AttFL: A Personalized Federated Learning Framework for Time-series Mobile and Embedded Sensor Data Processing

JAEYEON PARK, School of Integrated Technology, College of Computing, Yonsei University, South Korea
KICHANG LEE, School of Integrated Technology, College of Computing, Yonsei University, South Korea
SUNGMIN LEE, School of Integrated Technology, College of Computing, Yonsei University, South Korea
MI ZHANG, Department of Computer Science and Engineering, The Ohio State University, United States
JEONGGIL KO*, School of Integrated Technology, College of Computing, Yonsei University, South Korea

This work presents *AttFL*, a federated learning framework designed to continuously improve a personalized deep neural network for efficiently analyzing time-series data generated from mobile and embedded sensing applications. To better characterize time-series data features and efficiently abstract model parameters, *AttFL* appends a set of attention modules to the baseline deep learning model and exchanges their feature map information to gather collective knowledge across distributed local devices at the server. The server groups devices with similar contextual goals using cosine similarity, and re-distributes updated model parameters for improved inference performance at each local device. Specifically, unlike previously proposed federated learning frameworks, *AttFL* is designed specifically to perform well for various recurrent neural network (RNN) baseline models, making it suitable for many mobile and embedded sensing applications producing time-series sensing data. We evaluate the performance of *AttFL* and compare with five state-of-the-art federated learning frameworks using three popular mobile/embedded sensing applications (e.g., physiological signal analysis, human activity recognition, and audio processing). Our results obtained from CPU core-based emulations and a 12-node embedded platform testbed shows that *AttFL* outperforms all alternative approaches in terms of model accuracy and communication/computational overhead, and is flexible enough to be applied in various application scenarios exploiting different baseline deep learning model architectures.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; • **Computing methodologies → Machine learning algorithms**.

Additional Key Words and Phrases: Personalized federated learning; Mobile systems; Embedded sensing systems

ACM Reference Format:
JaeYeon Park, Kichang Lee, Sungmin Lee, Mi Zhang, and JeongGil Ko. 2023. AttFL: A Personalized Federated Learning Framework for Time-series Mobile and Embedded Sensor Data Processing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 3, Article 116 (September 2023), 31 pages. https://doi.org/10.1145/3610917

*Corresponding Author: jeonggil.ko@yonsei.ac.kr

Authors' addresses: JaeYeon Park, School of Integrated Technology, College of Computing, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul, South Korea; Kichang Lee, School of Integrated Technology, College of Computing, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul, South Korea; Sungmin Lee, School of Integrated Technology, College of Computing, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul, South Korea; Mi Zhang, Department of Computer Science and Engineering, The Ohio State University, 2015 Neil Ave, Columbus, OH, United States; JeongGil Ko, School of Integrated Technology, College of Computing, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul, South Korea.

2474-9567/2023/9-ART116 $15.00
https://doi.org/10.1145/3610917

1 INTRODUCTION

Mobile and embedded sensing applications gather data at high fidelity and diverse dimensions to capture detailed personal and environmental events. The recent trend of integrating deep learning algorithms on sensing devices has catalyzed the design of many mobile/embedded sensing applications as it suppresses sensitive data from leaving the local devices [28, 53, 55, 65, 84]. Still, the models used in these applications are generic, which (in most cases) are not tailored to the data currently being experienced at the device. As the next step of further improving sensing application performance, system designers are now examining the possibilities of performing some level of *local model training* [26, 83]. Despite the computing resource constraints of embedded and mobile sensing platforms, locally training deep learning models (even a small amount) can accelerate the design of more personalized and customized services [87].

Among various ways of training models locally on the devices, federated learning offers an option to train a deep learning model based on local data stored in geographically distributed devices without sending the data to a central server [33, 39, 78]. Specifically, devices participating in a federated learning process train their local models using self-collected data and share model parameters with a server. With model parameters collected from various devices, the server applies an aggregation mechanism to integrate the collected model parameters and trains a deep learning model without exposing individual device's local data. By re-distributing the updated model parameters, local devices can improve their model performance compared to the case of training with only locally collected data. As a result, federated learning offers a way to collaboratively achieve an improved model performance without sharing local data.

Concurrently, advances in low-power sensing technology have enabled a rich body of applications that exploit various types of complex sensor data [46, 66, 69]. Given that the purpose of many sensing applications is to capture and understand longitudinal information of the user or the environment, in many cases, their sensing outputs are in the form of *time-series data streams*. Hence, a federated learning framework for mobile and embedded sensing applications should be able to train deep learning models with time-series data streams.

Unfortunately, state-of-the-art federated learning frameworks are not designed to support complex models for time-series sensor data analysis. Our motivational studies (Sec. 3) show that even the most recently proposed frameworks such as FedMask [45] and FedDL [76] show inefficiencies or perform poorly when the baseline network becomes complex or a recurrent neural network (RNN) baseline model is used. Such limitations are mainly due to the optimization approaches of these existing frameworks. By design, to minimize data exchange, federated learning frameworks select portions of the model to share with the server. Ideally, this sub-model should be small but informative so the server can learn knowledge about the local data features of individual devices while reducing the communication overhead of the federated training process. FedMask, for example, shares a masked result of model parameters and FedDL exploits dynamic layer sharing in an iterative layer-wise manner. Unfortunately, when applied to deep models that suit time-series data analysis, both fail to offer competitive performance given that they are either limited to sparse models or due to operational characteristics of computing across all model layers; thus, are not scalable to deep/dense models.

To address the limitations of the state-of-the-arts, in this work, we propose *AttFL*[1], an efficient federated learning framework for time-series sensor data to support a wide range of mobile and embedded sensing applications. Compared to previous approaches, *AttFL* requires no modifications to the baseline deep learning model structure while only appending three attention layers, which focus the underlying model towards the local, sub-global, and global features of the input data. Local devices participating in *AttFL* fully exploit the temporal details of the input data using this baseline+attention architecture and share two main types of information with the server for federated learning operations: (i) attention feature maps and (ii) the convolution layers weights used

[1] An open source implementation of *AttFL* is available at https://github.com/eis-lab/attfl

Table 1. Comparison of proposed *AttFL* with representative existing federated learning frameworks.

	Computation improvement	Communication improvement	Model personalization	Time-series data support		Core features
				RNN baseline model validation	Scalable with increasing layers	
FedAvg [62]	X	X	X	X	X	Weight averaging
PerFed [21]	X	X	✓	X	X	Model-Agnostic Meta-Learning (MAML)
pFedAtt [59]	X	X	✓	X	X	Attentive functions for model personalization
pFedMe [73]	X	X	✓	X	X	Moreau envelope-based regularization loss function
LG-FedAvg [50]	X	✓	✓	X	X	Leveraging personalization and base layers
FedMask [45]	✓	✓	✓	△	X	Pruning-based sparsification and mask sharing
FedDL [76]	✓	✓	✓	X	X	Layer-wise sharing and affinity-based grouping
AttFL (Ours)	✓	✓	✓	✓	✓	Attachable attention modules

for inter-connecting the attention modules with the baseline DNN. These two types of information effectively abstract the underlying baseline model's weights/parameters in a light-weight manner (only 1.37 MB of data exchange needed for a Bi-LSTM baseline model) so that the server can understand which devices share similarities in input data characteristics. For this, the *AttFL* server exploits attention feature map-based cosine similarity to group local devices so that their model configuration information can be aggregated and re-distributed.

We evaluate the performance of *AttFL* using three representative mobile and embedded sensing applications – human activity recognition, physiological data analysis, and audio data processing – all based on time-series data. To do so, we have designed and implemented a testbed using 12 heterogeneous embedded platforms. Our results show that *AttFL* consistently outperforms state-of-the-arts in both model accuracy as well as computation and communication efficiency for mobile and embedded sensing applications. We also show that *AttFL* can be used for diverse deep learning model architectures with minimal computational burden on mobile/embedded devices.

Specifically, our work makes the following three key contributions:

- We present a motivational study on the performance of state-of-the-art federated learning frameworks using baseline models suited for time-series sensor data with increasing model complexity. Our results show that these existing frameworks are *not* flexible enough to support deep and complex neural network architectures needed for analyzing complex data, which are prevalent in modern sensor data processing applications.
- To address such limitations, we propose *AttFL*, a novel federated learning framework for time-series sensor data processing. The key idea behind *AttFL* is the incorporation of three attention modules to extract local, sub-global, and global temporal features of the time-series sensor data directly on top of the baseline model. As such, local devices in *AttFL* only share their attention feature maps and a small amount of attention module parameters with the server, which not only improves local model personalization performance but also minimizes both communication and computation costs.
- We perform extensive evaluations to understand the performance of *AttFL* and previously proposed federated learning frameworks with various sensing applications. We show that *AttFL* shows state-of-the-art performance in reducing communication and computational overhead while achieving high inference accuracy for different model architectures. We also present results from a heterogeneous embedded platform testbed to show that *AttFL*'s local operations can execute with minimal overhead on real embedded sensing platforms.

2 RELATED WORK & PRELIMINARY STUDY

2.1 Backgound on Federated Learning

• **Federated Learning.** Federated learning is a machine learning paradigm focusing on gathering knowledge from many distributed local models to collaboratively compute improved model parameters at the server. Specifically, in federated learning, after locally training models using self-collected data, local devices share sub-model information without exchanging the actual sensing data and the server exploits this information to re-compute and re-distribute an improved model to relevant devices. This offers a privacy-preserving approach of improving model performance at reduced costs. Commercial services such as Google Gboard [32], Apple Siri [1], and Nvidia Clara [23] use such operations by collecting information from end-users to eventually offer better (personalized) services. Various schemes have been proposed regarding which and how model parameters should be shared, with the goal of improving inference accuracy when aggregated over many participating devices while minimizing the overall overhead [48, 88]. Table 1 summarizes some representative existing work in this domain. A commercially widely used approach, FedAvg [62] aggregates randomly selected local devices' model weights and averages them to train a single global model for re-distribution. However, FedAvg does not target to optimize the computation and communication costs and lacks support for effective per-device personalization, meaning that only a global model is generated despite each device dealing with specific classes and data distributions.

PerFed [21] generates personalized models but fails in achieving communication and computation efficiency. pFedAtt [59] exploits attentive functions at the server for computing local model similarity for personalization and pFedMe [73] supports personalization by regularizing loss functions between the global/local models, but both approaches impose high communication and computation costs. Other approaches, such as LG-FedAvg [50] achieves good model personalization with reduced communication cost and pFedMe-based FCFL [89] introduces a fair personalized federated learning framework under inferior networking conditions. However, these frameworks still fail in suppressing computation costs.

• **Personalized Federated Learning.** Personalized federated learning essentially benefits from identifying and clustering similar clients/devices or models that can benefit from sharing a common model or model parameters. Specifically, this similarity-based clustering can be achieved via constructing client-based clusters or model-based clusters. Client cluster-based personalized federated learning groups clients into multiple clusters and then performs federated learning by designing global models specific for each cluster. IFCA [25] classifies clients into K clusters and assigns K global models by applying an iterative federated clustering to the last layer of all clusters. HypCluster [60] uses hypothesis-based clustering, clustering clients with the lowest loss and FedGroup [20] uses the K-means++ algorithm to group multiple clients into multiple clusters. More recently, FeSem [57] proposes a multi-center aggregation approach by designing multi-center federated loss for client clustering and k-FED [19] tries to find the best-effort number of device clusters based on the Lloyd's method used in k-means clustering.

In contrast, model cluster-based personalized federated learning allows each client to possess a personalized model in terms of model structure, parameters, and loss functions. This approach reduces the computation and communication overhead caused from the use and distribution of multiple global models. FedMask [45] (detailed in Table 1) is a framework that achieves personalized federated learning by sharing only bit masks of local models, and for the same purpose, FedDL [76] applies a model layer-wise sharing approach. FedProto [74] adopts prototypes representing the concept of classes instead of gradients and achieves a privacy-aware and efficient exchange of model parameters. pFedHN [71] delivers a representation of a local client to a global hypernetwork to output personalized heterogeneous models and FedMeta [14] achieves personalization using meta-learning based on FedAvg. DistFL [52] exploits statistical information encoded in batch normalization layers using an input sample to achieve model cluster-based personalized federated learning. Ditto [47] introduces a penalty term-based bi-level optimization framework to learn local models that are encouraged to be close together by global regularization. As a study extending Ditto in the mobile computing domain, FLAME [17]

proposes a user-centered federated framework that enables multiple mobile devices to participate in local model training based on a device selection algorithm using energy-efficient consideration. However, these methods using prototypes, hypernetworks, and meta-learning require a significant amount of additional memory usage together with additional data or training, making them less-suitable for mobile and embedded computing scenarios. Ditto-based works specifically lose the capability to capture complex relations among tasks related to classification or regression by exploiting simple personalization terms [61]. The federated learning framework we propose in this work can be classified to a model clustering approach. Specifically, we put extra emphasis on its practical usability in mobile and embedded computing environments by minimizing the memory and computational requirements.
• **Distributed Learning.** In the context of distributed learning, various techniques have been explored to minimize communication and computational costs in maintaining distributed deep learning operations. The two primary directions of research pursued in distributed learning are quantization and model sparsification. First, quantization involves minimizing the number of bits transmitted for each element, thereby reducing communication costs. Quantized SGD (QSGD) [4] introduces compression schemes that compress gradient updates at each client by leveraging three quantization levels: 0, 1, and -1. SignSGD [10] addresses the bottleneck problem of training large neural networks-based distributed learning by transmitting only the signs of each minibatch stochastic gradient. These approaches reduce the amount of traffic needed to exchange model parameters, but induce high variance in the resulting sparse gradients leading to slow model convergence. For this reason, GSpar [81] proposes rand-k sparsification, whereby k gradients are retained at random, biased by their absolute value, and the remaining gradients are zeroed. The remaining gradients are then rescaled to ensure the gradient is unbiased. Samuel et al. [34] introduce the natural compression technique that compresses updated vectors based on randomized rounding. Secondly, apart from quantization, sparsification focuses on exchanging a subset of the data elements rather than the full set. Alham and Kenneth [3] proposed a faster distributed stochastic gradient method by sparsifying the gradient via dropping a certain portion of elements with small gradient values. SKETCHED-SGD [38] effectively reduces communication cost without degrading model performance by transmitting sketches of the gradients rather than transmitting the entire set. As representative studies that combine both quantization and sparsification, 3LC [51] and Qsparse-local-SGD [9] improved the data compression ratio and speed while preserving the accuracy by combining quantization and sparsification methods.

