FedMask: Joint Computation and Communication-Efficient Personalized Federated Learning via Heterogeneous Masking

Ang Li[1], Jingwei Sun[1], Xiao Zeng[2], Mi Zhang[2], Hai Li[1], Yiran Chen[1]
[1]Department of Electrical and Computer Engineering, Duke University
[2]Department of Electrical and Computer Engineering, Michigan State University
[1]{ang.li630, jingwei.sun, hai.li, yiran.chen}@duke.edu, [2]{zengxia6, mizhang}@msu.edu

ABSTRACT

Recent advancements in deep neural networks (DNN) enabled various mobile deep learning applications. However, it is technically challenging to locally train a DNN model due to limited data on devices like mobile phones. Federated learning (FL) is a distributed machine learning paradigm which allows for model training on decentralized data residing on devices without breaching data privacy. Hence, FL becomes a natural choice for deploying on-device deep learning applications. However, the data residing across devices is intrinsically statistically heterogeneous (i.e., non-IID data distribution) and mobile devices usually have limited communication bandwidth to transfer local updates. Such statistical heterogeneity and communication bandwidth limit are two major bottlenecks that hinder applying FL in practice. In addition, considering mobile devices usually have limited computational resources, improving computation efficiency of training and running DNNs is critical to developing on-device deep learning applications. In this paper, we present FedMask – a *communication and computation efficient* FL framework. By applying FedMask, each device can learn a *personalized* and *structured sparse* DNN, which can run efficiently on devices. To achieve this, each device learns a sparse binary mask (i.e., 1 bit per network parameter) while keeping the parameters of each local model unchanged; only these binary masks will be communicated between the server and the devices. Instead of learning a shared global model in classic FL, each device obtains a personalized and structured sparse model that is composed by applying the learned binary mask to the fixed parameters of the local model. Our experiments show that compared with status quo approaches, FedMask improves the inference accuracy by 28.47% and reduces the communication cost and the computation cost by 34.48× and 2.44×. FedMask also achieves 1.56× inference speedup and reduces the energy consumption by 1.78×.

CCS CONCEPTS

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

SenSys '21, November 15–17, 2021, Coimbra, Portugal

ACM ISBN 978-1-4503-9097-2/21/11…$15.00
https://doi.org/10.1145/3485730.3485929

KEYWORDS

Efficient federated learning systems, Data heterogeneity, Personalization, On-device AI

ACM Reference Format:
Ang Li[1], Jingwei Sun[1], Xiao Zeng[2], Mi Zhang[2], Hai Li[1], Yiran Chen[1]. 2021. FedMask: Joint Computation and Communication-Efficient Personalized Federated Learning via Heterogeneous Masking. In *The 19th ACM Conference on Embedded Networked Sensor Systems (SenSys '21), November 15–17, 2021, Coimbra, Portugal.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3485730.3485929

1 INTRODUCTION

Recent breakthrough in deep neural network (DNN) research [32] has empowered many mobile deep learning applications, such as image recognition [58], video analytics [46], object detection [39], sign language translation [14], etc. Developing these mobile applications requires a large amount of data for training DNNs. However, mobile devices often hold only limited data, which is not sufficient to locally train DNNs and achieve a desirable inference accuracy. One traditional solution is that mobile devices offload the data to a server and then a centralized training is performed on that server. Unfortunately, such centralized training method raises serious privacy concerns due to data offloading.

Federated learning (FL) [42] is an emerging distributed machine learning paradigm that is able to effectively address the above privacy issue. FL enables a number of devices to train a shared model in a federated fashion without transferring their local data. A central server coordinates the FL process, where each participating device communicates only the model parameters with the central server while keeping local data private. However, the data generated by mobile devices is often non-IID (identically and independently distributed) across these devices. For example, the input words to a virtual keyboard [20] on a mobile phone are often user and context dependent. Hence, these text data is highly heterogeneous across the devices. In addition, the communication and computational resources are either very limited or very expensive for mobile devices. In summary, statistical heterogeneity (i.e., non-IID), communication bandwidth, and computation cost are three critical bottlenecks [29, 34] in the practical applications of FL.

Status Quo and their Limitations. Since the introduction of FL, many studies attempt to mitigate the statistical heterogeneity via personalization including local fine-tuning [4, 13, 28, 30, 45, 54], multi-task learning [49], and contextualization [40]. However, most of these approaches require two separate steps: 1) training a global model in the federated fashion; and 2) fine-tuning the global model to generate a personalized model for each device using its local

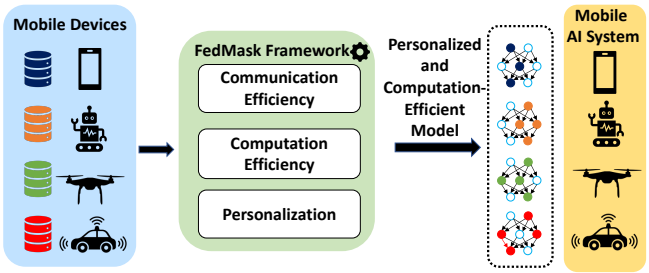

Figure 1: The high-level view of FedMask framework.

data. Such a two-step method inevitably incurs extra training efforts and overheads. In addition, some works aim to reduce communication cost, and the core idea is to compress the data communicated between the server and the devices. Common practices can be divided into three main categories: (1) *quantization* methods [2, 6, 10, 23, 29, 47, 56, 57] compress the communication by reducing the number of bits of each element in the transferred data; (2) *sparsification* methods [1, 12, 26] transmit only a subset of elements in the communicated data; and (3) *hybrid* methods [5, 27, 38, 50] combine quantization and sparsification.

However, very few efforts have been made to address the statistical heterogeneity and communication efficiency simultaneously. LG-FedAvg [37] leverages both local representation learning and global FL to learn personalized model for each participating device. Similar to aforementioned personalization methods, LG-FedAvg also requires two steps to achieve personalization. Specifically, LG-FedAvg partitions the model parameters into two groups, i.e., local parameters and global parameters. In the first step, each participating device optimizes and share the whole model with the central server like FedAvg [42]. In the second step, each device still updates the whole model but only the global parameters will be communicated with the central server, and hence the communication cost can be reduced. In doing so, each device can learn a personalized model due to its personalized local parameters. However, LG-FedAvg only improves communication efficiency in the second step and the heuristic partition of the model may lead to sub-optimal performance. In addition, LG-FedAvg is evaluated under an *unrealistic* FL setting, where each device holds sufficient training data (i.e., 200 images/class of MNIST and CIFAR-10). Such a condition implies that a device has already been able to obtain a good performance by training a model locally, which contradicts the intention of FL. For example, the classification accuracy of the model that is locally trained by each device can be as high as 97.17% on MNIST, but the performance of the model that is trained using FedAvg [42] only increases by 1.49% compared to the local model. HeteroFL [11] adaptively selects and assign a submodel to each device given the same base model. Each device only needs to train and communicate its submodel, and thus the both communication and computation cost can be reduced. However, the submodel is determined based on the computation capability of each device rather than the local data, leading to less flexible submodel allocations. Most importantly, none of the existing FL methods is designed with constraints of computation cost in training and inference. However, computation cost is a critical challenge for mobile devices due to their limited onboard resources.

Table 1: Comparison between FedMask and existing FL frameworks.

Method	Computation Efficiency	Communication Efficiency	Personalization
FedAvg [42]	X	X	X
Top-: [1]	X	X	X
Per-FedAvg [13]	X	X	X
LG-FedAvg [37]	X	X	X
FedMask	✔	✔	✔

Overview of the Proposed Approach. Motivated by the limitations of existing works, in this work, we propose FedMask, a *unified* FL framework that simultaneously (1) minimizes the *communication* cost; (2) reduces the *computation cost* for both *training* and *inference*; and (3) learns a *personalized* model for each participating device to mitigate statistical heterogeneity within a *single step*. As illustrated in Figure 1, FedMask automatically learns a personalized and sparse DNN model for each device. Such a model can be deployed for efficient mobile deep learning applications.

