FedMef: Towards Memory-efficient Federated Dynamic Pruning

Hong Huang[1] Weiming Zhuang[2] Chen Chen[2] Lingjuan Lyu[2*]
[1]City University of Hong Kong [2] Sony AI
hohuang@cityu.edu.hk {weiming.zhuang, chena.chen, lingjuan.lv}@sony.com

Abstract

Federated learning (FL) promotes decentralized training while prioritizing data confidentiality. However, its application on resource-constrained devices is challenging due to the high demand for computation and memory resources to train deep learning models. Neural network pruning techniques, such as dynamic pruning, could enhance model efficiency, but directly adopting them in FL still poses substantial challenges, including post-pruning performance degradation, high activation memory usage, etc. To address these challenges, we propose FedMef, a novel and memory-efficient federated dynamic pruning framework. FedMef comprises two key components. First, we introduce the budget-aware extrusion that maintains pruning efficiency while preserving post-pruning performance by salvaging crucial information from parameters marked for pruning within a given budget. Second, we propose scaled activation pruning to effectively reduce activation memory footprints, which is particularly beneficial for deploying FL to memory-limited devices. Extensive experiments demonstrate the effectiveness of our proposed FedMef. In particular, it achieves a significant reduction of 28.5% in memory footprint compared to state-of-the-art methods while obtaining superior accuracy.

1. Introduction

Federated learning (FL) has emerged as an important paradigm for the training of machine learning models across decentralized clients while preserving the confidentiality of local data [27, 31, 51]. In particular, cross-device FL, as described in [21], places emphasis on scenarios where FL clients predominantly consist of edge devices with resource constraints. Cross-device FL has gained significant attention in academic research and industry applications, fueling a wide range of applications, including Google Keyboard [15, 25], Apple Speech Recognition [38], etc. Despite its success, the resource-intensive nature of training models, which includes high computational and memory costs, poses challenges for the deployment of cross-device FL on resource-constrained devices.

Neural network pruning [14, 19, 33, 43] is a potential solution to improve the efficiency of the model and reduce the high demand for resources. However, a closer inspection of some previous work on the application of neural network pruning to FL [26, 29, 37, 41] reveals a potential pitfall: They often rely on initial training of dense models, similar to centralized pruning methodologies [14, 33, 43]. These federated pruning methods are not suitable for cross-device FL because the training of dense models still requires high computation and memory costs on resource-constrained devices.

To address these challenges, recent research has shifted to federated dynamic pruning [3, 17, 20, 39]. These frameworks derive specialized pruned models by iterative adjustment of sparse on-device models. Devices start with a randomly pruned model, followed by traditional FL training, and periodically adjust the sparse model structure through pruning and growing operations [11]. Through iterative training and adjustments, devices can develop specialized pruned models bypassing the need to train dense models, which reduces both computational and memory demands.

However, existing federated dynamic pruning frameworks [3, 17, 20, 39] face two issues: significant post-pruning accuracy degradation and substantial activation memory usage. First, these frameworks cause a significant decline in accuracy after magnitude pruning because they hastily eliminate low-magnitude parameters, regardless of the substantial information they may contain. Such incautious parameter pruning often results in the model's inability to regain its previous accuracy before the subsequent pruning iteration, ultimately leading to suboptimal end-of-training performance. Second, these frameworks fail to reduce the memory footprint of activation. For certain widely adopted models for edge deployment, like MobileNet [40], a significant portion of the total memory footprints is allocated to activation. However, current federated dynamic pruning methods focus primarily on reducing the model size, overlooking optimization for activation memory.

*corresponding author

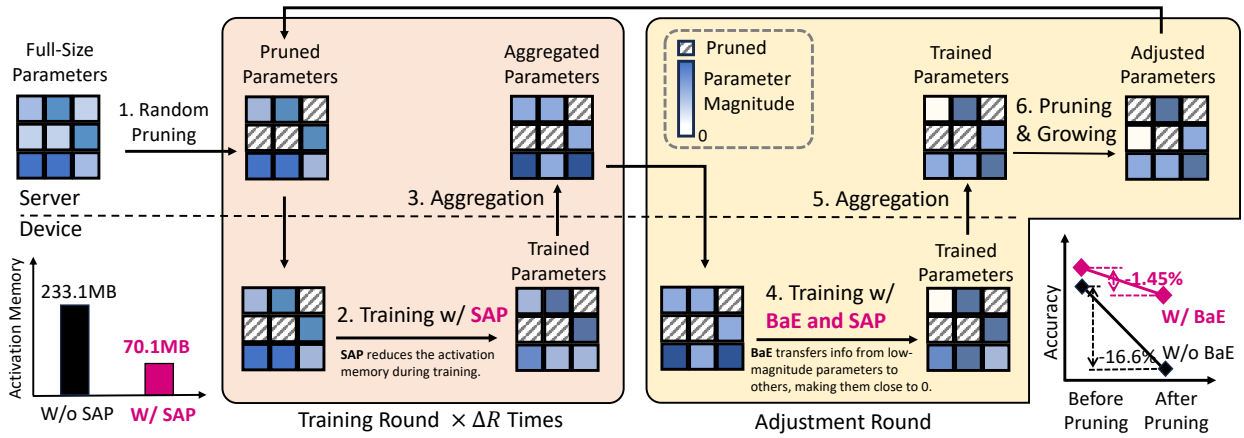

Figure 1. Overview of FedMef for the memory-efficient dynamic pruning in federated learning. FedMef proposes budget-aware extrusion (BaE) to preserve post-pruning accuracy by transferring essential information from low-magnitude parameters to the others, making them close to 0, and introducing scaled activation pruning (SAP) to reduce memory usage. In FedMef, the server distributes a randomly pruned model to devices for collaborative training with SAP. After multiple training rounds, devices employ BaE for information transfer. The server adjusts the model structure through magnitude pruning and growing. The newly activated parameters are initialized as 0.