Comprehensively, we acknowledge the efforts from these previous works in designing an efficient federated learning framework. However, most of these works consider a computation-rich environment compared to what embedded and mobile devices face in reality. Nevertheless, observations and research directions that these works introduce have provided essential guidelines for mobile/embedded-specific federated learning frameworks.

2.2 Federated Learning for Mobile and Embedded Sensing Applications

Mobile and embedded sensing applications are typically designed around low-power and resource-limited platforms. Often, these systems face the dilemma of having to delegate data analysis operations to an external server (due to limited local computing power) while sacrificing the communication and energy overhead of transmitting raw data [17, 30]. An alternative design is to perform local data analytics using device-embedded data processing algorithms. With improvements in low-power processor technology, we are seeing more of such designs in recently proposed systems where deep learning models are integrated into the sensing platforms themselves [22, 28, 53, 55, 65, 84]. Still, these systems are mostly limited to simply "utilizing" the model for inference operations. Nevertheless, since data captured from sensing platforms holds unique spatio-temporal characteristics specific to the target environment, some level of "model personalization" can improve the data analysis performance. One way to do so is to perform additional local training using data collected from each device. While this suppresses local data from leaving the device, the statistical diversity can be limited [62]. Another option is to have many devices share their local data with a server for model re-training, but this compromises

data privacy and induces communication overhead. Applying federated learning for such embedded/mobile sensing applications can alleviate these issues by offering an environment where (i) local data is not shared with any other entity and (ii) knowledge captured from other devices with similar target classes is shared to improve local model performance.

A handful of previous works have proposed federated learning frameworks that address the resource limitations of mobile and embedded sensing systems. Among many, FedMask [45] and FedDL [76] are notable works that show good performance and have been extensively validated. FedMask leverages pruning and masking as a way to reduce computation and communication costs. While FedMask shows good performance over various tasks, it can only be applied to neural network models that can be sparsified and its optimization requires modifications to the baseline deep learning model. FedDL exploits dynamic layer sharing in an iterative layer-wise manner. To recognize and group shared layers between local devices, FedDL measures affinity scores between different local models based on the Kullback-Leibler divergence. Unfortunately, this process increases computation overhead when a baseline model with a deep architecture (i.e., many layers) is used. Moreover, while these two approaches well-consider mobile/embedded sensing system characteristics such as data heterogeneity, communication efficiency, and computation efficiency, they were not validated, nor did they target, models with complex model architectures. Additionally, while analyzing data for typical sensing applications involve the understanding of complex time-series data, these frameworks have mostly been validated on convolutional neural network (CNN) architectures (with the exception of FedMask being tested for a simple two-layer LSTM), whereas recurrent neural network (RNN) architectures are known to be more suitable for effective time-series data analysis.

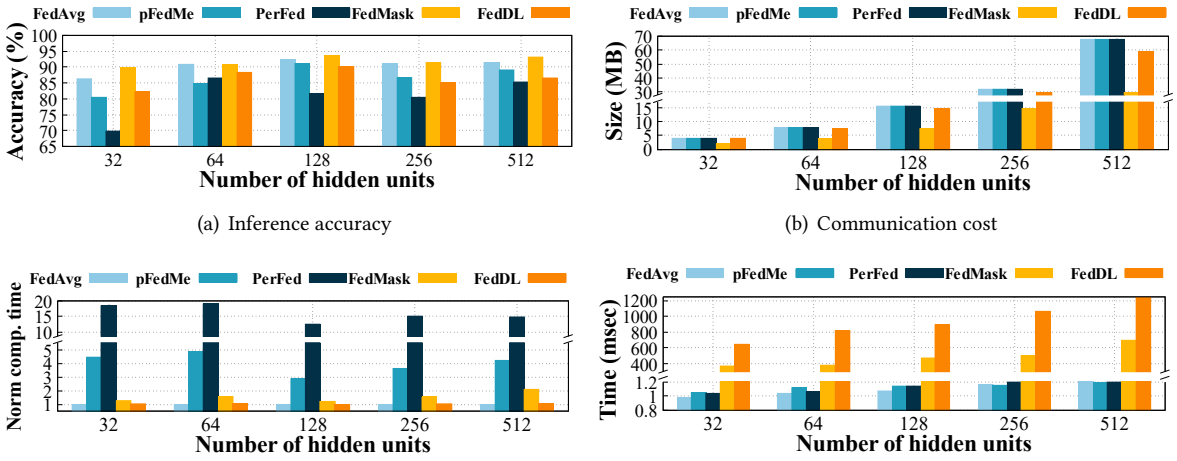
3 MOTIVATIONAL STUDY AND DESIGN CONSIDERATIONS

Recent advancements in low-power MEMS technology have offered embedded and mobile application designers with a rich set of sensors to integrate in their systems. Ranging from physiological/physical signals from humans to audio sensing data, over the recent decade we have seen an influx of human- and environment-centered applications implemented using various low-power sensing components [2, 65, 66]. In the earlier stages of sensing application research, the focus was mostly put on efficient data collection [12, 41]. In these systems, a large number of sensing devices were deployed to different users or environment locations to capture various features associated with the subjects or surroundings, typically over long periods. As a result, large time-series datasets are generated, and a server or a powerful edge device takes the role of analyzing the data [49, 64].

More recent systems eliminate the need for a separate server and operate deep learning models on the sensing devices themselves to minimize communication overhead and contain the data (which can potentially be privacy-sensitive) on-device [35, 36, 65]. Yet, models used in these systems are mostly generic and are trained on an external server with data from a generalized dataset. A next step towards improving the data processing abilities is to locally train these DNNs so that they better-suit the data characteristics specific to each sensing device.

Federated learning addresses such limitations by *personalizing* deep learning models with respect to locally captured data via local training. In addition to solely exploiting local data, by collaboratively gathering model information at the server from many participating devices, federated learning improves individual models using knowledge extracted from distant (yet logically similar) devices. However, as aforementioned, existing federated learning frameworks have mostly focused on CNN-type DNNs with (relatively) small number of layers (e.g., shallow networks). This constraint was, in some sense, inevitable given the resource limitations of mobile and embedded computing capabilities as they could not operate complex models or endure high computational costs. However, improvements in low-power processors and recently proposed low-memory DNN training frameworks [26] now makes local use of more complex models possible.

Furthermore, since many mobile/embedded sensing applications generate time-series data streams, systems can benefit from the use of more complex network architectures that expand beyond simple CNNs. In fact, architectures



(a) Inference accuracy

(b) Communication cost

(c) Normalized local device computation time (per-epoch)

(d) Server-side computation time

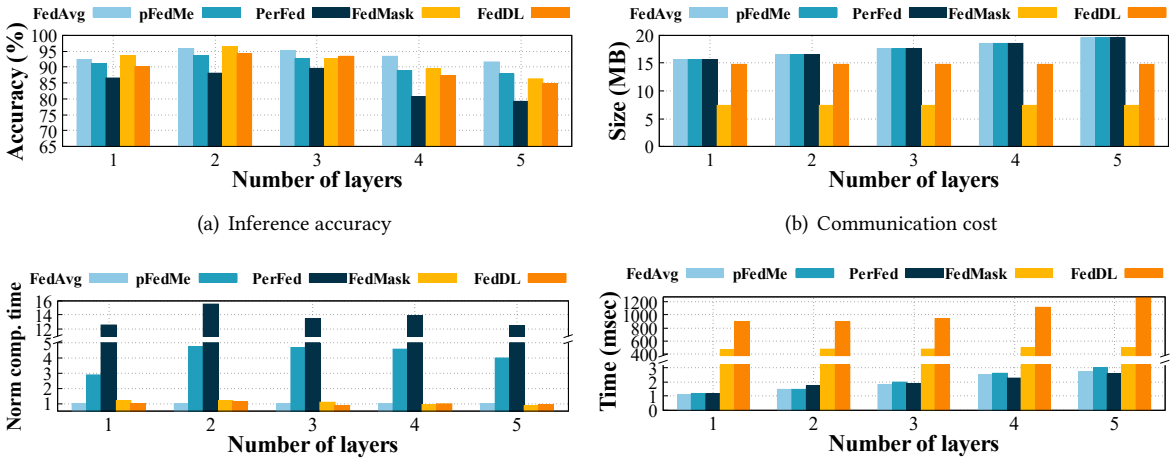
Fig. 1. Inference accuracy, communication cost (i.e., amount data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, and FedDL with *increasing Bi-LSTM's hidden units* using the *MIT-BIH dataset*. These results present the performance impact of *wider* baseline models on different federated learning models.

such as stacked bi-directional LSTMs (Bi-LSTMs) are commonly used in time-series data analysis [66] and their hidden unit counts and layer depth show increasing trends as researchers thrive to achieve higher accuracy[2].

Thus, to validate the suitability of applying existing federated learning frameworks for time-series data analysis, we take a physiological signal dataset and perform a motivational study to examine the performance of five federated learning frameworks (FedAvg, pFedMe, PerFed, FedMask, and FedDL) using a Bi-LSTM baseline DNN. Specifically, we use the MIT-BIH electrocardiogram (ECG) dataset and perform the cardio-signal classification task [63], which is a common mobile/remote healthcare application addressed in many previous work [11, 15, 27]. In this study, we vary the number of hidden units and the depth of the Bi-LSTM to examine how each approach scales to increasing model complexity. For model training, we apply a batch size of 64, use the Adam optimizer or stochastic gradient descent (with respect to the original work's specification), and select a learning rate of 1e-4. For federated learning parameters, we randomly select five participating local devices from a total of 47 and apply 5-epoch local training per federated learning round and test for 50 rounds. To assure non-IID for locally trained models, we randomly assign two classes of data for each local device and present results for a five-fold cross-validation. For FedDL, we use the first three layers for layer-wise sharing as in the original work and assign a separate CPU core to emulate each local device operation (2.2 GHz). While this environment can be seem powerful, most previous works were evaluated in similar settings and recent embedded platforms (e.g., high-end Raspberry Pis and NVIDIA Jetson GPUs) possess comparable processor specifications.

In Figure 1, we plot the inference accuracy, network communication cost (i.e., amount of data transmitted between the local devices and server), normalized local device-side computation time per-epoch (FedAvg as baseline), and server-side computation time for five different federated learning frameworks with increasing number of Bi-LSTM hidden units (single Bi-LSTM layer). As plots show, we see an increasing trend in classification accuracy with more hidden units. This is reasonable given the inherent complexity of ECG signals. However,

[2]Newer architectures such as Transformers have proven to show even higher accuracy compared to RNNs (e.g., LSTMs), but, despite improvements in embedded computing capabilities, they are still heavy to locally operate in real-time [16].



(a) Inference accuracy

(b) Communication cost

(c) Normalized local device computation time (per-epoch)

(d) Server-side computation time

Fig. 2. Inference accuracy, communication cost (i.e., amount of data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, and FedDL with *increasing Bi-LSTM layers* using the *MIT-BIH dataset*. By varying the number of layers, these results present the performance impact of *deeper* baseline models on different federated learning models.

this performance is achieved by sacrificing the overhead at different layers of the system. Notice that recently proposed schemes such as FedMask and FedDL, successfully suppress local computation despite increasing hidden units. However, the communication cost and server-side computation time show a steep increase as more hidden units are used, suggesting that these approaches are not scalable with model complexity. In Figure 2, where we present plots for increasing number of LSTM layers (128 hidden units), we can notice that the overhead, in terms of communication/computation costs, shows similar trends. It is interesting to note that the increased number of Bi-LSTM layers (more than 2) on the baseline DNN model *negatively* impacts the accuracy of the federated learning frameworks. This shows that exploiting existing frameworks could require careful engineering and the selection of baseline DNN models can be limited to only shallow networks.

From our motivational study, we can identify a number of challenges to be addressed when designing a federated learning framework specific for sensor data processing applications. First, given continuous sensing scenarios (with baseline DNNs that specialize in this domain), a federated learning framework should be capable of supporting models that analyze complex time-series data streams. Second, since mobile/embedded sensing platforms induce computational and energy constraints, it is important that the data exchange between the server and local devices is kept minimal while suppressing the local computational load. Third, the server's computational load is also an important aspect to optimize. Since sensing applications typically involve data collected from many devices, federated learning frameworks should also minimize server-side computational overhead to support scalable services. We now present details on our proposed federated learning framework to address these challenges in the sections that follow.

