Heterogeneous Federated Learning with Differentiable Architecture Compression

Guihong Li
The University of Texas at Austin
lgh@utexas.edu

Sunny Sanyal
The University of Texas at Austin
sanyal.sunny@utexas.edu

Abstract

Despite several merits, federated learning couldn't handle hardware heterogeneity well. Specifically, every client device has a distinct hardware configuration that differs in storage, power, and computation capability. Hence a global model that may work for a resource-rich device may not fit for a resource-constrained device. This phenomenon limits federated learning only to high-end resource-abundant devices. To address the challenge of hardware heterogeneity, we propose a neural architecture search-based differentiable architecture compression (DAC) approach that computes suitable neural architectures given device configurations of the participating devices. Our experiments show that our proposed algorithm outperforms the baseline regarding the compression of a MobileNet-V2 architecture, and the DAC generated models exhibit reasonable accuracy in multiple federated learning scenarios.

1. Introduction

The concept of Federated learning (FL) has been introduced in 2017 [1] by some researchers in google. Since then, it has emerged as one of the most popular alternatives for decentralized training. The tremendous growth in the storage and computation power of edge devices within decentralized networks has led to the advent of the federated learning paradigm. Federated learning (FL) involves training models over distributed client devices without centralizing the device data to the server. Privacy is a crucial characteristic of federated learning, where the data located in the client devices are never shared or transferred. This makes FL suitable for privacy-sensitive applications for hospitals, surveillance cameras, and smartphones. Moreover, in FL setup, a client can be an institution or an individual device.

Traditionally federated learning [2] assumes that all the devices are resource-abundant and homogeneous. Hence it computes a global model for all the participating devices. This assumption limits the application of federated learning only to resource-rich devices as resource-constrained devices with a small memory (10 15 MBs) may not load a big VGG16 with a size of 256 MB. Broadly the participating devices in federated learning scenarios differ in storage (RAM/ROM size), computational capacity (FLOPs), battery power due to the diversity in the hardware configurations. This diversity in hardware leads to the challenge of hardware heterogeneity in FL scenarios.

Handcrafted models cannot practically solve the issues caused by hardware heterogeneity as developers cannot create multiple models in runtime. To circumvent this open problem, we employ an AutoML technique that automatically rewires and compresses a given neural network architecture to produce the optimal architecture(s) for a particular resource budget based on hardware configurations. Particularly we propose the Differtiable Architecture Search (DAC) that is inspired by DARTS [3] based architecture search. DAC produces multiple neural network architectures suitable for distinct devices with diverse hardware configurations. It identifies and removes unwanted channels based on the entropy computations of the channel weights. The removal of unwanted channels is the key to higher compression of the neural network architecture.

This paper has two major contributions:

- The proposed DAC is a novel compression approach that could achieve 2x compression without affecting the test accuracy.
- We also present a clustering-based federated learning approach for resource-constrained devices that are shown to solve the device heterogeneity problem.

The rest of the paper is organized as follows. Section 2 compares and contrasts similar works. The following section provides useful background regarding federated learning and compression techniques. In section 4, we present the core of our paper, i.e., the proposed methodology for both DAC and federated learning. In section 5, we discuss our experiments and results. Finally, in section 6, we sum-

marize the important aspects of this paper and suggest future work.

2. Related Work and Background

The open challenges of federated learning are discussed in [2]. [4] has proposed a multitask federated learning approach to tackle both model and system-level heterogeneity. Some literature solves model and system heterogeneity separately, but they are not comparable to our approach. Recently NAS has also been introduced as a prospective solution in [5]. [6] has also presented inspiring results that show NAS can be used to search models for heterogeneous devices with limited resources. It applies compression and pruning to mobile-sized pre-trained models to fit in resource-constrained devices. Another approach MiLeNAS [7] applies federated NAS to cross-organization clients with GPU-equipped devices. Our approach is different in two important ways: First, the goal of our paper is to solve federated learning problems using NAS as a tool, unlike [5], [6], [7] where they have focused their efforts to build a better NAS-based system. Second, we also plan to build a model aggregation strategy, unlike the DecNAS [6] and MiLeNAS [7].

