CFL [70], MACFL [71], LBAA-FedAVG [72], and FedClamp [73] have been proposed. These frameworks utilize clustering to separate clients that behave unusually based on predefined rules. However, clustering approaches come with two main disadvantages. Firstly, they add complexity to the system. Secondly, not all clients can effectively learn from each other within this framework as multiple global models are required [74,75]. Additionally, there is the issue of scalability; with a higher number of clients, we need to manage a larger number of global models and clusters. For example, in [40], the dataset had nine clients, and the proposed framework divided them into three clusters, demonstrating the scalability limit. In [69], clustering is performed three times at each communication round, and all obtained clusters are tested on global data to check the performance of individual clusters. This adds extra complexity to the system and requires more time and resources.

Another technique that is used for managing heterogeneity in FL is parameter decoupling [28]. These methods consider the model's parameters as a collection of components, each contributing differently to the training or inference process. The granularity of parameter decoupling can vary: some methods split the model into a backbone and a head, while others decouple it more finely, such as by layers, channels, or even coordinates. FedPer [76] is a method that divides deep learning models into shared base layers and local personalization layers. The base layers are trained collectively to develop general representations, while the personalization layers are trained individually on each device to adapt to specific user preferences. Similarly, FedRep [77] splits the model into a representation section and a head section. Clients work together to train a global representation federatively, then independently train their own head sections using the shared representation. However, in these frameworks, the performance can be greatly influenced by how the models are divided into different parts. The optimal configuration varies depending on the task and data, requiring meticulous tuning. Furthermore, devices with very limited local data might not have enough samples to effectively train their personalization layers, thereby limiting the advantages of personalization.

In FL, knowledge distillation involves transferring insights from a well-trained "teacher" model to a simpler "student" model. This means devices share model predictions instead of raw data. By exchanging these predictions, devices can benefit from the collective knowledge of all participating devices, improving performance on diverse datasets while keeping local data private [78,79]. Knowledge distillation helps create a more consistent model by merging the outputs of several local models, which balances out differences in their underlying data. This combined approach improves the model's ability to generalize across various data distributions [80–82]. Knowledge distillation seeks to cut down communication costs in FL by sharing distilled knowledge instead of raw model parameters. However, the added complexity of the distillation process can sometimes lead to more information being exchanged than intended. Finding the right balance between model efficiency and communication needs is a key challenge [83,84].

Prototype FL also addresses data heterogeneity among clients by creating and using prototype representations of local data [85]. Each client generates lower-dimensional prototypes that capture key features and patterns of their data. These prototypes are sent to a central server, which aggregates them into global prototypes representing the overall data distribution. The global prototypes are then distributed back to the clients, who use them to refine their local models, ensuring they integrate knowledge from the collective data. This process helps manage diverse data distributions effectively [86]. However, prototype FL suffers from prototype quality issues, where prototypes must accurately capture the essential characteristics of the local data [87]. Additionally, scalability poses a challenge, as managing and aggregating prototypes efficiently becomes more difficult as the number of clients increases [87].

2.2. Communication efficiency

Communication and computational efficiency during FL training process is achieved by three well-known methods: (1) Model compression [88], and (2) Compression on communication (early stopping) [72], (3) Communication Topology Optimization.

2.2.1. Communication topology optimization

Optimizing communication topology in FL plays a crucial role in improving both efficiency and sustainability, particularly for resource-constrained devices [89]. By carefully selecting and managing the connections between devices, the amount of data exchanged during model updates is minimized, which reduces communication overhead and accelerates model convergence [90,91]. This optimization not only lowers energy consumption but also decreases the computational burden on devices with limited resources, extending their operational lifespan. Additionally, a well-optimized communication topology can enhance the robustness of FL systems by ensuring that the most critical data is prioritized, further improving the overall performance and feasibility of FL in resource-constrained environments.

2.2.2. Model compression

Model compression is a widely adopted technique [92] to reduce the computational and memory demands of a neural network in FL. Pruning [93], sparsification [94], gradient compression [94], and quantization [95] are three existing methods used to simplify the inference and training processes of neural networks.

(i) **Pruning:** Pruning selectively removes connections (weights) between neurons based on certain criteria, aiming to reduce computational complexity and memory requirements while maintaining or improving performance. Pruning can be performed either during or after training and is crucial for deploying large-scale deep learning models on resource-constrained edge devices [96]. While very wide and deep neural networks often improve the loss function, not all weights contribute equally. By assessing weights, those with minimal impact can be removed [97,98].
– Unstructured Pruning: This involves replacing certain weight values with zeros [99,100]. Although it can achieve higher compression rates, it may still make the inference stage computationally intensive unless supported by hardware like modern GPUs [101–103].
– Structured Pruning: This method removes entire channels or nodes, making convolution operations smaller but still dense [104–106]. Structured pruning is preferable when aiming to speed up inference on standard hardware.

(ii) **Sparsification:** Sparsification induces sparsity in the neural network's parameters or gradients by setting a proportion of them to zero [107]. This reduces communication overhead in FL by transmitting only non-zero parameters or gradients between clients and the server, significantly lowering the required communication bandwidth [107,108]. Sparsification is commonly used to enhance communication efficiency during distributed training and to simplify neural network complexity [99,109–112].
Gradient compression: Gradient compression for communication efficiency is a technique employed in distributed machine learning to mitigate the data volume exchanged between nodes, such as clients and a central server in FL. By reducing the bandwidth required for transmitting gradient updates, this method aims to enhance communication efficiency without compromising the quality of model training. It leverages various strategies, such as quantization and sparsification, to strike a balance between minimizing communication overhead and maintaining effective convergence of the model [94,113,114]. As the scale of distributed systems grows, the importance of gradient compression becomes increasingly critical in ensuring scalable and efficient training processes [115].

(iii) **Quantization:** Representing neural network parameters or gradients in lower-precision fixed-point numbers [116,117] instead of high-precision floating-point format reduces the precision. However, on the other side, quantizing the gradients in fixed-point data format reduces the memory, computational power and communication bandwidth requirements for transmitting parameters or gradients between clients and the server, thereby improving overall efficiency in FL.

Quantization, pruning, and sparsification each have different effects on neural networks. Research indicates that quantization is generally more effective than pruning at reducing the size and computational load of neural networks [118]. In [119], the authors introduced the AGQFL quantization strategy, which enhances FL by dynamically compressing gradient information. This approach uses a stochastic quantization function to map continuous gradient values to discrete levels while adapting precision through the Model Quantization Indicator (MQI) and a probability-based adjustment mechanism. However, this technique requires more time than traditional FL and is therefore not recommended for resource-constrained devices. Similarly, authors in Guo et al. [115] presented a partition-based gradient compression method, a novel communication compression technique, but it also requires more time than traditional FL. Consequently, in this work, we adopt a simpler form of quantization.

By default python uses 32 float system. Floating point representation works well for numbers with large dynamic range. Based on the number of bits, there are two representations in IEEE 754 standard [120]. In this standard a normalized floating point number x is stored in three parts: the sign s, the excess exponent e, and the mantissa m, whereas bias b has a fixed value which is 127 for 32-bit floating point number [121]:

$$x = (-1)^s \times (1+m) \times 2^{e-b} \tag{1}$$

This method offers a broad range of numbers with variable precision but requires increasing computational complexity and higher hardware resource consumption.

In contrast, $Qn.m$ fixed-point numbering format maintains a fixed number of bits n and m for the integer and fractional parts respectively. Hence, the number x using $n+m$ bits is represented as $b_{n-1}, b_{n-2}, \dots b_1, b_0, b_{-1}, b_{-2}, ..b_{-m}$ where $b \in \{0,1\}$. The value of the number is calculated using following expression [122]:

$$x = -2^{n-1} b_{n-1} + \sum_{i=-m}^{n-2} 2^i b_i \tag{2}$$

Fixed-point arithmetic offers the required precision and range, while simultaneously being simpler and more resource-efficient, facilitating far less complex hardware implementation when compared with its floating-point counterpart.

In microcontrollers, the disparity in processing between floating-point and fixed-point arithmetic is profound. Floating-point arithmetic relies on complex algorithms and hardware support to manage variable precision and number ranges, imposing a significant computational burden on microcontroller units (MCUs) with limited processing power and memory resources [123]. Conversely, fixed-point arithmetic presents a simpler and more efficient alternative, involving straightforward integer arithmetic operations without necessitating specialized hardware or software libraries [124]. By assigning a fixed number of bits for the integer and fractional parts of a number, fixed-point arithmetic enables precise calculations within a predetermined range. This simplicity renders fixed-point arithmetic well-suited for embedded systems, prioritizing computational efficiency and resource utilization [125]. Furthermore, the deterministic nature of fixed-point arithmetic facilitates predictable and reproducible results, critical for real-time applications prevalent in embedded systems. Consequently, transitioning from floating-point to fixed-point representation not only conserves computational resources but also enhances the reliability and efficiency of microcontroller-based systems [126].

2.2.3. *Compression on communication*

Compression on communication also known as stopping criteria [72]. Early stopping in FL is crucial to prevent overfitting and ensure efficient resource utilization [127]. It helps halt training when local models reach optimal performance, preventing wasted computational resources and potential divergence from the global model. Without early stopping, there is a risk of overfitting and inefficient resource usage. Conversely, stopping training too early may lead to underfitting and suboptimal performance. Implementing early stopping balances performance with resource efficiency, promoting better generalization and reducing training time and cost [128].

2.3. *Adaptive FL*

Adaptive FL refers to a variant of FL that dynamically adjusts its training process or communication strategies based on the characteristics and constraints of the participating devices or the data they hold. Unlike traditional FL, which relies on a fixed training process and communication strategy, Adaptive FL modifies these elements on the fly, tailoring them to the specific characteristics and limitations of the participating devices or the data they possess [129,130]. Adaptive FL takes into account the resource limitations of participating devices, such as limited computational power, memory, and network bandwidth, and adjusts the training process to fit these constraints. This adaptability enhances the overall efficiency and performance of the FL system [131,132] and this makes adaptive FL suitable for resource constraint devices [133,134]. Adaptive FL dynamically adjusts its processes based on factors such as device performance and network latency. In frameworks like ACE-Sniper [135], inference latency modeling is crucial for optimizing scheduling efficiency across heterogeneous devices. Similarly, in Adaptive FL, devices with varying hardware capacities (e.g., GPUs, NPUs) have different response times, and by incorporating performance and latency models, Adaptive FL can allocate resources more effectively by predicting device performance. Furthermore, Adaptive FL benefits from network analysis to dynamically adjust communication strategies. By leveraging hardware resource modeling (HRM) to predict computational loads, it balances tasks across devices and extends this approach to managing bandwidth and communication frequency, ensuring that resource-constrained devices can participate in training without being overwhelmed [136,137]. Adaptive FL can also enhance its adaptability by implementing task scheduling algorithms that consider device heterogeneity and network conditions, optimizing client selection based on available resources, which in turn reduces latency and improves convergence speed. This leads to more efficient task assignments across devices with varying computational capabilities and network conditions [138]. Moreover, client devices in Adaptive FL are selected based on their potential to contribute meaningfully to the global model updates. This involves prioritizing devices that provide high data utility, which is essential for improving the accuracy and generalizability of the model [139]. Additionally, Adaptive FL is better suited to manage non-i.i.d. (non-independent and identically distributed) data across devices, a common challenge in real-world FL scenarios. It employs techniques like Inverse Distance Aggregation (IDA) to address data heterogeneity [140]. Some Adaptive FL approaches also emphasize personalization by fine-tuning the global model for each client based on their local data, resulting in improved performance on individual devices [131].