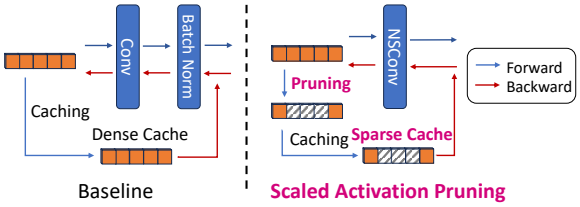

Figure 2. The illustration of training pipeline in baseline and the proposed scaled activation pruning method. During the forward pass, the scaled activation pruning generates near-zero activation via the Normalized Sparse Convolution (NSConv). Then, the dense activation caches are pruned based on magnitude. During the backward pass, these pruned caches are used to compute the gradients. Scaled activation pruning significantly saves activation memory footprints by more than 3 times in the CIFAR-10 dataset with the MobileNetV2 model.

In this work, we introduce FedMef, an **Fed**erated **M**emory-**ef**fcient dynamic pruning framework that adeptly addresses all the aforementioned challenges. Figure 1 illustrates the workflow of FedMef and highlights our two new proposed components. First, FedMef presents ***budget-aware extrusion*** **(BaE)** to address the challenge of post-pruning accuracy degradation. Rather than simply discarding low-magnitude parameters, our method salvages essential information from these potential pruning candidates by transferring it to other parameters through a surrogate loss function within a preset budget. Second, FedMef proposes ***scaled activation pruning*** **(SAP)** to address the problem of high activation memory. This method performs activation pruning during the training process to dramatically reduce the memory footprints of the activation caches, as illustrated in Figure 2. To enhance the efficacy of SAP, especially for devices with severe memory constraints, we are inspired by recent methods that remove batch normalization (BN) layers [4, 46] and replace convolution layers with Normalized Sparse Convolution (NSConv). NSConv can normalize most of the activation elements to be or close to zero. This reduces the disparity between original and sparse activation, mitigating the degradation of accuracy in SAP.

We conducted extensive experiments on three datasets: CIFAR-10 [22], CINIC-10 [9], and TinyImageNet [23], using ResNet18 [16] and MobileNetV2 [40] models. Extensive experimental results suggest that FedMef outperforms the state-of-the-art (SOTA) methods on all datasets and models. In addition, FedMef requires fewer memory footprints than SOTA methods. For example, FedMef significantly reduces the memory footprint of MobileNetV2 by 28.5% while improving the accuracy by more than 2%.

2. Related Work

2.1. Neural Network Pruning

Neural network pruning, which emerged in the late 1980s, aims to reduce redundant parameters in deep neural networks (DNNs). Traditional techniques focus on achieving a trade-off between accuracy and sparsity during inference. This typically involves assessing the importance of parameters in a pre-trained DNN and discarding those with lower scores. Various methods are employed to determine these scores, such as weight magnitude [14, 19] and Taylor expansion of the loss functions [24, 34, 36]. However, these methods need to train a dense model first, which increases both computational and memory costs.

Modern pruning has shifted its focus towards enhancing the efficiency of DNN training processes. For example, dynamic sparse training [10, 11, 32], actively adjusts the architecture of the pruned model throughout the train-

ing while maintaining desired sparsity levels. Nevertheless, these methods only simply prune low-magnitude parameters while neglecting the memory consumption of the activation caches, resulting in decreased accuracy and suboptimal memory optimization.

2.2. Federated Neural Network Pruning

Federated learning has recently emerged as a promising technique to navigate data privacy challenges in collaborative machine learning [31, 48–50]. However, numerous previous federated pruning efforts [26, 29, 37, 41] have encountered setbacks because they rely on the training of dense models on devices, which require high computation and memory. Thus, they are not suitable for cross-device FL [21], where clients are devices with resource constraints.

Recent studies [3, 17, 20, 39] introduce on-device pruning via the dynamic sparse training technique [10, 11, 32]. For example, ZeroFL [39] divides the weights into active and non-active weights for inference and sparsified weights and activation for backward propagation. FedDST [3] and FedTiny [17], inspired by RigL [11], perform pruning and growing on devices, with the server generating a new global model through sparse aggregation. However, these methods are unable to reduce the memory footprints of the activation cache and suffer from significant accuracy degradation after pruning, since they directly prune parameters that may contain important information.

Therefore, all existing federated neural network pruning works fail in creating a specialized pruned model that can concurrently satisfy both accuracy and memory constraints. To the best of our knowledge, FedMef is the first work that can simultaneously address both issues.

2.3. Activation Cache Compression

High-resolution activation tensors are a primary memory burden for modern deep neural networks. Gradient checkpoint [8, 12, 13], which stores specific layer tensors and recalculates others during the backward pass, offers a memory-saving solution, but at a high computational cost. Alternatively, adaptive precision quantization methods [5, 30, 44] compress activation caches through quantization but introduce time overhead from dynamic bit-width adjustments and dequantization. The activation pruning (sparsification) method [7], which sparsifies the activation caches, is lighter than other methods but relies heavily on batch normalization (BN) layers to guarantee that most of the elements in activation are zero or near zero. Relying on BN layers would be problematic to train with small batches and non-independent and identically distributed (non-i.i.d.) data [28, 46, 47]. As a result, current activation pruning methods are unsuitable for resource-constrained devices in FL. To address these challenges, our proposed FedMef utilizes scaled activation pruning, effectively compressing activation caches without relying on BN layers.

3. Methodology

This section first introduces the problem setup and then outlines the design principles of our proposed FedMef. We then introduce two key components in FedMef: budget-aware extrusion and scaled activation pruning.

