# REFT: Resource-Efficient Federated Training Framework for Heterogeneous and Resource-Constrained Environments

Humaid Ahmed H. Desai

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Amr Hilal, Chair
Hoda Eldardiry, Co-chair
Jin-Hee Cho

October 26, 2023
Blacksburg, Virginia

Keywords: Federated Learning, Variable Pruning, Knowledge Distillation, Resource Efficiency, Privacy

# REFT: Resource-Efficient Federated Training Framework for Heterogeneous and Resource-Constrained Environments

Humaid Ahmed H. Desai

(ABSTRACT)

Federated Learning (FL) is a sub-domain of machine learning (ML) that enforces privacy by allowing the user's local data to reside on their device. Instead of having users send their personal data to a server where the model resides, FL flips the paradigm and brings the model to the user's device for training. Existing works share model parameters or use distillation principles to address the challenges of data heterogeneity. However, these methods ignore some of the other fundamental challenges in FL: device heterogeneity and communication efficiency. In practice, client devices in FL differ greatly in their computational power and communication resources. This is exacerbated by unbalanced data distribution, resulting in an overall increase in training times and the consumption of more bandwidth. In this work, we present a novel approach for resource-efficient FL called *REFT* with variable pruning and knowledge distillation techniques to address the computational and communication challenges faced by resource-constrained devices. Our variable pruning technique is designed to reduce computational overhead and increase resource utilization for clients by adapting the pruning process to their individual computational capabilities. Furthermore, to minimize bandwidth consumption and reduce the number of back-and-forth communications between the clients and the server, we leverage knowledge distillation to create an ensemble of client models and distill their collective knowledge to the server. Our experimental results on image classification tasks demonstrate the effectiveness of our approach in conducting FL in a resource-constrained environment. We achieve this by training Deep Neural Network

(DNN) models while optimizing resource utilization at each client. Additionally, our method allows for minimal bandwidth consumption and a diverse range of client architectures while maintaining performance and data privacy.

# REFT: Resource-Efficient Federated Training Framework for Heterogeneous and Resource-Constrained Environments

Humaid Ahmed H. Desai

(GENERAL AUDIENCE ABSTRACT)

In a world driven by data, preserving privacy while leveraging the power of machine learning (ML) is a critical challenge. Traditional approaches often require sharing personal data with central servers, raising concerns about data privacy. Federated Learning (FL), is a cutting-edge solution that turns this paradigm on its head. FL brings the machine learning model to your device, allowing it to learn from your data without ever leaving your device. While FL holds great promise, it faces its own set of challenges. Existing research has largely focused on making FL work with different types of data, but there are still other issues to be resolved. Our work introduces a novel approach called REFT that addresses two critical challenges in FL: making it work smoothly on devices with varying levels of computing power and reducing the amount of data that needs to be transferred during the learning process. Imagine your smartphone and your laptop. They all have different levels of computing power. REFT adapts the learning process to each device's capabilities using a proposed technique called Variable Pruning. Think of it as a personalized fitness trainer, tailoring the workout to your specific fitness level. Additionally, we've adopted a technique called knowledge distillation. It's like a student learning from a teacher, where the teacher shares only the most critical information. In our case, this reduces the amount of data that needs to be sent across the internet, saving bandwidth and making FL more efficient. Our experiments, which involved training machines to recognize images, demonstrate that REFT works well, even on devices

with limited resources. It's a step forward in ensuring your data stays private while still making machine learning smarter and more accessible.

# Dedication

*Dedicated to my family, with a special tribute to my loving parents, Dr. Habibullah Desai and Mrs. Shahnaz Desai. Their unwavering dedication, sacrifices, and constant support have been the bedrock upon which I stand today. To my wonderful and loving siblings, Sahar, Junaid, and Nabila, whose unflagging faith in me has been a constant source of strength. And to my cherished cousin, Imran, whose support and encouragement have held immeasurable value in my journey. The love from all of you has meant the world to me.*

# Acknowledgments

I would like to begin by expressing my profound gratitude to Allah, the Almighty, for His boundless blessings, guidance, and strength He bestowed upon me to overcome the myriad challenges that dotted my path during this academic journey. I am forever grateful for His divine support, which has been my steadfast anchor.

I extend my profound appreciation to my dedicated advisor, Dr. Amr Hilal. His steadfast support, invaluable guidance, and perpetual motivation have been pivotal in shaping my research and nurturing my academic growth. I hold a special debt of gratitude for his patience and the time he dedicated to our weekly meetings, where he shared insights that greatly enriched my work.

Furthermore, I would like to convey my sincere gratitude to my co-advisor, Dr. Hoda Eldardiry, whose guidance and provision of essential resources have been invaluable throughout my research journey. Her profound insights, coupled with her unwavering support, have played a pivotal role in helping me not only traverse but also identify the path of this research direction.

My appreciation extends to Dr. Jin Hee-Cho for graciously accepting my request to be a part of my thesis committee and for her unwavering support of my dissertation.

To my family, especially my beloved parents and siblings, I owe a debt of gratitude that words can scarcely convey. Their unflagging support and understanding, even as I pursued my master's degree thousands of miles away from home, have been the foundation upon which my academic aspirations were built.

I extend my gratitude to my friends in Blacksburg, particularly Naveen, Priyanka, and Aravind, who aided me in surmounting numerous obstacles and shared and cherished long-

lasting memories. Their perpetual encouragement, assistance, and camaraderie during this journey have been genuinely invaluable. Lastly, I offer my heartfelt thanks to all those who, through various means, contributed to this captivating and beautiful odyssey. Your assistance and encouragement have played a substantial role in my academic accomplishments.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

CPU Central Processing Unit

DNN Deep Neural Network

FL Federated Learning

FLOPs Floating Point Operations per Second

GAN Generative adversarial network

GPU Graphics Processing Unit

IoT Internet of Things

ML Machine Learning

Non-IID Non-Independent and Identically Distributed

STC Sparse Ternary Compression

# Chapter 1

# Introduction

Recent advancements in deep learning have yielded significant progress across diverse domains, including image classification and natural language processing. Nonetheless, the training of Deep Neural Network (DNN) models for complex tasks, such as image classification and natural language processing, necessitates huge amounts of data. Training a DNN model with such a quantity of data works well in centralized scenarios where the model has access to all the training data. However, in the majority of cases, like IoT and edge devices, this wealth of data has been spread across multiple locations. This decentralized nature of data makes it challenging to collaborate across entities due to technical, privacy, and data ownership concerns. Consequently, it is infeasible to transfer and store sensitive data on a central server. To resolve these challenges, Federated Learning (FL) has been introduced as a promising approach that leverages distributed model training through decentralized data.

## 1.1 Problem Motivation

Traditional federated learning techniques involve iteratively sharing local model parameters, or gradients, during training. After each round of local training on local data, the local model sends its gradients to a central server. The central server aggregates the local model parameters using conventional data aggregation methods, and this process continues until the most recent global aggregation updates each local node's model. Classic federated algo-

rithms such as FedAvg [31] and its derivatives [4, 20, 27] are all based on directly averaging the model parameters. However, this communication strategy has a number of issues. First, it is only applicable to models with homogeneous architectures; averaging model parameters of varying architectures is non-trivial and is a research problem in itself. Second, parameter-based methods require frequent back-and-forth communication between the server and the clients [43]. This increases the communication bandwidth and makes the training process less efficient. Finally, naively averaging model parameters is known to have security vulnerabilities as a malicious user could intercept the sharing of parameters between the client and the server and obtain local private data from the shared gradients [7, 59]. So the first question is, "How can we perform FL in a way that consumes minimal bandwidth and involves a diverse range of client models while still maintaining the core principle of FL, data privacy?"

Moreover, client devices in federated learning are typically significantly more resource-constrained than servers in a data center in terms of processing power, memory, and storage. DNNs usually have millions of parameters, and training such networks on resource-constrained edge devices may require an inordinate amount of time and energy. This leads to the next question, "How can we perform FL wherein we can train DNN models while maximizing resource utilization at each client?"

Existing work aims to address challenges involved in the parameter-based communication strategy through the use of ensemble methods [1, 3, 6, 55] that allow us to average individual model predictions of multiple heterogeneous models. However, in the FL setting, due to high data heterogeneity and the number of clients, a typical ensemble algorithm is not feasible as it stores logits of all the models on the server to perform logit averaging. Another class of approaches uses knowledge distillation [16] to aggregate local models into a single central model. However, these approaches still pose privacy vulnerabilities as they either expose

private data or exchange model parameters.

Other approaches have also been proposed to make FL computationally efficient using model or gradient compression techniques. These approaches either apply quantization that replaces full 32-bit floating point numerical precision with a lower bit-width precision format such as INT8 or use a subset of the original model's parameter vector for training. However, both techniques have been known to significantly reduce the accuracy of the model.

## 1.2 Contribution

To address these challenges and answer the posed questions above, we propose *REFT*, a resource-efficient FL framework consisting of two primary components: variable structured model pruning and knowledge distillation. The pruning component reduces the number of model parameters, consequently reducing FLOPs (Floating Point Operations per Second) based on the client's computational capability, and the knowledge distillation focuses on improving communication efficiency through one-shot knowledge distillation while also addressing privacy vulnerabilities and the inclusion of heterogeneous model architectures. The knowledge distillation component is inspired by FedKD [9] which uses public, unlabeled data that is decoupled from the client's private data. This design eliminates the vulnerability to data privacy identified in the previous work. Moreover, unlike existing distillation-based approaches [24, 30, 45] that synchronously perform model updates by training local models at the client incrementally, we train local models asynchronously and independently before aggregating the local prediction on unlabeled public data. This procedure has two benefits. First, it reduces the repetitive communication requirements of synchronous updates, improving communication efficiency. Second, the use of public data ensures that the server's exposure to local knowledge (private data) is limited, thereby reducing the risk of information

leakage. Our extensive experiments on computer vision tasks with CIFAR10 and CIFAR100 demonstrate that we are able to drastically reduce the number of model parameters and FLOPs with comparable accuracy and much lower bandwidth consumption compared to the baselines.