2.4. *Gap analysis*

The literature survey indicates a pressing need for a lightweight, adaptive framework to address the challenges of diverse data and fluctuating conditions in load forecasting. Existing methods, including clustering, prototype generation, and distillation techniques, each present significant drawbacks. Clustering-based approaches complicate the system architecture, lack scalability, and consume substantial resources, often failing to leverage all client data by focusing only on high-performing clusters. Prototype generation, while reducing the data volume, can lead to loss of important information and model oversimplification. Distillation methods, which aim to transfer knowledge from complex models to simpler ones, often involve high computational and communication costs, making them impractical for resource-constrained environments. Consequently, there is a clear necessity for a novel, adaptive framework that not only enhances forecasting accuracy and performance but also conserves resources like bandwidth, energy, and computational power. Such an adaptive solution would dynamically adjust to the varying conditions and heterogeneity of data, streamline data processing, lower operational overhead, and support scalable and robust energy management systems, ultimately contributing to their sustainability and cost-effectiveness.

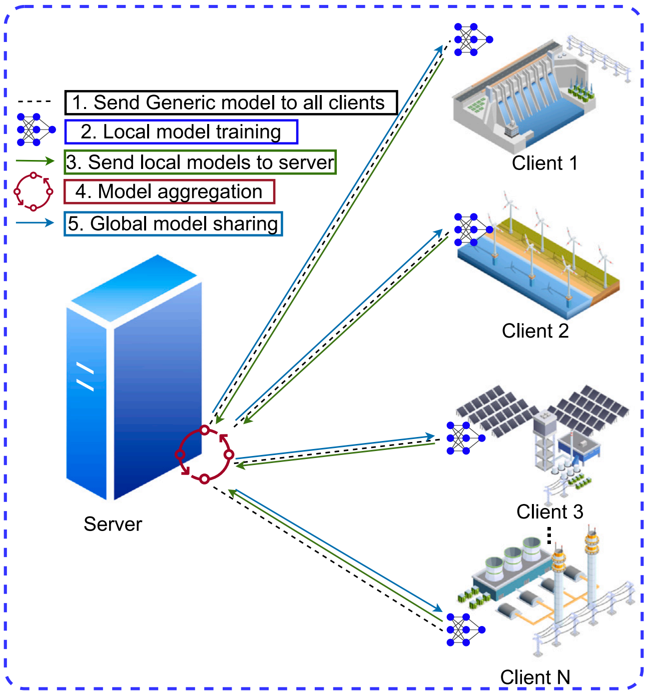

Fig. 1. An overview of FL assisted smart grids.

3. FL assisted load forecasting

FL in load forecasting networks involves sharing a machine learning model among various edge devices and a central server while maintaining the privacy of raw data. FL efficiently addresses privacy concerns and mitigates the data island problem. The FL training process for enhancing smart grid functionalities includes several key steps, as depicted in Fig. 1:

(1) **Sending Generic Models to Clients:** In smart grids, devices like smart meters and energy management systems collect energy consumption data. FL enables the central server to deploy a generic machine learning model, customized for specific energy forecasting needs, to all edge devices (clients) without compromising sensitive data. The generic model with L layers is distributed to all K edge devices, derived from historical data and expert knowledge.

(2) **Local Model Training:** Each edge device utilizes its locally stored energy consumption data at smart grids to train the generic machine learning model received from the server. This local training ensures that individual energy consumption patterns from each smart grids and behaviors are captured, contributing to the development of accurate and personalized local models.

(3) **Model Update and Loss Calculation:** After local training, edge devices communicate their local model's weights and biases ($UP = W^1_{t,1}, W^2_{t,2}, \dots, W^K_{t,L}$) along with their corresponding loss functions (σ_k) back to the server. Here, $W^1_{t,1}$ and $W^K_{t,L}$ represent the weights of the first layer of the first client and the L_{th} layer of the K_{th} client, respectively, at communication round t.

(4) **Model Aggregation:**The server aggregates all received local models to create a new machine learning model, known as the global model (G_t). We adopt a layer-by-layer aggregation approach, with G^k_t represented as:

$$G_t = \begin{cases} G^1_t \\ G^2_t \\ \dots \\ G^L_t \end{cases} = \begin{cases} \frac{\sum_{i=1}^{K} W^i_{t,1}}{K} \\ \frac{\sum_{i=1}^{K} W^i_{t,2}}{K} \\ \dots \\ \frac{\sum_{i=1}^{K} W^i_{t,L}}{K} \end{cases} \tag{3}$$

Here, G^1_t represents the aggregated model of the first layer, and $W^i_{t,L}$ indicates the weights of the L_{th} layer of the i_{th} client.

(5) **Global Model Sharing:** Finally, the server communicates the global model G^k_t back to all edge devices, completing one communication round. The iterative process enhances the accuracy of the global model over successive rounds

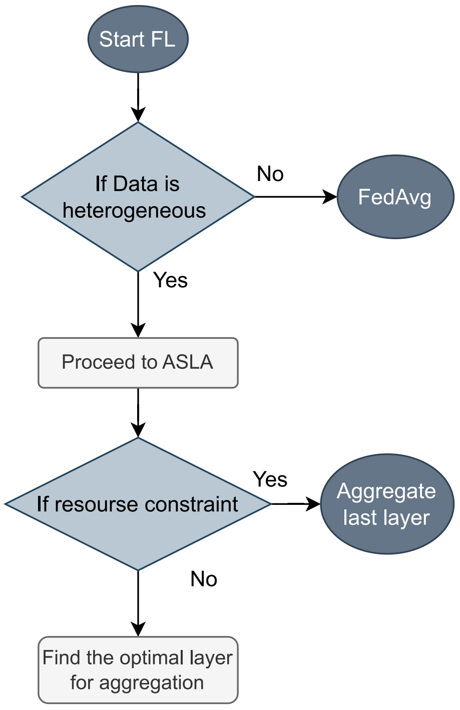

Fig. 2. Adaptive framework of ASLA.

4. Adaptive single layer aggregation

The Adaptive Single Layer Aggregation (ASLA) framework is designed to effectively address the challenges posed by data heterogeneity in load forecasting and resource constraints in energy networks. The framework incorporates three primary components: adaptive single-layer aggregation, stopping criteria, and weight quantization, each contributing to improved efficiency and performance in FL environments.

4.1. Adaptive single-layer aggregation

Traditional FL methods typically aggregate all layers of a neural network from local models, ensuring that all clients participate in global learning with maximum potential. However, to enhance local learning and limit global participation, the ASLA framework introduces partial participation of local models. This is achieved by aggregating only select layers of neural networks, leaving non-aggregated layers unchanged. The layer for aggregations selected based on the resources of the local devices. If devises have resources constraints then last layer is chosen otherwise an optimal layer for aggregation should be selected, based on experimental results. The adaptive framework is presented in Fig. 2. By doing so, it:

(1) Reduces the computational and communication burden on clients, as fewer parameters need to be transmitted and processed.
(2) Enhances local learning by allowing clients to optimize their models more effectively with locally relevant data.
(3) Increases security by limiting the potential impact of adversarial attacks to a single layer, thereby protecting the overall integrity of the local models [141].

4.2. Stopping criteria

Efficient training in FL requires carefully designed stopping criteria to optimize resource utilization. The ASLA framework introduces a dual approach for stopping criteria:

(1) At the client side, training is stopped if the client's loss function shows no improvement for certain consecutive communication rounds. This prevents unnecessary computation and transmission when further updates do not enhance model performance. Client will stop sending the updates to server.
(2) At the server level, training ceases if a predefined percentage, x of clients stop sending updates (UP), indicating that additional rounds are unlikely to yield significant improvements. If any individual client stop sending the updates then server will use the last update received for aggregation. This approach ensures that resources are not wasted on redundant training iterations. Stopping criteria is depicted in Fig. 3.

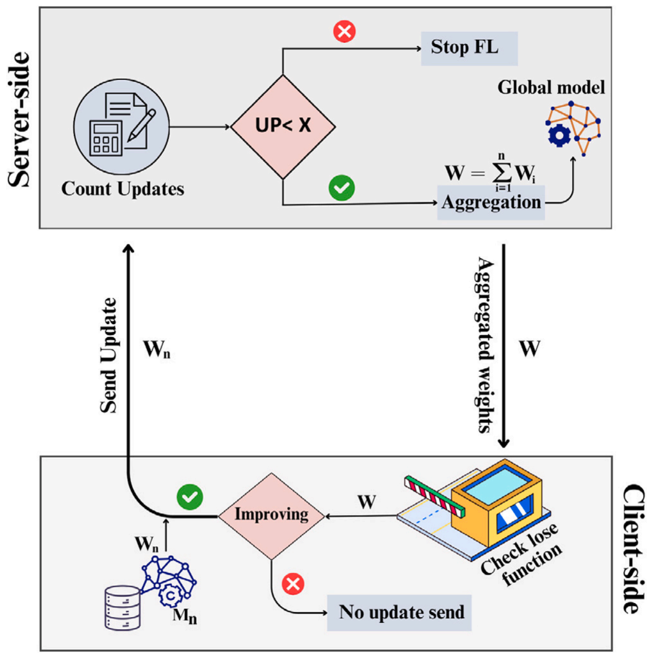

Fig. 3. Block diagram of early stopping criteria.

4.3. Weight quantization

Communication and computational overhead along side memory requirements in FL can be significantly reduced through weight quantization. In ASLA, neural network weights are quantized from 32-bit floating-point to 8-bit fixed-point precision. This reduction decreases the amount of data that needs to be transmitted during model updates, thereby conserving bandwidth and accelerating communication.

5. Simulation setup and results

In this section, we provide a detailed explanation of the simulation setup, used datasets, deep learning models and then discuss our results.

Data 1: To evaluate the performance of the ASLA framework and compare it with vanilla FL, we established a FL environment consisting of ten clients. We employed real-world energy data sourced from PJM Interconnection LLC [142], where each column represents energy usage by a specific substation. A sample of the dataset is illustrated in Fig. 4. Each client is allocated 13,896 samples, highlighting the dataset's significant diversity, with no two clients exhibiting similar characteristics. A clear data heterogeneity can be due to high data diversity can be seen in Fig. 4.

Data 2: The second dataset belongs to UK smart grids provided by Ausgrid [143]. We randomly selected 8 clients named Hutton, Aberdeen, Mona, Moonan, Morisset, Harbord, Kingsford, and Kirrawee. Each client has 5276 samples of energy consumption in MW, collected every 15 min. A sample of the data is plotted in Fig. 5. It can be seen that there is a significant difference in data distribution among the clients.

Levene's test: We investigated the heterogeneity of energy consumption patterns among clients within our smart grid network by applying Levene's test [144], a robust statistical method designed to assess the equality of variances between groups. This test offers a rigorous means of comparing the variability in energy consumption across different clients, thereby enabling us to discern whether disparities in consumption patterns are statistically significant.