To achieve the above three objectives, the key principle behind FedMask is to exploit the *redundancy* in both communication and computation during FL via the learned heterogeneous binary masks. As Equation 1 shows, the total cost of FL training is the sum of computation cost and communication cost as follows.

$$\text{Total Cost}=\text{Communication}+\text{Computation (gradient+mask)} \quad (1)$$

In particular, the computation cost includes the back-propagation via calculating the gradients and learning the heterogeneous binary mask. Although learning the mask incurs extra computation overhead, each device can learn a compact local model by leveraging the learned mask. Performing back-propagation on the compact model reduces the computation cost compared to the operations on the original local model. Since the overhead of learning the mask via pruning is negligible compared to that of performing back-propagation, the computation cost is thus significantly reduced. Moreover, since transmitting the binary mask is significantly more efficient than transmitting the local model, the communication cost is also reduced. As a result, the total cost of FL training is significantly reduced.

At the core of FedMask, each device learns a heterogeneous and structured sparse binary mask. In particular, before initializing local model training, each device first determines a heterogeneous structure of the binary mask using a pruning method based on the local data, and then optimizes the binary mask with a structured sparsity regularization. Such binary mask is then element-wisely applied to the local model weights to generate a sparse model for each device and the *computation cost* for both training and inference are reduced. Since only the binary mask (i.e., 1 bit per element) of each device is communicated in FedMask, the *communication cost* can be significantly reduced compared to transmitting the weights (i.e., 32 bits per element) of each local model. As data distributions across different devices are non-IID, the learned binary masks are heterogeneous across devices. Such heterogeneity embeds *personalized* information of the data distribution on each device.

System Implementation and Evaluation Results. We implemented FedMask and conducted extensive experiments to evaluate its performance. In particular, we applied FedMask to build three representative FL applications on mobile phones: image classification, human activity recognition, and next-character prediction. We have compared FedMask against five competitive and status quo FL methods as our baselines. Our results show that:

- Compared to the baseline methods, FedMask improves inference accuracy by 2.43%-28.47% while reducing communication cost by 32.25×-34.48× and saving 1.37×-2.44× computation cost during training.
- At runtime, the structured sparse and personalized model generated by FedMask significantly outperforms those trained with the baseline methods. FedMask achieves up to 1.56× speedup in inference latency while reducing as much as 39.87% memory footprint and 1.78× energy consumption.

Summary of Contributions. To the best of our knowledge, FedMask represents the first framework that jointly improves the communication and computation efficiency of personalized FL in a unified manner. Table 1 provides a comparison between FedMask and status quo FL methods. FedMask proposes a series of novel techniques that effectively address the limitations of status quo methods. In particular, FedMask differs from Top-$\dot{\iota}$ and Per-FedAvg as it improves the communication efficiency and achieves personalization simultaneously. Compared to LG-FedAvg, FedMask achieves personalization in a single step without local fine-tuning. Most importantly, FedMask is the only FL framework that takes the optimization of computation efficiency into consideration. We believe our work represents a significant step towards enhancing the efficiency of FL systems.

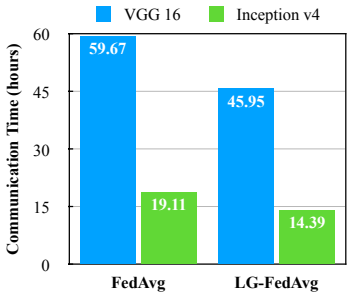

Figure 2: Comparison of communication cost between FedAvg and LG-FedAvg.

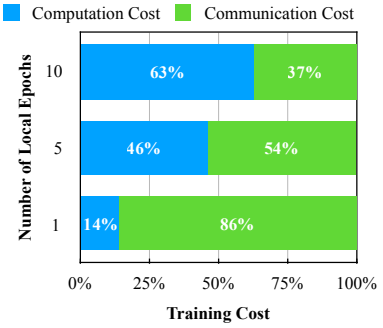

Figure 3: Training cost with different numbers of local training epochs.

2 BACKGROUND AND MOTIVATION

We begin with introducing the background of FL and the need for personalization. We then show the necessity of simultaneously reducing computation and communication cost in FL as well as the limitations of the status quo approaches, which are the key motivation of our work.

2.1 Background on Federated Learning and the Need for Personalization

Federated learning enables decentralized training without sharing local data [53]. FedAvg [42] is one of the most widely used FL methods. As presented in FedAvg, there is a central server and a number of devices acting as clients. At each communication round, a subset of the devices are selected. Each selected device performs training using local data with the same learning rate and number of local epochs to train a local model. Each local model is updated using stochastic gradient descent (SGD). The devices then transmit their local model updates to the central server, where these local model updates are averaged and the global model is updated accordingly. Finally, the global model is sent back to each device. In practice, the distribution of the data across the devices is inherently non-IID. Such statistical heterogeneity makes it difficult to train a shared global model that can be well generalized for all devices [35, 42]. Hence, personalization becomes necessary to handle the challenges posed by statistical heterogeneity.

2.2 Communication Constraint

Communication cost is a key bottleneck for FL. LG-FedAVg [37] is a state-of-the-art FL method that attempts to reduce communication cost and achieve personalization simultaneously. We conduct experiments to investigate how significantly the communication cost is reduced by LG-FedAvg compared with FedAvg. In this experiment, we apply FedAvg for training on CIFAR-10 for 2,000 epochs with 5 local epochs and adopt the VGG16 [48] and Inception-v4 [52] as the default model configurations. We assume the available bandwidth is 10 MB/s for each device. As Figure 2 shows, it takes 59.67 hours and 19.11 hours in terms of communication cost when applying FedAvg to train the VGG16 and Inception-v4, respectively. Unfortunately, such a high communication cost is unaffordable for mobile devices. Applying LG-FedAvg can reduce the communication cost to 45.95 hours and 14.39 hours accordingly, indicating 1.30× and 1.33× communication cost reductions. However, such an improvement of communication efficiency is not significant enough because the communication cost is still too high to be affordable for mobile devices. Therefore, we aim to push the state-of-the-art forward, i.e., to reduce communication cost by an order-of-magnitude compared to LG-FedAvg.

2.3 Computation Constraint

Besides communication cost, computation cost is another critical challenge for FL. As presented in FedAvg [42], the number of local epochs is a crucial hyperparameter for convergence. Specifically, performing more local epochs requires more local computation efforts but reduces communication cost. However, a larger number of local epochs may lead to divergence because each local model may be updated towards the optima of its local objective which is opposed to the global objective when performing more local epochs. We conduct experiments to investigate the bottleneck of FL training by dividing the total training cost into communication cost and computation cost. In our experiment, we keep the same setting as the experiment settings in §2.2 but use various local epochs of 1, 5 and 10. In addition, we only adopt VGG16 as the default model configuration for training. As Figure 3 shows, performing more local epochs results in higher local computation cost but less communication cost. For example, when performing 10 local epochs on each device, the computation cost becomes the bottleneck of FL training, i.e., computation cost accounts for 63% of the total training cost. If each device performs less local epochs, the communication cost will become the bottleneck of FL training. For example, communication cost comprises 86% of the total training cost when performing one local epoch on each device. However, the optimal number of local epochs is task-dependent and we need to tune this parameter based on the specific learning task. Therefore, it is necessary to reduce both communication and computation costs to improve the training efficiency. FedMask is designed to achieve this purpose.

3 FEDMASK DESIGN

3.1 Overview

Figure 4 depicts an overview of the proposed FedMask framework.