We summarize our key contributions as follows:

- We introduce a novel variable model pruning technique that takes into account the computational power of individual clients. By performing one-shot structured pruning on the initial model weights at the server, we customize the pruning level for each client. This approach utilizes client resources efficiently and reduces both computational and communication overhead, making the model more suitable for training on resource-constrained devices.
- We employ a one-way, one-shot client-to-server knowledge distillation approach using unlabeled, non-sensitive public data. This technique further enhances communication efficiency in federated learning by optimizing the transfer of knowledge from clients to the server. We also accommodate clients with heterogeneous model architectures, which are obtained after structured pruning, enabling their active participation in the training process.

Overall, our proposed approach combines variable model pruning and one-shot knowledge distillation to improve the efficiency and effectiveness of federated learning, making it more feasible for resource-constrained devices, resulting in better resource utilization on the client side, and accommodating diverse client architectures.

# Chapter 2

# Literature Review

The proposed framework, REFT, is related to three broad areas: Efficient Federated Learning, Model Compression, and Knowledge Distillation. In this chapter, we briefly review existing relevant work and describe how the proposed framework bridges the research gap.

## 2.1 Efficient Federated Learning

McMahan et al. [31] first coined the term "federated learning" to address the problem of decentralized training over distributed data sources, paying particular attention to user data privacy by limiting direct access to users' confidential data. The state-of-the-art method for FL according to their proposed algorithm, Federated Averaging (FedAvg), involves performing numerous local gradient calculation steps on each client's local data during each round of training, followed by a parameter averaging step through a server. This process is repeated until the stopping criterion is satisfied. Over the years, several derivatives of FedAvg have been proposed that perform weighting based on the client's loss [27, 34] or introduce novel aggregation methods [18, 50]. FedProx [41] aims to solve the client-drift problem caused by the heterogeneity of local data by including a proximal term for local training. Several other lines of work like FedAdam [38], FedYogi [38], and FedAcc [56] are aimed at expediting convergence. Although these methods can result in convergence in various circumstances when data on different clients is non-independently and identically distributed (non-IID)

[18, 29, 41, 51], high training costs continue to be one of the main barriers preventing FL from being used in practice.

Recent studies have been geared toward reducing the communication cost between the clients and the server during the training process [19, 39, 42, 53, 54]. Konečný et al. [21] introduce two ways, structured and sketched updates, to reduce the cost of uplink client-to-server communication. The first method learns an update from a limited parametrized space with fewer variables, while the second seeks to learn a full model update and then compress it before sending it to the server using a combination of quantization, random rotations, and subsampling. However, this work only takes into consideration client-to-server communication costs and ignores server-to-client communication expenses. On the contrary, [2, 42] proposed to reduce downlink (server-to-client) communication costs by performing novel sparse ternary compression (STC) techniques and employing dropout with lossy compression to server-client exchanges.

Another line of work [53] also dictates the reduction of transferred bits during training with quantization but aims to further reduce the communication overhead by pruning and excluding unimportant communication with the server, i.e., model updates. However, even though there is a reduction in the number of bits transferred during training, the number of back-and-forth communications with the server still remains the same and thus lacks consideration of communication efficiency. Furthermore, our proposed REFT is much more efficient in terms of communication and computation, as it can achieve better model compression and performance with just pruning compared to the quantization, pruning, and selective update techniques combined in [53]. Of course, the addition of the distillation component makes our approach even more efficient.

## 2.2 Model Compression

Many studies have proposed various techniques to reduce the size, number of operations, and complexity of deep neural networks. These include pruning, quantization, and low-rank factorization. In this section, we will briefly summarize the work related to neural network pruning.

Neural network pruning optimizes the performance of large networks by selectively removing redundant or irrelevant connections or nodes from the network architecture. Early research examined the second-order Taylor expansion for approximation [15, 23]. However, computation of the Hessian matrix is very difficult and impractical, especially for current DNNs. Later, a new approach for pruning convolutional kernels based on Taylor expansion was proposed in [35] that approximates the change in the cost function caused by pruning network parameters. Han et al.'s work [13, 14] has sparked research interest in magnitude-based pruning methods where parameters with small enough magnitudes are eliminated from the network. However, the removal of individual parameter values leads to unstructured sparsity and requires custom hardware and software support to work in practice [52]. The "lottery ticket hypothesis," which is based on a discovery showing that a network pruned by magnitude consists of an ideal substructure of the original network, was introduced in [5, 36]. It demonstrates that direct training of the pruned network can achieve an accuracy comparable to that of pre-trained original network pruning.

Few recent works [19, 53] have employed pruning in FL to reduce communication and computation overhead. Specifically, [19] uses a two-stage approach that first performs initial pruning on a selected client and then "adaptively" prunes the model further during the FL process. Here, they refer to the term "adaptive" as being able to perform reconfiguration on the model, i.e., removing and adding back the parameters using the unstructured ap-

proach. Although unstructured pruning results in promising model performance, it has little to no significance in real-world applications, as it leads to sparse weight matrices that are irregular and rely on indices to be stored in compressed format [52]. Consequently, they are less compatible with the data-parallel architecture of GPUs and multicore CPUs, requiring specialized hardware and software support.

One-shot structured pruning based on the L1 norm was adopted in [53] along with quantization and selective updating techniques. Unlike unstructured pruning, structured pruning produces hardware-friendly weight matrices. To realize the actual benefits of pruning, pruning should be followed by parameter shrinkage by reconfiguring the shape of the weight matrices to reduce inference latency and model size. However, most model compression works, including [53] use simulation to check the performance of the pruned model, i.e., create a mask of weight matrices to assess the improvements in inference latency and model/parameter reduction. Such approaches have no real speedup or improvements in model or parameter size. Since reducing the complexity of the model is the ultimate goal of pruning, we perform model compression by reconfiguring the input and output tensors using the generated mask to reduce model complexity and improve the inference latency. Furthermore, [53] only investigates their approach in scenarios where the data is identically distributed. This lack of consideration makes their approach unsuitable to be used in practice as the data distribution among clients is often non-IID in nature, making efficient optimization challenging [26]. Keeping this in mind, our approach is designed to be robust against data heterogeneity and promises convergence with good performance.

## 2.3 Knowledge Distillation

Knowledge distillation was first introduced by Hinton et al. [16]. Since its inception, there has been significant progress in the model ensemble, with a particular focus on the student-teacher learning paradigm, where the student model tries to approximate the output logits of the teacher model [40, 48, 58]. Existing work either attempts to average the logits from the ensemble of teacher models or extract the knowledge from the feature level. The majority of the work relies on using the existing training data for the distillation process. Some work [32, 37] shows that distillation can be accomplished by creating pseudo-data from the weights of the teacher model or through a generator adversarially trained with the student in case the real data are unavailable for training. FedDF [30] uses an unlabeled dataset for ensemble distillation generated from a pre-trained GAN [11].

Guha et al. [12] proposed a one-shot FL approach in which the server learns a global model of devices in the federated network in a single communication round. However, unlike their approach, which uses unlabeled public data collected from the same domain, we employ the approach used by Gong et al. [8, 9, 10] that aggregates local predictions on unlabeled public data from different domains for better privacy guarantee. Unlike Gong et al., whose focus is more on preserving privacy, our approach is more geared toward increasing resource utilization and communication efficiency while maintaining privacy.

# Chapter 3

# Preliminaries

A typical Federated Learning system consists of a central server $S$ and a group of participating clients $C$. Each client in the network possesses its own labeled private dataset $D_C$, which it uses to train a local model $M_C$. The server $S$ then combines these local updates using an aggregation method to obtain an updated global model, $M_G$. This iterative process continues until a stopping condition is met. The widely used FL algorithm, FedAvg [31], initially defines a training task, including setting hyperparameters, and selects a fraction of clients for training. The initial global model $M_G^0$ is then broadcast. In each round $R$, clients $C$ perform a local computation on their respective data and update their model parameters $W_C^R$. The primary objective is to minimize the global loss, which is a weighted average of the individual client's loss function.

$$L(w) = \sum_{c=1}^{C} \frac{k_c}{k} \ell_c(w) \tag{3.1}$$

$$\text{where } \ell_c(w) = \frac{1}{k_c} \sum_{i \in D_c} f_i(w)$$

Here, $k_c$ represents the number of data samples in the client's dataset $D_c$, $k$ denotes the total number of samples, and $\ell_c$ is the loss function of client $c$.

Ensuring constant communication with the server is essential to achieving the objective in Eq. 3.1. However, in the context of IoT and edge devices, these devices are often resource-

constrained, which significantly limits communication and computation resources. Training DNNs on such devices becomes time-consuming, and multiple rounds of communication with the server result in substantial bandwidth consumption, making this approach inefficient.

# Chapter 4

# Proposed Framework: REFT

Our goal is to reduce the communication and computational costs of the Federated Learning (FL) process to include devices with limited resources. While resource-constrained devices such as Internet of Things (IoT) devices may not be able to train complex DNNs, the data generated by them is crucial from a learning standpoint. This is particularly relevant in scenarios where the majority of devices have limited resources, emphasizing the need to include resource-limited devices in the FL training process. Furthermore, current pruning techniques [19, 53] employ static pruning that prunes all the clients equally regardless of their ability to fit the unpruned model or model with a pruning level best suited for their hardware. PruneFL [19] can be considered partially static as it is adaptive to the client's hardware and prunes iteratively while training the model on the client. However, when the server or a selected client broadcasts the initial model to all clients in the network, with or without initial pruning, it makes a strong assumption that all clients are capable of training the initial model even after some initial pruning. Such pruning results in poor resource utilization by participating clients.