Federated learning is a decentralized machine learning paradigm where distributed client devices train a global model located at a central server without sharing their data. The three major steps involved in federated learning are a) the server updates the clients with the current global model, b) the clients retrain their local models (previously the global model)for a small number of epochs with the local data and updates the server, c) the server aggregates the local models of a subset of sampled users and computes a new global model, this is repeated through many communication rounds until convergence. Typically the canonical federated learning aims to minimize the following objective function:

$$\min_{\omega} F(\omega), where\ F(\omega) := \sum_{k=1}^{m} p_k F_k(\omega)$$

The client devices in a federated learning scenario can have diverse hardware configurations that result in different storage, battery power, and computational capabilities among several devices. This phenomenon is known as system or hardware heterogeneity in federated learning. The hardware heterogeneity affects the overall performance as it causes stragglers. Stragglers are devices that drop out of an FL scenario during an active communication round. Traditionally federated learning approaches [] []assume that all the devices are resource-abundant with similar hardware configurations. Such assumptions limit the application of federated learning to high-end devices. Resource-constrained devices such as an internet of things (IoT) device that has an ARM Cortex M7 MCU STM32F746 with 320 KB SRAM and a 1 MB flash memory cannot run a VGG-16 (of size 258 MB) or a Resnet (of size 22.7 MB). Hence a methodology a combined methodology that develops customized architectures with NAS along with a federated learning framework that effectively handles hardware heterogeneity is needed.

MobileNet and its variant such as Efficient-Net have been widely used for resource-constrained scenarios [8, 9]. However, to achieve high accuracy, the expansion ratio of the Inverted Residual blocks is very high, which leads to a high volume of featuremap size and increases demand for memory usage. Hence, the better architectures still take the successful design of Inverted Residual blocks with much smaller expansion ratio values. Model compression is proposed to reduce the model size of the given architecture. There are a lot of different model compression techniques, such as knowledge distillation, network pruning and quantization [10, 11]. In this paper, we primarily consider networking pruning techniques. The previous network techniques usually take the weight magnitude as the metric for pruning [12] but these methods lead to the structural irregularity and hurts the hardware performance; to address structural irregularity issue, some methods take the sum of the weights of filters as the metric [13] but this type of techniques involves high accuracy loss. To address these drawbacks, we are inspired by the differentiable NAS techniques DARTS [3]. DARTS was proposed by combining the weight of each operation as part of the network. After training, only the operation with the highest importance will be kept. In [3], the authors use the softmax of the essence to measure the significance of each candidate operation. Our differentiable search method follows the same strategy.

3. Methodology

3.1. Overview of our proposed methodology

In this section, we introduce the technical roadmap of our proposed solution. We first describe how we perform NAS in the cloud server; then, we discuss the model aggregation based on network morphism methodology.

As shown in Figure. 1, at the beginning of the FL process, each of the local clients will send their resource budgets (e.g., #FLOPs, #Params) to the cloud server. Then the cloud server will search the optimal architecture by the proposed differentiable NAS approach. The reason why we mitigate the search process to the cloud server is two-fold: (i). As known to us all, NAS is a relatively time-consuming and resource-intensive process; in FL, the clients usually cannot afford the overhead of NAS. (ii). Doing NAS in the cloud server can control the local model have some shared architecture that enables the aggregation stage. Specifically, the cloud server will generate the architecture by a top-down approach, i.e., firstly generate the model with the