3.1. Problem Setup

In the cross-device FL scenario, numerous resource-constrained devices collaboratively train better models without direct data sharing [21]. In this setting, K devices, each with memory and computational constraints, cooperate to train the model with parameters θ. Every device possesses a distinct local dataset, denoted as $\mathcal{D}_k, k \in \{1, 2, \ldots, K\}$. The structure of the pruned model is represented using a mask, $m \in \{0, 1\}^{|\theta|}$, and $\theta \odot m$ denotes the sparse parameters of the pruned model. Our objective is to derive a specialized sparse model with mask m, using the local dataset $\mathcal{D}_k$, to optimize prediction accuracy in FL. During training, the sparsity levels of the mask s_m and the activation caches s_a must be higher than the target sparsity (s_{tm} and s_{ta}), which is determined by the memory constraints of the devices. Thus, our optimization challenge is to solve the following problem:

$$\begin{aligned} \min_{\boldsymbol{\theta}, m} L(\theta, m) &:= \sum_{k=1}^{K} p_k L_k(\theta, m, \mathcal{D}_k), \\ \text{s.t.} \quad & s_m \geq s_{tm}, s_a \geq s_{ta}, \end{aligned} \tag{1}$$

where L_k is the loss function of the k-th device (*e.g.*, cross-entropy loss), and p_k represents the weight of k-th device during model aggregation in the server. Before communicating with the server, each device trains its local model for E local epochs.

3.2. Design Principles

To ascertain that specialized sparse models can be developed on resource-constrained devices while maintaining privacy, the prevailing trend is to leverage federated dynamic pruning. However, contemporary methods [3, 17, 20, 39] face two pressing issues: significant post-pruning accuracy degradation and high activation memory usage. As illustrated in Figure 1, our framework, FedMef, introduces two solutions to these challenges: budget-aware extrusion and scaled activation pruning.

In the FedMef framework, the server starts by distributing a randomly pruned model to the devices. Subsequently, these devices collaboratively engage in training sparse models using scaled activation pruning. During this phase, the activation cache is pruned during the forward pass, effectively optimizing memory utilization, as illustrated in Figure 2. After several iterative training rounds, the devices

employ the budget-aware extrusion technique to transfer vital information from low-magnitude parameters to others. Subsequently, the server adjusts the model structure through magnitude pruning and gradient-magnitude-based growing. Due to the information transfer facilitated by budget-aware extrusion, the post-pruning accuracy degradation is slight. Finally, the framework continues with the training and adjustment of the sparse model until convergence.

Specifically, budget-aware extrusion achieves information transfer by employing a surrogate loss function with L_1 regularization of low-magnitude parameters. This process not only suppresses their magnitude but also transfers their information to alternate parameters. Additionally, the devices set up a budget-aware schedule to speed up the extrusion. In the scaled activation pruning, after each layer's forward pass, the activation caches are pruned to reduce memory usage. During the backward pass, the sparse activation caches are used directly. To ensure that the pruned elements are zero or nearly zero, even when training with small batch size, we adopt the Normalized Sparse Convolution (NSConv) to reparameterize the convolution layers instead of using batch normalization layers. Next, we delve into in-depth discussions of budget-aware extrusion and scaled activation pruning techniques.

3.3. Budget-aware Extrusion

It is essential to address the information loss that occurs during pruning, as the parameters to be pruned often retain valuable information. Direct pruning can cause a substantial accuracy drop, demanding considerable resources for recovery, as illustrated in Figure 1. This issue may become even more pronounced in federated contexts due to the heterogeneous data distribution across devices, potentially amplifying the negative impact on model performance during training.

To address this challenge, we take inspiration from the Dual Lottery Ticket Hypothesis (DLTH) [2]. The DLTH suggests that a randomly selected subnetwork can be transformed into one that achieves better, or at least comparable, performance to benchmarks. Building on this premise, we introduce budget-aware extrusion within our FedMef framework, which can extrude the information from the parameters to be pruned to other surviving parameters. After sufficient extrusion, the parameters designated for pruning retain only marginal influence on the network, ensuring minimal information and accuracy loss during pruning.

In alignment with the findings of the DLTH [2], we employ a regularization term to execute this information extrusion. Given the parameters θ and its associated mask m, the extrusion process on the k-th device can be realized through the surrogate loss function L_k^s:

$$L_k^s = L_k(\theta, m, \mathcal{D}_k) + \lambda||\theta_{low}||_2^2, \tag{2}$$

where λ is constant and θ_{low} represents the parameters earmarked for pruning, which is the subset of unpruned parameters $\theta \odot m$ with the lowest weight magnitudes.

The inherent constraints associated with edge device training resources require that information extrusion should be executed within a limited budget before the pruning process. However, adhering to the original learning schedule represented by η is sub-optimal, as in the later epochs of training, the learning rate following traditional decay mechanisms becomes significantly small, impeding the information extrusion process. To address this issue, we introduce a budget-aware schedule in the context of budget-aware extrusion. Our proposed budget-aware schedule is constructed to accelerate the extrusion process, especially when the original learning rate is insufficient for rapid extrusion. We have adopted one of the SOTA budgeted schedules, the REX schedule [6], whose learning rate in the t-th step is represented as $\beta_t = p(t)\eta_0$, where η_0 represents the initial learning rate in the original learning schedule and $p(t)$ is the REX schedule factor. However, the REX schedule does not take into account the status of the extrusion process, often resulting in an excessively high learning rate. Therefore, we introduce a scaling term that ranges from 0 to 1, $2\sigma(||\theta_{low}||) - 1$ to moderate the budgeted learning rate based on the extrusion status, which is represented by the magnitude of the marked parameters. Given T_{budget} as the training budget, our proposed budget-aware learning rate β_t at t-th step is mathematically defined as:

$$\beta_t = p(t)(2\sigma(||\theta_{low}||) - 1)\eta_0, \tag{3}$$

where the REX schedule factor $p(t)$ is defined by $p(t) = \frac{2T_{budget}-2t}{2T_{budget}-t}$. The main objective behind introducing this factor is to effectively adjust the learning rate based on the relative progression of the training and the preset training budget. During the information extrusion process, the learning rate μ_t is formulated as follows:

$$\mu_t = \max(\eta_t, \beta_t), \tag{4}$$

where η_t is the learning rate in the original schedule. The budget-aware schedule ensures efficient and timely information extrusion by adjusting an adequate learning rate even in the later stages of training. During the normal training stage, the learning rate is $\mu_t = \eta_t$.