To overcome these issues, we propose a REFT framework with a three-stage pipeline that reduces the computational cost of training a complex DNN, improves resource utilization while training the client, and improves communication efficiency by minimizing the number of repeated communications between the client and the server while maintaining data privacy. In the first stage, we estimate the computational capacity of the client device by obtaining

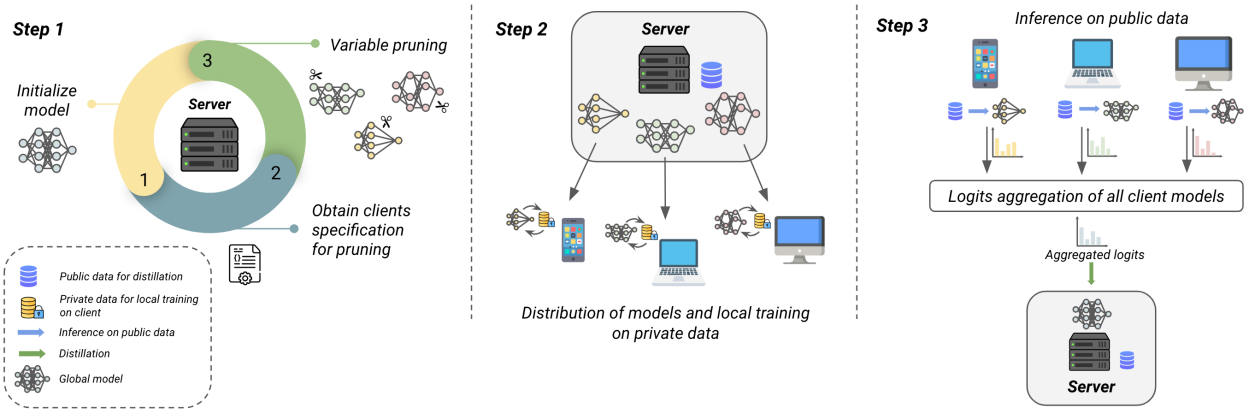



Figure 4.1: Overview of *REFT*

their FLOPS (Floating Point Operations Per Second) values and then prune the model to a level suitable for training on that client. This approach offers two advantages: First, pruning reduces the size of the model by reducing its parameters, making it less computationally demanding. Second, due to the reduced model size, less bandwidth is consumed in the initial model transfer from the server to the clients.

The second stage encompasses the distribution of the global model, followed by individual client model training on their respective private datasets. To maintain privacy, the server is restricted to accessing only public data. Notably, the server maintains its own public dataset, which we presume is universally available to all clients as an independent dataset. Consequently, for the purpose of bandwidth calculation, we omit the inclusion of this server-side public dataset, focusing solely bandwidth cost of server-client communication during FL training. This exclusion aligns with the established practices in existing works, as exemplified in [9, 30], wherein public datasets have been utilized. However, these studies have typically disregarded the associated communication costs of public datasets in their analyses.

In the final stage, we employ knowledge distillation. This method curbs the need for frequent and large model updates between clients and the server, significantly cutting communication costs. Additionally, this distillation-based approach enhances the framework's versatility,

allowing clients to possess distinct architectures aligned with their local data distribution and computational capabilities.

The details of the 3-stage REFT training process is depicted in Figure 4.1 and is further described below:

- *Variable model pruning*: In the initial round (round 0), the server $S$ initializes the global model parameters $W_G$ and prunes them according to client specifications. The pruned global model, $M^0_{G_P}$, is then sent to clients. Pruning is done just once using a one-shot approach, with pruning percentage $P$ ($100\% > P \geq 0\%$) varying based on client computational power. This method supports heterogeneous model architectures, allowing different clients to have distinct versions of $M^0_{G_P}$.

- *Model distribution and local training*: After pruning, the pruned model $M^0_{G_P}$ is broadcasted along with the public dataset $D_P$. Each client $C$ then trains the model on its private labeled dataset $D_c = \{(x^i_c, y^i_c)\}$ where $i = 1, 2, 3, ..., |D_c|$, and initializes the received global model $M^0_{G_P}$ with parameters $W_c$. It is important to note that the model architecture for each client may differ due to variable pruning or each client having its own custom model architecture.

- *Knowledge distillation*: To ensure privacy, the private datasets of clients are isolated. The public dataset $D_P$ on the server is employed for client-to-server knowledge distillation. An ensemble of local models $M_C$ and the global model $M_{G_P}$ forms a teacher-student arrangement. This maintains privacy while transferring knowledge from clients to the server.

## 4.1 Variable Pruning

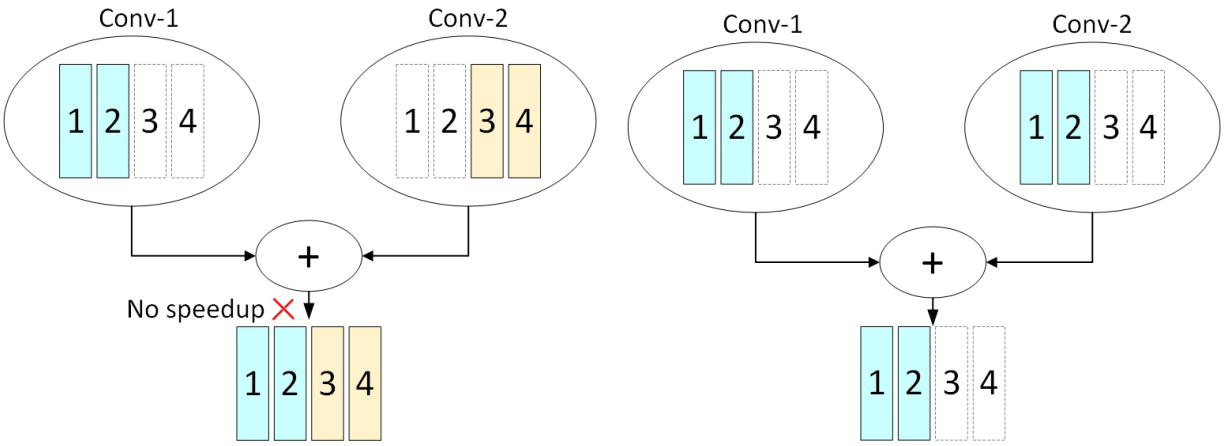
Training DNN on devices with limited resources has always been time-consuming, or sometimes it may not even be possible due to the complexity of neural networks. Pruning, as discussed in Section 2.2 doesn't just provide acceleration in inference time but also reduces model size and computational burden. However, pruning in the FL setting is not straightforward due to device heterogeneity. The high variation in the hardware capability of the clients participating in FL leads to large gaps in computational power among the clients, requiring clients with much less computational capability to train a model with a high degree of pruning. Static pruning methods often overlook the variability in computational power among clients in a federated learning network. These methods typically perform pruning based on the least powerful client, aiming to reduce communication bandwidth. However, this approach fails to consider clients with sufficient computational resources that can train an unpruned or less-pruned model effectively.

By applying a static pruning strategy based on the least powerful client, clients with higher computational power are underutilized as the pruning is not tailored to their hardware capabilities. This limitation hampers their ability to fully leverage their resources for training the pruned model. Consequently, the potential benefits of dynamic pruning and improved model performance remain unexploited. We exploit these areas with our variable pruning strategy, with the goal of achieving a good balance between computation, communication overhead, and performance. Our approach addresses the shortcomings of static pruning by tailoring the model pruning process to suit the computational capacities of each client. Unlike static pruning, which underestimates more potent clients, our strategy optimizes both computation and communication overheads. By assessing a client's computing capacity and estimating the necessary FLOPs for effective model training, we customize the pruning process. This results in models that are finely tuned for each client, enhancing the overall efficiency of

the FL process. Given the significance of FLOPs in hardware assessment for DNN training [46, 47], we prioritize FLOPS as the primary metric for computation assessment and pruning degree determination. FLOPS provides a reasonable and hardware-independent measure for assessing the feasibility of neural network training. As memory capacity and other client hardware parameters can also offer valuable insights, we complement this assessment with an analysis of memory requirements (RAM) and GPU utilization for the DNNs discussed in the following sections.

In our study, we adopted the L1 norm-based approach for one-shot structured pruning as it provides a straightforward way to measure the importance of weights in deep neural networks [25]. Other pruning criteria, such as those based on pruning during the training process [44], are not well-suited for our framework, as clients with limited resources may not be capable of training DNNs initially. In REFT, we apply pruning to both convolutional layers and fully connected layers. This approach is justified since convolutional layers typically contribute considerably to the computational overhead, while fully connected layers may primarily impact the storage space of the model size [25, 44]. The L1-norm pruning method described in [25] considers $c_i$ as the number of input channels for the $i^{th}$ convolutional layer, $h_i$ and $w_i$ as the height and width of the input feature map, and $\kappa \in \mathbb{R}^{k\times k}$ as the 2D kernel. By applying $c_{i+1}$ 3D filters $\mathcal{F}_{i,j} \in \mathbb{R}^{c_i k\times k}$ to the input channel, the input feature maps $x_i \in \mathbb{R}^{c_i\times h_i\times w_i}$ are transformed into output feature maps and used as input feature maps for the next convolutional layer. Removing a filter $\mathcal{F}_{i,j}$ also removes its corresponding feature map $x_{i+1,j}$, reducing the number of operations by $c_i k^2 h_{i+1} w_{i+1}$.

To realize the actual reduction in model size and improvement in inference time, the pruning needs to be followed by model shrinkage. To obtain a better reduction in model size and increase in speed gain, we employ dependency-aware pruning [33] and model shrinkage. Dependency-aware mode is a technique used in pruning algorithms to better take into account




(a) No speedup from pruning due to channel dependency

(b) Optimal pruning through dependency-aware mode

Figure 4.2: Addressing Channel Dependencies in Pruning using NNI's Dependency-aware Pruning.

the topological dependencies across different layers of a neural network.

Specifically, if we consider the addition of output channels from two convolutional layers, namely, conv1 and conv2 (in Figure 4.2a), it becomes evident that these two layers exhibit inter-channel dependencies. To elucidate dependency-aware pruning further, consider a scenario where we prune 50% of the output channels (filters) from conv1 and another 50% from conv2. While it might appear that both layers have undergone a 50% reduction in filters, the speedup module, which performs matrix reconfiguration, is still required to introduce zeros to align the output channels. In such a scenario, the anticipated gains in model pruning efficiency are not realized.

Dependency-aware Pruning harnesses the advantages of model pruning more effectively, as pruning decisions rely not only on the performance metrics (L1 norm) associated with each output channel but also on the broader network architecture and its topology. Specifically, the algorithm identifies and prunes output channels shared by layers that exhibit channel dependencies, as exemplified in Figure 4.2b (image courtesy [33]).


Figure 4.3: Overview of Pruning and Model Speedup using NNI Toolkit.

