DraftFed: a Draft-based Personalized Federated Learning Approach for Heterogeneous Convolutional Neural Networks

Yuying Liao, Le Ma, Bin Zhou, Xuechen Zhao, Feng Xie

***Abstract*—In conventional federated learning, each device is restricted to train a network model of a same structure. This greatly hinders the application of federated learning in edge devices and IoT scenarios where the data and devices are quite heterogeneous because of their different hardware equipment and communication networks. At the same time, most of the existing studies about federated learning of heterogeneous models are limited to horizontal heterogeneity which share a highly homogeneous vertical structure. Little work has been done on vertical heterogeneity such as models with different number of functional layers or different connection methods within them, not to mention the integrated heterogeneity scenarios. In DraftFed, a novel draft-based approach is proposed to implement personalized federated learning for integrated heterogeneous models. Unlike traditional federated learning in which the parameters/gradients are exchanged, DraftFed uses drafts as key knowledge to guide mutual learning of models, which makes it suitable for model structure personalization application scenarios..**

***Index Terms*—Personal Computing, Neural models, Distributed networks, Heterogeneous (hybrid) systemsArticle.**

I. Introduction

Federated learning (FL) is an emerging technology of artificial intelligence [1][2][3]. In common mobile scenarios, data is distributed across a large number of edge devices. However, privacy preservation and regulatory concerns may make it difficult to aggregate data to the cloud for centralized training. This creates the problem of data silos. FL allows multiple clients (or devices) to collaboratively train a shared global model without exposing data privacy. The central server coordinates multiple rounds of FL processes to obtain a final global model. FL has been applied in some scenarios (such as the next-word prediction of input methods [4]). It solves the problem of data aggregation and makes it possible to design and train some cross-institutional and cross-departmental deep learning models. In particular, for deep learning applications in IoT devices such as smart phones, FL has shown favorable performance and robustness [5]. Additionally, for clients who do not have enough private data to develop accurate local models, FL is an optional way to improve the performance of their models.

In most of the current assumption of FL configurations, global models are refined by averaging updated model parameters from devices. However, directly averaging model parameters is only possible if all models have a same structure, which could be a restrictive constraint in many real-world applications. An important step in advancing the application of FL in reality is to meet heterogeneous application requirements. We elaborate the need for model heterogeneity from system and strategy perspectives:

1. **Heterogeneous hardware resources:** Advances in hardware have made newly released mobile devices far superior to their predecessors in terms of both storage and computational power, such as the iPhone14, released in 2022, is equipped with A15 processor and 128G/256/512G storage capacity, while the iPhone7, released in 2019, is with A10 processor and 32G/128G storage capacity [6][7]. Furthermore, according to the research [8][9], people tend not to purchase new mobile devices frequently, leaving a serious hardware heterogeneity in federated learning involving mobile devices. Heterogeneous hardware resources usually refer to different computing powers, network connections or battery among multiple mobile edge devices [10][11]. As in the real mobile scenario, there are numerous cell phones with widely varying computing and storage capacities being used. When developing deep learning applications on mobile devices, it is necessary to take it into account. In addition, complex learning tasks generally rely on large and deep networks. To build a federated learning framework across multiple devices with heterogeneous hardware resources, it is either necessary to discard some users with under-performing devices or to collectively train a simpler global model that do not perform well. Therefore, model heterogeneity becomes a key requirement to enable the application of FL in heterogeneous system scenarios.
2. **Personal strategies:** Model security or other personal requirements may also generate the need for model heterogeneity. It is well known that in federated learning, the devices and the server perform multiple exchanges of complete parameters. This frequent communication makes the model parameters vulnerable to attacks, which may reveal their privacy [12] or be embedded with backdoors [13][14]. The data residing on each mobile device is statistically heterogeneous (i.e., non-IID data distribution). In the context of data heterogeneity, learning a single global model may not work well for all devices

Yuying Liao, Bin Zhou, Xuecheng Zhao and Feng Xie are with the Department of Computer Science, National University of Defense Technology, Changsha, China. E-mail: liaoyuying, binzhou, zhaoxuechen, xiefeng@nudt.edu.cn.

Le Ma is with the Xi'an Research Institute of High Technology, Xi'an, China. E-mail: maleaqmy@126.com.