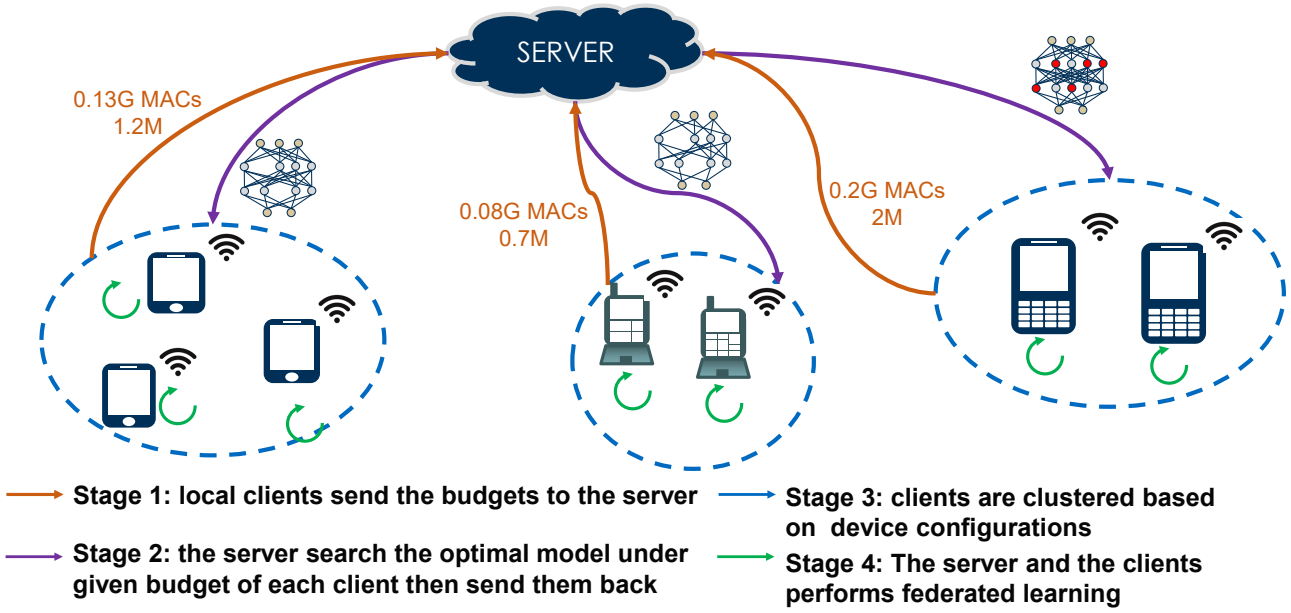

Figure 1. The four stages of the workflow of our proposed FL systems: (i) Given the hardware budgets of each client, local clients send the resource budgets of models to the server. (ii) The server searches the model with the highest performance on a reference dataset based on individual resource budgets for all client devices and updates them. (iii) The local clients with the same budgets are clustered into the same group. (iv) Local clients train their respective models, then updates the server with the local model; then, the cloud server aggregates the models with the same architecture.

biggest size, then shrink the model gradually until meeting the minimal budget of the local model. In this process, we will use an open dataset D as a reference to do the NAS, e.g., CIFAR10/100, ImageNet. After the cloud NAS process, the cloud server sends the searched models $\{s_i\}$ to each device according to their model budgets. In the next stage, the local clients train their respective models using the local data and send the trained model back to the cloud server.

3.2. DAC: Differentiable Network Architecture Compression

Problem Definition We formulate the NAS problem as followed:
Given a set of maximal MACs and number of parameters (#Params) constraint $\{M_i\}$, $\{P_i\}$, given a reference dataset D, search for a set of models $\{s_i\}$ from the search space S:

$$\underset{s_i \in S}{argmax}\ Acc(s_i\ on\ D)$$

$$subject\ to\ \ MACs < M_i,\ \#Params < P_i$$

Due to the wide usage for tiny devices, we take MobileNet-v2 as the base network. The main drawback of MobileNet-v2 is the high expansion ratio, which leads to the high volume of memory demand of hardware. To address this drawback, we propose a differentiable method to remove some useless channels. As shown in Fig. 2, inspired by the successful practice of Darts [3], we first assign a weight w_{ij} to the i^{th} input channel of the Depthwise Convolution layers of the j^{th} Inverted Residual blocks. Then each channel of featuremap will be multiplied by the corresponding weight w_{ij}.

