# Hermes: An Efficient Federated Learning Framework for Heterogeneous Mobile Clients

Ang Li[1], Jingwei Sun[1], Pengcheng Li[2]*, Yu Pu[2], Hai Li[1], Yiran Chen[1]
[1]Department of Electrical and Computer Engineering, Duke University
[2]Alibaba DAMO Academy
[1]{ang.li630, jingwei.sun, hai.li, yiran.chen}@duke.edu, [2]{landy0220@gmail.com, y.pu@@alibaba-inc.com}

## ABSTRACT

Federated learning (FL) has been a popular method to achieve distributed machine learning among numerous devices without sharing their data to a cloud server. FL aims to learn a shared global model with the participation of massive devices under the orchestration of a central server. However, mobile devices usually have limited communication bandwidth to transfer local updates to the central server. In addition, the data residing across devices is intrinsically statistically heterogeneous (i.e., non-IID data distribution). Learning a single global model may not work well for all devices participating in the FL under data heterogeneity. Such communication cost and data heterogeneity are two critical bottlenecks that hinder from applying FL in practice. Moreover, mobile devices usually have limited computational resources. Improving the inference efficiency of the learned model is critical to deploy deep learning applications on mobile devices. In this paper, we present Hermes – a *communication* and *inference-efficient* FL framework under data heterogeneity. To this end, each device finds a small subnetwork by applying the structured pruning; only the updates of these subnetworks will be communicated between the server and the devices. Instead of taking the average over all parameters of all devices as conventional FL frameworks, the server performs the average on only overlapped parameters across each subnetwork. By applying Hermes, each device can learn a *personalized* and *structured sparse* deep neural network, which can run efficiently on devices. Experiment results show the remarkable advantages of Hermes over the status quo approaches. Hermes achieves as high as 32.17% increase in inference accuracy, 3.48× reduction on the communication cost, 1.83× speedup in inference efficiency, and 1.8× savings on energy consumption.

## CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computing methodologies** → **Machine learning**.

* Corresponding author.


*ACM MobiCom '21, January 31-February 4, 2022, New Orleans, LA, USA*

ACM ISBN 978-1-4503-8342-4/22/01…$15.00
https://doi.org/10.1145/3447993.3483278

## KEYWORDS

Federated learning, Communication efficiency, Data heterogeneity, Personalization, Inference efficiency

**ACM Reference Format:**
Ang Li[1], Jingwei Sun[1], Pengcheng Li[2]*, Yu Pu[2], Hai Li[1], Yiran Chen[1]. 2022. Hermes: An Efficient Federated Learning Framework for Heterogeneous Mobile Clients. In *The 27th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '21), January 31-February 4, 2022, New Orleans, LA, USA*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3447993.3483278

## 1 INTRODUCTION

More and more intelligent applications based on deep neural networks (DNNs) [28] have been developed for mobile devices, such as image recognition [56], video analytics [42], object detection [34], etc. The key to achieving the full promise of these applications is to train DNNs with a massive amount of data. Thus, when training a DNN locally on a mobile device with only limited data, it is unlikely to obtain desirable inference accuracy. A straightforward solution is to transmit data from many mobile devices to a central server and conduct centralized training, which raises severe privacy preservation concerns considering the possibility of untrusted servers and/or insecure communication.

Recently, federated learning (FL) [36] emerges as a popular distributed machine learning paradigm for its advances in addressing the above privacy concerns and solving the problem of data silos [30]. In FL, a shared global model is trained in a decentralized manner, under the orchestration of a central server. Specifically, in every communication round, each participating device downloads the current model from the central server and then computes an updated model locally with its own data. These locally trained models are then aggregated by the central server, i.e., averaging weights. Finally, a single consolidated and improved global model is sent back to devices.

With the prevalence of FL, the inherently non-independent identical (non-IID) data distribution across devices becomes a major concern. For example, Google applies FL to make emoji predictions [41] in Gboard mobile keyboards. When mobile clients are tied to particular geo-regions, the distribution of data labels can be significantly different across clients – certain emojis are used by one demographic but not others. Prior works [27] have demonstrated that the global model learned via FL cannot generalize well for all devices. Communication cost is another critical bottleneck for applying FL to mobile devices, as the communication resource is either very limited or expensive [30]. Therefore, such data heterogeneity (i.e., non-IID) and communication cost are two critical bottlenecks to hinder the development of FL.

**Status Quo and Limitations.** Many studies have been conducted to mitigate the data heterogeneity through FL personalization. These approaches can be divided into different categories, including fine-tuning [4, 12, 23, 25, 39, 51], multi-task learning [46], and contextualization [35]. Most of these techniques involve two separate steps where a global model is learned collaboratively in the first step, and then the global model is fine-tuned to the personalized model by each device using its local data. Such two-step methods inevitably incur extra computational overhead [23].

Alleviating the communication cost in FL has also been studied. The common idea is to compress the communication between devices and the central server: (1) *quantization* methods [2, 6, 9, 19, 24, 44, 54, 55] aim to reduce the bit number of elements in the transferred data; (2) *sparsification* methods [1, 11, 21] transmit only a subset of elements of the communicated data; and (3) *hybrid* methods [5, 22, 33, 47] combine quantization and sparsification.

However, very few approaches have been proposed to address the data heterogeneity and communication efficiency simultaneously. LG-FedAvg [32] combines local representation learning and global federated training, which needs two-step learning to achieve personalization. In particular, LG-FedAvg splits the model parameters into two parts: local parameters (i.e., the first several layers) and global parameters (i.e., the last several layers). In the first step, the devices update and communicate the whole model with the central server following FedAvg [36] scheme. In the second step, the devices update the whole model but only communicate the global parameters with the central server. In doing so, each device can learn personalized local parameters for feature extraction. However, LG-FedAvg can only reduce the communication in the second step, and the partition of the model is performed in a heuristic way rather than a data-driven manner, which may lead to sub-optimal performance. Furthermore, LG-FedAvg is evaluated in an *unrealistic* FL setting, where each device owns sufficient training data (i.e., 200 images/class of MNIST and CIFAR-10). Under such a setting, a device can obtain good accuracy through local training, which indeed contradicts the motivation of FL. For instance, the classification accuracy of a locally trained model by each device can achieve 97.17% on MNIST, while its counterpart using LG-FedAvg [32] increases the performance merely 1.49%. HeteroFL [10] adaptively allocates submodels to devices based on their varying computation and communication capabilities. Each device only trains and communicates its submodel, and hence both communication and computation efficiency in federated training can be improved. But the submodel is determined by the computation capability of each device rather than the local data, which makes the submodel allocation less flexible. SFSL is a secure federated submodel learning framework for recommender system [40], where the submodel can be identified by a user's historical data on an e-commerce platform. Each device only needs to train and transmit the submodel, and hence the communication and computation efficiency can be improved in FL. However, this specific federated submodel learning scheme is designed for the recommender system with special input data structure, SFSL may not be applicable to other applications due to varying models and input data formats. More importantly, none of the existing approaches take the computation efficiency of inference into account when designing the FL framework, which is a crucial factor impeding the deployment of FL on resource-constrained mobile devices.

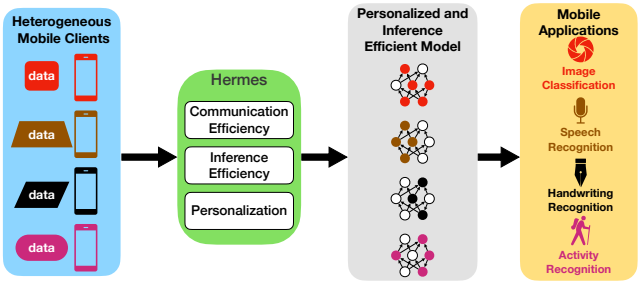



**Figure 1: The high-level view of Hermes.**

**Challenges and Our Solutions.** In this work, we propose Hermes, a FL framework that can simultaneously (1) reduce the *communication cost*, (2) improve the *computation efficiency* for inference, and (3) learn a *personalized* model for each participating device. The key idea of Hermes is that, instead of optimizing the base neural network in conventional FL frameworks, each device will train to find a subnetwork that can generalize well for its local data and only the parameters of the subnetwork will be communicated between devices and the server. The key intuition behind Hermes is inspired by the "lottery ticket hypothesis" [14], which finds that an optimal substructure of the original network can be identified through pruning and such a subnetwork can reach similar accuracy as the original network. In particular, the process of finding such a subnetwork on a device is driven by its local data, and the devices holding similar local data will share similar subnetwork architectures and more overlapped parameters. By exploiting the subnetwork, both the communication efficiency and inference efficiency can be significantly improved in Hermes due to the compact size of the subnetwork. In addition, the subnetwork should be able to learn representations embedded in local data, such that the personalized subnetwork can achieve desired inference accuracy on local data. Therefore, Hermes is able to effectively achieve personalization while jointly improving the efficiency of communication and inference.

However, the design of Hermes involves two key technical challenges. (i) The first challenge lies in designing a local training method that enables each device to learn a subnetwork that can embed the personalized information of local data while jointly improving communication and inference efficiency. (ii) Considering the non-IID data distribution across devices, the subnetworks are heterogeneous across devices, i.e., only parts of these subnetworks are overlapped across devices. However, in the conventional FL frameworks, the local model of each device shares the same architecture and the server can directly performs the aggregation on all elements in each local model. Unfortunately, directly applying the existing aggregation strategy will destroy the personalized information embedded in each subnetwork, potentially leading to significant degradation in terms of inference accuracy. Therefore, designing a specific aggregation strategy to maximally preserve the personalization property of each subnetwork poses another challenge.

To address the first challenge, Hermes employs a novel local training method which seeks a subnetwork within a base model of each client during each communication round. This is done by leveraging the *structured pruning*. Specifically, each device learns a *structured sparse* subnetwork, and then communicates only the parameters of the subnetwork with the server in each communication round. In doing so, both communication and inference efficiency can be significantly improved due to the compact size of the subnetwork. In addition, the structured pruning is performed on local data of each device, and hence the personalized information of local data is inherently embedded into the subnetwork. The details of addressing this challenge are presented in §3.2.

To tackle the second challenge, we design a *personalization-preserving aggregation* scheme for Hermes. In particular, the proposed aggregation method performs averaging on only intersections of subnetworks, i.e., overlapped parameters among each subnetwork, while keeping the remaining non-intersected elements unchanged. Such an aggregation strategy will prevent the personalized parameters (i.e., non-intersected elements across the subnetworks) of each subnetwork from being interfered by each other, and hence the personalization property can be effectively preserved. We describe the design of this aggregation strategy in §3.3.

Besides, we also consider a more *realistic* and *challenging* setting in this work, where each device holds limited data (e.g., 5 samples/class in image classification application). Thus, the devices cannot locally train a model with the desired inference accuracy due to insufficient data. As Figure 1 illustrates, given a set of mobile devices associated with heterogeneous data, Hermes enables each device to learn a personalized and inference-efficient model in a communication-efficient manner. Such a model can be deployed for various deep learning applications, such as image classification, human activity recognition, etc.

**System Implementation and Experimental Results.** We implemented Hermes and conducted a rich set of experiments to evaluate its performance. We applied Hermes to develop three representative deep learning applications on smartphones. These applications are developed based on three datasets that are widely used in computer vision and mobile sensing community. In addition, we also implemented four status quo approaches for comparisons, including FedAvg [36], Top-$k$ [1], Per-FedAvg [12], and LG-FedAvg [32]. Our results show that:

- Hermes outperforms the compared baselines, improving inference accuracy by 0.53%-32.17% and reducing communication cost by 1.92×-3.48×.
- At runtime, the structured sparse and personalized model that is learned using Hermes significantly outperforms those trained with the baseline methods. Hermes achieves as much as 1.83× speedup in inference latency, 70% reduction on memory footprint and 1.8× savings on energy consumption.