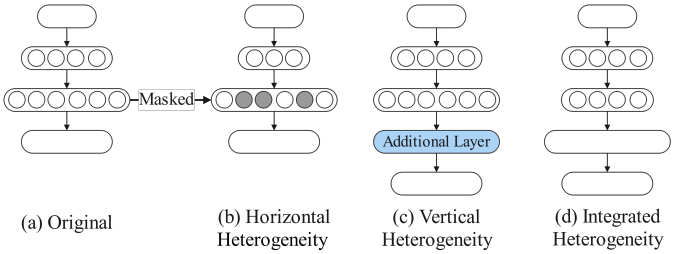


Fig. 1. Examples of different types of model heterogeneity. Model (b) is an example of horizontal heterogeneity with reference to model (a) The horizontal heterogeneity is manifested in the difference in the number of filters and the sparse parameter matrix after the mask operation; Model (c) is an example of vertical heterogeneity with an additional layer compared to model (a); Model (d) is an example of integrated heterogeneity of model (a), with different structures at both horizontal and vertical levels.

involved in FL[15].

These above application requirements enforce model heterogeneity in diverse ways. FL can be conducted only if the personal requirements are satisfied: jointly training and interacting with heterogeneous models, and that is what we are working on.

Currently, Federated Learning of heterogeneous models has gained much attention [15][16][17][18][19]. We summarize three types of model heterogeneity, as shown in Fig. 1, which, to our knowledge, covers all convolutional network model heterogeneity cases. Horizontal heterogeneity is mainly reflected in shape heterogeneity of parameter matrices caused by different numbers of filters or masked original model parameters, etc. Vertical heterogeneity refers to the addition or reduction of certain convolutional layers compared to the original model, while integrated heterogeneity is a combination of these two types of heterogeneity. Both horizontal and vertical heterogeneous models are regarded as the sparse or simplified versions of their original models. Besides, there are still implicit constraints to the model heterogeneity — the participants, no matter in horizontal or vertical heterogeneity scenarios, still have to negotiate and agree on a particular model before they can mask it. Furthermore, the integrated heterogeneity does not restrict the choice of model selection of clients to satisfy personalized needs, while being backward compatible with the newly joined devices, enabling lifelong learning.

Challenges and Our Solutions. The main challenge of our work is that different models are often integrated heterogeneous from one another, and simply aggregating the parameters of corresponding layers by mapping may lead to non-convergence results. Fortunately, we found that Personalized federated learning (PFL) has become a promising paradigm for many heterogeneity problems[20][21]. This paper presents a draft-based personalized federated learning approach to tackle the model heterogeneity problem. The idea is inspired by two basic observations from human knowledge acquisition process.

- In early stages, when we want to gain knowledge about a particular species, we observe itself by browsing its pictures, videos to extract, summarize and recognize its features or characteristics. The process is repeated and our cognition about it gradually improved.

- Apart from this, we also learn the summarized experiences from others. As long as the object is identical, the features observed by different individuals will share some common knowledge to some extent.

Similarly, in the model heterogeneity personalized federated learning context, we believe that although model structures are very different, they may possess common knowledge for same inputs, i.e., the feature extraction layer (e.g., Convolutional layers) generates similar feature maps, and the decision layer (e.g., Linear layers) gives similar confidence for classes. The outputs of these specific layers and models reflect sample-specific properties and can be understood as fuzzy images of the data. Therefore, we define these model outputs (intermediate or final) as *model drafts*.

Now, simulating human knowledge acquisition behavior, each PFL client is considered as a human-like intelligent agent, and the main idea of our approach is to learn the "common knowledge" of drafts when each client performs its own personalized local learning process. A new draft-based approach, DraftFed, is developed to address federated learning for integrated heterogeneous model where the model structure is so different that parameters cannot be aggregated directly. Fig. 2 shows the overview of DraftFed. In our method, draft learning process is independent of the local learning.The goal of draft learning is to allow devices to learn drafts extracted from other models, breaking the limitations of a unified model structure while protecting the private data. To obtain drafts, we set up a global data set separate from local training data and feed it into each integrated heterogeneous model to collect its drafts. Then the server aligns and aggregates the corresponding drafts and send back to devices for local draft learning. In local training process, devices perform routine model training based on local data. Each device learns both their local private data and the aggregated global drafts.

Summary of Contributions. In this work, we make the following contributions:

1. We observe that given a set of global data, integrated heterogeneous models may have similar drafts, which can be learned by other models.
2. Based on the observation, we propose an original draft learning approach, where devices learn from other devices indirectly by minimize the distance between local drafts to the aggregated global drafts.
3. We found that one single model draft is not sufficient for model collaborative learning, especially in the case of highly heterogeneous models. We present a joint draft learning whose objective is to solve a joint optimization problem on the combined drafts.
4. Finally, we design a general heterogeneous federated learning framework based on draft learning for integrated heterogeneous model scenarios, not only horizontal or vertica heterogeneity, but even more general heterogeneity scenarios across models. And compared to other related SOTA works, DraftFed has the highest inference accuracy in most of the cases among different global datasets.