To select the useful channels of the Depthwise Convolution layers, we use the entropy loss of the w_{ij} of Inverted Residual blocks. Given the property of entropy, the uniform distribution has maximal entropy values; the more nonuniform distribution has lower entropy values. Hence, minimizing entropy is making some w_{ij} approach to 1 and the rest approach to 0. Given the following analysis, we use the following equation as part of the loss function:

$$L_{entropy} = -\sum_j \sum_i \frac{e^{w_{ij}}}{\sum_i e^{w_{ij}}} log(\frac{e^{w_{ij}}}{\sum_i e^{w_{ij}}}) \qquad (1)$$

As shown in Equation. 1, to calculate the entropy of w_{ij}, we first use softmax function to convert w_{ij} into a standard probabilistic distribution. Then we use the sum of the entropy of each Inverted Residual blocks as the entropy loss.

In practice, we find that only considering the entropy loss tend to make w_{ij} approach to ∞ and $-\infty$. The ∞ and $-\infty$. However, to recognize the importance of each channel, we expect all the channel weights w_{ij} lie in the range $[0,\ 1]$. To this end, we involove the bound loss to constrain all the channel weights w_{ij} within $[0,\ 1]$:

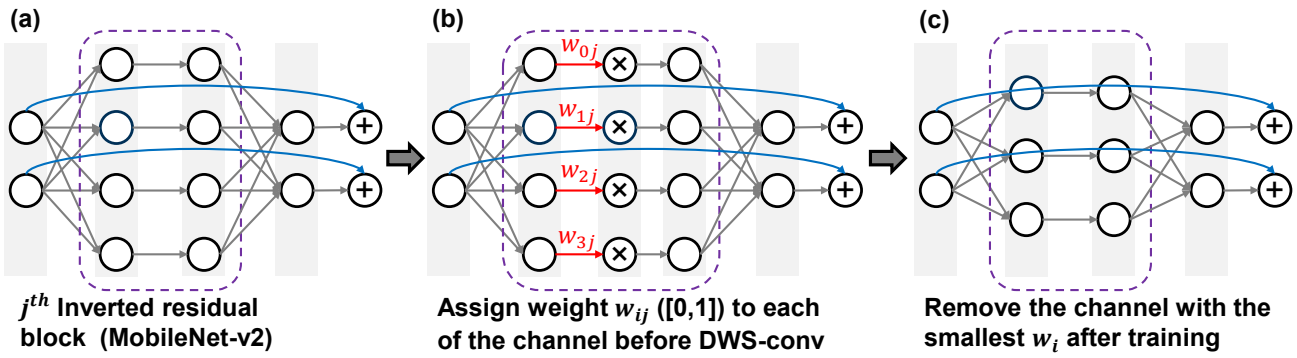

Figure 2. (a) The j^{th} standard Inverted Residual blocks of MobileNet-v2. (b) Modified Inverted Residual blocks by inserting a weight w_{ij} multiplication for each of the corresponding input channels of the Depthwise convolution layers. (c) Remove the useless channels with the smallest w_{ij} values from the Inverted Residual blocks.