**Summary of Contributions.** To the best of our knowledge, Hermes is the first framework that achieves personalization while jointly improving the efficiency of communication and inference in FL. We also provide a theoretical convergence guarantee for our framework. Hermes contributes novel techniques that address the limitations in state-of-the-arts as well as the unique challenges in FL under non-IID settings. We believe that our work represents a significant step towards applying FL to heterogeneous mobile clients.

## 2 BACKGROUND AND MOTIVATION

We will first discuss the background and necessity of personalization of FL, and then present its communication bottleneck to show the limitation of the state-of-the-art approaches, motivating the design of the proposed Hermes as a result.

### 2.1 Background on Federated Learning

As numerous training data increasingly grows, FL is employed to enable the distributed learning across a widespread set of devices without local data sharing. A central server is often leveraged to coordinate the objective of the global learning task, which can be generally formulated as:

$$\min_{\boldsymbol{W}} f(\boldsymbol{W}) = \sum_{k=1}^{N} \frac{n_k}{n} F_k(\boldsymbol{W}_k) = \mathbb{E}_k \left[ F_k(\boldsymbol{W}_k) \right], \tag{1}$$

where $N$ denotes the number of devices, $n_k$ the number of data samples on device $k$, and $n = \sum_{k=1}^{N} n_k$ the number of all data samples. The local objective function $F_k(\mathbf{W_k})$ measures the empirical risk over local data distribution $D_k$, where $\mathbf{W_k}$ is the model parameters of device $k$.

As an example, one of the most widely applied FL methods, FedAvg [36] selects a subset of devices of size $K \ll N$ to conduct model training in each communication round. Each selected device trains its own local model using local data with the same learning rate and number of local epochs and the optimizer of stochastic gradient descent (SGD). Afterwards, each selected device transmits its local model update $\mathbf{W_k}$ to the central server whereas these local updates are averaged with weights $\{p_k\}$ and merged into the global copy of model with $\mathbf{W} = \frac{1}{K} \sum_{k=1}^{K} p_k \mathbf{W_k}$.

### 2.2 Data Heterogeneity Enforces Model Personalization

The primary incentive for devices to participate in FL is learning a global model with better performance than learning locally. However, existing studies [18, 29, 36] have demonstrated that the model trained locally oftentimes outperforms the globally learned model when data heterogeneity appears (i.e., non-IID data distribution across devices), which just neglects the motivation of FL. To show a data point, we investigated the impact of data heterogeneity on the conventional FL framework, by applying FedAvg to train a global model using CIFAR-10 for 400 communication rounds with each round involving 5 local epochs on each training device . In each communication round, we select 20 devices for training and each device holds two class of data with 20 samples per class, leaving the rest configurations set the same as in [29]. Our tests show that the globally trained model by FedAvg achieves the classification accuracy of 47.67%, while the model that is carefully trained locally can realize up to 65.44% of classification accuracy. Therefore, it is of a great challenge to train a good, one-fits-all model working consistently well for all devices with heterogeneous data, and we must enforce the model personalization for high efficient FL.

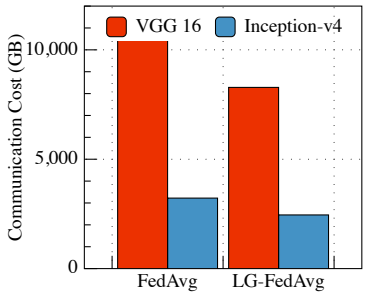



**Figure 2: Communication cost of FedAvg and LG-FedAvg.**

## 2.3 Communication Bottleneck

Another aspect to practically deploy FL is subject to the expensive communication overhead between the central server and training devices, even though tons of endeavors made. The outstanding method, LG-FedAvg [32] aims to improve communication efficiency as well as model personalization simultaneously. To demonstrate its significance, we experimentally compared FedAvg and LG-FedAvg for training on CIFAR-10 for 500 communication rounds, respectively, by measuring total data volume communicated between devices and the central server. In each communication round, we assume that there are 20 participating devices. We choose VGG16 [45] and Inception-v4 [49] as the default model architectures.

As Figure 2 shows, the total communication volume is as high as 10.59 TB and 32.22 TB data for FedAvg to train VGG16 and Inception-v4, respectively, incurring an unacceptable communication time overhead and hence power energy consumption for mobile services. Although reducing the communication cost by a significant amount to 8.28 TB and 2.45 TB for the two models, LG-FedAVG remains less than sufficient to be in good shape in practice. This work seeks to address this expensive communication efficiency issue.

# 3 DESIGN OF HERMES

## 3.1 Overview

In this work, we propose Hermes – a personalized FL framework that *jointly* improves the *communication* and *inference* efficiency. Instead of learning a shared global model like conventional FL frameworks, Hermes enables each participating device to obtain a personalized model.

Figure 3 illustrates the architecture of the proposed Hermes framework, which involves two key operations: *learning local subnetwork* on each device and performing the *subnetwork aggregation* on the central server. In particular, at the beginning, the central server initializes a dense network as the base model using random parameters and then distributes it to the participating devices.

To support simultaneously learning *personalized* models and reducing communication cost under data heterogeneity, we employ the structured pruning method to enable each device to learn a *heterogeneous* and *structured-sparse* subnetwork. Such a structure heterogeneity of the subnetwork is attributed to the non-IID data distribution across devices and embeds the personalized information of the local data distribution on each device. Specifically, in each communication round, a set of devices will be selected to

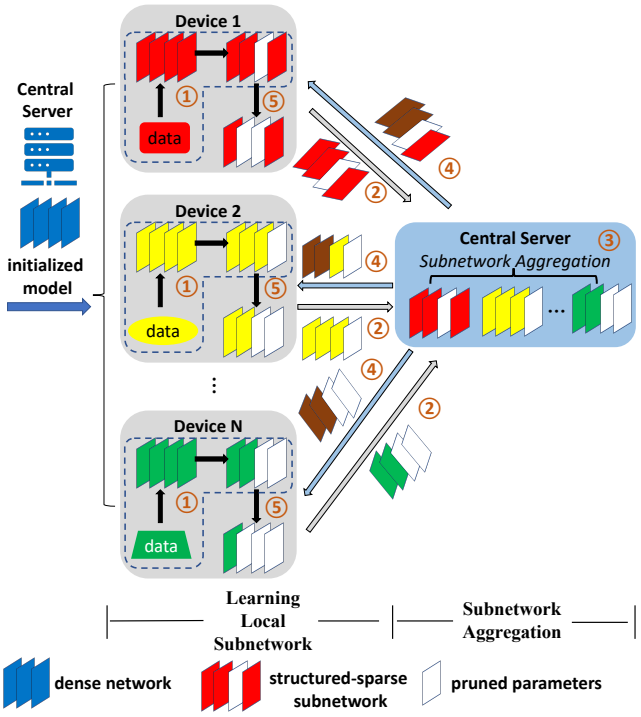



**Figure 3: The overview of the Hermes framework.**

participate in FL training. Instead of directly optimizing the local model, each device incorporates a structured sparsity regularization to learn a subnetwork when optimizing the local model (①). Only such subnetworks are transmitted from devices to the central server (②). Unlike communicating the local updates of the full dense model in FedAvg, the *communication efficiency* can be significantly improved due to the compact size of the subnetwork.

The central server performs the aggregation on the received subnetworks. However, the subnetworks are heterogeneous across devices and embed the personalized information of the local data distribution, applying existing aggregation strategy (e.g., FedAvg) to the subnetworks will destroy their personalization properties. Therefore, we design a specific strategy for aggregating the subnetworks such that shared knowledge can be learned across devices while the personalization of each subnetwork can be effectively preserved. Specifically, only the parameters that are intersected across the subnetworks of devices are averaged while keeping the remaining non-intersected parameters untouched (③). For example, as Figure 3 shows, only the first two parts of the subnetworks across *Device* 1, *Device* 2, and *Device N*, but the other parts of each subnetwork are not overlapped with each other. Hence, the aggregation is performed on only the first two parts (i.e., in brown color) of these three subnetworks, but the other parts of each subnetwork is not changed. Afterwards, the updated subnetworks will be distributed to each device (④). The above steps (①-④) repeat until reaching the predefined number of communication rounds.

Finally, each device will obtain a *personalized* and *structured-sparse* model (⑤). Due to the structured sparsity, the personalized

model runs more efficiently on each device than a dense model, thus significantly improving *computation efficiency* for inference.

**Notations.** Before diving into the details of learning the local subnetwork and the personalization-preserving aggregation, we first define the following notations used in this work. We denote $C = \{C_1, \ldots, C_N\}$ as $N$ available devices, where $C_k$ denotes the $k$th device; and $\mathcal{S}_c \subset C$ as a set of selected devices in each training round. Let $\boldsymbol{W}$ be the parameters of the base model on the global server, and $\boldsymbol{W}_k$ represent the local model parameters of $C_k$. We also use the superscript $T$, e.g., $\boldsymbol{W}^T$, to represent the model parameters learned in the $T$th round. Each device $C_k$ also learns a local mask $\boldsymbol{M}_k \in \{0,1\}^{|\boldsymbol{W}_k|}$, which indicates the subnetwork identified via applying the structured pruning. Therefore, $\boldsymbol{W}_k \odot \boldsymbol{M}_k$ that is a binary mask denotes the parameters of the corresponding subnetwork at device $C_i$. Given the data $D_k$ held by $C_k$, we split $D_k$ into the training data $D_k^{train}$, validation data $D_k^{val}$, and test data $D_k^{test}$.

## 3.2 Learn Subnetwork for Joint Efficiency and Personalization

Hermes differs existing FL frameworks in that each device learns a sparse subnetwork and communicates such a subnetwork with the central server only. The goal of seeking the sparse subnetwork is to jointly improve the communication and inference efficiency while achieving the personalization. Model pruning is a natural option to help each device find the subnetwork using the local data, because the key insight of pruning is to find the most significant parameters for accurately making inference on training data. In addition, the compact size of the subnetwork will be greatly helpful for improving the communication and computation efficiency. There are two different types of pruning techniques, i.e., *unstructured pruning* [17] and *structured pruning* [53]. Figure 4 shows the difference between these two pruning methods. Unstructured pruning is often operated on the parameter level, offering the high flexibility and generality for compression rate but requiring specific hardware supports to realize the improvement of computation efficiency. On the other hand, the structured pruning is usually performed in a channel-wise or filter-wise manner. Even though it is less flexible compared to unstructured pruning, it is very hardware-friendly for improving the computation efficiency. Therefore, we employ the structured pruning to learn the subnetworks during optimizing the local models on devices.