In particular, upon receiving the pruned model from the server, the devices mark the parameters θ_{low} that have the lowest weight magnitude. Then, the devices perform several epochs of budget-aware extrusion with the surrogate loss L_k^s in Equation 2. The learning rate for this phase is dynamic and is governed by the function presented in Equation 4. After the extrusion phase, each device calculates the Top-K gradients across all pruned parameters and uploads the gradients along with the parameters to the server,

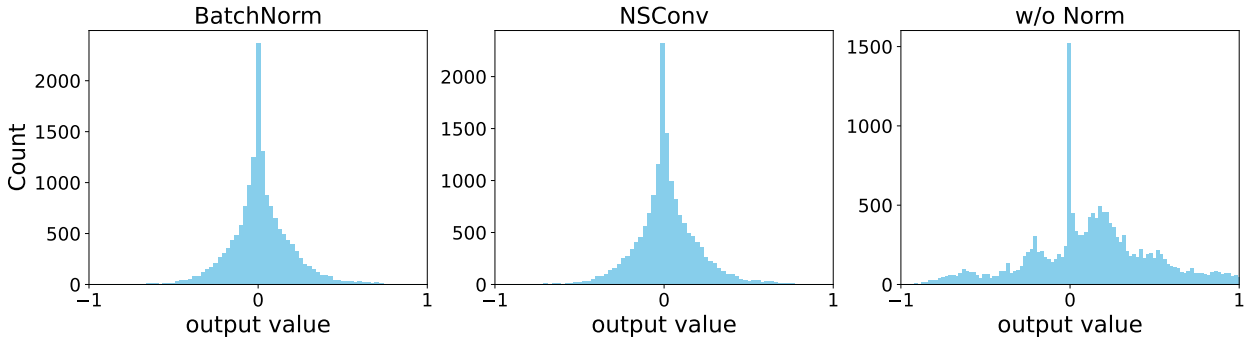

Figure 3. Distribution of output from a convolution layer in ResNet18 using batch normalization layers (BatchNorm), without normalization layers (w/o Norm), and with our proposed Normalized Sparse Convolution (NSConv). The output experiences an internal covariate shift when training without normalization layers, whereas NSConv effectively mitigates this issue. Figure 8 in the appendix shows the output distribution for all convolution layers in the ResNet18 model.

as indicated in [17]. The server then aggregates the sparse parameters and gradients to obtain the average parameters and average gradients. Finally, the server prunes the marked parameters θ_{low} and grows the same number of parameters with the largest averaged gradient magnitude.

According to the pruning and growing process, the server creates a global model with a new structure and then FedMef begins to train the new global model. FedMef periodically performs adjustments and training to deliver an optimal sparse neural network suitable for all devices. The pseudo code can be viewed in Algorithm 1 in the appendix.

3.4. Scaled Activation Pruning

In cross-device FL, where devices may have extremely limited memory, small batch sizes are often employed during training. This diminishes the effectiveness of batch normalization (BN) layers in such a scenario. However, current activation cache compression techniques, such as DropIT [7], are limited in their ability to conduct training without BN layers. To address this issue, we propose a scaled activation pruning technique that achieves superior performance even with small batch sizes, as illustrated in Figure 2.

Given a CNN model with ReLU-Conv ordering, in the l-th convolution layer, the sparse filters are represented as $\theta^l \in \mathrm{R}^{ks \times ks \times c_{in} \times c_{out}}$, where ks denotes the kernel size; c_{in} and c_{out} denote the number of input and output channels, respectively. For an input value a^{l-1}, the convolution operation in the l-th layer that yields the output value a^l is:

$$a^l = \mathrm{Conv}(\theta^l, f(a^{l-1})), \tag{5}$$

where $f(\cdot)$ is any activation function such as ReLU [1]. Note that a^{l-1} is not only an input of l layer but also the output of the $l-1$-th layer. During the forward pass, the activation $f(a^{l-1})$ must be retained in memory to compute the gradients of the filters θ^l during the backward pass. Similarly, for each layer, the activation $f(a^l)$ must be stored for later usage, which causes substantial memory footprints.

The activation pruning approach, DropIT [7], prunes $f(a^l)$ in the forward pass. It then uses sparse activation $S(f(a^l))$ for gradient computation in the backward pass. This approach requires that the input a^l be centered around zero. This centering ensures a minimized disparity between the sparse activation $S(f(a^l))$ and its original counterpart $f(a^l)$. However, this zero-centered requirement becomes unattainable when the efficacy of the batch normalization layer decreases. This ineffectiveness arises from internal covariate shift issues [4, 18]. Figure 3 shows the mean shift in activation within a ResNet18 model without a normalization layer, resulting in a non-zero mean in the activation distribution. The mathematical details of this effect can be found in Appendix 6.1.