4 *ATTFL*

We carefully consider the challenges discussed in Section 3 as we design *AttFL*, a novel federated learning framework for mobile/embedded sensing applications. Specifically, *AttFL* makes no explicit changes to the original baseline DNN architecture, rather it appends a set of attention modules to obtain detailed local and global knowledge of the input data and the baseline model structure. By appending attention modules to the

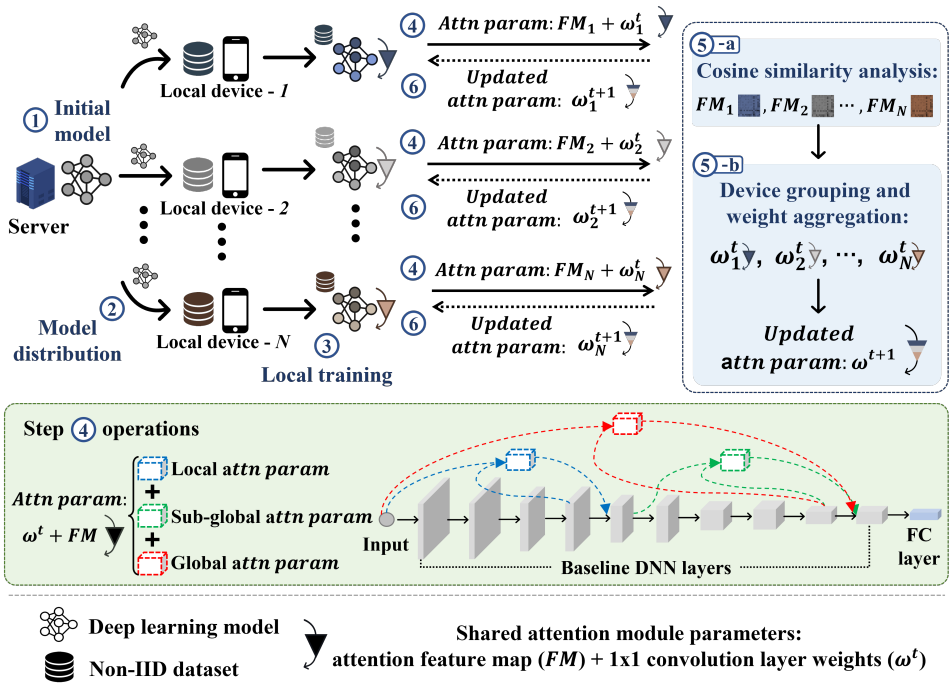


Fig. 3. Overall architecture and step-by-step flow of *AttFL*.

baseline DNN, we argue that the original model parameters can benefit from the additional knowledge related to the short term and long term characteristics of the input data, making our approach powerful (and desirable) for time-series data-based applications. More importantly, given that attention module computations are affected by interconnected baseline DNN parameters and the fact that attention outputs are re-connected to the baseline DNN as input for subsequent DNN layers, the attention modules can act as effective abstractions of the baseline model parameters and configuration. As a result, instead of directly exchanging large-sized baseline DNN model weights with the server as part of the federated learning process, as we detail later in this section, local devices in *AttFL* exchange two pieces of information from its attention modules: (i) the attention feature maps and the (ii) convolution layer weights used to interconnect the attention modules with the baseline DNN. The server, upon receiving this information from local devices, identifies devices with similar attention feature maps (i.e., dealing with similar local data classes) and aggregates their convolution layer weights for re-distribution. This allows local models to obtain knowledge learnt from distant devices that share similar data characteristics, offering an efficient way of improving local model performance despite the limited data it can self-acquire.

More specifically, as the green highlight box in Figure 3 illustrates, *AttFL* appends three attention modules to the baseline DNN to extract local and global features from the input data. Since attention modules learn spatio-temporal input features, they can easily integrate with (and be beneficial for) various DNNs for understanding time-series data features in-depth [58]. Similar to Transformer designs [77], attention modules in *AttFL* dynamically adjust the focus of underlying DNN towards the timestep of neural network channels via dynamically estimated corresponding weights [77]; thus, generating an adequate representation of the local input data and how they are dealt with within the model. We note that the use of local and global features is not new and has been employed in some previous work to well-analyze time-series data. As examples, DeepSense [85] exploits local and global concepts to extract features over multiple time-series modalities and works such as SADeepSense [86] and

GlobalFusion [54] also adopt similar concepts. Compared to *AttFL*, which targets to analyze the input data and the underlying model on a local and global scale, these previous work focus only on the input data themselves as they are not designed for federated learning applications. Nevertheless, as we later show, the use of local and global features are effective and schemes such as DeepSense can serve as baseline DNN models for *AttFL*.

As we illustrate in Figure 3, a federated learning system starts with the server initially training a target DNN (Step ①) and distributing the model to all participating local devices (Step ②). In operation, once models are trained locally using self-gathered data (Step ③), local devices share attention module parameters (Step ④), which include their attention feature maps (c.f., illustrated in Fig. 5) along with the convolution layer weights for computing the attention output (c.f., green 1x1 convolution layers in Fig. 5) and the weights for the convolution layer that connects the attention output back to the baseline DNN (c.f., purple 1x1 convolution layer in Fig. 5). With this information, since attention feature maps summarize information on the data classes that each local device experiences *under non-IID environments*, the server can compute a cosine similarity of the attention feature maps gathered from devices participating in the federated learning round to identify *device groups* sharing similar local data characteristics (c.f., Step ⑤-a in Fig. 3). Devices in the same "device group" are identified as nodes that can benefit from sharing aggregated weights (Step ⑤-b), and *AttFL*'s server re-distributes the aggregated information after averaging within the groups (c.f., Step ⑥ in Fig. 3). Finally, local models are updated with the new weights until the next federated learning round takes place. The remainder of this section presents the rationale behind using attention modules for effective federated learning support and details on the aforementioned respective operations for the local devices and server.

4.1 Impact of Appending Attention Modules to Baseline DNNs

We start detailing *AttFL* by discussing the design principle behind appending attention modules as a light-weight representation of the underlying DNN parameters/operations and the local data experienced at each device.

The attention mechanism is a technique widely used in deep learning models to weigh the importance of different "parts" of an input sequence. It allows the model to focus on the most relevant portions of the input and exploit more targeted information in its inference operations [13, 40, 77]. In other words, the attention mechanism identifies the correlation between an input and output to understand which information needs more *attention* to effectively process the data. Given that *AttFL* tries to support federated learning for time-series data inputs, exploiting the attention mechanism can be an effective way of capturing the input data characteristics. By appending attention modules to a baseline RNN architecture (given that RNNs are known to better analyze time-series data), attention modules can capture baseline model-processed features, extract additional features that well-represent the input data characteristics, and re-inject this information back to the RNN so that the baseline model parameters can adjust itself to focus on the more essential parts of the data. Specifically, in addition to this potential model performance improvement, in *AttFL*, we exploit the attention module to act as an *efficient abstraction of the underlying RNN model and its operations*. More technically, given that input data representations are more "raw" or local at the earlier parts of the RNN and become more "generalized" or global as the data progresses through RNN blocks, as Figure 4 illustrates, the attention module takes both (relatively) local and (relatively) global data representations (i.e., features) and extracts how the underlying RNN block operates to extract the (relatively) global features from the (relatively) local ones. Naturally, through this process, the attention module (and its parameters) will gain knowledge on the input data characteristics as well as the baseline RNN's operations/configurations; thus, becoming an effective abstraction of the input data features and baseline model's operational parameters. This means that when the server gathers attention module information from multiple federated learning participating devices, it can identify devices dealing with similar data characteristics, allowing it to suggest an improved model configuration for a targeted group of devices.

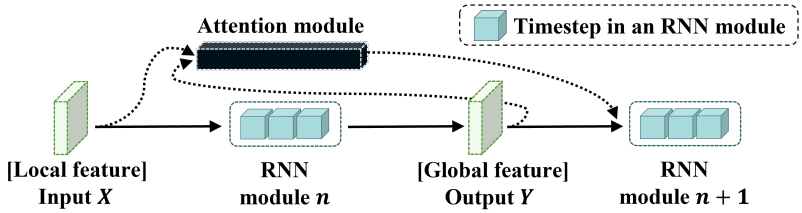


Fig. 4. Example of the common way of applying attention to a module.

Note in Figure 4 that the outputs from the attention module are fed back to the baseline model. As briefly mentioned, this is to re-affect the baseline model so that the attention information captured can gradually modify and improve the underlying model weights. Thus, changes to the attention module parameters will penetrate into the baseline model so that subsequent layers adjust their parameters. On a federated learning standpoint, this means that if the server were to send back updated attention module configurations (as a result of the federated learning aggregation process), simply updating the attention modules will effectively impact the underlying model to adjust to the new (improved) model configurations. Given that this requires less data exchange compared to directly exchanging model weights, our approach provides a cost-effective way of updating local model configurations in the federated learning process.

Based on these observations that appended attention modules can effectively extract and abstract underlying model characteristics and invoke improved operational modifications, we exploit and integrate the attention mechanism when designing *AttFL* as we detail in the following sections.

4.2 Local Device Operations: Attention-based DNN Feature Extraction

In a deep learning model, the lower layers (i.e., positioned earlier in the model pipeline) typically focus on features specific to the input sequence itself, while features from higher layers (i.e., layers positioned later in the model architecture) are more sensitive to the wider target class characteristics. Therefore, to fully understand the input data characteristics, it is important that both local and global features are well-extracted from the deep learning model. Note that as we observed in Section 4.1, attention modules hold the capability to well-capture time-series input characteristics [43, 68] and improve the feature extraction performance at different layers of the baseline model and assure a light-weight representation of the underlying model parameters. Given these benefits, *AttFL* appends attention modules at three different parts of the baseline DNN.

As the green box in Figure 3 shows (detailed illustration for Step ④), *AttFL* maintains the original structure of the baseline DNN (without modifications) while appending three attention modules: one for understanding the local features from the model (at the lower layers - blue), second for sub-global features (at the higher layers - green), and the third for global features (covering the full model - red). Based on empirical experiences, *AttFL* splits the target neural network at (approximately) the 50% point of the full network architecture and uses the 'input-to-50%' range as the local feature extracting attention module, and the remainder of the model for sub-global features. The attention module for global feature extraction covers the entire model pipeline from input to output. These attention modules intercept parameters from the underlying DNN to compute its outputs, which are then used as (partial) inputs to the subsequent layers of the baseline DNN. As a result of this connection, attention feature maps and its outputs are affected by both the underlying DNN parameters and the input data, which, since later re-connected with the baseline DNN, offers the baseline with an improved understanding of the input feature information with considerations for per-local device personalization by assuring that data classes (e.g., distinct data characteristics) that the device is experiencing are well represented.

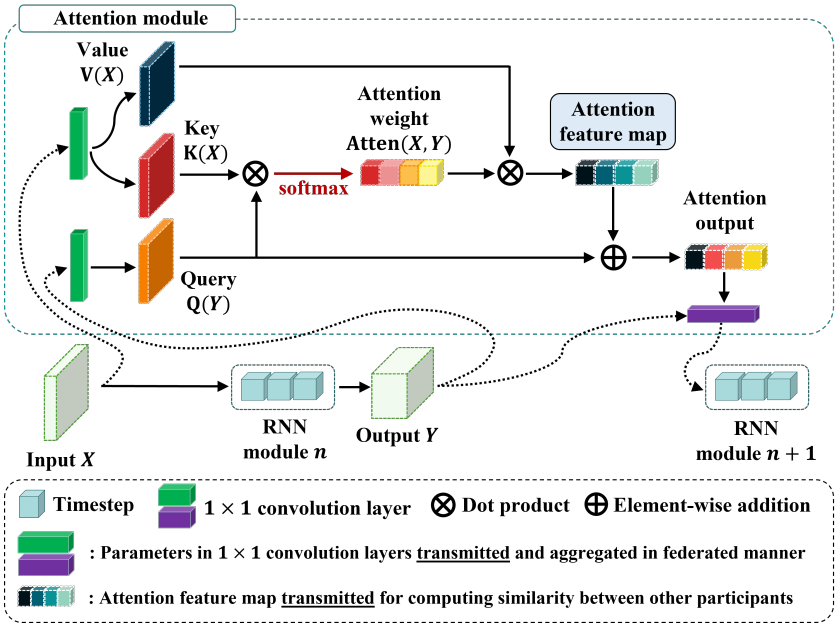


Fig. 5. Attention module architecture in *AttFL*.

Figure 5 presents details on the attention architecture used in *AttFL*. Specifically, *AttFL*'s attention mechanism adopts the *multiplicative attention mechanism* design, which leverages the dot-product between a query and key to estimate the attention weight corresponding to a specific axis of the local input. Here, we borrow concepts from the self-attention mechanism [77] and detail the module as the following.

Say that the input and output of the RNN module that the attention is connected to are $X \in \mathbb{R}^{B \times T_X \times H_X}$ and $Y \in \mathbb{R}^{B \times T_Y \times H_Y}$, respectively, where B is the batch size, T_X is the number of input timesteps, and T_Y is the number of output timesteps, H_X is the number of input hidden units, and H_Y is the number of output hidden units. We first pass the input and output through a dimensional transformation using 1×1 convolution to unify the dimensions. Via this dimensional transformation using input X (a relatively local feature captured earlier in the pipeline) and Y (a relatively global representation), we transform (potentially) different latent spaces of X and Y into a common latent space for computational efficiency. We then exploit an attention function $Attention(X, Y)$ that takes the transformed X and Y as input to compute the attention weight of each input source so that the attention weight corresponding to a specific axis of the latent space can be identified.

$$\begin{aligned} &Attention(X, Y) \rightarrow attention\ weights \in \mathbb{R}^{C_{1\times1\ conv} \times C_{1\times1\ conv}}, \\ &X \rightarrow \mathbb{R}^{B \times T_X \times H_X}, Y \rightarrow \mathbb{R}^{B \times T_Y \times H_Y} \end{aligned} \tag{1}$$

Based on this, local and global features, now inputs of the same dimension, can be expressed as a key $K(X)$ and a query $Q(Y)$, and the attention weights can be expressed via softmax as follows:

$$\begin{aligned} &Attention(X, Y) = softmax(\frac{Q(*(Y))K(*(X))^T}{\sqrt{h}}), \\ &*(X) \rightarrow \mathbb{R}^{B \times T_X \times H_{1\times1\ conv}}, *(Y) \rightarrow \mathbb{R}^{B \times T_Y \times H_{1\times1\ conv}}, \\ &* : \mathbb{R}^i \rightarrow \mathbb{R}^j,\ 1 \times 1\ Convolution\ where\ i, j \rightarrow\ hidden\ unit\ size \end{aligned} \tag{2}$$

Here, we prevent the vanishing gradients effect [77] by dividing with the root of the scale factor h, and use softmax to emphasize informative features and normalize the attention weights. Note that h is configured as the dimension of a specific latent space in Equation 2. Next, we multiply the attention weights $Attention(X, Y)$ with value $V(X)$ (c.f., Fig. 5) to obtain an attention local feature map by propagating the scalar attention weight to $V(X)$. By adding the query $Q(Y)$, we compute the attention output as below:

$$\begin{aligned} &Attention\ feature\ map = Attention(X, Y)V(*(X)), \\ &Attention\ output = Attention\ feature\ map + Q(*(Y)) \end{aligned} \tag{3}$$

Finally, we concatenate the obtained attention output with the existing input of the next layer (of the baseline DNN) by resizing it via a 1x1 convolution to match the layer's input size. As per our formulations, *AttFL*'s local devices share (with the server) the local, sub-global, and global attention module parameters. Since the attention module itself does not possess specific weights and biases of the baseline model (and rather a component abstracting the baseline DNN parameters and input features), *AttFL* exchanges the convolution layer weights for $Q(Y)$, $K(X)$ and $V(X)$ (dimensions of $K(X)$ and $V(X)$ are identical) used to compute each of the three attention modules (green blocks in Fig. 5), which embed the relationships between the baseline DNN and the attention parameters. In addition, local devices also share the attention feature maps and convolution layer weights that connects the attention output with the baseline DNN (purple block in Fig. 5). By exchanging only this information, *AttFL* effectively abstracts the information needed for understanding the baseline DNN parameters and input data characteristics that the device is experiencing. Quantitatively, when applying *AttFL* to a Bi-LSTM baseline, a local device shares only ~1.37 MB of data for all three attention modules per-learning round, indicating that *AttFL* is cost-effective in terms of communication overhead. While extremely small in size, we later show that this sub-model abstraction is sufficient enough to capture the similarity among different local devices and provide effective updated model weights for improved personalization.