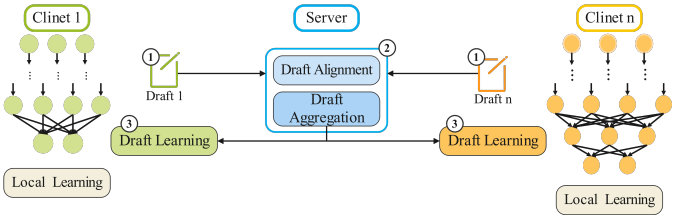


Fig. 2. An overview of DraftFed in model heterogeneity context. DraftFed includes two learning tasks: local learning and draft learning. In draft learning, (1) clients use the global data as input to calculate the drafts of local models and upload them to the parameter server; (2) the parameter server aggregates the drafts after alignment and returns the aggregated global draft to clients; (3) clients learn from the global draft and update their local models.

II. RELATED WORKS

A. Federated Learning

FL was design to carry out high-efficiency machine learning among multiple parties or devices under the premise of ensuring the security of terminal data. FL usually consists of a parameter server (or without [22]) and several local devices. Each device uses its own data to train a local model, and the parameter server integrates all local models into a global one. More specifically, in the federated learning training process, participating devices do not send their local data to a central server; instead, devices upload the updated local model parameters. In each round, the parameter server arbitrarily chooses several models from all devices to averagely aggregate and update a global model. The selected devices download the updated model from the parameter server, overwrite its local model parameters, and proceed to the next round of training; while the unselected devices continue to the next epoch on their original local models. Among them, there are different aggregation algorithms. FedAvg [4] is one of the classic FL aggregation algorithms. The parameters of each selected device update the global model equally. During the entire learning process, devices only communicate with the parameter server, and cannot obtain any information about the rest of the devices except for the global model that are jointly maintained, which guarantees the confidentiality of private data.

B. Personalized Federated Learning

However, conventional FL only roughly aggregates various models, and all devices obtain a same global model, ignoring that in the context of the IoT, each device is different, both in terms of hardware resources and local data distribution. Researchers summarize the challenges FL faced in practical applications as follows [20]:

- System heterogeneity such as storage, computing and communication capabilities;

- Data heterogeneity, like non-IID local data, such as feature distribution skew, label distribution skew and concept shift [22]).

- Model heterogeneity, where clients need different model structure according to their application scenarios.

To tackle these heterogeneity challenges, it is effective to perform personalization in device, data and model levels to attain well-trained personalized model for each device. Because of its promising application scenarios (such as IoT based personalized smart healthcare [23][24], smart home services and applications, and location - aware recommendation services [25][26][27]), personalized learning has recently attracted great attention [28][29]. Among them, system heterogeneity and data heterogeneity can be well addressed by modelheterogeneous- based approach[26], where developers can recommend models for different devices that best fit their hardware performance. And at the same time, pruning algorithms can be used to obtain different local models suitable for heterogeneous data.

C. FL of heterogeneous models

In conventional FL, all devices including parameter servers are restricted by a unified model structure. Moreover, in the process of interaction, complete model parameters are transmitted, which is not only unfavorable for applications in the IoT environment, but also increases the communication cost and the risk of privacy leakage.

At present, the FL of heterogeneous models have attracted the attention of some researchers. The uniform global model in the FL model has long been considered not to performed equally well to all the personalized devices, and the need for personalized models has gradually been proposed.

In terms of horizontal heterogeneity, as shown in Fig. 1 (b), there is the following work that implements federated learning across horizontally heterogeneous models. Some of the horizontal heterogeneous models have different number of filters or filter sizes between the corresponding convolutional layers, which directly affect the shapes of the parameter matrices; others perform mask operations on the original parameters due to pruning strategies, resulting in different sparse parameter matrices. In Hermes[15] and LotteryFL[17], each device can prune the global model based on its local data to obtain a local model that best fits its situation. HeteroFL[16] divides the domain of the global model so that each heterogeneous device model is responsible for updating partial global model. However, HeteroFL divides the global model parameter matrices from the origin. For the situation where the global model is far more complicated than local models, some neurons of the global model cannot be updated during the whole learning process, which affects the model performance. HFL[18], is a step forward than HeteroFL. The parameter server partitions the domain that each device is responsible for as needed, so that the global model parameters can be evenly updated. *Parameter/gradient - fusion* based methods have limited ability to adapt to heterogeneous hardware conditions such as storage, computation and communication, and FedProto[19] is proposed to tackle this challenge. In FedProto, the clients and server communicate the abstract class prototypes instead of gradients. Since the prototype is class-based, the aggregation of large amounts of complex data may result in information conflict or missing. Although averaged prototypes reduce individual privacy leakage to some extent, they still expose the statistical properties of local data. In summary, the above algorithm only discusses convolutional neural networks

with horizontal heterogeneity and do not have strategies on handling horizontal heterogeneity which cannot be extended to vertically or integrated heterogeneous scenarios.