At the initialization stage, the central server randomly initializes a model and then distributes it to the participating devices. To support learning *personalized* models under statistically heterogeneous settings, each device learns a heterogeneous binary mask via the proposed one-shot pruning method (①).

In each communication round, a set of devices is randomly selected to participate in FL training. Optimizing the masks while keeping the model parameters unchanged has been proved to be an effective alternative of model training [60]. Therefore, instead of optimizing local model parameters, in FedMask, each device optimizes the binary mask with a structured sparsity regularization while freezing the parameters of local model (②-a). As such, only the optimized binary masks are transmitted from the devices to the central server (②-b). Compared to FedAvg where local model parameter updates (32 bits per element) are communicated, transmitting binary masks can save about 97% of the communication cost, which is a significant improvement on *communication efficiency*.

The central server performs the aggregation on the received binary masks. The aggregation strategy is specifically designed such that only the elements that are overlapped across the binary masks of the devices are aggregated while keeping non-overlapping elements unchanged (③-a). In doing so, the personalization of the binary masks is preserved and the updated binary masks will be sent back to each device (③-b).

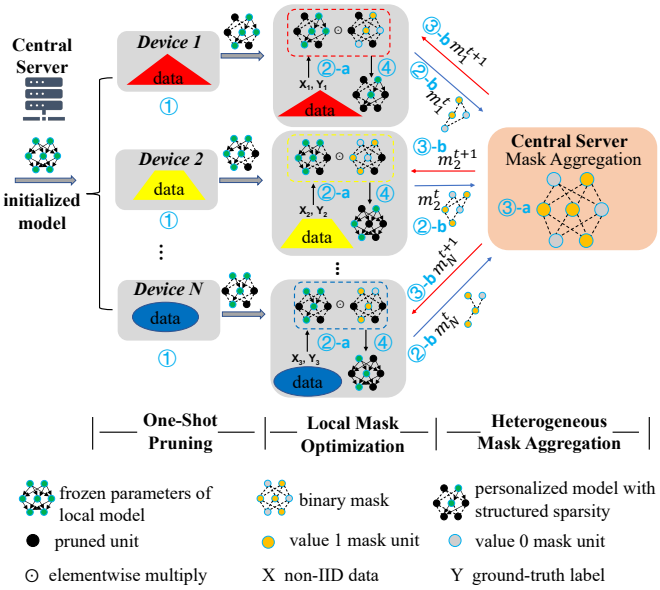

Figure 4: Overview of FedMask framework.

The above process (②-③) repeats until reaching a predefined number of communication rounds.

Finally, the binary mask will be elementwise applied to the frozen parameters to generate a personalized and structured sparse model (④). Since performing the feedforward and back-propagation computation over the structured sparse model is much more efficient compared to the original dense model, the *computation efficiency* for both training and inference is significantly improved as well.

3.2 Design Challenges

The design of FedMask has three key challenges.

Challenge#1: How to jointly improve communication and computation efficiency? In FedMask, the key to jointly improving communication and computation efficiency is the binary masks. In classical FL methods, each device optimizes local model weights using their local data via SGD. In contrast, as illustrated in Figure 4, FedMask aims to optimize the binary masks other than local model parameters and only transmits the optimized binary masks to the central server. Therefore, we need to design a training method to optimize the binary mask such that the performance of combining the learned binary mask and the frozen local model is at least comparable to that of directly optimizing local model parameters. However, directly applying existing back-propagation algorithms like SGD is technically infeasible because elements of the mask are in binary values and 0 values will stop passing corresponding gradients backward. Hence, we need to design a novel training method for the binary mask. Furthermore, if we optimize the binary mask without any constraints, combining the learned binary mask and the frozen local model may lead to non-structured sparsity, which is not hardware-friendly to improve computation efficiency [55].

Challenge#2: How to incorporate personalization for each device? In FedMask, the key to incorporating personalization is

the construction of heterogeneous binary masks which embed personalized information of local data distribution into the structure of each device's own binary mask. Pruning is a natural option to obtain data-dependent structure of the binary mask. However, existing pruning methods are designed for model weights with floating-point values rather than for masks with binary values. As such, they cannot be directly applied to binary masks in FedMask. We need to design a pruning method for generating heterogeneous binary masks to incorporate personalization for each device.

Challenge#3: How to aggregate heterogeneous binary masks on the central server while preserving personalization? Compared to classical FL methods, there are two key differences in the aggregation on the central server in FedMask: (1) the aggregation is operated on binary masks instead of model parameters; (2) these binary masks are heterogeneous rather than the same model architecture. Therefore, existing aggregation strategies cannot be directly applied. Designing an aggregation strategy for heterogeneous binary masks while preserving personalization represents another challenge.

Addressing the above challenges is not trivial. With careless design, heterogeneous binary masks cannot be effectively learned to represent personalized information while jointly improving communication and computation efficiency, and the aggregation may destroy the personalized information embedded in the heterogeneous structure of the binary masks.

In the remainder of this section, we first present the preliminaries on binary mask optimization in §3.3, which lays the foundation for the proposed techniques in FedMask. We then elaborate the key techniques we have developed in FedMask in §3.4 (①), §3.5 (②), §3.6 (③), and §3.7 (④).

3.3 Binary Mask Optimization

The key difference between FedMask and existing FL methods is that each device learns a heterogeneous binary mask and only transmit such binary mask to the central server. Learning the binary mask while freezing the model parameters is the foundation of our proposed techniques in FedMask. Therefore, we first present the preliminaries on binary mask optimization in this section.

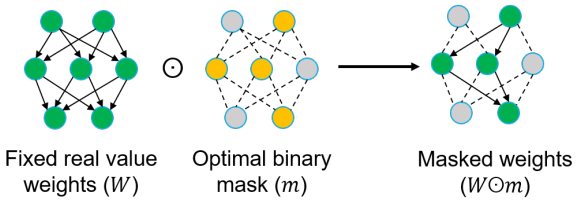

Figure 5: The process of masking weight.

For simplicity, we take the fully connected layer as an example to illustrate our proposed binary mask optimization method. Note that this method can also be easily applied to other types of layers (e.g., convolutional layers) in DNNs. For ease of notation, the bias term is ignored. A typical fully connected layer can be defined as $\sim = , \cdot G$, where $\sim \in \mathbb{R}^{<}$ is the output, $G \in \mathbb{R}^{=}$ is the input, and $, \in \mathbb{R}^{<\times=}$ denotes the weight matrix of the fully connected layer. Inspired by previous work [60], a mask based fully connected layer can be composed by applying a binary mask $< \in \{0, 1\}^{<\times=}$, which has the same size as $,$, to the fixed weight matrix via elementwise multiplication as illustrated in Figure 5. The mask based fully connected layer can be formulated by Equation 2:

$$\sim = (, \odot <) \cdot G, \tag{2}$$

where $\odot$ indicates the elementwise multiplication. However, it is technically infeasible to apply existing optimization algorithms (e.g., SGD) to $<$ due to its binary value. Therefore, we introduce a real-valued mask $<^A \in \mathbb{R}^{<\times=}$ to design the binary mask optimization algorithm. In the feedforward step, $<^A$ is binarized to $<$ using a threshold function, which is defined as Equation 3.

$$<_{89} = \begin{cases} 1, & <^A_{89} \geq g \\ 0, & <^A_{89} < g \end{cases}, \tag{3}$$

where $<_{89}$ represents the element located in 8th row and 9th column of $<$. In the back-propagation step, the gradients of $<$ are calculated using Equation 4.

$$\frac{m!}{m<} = \frac{m!}{m\sim} \cdot G^{)} \odot , , \tag{4}$$

Because the threshold function is non-differentiable, the existing method [60] directly applies the gradients of $<^A$ to that of binary mask $<$ as shown in Equation 5.

$$\frac{m!}{m<^A} = \frac{m!}{m<}, \tag{5}$$

Even though this strategy can realize the optimization of $<^A$ and $<$, it may cause a huge gradient scale variance which will impair the optimization of $<^A$ [9]. To reduce this gradient variance, we incorporate a sigmoid function denoted as $f(\cdot)$, to approximate the binarization from $<^A$ to $<$ during local training by Equation 6.

$$<_{89} = f \left(<^A_{89}\right)^{"} \tag{6}$$

By introducing the differentiable sigmoid function instead of the hard threshold function, the gradients of $<$ can be back-propagated to $<^A$ as shown in Equation 7:

$$\frac{m!}{m<^A} = \Psi \odot \frac{m!}{m<}, \tag{7}$$

where Ψ represents the gradient matrix of the sigmoid function and its values are explicitly calculated by Equation 8:

$$\Psi_{89} = <_{89} \cdot (1 - <_{89})^{"} \tag{8}$$

When transmitting $<$ to the central server and performing the feedforward operation, the result from Equation 6 will be binarized by applying a threshold function similar to Equation 3, and we set the threshold as 0.5.