Without loss of generality, we consider a convolutional neural network as the local model on devices. When optimizing the local model using SGD, each device aims to achieve channel-wise and filter-wise sparsity for convolutional layers, row-wise and column-wise sparsity for fully connected layers. Suppose the local model $\boldsymbol{W}$ consists of $\boldsymbol{W}_{conv}$ convolutional layers and $\boldsymbol{W}_{fc}$ fully connected layers, the structured sparsity regularization for the local model $\boldsymbol{W}$ is formulated as:

$$
\begin{aligned}
R(\boldsymbol{W}) &= R_{conv}(\boldsymbol{W}) + R_{fc}(\boldsymbol{W}), \\
R_{conv}(\boldsymbol{W}) &= \sum_{l=1}^{W_{conv}} \Big( \sum_{f_l=1}^{F_l} \|\boldsymbol{W}^{(l)}_{f_l,:,:,:}\|_g + \sum_{ch_l=1}^{Ch_l} \|\boldsymbol{W}^{(l)}_{:,ch_l,:,:}\|_g \Big), \\
R_{fc}(\boldsymbol{W}) &= \sum_{l=1}^{W_{fc}} \Big( \sum_{row_l=1}^{Row_l} \|\boldsymbol{W}^{(l)}_{row_l,:}\|_g + \sum_{col_l=1}^{Col_l} \|\boldsymbol{W}^{(l)}_{:,col_l}\|_g \Big),
\end{aligned}
\tag{2}
$$

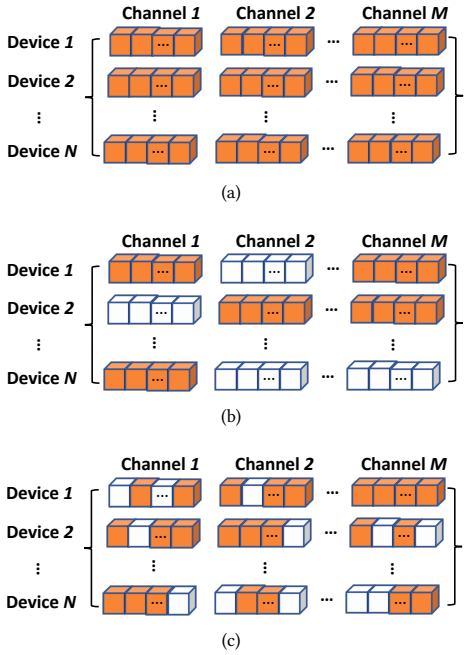



**Figure 4: Illustration of the difference between the structured pruning and unstructured pruning. (a) The parameter matrix representations of channels in a layer. (b) The effect of the channel-wise structured pruning, where the white and orange channels are the pruned ones and retained ones, respectively. (c) The unstructured pruning scheme.**

where $\|.\|_g$ is the group Lasso [53], $\|\boldsymbol{W}^{(g)}\|_g = \sqrt{\sum_{i=1}^{|\boldsymbol{W}^{(g)}|} (\boldsymbol{W}_i^{(g)})^2}$, $F_l$ the number of filters in the $l$th convolutional layer, $Ch_l$ the number of channels in the $l$th convolutional layer, $Row_l$ the number of rows in the $l$th fully connected layer, $Col_l$ the number of columns in the $l$th fully connected layer, and $|\boldsymbol{W}^{(g)}|$ the number of parameters in $\boldsymbol{W}^{(g)}$. After applying this structured sparsity regularization, the training loss for mask optimization is formulated as:

$$
F(\boldsymbol{W}) = F_D(\boldsymbol{W}) + \lambda R(\boldsymbol{W}) \tag{3}
$$

where $F_D(\boldsymbol{W})$ is the loss for the local data and $\lambda$ is the coefficient of the structured sparsity regularization. By optimizing the objective function in Equation 3 using local data, each device can derive $\boldsymbol{M}$ by identifying the zero-valued and non-zero valued parameters in $\boldsymbol{W}$. Specifically, the location that corresponds to the zero-valued or non-zero valued in $\boldsymbol{W}$ will be marked as 0 and 1 in $\boldsymbol{M}$, respectively.

Given the $T$th communication round, each participating device downloads the corresponding subnetwork $\boldsymbol{W}_k^T = \boldsymbol{W} \odot \boldsymbol{M}_k^T$, where $\boldsymbol{M}_k^T$ is the mask that is determined in the $(T-1)$th round using Equation 3. First, the device evaluates $\boldsymbol{W}_k^T$ on the local validation data $D_k^{val}$. If the accuracy of the current subnetwork $\boldsymbol{W}_k^T$ is better than a predefined threshold $acc_{threshold}$ and the current pruning rate $r_k^T$ does not reach the target pruning rate $r_{target}$, the device

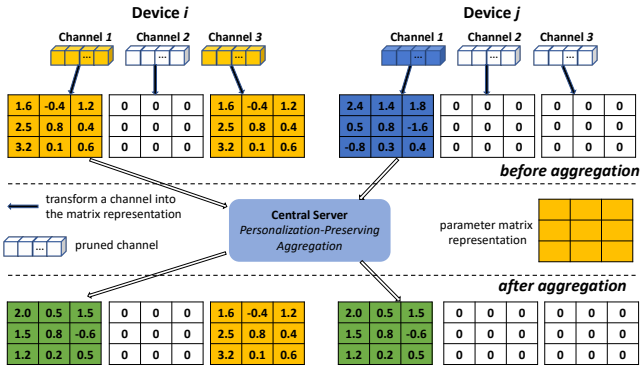



**Figure 5: Illustration of the personalization-preserving aggregation on the central server.**

will prune the low-magnitude weights of $\boldsymbol{W}_k^T$ using a fixed pruning rate $r_p$. After the structured pruning, the device can derive $\boldsymbol{M}_k^{T+1}$ which indicates the sparse structure for $\boldsymbol{W}_k^{T+1}$ and embeds the data-dependent features. Afterwards, the device performs mini-batch training for several epochs using $D_k^{train}$ based on $\boldsymbol{W}_k^T \odot \boldsymbol{M}_k^{T+1}$ and obtains the updated $\boldsymbol{W}_k^{T+1}$. In addition to sending the subnetwork parameters $\boldsymbol{W}_k^{T+1}$ to the server, each client also needs to transmit the corresponding binary mask $\boldsymbol{M}_k^{T+1}$ to the server. Such a binary mask helps the server identify the positions of pruned and remained parameters accordingly. Note that the binary mask only uses 1 bit to represent each model parameter, the additional communication cost of sending the binary mask is ignorable compared to transmitting the floating-point model parameters. Finally, the device transmits $\boldsymbol{W}_k^{T+1}$ and $\boldsymbol{M}_k^{T+1}$ to the server.

## 3.3 Personalization-Preserving Aggregation

In each communication round, the participating devices transmit the subnetworks to the central server, where the aggregation is performed on these subnetworks. However, existing FL methods (e.g., FedAvg) perform the aggregation on all parameters of each device's local model, and hence directly applying these aggregation strategies will inevitably disrupt the personalization of the subnetworks in Hermes.

Therefore, we design a novel aggregation strategy to preserve the personalized property of the subnetworks. Specifically, we propose a personalization-preserving aggregation scheme where only intersected parameters across the subnetworks are averaged while keeping the non-intersected parameters unchanged. Then, the central server sends the updated subnetworks back to devices accordingly. Figure 5 shows one example of performing the proposed aggregation method on one layer of two subnetworks, where each layer consists of three channels. As Figure 5 illustrates, only the first channel of *Device i*(in yellow) and *Device j*(in blue) are intersected, and hence the parameter of these two channels are averaged by the proposed aggregation scheme. Because the third channel of *Device j* is pruned, there is no intersection of the third channel between *Device i*(in yellow) and *Device j*(in white). Therefore, the third channel of *Device i* will not be changed. After the aggregation,

**Algorithm 1:** Training Algorithm of Hermes.

**Data:** $(D_1, \ldots, D_N)$ where $D_k$ is the local data on $k$th device
**Server Executes:**
initialize the global model $\boldsymbol{W}$
**for** *each round* $T = 1, 2, \ldots$ **do**
  $k \leftarrow \max(N \times K, 1)$ /* $N$ available devices, random sampling rate $K$ */
  $S_c \leftarrow \{C_1, \ldots, C_k\}$ /* the selected $k$ participating devices indexed by $k$ */
  **for** $C_k \in S_t$ ***in parallel*** **do**
    $\boldsymbol{W}_k^T = \boldsymbol{W}^T \odot \boldsymbol{M}_k^T$ /* the subnetwork of $C_k$ */
    $\boldsymbol{W}_k^{T+1} \leftarrow \text{ClientUpdate}(C_k, \boldsymbol{W}_k^T)$
  $\boldsymbol{W}^{T+1} \leftarrow (\text{aggregate } \{\boldsymbol{W}_k^{T+1}\})$ /* using the proposed personalization-preserving aggregation */
**ClientUpdate** $(C_k, \boldsymbol{W}_k^T)$**:**
$acc \leftarrow (\text{evaluate } \boldsymbol{W}_k^T \text{ on the local validation data } D_k^{val})$
**if** $acc > acc_{threshold}$ *and* $r_k^T < r_{target}$ **then** /* $r_k^T$ is the current pruning rate of $k$th client's model, $r_{target}$ is the target pruning rate */
  $\boldsymbol{M}_k^{T+1} \leftarrow (\text{prune } \boldsymbol{W}_k^T \text{ with the fixed pruning rate } r_p)$
$\mathcal{B} \leftarrow (\text{split local data } D_k^{train} \text{ into batches})$;
**for** *each local epoch* $i$ *from* 1 *to* $E$ **do**
  **for** *batch* $b \in \mathcal{B}$ **do**
    $\boldsymbol{W}_k^{T+1} \leftarrow \boldsymbol{W}_k^T \odot \boldsymbol{M}_k^{T+1} - \eta \nabla_{F_k}(\boldsymbol{W}_k^T \odot \boldsymbol{M}_k^{T+1}; b)$ /* $\eta$ is the learning rate, $F_k(\cdot)$ is the loss function */
return $\boldsymbol{W}_k^{T+1}$, $\boldsymbol{M}_k^{T+1}$ to server

the central server sends the updated parameters back to these two devices, where only the first channel is updated (in green) for both two devices but the other two channels keep the same. In doing so, the shared knowledge (e.g., the first channel of *Device i* and *Device j*) across devices is aggregated while the personalization information is retained (e.g., the third channel of *Device i*). Details of the training algorithm for Hermes are presented in Algorithm 1.

# 4 THEORETICAL ANALYSIS

We theoretically analyze the convergence rate of Hermes. The analysis shows that Hermes has strong theoretical guarantee of an exponential convergence rate, which is faster than the that of status quo. We first introduce some notations and terminologies that are used in the theoretical analysis. Then we elaborate the convergence analysis for Hermes. Note that the theoretical analysis is performed based on the one-step FL scheme like FedAvg, and hence the derived convergence rate cannot be directly compared to personalized FL methods that require additional local adaptations (e.g., Per-FedAvg [12]).

## 4.1 Notations and Terminologies

To make the convergence analysis, we formally describe the training process of Hermes. Given a communication round, the central

server first broadcasts the latest model $\boldsymbol{W}^t$ to all the participating devices. Afterwards, each device (e.g. the $k$-th) performs $U(\geq 1)$ steps of local training based on $\boldsymbol{W}_k^t = \boldsymbol{W}^t$. The local update is performed as:

$$\boldsymbol{W}_k^{t+i+1} \leftarrow \boldsymbol{M}_k^{t+i+1} \odot \boldsymbol{W}_k^{t+i} - \eta_{t+i} \nabla F_k^{t+i}(\boldsymbol{M}_k^{t+i+1} \odot \boldsymbol{W}_k^{t+i}), \quad (4)$$

where $t$ is the index of mini-batch, $\eta_{t+i}$ is the learning rate, and $\nabla F_k^{t+i}(\boldsymbol{W})$ represents $\nabla_{\boldsymbol{W}} F_k^{t+i}(\boldsymbol{W}, D_k^{t+i})$ where $D_k^{t+i}$ is a set of samples uniformly chosen from the local data. Finally, the server aggregates the local models (i.e., subnetworks) $\boldsymbol{W}_1^{t+U}, ..., \boldsymbol{W}_N^{t+U}$ to update the global model $\boldsymbol{W}^{t+U}$.