To reduce the disparity between the original and pruned activation, inspired by the recent methods that remove BN layers [4, 46], we introduce Normalized Sparse Convolution (NSConv) into activation pruning. Our primary objective is to ensure that the output of the convolution layer is consistently centered around zero, *i.e.,* the mean value is zero. The convolution operation of NSConv at the l-th layer is given by:

$$a^l = \mathrm{Conv}(\hat{\theta}^l, f(a^{l-1})), \tag{6}$$

where $\hat{\theta}^l$ represents the sparse normalized filters with filter-wise weight standardization. The filter-wise standardization formula of the i-th sparse filter, denoted by $\hat{\theta}^l_i \in \mathbb{R}^{ks \times ks \times c_{in}}$, is given by:

$$\hat{\theta}^l_i = \gamma \sqrt{c_{in}} \frac{\theta^l_i \odot m^l_i - \mu_\theta}{\sigma_\theta}, \tag{7}$$

where $\theta^l_i \in \mathbb{R}^{ks \times ks \times c_{in}}$ specifies the i-th filter of the original filters, γ is a constant, and m^l_i denotes the corresponding mask for the sparse filter θ^l_i. The terms μ_θ and σ_θ represent the mean and standardization value of the sparse filter θ^l_i, excluding the pruned parameters whose corresponding mask is 0.

Theorem 1 *Given a CNN model structured in a ReLU-Conv sequence and l-th convolution layer performing operations as depicted by the forward pass in Equation 6 and NSConv in Equation 7, for the i-th channel of the activation value, $f(a_i^{l-1})$, with its mean and variance denoted as μ_f, σ_f^2, the mean and variance for the i-th channel of the output value, a_i^l, will be:*

$$\mathbb{E}[a_i^l] = 0, \quad \text{Var}[a_i^l] = \gamma^2(\sigma_f^2 + \mu_f^2). \tag{8}$$

Theorem 1 reveals insights into the capabilities of scaled activation pruning. Specifically, it highlights its efficacy in addressing the disparity between sparse and original activation in CNNs without BN layers. A key factor in its effectiveness is NSConv's ability to normalize the output of each convolution layer, centering it around zero, as shown in Figure 3. By adjusting the hyperparameter γ, we can control the variance of the distribution, causing a large portion of the activation elements to be either zero or close to it. The proof of Theorem 1 can be found in Appendix 6.2.

Incorporating NSConv into scaled activation pruning brings several additional advantages: First, NSConv ignores pruned parameters, focusing solely on the remaining ones. This translates to minimal computational overhead and maintains the sparsity of the normalized parameters. Second, NSConv is suitable for training with small batch sizes because there are no inter-dependencies between batch elements. Lastly, NSConv ensures uniformity between the training and testing phases.

4. Evaluation

In this section, we dive into an in-depth evaluation of our framework, FedMef. We compare it against SOTA frameworks, demonstrating its effectiveness in various testing conditions. In addition, an ablation study reveals the components that make our proposed framework effective.

4.1. Experimental Setup

We assess the effectiveness of FedMef in image recognition tasks using three datasets: CIFAR-10 [22], CINIC-10 [9], and TinyImageNet [23]. Notably, the choice of these datasets is motivated by the imperative of ensuring a fair comparison, given that existing federated dynamic pruning frameworks [3, 17, 20, 39] focus on simple datasets. We employ the ResNet18 [16] and MobileNetV2 [40] models for evaluation. We conduct experiments on a landscape of 100 devices. The datasets are divided into heterogeneous partitions via a Dirichlet distribution characterized by a factor of $\alpha = 0.5$. We train the models for $R = 500$ federated learning rounds, where each round is composed of $E = 10$ local epochs. We set the training batch size as 64 by default. The target parameters sparsity and target activation sparsity are set to $s_{tm} = 0.9$, $s_{ta} = 0.9$ by default. The initial learning rate is set as $\eta_0 = 1$ with an exponential decay rate of 0.95. We conduct each experiment five times and report the average result and standard deviation.

We compare our proposed FedMef with FL-PQSU [45], FedDST [3], and FedTiny [17]. FL-PQSU is a static pruning method, which employs an initialized pruning based on the L_1 norm on the server prior to training. It can be considered as the lower bound of our method. FedDST and FedTiny are SOTA federated dynamic pruning frameworks. Both of them start with an initial pruned model, subsequently employing mask adjustments to adjust the model architecture. The key distinction among these methods lies in their locus of model structure adjustments. FedTiny centralizes this on the server, whereas FedDST decentralizes it to the devices. Certain federated pruning frameworks, such as ZeroFL [39] and PruneFL [20], which are memory-intensive to process dense models, are consciously excluded from our comparison.

In the FedDST, FedTiny, and FedMef frameworks, the adjustment of the model structure is applied after $\Delta R = 10$ training rounds. Upon reaching $R_{stop} = 300$ rounds, the framework suspends further adjustment, continuing its training until reaching $R = 500$ rounds. The pruning number for each layer is set to $0.2(1 + cos\frac{t\pi}{R_{stop}E})n$ in the t -th iteration, where n is the number of unpruned parameters in the l-th layer. Due to FedDST [3] necessitating a series of on-device training epochs for fine-tuning after adjustment, after 5 epochs of local training, we let FedDST adjust the model structure and then proceed with 5 training epochs. FedTiny's [17] adaptive batch normalization module is amputated from our experiments, as its memory overhead renders it infeasible for our device constraints.

4.2. Performance Evaluation

To demonstrate the effectiveness of FedMef, we compare it with other frameworks on the CIFAR-10, CINIC-10, and TinyImageNet datasets using ResNet18 and MobileNetV2. A holistic comparison is illustrated in Figure 4. The target sparsity of the parameters is set to $s_{tm} \in \{0.95, 0.9, 0.8\}$. Remarkably, FedMef outperforms all baseline frameworks in terms of accuracy and memory efficiency. For instance, FedMef achieves an accuracy improvement of 2.13% on the CIFAR-10 dataset with the MobileNetV2 model, while saving 28.5% memory usage compared to the best baseline framework, FedTiny. Such advances in accuracy can be attributed to budget-aware extrusion, while scaled activation pruning primarily augments memory conservation.