In terms of vertical heterogeneity, as shown in Fig. 1 (c), different models have the same convolutional layers at the corresponding depths, which are generally in the shallow part of the models. Many approaches are intended to address the privacy issues during parameter transfer and the fact that the global model does not fit well to all local tasks. These methods are similar to the *Base + Personalization Layers* pattern. Yang [30] and Collins [31] proposed to reserve the functional layers, like decision layers that are more relevant to personalization, and conducted FL of heterogeneous models by only fusing the layers before decision layers. Furthermore, Arivazhagan [32] divided the model into basic and personalized submodels, breaking through the previous limitation that the personalized layers only contained decision - making layers, which explored the personalization performance of other functional layers. However, in integrated heterogeneous models, there are very few identical layers available as shared layers, so the *Base + Personalization Layers* approach is only applicable to vertically heterogeneous models under certain restrictions.

The integrated heterogeneity is a combination of horizontal and vertical heterogeneity, as shown in Fig. 1 (d), which not only has different parameter matrix shapes of corresponding depth layers, but also has quite different number of convolution layers. Second, different functional layers are sandwiched between two adjacent convolutional layers. Therefore, neither the above horizontal based methods nor the vertical based methods can solve the problem of integrated heterogeneous models. Linet [33] and Lin [34] proposed to use *knowledge distillation* [35][36][37][38] to solve the model heterogeneity problem. The federated learning approaches based on modeldistillation learn the soft labels of each model, making them incapable of learning the full capabilities of the data processing and susceptible to the domain to which the global data belongs. Compared to model-distillationbased approach FedMD, DraftFed also learns the ability to extract features from samples, which enhances the generalization ability on global data of models. The method we present can be applied to the above three model heterogeneity scenarios simultaneously, which covers all convolutional network model heterogeneity cases.

III. METHOD

A. Preliminaries

Before diving into the details of DraftFed, we first define the following notations used in our work. Suppose there are n devices participating in FL, and each device trains the model $\boldsymbol{M}$ on its local model $\boldsymbol{data}_{local}$ independently. The model $\boldsymbol{M}$ contains l_c convolutional layers and l_{fc} fully-connected layers.

Fig. 3 illustrates a brief concept of DraftFed in an integrated model heterogeneity setting, where local models vary not only from horizontal but also vertical dimensions. Each device(e.g. clinet 1, 2, ..., n) has two learning tasks: local learning and draft learning. Devices use local data for local learning (denoted in green and yellow in Fig. 3), referring to solving the original optimization problem based on local images and their labels. And the global data, which may be completely independent of local data, or even sampled from a different data source, is used in draft learning (denoted in blue in Fig. 3) to compute the specified model drafts. Each model needs to send its model drafts to the server preparing for the subsequent draft learning. The server aligns the received drafts and aggregates them, then sends them back to devices. Essentially, draft learning is an optimization problem, where devices try to find the sets of local model parameters to minimize the distances between the local drafts and the aggregated global drafts. Throughout the learning process, local learning and draft leaning are executed cyclically in turns until the learning budget is run out or a specified accuracy is reached.

Local Learning As with conventional federated learning, each device optimizes its objective function locally:

$$Minimize \quad \mathcal{L}_{local}(\mathcal{F}(\boldsymbol{\omega}; \boldsymbol{x}_{local}), \boldsymbol{y}_{local}), \tag{1}$$

where $\boldsymbol{\omega}$ is the parameters of the model $\boldsymbol{M}$, $\boldsymbol{x}_{local}$ and $\boldsymbol{y}_{local}$ are training images and labels of local data $\boldsymbol{data}_{local}$, $\mathcal{F}(\boldsymbol{\omega}; \boldsymbol{x}_{local})$ is the output of the model $\boldsymbol{M}$, and $\mathcal{L}_{local}(\cdot)$ is the loss function e.g., a Cross-Entropy loss.

To improve the tolerance to heterogeneity caused by personalized requirements, we propose a general framework for personalized FL of integrated heterogeneous (both horizontally and vertically) models based on draft learning. We expect to illustrate our series of technical contributions by answering the following two questions: ***What ' s drafts and How models learn from them***.