For the local mask of each device (say the $k$-th), it has the following important properties:

$$\|\boldsymbol{M}_k^{t+1}\|^2 \leq \|\boldsymbol{M}_k^t\|^2, \quad (5)$$

$$\boldsymbol{M}_k^{t+1} \odot \boldsymbol{M}_k^t = \boldsymbol{M}_k^{t+1}, \quad (6)$$

$$\|\boldsymbol{M}_A \odot \boldsymbol{M}_k^t\|^2 \leq \|\boldsymbol{M}_k^t\|^2. \quad (7)$$

In the convergence analysis, we temporally consider $\boldsymbol{M}$ as a set, which contains elements of 1s. We define $\boldsymbol{M}^t = \bigcup_{i \in \mathbb{S}_t} \boldsymbol{M}_i^t$, and $\boldsymbol{M}_A \ominus \boldsymbol{M}_B = \boldsymbol{M}_A - \boldsymbol{M}_A \cap \boldsymbol{M}_B$.

To facilitate the analysis, we introduce an abstract global model $\hat{\boldsymbol{W}}^t$ in each step of local training:

$$\hat{\boldsymbol{W}}^t = \sum_{i \in \mathbb{S}_t} p_i \boldsymbol{M}_i^t \odot \boldsymbol{W}_i^t, \quad (8)$$

where $p_i$ represents the weight of each local model during aggregation, and $\sum_{i \in \mathbb{S}_t} p_i = 1$.

Similarly, we also introduce an abstract local model for each device (say the $k$-th) defined as:

$$\hat{\boldsymbol{W}}_k^t = \boldsymbol{M}_k \odot \hat{\boldsymbol{W}}^t + (\boldsymbol{M}_k^t - \boldsymbol{M}^t) \odot \hat{\boldsymbol{W}}_k^{\hat{t}-1}, \quad (9)$$

where $\hat{\boldsymbol{W}}_k^t$ can be fetched and equals to $\boldsymbol{W}_k^t$ when $t \in \{mU | m = 0, 1, 2, ...\}$. If we can establish the convergence of Hermes on $\hat{\boldsymbol{W}}_k^t$, then the convergence of Hermes on $\boldsymbol{W}_k^t$ is guaranteed.

## 4.2 Convergence Analysis

We conduct the theoretical analysis to show that each personalized model in Hermes converges to the local optimum. Without loss of generality, we derive the convergence guarantee considering the case that all the devices participate in the aggregation, but the derived convergence guarantee can be easily extended to the scenario where only a subset of devices are involved. We split the whole training process into two phases: 1) **pruning phase** when the devices keep pruning during local training (i.e., $r_{target}$ is not reached yet in Algorithm 1); 2) **sparse phase** when all the devices have identified their personalized model structures without the need of further pruning.

We make the following assumptions on the local masks $M_k$ and local loss functions $F_k(\cdot)$. Assumption 1- 2 are common in convergence analysis of gradient descent [31]. Assumption 3 states that the intersections of model parameters across devices are induced by the similarity of their local data distributions. Assumption 4 bounds the value changing among gradients and model parameters, and Assumption 5 bounds the pruning rate.

Assumption 1. *$F_1, ..., F_N$ are L-smooth: $\forall \boldsymbol{V}, \boldsymbol{W}, F_k(\boldsymbol{V}) \leq F_k(\boldsymbol{W}) + (\boldsymbol{V} - \boldsymbol{W})^T \nabla F_k(\boldsymbol{W}) + \frac{L}{2}||\boldsymbol{V} - \boldsymbol{W}||^2$.*

Assumption 2. *$F_1, ..., F_N$ are $\mu$-strongly convex: $\forall \boldsymbol{V}, \boldsymbol{W}, F_k(\boldsymbol{V}) \geq F_k(\boldsymbol{W}) + (\boldsymbol{V} - \boldsymbol{W})^T \nabla F_k(\boldsymbol{W}) + \frac{\mu}{2}||\boldsymbol{V} - \boldsymbol{W}||^2$.*

Assumption 3. *The divergence of the overlapped gradients is bounded: $\forall i, j, ||\boldsymbol{M}_i^t \odot \nabla F_i(\boldsymbol{W}_i^t) - \boldsymbol{M}_j^t \odot \nabla F_j(\boldsymbol{W}_j^t)||^2 \leq B$.*

Assumption 4. *The value changing among gradients and parameters is bounded: $\forall i, j, M, Q \leq |\frac{<\boldsymbol{M} \odot \nabla F_i(\boldsymbol{W}_i^t), \boldsymbol{W}_j^t>}{||\boldsymbol{M} \odot \boldsymbol{M}_i^t||^2 <\nabla F_i(\boldsymbol{W}_i^t), \boldsymbol{W}_j^t>}| \leq K$.*

Assumption 5. *The pruning rate for each local training step is bounded: $\forall i, t, ||(\boldsymbol{M}_i^{t+1} - \boldsymbol{M}_i^t) \odot \boldsymbol{W}_i^t||^2 \leq P, ||(\boldsymbol{M}_i^{t+1} - \boldsymbol{M}_i^t) \odot \nabla F_i(\boldsymbol{W}_i^t)||^2 \leq G$.*

We define $\boldsymbol{W}_k^*$ as the optimum of $\boldsymbol{W}_k$, and $F_k^*$ as the minimum value of $F_k$. We use $\Gamma(k,t) = \sum_{i \in [S_t]} p_i ||\boldsymbol{M}_i^t \odot \boldsymbol{M}_k^t||$ for quantifying similarity of the data distribution between device $k$ and the other devices. In particular, the smaller $\Gamma(k,t)$ indicates a higher data heterogeneity across devices. Then we have the following convergence result of pruning phase in Theorem 1.

Theorem 1. *Let Assumptions 1-5 hold for Hermes and $L$, $\mu$, $B$, $Q$, $K$, $P$, $G$ be defined therein. Choose $\eta_t \leq \frac{Q}{KL\Gamma(k,t)}$. Then $\forall k \in [N]$, the device in pruning phase satisfies:*

$$||\boldsymbol{W}_k^{\hat{t}+1} - \boldsymbol{W}_k^*||^2 \leq \{1 - \{\eta_t[\mu Q \Gamma(k,t) - 2] - \frac{1}{2}\}\}||\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*||^2 + \eta_t(B+G) + \eta_t^2 B + 3P \quad (10)$$

Proof. See our proof in Appendix A □

Remark 1. *We have three key observations for the pruning phase: 1) It is hard for a device to obtain the optimal model before the other devices stop pruning; 2) To make $\boldsymbol{W}_k$ close to the local optimum $\boldsymbol{W}_k^*$, each device should keep $\Gamma(k,t)$ larger than $\frac{2}{\mu Q}$, which represents the lower bound of the data distribution similarity between device $k$ and the other devices; 3) The pruning rate $P$ should not be too large to optimize local models during pruning phase.*

We assume that the sparse phase starts from the communication round $T_s$, and $\Gamma(k,t)$ is fixed to be $\Gamma(k)$ after $T_s$. Then we have the convergence guarantee of sparse phase in Theorem 2.

Theorem 2. *Let Assumptions 1-5 hold for Hermes and $L$, $\mu$, $B$, $Q$, $K$, $P$, $G$ be defined therein. Choose $\eta_t = \eta_s \leq \frac{Q}{KL\Gamma(k)}$. Then $\forall k \in [N]$, $T > T_s$, the device in sparse phase satisfies:*

$$F_k(\boldsymbol{W}_k^T) - F_k^* \leq \frac{L}{2}(1 - \eta_s R_k)^{U(T-T_s)}[F_k(\boldsymbol{W}_k^{T_s}) - F_k^* - Z_k] + \frac{L}{2} Z_k \quad (11)$$

*where*

$$Z_k = \frac{B}{\mu Q \Gamma(k) - 1}(1 + \frac{Q}{KL\Gamma(k)}) \quad R_k = \mu Q \Gamma(k) - 1 \quad (12)$$

Proof. See our proof in Appendix B □

**Table 1: Statistical information of datasets.**

| Dataset | Number of devices | Classes | Non-IID |
|---|---|---|---|
| EMNIST [8] | 2414 | 64 | ✓ |
| CIFAR10 [26] | 400 | 10 | ✓ |
| HAR [3] | 30 | 6 | ✓ |

Remark 2. *By choosing $\lambda_s \leq R_k$, the devices in Hermes can achieve an exponential convergence rate, which is faster than the convergence rate of FedAvg (i.e., $O(\frac{1}{T})$) presented in [31]. We observe that the lower bound of optimal local loss $F_k^* + \frac{L}{2}Z_k$ varies across devices. This lower bound is negatively correlated with $\Gamma(k)$ as shown in $Z_k$, which implies that the more closely one device's data distribution is similar to the other devices' data distributions, the smaller local loss this device can obtain by participating in federated training.*

# 5 EVALUATIONS

## 5.1 Applications, Datasets, and Models

To show the generality of Hermes across applications, we employ Hermes to develop two representative mobile AI applications that benefit significantly from FL. The statistics of the datasets that are applied to build these applications are summarized in Table 1. We follow the non-IID configurations in [29] to build the non-IID datasets.

**Application#1: Image Classification (IC).** Image classification is a popular computer vision application. With the advancement of the computation power, it has become attractive to deploy image classification applications on mobile devices. In this work, we use VGG16[45] as the base model for training, and EMNIST [8] and CIFAR10 [26] datasets for training and evaluation. EMNIST is a handwriting image classification dataset grouped by the writers. Hence, we can naturally distribute one writer's images to one device. In this application, we sample 2414 writers' data and distribute them to devices. For CIFAR10, each device holds 2-class data and this two classes can be varied across devices. In addition, the data volume of each class is unbalanced on a device. The test data follows the same distribution as the training data.

**Application#2: Human Activity Recognition (HAR).** Human activity recognition has become an attractive feature for smartphones using data collected from different types of on-board sensors, such as accelerometer, gyroscope, etc. This application aims to recognize activities performed by an individual based on the sensor data. To build this application, we use the HAR [3] dataset which collects smartphone accelerometer and gyroscope data from 30 individuals. HAR consists of six labeled activities: walking, walking-upstairs, walking-downstairs, sitting, standing, and lying-down. As a result, we distribute each individual's data to one device. We employ a 3-layer fully connected neural network to recognize human activities.

## 5.2 System Implementation

We implemented Hermes and three FL applications on Google Pixel 3 (CPU) smartphones running Android 9.0, based on PyTorch 1.5. The central server is equipped with an Intel Xeon E5-2630@2.6GHz and 128G RAM. We used Monsoon power monitor [38] to measure the power consumption at runtime. For the baselines and Hermes, we adopt a FL protocol which randomly feeds 20 data shards that are partitioned as described in §5.1 to each smartphone in each communication round, i.e., 20 participating devices per communication round. Each participating device performs $E = 5$ local training epochs for one communication round. In addition, we set the pruning rate $r_p$ as 0.2, $B = 16$, $acc_{threshold} = 0.5$, and $r_{target} = 0.3$ in Algorithm 1. The model configurations for each application are presented in Table 4 (Appendix C).