4.3 Server Operations: Device Feature Similarity Detection and Weight Aggregation

The *AttFL* server performs two major operations. First, given local device model features, it computes *device groups* that share similar local model characteristics. By doing so, the server gains knowledge on which devices target detect/classify of similar events/classes. Second, the server aggregates matrix weights collected from devices of the same group and re-distributes updated weights so that local models can exploit knowledge from relevant devices in the network.

For device grouping, *AttFL* exploits attention feature maps for local, sub-global, and global attention modules collected from respective participating local devices within a federated learning round. We do so as the feature map data captured from individual devices embed information on their input data distribution and characteristics, and if the feature map data are similar enough, we can roughly conclude that two (or more) devices share similar input characteristics. This also means that such devices operating with similar input data can benefit each other (in terms of model personalization) by sharing their model parameters.

To identify similarity groups using the attention feature maps captured from participating devices, the server computes the cosine similarity of the attention feature maps for all pairs of devices. Note that the following operations take place for all three attention feature maps (e.g., local, sub-global, and global) separately.

$$Cosine\ Similarity(\mathbf{u}, \mathbf{v}) = \frac{<\mathbf{u},\mathbf{v}>}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|} = \frac{\sum_{i=1}^{n} \mathbf{u}_i \mathbf{v}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{u}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{v}_i)^2}} \tag{4}$$

Here, u and v are the 1-D representation of attention feature maps gathered from two different local devices, and n denotes the number of elements in each feature map's 1-D representation. As also noted in many previous works, a major benefit of exploiting cosine similarity is its low computation cost and possibility for potential performance optimization [7, 59, 70]. While aggregation operations take place on the server, given the size of feature maps for

local, sub-global, and global attention modules individually, computing a greedy combination among many local devices can result in a significant amount of computational overhead. To make this process even more efficient, we further optimize the cosine similarity computation by exploiting the fact that the squared Euclidean distance offers proportional results at much less complexity [42]. Therefore, *AttFL* computes the Euclidean distance of features as an approximate representation for determining inter-local device cosine similarity. Specifically, *AttFL* computes the $l2$ norm for input feature map vectors and optimizes the similarity evaluation as the following.

$$\begin{aligned} \text{let } \tilde{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|_2},\ \tilde{\mathbf{v}} &= \frac{\mathbf{v}}{\|\mathbf{v}\|_2} \\ \text{then, } \tilde{\mathbf{u}}^\top \tilde{\mathbf{u}} = \tilde{\mathbf{v}}^\top \tilde{\mathbf{v}} &= 1 \ (\because \ \|\tilde{\mathbf{u}}\|_2 = \|\tilde{\mathbf{v}}\|_2 = 1) \\ \|\tilde{\mathbf{u}} - \tilde{\mathbf{v}}\|_2^2 &= (\tilde{\mathbf{u}} - \tilde{\mathbf{v}})^\top (\tilde{\mathbf{u}} - \tilde{\mathbf{v}}) \\ &= \tilde{\mathbf{u}}^\top \tilde{\mathbf{u}} - 2\tilde{\mathbf{u}}^\top \tilde{\mathbf{v}} + \tilde{\mathbf{v}}^\top \tilde{\mathbf{v}} \\ &= 2 - 2\tilde{\mathbf{u}}^\top \tilde{\mathbf{v}} \\ &= 2 - 2\cos(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}) \end{aligned} \tag{5}$$

where, $\|\mathbf{u}\|_2$ and $\|\mathbf{v}\|_2$ are the $l2$ norm for u and v respectively. Note that *AttFL* computes the pairwise similarity for all possible feature map combinations of local devices that participate in the federated learning round. Thus, resulting in a $k \times k$ matrix of similarity measures, with k participating devices. Our goal is to identify groups of devices that target similar class detection to improve the detection performance at respective local devices; thus, improve per-device personalization. Once the similarity scores for all three attention feature maps are computed (e.g., local, sub-global, and global), we take the mean of the three as the final similarity score for a specific device pair. In this work, we heuristically configure the similarity threshold to 0.5 based on empirical validations.

With similar local devices identified, *AttFL* aggregates the *convolution layer weights* (for $Q(Y)$, $K(X)$, and $V(X)$) from the local, sub-global, and global attention modules by taking the average values within each group. Taking the average of model parameters collected from different devices is a widely used approach and has been shown to be an effective way of optimizing the objective function [80, 82].

As a result, *AttFL* merges a DNN's spatio-temporal information captured from individual devices so that they enhance the inference performance of devices within the same similarity group. Putting this into a sensing application perspective, given a model deployed at local devices to detect the presence of a pre-defined event, using *AttFL*, the model is continuously updated via training results obtained from other devices that target the same (or similar enough) event types. Since a single target event can show various patterns, exploiting this collaborative knowledge can allow the model to prepare itself towards a wider distribution of event patterns.

We note that by using three attention modules that cover different portions of the baseline model at the local devices, *AttFL* targets to extract and abstract model and input data characteristics at different dimensions. Which allows the server to make accurate device grouping decisions. At the same time, this also allows for a more effective baseline model update to take place (at different locations within the model) when improved configurations are sent from the server to the local devices. Thus, *AttFL* achieves personalization by grouping local devices' models based on their cosine similarity score on a per- training round basis, rather than maintaining separate device-level clusters on the server.

From a security perspective, the sharing of attention feature maps, which are derived from input data, may raise concerns on violating privacy assurance, which is an important benefit of implementing federated learning. However, *AttFL* differs from previous works in the fact that it does not share the model weight directly and rather only a small portion of information at the attention modules is shared. Practically speaking, this makes it challenging for malicious attackers to reveal the original input data from the attention feature maps [24].

Algorithm 1: *AttFL*'s server and client operations

Data: $(D_1, D_2, ..., D_N)$ where D_k is the k-th local data.
initialize the global model W
Server executes:
for *each round* $r = 1, 2, ..., R$ **do**
    n $\leftarrow$ max($F * N$,1) // F; the fraction rate, N; the total number of local clients, n; the number of selected local clients.
    $S_n \leftarrow$ Random set of n local clients
    for *each client* $k \in S_n$ **in parallel do**
        $\omega^n_{t+1}, FM^n_{t+1} \leftarrow$ ClientUpdate(k, ω_n)
    /* Scores on local, sub-global, and global attention feature maps */
    Cosine similarity scores($\overline{Scores}$) $\leftarrow$ Cosine similarity calculation(FM^n_{t+1})
    /* Averaging weights among grouped users. */
    $G_n \leftarrow \overline{Scores}$-based aggregation($\omega^n_{t+1}$)
    for *each group* $g \in G_n$ **do**
        $\omega_{t+1} \leftarrow \sum_{k=1}^{g_K} \frac{1}{g_n} \omega^k_{t+1}$, client $k \in$ group g
    return ω_{t+1} // Transmit the updated weights of local users participating in the round r.
ClientUpdate (k, ω)**:**
$B \leftarrow$ (split D_k into batch size B)
/* E; the total number of epochs, FM: an attention feature map. */
for *each local epoch i from 1 to* E **do**
    for *batch* $b \in B$ **do**
        $\omega \leftarrow \omega - \eta \bigtriangledown \ell(\omega;$ b)
        $FM_{local} \leftarrow FM_{local} + FM^b_{local}$
        $FM_{sub-global} \leftarrow FM_{sub-global} + FM^b_{sub-global}$
        $FM_{global} \leftarrow FM_{global} + FM^b_{global}$
return $[\omega_{local}, \omega_{sub-global}, \omega_{global}]$, $[\frac{1}{\|B\|*E} FM_{local}, \frac{1}{\|B\|*E} FM_{sub-global}, \frac{1}{\|B\|*E} FM_{global}]$
/* Transmit (a) convolution layer weights (ω) and (b) attention feature maps (FM) corresponding to local, sub-global, and global attention modules. */

Furthermore, while out of the scope of this work, integration with existing privacy-preserving techniques such as noise injection, information compression, or sparsification can further alleviate privacy-related concerns [90].

4.4 *AttFL* as an Algorithm

We now present the core operations of *AttFL* discussed above in algorithm form using Algorithm 1. Note that the operations presented in the algorithm are executed at each round. As the algorithm shows, the server starts by selecting a fraction of devices connected to the network as *participating devices* at each round. Next, the selected devices are notified, and each device executes the `ClientUpdate()` operation, performs local model training for n epochs, and sends to the server (a) attention feature maps and (b) convolution layer weights for $Q(Y)$, $K(X)$, and $V(X)$ (i.e., (a) and (b) in the local, sub-global, and global attention modules).

Once the information is received at the server, cosine similarity is used to identify similarity-groups for server to separate local devices with similar model characteristics. Finally, the reported features within each similarity group are averaged as an aggregated weight and re-distributed for local model updates. Our open-source implementation of *AttFL* is available at: https://github.com/eis-lab/attfl.

5 EVALUATION

We evaluate the performance of *AttFL* using four metrics: i) average local classification performance (i.e., accuracy), ii) communication cost (amount of data exchanged between local device and server), iii) per-epoch local device computational latency and iv) server-side computation latency. We compare the performance of *AttFL* with the five frameworks used in our motivational studies: FedAvg, pFedMe, PerFed, FedMask, and FedDL.

5.1 Experimental Setup

All DNN models used in this study are *initially* trained with Pytorch on a server with Intel i9-9900K@ 3.6GHz CPU, 64G RAM, and two NVIDIA RTX 3090 GPUs. For training, we adopt a negative log-likelihood with softmax at the end of the connected layer as our loss function, and use either Adam or SGD algorithm-based optimization with a learning rate of 1e-3 to 1e-4 and a batch size of 32 to 64. We take parameters directly from respective original work where possible. For FedDL, we use an interval of 6 between grouping rounds, an interval decay rate of 0.5, and the number of merging layers as 3. For FedMask, we set the pruning rate to 0.2 and threshold of 0.5 for binary mask optimization. λ and μ were set to 1e-1 and 1e-3, respectively, for pFedMe and learning rate-related parameters α and β in PerFed were set to 1e-2 and 1e-3, respectively, with δ used for the Hessian matrix as 1e-3.

We note that there were a number of parameters not specified in the original work that required self-optimization. For example, we operate FedMask on 50% of the CNN layers and select the last two RNN layers (similar numbers were used in the original work, but not specified). Additionally, the learning rate is set from 1e-3 to 1e-4 with respect to the dataset. For models that do not specify an optimizer, we apply the Adam or SGD optimizer based on empirical validations (whichever performs better for the target model/dataset combination).

In our evaluations, except for Section 5.7, where we test with real embedded GPU platforms, our evaluations exploit CPU cores as individual local devices. We understand that this configuration is not ideal, but has been used to validate many federated learning works as an "emulation" configuration given that scaling towards many real embedded local devices can be challenging. Nevertheless, we would like to point out two things regarding this evaluation setting. First, we use an environment where each core operates at 2.2 GHz. While not identical, the processing speeds of embedded platforms, combined with embedded GPUs are fastly catching up; thus, our evaluations can still show meaningful trends in computational latency. Second, while the local computational latency can be affected from the use of different processing components, performance metrics such as server-side latency and device communication cost (transmission data size) are independent of the local processor configuration used for evaluations. Nevertheless, in Section 5.7 we present a real-device implementation-based evaluation using Raspberry Pi, Jetson Nano, and TX2 platforms, to show that *AttFL* can effectively work as a federated learning framework for embedded sensing environments.

5.2 Target Applications and Datasets

We evaluate *AttFL* using three popular mobile/embedded sensing applications and datasets as we detail below.

- **Physiological Data (ECG Signal) Analysis:** As a representative healthcare application, analyzing ECG signals is crucial in understanding chronic cardiac disorders. Various mobile/embedded sensors exist to capture ECG, and their local processing is extremely important as transmitting raw ECG can lead to significant communication overhead and privacy threats [6, 29]. We exploit the MIT-BIH dataset [63], which is a representative ECG dataset containing 48 half-hour 360 *Hz* ECG records collected from 47

Table 2. Details on the three datasets used in our evaluations.