It ensures that pruning is performed in a way that preserves these dependencies, which can help to improve the final speed gain after the reconfiguration process. The pruner calculates the L1 norm sum of all the layers in the dependency set for each channel and generates a weight mask. Here, the dependency set includes all the layers that have channel dependencies with each other. The number of channels that can be pruned in the end is determined by the minimum sparsity of the layers in this dependency set. The pruning algorithm will then prune the same minimum sparsity channels for all the layers in the dependency set. The generated mask is then subsequently utilized by the NNI's ModelSpeedup module to reconfigure the weight tensors, ensuring a meaningful reduction in both model size and inference speed. The overall variable pruning procedure is described in Algorithm 1. The model speedup process using the NNI Toolkit [33] is described in Figure 4.3 (image courtesy [33]).

## 4.2 Knowledge Distillation

In the knowledge distillation phase, we initiate local model training $M_c$ with the private labeled dataset $D_c$ at each client. Post-local training, the server dispatches an unlabeled public dataset $D_p = (x_p^i)$, with $i = 1, 2, 3, ..., |D_p|$, to every client for knowledge distillation. Referring to [9], the private dataset $D_c = (x_c^i, y_c^i)$, $i = 1, 2, 3, ..., |D_c|$ (with $c \in C$), entails existing classes $T_c \subset 1, 2, ..., T$ ($T$ as total classes across clients). The local model's output on the public data sample $x_p^i$ for class $t \in T_c$ is $z_{tc}^i = f(x_p^i, w_c, t)$, with $w$ representing model parameters. In high-data heterogeneity settings, traditional aggregation methods averaging all teachers' logits lack suitability, as the client's dataset may not share identical target classes. To address this, we introduce the importance weight $I$ for each client, reflecting local private data distribution. The weight is computed as the ratio of samples in local client $c$ belonging to class $t$ to total samples across clients:

$$I_c^t = \frac{N_c^t}{\sum_{c \in C} N_c^t} \tag{4.1}$$

where $N_c^t$ is the sample count in local client $c$ for class $t$. It is important to note that this distillation holds true as long as the target class $t$ of public data sample $x_p^i$ matches the target class $T_c$ of the client's private dataset.

Kullback-Leibler divergence is used to aggregate all teachers' soft labels. The loss function, denoted by $L$, is computed as the sum of the cross-entropy between the teacher and student models' predicted probabilities, where $p_t$ and $q_t$ represent the probabilities of a sample belonging to class $t$ predicted by the teacher and student models, respectively. This is represented as:

$$L = \sum p_t \log \frac{p_t}{q_t} \tag{4.2}$$

The output logits of the central model $\tilde{z}_t = f(x_p, w_s, t)$ form the student knowledge, and the aggregated logits $\hat{z}_t$ make up the teacher's knowledge. To compute the probabilities $p_t$ and $q_t$, the softmax function is applied to the logits with a temperature parameter, $\tau$. [16] demonstrated that minimizing the loss function defined by Eq. (4.2) with a high-temperature parameter $\tau$ is equivalent to minimizing the L2 norm between the logits of the teacher and student networks. As a result, the loss function can be simplified as follows:

$$L = ||\tilde{z} - \hat{z}||, \tag{4.3}$$

where $\hat{z}$ and $\tilde{z}$ represent the logits of the teacher and student networks, respectively.

REFT employs a one-shot offline distillation, predicting with each public data sample once and iteratively training the central model. This boosts privacy, reduces queries to local models, and limits local knowledge exposure. Moreover, synchronous updates and repetitive communication are eliminated, enhancing communication efficiency and flexibility.

# Chapter 5

# Comparative Analysis of Resource Utilization Strategies

In this chapter, we present a comprehensive mathematical analysis centered on resource utilization in the context of REFT's variable pruning strategy and the established static pruning method. Our objective in this investigation is to compare and contrast the resource utilization patterns between Variable Pruning, which dynamically adjusts pruning rates based on runtime conditions, and Static Pruning, where pruning rates remain constant throughout execution. Understanding how these pruning strategies influence resource usage is paramount for optimizing model performance and enhancing system efficiency in a federated environment.

## 5.1 Variable and Static Pruning

In the context of federated learning with $n$ clients $(c_1, c_2, ..., c_n)$, each client $(c_i)$ possesses unique hardware capability $h_{c_i}$ and computation capacity $F_{c_i}$ in FLOP. Our analysis centers on variable pruning versus static pruning's efficiency in harnessing resources. The pruning level $P_{c_i}$ for a client depends on its hardware, with $NP_{c_i} = N \times (1 - P_{c_i})$ total pruned model parameters, where $N$ is the total number of parameters in the unpruned model. We assume uniform pruning across layers and connections and a uniform distribution of FLOPs across

pruned and unpruned parameters.

By reducing the number of parameters, we can reduce the training overhead and communication load. We define the FLOP reduction factor for client $c_i$ as $FP_{c_i} = F \times (1 - P_{c_i})$. The accuracy for training the pruned model is denoted as $AP_{c_i}$. It is worth noting that typically $A_{c_i} \geq AP_{c_i}$, where $A_{c_i}$ is unpruned model accuracy.

Static pruning ($P_{\text{static}}$) uses $P_{\min}$ based on least capable hardware ($h_{\min} = \min(h_1, h_2, ..., h_n)$). All clients, including those with higher hardware capabilities, are then pruned at the level $P_{\text{static}} = P_{\min}$, resulting in $NP_{\text{static}} = N \times (1 - P_{\text{static}})$ total number of parameters. Thus, higher-capability clients with $F_{c_i} > F_{\text{static}}$ might be underutilized, leading to reduced training performance. This underutilization results in the following inequality: $A_{c_i} \geq AP_{c_i} \geq AP_{\text{static}}$, where $1 \geq P_{\text{static}} \geq P_{c_i} \geq 0$ We can define the utilization factor for client $c_i$ as:

$$U_{c_i} = \frac{F_{\text{static}}}{F_{c_i}}, \tag{5.1}$$

indicating the ratio of the least performing client's hardware capacity to the client $c_i$'s hardware capacity. Our variable pruning strategy overcomes this underutilization by identifying a link between pruning level and FLOP reduction, which can be expressed as:

$$P_{c_i} = 1 - \frac{F_{c_i}}{F_\lambda} \tag{5.2}$$

Here, $F_{c_i}$ signifies client $c_i$'s computational capability (FLOPS), and $F_\lambda$ is a tradeoff coefficient between communication efficiency and accuracy. The choice of $F_\lambda$ allows the administrator (orchestrator of FL) to prioritize either more efficient communication or better accuracy/performance. Note that if $F_{c_i} \geq F_\lambda$, no pruning will be performed for client $c_i$.

For instance, consider 5 clients with FLOP capacities of 10, 20, 40, 60, and 100 GFLOPS.

**Algorithm 1** Variable Pruning

**Require:** $n$ clients, each client $c_i \in C$ has computation capacity of $F_{c_i}$ FLOPS where $i = 1, .., n$ , trade-off coefficient $F_\lambda$, client model $M_{c_i}$

**for** each client $c_i$ in $C$ **do**

$F_{c_i} \leftarrow$ Request client's estimated FLOPS

Calculate pruning ratio:

$P_{c_i} \leftarrow 1 - \frac{F_{c_i}}{F_\lambda}$ ▷ Eq. 4

Apply L1 Norm pruning and create weight mask:

$m_{c_i} \leftarrow L1NormPruner(M_{c_i}, P_{c_i})$

$ModelSpeedup(M_{c_i}, m_{c_i})$ ▷ Model reconfiguration

**end for**

By setting $F_\lambda$ to 100 GFLOPS, pruning percentages are: $P_{c_1} = 90\%$, $P_{c_2} = 80\%$, $P_{c_3} = 60\%$, $P_{c_4} = 40\%$, and $P_{c_5} = 0\%$ (no pruning). This choice reflects the administrator's preference for communication efficiency over performance. Alternatively, opting for $F_\lambda = 50$ GFLOPS, clients $c_1$, $c_2$, and $c_3$ are pruned to 80%, 60%, and 20% respectively, while clients $c_4$ and $c_5$ remain unpruned. This choice reflects the administrator's emphasis on accuracy or performance over communication costs. The variable pruning process utilizing Eq. (5.2) and the tradeoff coefficient $F_\lambda$ is described in Algorithm 1

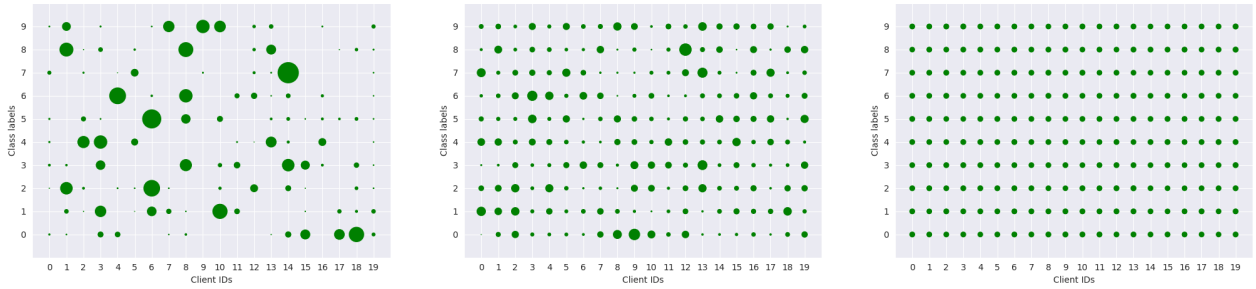
# Chapter 6

# Experimental Setup

This chapter outlines the comprehensive experimental framework employed in this study to evaluate the proposed REFT approach. In this chapter, we delve into the specifics of the datasets, models, baselines, and evaluation metrics that constitute the foundation of our research. Furthermore, we address the critical aspects of resource-efficient training, communication bandwidth, and implementation details that play a pivotal role in shaping the outcomes of our experimentation. The systematic presentation of our experimental setup serves to provide a clear roadmap for understanding the subsequent analysis and results, ultimately contributing to the comprehensive exploration of federated learning in resource-constrained environments.