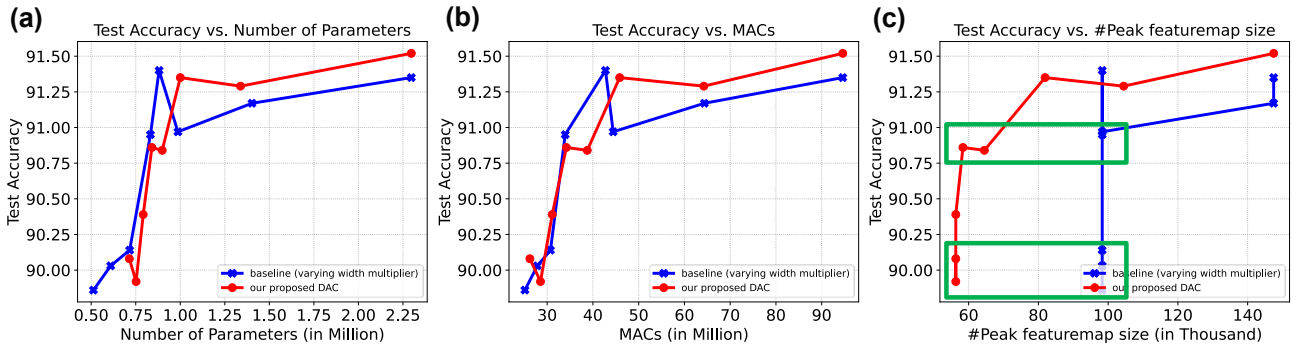

Figure 3. Comparison between our proposed DAC and the baseline method (varying with multiplier wm of standard MobileNet-v2 network). (a) Test accuracy vs. the number of parameters of both DAC and the baseline method. (b) Test accuracy vs. MACs of both DAC and the baseline method. (c) Test accuracy vs. peak intermediate featuremap size of both DAC and the baseline method.

$$L_{bound} = -\sum_{j}\sum_{i}\begin{cases} w_{ij} - 1, & if\ \ 1 < w_{ij} \\ -w_{ij}, & if\ \ 0 > w_{ij} \end{cases} \quad (2)$$

By combing the above two loss functions, we use the following loss function to do the training on the open reference dataset:

$$Loss = L_{CE} + \lambda_1 L_{entropy} + \lambda_2 L_{bound} \quad (3)$$

where L_{CE} is the cross entropy loss, which is used to account for the accuracy of the networks; $L_{entropy}$ is used to recognize the useless channels and L_{bound} is used to constrain all the channel weights w_{ij} within $[0,\ 1]$.

Ideally, the optimization objective should be the accuracy on the local data that each device maintains. However, due to the privacy preservation consideration, the cloud server cannot access the local data. Hence, currently, we plan to use the open dataset D as a reference dataset.

Algorithm 1: DAC algorithm

Result: Compressed Architecture: s_i
Input: MACs constraints M_i;
#Params constraints P_i;
Base network Net;
Reference Dataset D;
while *MACs* $\geq M_i$ *or #Params* $\geq P_i$ **do**
  Train Net on D;
  if $w_{ij} \leq 0.1$ **then**
    remove corresponding channel from Net;
  else
  end

4. Experimental Results

To evaluate our approach, we perform two separate sets of experiments, one with DAC and the other with a federated learning setup. The main goal for our experiments with DAC is to achieve a highly compressed model with all the characteristics of MobileNet-v2 except its high expansion ratio that demands a huge amount of memory. In the federated learning experiments, we use image classification as the target task, and we use the number of parameters and number of MACs to differentiate between a resource-constrained and a resource-abundant device. Here we assume that a model configuration is resource-constrained if it