In Levene's test, the test statistic W is computed based on the absolute deviations of individual data points from their respective group means. The formula for Levene's test statistic W is:

$$W = \frac{(N-k)}{(k-1)} \times \frac{\sum_{i=1}^{k} N_i(\bar{Z}_{i.} - \bar{Z}_{..})^2}{\sum_{i=1}^{k}\sum_{j=1}^{N_i}(Z_{ij} - \bar{Z}_{i.})^2} \tag{4}$$

Where W is the test statistic, N is the total number of observations, k is the number of groups being compared, N_i is the number of observations in the ith group, $\bar{Z}_{i.}$ is the mean of the ith group, $\bar{Z}_{..}$ is the overall mean, and Z_{ij} is the jth observation in the ith group.

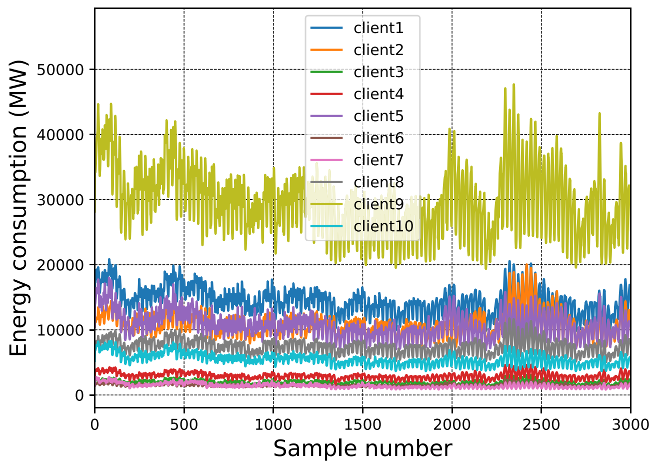

Fig. 4. A sample of data 1.

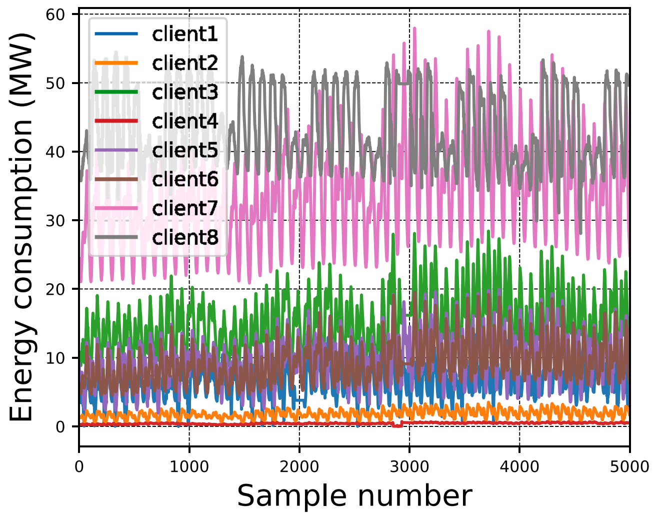

Fig. 5. A sample of data 2.

This test statistic follows an F-distribution with $(k-1)$ and $(N-k)$ degrees of freedom, under the assumption that the variances are equal across groups. The p-value tells us how likely it is to see a test result like W if the groups really do have the same variances, based on the F-distribution. So, if the p-value is small, it means W is unusual if the variances are equal. And if the p-value is large, it means W is more common, suggesting the variances might actually be similar.

A low p-value, typically below a chosen significance level (e.g., 0.05), suggests that the observed differences in variance between groups are statistically significant. In other words, if the p-value is below the significance level, it indicates that the variances are sufficiently different to reject the null hypothesis of homogeneity, implying data heterogeneity. Conversely, a high p-value, above the chosen significance level, suggests that there is insufficient evidence to reject the null hypothesis. This indicates that the observed differences in variance between groups are not statistically significant, supporting the notion of data homogeneity. The p-values from Levene's test for data 1 and data 2 are tabulated in Table 1 and in Table 2, respectively. Therefore, in our analysis, consistently low p-values across all client pairs highlight significant differences in energy consumption variability, reinforcing the presence of data heterogeneity within our smart grid network.

Bartlett's test: We also used Bartlett's test [145] to further test the data heterogeneity. Bartlett's test is a statistical procedure used to determine whether multiple samples have equal variances, also known as testing for *homoscedasticity*. It is particularly useful when comparing the variances of multiple datasets to assess whether they can be assumed to have the same level of variability. The null hypothesis (H_0) for Bartlett's test is that all the variances are equal, while the alternative hypothesis (H_1) posits that at least one of the variances is different.

Table 1
Matrix of p-values of Levene's test for Data 1.

	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9	C 10
C 1	0.0	1.02×10^{-59}	0.0	0.0	8.39×10^{-1}	0.0	0.0	0.0	0.0	0.0
C 2	1.02×10^{-59}	0.0	0.0	0.0	1.17×10^{-56}	0.0	0.0	0.0	0.0	0.0
C 3	0.0	0.0	0.0	0.0	0.0×10^{0}	3.64×10^{-226}	2.94×10^{-1}	0.0	0.0	0.0
C 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C 5	8.39×10^{-1}	1.17×10^{-56}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C 6	0.0	0.0	3.64×10^{-226}	0.0	0.0	0.0	4.02×10^{-183}	0.0	0.00	0.0
C 7	0.0	0.0	2.94×10^{-1}	0.0	0.0	4.02×10^{-183}	0.0	0.0	0.0	0.0
C 8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.83×10^{-118}
C 9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C 10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.83×10^{-118}	0.0	0.0

Table 2
Matrix of p-values of Levene's test for Data 2.

	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8
C 1	0.0	0.0	0.0	0.0	8.06×10^{-91}	2.019×10^{-20}	0.0	0.0
C 2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C 3	0.0	0.0	0.0	0.0	2.79×10^{-23}	1.15×10^{-83}	0.0	0.0
C 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C 5	8.066×10^{-91}	0.0	2.79×10^{-23}	0.0	0.0	2.054×10^{-23}	0.0	0.0
C 6	2.019×10^{-20}	0.0	1.15×10^{-83}	0.0	2.054×10^{-23}	0.0	0.0	0.0
C 7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.799×10^{-59}
C 8	0.0	0.0	0.0	0.0	0.0	0.0	1.799×10^{-59}	0.0

The Bartlett test statistic T is computed using the following formula:

$$T = \frac{(N-k)\ln(S_p^2) - \sum_{i=1}^{k}(n_i - 1)\ln(S_i^2)}{1 + \frac{1}{3(k-1)}\left(\sum_{i=1}^{k}\frac{1}{n_i-1} - \frac{1}{N-k}\right)} \quad (5)$$

Where k is the number of groups (datasets), N is the total number of observations from all groups, n_i is the number of observations in the ith group, S_i^2 is the variance of the ith group, S_p^2 is the pooled variance, calculated as:

$$S_p^2 = \frac{\sum_{i=1}^{k}(n_i - 1)S_i^2}{N-k} \quad (6)$$

After conducting Bartlett's test to assess the homogeneity of variances for two datasets, the results are as follows: For Data 1, the test statistic is 220,985.198295 with a p-value of 0.0. For Data 2, the test statistic is 57,220.713641 with a p-value of 0.0. Bartlett's test evaluates whether multiple samples have equal variances. The null hypothesis of the test states that the variances are equal across the groups. In this case, both datasets exhibit extremely small p-values (0.0), indicating that the null hypothesis of equal variances is strongly rejected. This suggests significant differences in variances between the groups for both datasets. Therefore, the data do not meet the assumption of homogeneity of variances, which could affect the validity of analyses that assume equal variances across groups.

Preprocessing: The data sets are utilized for STLF using five distinct features, which include the previous hour's value, the previous day's value, the previous week's value, the average of the last 24 h, and the average of the last week [40].

Deep Learning Model 1: For Data 1, a three layered ANNs was constructed. The ANN was composed of three dense layers: the initial layer featured 100 neurons, followed by a layer with 50 neurons, and concluded with a single neuron in the final layer. All layers had ReLU activation functions, and the Adam optimizer was employed with mean square error as the loss function. The dataset was split into a 70/30 ratio for training and testing purposes. We conducted FL for 100 communication rounds, with each client undergoing 1 local epoch and utilizing a batch size of 300. Additionally, FedAvg [146] was chosen as the aggregation method at the server.

Deep Learning Model 2: For Data 2, a four-layer LSTM-based deep learning model was constructed. The network comprised four layers: the initial LSTM layer featured 64 neurons, followed by a dense layer with 64 neurons, which was connected to another dense layer with 32 neurons, and concluded with a single neuron in the final dense layer. All layers used ReLU activation functions, and the Adam optimizer was employed with mean squared error as the loss function. The dataset was split into a 70/30 ratio for training and testing purposes. We conducted FL for 30 communication rounds, with each client undergoing 1 local epoch and utilizing a batch size of 64. As with Data 1, FedAvg [146] was chosen as the aggregation method at the server. All simulations were performed in Python, and the deep models were created using TensorFlow [147].

5.1. Baseline results

Data 1: After running FL for 100 communication rounds, we found that the loss function, Mean Absolute Percentage Error (MAPE), of local models did not converge. The results are depicted in Fig. 6. Among all ANNs, only clients numbered 2, 8, and 10

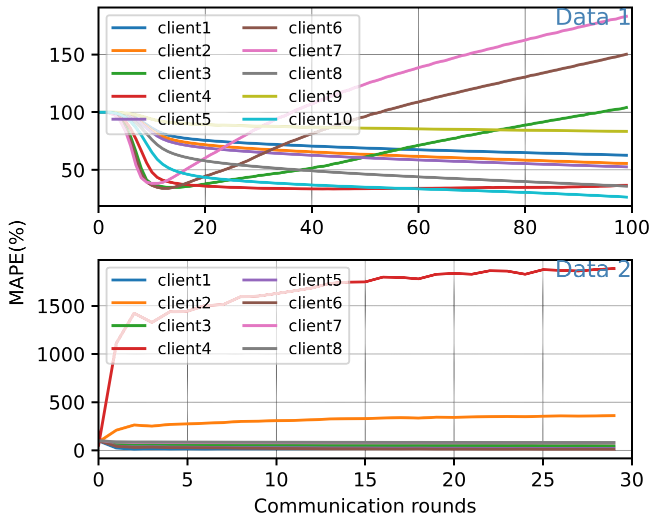

Fig. 6. MAPE of all local clients in baseline simulation.

Table 3
Average client MAPE of step by step aggregation of Data 1 and Data 2.

	Layer 1	Layer 1–2	Layer 1–3	Layer 1–4
Data 1	3.26	3.29	79.1	–
Data 2	8.4	9.6	14.6	312.97

appear to be converging, while clients 6 and 7 are the most affected. The calculated average MPAE was 79.1%. This discrepancy suggests that clients may be overly influenced by the updates from other clients, leading to suboptimal convergence. It is worth mentioning that using all the designed deep network can achieve a MAPE of approximately 3.1% if it is directly applied on individual clients (centralized machine learning). The increased MAPE is attributed to the fact that clients are learning too much from each other, potentially resulting in high model diversity.