3.4 One-Shot Pruning for Mask Initialization

The first step of FedMask is to learn a heterogeneous binary mask from each device's local data at the initialization stage. To this end, inspired by existing weight pruning methods [18, 24, 36], we design a one-shot pruning method to learn the heterogeneous structure using local data. Weight pruning [18] has been a popular technique for model compression. By pruning parts of the model parameters, the significant model parameters are preserved for preserving the

performance. There were various methods to determine the significance of parameters, such as threshold [18], kernel sparsity and entropy [36], filter importance [24], etc. However, existing methods are designed for real-valued parameters, and thus they cannot be directly applied to prune the binary mask in FedMask.

Since we have introduced the real-valued mask in the binary mask optimization algorithm (§3.3), one naïve solution is to directly apply existing weight pruning methods to the real-valued mask. However, based on Equation 4 and Equation 7, the gradients of real-valued masks are directly scaled by the values of fixed weight. Therefore, using only the magnitude of real-valued masks is not a fair criterion for pruning. Based on this observation, we design the one-shot pruning method based on the combination of real-valued masks and the fixed weights, i.e., $, \odot <^A$.

As described in §3.1, each device preserves the dense structure of the top layers in the binary mask and only prunes the last several layers which compose the classifier part. Instead of using the absolute value after optimization, we use the absolute changing value of $, \odot <^A$ to prune masks after optimization. We define the pruning rate as $?_A$, and the pruning procedure consists of two steps: (1) devices update their real-valued masks for one epoch; (2) by sorting the values of elements in $, _{89} \cdot <^A_{89}$, the largest $?_A$ portion of elements will be preserved, and the rest elements will be set to zero and frozen. Local training is then performed based on the heterogeneous binary mask.

3.5 Local Binary Mask Optimization

After performing the one-shot pruning at the initialization stage, each device obtains a heterogeneous mask and can start to perform local mask optimization in each round of FL. In particular, in each round of FL, each device iteratively optimizes the binary mask using local data while keeping local model parameters frozen, such that the combination of the learned binary mask and fixed local model parameters results in a model with desirable performance.

To improve computation efficiency on device in a hardware-friendly way, we employ the structured sparsity regularization during mask optimization to learn binary masks with structured sparsity. We aim to achieve channel-wise and filter-wise sparsity for convolutional layers, row-wise and column-wise sparsity for fully connected layers. Suppose a network with $!_{2>=E}$ convolutional layers and $!_{52}$ fully connected layers, the structured sparsity regularization for the mask of this network is formulated as:

$$
\begin{aligned}
'_6(<) &= '_{2>=E}(<) + '_{52}(<)\cdot \\
'_{2>=E}(<) &= \sum_{;=1}^{!_{2\tilde{O}=E}} \Big(\sum_{=;=1}^{\tilde{O}} \|<^{(;)}_{=,\cdot,\cdot,;}\|_6 + \sum_{2;=1}^{\tilde{O}} \|<^{(;)}_{;2,\cdot,\cdot}\|_6 \Big)\cdot \\
'_{52}(<) &= \sum_{;=1}^{!_{\tilde{O}?}} \Big(\sum_{A>F;=1}^{'_{\tilde{O}F;}} \|<^{(;)}_{A>F,\cdot,;}\|_6 + \sum_{2>;;=1}^{\tilde{O}_{;}} \|<^{(;)}_{;,2>;;}\|_6 \Big)\cdot
\end{aligned}
\tag{9}
$$

where $\|"\|_6$ is the group Lasso [55], $\|<^{(6)}\|_6 = \sqrt{\sum_{8=1}^{|<^{(6)}|} (<^{(6)}_8)^2}$, $|<^{(6)}|$ is the number of parameters in $<^{(6)}$. After applying this structured sparsity regularization, the training loss for mask optimization is formulated as:

$$
!(<) = !\ \ (<) + \_' '_6(<) \tag{10}
$$

where $!\ \ (<)$ is the loss on data and $\_'$ is the coefficient of structured sparsity regularization.

To demonstrate the effectiveness of our proposed method, we conduct experiments to compare the performance between the model learned by applying our mask based method and the one trained via directly optimizing the model parameters. In this experiment, the models are trained locally without considering the federated fashion. In particular, for each method, we train a 2-layer convolutional network (CNN), a 3-layer multilayer perceptron (MLP), and a 1-layer long short-term memory (LSTM) using MNIST. The structured sparsity regularization coefficient $\_'$ is set as $24 - 4$ for each network. The test accuracy is reported in Table 2. Although there is only tiny performance drop on the CNN and MLP when applying our proposed method, the performance of the LSTM decreases significantly.

Table 2: Accuracy comparison of the training method: optimizing model parameters vs. optimizing binary masks.

Method	CNN	MLP	LSTM
optimizing model parameters	97.15%	96.59%	96.35%
optimizing binary masks	96.85%	96.07%	89.23%

To investigate the reason of performance drop on the LSTM, we look into the formulation of the mask based LSTM. LSTM [22] is explicitly designed to address the long-term dependency problem caused by traditional recurrent neural networks (RNNs). As same as other RNNs, LSTM has a chain-like structure. However, besides hidden signal $_C$, the cell state 2_C is also involved in the information flow through the chain. In addition, different from the simple RNNs having a single neural network layer, LSTM has four units: $, _8$, $, _5$, $, _6$, $, _>$. These four units are interconnected in a specific way described as Equation 11.

$$
\begin{aligned}
8_C &= f\,(, _8\,[G_C\cdot\ _{C-1}]) \\
5_C &= f\ \ , _5\,[G_C\cdot\ _{C-1}] \\
6_C &= C0= \ \ , _6\,[G_C\cdot\ _{C-1}] \\
>_C &= f\,(, _>\,[G_C\cdot\ _{C-1}]) \\
2_C &= 5_C 2_{C-1} + 8_C 6_C \\
_C &= >_C C0= \ (2_C)
\end{aligned}
\tag{11}
$$

Based on the above formulation of LSTM, the mask based LSTM is composed by applying binary masks to all the four units, which is defined as Equation 12.