B. Draft Definition

Global data is the input to the draft calculation in draft learning, which is distinguished from local privacy data used for local learning. Global data is collected by the server and it can be completely irrelevant to the local data. In the draft computation process, only the *forward propagation* is involved (without *back propagation*), therefore, the unlabeled data can also be used as global data. As shown in Fig. 3, each local data is sampled from a data set, e.g., the CIFAR-10 dataset, while the global data is from another one, e.g., Fashion MNIST dataset.

Draft Given a set of global data inputs $\boldsymbol{data}_{local}$, we define a draft $\mathcal{F}(\boldsymbol{\omega}[:L]; \boldsymbol{x}_{local})$ to represent the output of layer $\#L$ of model $\boldsymbol{M}$.

Joint Drafts We found that one single model draft is not sufficient for draft learning, especially in the case of highly heterogeneous models. Therefore, we consider the joint draft combined by $[\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{D}_3]$ as composite metric for draft learning.

$$\begin{aligned} \boldsymbol{D}_1 &= \mathcal{F}(\boldsymbol{\omega}[:L_1]; \boldsymbol{x}_{local}), \\ \boldsymbol{D}_2 &= \mathcal{F}(\boldsymbol{\omega}[:L_{l_c}]; \boldsymbol{x}_{local}), \\ \boldsymbol{D}_3 &= \mathcal{F}(\boldsymbol{\omega}[:L_{l_c+l_{fc}}]; \boldsymbol{x}_{local}), \end{aligned} \tag{2}$$

where $\boldsymbol{D}_1$, $\boldsymbol{D}_2$ and $\boldsymbol{D}_3$ denote the output of layer $\#1$, layer $\#l_c$ and $\#l_c + l_{fc}$, respectively. In essence, $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are the initial and final feature maps of the model $\boldsymbol{M}$, while $\boldsymbol{D}_3$ is the embedding decision vector based on global data $\boldsymbol{data}_{local}$.