Data 2: After performing FL for 30 communication rounds, we observed that the Mean Absolute Percentage Error (MAPE) of the local models did not stabilize. This is illustrated in Fig. 6, which shows that Clients 2 and 4 experienced the most significant deviations. The calculated average MPAE was 312.97%. This issue suggests that there is a need to further explore the communication and learning dynamics between clients in FL environments. Notably, if the full deep network were applied individually to each client in a centralized setting, it could achieve a MAPE of about 2.6%. The higher MAPE in the FL setting is likely due to the excessive exchange of information among clients, which may lead to increased model diversity.

5.2. Signal layer aggregation

To reduce the interdependence of local models and enhance localized learning, we conducted a series of simulations focusing on layer aggregation. Initially, we aggregated only the first layer while leaving subsequent layers unchanged. We repeated this process incrementally, aggregating additional layers step by step, across all designed ANNs. The results of these simulations are summarized in Table 3. It is notable that as long as all layers of any deep models are not aggregated, the MAPE for clients significantly improved. However, aggregating all layers in deep models for respective data sets resulted in significant increases in average MAPE for clients, reaching 79.1%, and 312.97%, respectively. This stark rise in average MAPE underscores the importance of partial layer aggregation, which enhances local learning while reducing the influence of other clients' model diversity. By adopting a partial aggregation approach, clients retain their individual learning trajectories, contributing to a more diverse ensemble of models and thereby improving overall performance.

Furthermore, we investigated the effectiveness of aggregating individual layers independently. We aggregated each layer individually while keeping the remaining layers unchanged. The results, summarized in Table 4, indicate that any single layer can be aggregated while still achieving an average MAPE of approximately 3.3% for clients in data 1. However, better results of approximately 2.8% are obtained for layer 2, 3 and 4 in data 2. This finding suggests that aggregating a single layer suffices, eliminating the need for aggregating all layers in the FL process. This streamlined approach not only addresses convergence issues resulting from high data diversity but also facilitates more efficient communication between the server and clients. With only one layer being exchanged, communication overhead is reduced, improved performance. The local MAPE of all clients from data 1 and data 2 are depicted in Fig. 7. It can be see that MAPE of all clients is converging when only first layer was used fro aggregation.

Table 4
Only a single layer aggregation of Data 1 and Data 2.

	Layer 1	Layer 2	Layer 3	Layer 4
Data 1	3.29	3.3	3.32	–
Data 2	8.4	2.76	2.84	2.96

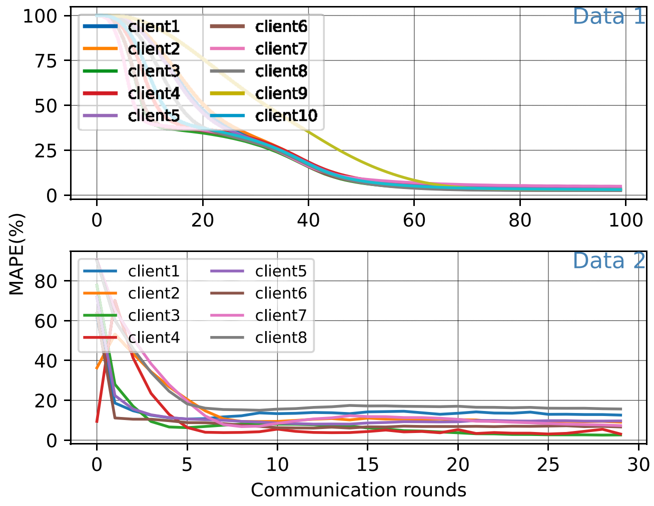

Fig. 7. MAPE of all clients during training of global model when only first layers were aggregated.

5.3. Quantization

Optimizing communication overhead can be achieved by sharing just a single layer, a strategy enhanced by quantization of the weights of both local and global models. Our experimentation in Python, employing float32 format, explored various quantization options, including float16 bits, fixed-point 32 bits, fixed-point 16 bits, and fixed-point 8 bits. We monitored the averaged client MAPE, focusing on a 3-layered neural network and aggregating solely the first layers of local models.

For a 32-bit fixed-point system, 8 bits are allocated for the integer part and 24 bits for the fractional part. In a 16-bit fixed-point system, 5 bits are allocated for the integer part and 11 bits for the fractional part. In an 8-bit fixed-point system, 2 bits are allocated for the integer part and 6 bits for the fractional part. This configuration reduces memory usage and allows the use of embedded systems with limited computational resources for this application. The results for Data 1, shown in Fig. 8, reveal negligible differences when reducing the number of bits from 32 to 16 in both floating-point and fixed-point formats, with both yielding an approximate MAPE of 3.3%. Even with the 8-bit fixed-point format, the resulting MAPE is 3.4%, which is acceptable for this application. Similarly, for Data 2, the best MAPE of 8.05% and the worst MAPE of 9.33% were obtained from clients using the 32-bit floating-point and 8-bit fixed-point systems, respectively.

5.4. Effect of layer-wise aggregation and stopping criteria

To enhance system energy efficiency, communication efficiency, and resource utilization, an early stopping criterion was introduced (see Fig. 3). If a client does not observe any improvement in the MAPE over five consecutive communication rounds, it will halt sending updates to the server. If the server does not receive updates from a client, it will automatically use the last received update for aggregation. Additionally, if the server does not receive updates from a predefined number of clients, denoted as x, it will stop the aggregation process. The value of x can be customized based on user requirements; in our experiments, we set $x = 3$.

In previous experiments, the number of communication rounds was fixed at 100. However, in this set of experiments, the number of communication rounds was determined by a stopping criterion. For Data 1 and Data 2, with a fixed 8-bit quantization scheme and only aggregating the first layer of local models at the server, the average client MAPE was plotted in Fig. 9. The objective of these experiments was to determine which layer aggregation method would yield the best results in terms of MAPE and the number of communication rounds. For Data 1, aggregating the first layer achieved an MAPE of 3.26% with 114 communication rounds, the second layer achieved an MAPE of 3.25% with 102 communication rounds, and the third layer provided an MAPE of 3.50% with 110 communication rounds. These results indicate that the second layer produced the most optimal outcome. For Data 2, layer aggregation required 142 rounds with an MAPE of 4.4%, the second layer required 72 rounds with an MAPE of 2.5%, the third layer required 48 rounds with an MAPE of 2.58%, and the final layer required 52 rounds with an MAPE of 2.4%.

These results highlight the importance of layer-wise optimization in distributed learning systems. By strategically allocating computational resources and communication efforts, significant improvements in model performance can be achieved. The observed

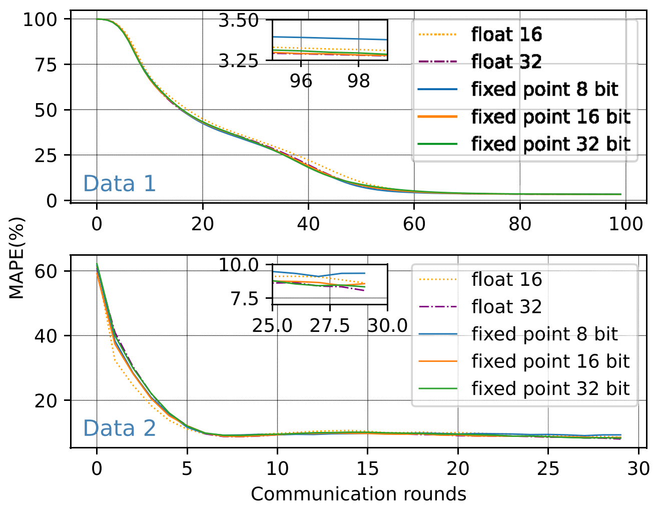

Fig. 8. The effect of quantization on communication rounds and on average client MAPE.

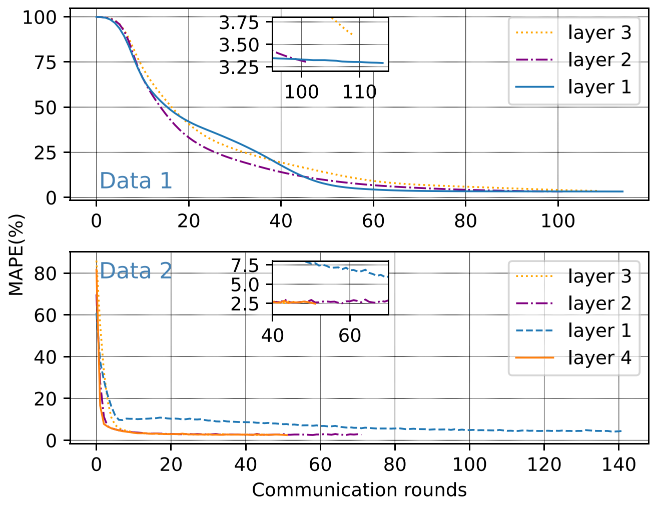

Fig. 9. The effect of quantization on communication rounds and on average client MAPE.

trade-off between accuracy and the number of communication rounds underscores the need for careful consideration of network architecture and communication protocols in distributed learning frameworks.

6. Findings and insights

Our proposed framework can not only address the issue of data heterogeneity in distributed load forecasting but also enhances the communication and computational efficiency of the system. This is achieved by sharing weights of only a single layer of neural network and implementing stopping criteria along with quantization requiring minimum bits.

The main advantages ASLA framework:

(1) Enhanced privacy
(2) Reduction in computation
(3) Reduction in memory requirement
(4) Reduced communication cost

These advantages are discussed below in detail.

6.1. Enhanced privacy

Privacy is the protection of personal information that holds sensitivity for an individual. While FL effectively guards against direct access to local private data by attackers, researchers have also highlighted a deeper privacy concern: the potential for information leakage arising from shared models between servers and clients [148,149]. Machine learning models often exhibit superior prediction accuracy on the data they are trained on, which can lead to over-fitting. Consequently, this accuracy gap can be exploited by adversaries, enabling them to potentially reverse-engineer users' private data using the shared models [150]. A disclosure attack involves scrutinizing traffic patterns, through which adversaries can discern specific groups of recipients based on observed network activity [151,152]. In the ASLA framework, we only share a single layer of local data, making it more difficult for attackers to replicate the original data, thereby enhancing privacy.

6.2. Reduced computation

In digital systems, the choice between floating-point and fixed-point arithmetic significantly impacts hardware requirements and performance. Floating-point systems, characterized by their higher complexity and larger size, require managing exponent calculations, normalization, and rounding, which demand additional logic circuits [153]. Consequently, floating-point units (FPUs) consume more silicon area on a chip, making them larger and more intricate. Moreover, the complexity of floating-point operations generally results in slower computational speeds and higher energy consumption, making them less suitable for power-sensitive applications.

In contrast, fixed-point systems offer simplicity, compactness, and speed. Arithmetic operations in fixed-point units omit exponent handling, significantly reducing design complexity [154]. This simplification leads to faster computation speeds as operations require fewer processing cycles. Additionally, fixed-point systems consume less silicon area, resulting in smaller hardware implementations. Reduced complexity and faster operations lead to lower power consumption [155], making fixed-point arithmetic particularly advantageous for applications prioritizing energy efficiency and speed, such as embedded systems, real-time processing tasks, and battery-powered devices.