## 6.1 Dataset and Models

We evaluate our proposed approach on image classification tasks using the CIFAR10 and CIFAR100 datasets [22]. While our experiments were exclusively conducted on these datasets, it is worth noting that our approach exhibits versatility and can potentially be adapted to accommodate different datasets, including but not limited to large-scale datasets like ImageNet, provided that the prerequisites outlined in Section 4.2 regarding the private and public dataset conditions are met. We construct private datasets for local training by leveraging heterogeneous data splits generated by a Dirichlet distribution, as suggested in previous




$\alpha = 0.1$ $\alpha = 1.0$ $\alpha = 1000.0$

Figure 6.1: Illustration of Dirichlet distribution on data heterogeneity across 20 clients. The dot size indicates the number of samples per class allocated to each client.

works [17, 57]. In this distribution, the parameter $\alpha$ controls the degree of non-IID-ness in the local datasets. Specifically, a larger $\alpha$ results in more similar data distributions across clients, whereas a smaller $\alpha$ leads to higher diversity. As shown in Figure 6.1, we visualize different $\alpha$ values to demonstrate the effect of data heterogeneity on 20 clients.

In line with established research practices and to facilitate rigorous comparisons with prior work, we have meticulously adhered to the experimental framework outlined in FedKD [9]. In scenarios where CIFAR10 serves as the private dataset, CIFAR100, a well-established public dataset, has been judiciously employed for the purposes of knowledge distillation. This configuration allows us to conduct a comprehensive evaluation of our proposed framework, with a primary focus on meticulously assessing test accuracy and bandwidth consumption and benchmarking it against established baselines.

In line with the principle of reproducibility and to ensure continuity with previous studies, we have opted to employ the ResNet-8 and VGG-16 model architectures. These architectural choices have been made in alignment with the specifications delineated in prior works, most notably FedKD [9] and FL-PQSU [53]. This consistent selection of model architectures ensures that our experiments are conducted under conditions that enable meaningful comparisons and evaluations.

Furthermore, to provide empirical evidence regarding the robustness and real-world applicability of our approach, we have extended our experiments to encompass non-IID data settings. In these experiments, we thoughtfully constructed training datasets for individual clients, each characterized by a value of $\alpha$ set to 1.0. This empirical approach allows us to explore the performance of our framework under conditions that more closely mirror the complexity of real-world data distributions in federated learning scenarios.

## 6.2 Baselines

In the scope of this study, our primary objective is to enhance the efficiency of resource utilization within resource-constrained environments while maintaining performance levels comparable to existing standards and minimizing communication bandwidth requirements. To achieve this objective with precision, we have consciously excluded the comparison with personalized federated learning methods such as Per-FedAvg [4] and q-FedAvg [28], which are tailored to adapt to the local data distributions unique to individual clients.

In our pursuit of assessing the effectiveness of our proposed REFT approach, we have undertaken a meticulous comparative analysis with four salient baseline methods. These methods encompass (i) FedAvg [31], which embodies the conventional federated learning approach; (ii) PruneFL [19], a technique that introduces adaptive distributed parameter-based pruning to reduce training time and energy consumption within the federated learning framework; (iii) FedKD [9], a method that leverages privacy preservation techniques with an emphasis on minimal communication; and (iv) FL-PQSU [53], an approach that introduces acceleration through a combination of pruning, quantization, and selective update technique, effectively managing both communication and training overhead.

## 6.3 Metrics

In our experiments, we assess model performance through test accuracy, comparing it against baseline methods. Our primary focus is on resource-efficient training, emphasizing substantial reductions in model size, parameter count, and FLOPs while maintaining negligible accuracy loss. Additionally, we analyze communication bandwidth, a critical factor in federated learning that directly impacts training time and costs.

For the non-distillation phase of training, we calculated communication bandwidth using the approach detailed in [49]. This calculation depends on several factors, including the parameter size $W$, the count of participating clients $C$, the number of communication rounds $R$, and the bits $B$ used for parameter and logit representation. This computation is expressed as:

$$Bandwidth = C \times R \times W \times B \tag{6.1}$$

To gain a finer-grained understanding of bandwidth impact, we compare both downstream (server-to-client) and upstream (client-to-server) communication costs per client for each round. In this context, Eq. (6.1) simplifies to $Bandwidth = W \times B$ for each communication. It's important to note that Eq. (6.1) is not applicable to response-based knowledge distillation since model parameters $W$ are not transferred during this process. For the distillation methods, communication bandwidth is calculated as follows:

$$Bandwidth = L \times S \times B \tag{6.2}$$

Here, $L$ represents the logits of each client, and $S$ is the number of distillation steps.

## 6.4 Implementation Details

We employed simulations to emulate the training of diverse clients on HPC clusters. To accomplish this, we gathered estimated FLOPS values for various potential client devices and integrated these values into our simulation framework. For instance, devices in the category of Raspberry Pi models 3 and 4, and similar counterparts, were categorized as weak clients due to their FLOPS capabilities falling within the range of 8 to 40 GFLOPS. In a similar context, wearable devices, including high-end smartwatches and mobile phones, were classified as moderate to good clients, exhibiting FLOPS capacities below 150 GFLOPS. Conversely, devices equipped with high-performance computing units, such as laptops and desktops featuring dedicated GPUs, were designated as strong clients. It is noteworthy that our hardware configuration encompassed two 12 GB NVIDIA GRID P40-12Q GPUs and an Intel(R) Xeon(R) CPU E5-2640 v4 clocked at 2.40GHz.

In the case of ResNet-8, each client model is trained using an SGD optimizer with a momentum parameter of 0.9 and weight decay set to $3 \times 10^{-4}$. We incorporate a Cosine Annealing scheduler, which gradually reduces the learning rate from 0.0025 to 0.001 over 500 epochs. A batch size of 16 is used during training.

For the VGG-16 model, we adopt a fixed learning rate of 0.1 without applying weight decay. The batch size is set to 128, and the optimizer and momentum parameters mirror those used for ResNet-8. In the context of distillation, we employ a consistent learning rate of $10^{-3}$ and a batch size of 512, utilizing the Adam optimizer.

To ensure a fair and direct comparison, we align our hyperparameters with those established in the FedKD framework. Thus, we employ an SGD optimizer with a momentum value of 0.9 and a weight decay of $3 \times 10^{-4}$. Similar to the ResNet-8 setup, we implement a Cosine Annealing scheduler that gradually reduces the learning rate from 0.0025 to 0.001 over the

course of 500 epochs. The batch size for this setup is 16.

For distillation, we maintain consistency with FedKD, utilizing a constant learning rate of $10^{-3}$ and a batch size of 512, employing the Adam optimizer.

# Chapter 7

# Results

This chapter presents the outcomes of our evaluation and analysis of the proposed REFT methodology. Our assessment covers various critical areas that influence the practicality and performance of the approach. These areas include:

- Resource Utilization: An evaluation of how efficiently our methodology employs available hardware resources.
- Model Size, Computation, and Inference Time: An exploration of the methodology's impact on model characteristics, computational demands, and inference speed.
- Communication Efficiency: An assessment of how well our approach handles data transfer and communication requirements, particularly in distributed and collaborative systems.

## 7.1 Resource Utilization

To demonstrate the effectiveness of our variable pruning strategy in optimizing hardware resource utilization, we compared accuracies per client across different pruning strategies (Figure 7.1). This experiment involved five clients with varying hardware capabilities. Clients $c_1$ and $c_2$ had limited computational capacity, necessitating a high pruning level (90%) to

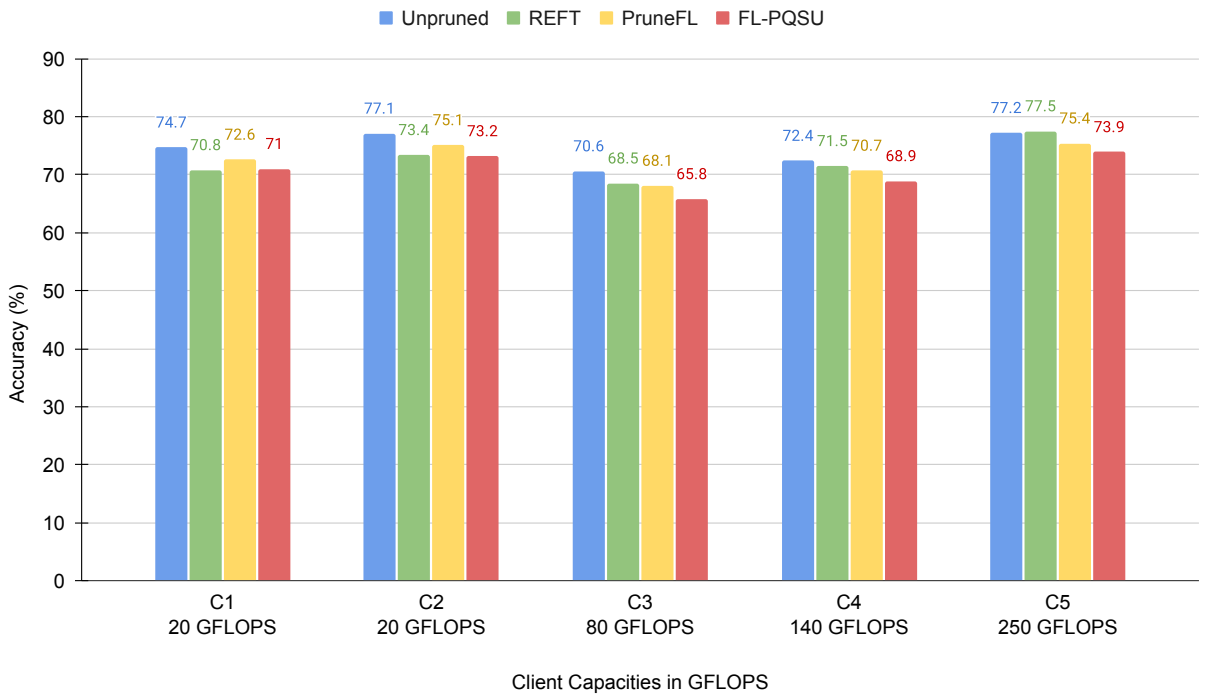



Figure 7.1: Comparing client accuracies: static pruning (FL-PQSU and PruneFL) vs. REFT on VGG-16 model.

accommodate the model. Pruning levels were determined using Eq. (5.2), with $F_\lambda$ set to 200 GFLOPS. Clients $c_3$ and $c_4$ possessed better hardware capabilities and required 60% and 30% pruning, respectively, for model training. Client $c_5$ had ample hardware resources and required no pruning.