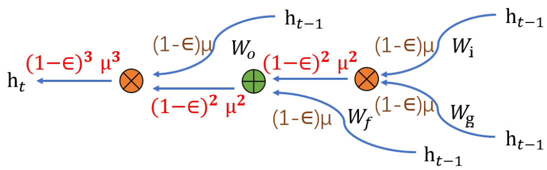

Figure 6: Expressiveness decay accumulated through the computation flow in LSTM.

$$\begin{cases} 8_C = f\left((,_8 \odot <_8)\,[G_C \bullet \ _{C-1}]\right) \\ 5_C = f \quad ,_5 \odot <_5 \quad [G_C \bullet \ _{C-1}] \\ 6_C = CO= \quad ,_6 \odot <_6 \quad [G_C \bullet \ _{C-1}] \\ >_C = f\left((,_> \odot <_>)\,[G_C \bullet \ _{C-1}]\right) \\ 2_C = 5_C 2_{C-1} + 8_C 6_C \\ \ _C = >_C CO=\ (2_C) \end{cases} \quad (12)$$

where $<_8$, $<_5$, $<_6$ and $<_>$ represent the corresponding binary masks of the four units in LSTM accordingly. Even though the performances drop of the mask based CNN and MLP is neglectable, the expressiveness of mask based DNN models is limited compared to ones than trained by directly optimizing model parameters. However, this expressiveness reduction will be even worse in LSTM, because it will be accumulated through the nested mask structure as presented in Equation 12. Such an accumulated expressiveness reduction results in the significant performance drop of LSTM. To further illustrate the accumulated expressiveness reduction, we define the expressive ability of a real-valued unit that is directly optimized as `, and the expressiveness decay of a masked unit as $n$ and $n$ Ÿ 1. Then, during one feedforward process, the expressiveness of the mask based LSTM will be decayed to $(1-n)^{3}$`^{3}, where the expressiveness decay is accumulated to $1-(1-n)^3$ when the information flow passes through $_{C-1}$ to $_C$ as shown in Figure 6.

To reduce the expressiveness decay in LSTM, we can remove the binary mask from either of $,_>$, $,_6$, and $,_8$. In doing so, the expressiveness decay can be reduced to $1-(1-n)^2$ with only about 25% additional communication cost. It is not useful to remove the binary mask of $,_5$, because this cannot help to reduce the expressiveness decay. Such an observation is demonstrated by empirical results in Table 3. Based on this key observation, we design a partial mask for LSTM where the mask based optimization is applied to $,_5$, $,_6$, and $,_8$, but directly optimizing the real-value $,_5$.

Table 3: Accuracy comparison between different masking strategy on LSTM (dataset: MNIST).

Strategy	Accuracy (%)
LSTM	96.35
LSTM+Mask	89.23
LSTM+Mask+$,_5$ real	91.13
LSTM+Mask+$,_?$ real	94.53
LSTM+Mask+$,_>$ real	95.25
LSTM+Mask+$,_8$ real	94.85

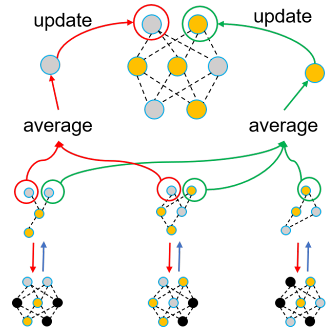

Figure 7: Mask aggregation process.

After the local mask optimization, devices transmit the learned binary masks to the central server, where the aggregation is performed on these binary masks.

3.6 Aggregate Heterogeneous Binary Masks

Most FL methods perform the aggregation using the averaging strategy from FedAvg [42]. However, the aggregation is performed on binary masks rather than real-valued local updates in FedMask. Moreover, binary masks are heterogeneous across devices, which means not all the elements are overlapped across binary masks. Thus, we cannot simply apply the averaging strategy to perform the aggregation in FedMask.

When designing the aggregation strategy, our goal is to maximally retain the personalized information embedded in heterogeneous binary masks. To achieve this goal, we propose a mask aggregation scheme where each element in binary masks is aggregated independently. As Figure 7 shows, the central server only performs the averaging aggregation on elements that appear in two or more binary masks. Then, the central server updates these elements in corresponding binary masks using the aggregated value. For the elements that are not shared among binary masks, the central server keeps them unchanged without performing aggregation. Finally, the updated binary masks with preserved heterogeneous structures will be sent back to devices accordingly. Figure 8 shows an example of the personalization-preserving mask aggregation. In this example, only the first channel of *Device* 8 (in yellow) and *Device* 9 (in blue) are intersected, and hence the mask of these two channels are averaged by the proposed aggregation scheme (in green). Because the third channel of *Device* 9 is pruned, there is no intersection of the third channel between *Device* 8 (in yellow) and *Device* 9 (in white). Therefore, the mask of the third channel of *Device* 8 will not be changed.

3.7 Final Personalized Model Generation

As the final step, as illustrated in Figure 9, once the training of the heterogeneous binary masks is finished, the heterogeneous binary mask of each device is elementwise applied to the frozen local model to generate the personalized model. In addition to addressing the statistical heterogeneity, the heterogeneous binary mask results in a more sparse local model such that both the communication and computation cost can be further reduced.

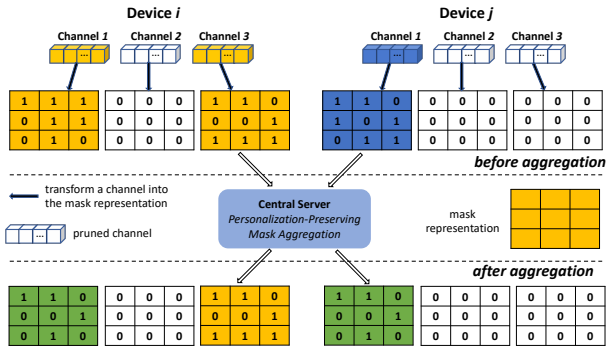

Figure 8: Illustration of the personalization-preserving mask aggregation on the central server. The *yellow* and *blue* matrices represent the unpruned masks, the green ones stand for the updated masks which are intersected between devices.

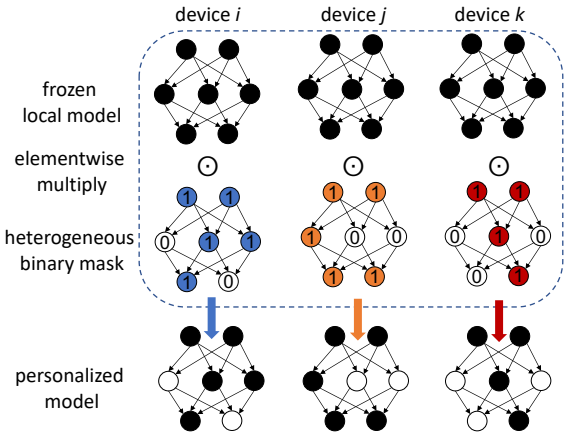

Figure 9: Illustration of achieving personalization via heterogeneous binary masks.

Putting All the Pieces Together: we summarize the complete FedMask framework in Algorithm 1.

Algorithm 1: FedMask.

Data: ($_1$, """, $_\#$) where $_:$ is the local data on $:$th device
Server Executes:
initialize the global model ,
initialize the global mask $<^0$
(← { $_1$, """, $_\#$ }
for $_:$ ∈ (***in parallel*** **do**
  download , from the central server
  $<_:$ ← ClientPruning($_:$, $<^0$) /* ① in Figure 4 */
for *each round*) = 1, 2, """ **do**
  $:$ ← max(# × , 1) /* # available devices, random sampling rate */
  (C ← { $_1$, """, $_:$ } /* the selected $:$ participating devices indexed by $:$ */
  for $_:$ ∈ (C ***in parallel*** **do**
    $<_:^{)+1}$ ← ClientUpdate($_:$, $<^)$ ⊙ $<_:$) /* ②-a and ③-b in Figure 4 */
  $<^{)+1}$ ← (aggregate {$<_:^{)+1}$}) /* using the proposed personalization-preserving aggregation (③-a in Figure 4) */
ClientUpdate ($_:$, $<_:^)$):
$<^A$ ← (initialize a real-valued mask using $<_:^)$)
$\mathcal{B}$ ← (split local data $_:^{CAO8=}$ into batches);
for *each local epoch* 8 *from* 1 *to* **do**
  for *batch* 1 ∈ $\mathcal{B}$ **do**
    $<^A$ ← $<^A$ − [∇$_{<^A}$ $_:$(, ⊙ $f(<^A)$; 1) /* [is the learning rate, $_:$(·) is the loss function */
$<_:^{)+1}$ ← $f(<^A)$ /* binarize $<^A$ with threshold 0.5 */
return $<_:^{)+1}$ to server /* ②-b in Figure 4 */
ClientPruning ($_:$, $<$):
$<^A$ ← (initialize a real-valued mask using $<$)
$\mathcal{B}$ ← (split local data $_:^{CAO8=}$ into batches);
for *batch* 1 ∈ $\mathcal{B}$ **do**
  $<^A$ ← $<^A$ − [∇$_{<^A}$ $_:$(, ⊙ $f(<^A)$; 1) /* [is the learning rate, $_:$(·) is the loss function */
$<^A$ ← (prune $<^A$ by preserving the largest $?_A$ portion of elements in $|, _{89} \cdot <^A_{89}|$)
$<_:$ ← $f(<^A)$ /* binarize $<^A$ with threshold 0.5 */
return $<_:$ to server