Transitioning from a 32-bit floating point to an 8-bit fixed-point data format yields substantial savings in hardware design, speed, and power consumption. An 8-bit system requires significantly less silicon area due to smaller data paths, registers, and arithmetic units, reducing production as well as operational costs (in terms of battery life) . Despite potentially slower performance in large data operations, 8-bit systems handle smaller data chunks and have simpler instruction sets, resulting in faster execution of basic instructions [156]. Additionally, power consumption is greatly reduced in 8-bit systems. The smaller data width decreases transistor switching activity, leading to lower dynamic power usage, while simpler operations decrease static power dissipation. This enhanced energy efficiency makes 8-bit systems ideal for battery-powered and energy-constrained applications such as IoT devices and low-power sensors [157,158].

6.3. Reduction in memory

In resource-constrained environments, such as IoT devices with limited processing and storage capabilities, efficient memory usage is crucial [117]. To calculate the size of a neural network, we usually focus on the weights and exclude biases for simplicity. The process involves counting the total number of weights by summing up the connections between each layer. For example, in a 3-layer deep neural network for Data 1, you would count the connections between the input layer and the first hidden layer, between the first and second hidden layers, and between the second hidden layer and the output layer. Once you have the total number of weights, determine the size of each weight in bytes, which depends on the data type used (e.g., 32-bit floating-point numbers). Multiply the total number of weights by the size of each weight in bytes to find the total weight size in bytes. Finally, to convert bytes to kilobytes, divide the total size in bytes by 1024. This will give you the weight size of your neural network in kilobytes.

For a 3-layered neural network, considering the connections between these layers, you can calculate the total number of weights:

- Between the input and the first hidden layer: $100 \times 100 = 10000$.
- Between the first and second hidden layer: $100 \times 50 = 5000$.
- Between the second hidden layer and the output layer: $50 \times 1 = 50$.

The total number of weights is $10000 + 5000 + 50 = 15050$.

For traditional 32-bit floating-point numbers (4 bytes), each weight will be 4 bytes. The total size of the weights is 15050 $\times$ 4 bytes, which equals 60 200 bytes or 58.72 KB. This can be broken down as follows: the first layer is 40 000 bytes (39.06 KB), the second layer is 20 000 bytes (19.53 KB), and the last layer is 200 bytes (0.19 KB). Although biases are also present, with a total of 151 biases (100 in the first hidden layer, 50 in the second hidden layer, and 1 in the output layer), their contribution to the overall size is minimal compared to the 15050 weights. Given that each bias is also 4 bytes, the total size of the biases is $151 \times 4 = 604$ bytes or approximately 0.59 KB. Therefore, we can ignore them in this calculation for simplicity.

Similarly, we can calculate the size of each layer for both 16-bit and 8-bit systems and for the LSTM-based deep learning model used with Data 2. The details are summarized in Table 5.

It can be seen from Table 5 moving form 32 bit system to 16 and 8 bit system, we can save 50% and 25% memory.

6.4. Communication cost

Communication cost is calculated as the amount of data transfer between clients and server [80]. Energy efficiency of any system is directly related to amount of transmitted data [40,159]. We are confident that consolidating into a single layer streamlines

Table 5
Sizes of different layers of local model.

		32 bit (KB)	16 bit (KB)	8 bit (KB)
Data 1	Entire NN	58.72	29.36	14.68
	Layer 1	39.06	19.53	9.765
	Layer 2	19.53	9.765	4.8825
	Layer 3	0.19	0.095	0.0475
Data 2	Entire NN	90.5	45.25	22.625
	Layer 1	67.58	33.79	16.895
	Layer 2	16.64	8.32	4.16
	Layer 3	8.32	4.16	2.08
	Layer 4	0.132	0.066	0.033

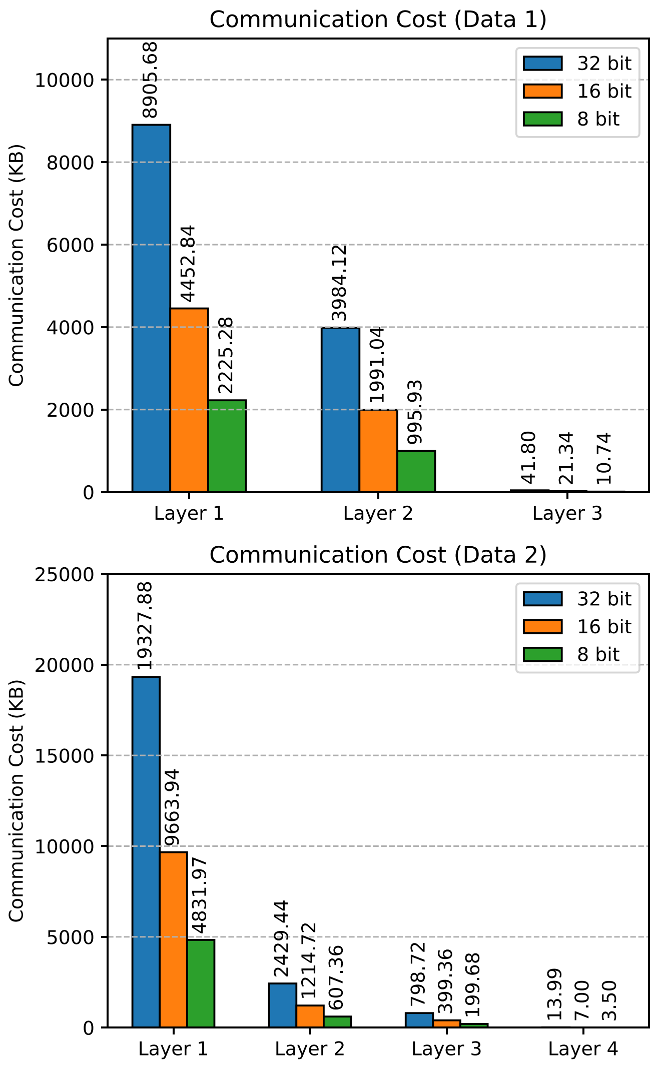

Fig. 10. Communication costs for different layers of a three-layered neural network.

communication, thereby decreasing communication overhead. This reduction in parameters to transfer alleviates the communication load, enabling more efficient utilization of network resources and enhancing scalability.

The communication cost for each device can be quantified by computing the volume of data transferred between the client and server, multiplied by the number of communication rounds. Let us consider the number of communication rounds discussed in Section 5.4. For Data 1 with a three-layer deep network, the rounds were: 114 for Layer 1, 102 for Layer 2, and 110 for Layer 3. For Data 2 with a four-layer deep network, the rounds were: 142 for Layer 1, 72 for Layer 2, 48 for Layer 3, and 52 for Layer 4. The volume of data transfer for each layer is presented graphically in Fig. 10. The choice of which layer to aggregate has a significant impact on communication costs. The percentage improvement in communication cost can be calculated by comparing the first layer with a 32-bit float system to the last layer with an 8-bit fixed system. By selecting the last layer for Data 1 and Data 2, communication costs can be reduced by 829.2 and 5522 times, respectively.

7. Discussion

The experiments conducted demonstrate the significant impact of layer aggregation on communication efficiency and model performance in FLenvironments. For Data 1, aggregating the second layer yielded the optimal MAPE with fewer communication rounds compared to other layers. However, using the last layer resulted in only a negligible increase in the loss function, indicating that it can effectively balance accuracy and communication efficiency. Similarly, for Data 2, while aggregating the third layer required the fewest communication rounds and provided a MAPE comparable to that of the second layer, the last layer also showed an acceptable increase in loss. This suggests a potential trade-off between communication cost and model precision. These findings are consistent with the work presented in [86], where input layers are used for personalized training and the later layers of local models are utilized for model aggregation. However, it is recommended to adopt an adaptive approach when selecting the aggregation layer: if devices have abundant resources, the first layer should be used; otherwise, the optimal layer should be selected, based on experiment results. This approach efficiently addresses the challenge of data heterogeneity in both resource-abundant and resource-constrained devices.

The impact of quantization on FL performance was also examined, revealing that reducing the bit-width of weights from 32 to 8 bits resulted in only a minor increase in MAPE for both datasets. This finding underscores the feasibility of using lower bit-widths to significantly reduce memory usage and communication overhead, especially in resource-constrained environments such as IoT devices. The ability to maintain acceptable accuracy with 8-bit quantization suggests that FL can be effectively implemented on devices with limited computational resources, broadening its applicability.

Furthermore, the analysis of communication costs revealed substantial savings when aggregating later layers and using fixed-point arithmetic. Specifically, the communication cost for the last layer with an 8-bit fixed-point system was significantly lower, resulting in communication cost savings of 829.2-fold for Data 1 and 5522-fold for Data 2 compared to the first layer with a 32-bit floating-point system. This highlights the potential for optimizing FL systems by strategically selecting layers and quantization schemes, thereby improving efficiency while maintaining model accuracy. For optimized results, it is recommended to use the last layer, which provides acceptable results and the lowest communication cost. Our framework, ASLA notably outperforms many state-of-the-art frameworks. For instance, FedKD [80], a communication-efficient framework using knowledge distillation, only improved communication cost by 19-fold. Another framework, FedProto [86], which used prototype learning to handle heterogeneity in FL, improved communication cost by 161.25-fold compared to FedAvg. A framework presented in [160] aimed to reduce communication cost but only achieved a 34% improvement. Similarly, the authors in [161] achieved a 95% improvement in communication cost. SmartIdx [162] achieved a 69.2-fold improvement in communication cost. Lastly, FedPSO [163] employed partial swarm optimization to reduce communication cost, yielding a 55% improvement.

Additionally, our framework is much more simplistic and can be scaled to larger applications with comparatively less complexity. This simplicity not only reduces implementation challenges but also ensures that our approach can be easily adopted and deployed in real-world scenarios, offering substantial benefits in terms of both communication efficiency and operational feasibility.

8. Conclusion

Data heterogeneity in smart grids, arising from diverse device types and varying data quality, presents a significant challenge for conventional FL techniques, especially for resource-constrained devices. To address these challenges, we introduced the Adaptive Single Layer Aggregation (ASLA) framework, specifically designed for resource-constrained smart grid networks. The ASLA framework incorporates three key components: adaptive single-layer aggregation, effective stopping criteria, and weight quantization. In ASLA, aggregation is performed using only a single layer of the neural network, chosen adaptively, regardless of the total number of layers. The stopping criteria efficiently conclude the FL training process when no further improvements are detected and halt server-side training if a substantial percentage of devices stop sending updates. Additionally, weight quantization reduces precision from 32-bit floating-point to 8-bit fixed-point, resulting in an acceptable increase in the loss function (0.01% for Data 1 and 1.25% for Data 2). Our results indicate that using the last layer of local models for aggregation effectively addresses data heterogeneity, while 8-bit quantization significantly reduces communication costs. However, for resource-abundant devices, an optimal layer should be selected based on experimental results. The combination of single-layer aggregation, quantization, and effective stopping criteria makes the ASLA framework highly suitable for resource-constrained devices, achieving communication cost reductions of 829.2-fold for Data 1 and 5522-fold for Data 2, along with a 75% reduction in memory requirements. Thus, the ASLA framework effectively tackles the challenges of data heterogeneity while enhancing both communication and computational efficiency.