## 5.3 Experimental Setup

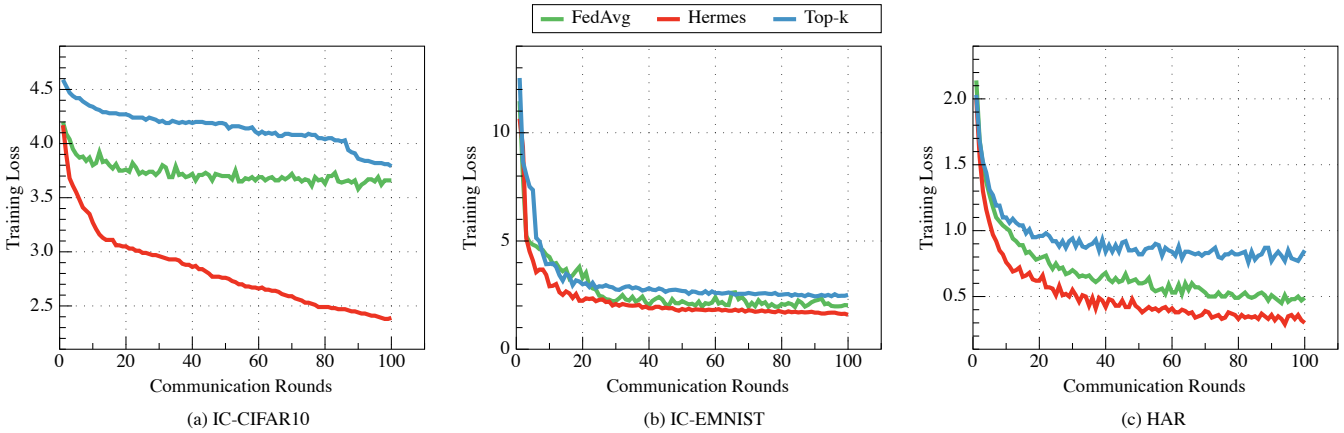
**Baselines.** To make fair comparison, we compare Hermes with five baselines:

- **Standalone** trains a model locally using local data only by each device. Note that we provide the same total training time for Standalone method as Hermes. Since Standalone method does not have to pay for the communication overhead, the number of the local training epochs in Standalone will be greater than that of Hermes.
- **FedAvg** [36] is the most classical FL method. Devices communicate updated local parameters to the central server and download the aggregated global model for continuous local training.
- **Top-k** [1] is a sparsification method to compress the communicated gradients by selecting the largest $k$ elements of the gradients. Here we set $k$ as 10%.
- **Per-FedAvg** [12] incorporates MAML [13] into FedAvg for personalization. Per-FedAvg first finds an initial shared model and then each devices can adapt this shared model to their local data by performing a few steps of gradient descent.
- **LG-FedAvg** [32] is the state-of-the-art FL method that improves personalization and communication efficiency simultaneously. This is done by jointly learning compact local representations on each device and a shared global model.

Besides, for each application, we adopt the same base model configuration for each baseline method as Hermes. Training settings and data configurations of baseline methods are set the same as Hermes. The results are reported after the same number of training epochs except for Standalone.

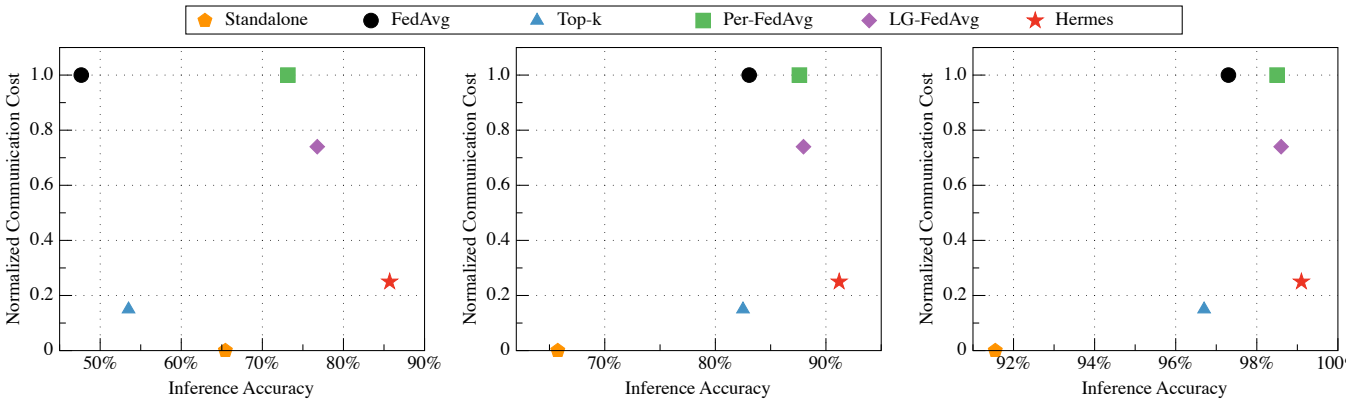
**Evaluation Metrics.** We use two sets of metrics to evaluate training and runtime performance respectively:

- **Metrics for Training Performance:** (1) *inference accuracy:* we evaluate the inference accuracy on each device's test data, and report the average accuracy over all devices for evaluations; (2) *communication cost:* we measure the time cost of communication including both uploading and downloading during training, and normalize it as the ratio to the communication time of FedAvg as reported communication cost.
- **Metrics for Runtime Performance:** (1) *memory footprint:* we measure the memory footprint of different applications on devices, and calculate the memory footprint reduction; (2) *inference latency:* we measure the average time cost of performing one inference and calculate the inference time reduction percentage; (3) *energy consumption:* we measure




(a) IC-CIFAR10 (b) IC-EMNIST (c) HAR

**Figure 6: Comparison of convergence speed between FedAvg and Hermes.**




(a) IC-CIFAR10 (b) IC-EMNIST (c) HAR

**Figure 7: Comparison between Hermes and baselines in inference accuracy-communication cost space.**

the average energy consumption per inference and calculate the energy consumption saving percentage.

## 5.4 Convergence Speed

To verify our theoretical analysis, we compare the convergence speed of Hermes with that of FedAvg and Top-k. In particular, the training curves of these three methods are presented in Figure 6. The results show that Hermes (red curve) not only converges significantly faster than FedAvg (green curve) and Top-k (blue curve) but also achieves consistently better performance (i.e., smaller training loss) in all three applications.

## 5.5 Inference Accuracy vs. Communication Cost

Figure 7 compares Hermes with the baselines in the inference accuracy-communication cost space. Even though Hermes cannot always outperform baselines in both inference accuracy and communication cost, it offers the best tradeoff between inference accuracy and communication cost.

First, compared to the state-of-the-art method LG-FedAvg, Hermes is able to achieve higher inference accuracy and lower communication cost simultaneously. Hermes improves inference accuracy by 8.93%, 3.23%, and 0.53% on IC-CIFAR10, IC-EMNIST, and HAR, respectively. Meanwhile, it also saves 1.96×, 1.92×, and 1.96× communication cost in these applications, respectively.

Second, compared to Per-FedAvg that is specifically designed for personalization, Hermes is able to dramatically reduce communication cost. In particular, Hermes reduces 3.05×, 3.25×, and 3.48× communication cost on IC-CIFAR10, IC-EMNIST, HAR, and NCP, respectively. More surprisingly, Hermes also achieves a significant improvement in inference accuracy. The inference accuracy increases by 12.55%, 3.46%, and 0.6%, respectively.

Third, Top-k shows the most significant communication reductions on the three developed applications. Compared to Hermes, Top-k offers 0.67×, 0.63×, 0.67×, 3.36× savings in communication cost on IC-CIFAR10, IC-EMNIST, and HAR, respectively. However, Hermes significantly outperforms Top-k in inference accuracy, increasing inference accuracy by 32.17%, 8.71%, and 2.39%.

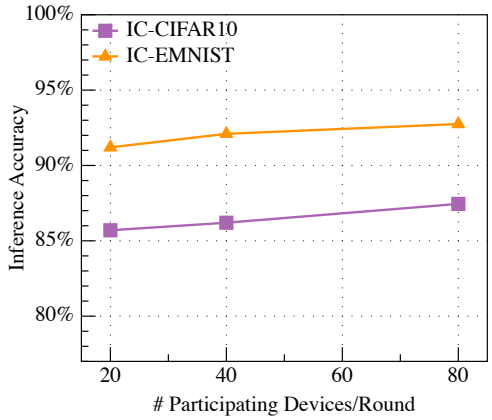



**Figure 8: The impact of the number of participating devices on Hermes performance.**

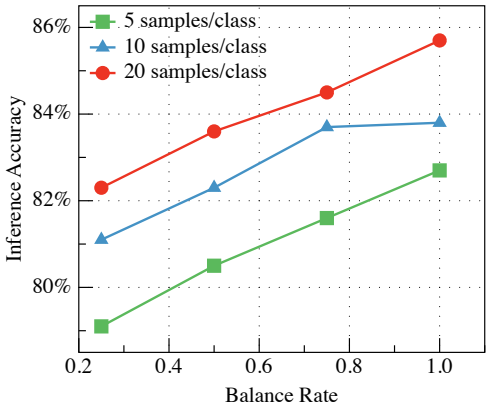



**Figure 9: The impact of data volume and balance rate on Hermes performance (IC-CIFAR10).**

As FedAvg is a general FL method without specific optimizations for communication and personalization, Hermes significantly outperforms FedAvg in terms of both inference accuracy and communication cost as expected. Although Standalone does not incur any communication cost, it shows substantially worse performance compared to Hermes due to only local training with limited data samples on each device.

## 5.6 Hyper-Parameter Evaluation

**Number of Participating Devices:** We evaluate the impact of the number of participating devices in each communication round on the Hermes performance. We conduct experiments on IC-EMNIST and IC-CIFAR10 by varying the number of participating devices with {20, 40, 80}. As Figure 8 shows, the inference accuracy has a slight improvement with a larger number of participating devices in each communication round. For example, the inference accuracy on IC-CIFAR10 increases by 1.75% when the number of participating devices increases from 20 to 80. However, such a larger number of participating devices incurs 4× bandwidth usage, which diminishes the benefit of the slight accuracy improvement.

**Data Volume and Unbalance Rate:** The data volume on devices is a critical hyper-parameter that significantly impacts the performance of FL methods. In practice, there exist some challenging scenarios where the data volume is extremely limited. Besides limited data, it is also common that data on a device shows unbalance across different classes. The limited and unbalanced data raise challenges for FL methods to train personalized models achieving equally high performance on different classes. To evaluate the impact of data volume and the degree of unbalance on Hermes performance, we conduct experiments on IC-CIFAR10. In this experiment, each device holds 2-class data, and we define *balance rate* as the ratio between the data volume of one class and the counterpart of the other class. Therefore, the lower value of balance rate indicates the higher degree of unbalance. We keep each device holding 2-class data, but vary the volume of data with {5, 10, 20} samples/class and the balance rate with {1, 0.75, 0.5, 0.25}. Note that the data is balanced when setting the balance rate as 1. As Figure 9 illustrates, given the same volume of data, the inference accuracy has a slight drop when the balance rate becomes smaller. For example, with setting the data volume as 20 samples/class, the inference accuracy decreases from 84.35% to 83.67% when varying the balance rate from 0.75 to 0.5. In addition, given the same balance rate, the inference accuracy slightly degrades with a decreasing volume of data. For example, with setting the balance rate as 0.75, the inference accuracy decreases by only 0.92% when changing the data volume from 10 samples/class to 5 samples/class. The above results demonstrate that Hermes can also effectively handle very challenging settings.

**Target Pruning Rate:** The target pruning rate $r_{target}$ determines the degree of the subnetwork's sparsity on the device, i.e., a smaller $r_{target}$ results in a more compact subnetwork and a larger $r_{target}$ leads to a more dense subnetwork, and hence the $r_{target}$ impacts both communication cost and inference accuracy. To evaluate the impact of $r_{target}$ on Hermes performance, we conduct experiments on IC-CIFAR10 and IC-EMNIST by varying the $r_{target}$ with {0.3, 0.5, 0.8}. In this experiment, we measure the volume of communicated data including both uploading and downloading as the communication cost, and normalize it as the ratio to the volume of communicated data when setting $r_{target} = 0.8$ as the reported communication cost. As Table 2 presents, the inference accuracy has a marginal drop when the $r_{target}$ becomes smaller, but the communication cost can be significantly reduced. For example, the inference accuracy marginally decreases from 86.35% to 85.72% when varying the $r_{target}$ from 0.8 to 0.3 on IC-CIFAR10, but the communication cost can be reduced by 46%.