An obvious trend is the superior accuracy benchmarks set by ResNet18 over MobileNetV2 in all datasets. The design of MobileNetV2 is tailored to large-scale image classification, which may make it less suitable for relatively small datasets. A noteworthy observation is that FedTiny gener-

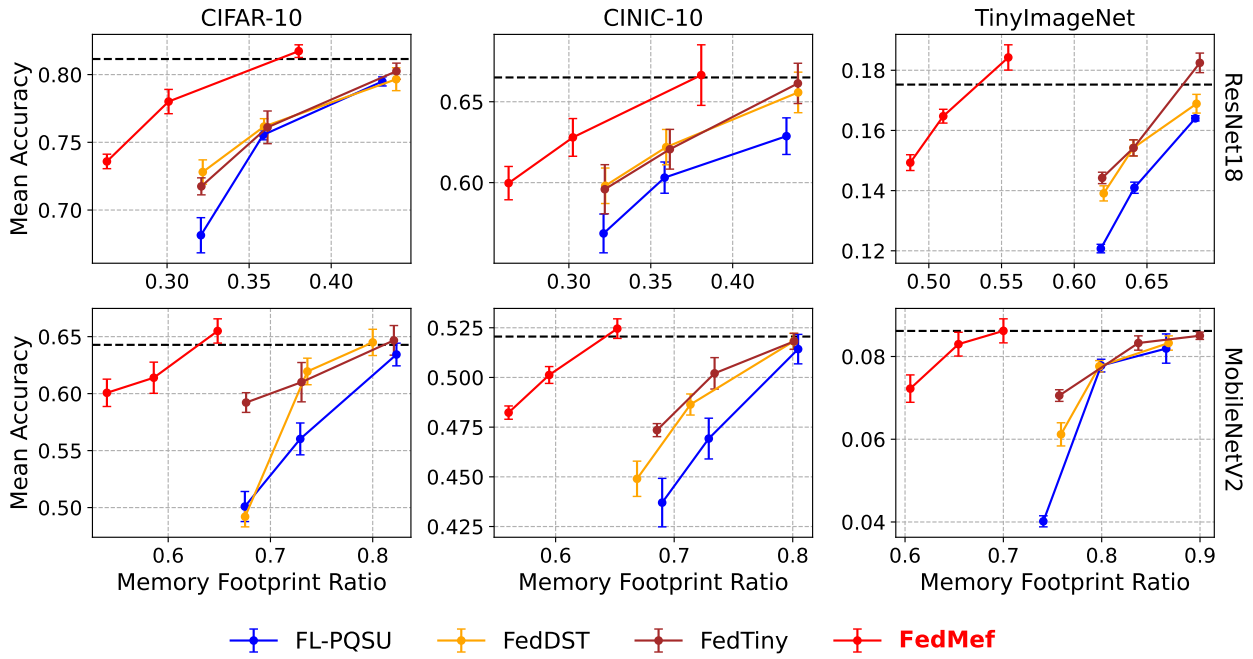

Figure 4. Comparison of accuracy and memory footprint of our proposed FedMef with the existing federated pruning methods on three datasets. The black dashed line marks the accuracy of training a full-size model (without pruning) in FedAvg. The memory footprint ratio is the memory footprint relative to training a full-size model in FedAvg.

ally outperforms other baseline methods within comparable memory footprints. Given this empirical trend, ResNet18 is chosen as the default model and FedTiny serves as the primary reference for subsequent experiments.

Computational and Communication Cost. While FedMef exhibits superior performance compared to various baseline frameworks with relatively low memory footprints, it is imperative to conduct a comprehensive analysis of the computational and communication costs inherent in the FedMef framework. To illustrate the computational and communication costs, we evaluated FedMef on the CIFAR-10 dataset using the ResNet18 model. The results, presented in Table 1, elucidate that FedMef induces marginal communication and computational overhead. For example, with a target density of $s_{tm} = 90\%$, FedMef introduces only a mere computational overhead of $0.003\times$ and a communication overhead of $0.005\times$, while improving the accuracy by 2% compared to other baselines. The detailed analysis for the training cost is shown in Appendix 7.

Training with Small Batch Size. Under strict memory constraints, training requires a smaller batch size. However, this compromises statistical robustness and often hinders the effectiveness of batch normalization. To address this issue, we propose scaled activation pruning. The evaluations conducted on the CIFAR-10 dataset with ResNet18, where the batch size is set to 1, as shown in Figure 5 (left), demonstrate that FedMef outperforms all baseline methodologies.

s_{tm}	Framework	Mean Accuracy	Training FLOPs	Data Exchange
0	FedAVG	81.2%	1× (8.33e12)	1× (89.52MB)
95%	FedDST	72.8%	0.057×	0.083×
	FedTiny	71.8%	0.057×	0.086×
	FedMef	**73.6%**	0.061×	0.086×
90%	FedDST	76.2%	0.113×	0.133×
	FedTiny	76.1%	0.113×	0.138×
	FedMef	**78.1%**	0.116×	0.138×
80%	FedDST	79.7%	0.218×	0.232×
	FedTiny	80.3%	0.218×	0.243×
	FedMef	**81.7%**	0.221×	0.243×

Table 1. Accuracy and training cost of proposed FedMef and other baseline framework. We report the maximum FLOPs for training (Training FLOPs) and the maximum data exchange for communication (Data Exchange). All cost measurements are for one device in one adjustment round.