Dataset	Number of classes in dataset	Total number of samples device	Number of local devices	Average samples per local device	Non-IID?
MIT-BIH	7	98,676	47	2099.48	✓
HAR-UCI	6	10,299	300	34.34	✓
RAVDESS	8	7,356	240	30.65	✓

subjects. The dataset includes seven cardiac activity types and is widely used. For this task, we apply a Bi-LSTM and the DeepSense [85] network, a state-of-the-art CNN/RNN hybrid neural network designed for time-series data analysis, as baseline models for federated learning.

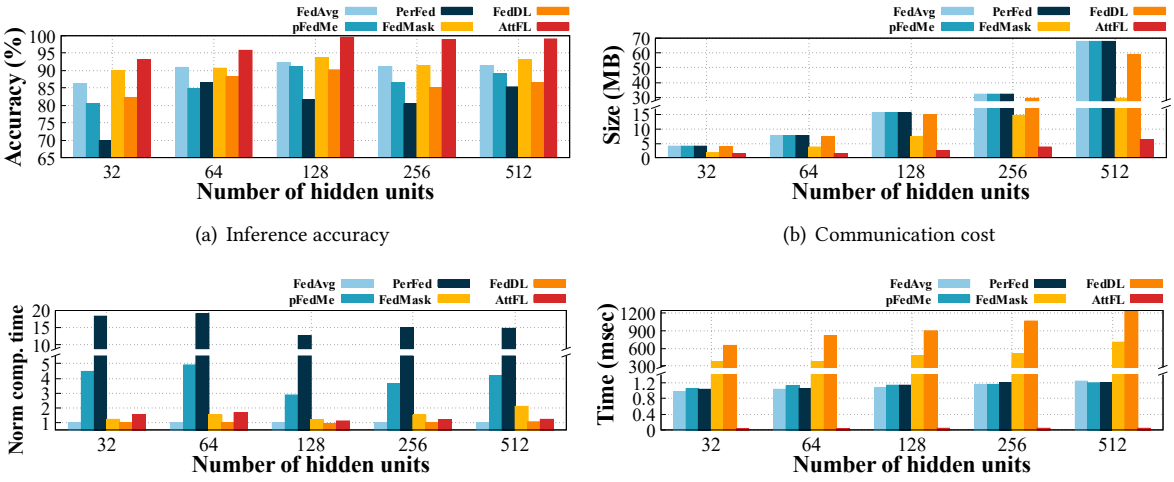
- **Human Activity Recognition:** For the human activity recognition application scenario, we use the HAR-UCI dataset [5] containing IMU data collected from 30 users via a wrist-mounted smartphone. Here, subjects performed six predefined tasks each representing different classes. The smartphone's IMU data was captured at 50 Hz to contain a total of 10,299 samples in the dataset. Similar to the ECG application, we use a Bi-LSTM and the DeepSense network as baseline DNNs.
- **Audio Classification:** With the ubiquitous deployment of smartphones, audio has become a representative sensing modality used in various applications from environmental analysis [67] and user emotion classification [37] to COVID cough detection [18]. In this work, we use the RAVDESS dataset [56] for an emotion classification task. The RAVDESS dataset contains 7,356 samples collected from 24 actors (12 female, 12 male) representing eight different emotions. As we later discuss, we use three LSTM variants (LSTM, Bi-LSTM, and Seq-LSTM), the DeepSense network, and a ResNet-18 implementation of the task as baseline DNNs to deeply examine the impact of *AttFL* on various neural network architectures.

For all evaluations, we use a five-fold cross-validation and split the data among different local devices to be non-IID as specified in Table 2. We emphasize that our target tasks focus on capturing the intra-sample temporal characteristics rather than inter-sample ordering. Thus, the five-fold cross-validation approach used in this work will not show critical bias from the temporal dependency inherent for each data sample. Note that our configuration ensures that redundant user samples are not used across folds comprising the training and testing sets, similar to the leave-one-subject-out (LOSO) method. In other words, by containing each user's data within a single fold, the DNN does not learn from samples belonging to subjects included in the test or validation sets. This frees our experiments from the common issues arising from cross-validation-based evaluations in time-series data [8, 31]. We randomly feed 10 data shards for each subject's data for the HAR-UCI and RAVDESS datasets. Note that we do not apply sharding to the MIT-BIH dataset given that ECG data is specific to a user and separately managing (or splitting) this information among more devices is contextually less meaningful. Finally, at each federated learning round, a subset among all local devices is randomly selected to participate in each round. The actual number of participating nodes in each round is defined in each subsection specific to the task.

5.3 Comparison Baselines

While we point the readers to the respective original work for details, below, we present an overview of each comparison baseline we use to compare the performance of *AttFL* in this work.

- **FedAvg** [62] is a general federated learning framework that transmits all model parameters from a local device to the server and re-distributes aggregated (averaged) global parameters to update local model weights.
- **pFedMe** [73] is a federated learning framework that exploits a server-side average aggregation strategy similar to FedAvg with a regularization loss function based on local user-side Moreau envelopes.



(a) Inference accuracy

(b) Communication cost

(c) Normalized local device computation time (per-epoch)

(d) Server-side computation time

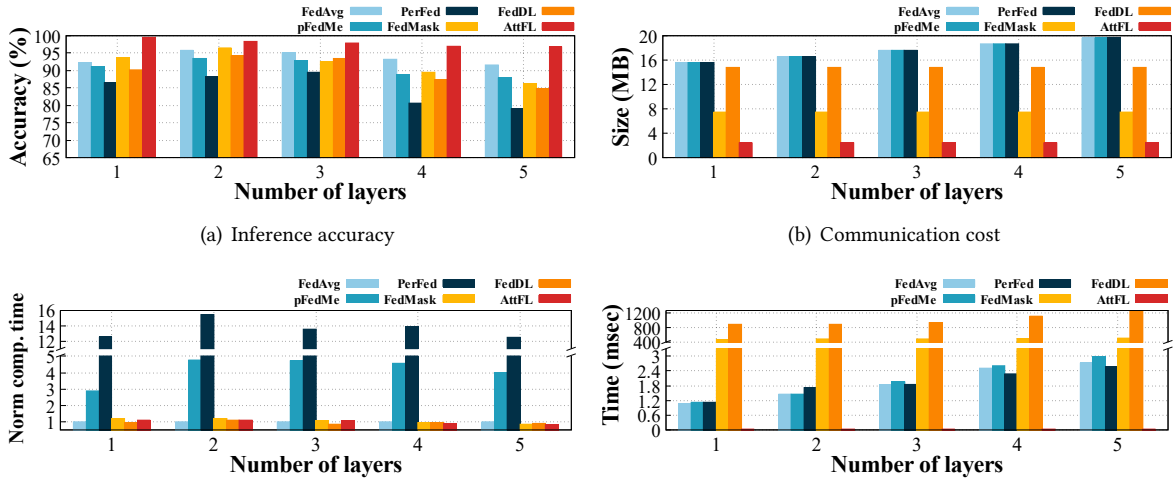
Fig. 6. [Application 1] Inference accuracy, communication cost (i.e., amount of data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, FedDL, and *AttFL* with *increasing Bi-LSTM hidden units* using the *MIT-BIH dataset.*

- **PerFed** [21] is an optimization scheme for federated learning using gradient and Hessian via a meta-learning approach. It follows the same structure as FedAvg and allows for improved model personalization.
- **FedMask** [45] exploits one-shot pruning to learn a personalized and structured sparse local model. It maintains efficiency by sharing and updating only the global mask information to and from the server. Mask information required to maintain a locally personalized model are separately preserved apart from the global mask.
- **FedDL** [76] leverages a dynamic layer-sharing scheme that learns the similarity among local devices' model weights and aggregates weights at the server based on a model affinity-based user grouping approach.

5.4 Performance on Application 1: Physiological Data (ECG Signal) Analysis

We start our evaluations with the ECG signal analysis application using the MIT-BIH dataset. As the baseline deep learning model for the MIT-BIH dataset, as done in our motivational studies, we use a Bi-LSTM with time steps of 30, input size of 60, and varying number of hidden units. The same model and configuration are applied to all federated learning frameworks, and for all cases, we perform 50 federated learning rounds with a network of 47 local devices. At each round, we randomly select five of them to participate each with 5-epoch local training.

Figures 6 and 7 present the accuracy, communication cost (in data size transmitted from the local device) and the normalized (FedAvg as the baseline) computation time at local devices (per-epoch) and server for *AttFL* along with five existing federated learning frameworks. Similar to Section 3, in Figure 6 we vary the number of Bi-LSTM hidden layers and Figure 7 plots results for varying number of Bi-LSTM layers. We focus on comparing the performance of previously proposed schemes with *AttFL* given that most results were discussed in our motivational studies. Notice that unlike others, *AttFL* shows good performance in all aspects regardless of the model configuration (i.e., achieves high accuracy and low overhead). Especially in terms of server-side computation time, the optimizations presented in Section 4.3 lead to extremely efficient performance, suggesting that *AttFL* can scale to many local devices even with a single server. Through such results, we can notice that *AttFL* shows a much lower server-side computation cost compared to FedDL's $O(NlogN)$ time complexity and FedMask, which requires the collection of mask bit information of all layers in the local device to update the global



(a) Inference accuracy

(b) Communication cost

(c) Normalized local device computation time (per-epoch)

(d) Server-side computation time

Fig. 7. [Application 1] Inference accuracy, communication cost (i.e., amount of data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, FedDL, and *AttFL* with *increasing Bi-LSTM's layers* using the *MIT-BIH dataset.*

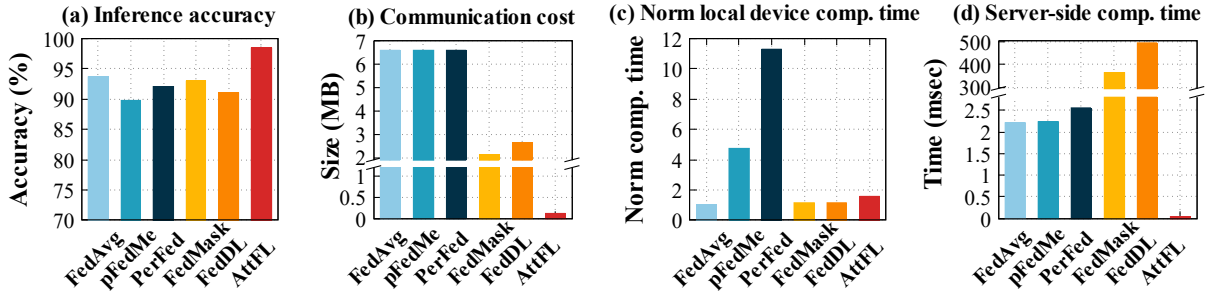
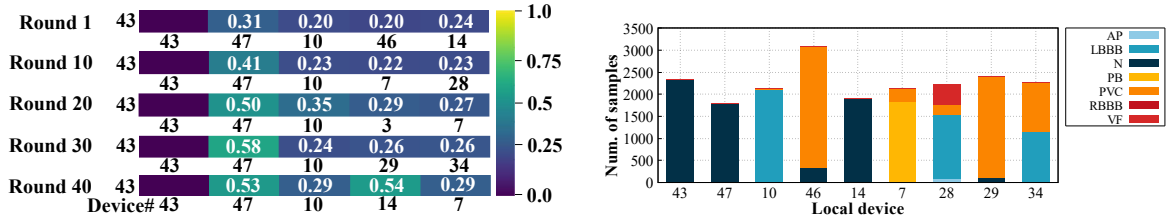


Fig. 8. [Application 1 - DeepSense baseline] Inference accuracy, communication cost (i.e., amount of data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, FedDL, and *AttFL* with the *DeepSense network* as the baseline model using the *MIT-BIH dataset.*

mask bit. By observing Figure 8, where we plot the four performance metrics when using the DeepSense network as the baseline model, we can also observe similar performance trends. Note that despite DeepSense using a CNN for input processing, we configured the model to take time series inputs directly in its initial convolution layer. Looking at the plots, we can observe that *AttFL* offers high classification accuracy at low computation and communication overhead: confirming that *AttFL* can be applied to various baseline model architectures and show improved performance over existing federated learning frameworks.

Next, we take a deeper look on the performance of *AttFL* and especially present details on the feature aggregation performance at the server. As an example, we focus on the cosine similarity heatmap between local devices sharing the attention feature map for three devices (#10, #43, and #47), and present their similarity scores computed for five federated learning rounds (out of 50) where all three devices participated in. Figure 9(a) illustrates the similarity at different rounds on device 43's perspective. Note that all devices besides #10, #43, and #47 followed



(a) Cosine similarity heatmap of the device #43 in MIT-BIH dataset

(b) Data class distribution on each local device in Figure 9(a)

Fig. 9. [Application 1] Example of local device grouping process executed at the *AttFL* server for *MIT-BIH dataset.*

the random participant selection mechanism; thus, not all nodes are trained together in the presented rounds. As the figure shows, these three devices participated together in the 1st, 10th, 20th, 30th and 40th update rounds. Here, we can notice that while the cosine similarity of devices 43 and 47 started low (i.e., 0.31), at round 40, the similarity score reached 0.53. Similar phenomena can be observed for device 14 (0.24 in the first round to 0.54 at round 40). We point out that #14 was not one of the devices we fixed for the experiment, but appeared in several training rounds (based on the random participant selection process); thus, *AttFL* successfully captured its similarity with device #43. This means that the server, when aggregating data, noticed that these devices were dealing with similar data classes. Thus, sharing information between these devices could improve the respective local model performance. Figure 9(b) confirms this observation in the label-space, where we plot the distribution of cardio pattern classes (presented as seven different labels) observed for the local devices observed in Figure 9(a). We can notice that devices 14, 43, and 47 all deal with normal heart beat cases (label 'N'); confirming that *AttFL* successfully groups local devices dealing with common data classes.