## 5.7 Runtime Performance

**Reduction of Memory footprint:** One key benefit of applying Hermes is that each device can obtain a structured sparse and personalized model for deployment. To quantify the benefit of structured sparsity on reducing memory footprint, we compare the model size between the model learned via applying Hermes and the ones trained using the baselines. Note that the models learned

**Table 2: The impact of target pruning rate on Hermes performance.**

| Application | $r_{target}$ | Accuracy | Communication Cost |
|---|---|---|---|
| IC-CIFAR10 | 0.3 | 85.72% | 0.54 |
| | 0.5 | 85.91% | 0.79 |
| | 0.8 | 86.35% | 1 |
| IC-EMNIST | 0.3 | 91.24% | 0.53 |
| | 0.5 | 91.35% | 0.75 |
| | 0.8 | 91.66% | 1 |

**Table 3: Memory footprint reduction of Hermes. ($r_{target} = 0.3$)**

| Application | Hermes Model Size (MB) | Baseline Model Size (MB) |
|---|---|---|
| IC-CIFAR10 | 161.16 | 537.21 |
| IC-EMNIST | 161.43 | 538.09 |
| HAR | 1.32 | 4.41 |
| All Included | 323.91 | 1081.24 |

using the baselines are dense neural networks and share the same model size.

Table 3 lists comparison of the memory footprint across all the applications. Compared to the other baselines, Hermes is able to significantly reduce the memory footprint. For example, Hermes can save 376.05MB on memory footprint in IC-MNIST, which is equivalent to 70% of the memory footprint by the baseline model. In addition, considering to deploy all the application applications concurrently on smartphones, Hermes achieves 757.33MB memory footprint reduction in total, showing the strong applicability of Hermes in practice.

**Inference Speedup:** The structured sparsity of the model learned using Hermes also introduces a significant speedup on inference. To quantify this benefit, we compare the average inference latency between the model learned using Hermes and the ones trained by the baselines. As Figure 10 illustrates, compared with the baselines, Hermes is able to achieve 1.83×, 1.79×, and 1.82× inference speedup on IC-CIFAR10, IC-EMNIST, and HAR respectively.

**Reduction on Energy Consumption:** Besides reducing memory footprint and inference latency, Hermes also saves more energy compared to the baselines. In this experiment, we compare the average energy consumption per inference across all the applications between Hermes and the baselines. As Figure 11 shows, Hermes is able to save 1.8×, 1.76×, and 1.78× energy consumption on IC-CIFAR10, IC-EMNIST, and HAR respectively.

## 6 DISCUSSION

**Generality of Hermes.** In this work, we apply Hermes to develop two representative applications using three non-IID datasets (as presented in Table 1). However, Hermes can be generalized to develop many other popular mobile AI applications. For example, we can apply Hermes to build next-character prediction

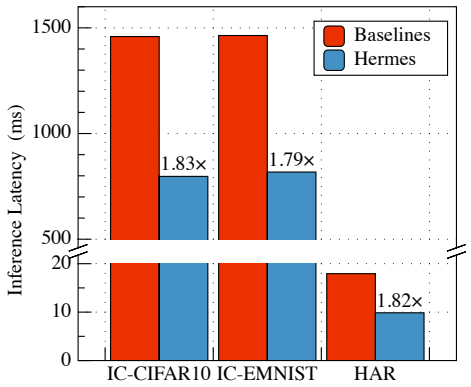



**Figure 10: Comparison between Hermes and the baselines on inference time. ($r_{target} = 0.3$)**

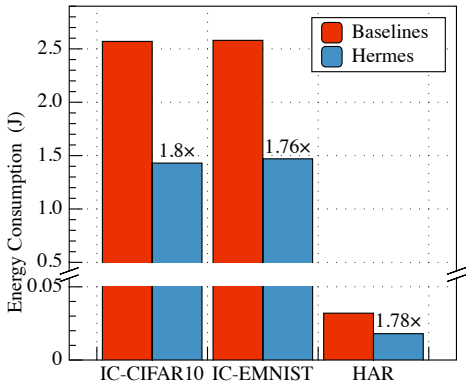



**Figure 11: Comparison between Hermes and the baselines on energy consumption. ($r_{target} = 0.3$)**

application on smartphones using text datasets (e.g., Sentiment140, Shakespeare) [7]. In this sense, Hermes is a generic framework for achieving personalized and efficient FL.

**Defending Against Privacy Leakage.** Privacy preservation is the major motivation for proposing FL. However, recent works demonstrated that sharing model updates or gradients also makes FL vulnerable to inference attack, e.g., property inference attack [37] and model inversion attack [15, 16, 52, 57]. Recent study [48] reveals the essential cause of privacy leakage in FL – data representations of each device's local data are embedded in communicated local model updates and such data representations can be inferred to perform model inversion attacks. In Hermes, only subnetworks (e.g., a small portion of the local model) are communicated between devices and the central server, i.e., some gradients have been pruned on the device and partial information of local data has been hidden. This indicates that Hermes could potentially help to defend against the privacy leakage in FL, and we plan to evaluate the effectiveness of Hermes in such defenses in the future work.

## 7 RELATED WORK

**Communication-Efficient Distributed Deep Learning.** Communication cost is one the major bottlenecks of distributed deep learning. Many works have been done to improve the communication efficiency by reducing the volume of transmitted data (i.e., gradient or weights). These prior arts can be divided into three main classes: (1) *quantization* methods [2, 6, 9, 19, 24, 44, 54, 55] compress the communication by reducing the number of bits of each element in the transferred data; (2) *sparsification* methods [1, 11, 21] transmit only a subset of elements in the communicated data; (3) *hybrid* methods [5, 22, 33, 47] combine quantization with sparsification. There are two major quantization schemes: *limited-bit* and *codebook-based*. The limited-bit quantization method [6, 9] maps each gradient element to fewer bits, and the codebook-based approach [2] projects the gradient coordinates into predefined code-words. For example, Dettmers *et al.* [9] design a limited-bit technique to map each float32 element of the gradients to 8 bits. Similarly, SignSGD [6] is another typical limited-bit method, where the sign of each gradient element is transmitted by mapping the negative element to -1 and the others to 1. Alistarh *et al.* propose QSGD [2], which is codebook-based quantization scheme, to quantizes each element of the gradients via randomized rounding to a discrete set of code-words that preserve the statistical properties of the original gradients. In terms of sparsification methods, the core idea is to select a subset of the gradients to transmit. Therefore, there are many selection criterions have been proposed for improving the sparsification. Top-$k$ [1] is a sparsification method to select the $k$ largest elements of the gradients. Unlike Top-$k$ where the number of selected elements is fixed, Threshold-$v$ [11] selects the elements which are larger than a predefined threshold value, hence, the number of selected elements is adapted during runtime. Ivkin *et al.* propose Sketched-SGD [21], which adopts count-sketch to approximate the top-$k$ elements of the gradients. In addition, hybrid methods have been proposed via combing quantization with sparsification. For example, Basu *et al.* propose a hybrid method [5] by combining quantization with Top-$k$ or Random-$k$ sparsification.

In addition to these data compression based methods, federated submodel learning is another scheme for improving communication efficiency. In general, each participating device only trains and transmits a submodel given a shared global model. FedMA [50] proposes a layer-wise matching scheme by exploiting the model architecture. In particular, FedMA updates a layer of the global model each time using the matched averaging, which reduces the size of communicated data. HeteroFL [10] adaptively allocate submodels to devices based on their varying computation and communication capabilities, and hence it can improve communication and computation efficiency in federated training. Our work also aims to reduce communication cost by exploiting the submodel learning scheme, but the submodel is learned using local data in Hermes with taking the data heterogeneity into account.

**Personalization for FL.** Due to statistical heterogeneity (i.e., non-IID data distribution across clients), it is necessary to adapt the global model to achieve personalization. Most existing works achieve personalization in two separate steps that are associated with extra overhead: 1) a global model is learned in a federated fashion, and 2) the global model is fine-tuned for each client using the local data. There are three primary categories of methods for adapting global model to personalized models: *local fine-tuning*, *multi-task learning*, and *contextualization*. The local fine-tuning is the dominant approach for personalization [12, 23, 25, 51], where each client tunes a global model using its own local data with several gradient descent steps. Jiang *et al.* [23] analyze the similarity between FedAvg [36] and Reptile [39] which is a meta-learning method. They observe that only optimizing the performance of global model could limit the global model's capacity for personalization. Based on the observation, they propose to combine meta learning with FL for personalizing local models. Fallah *et al.* propose Per-FedAvg [12] by incorporating MAML [13] into FL for personalization, where each local model is fine-tuned by using the second-order information to update the global with several gradient descent steps.Khodak *et al.* design ARUBA [25], which is a meta learning algorithm based on online convex optimization to improve the personalization in FL. In addition to the above meta-learning based approaches, transfer learning has also been applied to FL for personalization. Wang *et al.* [51] and Arivazhagan *et al.* [4] propose methods to fine-tune some or all parameters of a trained global model using each client's local data. Besides, we can also consider the personalization problem in FL as a multi-task learning problem. Smith *et al.* propose MOCHA [46], which is a multi-task learning approach to learn personalized federated models. Integrating contextualization with FL is another approach for personalization, Masour *et al.* [35] propose a method to add user context to FL, which first clusters users based on the some features (e.g., region) and then train a separate model for each group. Sattler *et al.* [43] integrate hierarchical clustering into FL by using a bi-partitioning algorithm to group clients into clusters. Huang *et al.* [20] present a community-based FL to predict patient hospitalization time and mortality. But such contextualization methods often incur high computation and communication costs, leading to limited feasibility in practice. In contrast, Hermes achieves personalization by learning the heterogeneous subnetwork in an end-to-end manner with jointly optimizing communication and computation efficiency.

## 8 CONCLUSION

In this paper, we present the design, implementation and evaluation of Hermes, a FL framework that significantly improves communication and inference efficiency simultaneously while achieving personalization under data heterogeneity. By applying Hermes, each device learns a personalized and structured sparse DNN model rather than a shared global model. Hermes incorporates the structured pruning into the local training, each device learns a sparse subnetwork and only the parameters of subnetwork are communicated between devices and the central server. The aggregation strategy is carefully designed to average only the intersected parameters across subnetworks, such that the personalization of subnetworks can be preserved. We evaluate Hermes using two representative FL applications on three benchmark datasets. Our results demonstrate that Hermes significantly outperforms the state-of-the-art methods in inference accuracy, communication cost, memory footprint, inference latency, and energy consumption. We believe Hermes represents a significant step towards the realization of efficient FL under heterogeneous mobile clients.

# REFERENCES


[1] Alham Fikri Aji and Kenneth Heafield. 2017. Sparse Communication for Distributed Gradient Descent. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 440–445. https://doi.org/10.18653/v1/D17-1045

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In *Advances in Neural Information Processing Systems*. 1709–1720.

[3] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones.. In *Esann*, Vol. 3. 3.

[4] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated Learning with Personalization Layers. *arXiv:1912.00818 [cs, stat]* (Dec. 2019).

[5] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. 2019. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*. 14695–14706.

[6] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. 2018. signSGD: Compressed Optimisation for Non-Convex Problems. *International Conference on Machine Learning* (2018), 560–569.