4 EVALUATION

4.1 Applications, Datasets, and Models

To demonstrate the generality of FedMask across applications, we apply FedMask to build three representative mobile AI applications that benefit significantly from FL. The information of the datasets that are applied to build these applications is summarized in Table 4. We follow the non-IID configurations in [33] to build the non-IID datasets in FL.

Application#1: Image Classification (IC). Image classification is a popular computer vision application to classify images into categories. With the increasing computation capabilities on device, image classification has become attractive to be deployed on mobile devices. In this work, we use VGG16[48] as the base model for training. We use EMNIST [8] and CIFAR10 [31] datasets to develop two image classification applications, i.e., IC-CIFAR10 and IC-EMNIST. EMNIST is a handwriting image classification dataset grouped by the writers, and 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 also follows the same distribution as the training data on each device.

Table 4: Statistical information of datasets.

Dataset	Number of devices	Average samples per device	Classes	Non-IID
EMNIST [8]	2414	232.8	64	X
CIFAR10 [31]	400	25	10	X
HAR [3]	30	364.3	6	X
Shakespeare [42]	1129	3743.2	80	X

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 is developed for recognizing various activities performed by a user based on the sensor data. In this work, we use HAR [3] dataset to build this application. HAR collects smartphone accelerometer and gyroscope data from 30 individuals, including six labeled activities: walking, walking-upstairs, walking-downstairs, sitting, standing, and lying-down. We naturally distribute each individual's data to one smartphone. We employ a 3-layer fully connected neural network to recognize human activities.

Application#3: Next-Character Prediction (NCP). Next-character prediction is a very practical application on smartphones, e.g., text auto-completion in the virtual keyboard. This application aims to predict what character comes next given then current input. We apply Shakespeare [42] dataset to develop this application. This dataset is built on *The Complete Works of William Shakespeare* by separately extracting different roles' dialogues. In this dataset, the dialogues are distributed to devices according to the speaking role. We build an RNN constructed by an 8-D encoder, including two LSTM layers and three fully connected layers, as the base model for this application.

4.2 System Implementation

We implemented three FL applications on NVIDIA Jetson TX2 and Raspberry Pi 4. The central server is equipped with an Intel Xeon E5-2630@2.6GHz and 128G RAM. We used Monsoon power monitor [44] to measure the power consumption at runtime. For the baselines and FedMask, we adopt a FL protocol which randomly feeds 20 data shards that are partitioned as described in Table 4 to each smartphone in each communication round, i.e., 20 participating devices per communication round. Each participating device performs 5 local training epochs for one communication round. For FedMask, we apply the proposed one-shot mask pruning method to the last 8 layers of VGG16 and the last 2 layers of the base models for HAR and NCP. In addition, we set the pruning rate $?_A$ as 0.2, which means only 20% of mask elements are preserved in those pruned layers.

4.3 Experimental Setup

Baselines. To comprehensively evaluate the performance of FedMask, we compare FedMask against six baselines:

- **Standalone** trains a model using local data only on each device without collaborations between devices. Note that there will be no communication cost for Standalone method. To make fair comparisons, we provide the same total training time for Standalone method and FedMask, i.e., the total number of local training epochs in Standalone method will be greater than that of FedMask.
- **FedAvg** [42] is the most classical FL method and has been applied to commercial products [7]. 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 $:$ elements of the gradients. Here we set $:$ as 10%.
- **BNN-FedAvg** applies FedAvg to train a Binary Neural Network (BNN) [25], which is an inherently sparse model with binary weights. We use the same configurations of BNN as presented in [25]. Similar with FedMask, the parameters communicated between devices and the server in BNN-FedAvg are also binary. However, the optimization is actually performed on real-value parameters during local training on each device. Therefore, it is necessary to map the real-value parameters to the binary ones before transmit these parameters to the server and convert them back after receiving the updated binary parameters from the server.
- **Per-FedAvg** [13] incorporates MAML [15], which is a popular meta-learning method, into FedAvg for personalization.
- **LG-FedAvg** [37] is the state-of-the-art FL method that enables personalization and improves communication efficiency but not computation efficiency.

In addition, for each application, we adopt the same base model configuration for each baseline method as FedMask. We also apply the same training settings and data configurations as FedMask to baseline methods, the results are reported after the same number of training epochs except for Standalone.

Evaluation Metrics. We evaluate both training and runtime performance of FedMask using two sets of metrics:

- **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; (3) *computation cost:* we measure the time cost for computation performed on devices of 2000 training epochs, and normalize it as the ratio to the computation time of FedAvg. We report the normalized computation time as the computation cost.
- **Metrics for Runtime Performance:** (1) *memory footprint:* we measure the memory footprint of different applications on device, and calculate the memory footprint reduction; (2) *inference time:* we measure the average time cost of performing one inference and report the inference time reduction percentage; (3) *energy consumption:* we measure the average energy consumption per inference and calculate the energy consumption saving percentage.

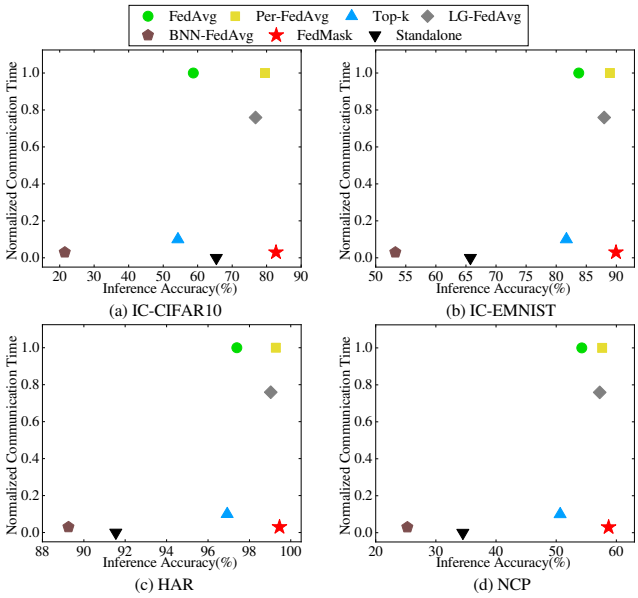

Figure 10: Comparison between FedMask and baselines in inference accuracy-communication cost space.

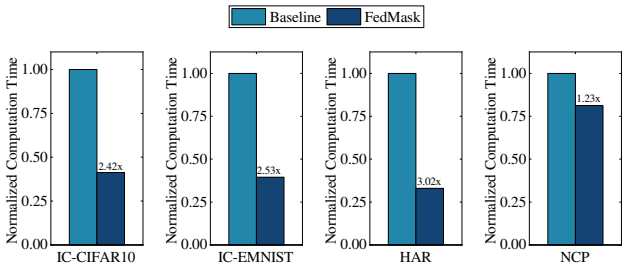

Figure 11: Comparison between FedMask and baselines in computation cost space.