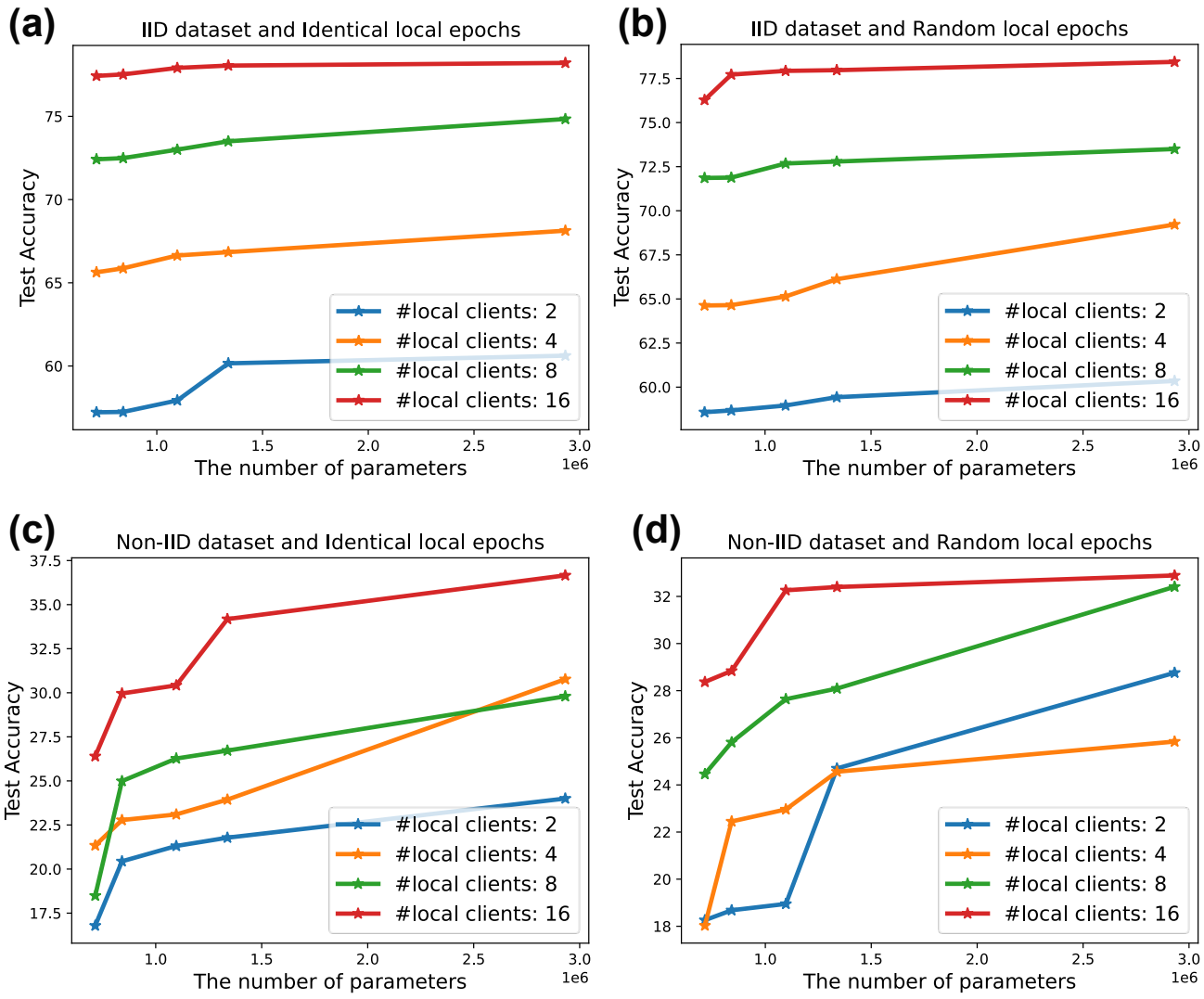

Figure 4. Test accuracy of the merged models vs. the number of parameters with different number of local clients. (a) IID dataset with the identical number of local epochs. (b) IID dataset with random sampled local epochs. (c) Non-IID dataset with the identical number of local epochs. (d) Non-IID dataset with random sampled local epochs.

Table 1. Experimental Setup of dataset and local training epochs

Experimental Setup	True	False
IID datset	all 10 classes & 300 samples/classes	randomly sampled 2 classes & 1000 samples/classes
Random local epochs	all clients train 5 epochs per round	uniformly sampled from $[3, 4, 5]$

has a lesser number of parameters as well as Macs and vice versa. We ran multiple configurations of the compressed model and presented our observations as plots.