[7] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2019. LEAF: A Benchmark for Federated Settings. arXiv:1812.01097 [cs.LG]

[8] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2921–2926.

[9] Tim Dettmers. 2015. 8-bit approximations for parallelism in deep learning. *arXiv preprint arXiv:1511.04561* (2015).

[10] Enmao Diao, Jie Ding, and Vahid Tarokh. 2020. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. In *International Conference on Learning Representations*.

[11] Aritra Dutta, El Houcine Bergou, Ahmed M. Abdelmoniem, Chen-Yu Ho, Atal Narayan Sahu, Marco Canini, and Panos Kalnis. 2020. On the Discrepancy between the Theoretical Analysis and Practical Implementations of Compressed Communication for Distributed Deep Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (April 2020), 3817–3824. https://doi.org/10.1609/aaai.v34i04.5793

[12] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized Federated Learning: A Meta-Learning Approach. *arXiv:2002.07948 [cs, math, stat]* (Feb. 2020).

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), 1126–1135.

[14] Jonathan Frankle and Michael Carbin. 2018. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*.

[15] 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*.

[16] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting Gradients–How easy is it to break privacy in federated learning?. In *Advances in Neural Information Processing Systems*.

[17] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[18] Filip Hanzely and Peter Richtárik. 2020. Federated Learning of a Mixture of Global and Local Models. *arXiv:2002.05516 [cs, math, stat]* (Feb. 2020).

[19] Samuel Horvath, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtarik. 2019. Natural compression for distributed deep learning. *arXiv preprint arXiv:1905.10988* (2019).

[20] Li Huang, Andrew L Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. 2019. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of biomedical informatics* 99 (2019), 103291.

[21] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. 2019. Communication-Efficient Distributed SGD with Sketching. *arXiv:1903.04488 [cs, math, stat]* (June 2019).

[22] Jiawei Jiang, Fangcheng Fu, Tong Yang, and Bin Cui. 2018. SketchML: Accelerating Distributed Machine Learning with Data Sketches. *Proceedings of the 2018 International Conference on Management of Data - SIGMOD '18*, 1269–1284. https://doi.org/10.1145/3183713.3196894

[23] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. 2019. Improving Federated Learning Personalization via Model Agnostic Meta Learning. *arXiv:1909.12488 [cs, stat]* (Sept. 2019).

[24] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and Open Problems in Federated Learning. *arXiv:1912.04977 [cs, stat]* (Dec. 2019).

[25] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. 2019. Adaptive Gradient-Based Meta-Learning Methods. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett (Eds.). 5917–5928.

[26] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html* 55 (2014).

[27] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. 2020. Survey of Personalization Techniques for Federated Learning. *arXiv:2003.08673 [cs, stat]* (March 2020).

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[29] Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li. 2020. LotteryFL: Personalized and Communication-Efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets. *arXiv preprint arXiv:2008.03371* (2020).

[30] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2019. Federated Learning: Challenges, Methods, and Future Directions. *arXiv:1908.07873 [cs, stat]* (Aug. 2019).

[31] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[32] Paul Pu Liang, Terrance Liu, Liu Ziyin, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Think Locally, Act Globally: Federated Learning with Local and Global Representations. *arXiv:2001.01523 [cs, stat]* (Feb. 2020).

[33] Hyeontaek Lim, David G Andersen, and Michael Kaminsky. 2018. 3LC: Lightweight and Effective Traffic Compression for Distributed Machine Learning. *arXiv preprint arXiv:1802.07389* (2018).

[34] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.

[35] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three Approaches for Personalization with Applications to Federated Learning. *arXiv:2002.10619 [cs, stat]* (Feb. 2020).

[36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. *Artificial Intelligence and Statistics*, 1273–1282.

[37] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE.

[38] Monsoon Power Monitor. 2013. online]: http://www. msoon. com/LabEquipment. *PowerMonitor/, visited Nov* (2013).

[39] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On First-Order Meta-Learning Algorithms. *arXiv:1803.02999 [cs]* (Oct. 2018).

[40] Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. 2020. Billion-scale federated learning on mobile clients: a submodel design with tunable privacy. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[41] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint arXiv:1906.04329* (2019).

[42] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. 2018. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1421–1429.

[43] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. *IEEE Transactions on Neural Networks and Learning Systems* (2019), 1–14. https://doi.org/10.1109/TNNLS.2019.2944481

[44] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 2014. 1-Bit Stochastic Gradient Descent and Its Application to Data-Parallel Distributed Training of Speech DNNs. *Fifteenth Annual Conference of the International Speech Communication Association* (2014).

[45] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[46] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated Multi-Task Learning. In *Advances in Neural Information Processing Systems*. 4424–4434.

[47] Nikko Strom. 2015. Scalable Distributed DNN Training Using Commodity GPU Cloud Computing. *Sixteenth Annual Conference of the International Speech Communication Association* (2015).

[48] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. 2020. Provable Defense against Privacy Leakage in Federated Learning from Representation Perspective. arXiv:2012.06043 [cs.LG]

[49] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning.

*arXiv preprint arXiv:1602.07261* (2016).

[50] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2019. Federated Learning with Matched Averaging. In *International Conference on Learning Representations*.

[51] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. 2019. Federated Evaluation of On-Device Personalization. *arXiv:1910.10252 [cs, stat]* (Oct. 2019).

[52] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE.

[53] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems* 29 (2016), 2074–2082.

[54] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2017. TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning. In *Advances in neural information processing systems*. 1509–1519.

[55] Yue Yu, Jiaxiang Wu, and Junzhou Huang. 2019. Exploring Fast and Communication-Efficient Algorithms in Large-Scale Distributed Networks. *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), 674–683.

[56] Xiao Zeng, Kai Cao, and Mi Zhang. 2017. MobileDeepPill: A small-footprint mobile deep learning system for recognizing unconstrained pill images. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. 56–67.

[57] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*.

# A PROOF OF THEOREM 1

Proof.

$$\|\hat{\boldsymbol{W}}_k^{t+1} - \boldsymbol{W}_k^*\|^2 = \|(\hat{\boldsymbol{W}}_k^{t+1} - \hat{\boldsymbol{W}}_k^t) + (\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*)\|^2 \tag{13}$$

$$= \|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2 + \underbrace{\|\hat{\boldsymbol{W}}_k^{t+1} - \hat{\boldsymbol{W}}_k^t\|^2}_{A} \tag{14}$$

$$+ \underbrace{2 < \hat{\boldsymbol{W}}_k^{t+1} - \hat{\boldsymbol{W}}_k^t, \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* >}_{B} \tag{15}$$

Both $A$ and $B$ are related with $\hat{\boldsymbol{W}}_k^{t+1} - \hat{\boldsymbol{W}}_k^t$, then we split $\hat{\boldsymbol{W}}_k^{t+1} - \hat{\boldsymbol{W}}_k^t$ into three terms:

$$\hat{\boldsymbol{W}}_k^{t+1} - \hat{\boldsymbol{W}}_k^t = \underbrace{-\boldsymbol{M}_k^{t+1} \odot \sum_{i\in[N]} p_i \eta_t \odot \boldsymbol{M}_i^{t+1} \odot \nabla F_i^t(\boldsymbol{W}_i^t)}_{C1} \tag{16}$$

$$+ \underbrace{\sum_{i\in[N]} p_i (\boldsymbol{M}_k^{t+1} \odot \boldsymbol{M}_i^{t+1} - \boldsymbol{M}_k^t \odot \boldsymbol{M}_i^t) \odot \boldsymbol{W}_j^t}_{C2} \tag{17}$$

$$+ \underbrace{(\boldsymbol{M}_k^{t+1} \ominus \boldsymbol{M}^{t+1}) \odot \hat{\boldsymbol{W}}_k^t + (\boldsymbol{M}_k^t \ominus \boldsymbol{M}^t) \odot \hat{\boldsymbol{W}}_k^{t-1}}_{C3} \tag{18}$$

We next focus on bounding $A$. By the convexity of $\|.\|^2$, we have:

$$\|\hat{\boldsymbol{W}}_k^{t+1} - \hat{\boldsymbol{W}}_k^t\|^2 = \|C1 + C2 + C3\|^2 \tag{19}$$

$$\leq \|C1\|^2 + \|C2\|^2 + \|C3\|^2 \tag{20}$$

Note that $\boldsymbol{M}_k^t \subset \boldsymbol{M}^t$, we have $\|C3\|^2 = 0$. To bound $\|C1\|^2$, we have

$$\|C1\|^2 = \|\boldsymbol{M}_k^{t+1} \odot \sum_{i\in[N]} p_i \eta_t \boldsymbol{M}_i^{t+1} \odot \nabla F_i^t(\boldsymbol{W}_i^t)\|^2 \tag{21}$$

$$\leq \sum_{i\in[N]} \|p_i \eta_t \boldsymbol{M}_k^{t+1} \odot \boldsymbol{M}_i^{t+1} \odot \nabla F_i^t(\boldsymbol{W}_i^t)\|^2 \tag{22}$$

$$\leq \sum_{i\in[N]} \|p_i \eta_t \boldsymbol{M}_k^t \odot \boldsymbol{M}_i^t \odot \nabla F_i^t(\boldsymbol{W}_i^t)\|^2 \tag{23}$$

$$\leq \sum_{i\in[N]} p_i^2 \eta_t^2 \|\boldsymbol{M}_i^t \odot \nabla F_k^t(\boldsymbol{W}_k^t)\|^2 + \sum_{i\in[N]} p_i^2 \eta_t^2 B \tag{24}$$

$$\leq \sum_{i\in[N]} p_i^2 \eta_t^2 K \|\boldsymbol{M}_i^t \odot \boldsymbol{M}_k^t\|^2 \|\nabla F_k^t(\boldsymbol{W}_k^t)\|^2 + \eta_t^2 B \tag{25}$$

where the first inequality results from the convexity of $\|.\|^2$, the second inequality results from the shrinking property 5 of local mask, the third inequality results from Assumption 3 and the convexity of $\|.\|^2$, the forth inequality results from Assumption 4 and $p_i \leq 1$.

From the $L$-smoothness of $F_k$, it follows that

$$\|\nabla F_k^t(\boldsymbol{W}_k^t)\|^2 \leq 2L(F_k^t(\boldsymbol{W}_k^t) - F_k^*). \tag{26}$$

By combining inequality 25 and 26, it follows that

$$\|C1\|^2 \leq 2\eta_t^2 KL(F_k^t(\boldsymbol{W}_k^t) - F_k^*) \sum_{i\in[N]} p_i^2 \|\boldsymbol{M}_i^t \odot \boldsymbol{M}_k^t\|^2 + \eta_t^2 B \tag{27}$$

$$\leq 2\eta_t^2 KL(F_k^t(\boldsymbol{W}_k^t) - F_k^*)\Gamma(k,t)^2 + \eta_t^2 B \tag{28}$$

To bound $\|C2\|^2$, we have

$$\|C2\|^2 = \|\sum_{i\in[N]} p_i (\boldsymbol{M}_k^{t+1} \odot \boldsymbol{M}_i^{t+1} - \boldsymbol{M}_k^t \odot \boldsymbol{M}_i^t) \odot \boldsymbol{W}_i^t\|^2 \tag{29}$$

$$\leq \sum_{i\in[N]} p_i^2 \|(\boldsymbol{M}_k^{t+1} \odot \boldsymbol{M}_i^{t+1} - \boldsymbol{M}_k^t \odot \boldsymbol{M}_i^t) \odot \boldsymbol{W}_i^t\|^2 \tag{30}$$

$$\leq \sum_{i\in[N]} p_i^2 \|(\boldsymbol{M}_k^t \odot \boldsymbol{M}_i^{t+1} - \boldsymbol{M}_k^t \odot \boldsymbol{M}_i^t) \odot \boldsymbol{W}_i^t\|^2 \tag{31}$$

$$= \sum_{i\in[N]} p_i^2 \|\boldsymbol{M}_k^t \odot (\boldsymbol{M}_i^{t+1} - \boldsymbol{M}_i^t) \odot \boldsymbol{W}_i^t\|^2 \tag{32}$$

$$\leq \sum_{i\in[N]} p_i^2 \|(\boldsymbol{M}_i^{t+1} - \boldsymbol{M}_i^t) \odot \boldsymbol{W}_i^t\|^2 \tag{33}$$

$$\leq \sum_{i\in[N]} p_i^2 P \tag{34}$$

$$\leq P \tag{35}$$

where the first inequality results from the convexity of $\|.\|^2$, the second and the third inequality comes from the properties 57 of local mask, the forth inequality results from Assumption 5.