On the other hand, when observing cosine similarity score changes for device 10, we can notice that the score is constantly low (i.e., 0.20 in round 1 and 0.29 in round 40). This suggests that device 43 and device 10 do not share statistically relevant information, and we confirm this using Figure 9(b) by noticing that all of the data that device 10 deals with are left bundle branch block (LBBB) arrhythmia types, different from the data that device 43 is experiencing. Overall, these results together show that *AttFL* successfully identifies devices with statistical similarity and well-exploits such similarities to achieve high classification accuracy with low overhead.

5.5 Performance on Application 2: Human Activity Recognition

The second application we test for is a human activity recognition task, a common application used to identify different daily activities as part of various chronic physical disorder monitoring protocols and also for continuous health management. The HAR-UCI dataset that we use consists of time-series samples captured from the smartphone IMU sensor, and we use a Bi-LSTM with a time step of 18 and input size of 64. A total of 300 local devices take part in the network and we randomly select five devices to participate at each federated learning round. All other experimental parameters are kept the same as our previous experiment. By nature, this human activity recognition task is similar to the ECG analysis task given that it involves time-series data. However, compared to ECG, IMU data includes time-synchronized data sequences captured from multiple channels, which can complicate the aggregation and device similarity matching process.

As Figure 10 shows, *AttFL* achieves a high inference accuracy of 94.21%. Note that FedMask and FedDL were validated for HAR datasets in their original work and showed high inference performance. However, CNN architectures were used in the original work by representing snippets of time-series data as images. In this experiment, we keep the original time-series *as is* and test with a model suitable for such data types, as not in all cases can we have accurate image representations of time-series samples and the operations for transforming

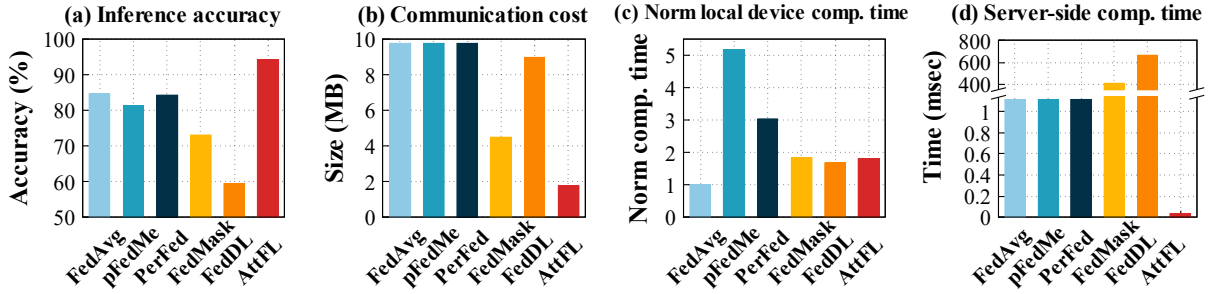
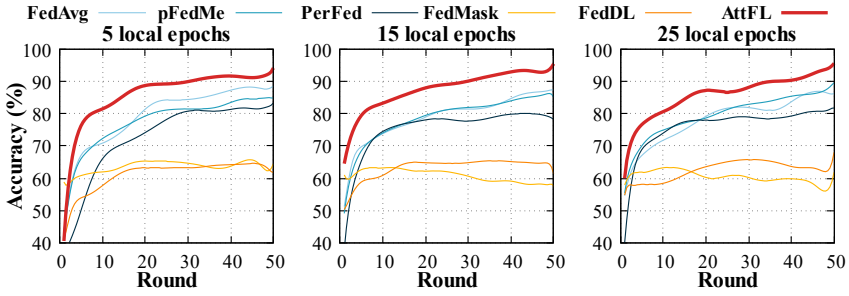
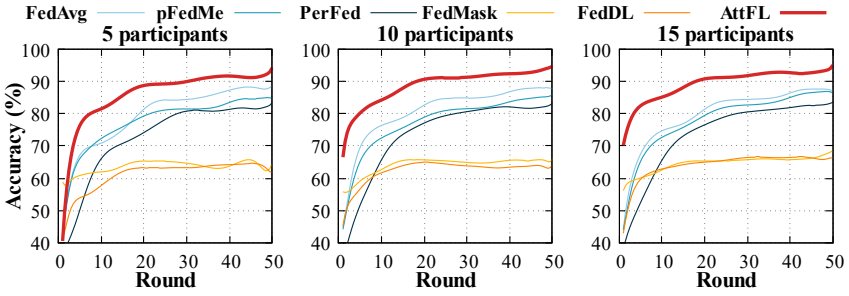


Fig. 10. [Application 2] Inference accuracy, communication cost (i.e., amount of data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, FedDL, and *AttFL* with a *Bi-LSTM* as the baseline model using the *HAR-UCI dataset.*



(a) Local epoch variation



(b) Increasing number of participants

Fig. 11. [Application 2] Inference accuracy trends with increasing rounds for varying local epochs and number of participants for FedAvg, pFedMe, PerFed, FedMask, FedDL, and *AttFL* using the *HAR-UCI dataset.*

time-series data to images can induce additional computational overhead. In any case, the performance of *AttFL* generally agrees with observations made for the ECG analysis task by exhibiting the highest accuracy and lowest overhead. While *AttFL* shows slightly higher local computation overhead compared to FedDL, this added computation (mostly caused from attention feature map computing) results in a significantly high accuracy.

To examine the results in different dimensions, in Figures 11(a) and 11(b), we plot the overall accuracy observed at each federated learning round for three epoch configurations (5 participants) and with varying number of participants (5 epochs), respectively. Notice that with increasing local epochs (i.e., more local training), the

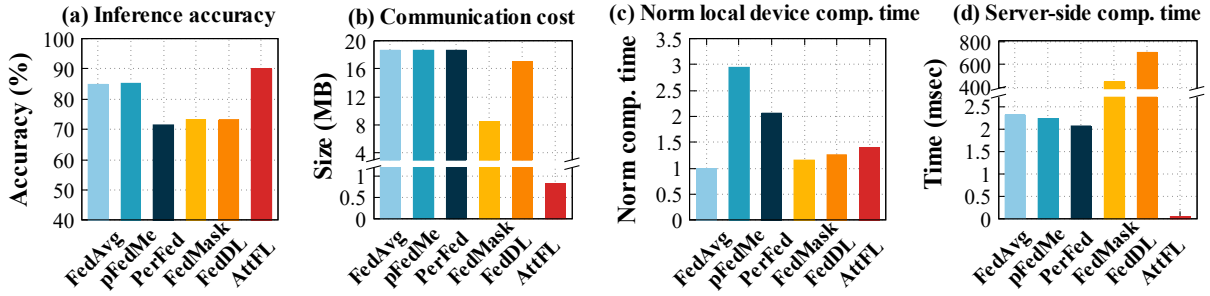


Fig. 12. [Application 2 - DeepSense baseline] Inference accuracy, communication cost (i.e., amount of data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, FedDL, and *AttFL* with the *DeepSense network* as the baseline model using the *HAR-UCI dataset.*

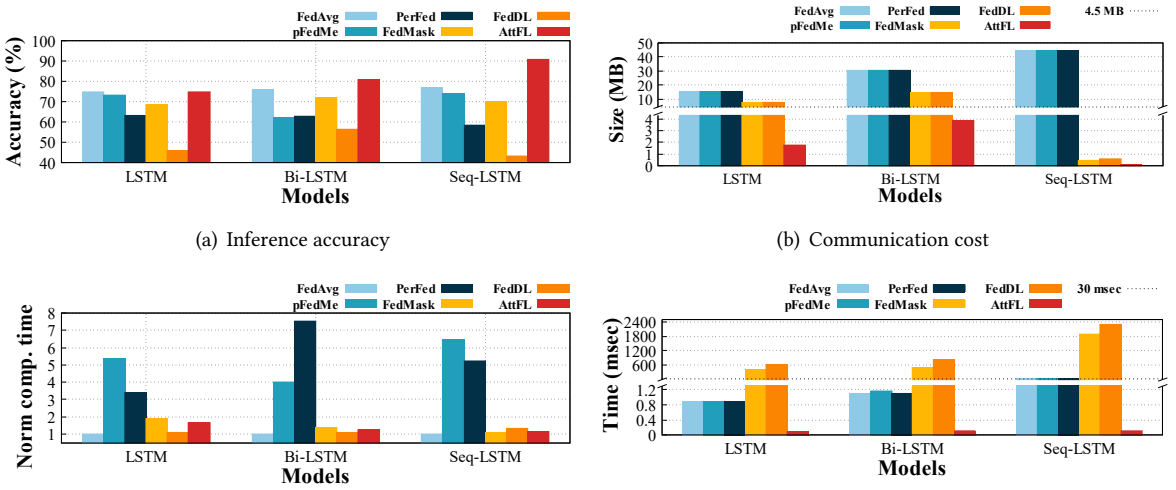
accuracy converges more quickly to the maximum and with a slight increase in accuracy. Similar convergence trends can also be seen with more participating devices since more knowledge can be aggregated per training round. One interesting observation we make here is that since *AttFL* updates local models via attention module convolution layer weight updates, the models themselves are updated "indirectly" and the changes need to eventually propagate to the baseline DNN for it to generate more accurate/updated inference results. Our evaluations on examining the performance for different training epochs show that even with a small number of training operations, the baseline model parameters/weights are updated sufficiently to generate improved inference results. Comparatively, for all cases, we see that *AttFL* shows quicker convergence compared to other approaches, suggesting that, despite the data complexity, *AttFL* effectively identifies local device similarity and shares meaningful model features for improved model accuracy.

To validate the effectiveness of *AttFL* across different baseline architectures, we switch the baseline model to the DeepSense network in Figure 12 and observe the four performance metrics. As with what we observed for the first application scenario, results here also agree with the trend observed with the Bi-LSTM baseline.

5.6 Performance on Application 3: Audio Data Analysis

As the final application, we select an audio data processing scenario. A number of IoT and mobile applications exploit audio data processing to implicitly capture and understand user intentions (even emotions), or gather environmental characteristics [44, 56, 72, 84]. In this work, we use the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset [56]. Consisting of 7,356 files (total size: 24.8 GB), the RAVDESS dataset holds speech and song data of different emotions. We perform classification for the eight target emotions included in the speech data (neutral, calm, happy, sad, angry, fearful, surprise, and disgust) of this dataset.

With this dataset, we examine the performance for three different baseline LSTM model variants: LSTM, Bi-LSTM, and Seq-LSTM, and later present an additional experiment with a CNN baseline model. Note that the Bi-LSTM is the same model used in our evaluations until now and the LSTM is a single direction version of the Bi-LSTM. The Seq-LSTM employs four sequencer blocks consisting of Bi-LSTMs to achieve memory-efficient and global mixing of spatial information using a multi-layer perception (MLP) for channel-mixing. This model enables parallel processing at the top/bottom and left/right directions, which improves its accuracy and efficiency due to its reduced sequence length and yields a spatially meaningful receptive field. We adopt the vanilla sequencer and use a batch size of 16, embedding dimensions of 128, 192, 192, and 192, hidden units of 48, 96, 96, and 96 for each sequencer block, respectively, and the SGD algorithm with a learning rate of 1e-2 to 1e-3. All other parameters are kept identical to the original work [75]. For all cases, we perform federated learning for 50 rounds with 5-epoch



(a) Inference accuracy

(b) Communication cost

(c) Normalized local device computation time (per-epoch)

(d) Server-side Computation time

Fig. 13. [Application 3] Inference accuracy, communication cost (i.e., amount of data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, FedDL, and *AttFL* with *three different LSTM baseline models* using the *RAVDESS speech dataset*.

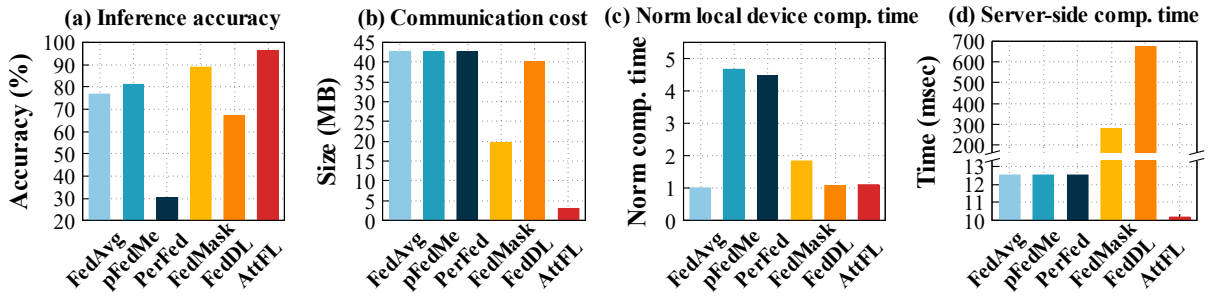


Fig. 14. [Application 3] Inference accuracy, communication cost (i.e., amount of data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, FedDL, and *AttFL* with *ResNet18* using the *RAVDESS speech dataset*.

local training. Among the 240 local devices (considering the 10-sharded components), five are selected randomly to participate at each round. Note that the LSTM offers a light-weight, yet, effective architecture for time-series data processing; thus, we examine the performance of *AttFL* and different federated learning models with its variants. As aforementioned, while heavier DNNs such as the Transformer can potentially show higher accuracy, their computation complexity still limits their practical use in embedded applications.