4.1. DAC results

To demonstrate the efficiency of DAC, we use MobileNet-v2 with width multiplier $wm = 1$. As shown in Fig. 2, we first expand the standard MobileNet-v2 architecture by inserting a channel weight w_{ij} multiplication before the depth-wise convolutional layers of each block. We use $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$ as the coefficient of entropy loss and bound control loss. All the evaluations are done based on the CIFAR-10 dataset.

We take the standard MobileNet-v2 architecture with varying width multiplier wm as the baseline. We use the maximal intermediate featuremap size as the proxy of memory usage. As shown in Fig. 3(a,b), our searched model achieve the comparable test accuracy as the baseline method with similar computation workloads, such as MACs and the number of parameters. Moreover, as shown in Fig. 3(c),

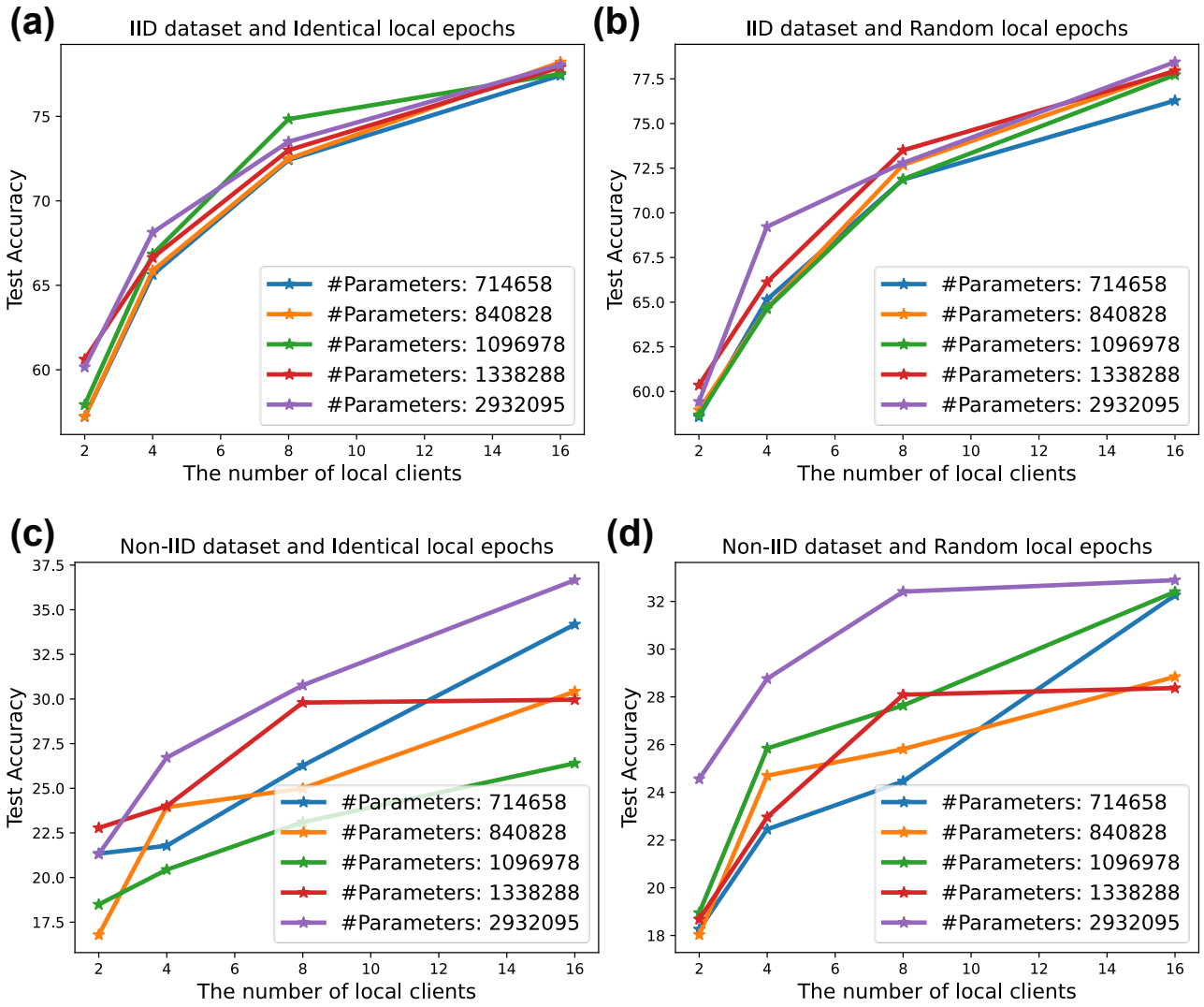

Figure 5. Test accuracy of the merged models vs. the number of local clients with different number of parameters. (a) IID dataset with the identical number of local epochs. (b) IID dataset with random sampled local epochs. (c) Non-IID dataset with the identical number of local epochs. (d) Non-IID dataset with random sampled local epochs.

our compressed architectures have reduced the peak memory usage half without accuracy loss.

4.2. Federated Learning results

dataset: We have conducted our experiments using the CIFAR-10 dataset for both IID and non-IID distributions. In the IID dataset, we have divided the training dataset uniformly across all the client devices for each communication rounds. In the non-IID dataset, we have followed the data splitting and distribution methodology mentioned in [6]. We split the training data among K distributed clients in an unbalanced fashion, where we sample data based on a Dirichlet distribution $p_c \sim Dir(\alpha)$ we allocate c classes to a local client, say k in every communication round. The test is performed using the global model located in the server using the test dataset that has 10,000 images.

Setup: We use the searched architectures by DAC with different MACs and number of parameters. We use SGD with learning 0.01 and momentum 0.5 as the optimizer for local training. We also consider the random variation of local hardware resources. We summarize the dataset and local training epochs in Table. 1. We evaluate the test accuracy of the merged models on the entire test set after 50 communication rounds. We vary the number of local clients within $\{2, 4, 8, 16\}$ and the number of parameters within $\{714658, 840828, 1096978, 1338288, 2932095\}$.