By combining 20, 28 and 35, we have

$$\|\hat{\boldsymbol{W}}_k^{t+1} - \hat{\boldsymbol{W}}_k^t\|^2 \tag{36}$$

$$\leq 2\eta_t^2 KL(F_k^t(\boldsymbol{W}_k^t) - F_k^*)\Gamma(k,t)^2 + \eta_t^2 B + P \tag{37}$$

We next focus on $B$. Again, we split $B$ into three terms:

$$2 < \hat{\boldsymbol{W}}_k^{t+1} - \hat{\boldsymbol{W}}_k^t, \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* > \tag{38}$$

$$= \underbrace{2 < C1, \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* >}_{B1} + \underbrace{2 < C2, \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* >}_{B2} \tag{39}$$

$$+ \underbrace{2 < C3, \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* >}_{B3} \tag{40}$$

Note that $\boldsymbol{M}_k^t \subset \boldsymbol{M}^t$, we have $B3 =< C3, \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* >= 0$. We split $B1$ into two terms:

$$< -\boldsymbol{M}_k^{t+1} \odot \sum_{i\in[N]} p_i \eta_t \boldsymbol{M}_i^{t+1} \odot \nabla F_i^t(\boldsymbol{W}_i^t), \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* > \tag{41}$$

$$= \underbrace{< -\boldsymbol{M}_k^t \odot \sum_{i\in[N]} p_i \eta_t \boldsymbol{M}_i^t \odot \nabla F_i^t(\boldsymbol{W}_i^t), \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* >}_{B1-1} \tag{42}$$

$$+ \underbrace{< (\boldsymbol{M}_k^t \odot \boldsymbol{M}_k^t - \boldsymbol{M}_k^{t+1} \odot \boldsymbol{M}_i^{t+1}) \odot \sum_{i\in[N]} p_i \eta_t \nabla F_i^t(\boldsymbol{W}_i^t), \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* >}_{B1-2} \tag{43}$$

We next focus on $B1-1$. We split $B1-1$ into two terms:

$$< -\boldsymbol{M}_k^t \odot \sum_{i\in[N]} p_i\eta_t \boldsymbol{M}_i^t \odot \nabla F_i^t(\boldsymbol{W}_i^t), \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* > \tag{44}$$

$$= \underbrace{< -\sum_{i\in[N]} p_i\eta_t \boldsymbol{M}_i^t \odot \nabla F_k^t(\boldsymbol{W}_k^t), \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* >}_{D} \tag{45}$$

$$+ \underbrace{< \sum_{i\in[N]} p_i\eta_t [\boldsymbol{M}_i^t \odot \nabla F_k^t(\boldsymbol{W}_k^t) - \boldsymbol{M}_k^t \odot \nabla F_i^t(\boldsymbol{W}_i^t)], \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* >}_{E} \tag{46}$$

To bound $D$, we have

$$< -\sum_{i\in[N]} p_i\eta_t \boldsymbol{M}_i^t \odot \nabla F_k^t(\boldsymbol{W}_k^t), \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* > \tag{47}$$

$$\leq \sum_{i\in[N]} p_i\eta_t Q \|\boldsymbol{M}_i^t \odot \boldsymbol{M}_k^t\| < -\nabla F_k^t(\boldsymbol{W}_k^t), \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* > \tag{48}$$

$$\leq [-\eta_t Q(F_k^t(\boldsymbol{W}_k^t) - F_k^*) - \frac{\eta_t\mu Q}{2}\|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2]\Gamma(k,t) \tag{49}$$

where the first inequality results from Assumption 4, the second inequality results from the $\mu$-convexity of $F_k$, Cauchy-Schwarz inequality and AM-GM inequality.

To bound $E$, we have

$$< \sum_{i\in[N]} p_i\eta_t [\boldsymbol{M}_i^t \odot \nabla F_k^t(\boldsymbol{W}_k^t) - \boldsymbol{M}_k^t \odot \nabla F_i^t(\boldsymbol{W}_i^t)], \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* > \tag{50}$$

$$\leq \frac{1}{2}\eta_t \sum_{i\in[N]} p_i(B + \|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2) \tag{51}$$

$$= \frac{1}{2}\eta_t B + \frac{1}{2}\eta_t \|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2 \tag{52}$$

where the inequality results from Assumption 3.

We next focus on $B1-2$. By Assumption 5, Cauchy-Schwarz inequality and AM-GM inequality, we have

$$(\boldsymbol{M}_k^t \odot \boldsymbol{M}_k^t - \boldsymbol{M}_k^{t+1} \odot \boldsymbol{M}_i^{t+1}) \odot \sum_{i\in[N]} p_i\eta_t \nabla F_i^t(\boldsymbol{W}_i^t) \tag{53}$$

$$, \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* > \tag{54}$$

$$\leq \frac{1}{2}\eta_t G + \frac{1}{2}\eta_t \|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2 \tag{55}$$

By combining 43, 46, 49, 52 and 55, we have

$$B1 \leq -\eta_t Q\Gamma(k,t)[(F_k^t(\boldsymbol{W}_k^t) - F_k^*)] \tag{56}$$

$$-\frac{\eta_t}{2}[\mu Q\Gamma(k,t) - 2]\|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2 + \frac{\eta_t}{2}(B+G) \tag{57}$$

We next focus on $B2$. By apply Cauchy-Schwarz inequality and AM-GM inequality, we have

$$B2 = < C2, \hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^* > \tag{58}$$

$$\leq \frac{1}{2}(2\|C2\|^2 + \frac{1}{2}\|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2) \leq \quad P + \frac{1}{4}\|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2 \tag{59}$$

where the second inequality results from Assumption 5.
By combining 40, 57 and 59, we have

$$B \leq -2\eta_t Q\Gamma(k,t)[(F_k^t(\boldsymbol{W}_k^t) - F_k^*)] + \eta_t(B+G) + 2P \tag{60}$$

$$-\eta_t\{[\mu Q\Gamma(k,t) - 2] - \frac{1}{2}\}\|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2 \tag{61}$$

By choosing $\eta_t \leq \frac{Q}{KL\Gamma(k,t)}$, we have

$$2\eta_t^2 KL(F_k^t(\boldsymbol{W}_k^t) - F_k^*) - 2\eta_t Q\Gamma(k,t)[(F_k^t(\boldsymbol{W}_k^t) - F_k^*)] \leq 0 \tag{62}$$

By combining 37, 61 and 62, we have

$$\|\boldsymbol{W}_k^{\hat{t}+1} - \boldsymbol{W}_k^*\|^2 \leq \{1 - \{\eta_t[\mu Q\Gamma(k,t) - 2] - \frac{1}{2}\}\}\|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2 + \eta_t(B+G) + \eta_t^2 B + 3P \tag{63}$$

□

## B PROOF OF THEOREM 2

To convey our proof clearly, we introduce the following lemma in sparse phase. We defer the proof of this lemma after proving the main theorem.

Lemma 1. *Choose $\eta_s \leq \frac{Q}{KL\Gamma(k,t)}$ in sparse phase, we have*

$$\|\boldsymbol{W}_k^{\hat{t}+1} - \boldsymbol{W}_k^*\|^2 \leq \{1 - \{\eta_s[\mu Q\Gamma(k) - 1]\}\}\|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2 + \eta_s B(1 + \frac{Q}{KL\Gamma(k)}) \tag{64}$$

Proof. We let $\triangle t = \|\hat{\boldsymbol{W}}_k^t - \boldsymbol{W}_k^*\|^2$. From Lemma 1, it follows that

$$\triangle t + 1 \leq (1 - \eta_s R_k)\triangle t + \eta_s H_k, \tag{65}$$

where

$$R_k = \mu Q\Gamma(k) - 1 \tag{66}$$

$$H_k = B(1 + \frac{Q}{KL\Gamma(k)}) \tag{67}$$

From 69, it follows that

$$\triangle t + 1 - \frac{H_k}{R_k} \leq (1 - \eta_s R_k)(\triangle t - \frac{H_k}{R_k}), \tag{68}$$

Then we have

$$\|\boldsymbol{W}_{k,T} - \boldsymbol{W}_k^*\|^2 - \frac{H_k}{R_k} \leq (1 - \eta_s R_k)^{E(T-T_s)}(\|\boldsymbol{W}_{k,T_s} - \boldsymbol{W}_k^*\|^2 - \frac{H_k}{R_k}). \tag{69}$$

By the $L$-smoothness of $F_k$,

$$F_k(\boldsymbol{W}_k, T) - F_k^* \leq \frac{L}{2}||\boldsymbol{W}_{k,T} - \boldsymbol{W}_k^*||^2 \tag{70}$$

$$\leq \frac{L}{2}(1-\eta_s R_k)^{E(T-T_s)}(||\boldsymbol{W}_{k,T_s} - \boldsymbol{W}_k^*||^2 - Z_k) + \frac{L}{2}Z_k. \tag{71}$$

where

$$Z_k = \frac{H_k}{R_k} = \frac{B}{\mu Q \Gamma(k) - 1}(1 + \frac{Q}{KL\Gamma(k)}) \tag{72}$$

□

*Proof of Lemma 1.* To prove Lemma 1, it is necessary to introduce a key property of local mask in sparse phase ($t > Ts * E$):

$$\boldsymbol{M}_k^{t+1} = \boldsymbol{M}_k^t \tag{73}$$

Proof. From the property 73, it follows that $A2$, $B1-2$ and $B2$ would be zero. Then by applying new bounds of $A2$, $B1-2$ and $B2$ in 63 and choosing $\eta_s \leq \frac{Q}{KL\Gamma(k,t)}$ in sparse phase, we have

$$F_k(\boldsymbol{W}_{k,T}) - F_k^* \leq \frac{L}{2}(1-\eta_s R_k)^{U(T-T_s)}[F_k(\boldsymbol{W}_{k,T_s}) - F_k^* - Z_k] + \frac{L}{2}Z_k \tag{74}$$

where

$$Z_k = \frac{B}{\mu Q \Gamma(k) - 1}(1 + \frac{Q}{KL\Gamma(k)}) \qquad R_k = \mu Q \Gamma(k) - 1 \tag{75}$$

□

# C EXPERIMENT SETUP

**Table 4: Model configurations.**

| IC-CIFAR10 | IC-EMNIST | HAR |
|---|---|---|
| Conv 3-6 | Conv 1-10 | FC-300 |
| Maxpool | Maxpool | FC-100 |
| Conv 6-16 | Conv 10-20 | FC-6 |
| Maxpool | Maxpool | Softmax |
| FC-120 | FC-512 | |
| FC-84 | FC-62 | |
| FC-10 | Softmax | |
| Softmax | | |