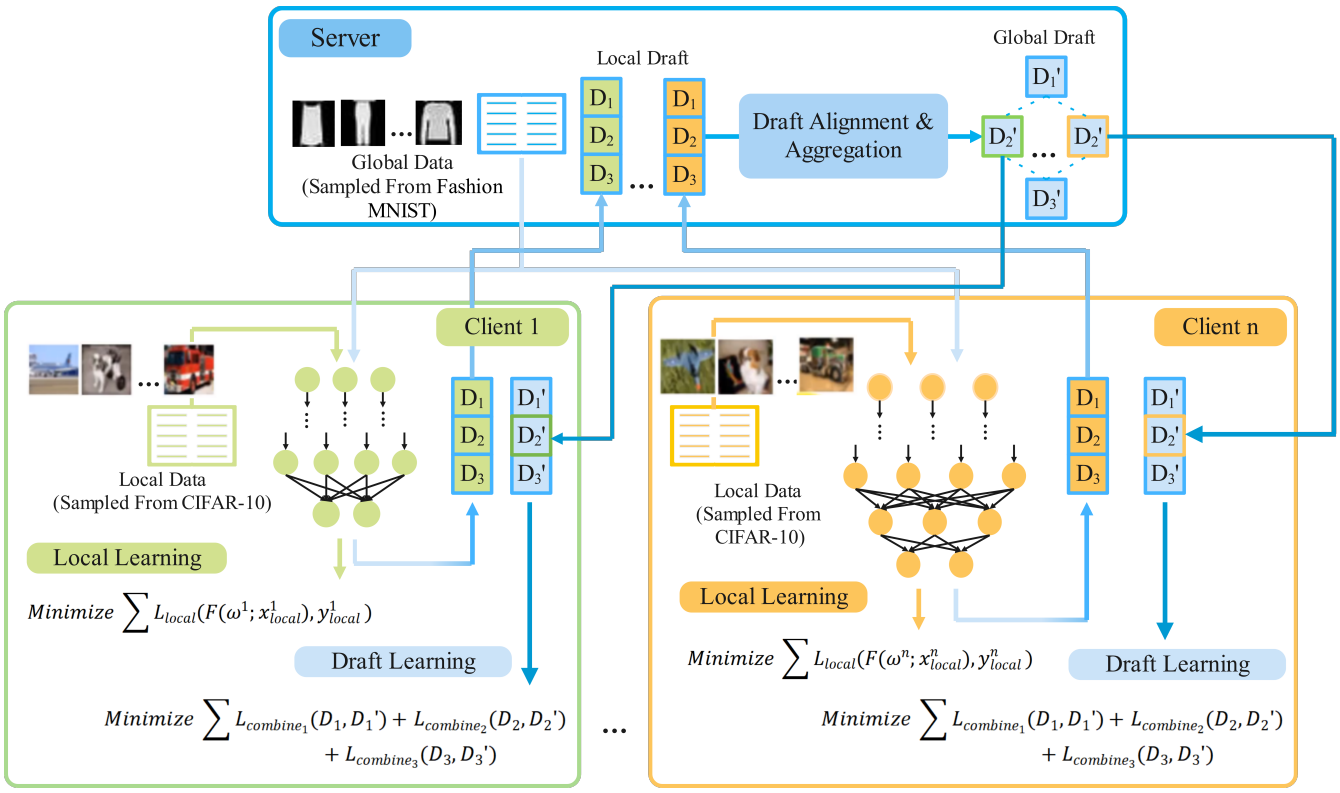


Fig. 3. A Detailed architecture of DraftFed in the integrated heterogeneous setting.

If these target convolutional layers are closely followed by Batch Normalization layers, the drafts indicate the output of the *Batch Normalization* layers. This is because the *Batch Normalization* layers can filter the personalized information brought by heterogeneity in the feature maps, which is not easily digested by other models.

C. Draft Learning

We then answer the ***How models learn from drafts*** question.

Optimization Objective of Draft Learning The objective of DraftFed is to solve a joint optimization problem on the combined drafts mentioned above. The objective of federated draft learning across heterogeneous clients can be formulated as,

$$Minimize \quad \lambda_1 \sum_{i=1}^{n} loss_1^i + \lambda_2 \sum_{i=1}^{n} loss_2^i + \lambda_3 \sum_{i=1}^{n} loss_3^i \quad (3)$$

where λ_1, λ_2 and λ_3 are the weights of the three loss functions in the combined optimization objective. In our experiments, $\lambda_1 = \lambda_2 = \lambda_3 = 1$. In particular, the loss function $loss_1^i$ and $loss_3^i$ are defined as follow:

$$\begin{aligned} loss_1^i &= \left\| \boldsymbol{D}_1^i, \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{D}_1^j \right\|, \\ loss_3^i &= \mathcal{L}\left(\boldsymbol{D}_3^i, \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{D}_3^j \right), \end{aligned} \quad (4)$$

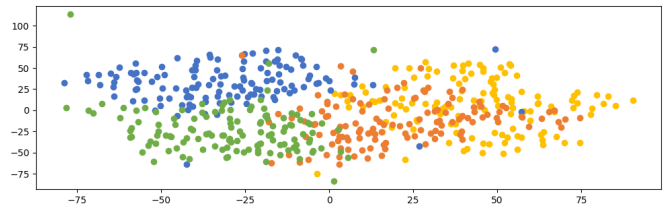
where $\|\cdot\|$ is the distance loss function, and in our experiments, MSELoss is used. $\mathcal{L}(\cdot)$ is the loss function e.g., a Cross-Entropy loss. Here, we tentatively use $\frac{1}{n}\sum_{j=1}^{n} \boldsymbol{D}_1^j$ to denote the average aggregation of $\boldsymbol{D}_1$, but the shapes of the draft matrices may not be the same. We leave the details of shape-mismatched draft aggregation to a later section.

Draft Alignment of $\boldsymbol{D}_2$ For integrated heterogeneous convolutional networks, the output layers of $\boldsymbol{D}_2$ may be at different vertical positions. As shown in Fig. 4(a), the t-SNE visualization of $\boldsymbol{D}_2$ from different vertical positions are more separable compared to the drafts in Fig. 4(b), which are from the same vertical positions. Aggregation of $\boldsymbol{D}_2$ from different models may introduces noise.

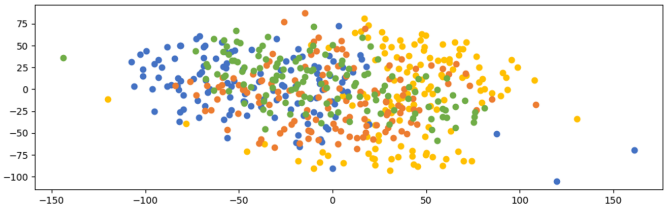
And Fig. 4(b) presents the closer distance within the drafts of the same vertical position, which are reasonable to be aggregated. To alleviate the draft fusion conflicts, $\boldsymbol{D}_2$ needs to be aligned before aggregation.

As shown in Fig. 3, local models have different number of convolutional layers. For a given model, its $\boldsymbol{D}_2$ is fused with the drafts of layers that are at the same vertical positions in other models. Therefore, the target $\boldsymbol{D}_2$ (after aggregation) is model-specific. The server computes and maintains the shadow target $\boldsymbol{D}_2$ for each model. It may be noticed that, the devices need to provide not only their $\boldsymbol{D}_2$, but also drafts of other model-specified layers. Regarding the position of models, we only consider the order at the initialization phase, not after adding shortcuts.