For comparison, our proposed REFT employs three pruning levels: 30%, 60%, and 90%, based on client device FLOPs. As seen in Figure 7.2, increasing pruning lowers GPU utilization from around 84% to about 65%. Figure 7.3 shows accuracy variations with increased pruning. Pruning VGG-16 to 90% accelerates training by approximately 48%, reducing training time. In Figure 7.4, pruning to 90% decreases inference time by roughly 30%. This accelerates both training and inference, enhancing FL efficiency. Since REFT utilizes structured pruning, specifically based on the L1 norm (discussed in Section 4.1), to reduce

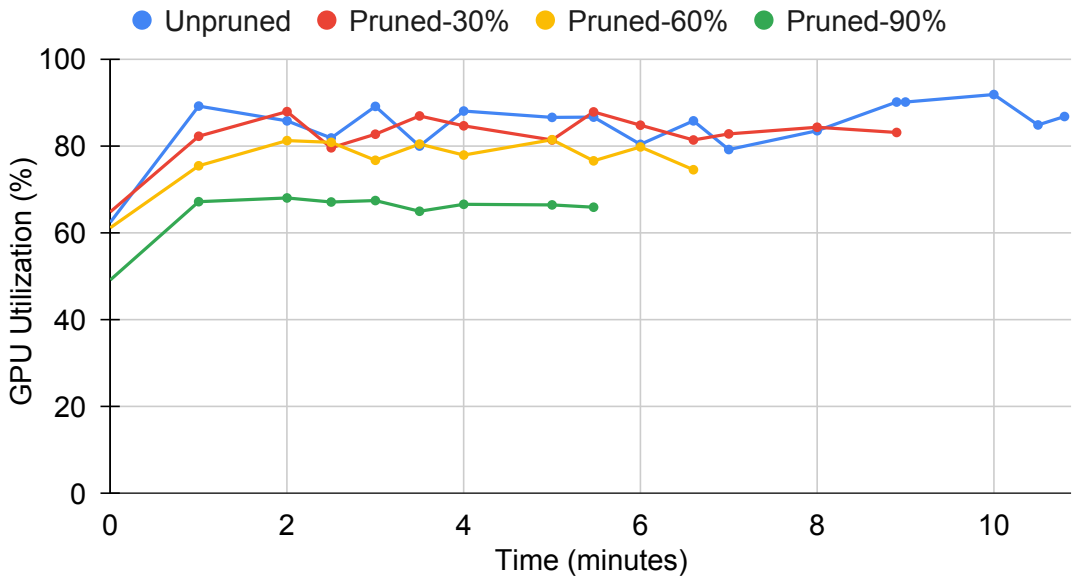



Figure 7.2: Effect of pruning on GPU utilization and training time of VGG-16.

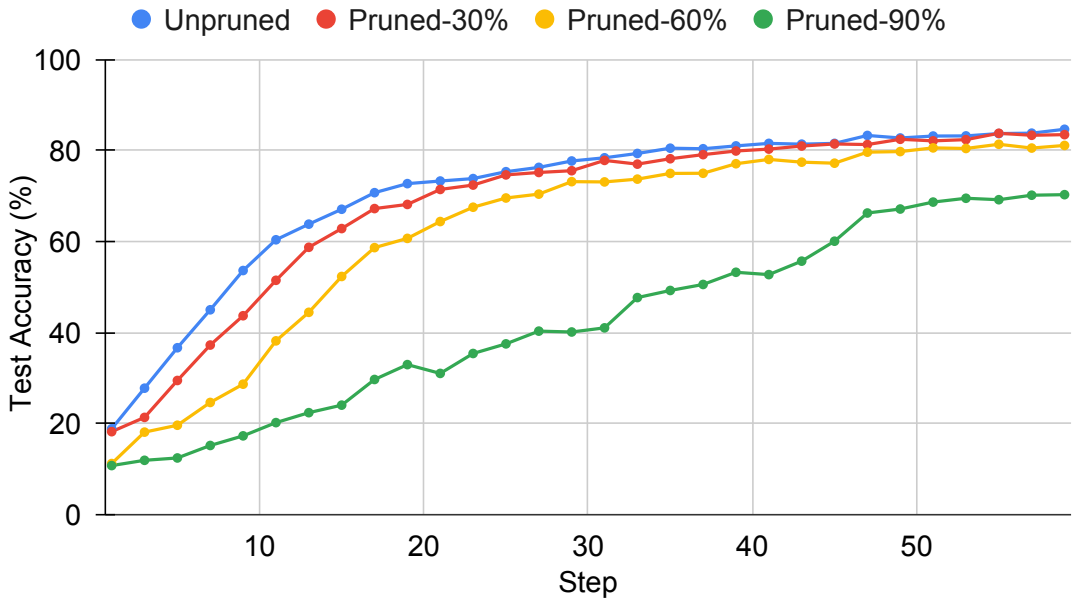



Figure 7.3: Effect of pruning on test accuracy of VGG-16.

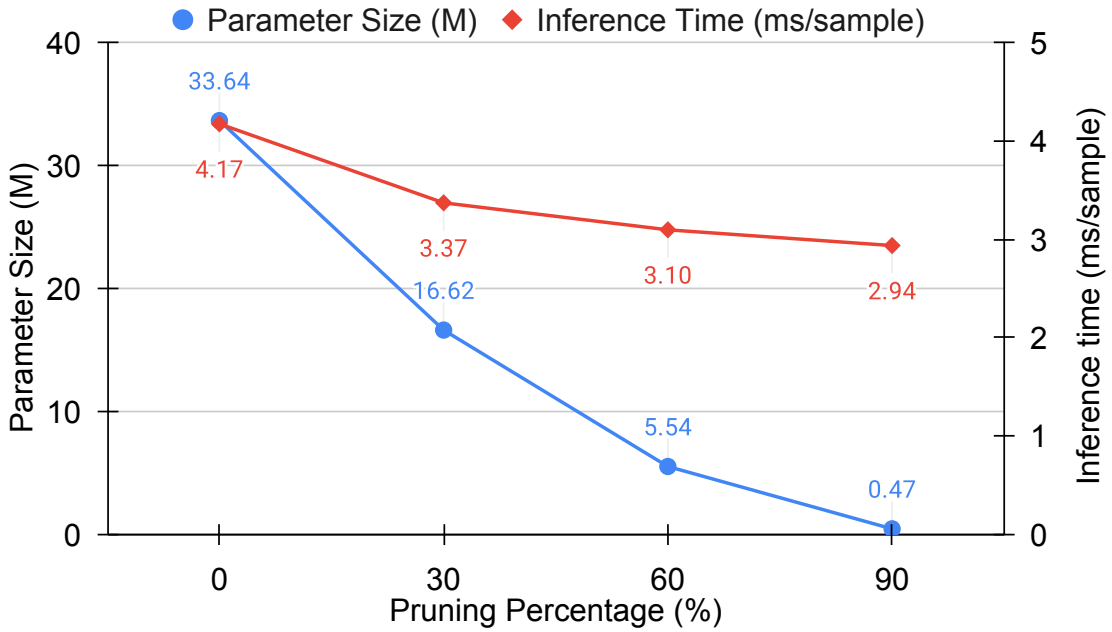



Figure 7.4: Parameter size and inference time reduction with an increase in pruning of VGG-16 by REFT.

model complexity, it's noteworthy that the structured pruning approach may result in a minor decline in accuracy, as illustrated in Figure 7.3. To retain performance, our approach avoids pruning clients capable of unpruned training. In contrast, approaches like FL-PQSU prune all clients to the least capable hardware level, not considering individual capacities and potentially increasing performance loss.

In Figure 7.1, the FL-PQSU method prunes the model for all clients using a level best suited for clients $c_1$ and $c_2$. While clients $c_1$ and $c_2$ achieve satisfactory model training, the hardware resources of $c_3, c_4, \text{and} c_5$ are underutilized, resulting in a decline in accuracy. This reduction in accuracy stems from the model being pruned to a higher level than what is ideally suited for their hardware, thereby leading to suboptimal overall performance. As a result, we observe that clients $c_3, c_4$, and $c_5$ exhibit higher accuracy when our variable pruning strategy is employed compared to the static pruning methods. This observation underscores the effectiveness of our approach in tailoring the pruning level to match the specific hardware

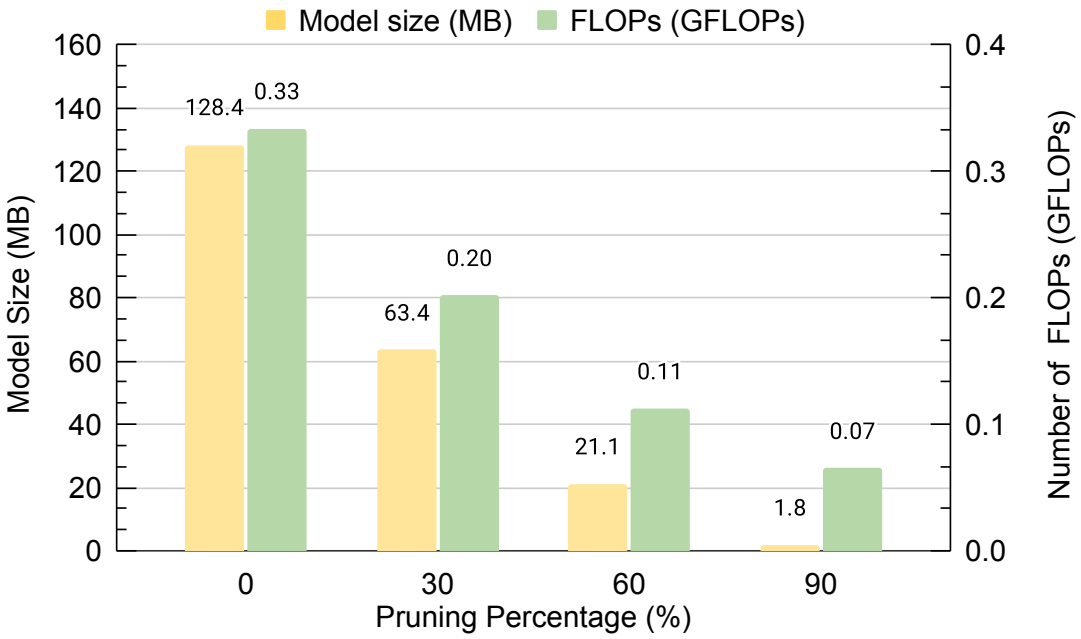



Figure 7.5: Reduction in model size and FLOPs of VGG-16 by REFT.

capabilities of each client. By leveraging the benefits of variable pruning, we achieve higher accuracy rates and overall improved performance. In addition to the observed improvements in resource utilization and performance, we observed RAM utilization during our experiments. We found that RAM usage ranged from 3 to 3.3 GB across different client models. This indicates that the optimal execution of our approach aligns with industry standards, as even a relatively weak client device like the Raspberry Pi 4, which features up to 8 GB of RAM, comfortably meets the minimum memory requirement, highlighting the practicality and accessibility of our proposed approach across a range of hardware configurations.