The significant improvement in accuracy is mainly due to the use of NSConv in scaled activation pruning, which is further demonstrated in the appendix 6.3.

The Impact of Adjustment Period. After model structure adjustment, it is necessary to restore accuracy loss through several training rounds. Therefore, the adjustment period should be longer. Unfortunately, given the computational constraints of certain devices, there is an urgent need

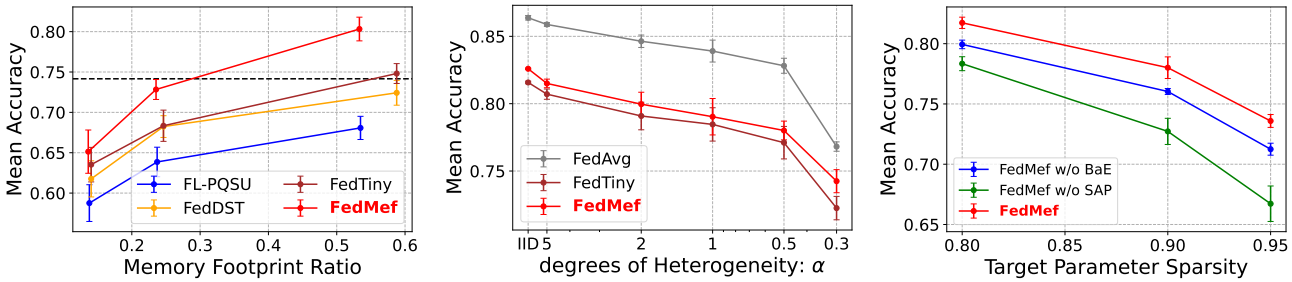

Figure 5. FedMef's average accuracy and standard deviation are compared against: *(left)* various federated pruning frameworks when the training batch size is 1, where the black dashed line represents the accuracy of FedAvg framework; *(middle)* FedAvg and FedTiny across varying degrees of data heterogeneity; *(right)* modified versions of FedMef - one excluding BaE (similar to FedTiny's approach) and the other without SAP (omitting NSConv).

ΔR	FedTiny	FedMef
3	55.09%(1.82%)	61.94%(0.49%)
5	58.73%(1.62%)	62.12%(0.58%)
10	61.18%(1.08%)	62.77%(0.78%)

Table 2. Mean accuracy (standard deviation) for FedMef and FedTiny on CIFAR-10 with various adjustment periods, ΔR.

to limit the number of interval training rounds and local epochs. The empirical results of the experiments on the CIFAR-10 dataset with various adjustment periods, ΔR, and a single local epoch are shown in Table 2. When FedTiny performance decreases under resource constraints, FedMef remarkably maintains the performance.

Analysis on Convergence Behavior. Given FedMef's capability to preserve post-pruning performance through budget-aware extrusion, we anticipate that its convergence speed will surpass that of other baseline frameworks, particularly with a smaller adjustment period. To assess this, we evaluate the performance of FedMef and other baseline frameworks in the CIFAR-10 datasets with the adjustment period (ΔR) set to 5. The results illustrate that the convergence trajectory of FedMef is notably more stable than other baselines, reaching a higher final accuracy, as shown in Figure 6. This observation underscores the efficacy of BaE in enhancing the convergence behavior of FedMef.

Analysis on Different Degrees of Data Heterogeneity. We test the effectiveness of FedMef on heterogeneous data distributions by modulating the Dirichlet distribution factor α, where the lower α indicates a higher degree of heterogeneity. For reference, we compare our results with the full-size model and FedTiny on the CIFAR-10 dataset and the results are shown in Figure 5 (middle). FedMef retains its superior performance compared to the SOTA framework.

4.3. Ablation Study

We further analyze the individual contributions of budget-aware extrusion and scaled activation pruning using trials on the CIFAR-10 dataset with ResNet18. The variants include a FedMef without budget-aware extrusion (akin to FedTiny's mechanism) and a FedMef without scaled activation pruning (mirroring DropIT's approach [7] without NSConv). The findings presented in Figure 5 (right) indicate that both budget-aware extrusion and scaled activation pruning boost FedMef's performance. In particular, removing scaling in activation pruning results in substantial information loss during backpropagation and leads to performance degradation.

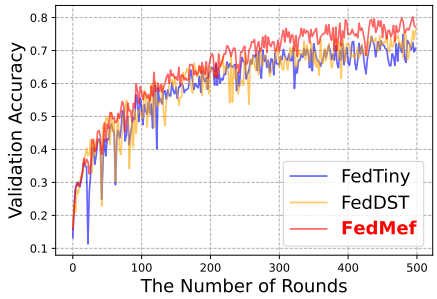

Figure 6. The validation accuracy during the training in FedMef and the baseline frameworks.

5. Conclusion

This paper introduces FedMef, a memory-efficient federated dynamic pruning framework designed to generate specialized models on resource-constrained devices in cross-device FL. FedMef addresses the issues of post-pruning accuracy degradation and high activation memory usage that current federated pruning methods suffer from. It proposes two new components: budget-aware extrusion and scaled activation pruning. Budget-aware extrusion reduces information loss in pruning by extruding information from parameters marked for pruning to other parameters within a limited budget. Scaled activation pruning allows activation caches to be pruned to save more memory footprints without compromising accuracy. Experimental results demonstrate that FedMef outperforms existing approaches in terms of both accuracy and memory footprint.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 5

[2] Yue Bai, Huan Wang, Zhiqiang Tao, Kunpeng Li, and Yun Fu. Dual lottery ticket hypothesis. *arXiv preprint arXiv:2203.04248*, 2022. 4

[3] Sameer Bibikar, Haris Vikalo, Zhangyang Wang, and Xiaohan Chen. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6080–6088, 2022. 1, 3, 6