Given the number of convolution layers $\{l_c^1, l_c^2, ..., l_c^n\}$ of n model, device i needs to provide the following drafts:



(a) The $\boldsymbol{D}_2$s of DenseNet121/161/169/201 models which are at different vertical positions.



(b) The $\boldsymbol{D}_2$ of DenseNet121 and the aligned drafts of Dense- Net161/169/201, which are at the same vertical positions.

Fig. 4. t-SNE visualization of the drafts produced by integrated heterogeneous convolutional networks. We consider 20 DenseNetx (DenseNet121, DenseNet161, DenseNet169 and DenseNet201) mixed. Each color indicates a draft of a model-specific layer.

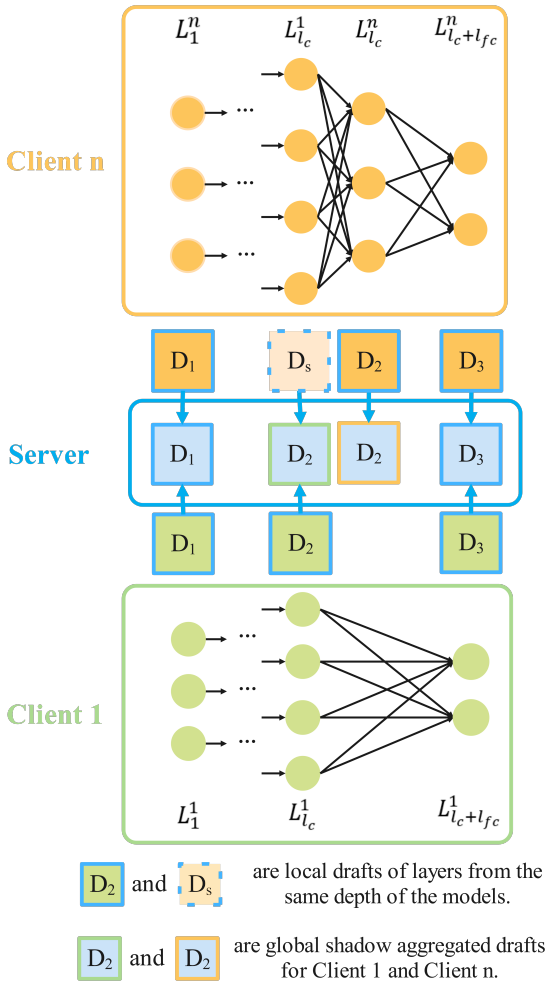


Fig. 5. Examples on $\boldsymbol{D}_2$ alignment and the computation of shadow target $\boldsymbol{D}_2$.

$$\mathcal{D}^i = \left\{ \boldsymbol{S}_1^i, \boldsymbol{S}_2^i, \boldsymbol{S}_3^i, \mathcal{F}\left(\boldsymbol{\omega}^i \left[: L_{l_c^j}\right]; \boldsymbol{x}_{global}^i\right), \ldots \right\}, \quad \text{if } l_c^j < l_c^i, \forall j \in [1, n], \tag{5}$$

where $\mathcal{F}\left(\boldsymbol{\omega}^i \left[: L_{l_c^j}\right]; \boldsymbol{x}_{global}^i\right)$ denotes the output draft of the layer $L_{l_c^j}$ (specified by model $\boldsymbol{M}_j$) of model $\boldsymbol{M}_i$ with global data $\boldsymbol{x}_{global}^i$ as input, when and only when $l_c^j < l_c^i$. The server constitutes n shadow $\boldsymbol{D}_2$, just as shown in fig. 5. Finally, $loss_2^i$ can be formulated as

$$loss_2^i = \left\| \boldsymbol{D}_2^i, \frac{\boldsymbol{D}_2^i + \sum_i^{n_s} \mathcal{F}\left(\boldsymbol{\omega}^i \left[: L_{l_c^j}\right]; \boldsymbol{x}_{global}^i\right)}{n_s + 1} \right\| \quad \text{if } l_c^j < l_c^i, \forall j \in [1, n], \tag{6}$$

where n_s denotes the number of models with less convolution layers than model $\boldsymbol{M}_i$. And $\|\cdot\|$ is also refers to the distance loss function, which is MSELoss.

Draft Aggregation In addition, for feature-map-based drafts, the shapes of the draft matrices may not be the same. Let' s take the example of $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, which are four-dimensional matrices.