CRediT authorship contribution statement

Habib Ullah Manzoor: Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Atif Jafri:** Writing – review & editing, Validation, Formal analysis. **Ahmed Zoha:** Writing – review & editing, Validation, Supervision, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link of used data set in references.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to proofread the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] D. Kaur, S.N. Islam, M.A. Mahmud, M.E. Haque, Z.Y. Dong, Energy forecasting in smart grid systems: recent advancements in probabilistic deep learning, IET Gener. Transm. Distrib. 16 (22) (2022) 4461–4479.
[2] A. Mystakidis, P. Koukaras, N. Tsalikidis, D. Ioannidis, C. Tjortjis, Energy forecasting: A comprehensive review of techniques and technologies, Energies 17 (7) (2024) 1662.
[3] D. Kajela, M.S. Manshahia, Optimization of renewable energy systems: a review, Int. J. Sci. Res. Sci. Technol 3 (8) (2017) 769–795.
[4] H.U. Manzoor, A.R. Khan, M. Al-Quraan, L. Mohjazi, A. Taha, H. Abbas, S. Hussain, M.A. Imran, A. Zoha, Energy management in an agile workspace using AI-driven forecasting and anomaly detection, in: 2022 4th Global Power, Energy and Communication Conference, GPECOM, IEEE, 2022, pp. 644–649.
[5] T.B. Nadeem, M. Siddiqui, M. Khalid, M. Asif, Distributed energy systems: A review of classification, technologies, applications, and policies: Current policy, targets and their achievements in different countries (continued), Energy Strategy Rev. 48 (2023) 101096.
[6] R. Ren21, Global Status Report. 2022, Renewable Energy Policy Network for the 21st Century Paris, France, 2020.
[7] Z.U. Abideen, A. Aslam, H.U. Manzoor, T. Manzoor, N. Bashir, Cost optimization of off grid photovoltaic system by increasing conversion efficiency, in: 2019 International Conference on Electrical, Communication, and Computer Engineering, ICECCE, IEEE, 2019, pp. 1–5.
[8] D. Mhlanga, Artificial intelligence and machine learning for energy consumption and production in emerging markets: A review, Energies 16 (2) (2023) 745.
[9] N.-T. Ngo, A.-D. Pham, T.T.H. Truong, N.-S. Truong, N.-T. Huynh, Developing a hybrid time-series artificial intelligence model to forecast energy use in buildings, Sci. Rep. 12 (1) (2022) 15775.
[10] N.-S. Truong, N.-T. Ngo, A.-D. Pham, Forecasting time-series energy data in buildings using an additive artificial intelligence model for improving energy efficiency, Comput. Intell. Neurosci. 2021 (2021).
[11] S. Bourhnane, M.R. Abid, R. Lghoul, K. Zine-Dine, N. Elkamoun, D. Benhaddou, Machine learning for energy consumption prediction and scheduling in smart buildings, SN Appl. Sci. 2 (2) (2020) 297.
[12] X. Qu, C. Guan, G. Xie, Z. Tian, K. Sood, C. Sun, L. Cui, Personalized federated learning for heterogeneous residential load forecasting, Big Data Min. Anal. 6 (4) (2023) 421–432.
[13] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H.B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1175–1191.
[14] L. Li, Y. Fan, M. Tse, K.-Y. Lin, A review of applications in federated learning, Comput. Ind. Eng. 149 (2020) 106854.
[15] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl.-Based Syst. 216 (2021) 106775.
[16] H.U. Manzoor, S. Hussain, D. Flynn, A. Zoha, Centralised vs. Decentralised federated load forecasting: Who holds the key to adversarial attack robustness? 2024, Authorea Preprints, Authorea.
[17] A. Shabbir, H.U. Manzoor, R.A. Ahmed, Z. Halim, Resilience of federated learning against false data injection attacks in energy forecasting, in: 2024 International Conference on Green Energy, Computing and Sustainable Technology, GECOST, IEEE, 2024, pp. 245–249.
[18] Y. Huang, Y. Song, Z. Jing, Load forecasting using federated learning with considering electricity data privacy preservation of EASP, Ain Shams Eng. J. (2024) 102724.
[19] C. Wang, J. Song, D. Shi, J.L. Reyna, H. Horsey, S. Feron, Y. Zhou, Z. Ouyang, Y. Li, R.B. Jackson, Impacts of climate change, population growth, and power sector decarbonization on urban building energy use, Nature Commun. 14 (1) (2023) 1–16.
[20] N. Balta-Ozkan, J. Yildirim, P.M. Connor, I. Truckell, P. Hart, Energy transition at local level: Analyzing the role of peer effects and socio-economic factors on UK solar photovoltaic deployment, Energy Policy 148 (2021) 112004.
[21] H.-B. Chen, L.-L. Pei, Y.-F. Zhao, Forecasting seasonal variations in electricity consumption and electricity usage efficiency of industrial sectors using a grey modeling approach, Energy 222 (2021) 119952.
[22] R. Joseph, M. Ting, P. Kumar, Multiple-scale spatio–temporal variability of precipitation over the coterminous United States, J. Hydrometeorol. 1 (5) (2000) 373–392.
[23] W. Zhao, T. Li, D. Xu, Z. Wang, A global forecasting method of heterogeneous household short-term load based on pre-trained autoencoder and deep-LSTM model, Ann. Oper. Res. (2022) 1–33.
[24] O.R.A. Almanifi, C.-O. Chow, M.-L. Tham, J.H. Chuah, J. Kanesan, Communication and computation efficiency in federated learning: A survey, Internet Things 22 (2023) 100742.
[25] J. Wang, F. Gao, Y. Zhou, Q. Guo, C.-W. Tan, J. Song, Y. Wang, Data sharing in energy systems, Adv. Appl. Energy 10 (2023) 100132.
[26] B.P. Bhattarai, S. Paudyal, Y. Luo, M. Mohanpurkar, K. Cheung, R. Tonkoski, R. Hovsapian, K.S. Myers, R. Zhang, P. Zhao, et al., Big data analytics in smart grids: state-of-the-art, challenges, opportunities, and future directions, IET Smart Grid 2 (2) (2019) 141–154.
[27] M.H. Zafar, S.M.S. Bukhari, M. Abou Houran, S.K.R. Moosavi, M. Mansoor, N. Al-Tawalbeh, F. Sanfilippo, Step towards secure and reliable smart grids in Industry 5.0: A federated learning assisted hybrid deep learning model for electricity theft detection using smart meters, Energy Rep. 10 (2023) 3001–3019.
[28] C. Chen, T. Liao, X. Deng, Z. Wu, S. Huang, Z. Zheng, Advances in robust federated learning: Heterogeneity considerations, 2024, arXiv preprint arXiv:2405.09839.
[29] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-iid data with reinforcement learning, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 1698–1707.
[30] X. Ma, J. Zhu, Z. Lin, S. Chen, Y. Qin, A state-of-the-art survey on solving non-iid data in federated learning, Future Gener. Comput. Syst. 135 (2022) 244–258.

[31] J. Zhang, Z. Li, B. Li, J. Xu, S. Wu, S. Ding, C. Wu, Federated learning with label distribution skew via logits calibration, in: International Conference on Machine Learning, PMLR, 2022, pp. 26311–26329.
[32] Z. Luo, Y. Wang, Z. Wang, Z. Sun, T. Tan, Disentangled federated learning for tackling attributes skew via invariant aggregation and diversity transferring, 2022, arXiv preprint arXiv:2206.06818.
[33] T. Zhou, J. Zhang, D.H. Tsang, FedFA: Federated learning with feature anchors to align features and classifiers for heterogeneous data, IEEE Trans. Mob. Comput. (2023).
[34] S. Yang, H. Park, J. Byun, C. Kim, Robust federated learning with noisy labels, IEEE Intell. Syst. 37 (2) (2022) 35–43.
[35] J. Xu, Z. Chen, T.Q. Quek, K.F.E. Chong, Fedcorr: Multi-stage federated learning for label noise correction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10184–10193.
[36] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, Y. Cheng, Tifl: A tier-based federated learning system, in: Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing, 2020, pp. 125–136.
[37] Q. Li, Y. Diao, Q. Chen, B. He, Federated learning on non-iid data silos: An experimental study, in: 2022 IEEE 38th International Conference on Data Engineering, ICDE, IEEE, 2022, pp. 965–978.
[38] P. Labonne, Asymmetric uncertainty: Nowcasting using skewness in real-time data, Int. J. Forecast. (2024).
[39] H.S. Oliveira, H.P. Oliveira, Transformers for energy forecast, Sensors 23 (15) (2023) 6840.
[40] H.U. Manzoor, A.R. Khan, D. Flynn, M.M. Alam, M. Akram, M.A. Imran, A. Zoha, FedBranched: Leveraging federated learning for anomaly-aware load forecasting in energy networks, Sensors 23 (7) (2023) 3570.
[41] D. Lin, Y. Guo, H. Sun, Y. Chen, Fedcluster: A federated learning framework for cross-device private ecg classification, in: IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, IEEE, 2022, pp. 1–6.
[42] Z. Chen, D. Li, R. Ni, J. Zhu, S. Zhang, FedSeq: A hybrid federated learning framework based on sequential in-cluster training, IEEE Syst. J. (2023).
[43] J. Zhang, S. Lv, Fedlabcluster: A clustered federated learning algorithm based on data sample label, in: 2021 International Conference on Electronic Information Engineering and Computer Science, EIECS, IEEE, 2021, pp. 423–428.
[44] H. Lee, D. Seo, FedLC: Optimizing federated learning in non-IID data via label-wise clustering, IEEE Access (2023).
[45] C. You, Z. Lu, J. Wang, C. Yan, FedDDB: Clustered federated learning based on data distribution difference, in: Proceedings of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence, 2022, pp. 1–6.
[46] C. Li, G. Li, P.K. Varshney, Federated learning with soft clustering, IEEE Internet Things J. 9 (10) (2021) 7773–7782.
[47] F. Sattler, K.-R. Müller, W. Samek, Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints, IEEE Trans. Neural Netw. Learn. Syst. 32 (8) (2020) 3710–3722.
[48] Y. Kim, E. Al Hakim, J. Haraldson, H. Eriksson, J.M.B. da Silva, C. Fischione, Dynamic clustering in federated learning, in: ICC 2021-IEEE International Conference on Communications, IEEE, 2021, pp. 1–6.
[49] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, B. Caputo, Cluster-driven graph federated learning over multiple domains, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2749–2758.
[50] X. Ouyang, Z. Xie, J. Zhou, G. Xing, J. Huang, ClusterFL: A clustering-based federated learning system for human activity recognition, ACM Trans. Sensor Netw. 19 (1) (2022) 1–32.
[51] Y. Ruan, C. Joe-Wong, Fedsoft: Soft clustered federated learning with proximal local updating, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, No. 7, 2022, pp. 8124–8131.
[52] L.U. Khan, Z. Han, D. Niyato, C.S. Hong, Socially-aware-clustering-enabled federated learning for edge networks, IEEE Trans. Netw. Serv. Manag. 18 (3) (2021) 2641–2658.
[53] Y. Fraboni, R. Vidal, L. Kameni, M. Lorenzi, Clustered sampling: Low-variance and improved representativity for clients selection in federated learning, in: International Conference on Machine Learning, PMLR, 2021, pp. 3407–3416.
[54] A. Taik, Z. Mlika, S. Cherkaoui, Clustered vehicular federated learning: Process and optimization, IEEE Trans. Intell. Transp. Syst. 23 (12) (2022) 25371–25383.
[55] M. Duan, D. Liu, X. Ji, R. Liu, L. Liang, X. Chen, Y. Tan, Fedgroup: Efficient federated learning via decomposed similarity-based clustering, in: 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, ISPA/BDCloud/SocialCom/SustainCom, IEEE, 2021, pp. 228–237.
[56] M. Duan, D. Liu, X. Ji, Y. Wu, L. Liang, X. Chen, Y. Tan, A. Ren, Flexible clustered federated learning for client-level data distribution shift, IEEE Trans. Parallel Distrib. Syst. 33 (11) (2021) 2661–2674.
[57] G. Luo, N. Chen, J. He, B. Jin, Z. Zhang, Y. Li, Privacy-preserving clustering federated learning for non-IID data, Future Gener. Comput. Syst. 154 (2024) 384–395.
[58] P. Tian, W. Liao, W. Yu, E. Blasch, WSCC: A weight-similarity-based client clustering approach for non-IID federated learning, IEEE Internet Things J. 9 (20) (2022) 20243–20256.
[59] D.K. Dennis, T. Li, V. Smith, Heterogeneity for the win: One-shot federated clustering, in: International Conference on Machine Learning, PMLR, 2021, pp. 2611–2620.
[60] M. Ahmadi, A. Taghavirashidizadeh, D. Javaheri, A. Masoumian, S.J. Ghoushchi, Y. Pourasad, DQRE-SCnet: a novel hybrid approach for selecting users in federated learning with deep-Q-reinforcement learning based on spectral clustering, J. King Saud Univ. Comput. Inf. Sci. 34 (9) (2022) 7445–7458.
[61] L. Cui, X. Su, Y. Zhou, L. Zhang, ClusterGrad: Adaptive gradient compression by clustering in federated learning, in: GLOBECOM 2020-2020 IEEE Global Communications Conference, IEEE, 2020, pp. 1–7.
[62] B. Gong, T. Xing, Z. Liu, W. Xi, X. Chen, Adaptive client clustering for efficient federated learning over non-iid and imbalanced data, IEEE Trans. Big Data (2022).
[63] M. Morafah, S. Vahidian, W. Wang, B. Lin, FLIS: Clustered federated learning via inference similarity for non-IID data distribution, IEEE Open J. Comput. Soc. 4 (2023) 109–120.
[64] H. Huang, W. Shi, Y. Feng, C. Niu, G. Cheng, J. Huang, Z. Liu, Active client selection for clustered federated learning, IEEE Trans. Neural Netw. Learn. Syst. (2023).
[65] Z. He, L. Wang, Z. Cai, Clustered federated learning with adaptive local differential privacy on heterogeneous iot data, IEEE Internet Things J. (2023).
[66] L. Cai, N. Chen, Y. Cao, J. He, Y. Li, FedCE: Personalized federated learning method based on clustering ensembles, in: Proceedings of the 31st ACM International Conference on Multimedia, 2023, pp. 1625–1633.
[67] A.A. Al-Saedi, V. Boeva, E. Casalicchio, Fedco: Communication-efficient federated learning via clustering optimization, Future Internet 14 (12) (2022) 377.
[68] W. Pedrycz, Federated FCM: Clustering under privacy requirements, IEEE Trans. Fuzzy Syst. 30 (8) (2021) 3384–3388.
[69] Z. Chen, P. Tian, W. Liao, W. Yu, Zero knowledge clustering based adversarial mitigation in heterogeneous federated learning, IEEE Trans. Netw. Sci. Eng. 8 (2) (2020) 1070–1083.
[70] S. Agrawal, S. Sarkar, M. Alazab, P.K.R. Maddikunta, T.R. Gadekallu, Q.-V. Pham, et al., Genetic CFL: Hyperparameter optimization in clustered federated learning, Comput. Intell. Neurosci. 2021 (2021).
[71] C. Feng, H.H. Yang, D. Hu, Z. Zhao, T.Q. Quek, G. Min, Mobility-aware cluster federated learning in hierarchical wireless networks, IEEE Trans. Wireless Commun. 21 (10) (2022) 8441–8458.