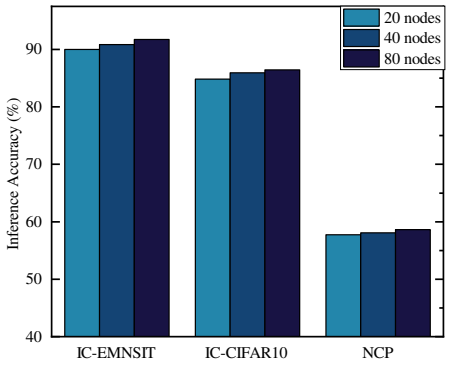

Figure 12: The impact of the number of participating devices on FedMask performance.

4.4 Training Performance

Inference Accuracy vs. Communication Cost: We compares FedMask with the baselines in terms of the accuracy-communication tradeoff. The results in Figure 10 demonstrate that FedMask is able to achieve a large gain in inference accuracy with a significant reduction on communication cost.

First, compared to LG-FedAvg, FedMask is able to improve inference accuracy and communication efficiency simultaneously. In particular, FedMask improves inference accuracy by 5.97%, 1.97%, 0.43%, and 1.49% on IC-CIFAR10, IC-EMNIST, HAR, and NCP, respectively. Besides, it also reduces 25.64×, 24.39×, 25.64×, 25× communication cost in these applications, respectively.

Second, FedMask is able to dramatically reduce communication cost compared to Per-FedAvg that is specifically designed for personalization. In particular, FedMask reduces 32.25×, 31.25×, 34.48×, 32.25× communication cost on IC-CIFAR10, IC-EMNIST, HAR, and NCP, respectively. More surprisingly, FedMask also achieves a substantial improvement in inference accuracy. FedMask increases inference accuracy by 3.17%, 0.96%, 0.12%, and 1.09%, respectively.

Third, compared to Top-k which is proposed for reducing communication cost, FedMask obtains a larger saving in communication cost. Specifically, FedMask offers 3.33×, 3.35×, 3.33×, 3.36× savings in communication cost on IC-CIFAR10, IC-EMNIST, HAR, and NCP, respectively. More important, FedMask increases inference accuracy by 28.47%, 8.17%, 2.43%, and 8.09%, respectively.

Forth, compared to BNN-FedAvg which also transmits binary parameters between devices and the server, FedMask does not significantly outperform BNN-FedAvg on communication efficiency. However, FedMask considerably increases inference accuracy by 61.23%, 36.66%, 10.2%, and 33.47% compared to BNN-FedAvg, respectively.

As FedAvg is a general FL method, which is designed without any specific optimizations for communication and personalization, it is not surprise that FedMask significantly outperforms FedAvg in terms of both inference accuracy and communication cost. In addition, although Standalone does not pay for any communication cost, FedMask shows substantially better performance compared to Standalone due to exploiting all the local data rather than training the model using only limited training samples on each device.

High Computation Efficiency: Figure 11 illustrates the comparison of computation cost between FedMask and the baselines. Note that all the baselines consume the same computation cost when the same base model and training strategy is applied. As Figure 11 shows, due to the structured sparsity, FedMask is more computation efficient in all the applications compared to the baselines. Specifically, FedMask can save 2.42×, 2.53×, 3.02×, and 1.23× computation cost on IC-CIFAR10, IC-EMNIST, HAR, and NCP, respectively.

4.5 FL Hyper-Parameter Evaluation

Number of Participating Devices: We evaluate the impact of the number of participating devices in each communication round on the FedMask performance. We conduct experiments on IC-EMNIST, IC-CIFAR10, NCP, and vary the number of participating devices with {20, 40, 80}. As Figure 12 illustrates, the inference accuracy slightly increases with a larger number of participating devices in each communication round. For example, the inference accuracy on IC-CIFAR10 increases by 1.67% when the number of participating

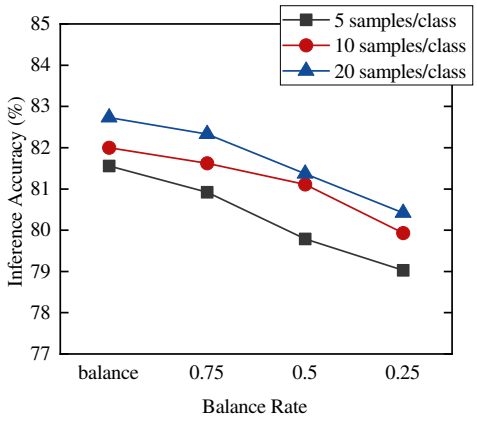

Figure 13: The impact of data volume and balance rate on FedMask performance (IC-CIFAR10).

devices increases from 20 to 80. However, such a larger number of participating devices consumes 4× communication cost, which diminishes the benefit of the slight accuracy improvement.

Data Volume and Unbalance Rate: The volume of Data on devices is a critical 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 FedMask 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 vary the volume of data with {5, 10, 20} 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 13 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 82.33% to 81.37% 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 0.72% when changing the data volume from 10 samples/class to 5 samples/class. The results demonstrate that FedMask can also effectively handle very challenging settings.

4.6 Runtime Performance

Reduction of Memory footprint: One key feature of FedMask is applying the binary mask to frozen weights to compose the structured sparse and personalized model for deployment. To quantify the benefit of sparsity on reducing memory footprint, we compare the model size between the model learned by FedMask and the ones trained using the baselines. Note that the models learned using the baselines are dense neural networks and share the same model size.

Table 5: Memory footprint reduction of FedMask.

Application	FedMask Model Size (MB)	Baseline Model Size (MB)	BNN-FedAvg Model Size (MB)
IC-CIFAR10	365.30	537.21	16.78
IC-EMNIST	364.72	538.09	16.82
HAR	2.69	4.41	0.14
NCP	0.92	1.53	0.05
All Included	733.63	1081.24	33.79

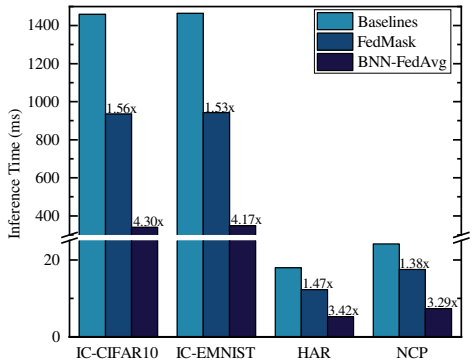

Figure 14: Comparison between FedMask and the baselines on inference time.

Table 5 lists comparison of the memory footprint across all the applications. It is obvious that BNN-FedAvg achieves the largest memory reduction, because it only consumes 1/32 memory usage compared to the other baseline methods, i.e., 1 bit for one parameter in BNN-FedAvg but 32 bits for one parameter in the other methods. Unfortunately, as presented in Figure 10, the accuracy of BNN-FedAvg is too low to be applicable for real-world applications. Compared to the other baselines, FedMask is able to significantly reduce the memory footprint. For example, FedMask can save 173.37MB on memory footprint in IC-MNIST, which is equivalent to 32.2% of the memory footprint by the baseline model. In addition, considering to deploy all the application applications concurrently on smartphones, FedMask achieves 347.61MB memory footprint reduction in total, showing the strong applicability of FedMask in practice.

Inference Speedup: The structured sparsity of the model learned using FedMask also introduces a significant speedup on inference. To quantify this benefit, we compare the average inference latency between the model learned using FedMask and the ones trained by the baselines. As Figure 14 illustrates, compared with the baselines, FedMask is able to achieve 1.56×, 1.53×, 1.47×, and 1.38× inference speedup on IC-CIFAR10, IC-EMNIST, HAR, and NCP respectively. Even though BNN-FedAvg shows a better improvement on inference efficiency, but it is useless in practice due to its very bad performance in terms of accuracy.