As shown in Fig. 4, bigger models tend to have higher test accuracy, especially for the NON-IID dataset. Also, the

local training epochs don't have a significant impact on the test accuracy. Hence, to reduce the training cost, we could use fewer local training epochs for resource-constrained devices or scenarios. As shown in Fig. 5, more local clients lead to higher test accuracy, especially for both IID and Non-IID datasets. Also, the local training epochs don't have a significant impact on the test accuracy.

5. Conclusion

In this paper, we have highlighted the problem of hardware heterogeneity in federated learning settings and have proposed a compression strategy known as Differentiable Architecture Compression (DAC). The proposed compression strategy can achieve a 2x compression without affecting the test accuracy of the overall inference. We have also proposed a clustering-based federated learning setup where clients with diverse hardware configurations can also be accommodated. The empirical evaluations of the FL scenario have also brought some interesting insights regarding performing FL using a complex neural network architecture.

Task Assignment

Guihong Li: write the code of DAC and Federated Learning and write the reports.
Sunny Sanyal: Conduct the experiments of Federated Learning and write the reports.

References

[1] Moore E. Ramage D. Hampson S. McMahan, H. B. and B. A. y. Arcas. Communication-efficient learning of deep networks from decentralized data. *In International Conference on Artificial Intelligence and Statistics*, 2017. 1

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020. 1, 2

[3] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1, 2, 3

[4] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2018. 2

[5] P. Kairouz, H. McMahan, and et al. B. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019. 2

[6] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Fednas: Federated deep learning via neural architecture search. *arXiv preprint arXiv:2004.08546*, 2020. 2, 6

[7] Mengwei Xu, Yuxin Zhao, Kaigui Bian, Gang Huang, Qiaozhu Mei, and Xuanzhe Liu. Federated neural architecture search. *arXiv preprint arXiv:2002.06352*, 2020. 2

[8] 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. 2

[9] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 2

[10] 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. 2

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[12] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015. 2

[13] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 2