[72] H.U. Manzoor, A.R. Khan, T. Sher, W. Ahmad, A. Zoha, Defending federated learning from backdoor attacks: Anomaly-aware fedavg with layer-based aggregation, in: 2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, IEEE, 2023, pp. 1–6.
[73] H.U. Manzoor, M.S. Khan, A.R. Khan, F. Ayaz, D. Flynn, M.A. Imran, A. Zoha, FedClamp: An algorithm for identification of anomalous client in federated learning, in: 2022 29th IEEE International Conference on Electronics, Circuits and Systems, ICECS, IEEE, 2022, pp. 1–4.
[74] M. Yan, L. Wang, X. Wang, L. Li, L. Xu, A. Fei, Matching theory aided federated learning method for load forecasting of virtual power plant, in: 2021 17th International Conference on Mobility, Sensing and Networking, MSN, IEEE, 2021, pp. 327–333.
[75] Y. Shi, X. Xu, Deep federated adaptation: An adaptative residential load forecasting approach with federated learning, Sensors 22 (9) (2022) 3264.
[76] M.G. Arivazhagan, V. Aggarwal, A.K. Singh, S. Choudhary, Federated learning with personalization layers, 2019, arXiv preprint arXiv:1912.00818.
[77] L. Collins, H. Hassani, A. Mokhtari, S. Shakkottai, Exploiting shared representations for personalized federated learning, in: International Conference on Machine Learning, PMLR, 2021, pp. 2089–2099.
[78] H. Seo, J. Park, S. Oh, M. Bennis, S.-L. Kim, 16 Federated knowledge distillation, Mach. Learn. Wirel. Commun. 457 (2022).
[79] G. Yang, H. Tae, Federated distillation methodology for label-based group structures, Appl. Sci. 14 (1) (2023) 277.
[80] C. Wu, F. Wu, L. Lyu, Y. Huang, X. Xie, Communication-efficient federated learning via knowledge distillation, Nature Commun. 13 (1) (2022) 2032.
[81] E. Tanghatari, M. Kamal, A. Afzali-Kusha, M. Pedram, Federated learning by employing knowledge distillation on edge devices with limited hardware resources, Neurocomputing 531 (2023) 87–99.
[82] T. Lin, L. Kong, S.U. Stich, M. Jaggi, Ensemble distillation for robust model fusion in federated learning, Adv. Neural Inf. Process. Syst. 33 (2020) 2351–2363.
[83] A. Mora, I. Tenison, P. Bellavista, I. Rish, Knowledge distillation for federated learning: a practical guide, 2022, arXiv preprint arXiv:2211.04742.
[84] Y. Qiao, C. Zhang, H.Q. Le, A.D. Raha, A. Adhikary, C.S. Hong, Knowledge distillation in federated learning: Where and how to distill? in: 2023 24st Asia-Pacific Network Operations and Management Symposium, APNOMS, IEEE, 2023, pp. 18–23.
[85] S. Wu, J. Chen, X. Nie, Y. Wang, X. Zhou, L. Lu, W. Peng, Y. Nie, W. Menhaj, Global prototype distillation for heterogeneous federated learning, Sci. Rep. 14 (1) (2024) 12057.
[86] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, C. Zhang, Fedproto: Federated prototype learning across heterogeneous clients, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, No. 8, 2022, pp. 8432–8440.
[87] B. Yan, H. Zhang, M. Xu, D. Yu, X. Cheng, FedRFQ: Prototype-based federated learning with reduced redundancy, minimal failure, and enhanced quality, IEEE Trans. Comput. (2024).
[88] X. Zhu, J. Wang, W. Chen, K. Sato, Model compression and privacy preserving framework for federated learning, Future Gener. Comput. Syst. 140 (2023) 376–389.
[89] J. Cao, R. Wei, Q. Cao, Y. Zheng, Z. Zhu, C. Ji, X. Zhou, FedStar: Efficient federated learning on heterogeneous communication networks, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2023).
[90] S. Huang, Z. Zhang, S. Wang, R. Wang, K. Huang, Accelerating federated edge learning via topology optimization, IEEE Internet Things J. 10 (3) (2022) 2056–2070.
[91] J. Wu, F. Dong, H. Leung, Z. Zhu, J. Zhou, S. Drew, Topology-aware federated learning in edge computing: A comprehensive survey, ACM Comput. Surv. 56 (10) (2024) 1–41.
[92] Y. Cheng, D. Wang, P. Zhou, T. Zhang, Model compression and acceleration for deep neural networks: The principles, progress, and challenges, IEEE Signal Process. Mag. 35 (1) (2018) 126–136.
[93] Y. Jiang, S. Wang, V. Valls, B.J. Ko, W.-H. Lee, K.K. Leung, L. Tassiulas, Model pruning enables efficient federated learning on edge devices, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[94] B. Wang, J. Fang, H. Li, B. Zeng, Communication-efficient federated learning: A variance-reduced stochastic approach with adaptive sparsification, IEEE Trans. Signal Process. (2023).
[95] S. Chen, C. Shen, L. Zhang, Y. Tang, Dynamic aggregation for heterogeneous quantization in federated learning, IEEE Trans. Wireless Commun. 20 (10) (2021) 6804–6819.
[96] H. Huang, L. Zhang, C. Sun, R. Fang, X. Yuan, D. Wu, Distributed pruning towards tiny neural networks in federated learning, in: 2023 IEEE 43rd International Conference on Distributed Computing Systems, ICDCS, IEEE, 2023, pp. 190–201.
[97] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, A. Peste, Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, J. Mach. Learn. Res. 22 (1) (2021) 10882–11005.
[98] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, S. Han, On-device training under 256kb memory, Adv. Neural Inf. Process. Syst. 35 (2022) 22941–22954.
[99] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2015, arXiv preprint arXiv:1510.00149.
[100] D. Molchanov, A. Ashukha, D. Vetrov, Variational dropout sparsifies deep neural networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 2498–2507.
[101] Z. Wang, Sparsert: Accelerating unstructured sparsity on gpus for deep learning inference, in: Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques, 2020, pp. 31–42.
[102] O. Zachariadis, N. Satpute, J. Gómez-Luna, J. Olivares, Accelerating sparse matrix–matrix multiplication with GPU Tensor Cores, Comput. Electr. Eng. 88 (2020) 106848.
[103] C. Hong, A. Sukumaran-Rajam, B. Bandyopadhyay, J. Kim, S.E. Kurt, I. Nisa, S. Sabhlok, Ü.V. Çatalyürek, S. Parthasarathy, P. Sadayappan, Efficient sparse-matrix multi-vector product on gpus, in: Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing, 2018, pp. 66–79.
[104] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, 2018, arXiv preprint arXiv:1808.06866.
[105] J.-H. Luo, J. Wu, W. Lin, Thinet: A filter level pruning method for deep neural network compression, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5058–5066.
[106] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, J. Kautz, Importance estimation for neural network pruning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11264–11272.
[107] L. Lei, Y. Yuan, Y. Yang, Y. Luo, L. Pu, S. Chatzinotas, Sparsification and optimization for energy-efficient federated learning in wireless edge networks, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE, 2022, pp. 3071–3076.
[108] D. Stripelis, U. Gupta, G.V. Steeg, J.L. Ambite, Federated progressive sparsification (purge, merge, tune)+, 2022, arXiv preprint arXiv:2204.12430.
[109] Y. Lin, S. Han, H. Mao, Y. Wang, W.J. Dally, Deep gradient compression: Reducing the communication bandwidth for distributed training, 2017, arXiv preprint arXiv:1712.01887.
[110] F. Sattler, S. Wiedemann, K.-R. Müller, W. Samek, Sparse binary compression: Towards distributed deep learning with minimal communication, in: 2019 International Joint Conference on Neural Networks, IJCNN, IEEE, 2019, pp. 1–8.
[111] Y. Tsuzuku, H. Imachi, T. Akiba, Variance-based gradient compression for efficient distributed deep learning, 2018, arXiv preprint arXiv:1802.06058.
[112] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient convnets, 2016, arXiv preprint arXiv:1608.08710.
[113] A. M Abdelmoniem, A. Elzanaty, M.-S. Alouini, M. Canini, An efficient statistical-based gradient compression technique for distributed training systems, Proc. Mach. Learn. Syst. 3 (2021) 297–322.
[114] H. Sun, X. Ma, R.Q. Hu, Adaptive federated learning with gradient compression in uplink NOMA, IEEE Trans. Veh. Technol. 69 (12) (2020) 16325–16329.