## 7.2 Model Size, Computation, and Inference Time

In this section, we delve into an exploration of the impacts of REFT on model size, computational efficiency, and inference speed (as depicted in Figures 7.4 and 7.5). Our investigations

illuminate that REFT yields noteworthy reductions in model size, FLOPs, and inference time, all while incurring negligible losses in accuracy.

We exclude FL-PQSU and PruneFL from the direct computational and inference time comparisons. This exclusion is attributed to the characteristics of their pruning methodologies. Both FL-PQSU and PruneFL induce sparsity in the model through their pruning techniques, necessitating specialized hardware and software infrastructure for computation and inference time measurement, as comprehensively discussed in Section 2.2.

For a comprehensive assessment of the impact of REFT on model size and computational efficiency, we measured the number of model parameters and FLOPs for VGG-16 across various pruning levels. Figure 7.5 illustrates the reductions achieved by REFT. Our pruning strategy compresses the model from its original size of 128.4 MB to a mere 1.8 MB, representing a reduction of 98.5%. Simultaneously, the FLOPs (computation) decreased from 0.33 GFLOPs to a mere 0.07 GFLOPs.

Moreover, our pruning strategy extends its influence to enhance model speed, chiefly through a reduction in inference time. Figure 7.4 demonstrates these improvements in inference time in conjunction with the parameter size reduction. Through REFT, we reduce the parameter size of VGG-16 from an initial 33.6 million (M) parameters to a mere 0.47 million parameters, marking a reduction of approximately 98.3%. Correspondingly, the inference time experiences a noteworthy reduction, plummeting from 4.17 milliseconds per sample to a streamlined 2.94 milliseconds per sample, all across a sample size of 1000.

These results underscore the compelling advantages of REFT in reducing model complexity and computational demands while concurrently enhancing resource utilization—a testament to the practicality and potential of our pruning strategy in resource-constrained environments.

## 7.3 Communication Efficiency

In this section, we conduct a comprehensive evaluation of REFT's communication efficiency, leveraging the ResNet-8 and VGG-16 models as benchmarks for our analysis. The results, summarized in Table 7.1, provide insights into both total communication bandwidth and test accuracy, facilitating a thorough assessment of REFT's performance relative to the baseline methods.

Notably, REFT achieves the lowest total communication bandwidth while maintaining comparable or superior accuracy compared to the baselines. Due to the variable pruning employed by REFT, we present accuracy and bandwidth as a range, reflecting the diversity of pruned clients and associated ratios. This range spans various scenarios, accommodating highly resource-constrained clients (requiring up to 90% pruning) to those with more abundant resources (needing as little as 30% pruning). This variability underscores REFT's adeptness at striking an optimal balance between model accuracy and bandwidth efficiency, catering effectively to the demands of diverse federated learning environments.

For both FedKD and REFT, the knowledge distillation process incurs minimal communication costs as it involves only 200 distillation steps. In contrast, FedAvg, FL-PQSU, and PruneFL require a significantly greater number of communication rounds (ranging from 900 to 1000 for ResNet-8 and approximately 500 for VGG-16) to complete the federated learning training. Moreover, given the dataset that we are using, the cost associated with transferring logits is considerably less than that associated with transferring model weights. Therefore, the total bandwidth of REFT and FedKD is far less than the other baselines. Since PruneFL conducts iterative pruning and reconfiguration at fixed intervals, the pruning ratio is not constant throughout the training process. To simplify this for analysis and comparison, we consider its final pruning ratio. Consequently, we provide approximate values for bandwidth

| Model | Method | Pruning Ratio (%) | Accuracy (%) | Bandwidth | | |
|---|---|---|---|---|---|---|
| | | | | Downstream | Upstream | Total |
| ResNet-8 | FedAvg | - | 68.3 | 37.63 MB | 37.63 MB | 73.5 GB |
| | FL-PQSU | 30% | 73.7 | 37.63 MB | **4.7 MB** | 164 GB |
| | | 60% | 71.9 | 37.63 MB | **4.7 MB** | 164 GB |
| | | 90% | 70.3 | 37.63 MB | **4.7 MB** | 164 GB |
| | PruneFL | Adaptive (∼60%) | 76.2 | ∼20.8 MB | ∼20.8 MB | ∼36.6 GB |
| | FedKD | - | 81.5 | 37.63 MB | 4.92 MB | 42.55 MB |
| | REFT | Variable (30-90%) | 80.8 - **81.7** | **3.65 - 25.74 MB** | 7.81 MB | **11.46 - 33.5 MB** |
| VGG-16 | FedAvg | - | 59.3 | 256.6 MB | 256.6 MB | 200 GB |
| | FL-PQSU | 30% | 74.5 | 256.6 MB | 32 MB | 140.1 GB |
| | | 60% | 73.1 | 256.6 MB | 32 MB | 140.1 GB |
| | | 90% | 71.3 | 256.6 MB | 32 MB | 140.1 GB |
| | PruneFL | Adaptive (∼60%) | 78.9 | ∼102 MB | ∼102 MB | ∼93.6 GB |
| | FedKD | - | 79.3 | 256.6 MB | **4.92 MB** | 261.5 MB |
| | REFT | Variable (30-90%) | 77.6 - **80.8** | **3.6 - 126.8 MB** | 7.81 MB | **11.4 - 134.6 MB** |

Table 7.1: Comparison of accuracy (central) and communication efficiency on RestNet-8 and VGG-16 models with 20 clients ($C = 20$). The table presents the downstream and upstream communication costs per client for each round, excluding FedKD and REFT. The total column represents the total bandwidth cost for a client to complete the federated learning training.

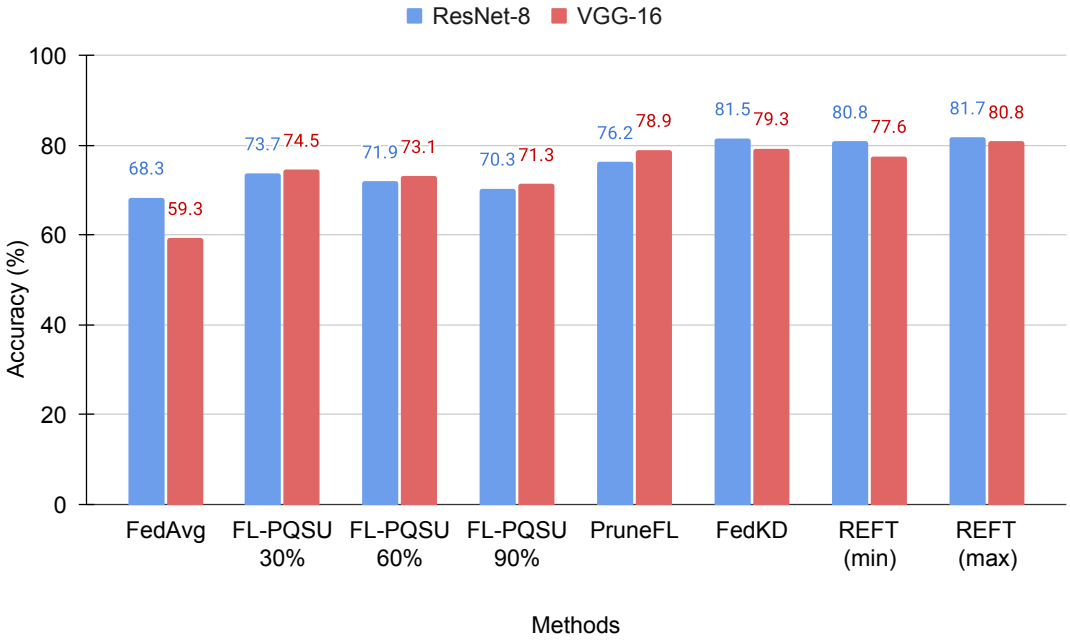



Figure 7.6: Comparision of accuracies at the server after completion of FL training.

to enhance clarity and simplify the comparison process.

FL-PQSU stands out in terms of upstream cost reduction among weight-sharing methods like FedAvg and PruneFL. This advantage can be attributed to its utilization of INT8 quantization during client-server communication. As elaborated in Section 2.2, FL-PQSU maintains consistent total bandwidth usage across various pruning levels, primarily because it avoids the need for post-pruning model weight reconfiguration. Consequently, this approach leads to the emergence of sparse weight tensors and a model architecture that remains unaltered. As a result, the actual model size and computational requirements experienced no meaningful change.

Table 7.1 highlights our method's superior downstream communication efficiency for both ResNet-8 and VGG-16 models, outperforming other pruning-based techniques like FL-PQSU and PruneFL. These methods anticipate sparse matrix support in future hardware and soft-

ware developments, but such resources are not yet available. As a result, they encounter practical constraints, particularly PruneFL, which relies heavily on sparse matrices. In contrast, REFT adopts structured pruning, yielding hardware-friendly weight matrices that undergo size reduction through shape reconfiguration. This strategy significantly reduces parameter size and enhances latency. Furthermore, REFT and FedKD exhibit lower bandwidth needs than other baselines, leveraging one-shot distillation. Remarkably, FedKD employs quantization before transmitting logits to the server, further curbing upstream communication costs.