Figure 13 plots the accuracy, communication overhead, and local device (normalized) / server-side computation latency for the three LSTM-based baseline models and different federated learning schemes. As the plots show, *AttFL* achieves high accuracy for all baseline DNNs with efficient communication and computational overhead. It is interesting to see that despite Seq-LSTM's complexity, with *AttFL*, an extremely high accuracy (~90%) can be achieved with little communication and server computational overhead. Nevertheless, due to the complexity of

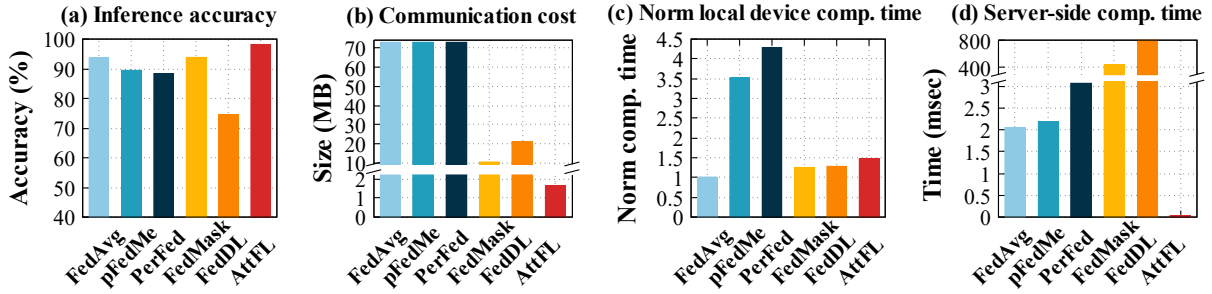


Fig. 15. [Application 3 - DeepSense baseline] Inference accuracy, communication cost (i.e., amount of data exchanged between a local device and server), and computation time at the local device and server for FedAvg, pFedMe, PerFed, FedMask, FedDL, and *AttFL* with the *DeepSense network* as the baseline model using the *RAVDESS speech dataset*.

the Seq-LSTM, the local processing overhead is relatively high compared to other baseline DNNs (still performs the best compared to other federated learning frameworks).

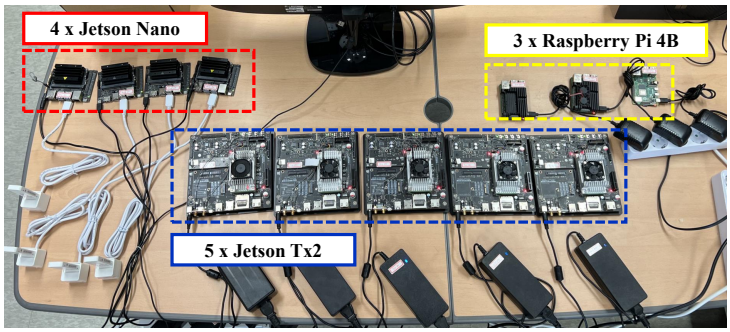
As the next experiment configuration with the RAVDESS dataset, we examine the performance of *AttFL* with a ResNet-18 baseline model targeting the same audio processing task. Note that such CNN-based models are often used for time-series acoustic data analysis by converting audio clips to spectrogram form and considering them as images (representing a target time frame). The ResNet-18 model we use for this experiment takes these images as input and consists of five convolution blocks and five fully-connected layers. For effective evaluations, based on the observations from respective original work, we applied FedMask to 50% of the layers, and FedDL exploits to 75% of the model's layers. Other experimental configurations are kept identical to our previous experiments. Results presented in Figure 14 show that *AttFL* achieves a high inference accuracy of 96.19%, outperforming all other alternative frameworks even with a completely different neural network architecture. Furthermore, as Figure 15 confirms, where we apply the DeepSense network as the baseline model, *AttFL* is capable of extending its capabilities towards CNN-based architectures. Note that while the DeepSense model was configured to directly use time-series data input for the first two experiments with ECG and human activity recognition data, for the RAVDESS dataset, the inputs were provided as spectrogram images, similar to the ResNet-18 model case. Nevertheless, given that frameworks such as FedMask and FedDL were extensively tested with CNNs, they show performance improvement over the LSTM-based cases. For the other three metrics we present, we can notice that *AttFL*'s performance agrees with the observations drawn until now, generally showing the lowest overhead.

Overall, our results gathered from three different applications (and various configurations) show that *AttFL* is capable of supporting efficient federated learning to support high application-level accuracy with minimal (or manageable) overhead for diverse baseline DNN architectures.

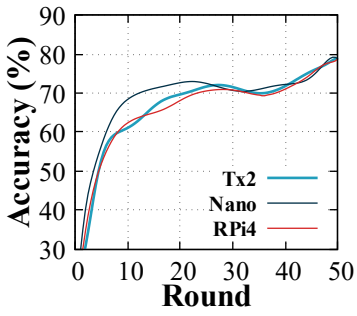
5.7 Performance on Embedded Platform Testbed

Finally, we validate the performance of *AttFL* on real platforms using a heterogeneous embedded computing platform testbed consisting of three Raspberry Pi 4B devices, four NVIDIA Jetson Nanos (5W mode), and five NVIDIA Jetson TX2 platforms connected via Wifi to a local server (c.f., Fig. 16(a)). This heterogeneous testbed offers us a chance to observe the performance of embedded platforms of different computational capabilities: from widely used resource-limited devices to high-end embedded GPU platforms.

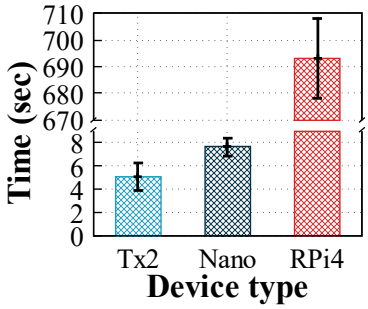
We use the audio data processing application discussed in Section 5.6 for our testbed experiments. Specifically, each device is pre-loaded with data from the RAVDESS dataset while separating the data with respect to different subjects. Specifically, the data from 24 study participants are split into 12 pairs and each pair's data is loaded to



(a) Experimental environment



(b) Classification accuracy



(c) Per-epoch local computation time

Fig. 16. Heterogeneous embedded platform testbed consisting of three Raspberry Pi 4Bs, four NVIDIA Jetson Nanos, and five NVIDIA Jetson TX2s with testbed-based experimental results (classification accuracy over time and per-epoch training time) for the *RAVDESS speech dataset* using the *Bi-LSTM model*.

one device hosting two (logical) local device instances (no sharding applied). For the 24 local device instances, we perform 50 federated learning rounds and randomly select five local device instances to participate at each round each with 5-epoch local training. We use the Bi-LSTM as the baseline model in this experiment.

Figure 16(b) plots the accuracy of each platform type over federated learning rounds. As the plots show, *AttFL* enhances classification accuracy as more federated learning rounds are completed. Note that despite different processing powers, since the model re-distribution operations complete after all devices participating in the federated learning round complete their operations, model improvements are seen at similar times. This result also validates that the accuracy results we observed in Section 5.6 hold on real embedded platforms.

In Figure 16(c) we present the per-epoch latency observed for each of the three testbed platforms. Note that the local device latency results presented until now were normalized to FedAvg's performance to assure that we do not make absolute number-based claims when using the CPU core emulation environment. Rather, using this result, we present readers with absolute latency measurements that can be used as reference. We can see that the Raspberry Pi 4 takes ~690 seconds to train the Bi-LSTM model with *AttFL* for five training rounds (i.e., epoch). Having more processing power and by exploiting its powerful on-board GPUs, the NVIDIA Jetson Nano and TX2 platforms show multiple orders of magnitude improvements in local computational latency, which suggests that *AttFL* can be a cost-effective solution for supporting federated learning on real mobile/embedded platforms.

6 DISCUSSION

We share a few interesting observations from our experiences in designing, implementing, and evaluating *AttFL*.

- **Trading off accuracy with efficiency:** An efficient federated learning framework should converge to a model with high inference accuracy while minimizing the communication and computational overheads. Thus, the goal of these frameworks is to identify essential portions of the model to share, while preserving these two goals. In *AttFL*, local devices, upon completing local training, share three types of information with the server: the attention feature map, convolution layer weights for Q, K, V, and attention output. One possible optimization is in suppressing the transmission of convolution layer weights for the attention output (purple block in Fig. 5). This component is relatively large in size as it re-connects the attention module to the baseline DNN. A simple experiment with the Bi-LSTM and the MIT-BIH dataset suggests that this optimization reduces the communication cost by nearly 5-fold (from 1.37 MB to 0.29 MB) with only 2% accuracy loss. While this work focuses on a configuration for achieving high accuracy, for applications where small accuracy loss is tolerable, such optimizations can be a reasonable option.
- **On-device model training:** A number of recent works have proposed schemes for efficient on-device model training [26, 79]. These works target to alleviate the memory constraints, a critical limitation for large-sized model training. Nevertheless, these mechanisms are still limited to a specific set of network architectures. For example, the current implementations for both Sage [26] and Melon [79] are not yet optimized for the training of RNN architectures. Resolving this and supporting diverse model training on mobile platforms remains an on-going task, but given the needs in many applications, we see this as an issue that will soon be addressed.
- **Energy usage on mobile and embedded platforms:** Our work along with many other federated learning frameworks acknowledge the fact that mobile and embedded platforms are resource limited, both in terms of computational power and energy. Therefore, these frameworks target to minimize the communication cost and also limit the local computational burden. One point to note is that diverse platforms show different energy profiling results for the processing and networking components; thus, is challenging to propose a perfectly balanced solution that addresses the trade-off between these components. For example, while additional local computation can reduce the communication overhead, for some platforms, this can eventually lead to even more energy usage as the processor uses more power than the networking module; whereas, for other platforms, the opposite may hold. Eventually, this is a design choice that can be determined once the target hardware is defined, and we point out that achieving such adaptiveness in framework and/or parameter selection can be an interesting direction of future research.
- **Improved capabilities of recent mobile and embedded platforms:** Extending on our previous discussion, it is worth noting that the hardware and resource capabilities of recently introduced mobile/embedded platforms are not as tight (or scarce) as a decade ago. Specifically, the communication and resource capabilities of mobile and embedded computing platforms have improved over the years, which means that the *relative overhead* of transmitting a couple tens of MBs for federated learning may not be as significant as before. At the same time, we point out the fact that the quantity of these "capable" devices have also increased, with more users demanding high-quality application services. Adding a few MBs at each user (or client device) is not much of an issue any more, but at the server, where federated learning parameter aggregation takes place, the overall amount of data to process can increase quite quickly if clients do not optimize/minimize the amount of data transmissions. Furthermore, even on a single client's perspective depending on the frequency of federated learning usage (potentially with many applications requesting federated learning requests simultaneously), the small amounts of data transmissions can add up; thus, still requiring the need for a more optimized solution that reduces data exchange overhead for federated learning.
- **Possibilities of "left-out clients" in the federated learning process:** As in other federated learning frameworks, at each federated learning round, *AttFL* randomly selects a subset of clients to participate. In this

process, there is a chance (due to the nature of random selection) that some clients are left out throughout the process. In this case, these left-out clients will not be able to benefit from the federated learning operations. Especially for personalized federated learning systems, such as *AttFL*, FedMask, FedML, not being able to participate in a federated learning round would mean that the server will have no knowledge on in input/model characteristics of the specific client. While this is a common limitation for all personalized federated learning protocols, an interesting direction of future research is to identify an improved baseline model for such devices. Specifically, while the server identifies device-suited model parameters in personalized federated learning, we can design them to also learn a global/generic model simultaneously, that aggregates information from all its clients. Later, this improved model (while not personalized) can be shared with unfortunate clients that were left-out in the federated learning process. Again, we point out that the chances of such cases happening may be slim, but a fall-back scheme as such can be an interesting direction of future research.

7 CONCLUSION

This work presents *AttFL* as a personalized federated learning framework for mobile and embedded sensing applications. *AttFL* is designed so that local devices (and their applications) can enjoy the performance of personalized local DNN operations with minimal computational and communication overhead, and is flexible enough to be adopted to various DNNs that are specialized for time-series data analysis. By exploiting a set of attention modules appended to the baseline DNN, *AttFL* effectively abstracts the input data features and underlying model parameters. We validate the performance of *AttFL* using three mobile/embedded sensing application scenarios via CPU core-based emulations and a heterogeneous embedded platform testbed to show that *AttFL* outperforms previous state-of-the-art federated learning frameworks in terms of both accuracy and overhead reduction.

ACKNOWLEDGMENTS

This work was financially supported by the National Research Foundation of Korea (NRF) Grant funded by the Ministry of Science and ICT (No. 2021R1A2C4002380), the ITRC Program supervised by IITP (IITP-2022-2020-0-01461), the Ministry of Culture, Sports and Tourism and Korea Creative Content Agency (R2021040018), and the Ministry of Trade, Industry and Energy and KIAT through the International Cooperative R&D program under Grant (P0016150).

REFERENCES

[1] 2019. learning with privacy at scale. Accessed May. 14, 2022 [Online]. (2019). https://docs-assets.developer.apple.com/ml-research/papers/learning-with-privacy-at-scale.pdf

[2] Jungmo Ahn, JaeYeon Park, Sung Sik Lee, Kyu-Hyuk Lee, Heesung Do, and JeongGil Ko. 2023. SafeFac: Video-based smart safety monitoring for preventing industrial work accidents. *Expert Systems with Applications* 215 (2023), 119397. DOI: https://doi.org/10.1016/j.eswa.2022.119397

[3] Alham Fikri Aji and Kenneth Heafield. 2017. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021* (2017).

[4] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. *Advances in neural information processing systems* 30 (2017).

[5] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, Ortiz Reyes, and Luis Jorge. 2013. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the European symposium on artificial neural networks, computational intelligence and machine learning*. 437–442.

[6] Juan Sebastian Arteaga-Falconi, Hussein Al Osman, and Abdulmotaleb El Saddik. 2015. ECG authentication for mobile devices. *IEEE Transactions on Instrumentation and Measurement* 65, 3 (2015), 591–600.

[7] David Arthur and Sergei Vassilvitskii. 2007. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 1027–1035.

[8] Kasimir Aula, Eemil Lagerspetz, Petteri Nurmi, and Sasu Tarkoma. 2022. Evaluation of low-cost air quality sensor calibration models. *ACM transactions on sensor networks* 18, 4 (2022), 1–32.