[115] B. Guo, Y. Liu, C. Zhang, A partition based gradient compression algorithm for distributed training in aiot, Sensors 21 (6) (2021) 1943.
[116] H. Jin, D. Wu, S. Zhang, X. Zou, S. Jin, D. Tao, Q. Liao, W. Xia, Design of a quantization-based dnn delta compression framework for model snapshots and federated learning, IEEE Trans. Parallel Distrib. Syst. 34 (3) (2023) 923–937.
[117] F. Pereira, R. Correia, P. Pinho, S.I. Lopes, N.B. Carvalho, Challenges in resource-constrained IoT devices: Energy and communication as critical success factors for future IoT deployment, Sensors 20 (22) (2020) 6420.
[118] A. Kuzmin, M. Nagel, M. Van Baalen, A. Behboodi, T. Blankevoort, Pruning vs quantization: Which is better? Adv. Neural Inf. Process. Syst. 36 (2024).
[119] Z. Lian, J. Cao, Y. Zuo, W. Liu, Z. Zhu, AGQFL: communication-efficient federated learning via automatic gradient quantization in edge heterogeneous systems, in: 2021 IEEE 39th International Conference on Computer Design, ICCD, IEEE, 2021, pp. 551–558.
[120] P.R. Antony, A.M. Joseph, Design and implementation of double precision floating point comparator, Procedia Technol. 25 (2016) 528–535.
[121] IEEE Computer Society Standards Committee. Working group of the Microprocessor Standards Subcommittee and American National Standards Institute, IEEE Standard for Binary Floating-Point Arithmetic, vol. 754, IEEE, 1985.
[122] C. Chung-Kuan, Computer Arithmetic Algorithms and Hardware Design, in: Lecture notes, 2006.
[123] O. Sentieys, D. Menard, Customizing number representation and precision, in: Approximate Computing Techniques: From Component-to Application-Level, Springer, 2022, pp. 11–41.
[124] D. Zoni, A. Galimberti, Cost-effective fixed-point hardware support for RISC-V embedded systems, J. Syst. Archit. 126 (2022) 102476.
[125] M.T. Lê, P. Wolinski, J. Arbel, Efficient neural networks for tiny machine learning: A comprehensive review, 2023, arXiv preprint arXiv:2311.11883.
[126] S. Branco, A.G. Ferreira, J. Cabral, Machine learning in resource-scarce embedded systems, FPGAs, and end-devices: A survey, Electronics 8 (11) (2019) 1289.
[127] Z. Niu, H. Dong, A.K. Qin, T. Gu, Flrce: Resource-efficient federated learning with early-stopping strategy, 2024, arXiv:2310.09789.
[128] F. Dehrouyeh, L. Yang, F.B. Ajaei, A. Shami, On TinyML and cybersecurity: Electric vehicle charging infrastructure use case, 2024, arXiv preprint arXiv:2404.16894.
[129] Y. Wang, L. Lin, J. Chen, Communication-efficient adaptive federated learning, in: International Conference on Machine Learning, PMLR, 2022, pp. 22802–22838.
[130] J.L. Kim, M.T. Toghani, C.A. Uribe, A. Kyrillidis, Adaptive federated learning with auto-tuned clients, 2023, arXiv preprint arXiv:2306.11201.
[131] D. Deng, X. Wu, T. Zhang, X. Tang, H. Du, J. Kang, J. Liu, D. Niyato, FedASA: A personalized federated learning with adaptive model aggregation for heterogeneous mobile edge computing, IEEE Trans. Mob. Comput. (2024).
[132] Z. Chen, H. Cui, E. Wu, X. Yu, Computation and communication efficient adaptive federated optimization of federated learning for Internet of Things, Electronics 12 (16) (2023) 3451.
[133] S. Wang, T. Tuor, T. Salonidis, K.K. Leung, C. Makaya, T. He, K. Chan, Adaptive federated learning in resource constrained edge computing systems, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1205–1221.
[134] F. Ilhan, G. Su, L. Liu, Scalefl: Resource-adaptive federated learning with heterogeneous clients, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 24532–24541.
[135] W. Liu, J. Geng, Z. Zhu, Y. Zhao, C. Ji, C. Li, Z. Lian, X. Zhou, Ace-sniper: Cloud-edge collaborative scheduling framework with dnn inference latency modeling on heterogeneous devices, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2023).
[136] W. Pan, Z. Xu, S. Rajendran, F. Wang, An adaptive federated learning framework for clinical risk prediction with electronic health records from multiple hospitals, Patterns 5 (1) (2024).
[137] W. Deng, X. Chen, X. Li, H. Zhao, Adaptive federated learning with negative inner product aggregation, IEEE Internet Things J. (2023).
[138] J. Han, A.F. Khan, S. Zawad, A. Anwar, N.B. Angel, Y. Zhou, F. Yan, A.R. Butt, Heterogeneity-aware adaptive federated learning scheduling, in: 2022 IEEE International Conference on Big Data, Big Data, IEEE, 2022, pp. 911–920.
[139] Y. Li, X. Qin, K. Han, N. Ma, X. Xu, P. Zhang, Accelerating wireless federated learning with adaptive scheduling over heterogeneous devices, IEEE Internet Things J. (2023).
[140] Y. Zeng, Y. Mu, J. Yuan, S. Teng, J. Zhang, J. Wan, Y. Ren, Y. Zhang, Adaptive federated learning with non-IID data, Comput. J. 66 (11) (2023) 2758–2772.
[141] H.U. Manzoor, K. Arshad, K. Assaleh, A. Zoha, Enhanced adversarial attack resilience in energy networks through energy and privacy aware federated learning, 2024.
[142] R. Mulla, Hourly energy consumption, 2022, https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption. (Accessed 8 August 2022), [Online]. Available: https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption.
[143] Ausgrid, Ausgrid distribution zone substation data FY23, 2024, https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Distribution-zone-substation-data. (Accessed 22 July 2024), [Online]. Available: https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Distribution-zone-substation-data.
[144] W.-y. Loh, Some modifications of Levene's test of variance homogeneity, J. Stat. Comput. Simul. 28 (3) (1987) 213–226.
[145] H. Arsham, M. Lovric, Bartlett's test, in: M. Lovric (Ed.), International Encyclopedia of Statistical Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 87–88, http://dx.doi.org/10.1007/978-3-642-04898-2_132.
[146] Y. Zhou, Q. Ye, J. Lv, Communication-efficient federated learning with compensated overlap-fedavg, IEEE Trans. Parallel Distrib. Syst. 33 (1) (2021) 192–205.
[147] B. Pang, E. Nijkamp, Y.N. Wu, Deep learning with tensorflow: A review, J. Educ. Behav. Stat. 45 (2) (2020) 227–248.
[148] L. Zhu, Z. Liu, S. Han, Deep leakage from gradients, Adv. Neural Inf. Process. Syst. 32 (2019).
[149] M. Nasr, R. Shokri, A. Houmansadr, Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning, in: 2019 IEEE Symposium on Security and Privacy, SP, IEEE, 2019, pp. 739–753.
[150] S. Yeom, I. Giacomelli, M. Fredrikson, S. Jha, Privacy risk in machine learning: Analyzing the connection to overfitting, in: 2018 IEEE 31st Computer Security Foundations Symposium, CSF, IEEE, 2018, pp. 268–282.
[151] D. Agrawal, D. Kesdogan, Measuring anonymity: The disclosure attack, IEEE Secur. Priv. 1 (6) (2003) 27–34.
[152] R. Boussada, M.E. Elhdhili, L.A. Saidane, A survey on privacy: Terminology, mechanisms and attacks, in: 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications, AICCSA, IEEE, 2016, pp. 1–7.
[153] J.W. Woods, Multidimensional Signal, Image, and Video Processing and Coding, Elsevier, 2006.
[154] M. Junaid, H. Aliev, S. Park, H. Kim, H. Yoo, S. Sim, Hybrid precision floating-point (HPFP) selection to optimize hardware-constrained accelerator for CNN training, Sensors 24 (7) (2024) 2145.
[155] J. Multanen, H. Kultala, K. Tervo, P. Jääskeläinen, Energy efficient low latency multi-issue cores for intelligent always-on IoT applications, J. Signal Process. Syst. 92 (10) (2020) 1057–1073.
[156] D. Kerr-Munslow, Advantages and pitfalls of moving from an 8 bit system to 32 bit architectures, in: Embedded Real Time Software and Systems (ERTS2010), 2010.
[157] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B.C. Lee, S. Richardson, C. Kozyrakis, M. Horowitz, Understanding sources of ineffciency in general-purpose chips, Commun. ACM 54 (10) (2011) 85–93.
[158] A. Waterman, Y. Lee, D. Patterson, K. Asanovic, Volume I User level Isa, A. Waterman, Y. Lee, D. Patterson, The RISC-v instruction set manual, 2, 2014, pp. 1–79, Volume I: User-Level ISA', version 2.

[159] A.N. Mian, S.W.H. Shah, S. Manzoor, A. Said, K. Heimerl, J. Crowcroft, A value-added IoT service for cellular networks using federated learning, Comput. Netw. 213 (2022) 109094.
[160] G. Paragliola, A. Coronato, Definition of a novel federated learning approach to reduce communication costs, Expert Syst. Appl. 189 (2022) 116109.
[161] G. Paragliola, Evaluation of the trade-off between performance and communication costs in federated learning scenario, Future Gener. Comput. Syst. 136 (2022) 282–293.
[162] D. Wu, X. Zou, S. Zhang, H. Jin, W. Xia, B. Fang, Smartidx: Reducing communication cost in federated learning by exploiting the cnns structures, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, No. 4, 2022, pp. 4254–4262.
[163] S. Park, Y. Suh, J. Lee, FedPSO: Federated learning using particle swarm optimization to reduce communication costs, Sensors 21 (2) (2021) 600.