The accuracies presented in Table 7.1 are graphically visualized in Figure 7.6. It's worth noting that these accuracies represent values reported at the server following the completion of the federated learning training process. The collective findings in this section underscore REFT's capability of reducing communication overhead while concurrently preserving model performance.

# Chapter 8

# Conclusion and Future Directions

## 8.1 Conclusion

In this work, we address the challenges of communication efficiency and resource constraints in federated learning through our proposed approach, REFT. By incorporating variable pruning and knowledge distillation techniques, we have demonstrated improvements in resource utilization, training time, and bandwidth consumption. Our experiments have shown that REFT outperforms existing baselines in terms of accuracy while significantly reducing computational and communication overhead. The adaptive nature of our variable pruning technique allows clients with varying computational capabilities to effectively contribute to the training process, ensuring better utilization of their resources. Additionally, the one-way, one-shot knowledge distillation approach on unlabeled public data reduces the need for iterative and repetitive communications while preserving privacy, further enhancing communication efficiency and privacy.

## 8.2 Future Directions

While our work has provided promising results in communication-efficient federated learning, there are several avenues for future exploration and enhancement. One potential direction is the incorporation of quantization techniques to further reduce upstream bandwidth commu-

nication. Quantization can help compress the transmitted gradients or model updates into low-precision representations, effectively reducing the amount of data exchanged between clients and the server. By exploring the application of quantization methods in conjunction with our variable pruning and knowledge distillation techniques, we can achieve even greater improvements in communication efficiency without compromising model performance.

# Bibliography


[1] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Róbert Ormándi, George E. Dahl, and Geoffrey E. Hinton. Large scale distributed neural network training through online distillation. *CoRR*, abs/1804.03235, 2018. URL http://arxiv.org/abs/1804.03235.

[2] Sebastian Caldas, Jakub Konečný, H. Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *CoRR*, abs/1812.07210, 2018. URL http://arxiv.org/abs/1812.07210.

[3] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. *CoRR*, abs/1903.11341, 2019. URL http://arxiv.org/abs/1903.11341.

[4] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning: A meta-learning approach. *CoRR*, abs/2002.07948, 2020. URL https://arxiv.org/abs/2002.07948.

[5] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. URL http://arxiv.org/abs/1803.03635.

[6] Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks, 2018.

[7] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? *CoRR*, abs/2003.14053, 2020. URL https://arxiv.org/abs/2003.14053.

[8] Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyan Wu, Terrence Chen, David Doermann, and Arun Innanje. Ensemble attention distillation for privacy-preserving federated learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15056–15066, 2021. doi: 10.1109/ICCV48922.2021.01480.

[9] Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyan Wu, Terrence Chen, David S. Doermann, and Arun Innanje. Preserving privacy in federated learning with ensemble cross-domain knowledge distillation. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 11891–11899. AAAI Press, 2022. URL https://ojs.aaai.org/index.php/AAAI/article/view/21446.

[10] Xuan Gong, Liangchen Song, Rishi Vedula, Abhishek Sharma, Meng Zheng, Benjamin Planche, Arun Innanje, Terrence Chen, Junsong Yuan, David Doermann, and Ziyan Wu. Federated learning with privacy-preserving ensemble attention distillation, 2022.

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014. URL http://arxiv.org/abs/1406.2661.

[12] Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *CoRR*, abs/1902.11175, 2019. URL http://arxiv.org/abs/1902.11175.

[13] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, vol-

ume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf.

[14] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1510.00149.

[15] Babak Hassibi, David Stork, and Gregory Wolff. Optimal brain surgeon: Extensions and performance comparisons. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993. URL https://proceedings.neurips.cc/paper_files/paper/1993/file/b056eb1587586b71e2da9acfe4fbd19e-Paper.pdf.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[17] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *CoRR*, abs/1909.06335, 2019. URL http://arxiv.org/abs/1909.06335.

[18] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. *CoRR*, abs/2003.08082, 2020. URL https://arxiv.org/abs/2003.08082.

[19] Yuang Jiang, Shiqiang Wang, Bong Jun Ko, Wei-Han Lee, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *CoRR*, abs/1909.12326, 2019. URL http://arxiv.org/abs/1909.12326.

[20] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, abs/1910.06378, 2019. URL http://arxiv.org/abs/1910.06378.

[21] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492, 2016. URL http://arxiv.org/abs/1610.05492.

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[23] Yann Lecun, J. S. Denker, Sara A. Solla, R. E. Howard, and L.D. Jackel. Optimal brain damage. In David Touretzky, editor, *Advances in Neural Information Processing Systems (NIPS 1989), Denver, CO*, volume 2. Morgan Kaufmann, 1990.

[24] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *CoRR*, abs/1910.03581, 2019. URL http://arxiv.org/abs/1910.03581.

[25] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rJqFGTslg.

[26] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *CoRR*, abs/1908.07873, 2019. URL http://arxiv.org/abs/1908.07873.

[27] Tian Li, Maziar Sanjabi, and Virginia Smith. Fair resource allocation in federated learning. *CoRR*, abs/1905.10497, 2019. URL http://arxiv.org/abs/1905.10497.

[28] Tian Li, Maziar Sanjabi, and Virginia Smith. Fair resource allocation in federated learning. *CoRR*, abs/1905.10497, 2019. URL http://arxiv.org/abs/1905.10497.

[29] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJxNAnVtDS.

[30] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/18df51b97ccd68128e994804f3eccc87-Abstract.html.

[31] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016. URL http://arxiv.org/abs/1602.05629.

[32] Paul Micaelli and Amos J. Storkey. Zero-shot knowledge transfer via adversarial belief matching. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9547–9557, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/fe663a72b27bdc613873fbbb512f6f67-Abstract.html.

[33] Microsoft NNI Contributors. Overview of NNI Model Pruning; Neural Network Intelligence — nni.readthedocs.io. https:

//nni.readthedocs.io/en/stable/compression/pruning.html#dependency-aware-mode-for-output-channel-pruning, 2023. [Accessed 14-08-2023].

[34] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. *CoRR*, abs/1902.00146, 2019. URL http://arxiv.org/abs/1902.00146.

[35] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR*, abs/1611.06440, 2016. URL http://arxiv.org/abs/1611.06440.

[36] Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/a4613e8d72a61b3b69b32d040f89ad81-Paper.pdf.

[37] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, R. Venkatesh Babu, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. *CoRR*, abs/1905.08114, 2019. URL http://arxiv.org/abs/1905.08114.

[38] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. *CoRR*, abs/2003.00295, 2020. URL https://arxiv.org/abs/2003.00295.

[39] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In Silvia Chiappa and Roberto Calandra, editors, *Proceed-*

*ings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2021–2031. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/reisizadeh20a.html.

[40] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015.

[41] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018. URL http://arxiv.org/abs/1812.06127.

[42] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3400–3413, 2020. doi: 10.1109/TNNLS.2019.2944481.

[43] Osama Shahid, Seyedamin Pouriyeh, Reza M. Parizi, Quan Z. Sheng, Gautam Srivastava, and Liang Zhao. Communication efficiency in federated learning: Achievements and challenges. *CoRR*, abs/2107.10996, 2021. URL https://arxiv.org/abs/2107.10996.

[44] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P. Namboodiri. Play and prune: Adaptive filter pruning for deep model compression. *CoRR*, abs/1905.04446, 2019. URL http://arxiv.org/abs/1905.04446.

[45] Dianbo Sui, Yubo Chen, Jun Zhao, Yantao Jia, Yuantao Xie, and Weijian Sun. FedED: Federated learning via ensemble distillation for medical relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2118–2128, Online, November 2020. Association for Computational

Linguistics. doi: 10.18653/v1/2020.emnlp-main.165. URL https://aclanthology.org/2020.emnlp-main.165.

[46] Raphael Tang, Weijie Wang, Zhucheng Tu, and Jimmy Lin. An experimental analysis of the power consumption of convolutional neural networks for keyword spotting. *CoRR*, abs/1711.00333, 2017. URL http://arxiv.org/abs/1711.00333.

[47] Raphael Tang, Ashutosh Adhikari, and Jimmy Lin. Flops as a direct optimization objective for learning sparse neural networks. *CoRR*, abs/1811.03060, 2018. URL http://arxiv.org/abs/1811.03060.

[48] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[49] Saeed Vahidian, Mahdi Morafah, and Bill Lin. Personalized federated learning by structured and unstructured pruning under data heterogeneity. *CoRR*, abs/2105.00562, 2021. URL https://arxiv.org/abs/2105.00562.

[50] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris S. Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *CoRR*, abs/2002.06440, 2020. URL https://arxiv.org/abs/2002.06440.

[51] Jiayi Wang, Shiqiang Wang, Rong-Rong Chen, and Mingyue Ji. Local averaging helps: Hierarchical federated learning and convergence analysis. *CoRR*, abs/2010.12998, 2020. URL https://arxiv.org/abs/2010.12998.

[52] Yanzhi Wang, Shaokai Ye, Zhezhi He, Xiaolong Ma, Linfeng Zhang, Sheng Lin, Geng Yuan, Sia Huat Tan, Zhengang Li, Deliang Fan, Xuehai Qian, Xue Lin, and Kaisheng

Ma. Non-structured DNN weight pruning considered harmful. *CoRR*, abs/1907.02124, 2019. URL http://arxiv.org/abs/1907.02124.

[53] 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. doi: 10.1109/ACCESS.2021.3063291.

[54] Zhaohui Yang, Mingzhe Chen, Walid Saad, Choong Seon Hong, and Mohammad Shikh-Bahaei. Energy efficient federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications*, 20(3):1935–1949, 2021. doi: 10.1109/TWC.2020.3037554.

[55] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1285–1294, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098135. URL https://doi.org/10.1145/3097983.3098135.

[56] Honglin Yuan and Tengyu Ma. Federated accelerated stochastic gradient descent. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5332–5344. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/39d0a8908fbe6c18039ea8227f827023-Paper.pdf.

[57] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks, 2019.

[58] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improv-

ing the performance of convolutional neural networks via attention transfer. *CoRR*, abs/1612.03928, 2016. URL http://arxiv.org/abs/1612.03928.

[59] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *CoRR*, abs/1906.08935, 2019. URL http://arxiv.org/abs/1906.08935.