[4] Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. *arXiv preprint arXiv:2101.08692*, 2021. 2, 5

[5] Jianfei Chen, Lianmin Zheng, Zhewei Yao, Dequan Wang, Ion Stoica, Michael Mahoney, and Joseph Gonzalez. Actnn: Reducing training memory footprint via 2-bit activation compressed training. In *International Conference on Machine Learning*, pages 1803–1813. PMLR, 2021. 3

[6] John Chen, Cameron Wolfe, and Tasos Kyrillidis. Rex: Revisiting budgeted training with an improved schedule. *Proceedings of Machine Learning and Systems*, 4:64–76, 2022. 4

[7] Joya Chen, Kai Xu, Yuhui Wang, Yifei Cheng, and Angela Yao. Dropit: Dropping intermediate tensors for memory-efficient dnn training. *arXiv preprint arXiv:2202.13808*, 2022. 3, 5, 8

[8] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 3

[9] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018. 2, 6

[10] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019. 2, 3, 6

[11] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020. 1, 2, 3, 6

[12] Jianwei Feng and Dong Huang. Optimal gradient checkpoint search for arbitrary computation graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11433–11442, 2021. 3

[13] Audrunas Gruslys, Rémi Munos, Ivo Danihelka, Marc Lanctot, and Alex Graves. Memory-efficient backpropagation through time. *Advances in neural information processing systems*, 29, 2016. 3

[14] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1, 2

[15] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018. 1

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 6

[17] Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. Fedtiny: Pruned federated learning towards specialized tiny models. *arXiv preprint arXiv:2212.01977*, 2022. 1, 3, 5, 6

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 5

[19] Steven A Janowsky. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600, 1989. 1, 2

[20] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 1, 3, 6

[21] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021. 1, 3

[22] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009. 2, 6

[23] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 2, 6

[24] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989. 2

[25] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. Federated learning for keyword spotting. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6341–6345. IEEE, 2019. 1

[26] Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li. Lotteryfl: Empower edge intelligence with personalized and communication-efficient federated learning. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 68–79. IEEE, 2021. 1, 3

[27] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019. 1

[28] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021. 3

[29] Shengli Liu, Guanding Yu, Rui Yin, and Jiantao Yuan. Adaptive network pruning for wireless federated learning. *IEEE Wireless Communications Letters*, 10(7):1572–1576, 2021. 1, 3

[30] Xiaoxuan Liu, Lianmin Zheng, Dequan Wang, Yukuo Cen, Weize Chen, Xu Han, Jianfei Chen, Zhiyuan Liu, Jie Tang,

Joey Gonzalez, et al. Gact: Activation compressed training for generic network architectures. In *International Conference on Machine Learning*, pages 14139–14152. PMLR, 2022. 3

[31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 1, 3

[32] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018. 2, 3, 6

[33] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019. 1

[34] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2019. 2

[35] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pages 4646–4655. PMLR, 2019. 6

[36] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*, 1, 1988. 2

[37] Muhammad Tahir Munir, Muhammad Mustansar Saeed, Mahad Ali, Zafar Ayyub Qazi, and Ihsan Ayyub Qazi. Fedprune: Towards inclusive federated learning. *arXiv preprint arXiv:2110.14205*, 2021. 1, 3

[38] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, et al. Federated evaluation and tuning for on-device personalization: System design & applications. *arXiv preprint arXiv:2102.08503*, 2021. 1

[39] Xinchi Qiu, Javier Fernandez-Marques, Pedro PB Gusmao, Yan Gao, Titouan Parcollet, and Nicholas Donald Lane. Zerofl: Efficient on-device training for federated learning with local sparsity. *arXiv preprint arXiv:2208.02507*, 2022. 1, 3, 6

[40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1, 2, 6

[41] Rulin Shao, Hui Liu, and Dianbo Liu. Privacy preserving stochastic channel-based federated learning with neural network pruning. *arXiv preprint arXiv:1910.02115*, 2019. 1, 3

[42] Maying Shen, Pavlo Molchanov, Hongxu Yin, and Jose M Alvarez. When to prune? a policy towards early structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12247–12256, 2022. 6

[43] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020. 1

[44] Guanchu Wang, Zirui Liu, Zhimeng Jiang, Ninghao Liu, Na Zou, and Xia Hu. Division: Memory efficient training via dual activation precision. *arXiv preprint arXiv:2208.04187*, 2023. 3

[45] Wenyuan Xu, Weiwei Fang, Yi Ding, Meixia Zou, and Naixue Xiong. Accelerating federated learning for iot in big data analytics with pruning, quantization and selective updating. *IEEE Access*, 9:38457–38466, 2021. 6

[46] Weiming Zhuang and Lingjuan Lyu. Is normalization indispensable for multi-domain federated learning? *arXiv preprint arXiv:2306.05879*, 2023. 2, 3, 5

[47] Weiming Zhuang and Lingjuan Lyu. Fedwon: Triumphing multi-domain federated learning without normalization. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024. 3

[48] Weiming Zhuang, Yonggang Wen, Xuesen Zhang, Xin Gan, Daiying Yin, Dongzhan Zhou, Shuai Zhang, and Shuai Yi. Performance optimization of federated person re-identification via benchmark analysis. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 955–963, 2020. 3

[49] Weiming Zhuang, Xin Gan, Yonggang Wen, Shuai Zhang, and Shuai Yi. Collaborative unsupervised visual representation learning from decentralized data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4912–4921, 2021.

[50] Weiming Zhuang, Yonggang Wen, and Shuai Zhang. Divergence-aware federated self-supervised learning. In *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2022. 3

[51] Weiming Zhuang, Yonggang Wen, Lingjuan Lyu, and Shuai Zhang. Mas: Towards resource-efficient federated multiple-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23414–23424, 2023. 1