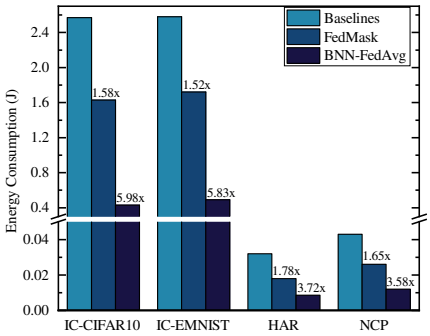

Figure 15: Comparison between FedMask and the baselines on energy consumption.

Reduction on Energy Consumption: Besides reducing memory footprint and inference latency, FedMask also saves more energy compared to the baselines. In this experiment, we compare the average energy consumption per inference across all the applications between FedMask and the baselines. As Figure 15 shows, FedMask is able to save 1.58×, 1.52×, 1.78×, and 1.65× energy consumption on IC-CIFAR10, IC-EMNIST, HAR, and NCP respectively. Again, BNN-FedAvg saves more energy consumption compared to FedMask, but the unsatisfactory performance in accuracy results in its uselessness for real-world applications.

5 DISCUSSION

Defending Against Privacy Leakage. Privacy preservation is the critical property offered by FL. Unfortunately, recent studies have demonstrated that FL is vulnerable to various inference attack, e.g., property inference attack [43] and model inversion attack [16, 17, 61], due to sharing model updates or gradients. Recent study [51] identify 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 FedMask, only local binary masks are communicated between devices and the central server, i.e., partial information of the local data has been hidden. Therefore, FedMask may be potentially helpful for defending against the privacy leakage in FL, and we plan to investigate the impact of FedMask on such defenses in the future work.

One Base Model for All. When a device needs to handle various AI-related tasks, one common practice is to train and deploy an individual model for each task. However, this will occupy a large amount of storage space especially for deep neural networks, e.g., VGG19 requires 558.4 MB to store [19], leading to dramatically decrease of available storage on device. However, we may be able to leverage the mask-based learning mechanism in FedMask to handle various tasks with only a single base model. In particular, base on the same base model, we can learn a specific mask for each task. We may need to slightly adjust the output layer of the base model upon the requirements of learning tasks, but most layers of the base model can be shared across tasks. Similar to FedMask, we can obtain task-specific model by elementwise applying the corresponding mask to the base model. In doing so, we only need to store the shared base model and multiple binary masks, but the storage cost of binary masks are significantly lower than storing deep neural networks. We will explore this promising direction as our future work.

6 RELATED WORK

Communication-Efficient Distributed Deep Learning. Communication cost is one the major bottlenecks of distributed deep learning [42, 59]. Many works focus on improving the communication efficiency by reducing the volume of transmitted data (i.e., gradient or weights). These prior arts can be divided into three classes: (1) *quantization* methods [2, 6, 10, 23, 29, 47, 56, 57] compress communication by reducing the number of bits of each element in the transferred data; (2) *sparsification* methods [1, 12, 26] transmit only a subset of elements in the communicated data; (3) *hybrid* methods [5, 27, 38, 50] combine quantization with sparsification. Our work also aims to reduce communication cost and our approach outperforms the state-of-the-art methods. Moreover, unlike existing data compression methods, our proposed approach only communicates binary masks (i.e., 1 bit per parameter) learned by devices.

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 [21]. 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* [13, 28, 30, 54], *multi-task learning* [49], and *contextualization* [41]. The local fine-tuning is the dominant approach for personalization, where each client tunes a global model using its own local data with several gradient descent steps. In contrast, FedMask achieves personalization using the heterogeneous binary mask in a single step without the requirements of local fine-tuning.

7 CONCLUSION

In this paper, we present the design, implementation and evaluation of FedMask, a FL framework that significantly improves communication and computation efficiency simultaneously. By applying FedMask, each device learns a personalized and sparse DNN model rather than a shared global model. Instead of directly optimizing model parameters, each device learns a heterogeneous binary mask while freezing the parameters of local model. Only these binary masks are communicated between devices and the central server. The aggregation strategy is carefully designed to independently process each element in the binary mask, such that the heterogeneity of binary masks can be preserved. We evaluate FedMask using three representative FL applications. Our results demonstrate that FedMask significantly outperforms the state-of-the-art methods in inference accuracy, communication cost, computation cost, memory footprint, inference latency, and energy consumption. We believe FedMask represents a significant step towards the realization of efficient FL into reality.

ACKNOWLEDGEMENT

We sincerely thank the anonymous reviewers and our shepherd for their valuable feedback. This work was partially supported by NSF-2112562, NSF-2140247, NSF-1617627, NSF-1814551, NSF-1632051, and Amazon Research Award.

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] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046* (2019).

[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] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* (2016).

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

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

[12] 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

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

[14] Biyi Fang, Jillian Co, and Mi Zhang. 2017. Deepasl: Enabling ubiquitous and non-intrusive word and sentence-level sign language translation. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 1–13.

[15] 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.

[16] 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*.

[17] 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*.

[18] 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).

[19] Jie Hang, Dexiang Zhang, Peng Chen, Jun Zhang, and Bing Wang. 2019. Classification of plant leaf diseases based on improved convolutional neural network. *Sensors* 19, 19 (2019), 4161.

[20] 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).

[21] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. 2020. FedML: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518* (2020).

[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[23] 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).

[24] Zehao Huang and Naiyan Wang. 2018. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*. 304–320.

[25] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. *Advances in neural information processing systems* 29 (2016), 4107–4115.

[26] 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).

[27] 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

[28] 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).

[29] 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).

[30] 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.

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

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

[33] 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).

[34] 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).

[35] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2019. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873* (2019).

[36] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. 2019. Exploiting kernel sparsity and entropy for interpretable CNN compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2800–2809.

[37] 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).

[38] 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).

[39] 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.

[40] 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).

[41] 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).

[42] 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.

[43] 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.

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

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

[46] 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.

[47] 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).

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

[49] 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.

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

[51] 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]

[52] 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).

[53] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H. Brendan McMahan, Blaise Aguera y Arcas, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, Suhas Diggavi, Hubert Eichner, Advait Gadhikar, Zachary Garrett, Antonious M. Girgis, Filip Hanzely, Andrew Hard, Chaoyang He, Samuel Horvath, Zhouyuan Huo, Alex Ingerman, Martin Jaggi, Tara Javidi, Peter Kairouz, Satyen Kale, Sai Praneeth Karimireddy, Jakub Konecny, Sanmi Koyejo, Tian Li, Luyang Liu, Mehryar Mohri, Hang Qi, Sashank J. Reddi, Peter Richtarik, Karan Singhal, Virginia Smith, Mahdi Soltanolkotabi, Weikang Song, Ananda Theertha Suresh, Sebastian U. Stich, Ameet Talwalkar, Hongyi Wang, Blake Woodworth, Shanshan Wu, Felix X. Yu, Honglin Yuan, Manzil Zaheer, Mi Zhang, Tong Zhang, Chunxiang Zheng, Chen Zhu, and Wennan Zhu. 2021. A Field Guide to Federated Optimization. arXiv:2107.06917 [cs.LG]

[54] 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).

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] Mi Zhang, Faen Zhang, Nicholas D Lane, Yuanchao Shu, Xiao Zeng, Biyi Fang, Shen Yan, and Hui Xu. 2020. Deep Learning in the Era of Edge Computing: Challenges and Opportunities. *Fog Computing: Theory and Practice* (2020), 67–78.

[60] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. 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.). 3597–3607.

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