[9] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. 2019. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. *Advances in Neural Information Processing Systems* 32 (2019).

[10] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. 2018. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*. PMLR, 560–569.

[11] Yetong Cao, Fan Li, Huijie Chen, Xiaochen liu, Li Zhang, and Yu Wang. 2022. Guard Your Heart Silently: Continuous Electrocardiogram Waveform Monitoring with Wrist-Worn Motion Sensor. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 3 (2022), 1–29.

[12] Matteo Ceriotti, Luca Mottola, Gian Pietro Picco, Amy L Murphy, Stefan Guna, Michele Corra, Matteo Pozzi, Daniele Zonta, and Paolo Zanon. 2009. Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment. In *IPSN*. IEEE, 277–288.

[13] Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. 2021. An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)* 12, 5 (2021), 1–32.

[14] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. 2018. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876* (2018).

[15] Jintai Chen, Xiangshang Zheng, Hongyun Yu, Danny Z Chen, and Jian Wu. 2021. Electrocardio panorama: Synthesizing new ecg views with self-supervision. *arXiv preprint arXiv:2105.06293* (2021).

[16] Yanjiao Chen, Baolin Zheng, Zihan Zhang, Qian Wang, Chao Shen, and Qian Zhang. 2020. Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions. *Comput. Surveys* 53, 4 (2020), 1–37.

[17] Hyunsung Cho, Akhil Mathur, and Fahim Kawsar. 2022. Flame: Federated learning across multi-device environments. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 3 (2022), 1–29.

[18] Nihad Karim Chowdhury, Muhammad Ashad Kabir, Md Muhtadir Rahman, and Sheikh Mohammed Shariful Islam. 2022. Machine learning for detecting COVID-19 from cough sounds: An ensemble-based MCDM method. *Computers in Biology and Medicine* 145 (2022), 105405.

[19] Don Kurian Dennis, Tian Li, and Virginia Smith. 2021. Heterogeneity for the win: One-shot federated clustering. In *International Conference on Machine Learning*. PMLR, 2611–2620.

[20] Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. 2020. FedGroup: Efficient clustered federated learning via decomposed data-driven measure. *arXiv preprint arXiv:2010.06870* (2020).

[21] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948* (2020).

[22] Biyi Fang, Xiao Zeng, and Mi Zhang. 2018. NestDNN: Resource-Aware Multi-Tenant On-Device Deep Learning for Continuous Mobile Vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom)*. New Delhi, India, 115–127.

[23] MONA FLORES. 2020. Nvidia blogs: Nvidia Blogs: AI models for Mammogram Assessment. Accessed May. 14, 2022 [Online]. (Apr 2020). https://blogs.nvidia.com/blog/2020/04/15/federated-learning-mammogram-assessment/

[24] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.

[25] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 19586–19597.

[26] In Gim and JeongGil Ko. 2022. Memory-efficient DNN training on mobile devices. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*. 464–476.

[27] Tomer Golany, Gal Lavee, Shai Tejman Yarden, and Kira Radinsky. 2020. Improving ECG classification using generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 13280–13285.

[28] Jian Gong, Xinyu Zhang, Yuanjun Huang, Ju Ren, and Yaoxue Zhang. 2021. Robust inertial motion tracking through deep sensor fusion across smart earbuds and smartphone. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 2 (2021), 1–26.

[29] Stefan Gradl, Patrick Kugler, Clemens Lohmüller, and Bjoern Eskofier. 2012. Real-time ECG monitoring and arrhythmia detection using Android-based mobile devices. In *2012 annual international conference of the IEEE engineering in medicine and biology society*. IEEE, 2452–2455.

[30] Yeting Guo, Fang Liu, Zhiping Cai, Hui Zeng, Li Chen, Tongqing Zhou, and Nong Xiao. 2021. PREFER: Point-of-interest REcommendation with efficiency and privacy-preservation via Federated Edge leaRning. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–25.

[31] Nils Y Hammerla and Thomas Plötz. 2015. Let's (not) stick together: pairwise similarity biases cross-validation in activity recognition. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*. 1041–1051.

[32] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[33] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Xinghua Zhu, Jianzong Wang, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. 2020. FedML: A Research Library and Benchmark for Federated Machine Learning. In *Conference on Neural Information Processing Systems (NeurIPS) Federated Learning Workshop*.

[34] Samuel Horvóth, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtárik. 2022. Natural compression for distributed deep learning. In *Mathematical and Scientific Machine Learning*. PMLR, 129–141.

[35] Sinh Huynh, Rajesh Krishna Balan, and JeongGil Ko. 2022. IMon: Appearance-Based Gaze Tracking System on Mobile Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 4, Article 161 (dec 2022), 26 pages. DOI: https://doi.org/10.1145/3494999

[36] Sinh Huynh, Rajesh Krishna Balan, JeongGil Ko, and Youngki Lee. 2019. VitaMon: measuring heart rate variability using smartphone front camera. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 1–14.

[37] Dias Issa, M Fatih Demirci, and Adnan Yazici. 2020. Speech emotion recognition with deep convolutional neural networks. *Biomedical Signal Processing and Control* 59 (2020), 101894.

[38] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, Raman Arora, and others. 2019. Communication-efficient distributed SGD with sketching. *Advances in Neural Information Processing Systems* 32 (2019).

[39] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, and others. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[40] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887* (2017).

[41] Jeonggil Ko, Jong Hyun Lim, Yin Chen, Rvăzvan Musvaloiu-E, Andreas Terzis, Gerald M. Masson, Tia Gao, Walt Destler, Leo Selavo, and Richard P. Dutton. 2010. MEDiSN: Medical Emergency Detection in Sensor Networks. *ACM Trans. Embed. Comput. Syst.* 10, 1, Article 11 (aug 2010), 29 pages. DOI: https://doi.org/10.1145/1814539.1814550

[42] Marzena Kryszkiewicz. 2014. The cosine similarity in terms of the euclidean distance. In *Encyclopedia of Business Analytics and Optimization*. IGI Global, 2498–2508.

[43] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 95–104.

[44] Nicholas D Lane, Petko Georgiev, and Lorena Qendro. 2015. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *ACM international joint conference on pervasive and ubiquitous computing*. 283–294.

[45] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. 2021. FedMask: Joint Computation and Communication-Efficient Personalized Federated Learning via Heterogeneous Masking. In *Proceedings of the 19th ACM SenSys*. 42–55.

[46] Ke Li, Ruidong Zhang, Bo Liang, François Guimbretière, and Cheng Zhang. 2022. Eario: A low-power acoustic sensing earable for continuously tracking detailed facial movements. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–24.

[47] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*. PMLR, 6357–6368.

[48] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[49] Xinyu Li, Yanyi Zhang, Jianyu Zhang, Moliang Zhou, Shuhong Chen, Yue Gu, Yueyang Chen, Ivan Marsic, Richard A Farneth, and Randall S Burd. 2017. Progress estimation and phase detection for sequential processes. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies* 1, 3 (2017), 1–20.

[50] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523* (2020).

[51] Hyeontaek Lim, David G Andersen, and Michael Kaminsky. 2019. 3lc: Lightweight and effective traffic compression for distributed machine learning. *Proceedings of Machine Learning and Systems* 1 (2019), 53–64.

[52] Bingyan Liu, Yifeng Cai, Ziqi Zhang, Yuanchun Li, Leye Wang, Ding Li, Yao Guo, and Xiangqun Chen. 2021. DistFL: Distribution-aware Federated Learning for Mobile Scenarios. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 4 (2021), 1–26.

[53] Sicong Liu, Bin Guo, Ke Ma, Zhiwen Yu, and Junzhao Du. 2021. AdaSpring: Context-adaptive and runtime-evolutionary deep model compression for mobile applications. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–22.

[54] Shengzhong Liu, Shuochao Yao, Jinyang Li, Dongxin Liu, Tianshi Wang, Huajie Shao, and Tarek Abdelzaher. 2020. Giobalfusion: A global attentional deep learning framework for multisensor information fusion. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 1 (2020), 1–27.

[55] Xin Liu, Yuang Li, Josh Fromm, Yuntao Wang, Ziheng Jiang, Alex Mariakakis, and Shwetak Patel. 2021. SplitSR: An end-to-end approach to super-resolution on mobile devices. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–20.

[56] Steven R Livingstone and Frank A Russo. 2018. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PloS one* 13, 5 (2018), e0196391.
[57] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. 2023. Multi-center federated learning: clients clustering for better personalization. *World Wide Web* 26, 1 (2023), 481–500.
[58] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).
[59] Zichen Ma, Yu Lu, Wenye Li, Jinfeng Yi, and Shuguang Cui. 2021. PFedAtt: Attention-based Personalized Federated Learning on Heterogeneous Clients. In *Asian Conference on Machine Learning*. PMLR, 1253–1268.
[60] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619* (2020).
[61] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. 2021. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems* 34 (2021), 15434–15447.
[62] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.
[63] George B Moody and Roger G Mark. 2001. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine* 20, 3 (2001), 45–50.
[64] Kazuya Ohara, Takuya Maekawa, and Yasuyuki Matsushita. 2017. Detecting state changes of indoor everyday objects using Wi-Fi channel state information. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies* 1, 3 (2017), 1–28.
[65] HyeonJung Park, Youngki Lee, and JeongGil Ko. 2021. Enabling Real-Time Sign Language Translation on Mobile Platforms with On-Board Depth Cameras. *Proc. ACM IMWUT* 5, 2, Article 77 (jun 2021), 30 pages.
[66] Jaeyeon Park, Hyeon Cho, Rajesh Krishna Balan, and JeongGil Ko. 2020. Heartquake: Accurate low-cost non-invasive ecg monitoring using bed-mounted geophones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 1–28.
[67] Karol J Piczak. 2015. Environmental sound classification with convolutional neural networks. In *2015 IEEE MLSP*. IEEE, 1–6.
[68] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971* (2017).
[69] Nafiul Rashid, Berken Utku Demirel, Mohanad Odema, and Mohammad Abdullah Al Faruque. 2022. Template matching based early exit cnn for energy-efficient myocardial infarction detection on low-power wearable devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–22.
[70] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems* 32, 8 (2020), 3710–3722.
[71] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*. PMLR, 9489–9502.
[72] Liu Sicong, Zhou Zimu, Du Junzhao, Shangguan Longfei, Jun Han, and Xin Wang. 2017. UbiEar: Bringing location-independent sound awareness to the hard-of-hearing people with smartphones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 1–21.
[73] Canh T Dinh, Nguyen Tran, and Josh Nguyen. 2020. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems* 33 (2020), 21394–21405.
[74] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. 2022. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8432–8440.
[75] Yuki Tatsunami and Masato Taki. 2022. Sequencer: Deep LSTM for Image Classification. *arXiv preprint arXiv:2205.01972* (2022).
[76] Linlin Tu, Xiaomin Ouyang, Jiayu Zhou, Yuze He, and Guoliang Xing. 2021. FedDL: Federated Learning via Dynamic Layer Sharing for Human Activity Recognition. In *Proceedings of the 19th ACM SenSys*. 15–28.
[77] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[78] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, and others. 2021. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917* (2021).
[79] Qipeng Wang, Mengwei Xu, Chao Jin, Xinran Dong, Jinliang Yuan, Xin Jin, Gang Huang, Yunxin Liu, and Xuanzhe Liu. Breaking the Memory Wall for On-Device Learning. In *MobiSys '22: The 20th ACM International Conference on Mobile Systems, Applications, and Services June 27 - July 1, 2022, Portland, Oregon*. ACM.
[80] Zheng Wang, Xiaoliang Fan, Jianzhong Qi, Chenglu Wen, Cheng Wang, and Rongshan Yu. 2021. Federated learning with fair averaging. *arXiv preprint arXiv:2104.14937* (2021).
[81] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. 2018. Gradient sparsification for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems* 31 (2018).
[82] Li Xiang, Huang Kaixuan, Yang Wenhao, Wang Shusen, and Zhang Zhihua. 2020. On the convergence of fedavg on non-iid data. In *Int. Conf. Learning Representations*.

[83] Mengwei Xu, Feng Qian, Qiaozhu Mei, Kang Huang, and Xuanzhe Liu. 2018. Deeptype: On-device deep learning for input personalization service with minimal privacy concern. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–26.

[84] Xuhai Xu, Ebrahim Nemati, Korosh Vatanparvar, Viswam Nathan, Tousif Ahmed, Md Mahbubur Rahman, Daniel McCaffrey, Jilong Kuang, and Jun Alex Gao. 2021. Listen2cough: Leveraging end-to-end deep learning cough detection model to enhance lung health assessment using passively sensed audio. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–22.

[85] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th international conference on world wide web*. 351–360.

[86] Shuochao Yao, Yiran Zhao, Huajie Shao, Dongxin Liu, Shengzhong Liu, Yifan Hao, Ailing Piao, Shaohan Hu, Su Lu, and Tarek F Abdelzaher. 2019. Sadeepsense: Self-attention deep learning framework for heterogeneous on-device sensors in internet of things applications. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 1243–1251.

[87] Mi Zhang, Faen Zhang, Nicholas Lane, Yuanchao Shu, Xiao Zeng, Biyi Fang, Shen Yan, and Hui Xu. 2020. Deep Learning in the Era of Edge Computing: Challenges and Opportunities. In *Book chapter in Fog Computing: Theory and Practice, Wiley*.

[88] Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari, and Salman Avestimehr. 2022. Federated Learning for Internet of Things: Applications, Challenges, and Opportunities. *IEEE Internet of Things Magazine (IEEE IoTM)* (2022).

[89] Pengyuan Zhou, Hengwei Xu, Lik Hang Lee, Pei Fang, and Pan Hui. 2022. Are you left out? an efficient and fair federated learning for personalized profiles on wearable devices of inferior networking conditions. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–25.

[90] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).