Given the $\{\boldsymbol{D}_1^1, \boldsymbol{D}_1^2, ..., \boldsymbol{D}_1^n\}$ of n models, the shape of matrix $\boldsymbol{D}_1^i$ is $[global\_bs, c\_out_1^i, w_1^i, h_1^i]$. For $\boldsymbol{D}_1^j$, whose $[w_1^j, h_1^j] \neq [w_1^i, h_1^i]$, we make the widths and heights uniform by using up/down sampling. For out-channel dimensions of $\boldsymbol{D}_1^j$, after we align the slice matrix of this dimension, the modified slice matrix $\boldsymbol{D}_{c\_out}^j$ can be calculated as:

$$\boldsymbol{D}_{c\_out}^{j\ \ i} = \begin{cases} [e_1^j, ..., e_{c\_out_1^i}^j], \boldsymbol{if} & c\_out_1^i \leq c\_out_1^j \\ [e_1^j, ..., e_{c\_out_1^j}^j, 0...0], \boldsymbol{if} & c\_out_1^i > c\_out_1^j \end{cases}, \quad \exists i \neq j. \tag{7}$$

where $\boldsymbol{D}_{c\_out}^{j\ \ i}$ denotes the modified slice matrix $\boldsymbol{D}_{c\_out}^j$ with $c\_out_1^i$ as the reference, and e_k^j is the kth element of $\boldsymbol{D}_{c\_out}^j$. Therefore, the $loss_1^i$ of (4) is adjusted to

$$\boldsymbol{loss}_1^i = \left\| \boldsymbol{D}_1^i, \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{D}_1^{j^i} \right\|. \tag{8}$$

Likewise, $\frac{1}{n}\sum_{j=1}^n \boldsymbol{D}_1^{j^i}$ in (8) is the modified matrices with $\boldsymbol{D}_1^i$ as the reference using (7).

As for $\boldsymbol{D}_3$, which is a two-dimensional matrix, it is simpler to aggregate $\boldsymbol{D}_3$ than $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$.The two dimensions of $\boldsymbol{D}_3$ denote *global batchsize* and *number of classes* respectively, which, in general, are the same among heterogeneous models. Therefore, $\boldsymbol{D}_3$ can be aggregated directly.

D. DraftFed

We propose a general heterogeneous federated learning framework that uses joint drafts as the key knowledge to learn from other models.

The details of our approach are specified in Algorithm 1. In DraftFed, the devices coordinate with the parameter

server for multiple iterations. In each iteration, first of all, the devices compute the drafts of their local models using the global data as input (*DraftComputing()*) in parallel to prepare for the subsequent draft learning. Then, the server is mainly responsible for collecting the drafts of each model and the draft aggregation, including $\boldsymbol{D}_1$, $\boldsymbol{D}_3$, and model-specific $\boldsymbol{D}_2$, which is the function *DraftAggregation()* in Algorithm 1. After the global draft is obtained, the device performs *DraftLearning()* (on global draft) and *LocalLearning()* (on private local data) successively.

Drafts are the output of the specified layers, which can be obtained by reserving channels when defining the local models. The outputs of the hidden layers can also be accessed by using the APIs provided by deep learning frameworks. We use the hook function in *PyTorch: register_forward_hook()*, with the ability to obtain or even change the intermediate feature maps and gradients during the *forward propagation*. In both *DraftComputing()* and *DraftLearning()* function, the hook function is used to hook the model drafts.

In *DraftComputing()*, the devices need to provide not only their own joint $[\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{D}_3]$, but also drafts of other model-specified layers, as described in the **Joint Drafts** section. In *DraftAggregation()*, the parameter server aggregates the model drafts uploaded by devices after alignment, as described in **Draft Alignment** and **Draft Aggregation** sections, respectively. The output of *DraftAggregation()* is the global draft. In *DraftLearning()*, after receiving the global drafts from the server, devices calculate the current model drafts aiming to minimize the distance between the local drafts and the global draft. The joint loss function is calculated using (3), (4) and (6). The detailed explanation of the draft learning process is shown in **Optimization Objective of Draft Learning** above.

Algorithm 1 DraftFed

Input: local models $M = \{M_1, M_2, ...M_n\}$, local data $Data = \{Data_1, Data_2, ...Data_n\}$, global data set $Data_g$, global epoch E.
Output: Trained local models $M^E = \{M_1^E, M_2^E, ...M_n^E\}$.

Initialize D_2 target layer $TL = \{L_{lc_1}, L_{lc_2}, ...L_{lc_n}\}$;
for *each global epoch* $e = 1, 2, \ldots, E$ **do**
  for *each node* $i \in N_c$ *in parallel* **do**
    $[D^e_{i\_1}, D^e_{i\_2}, D^e_{i\_3}, D^e_{i\_supplement}]$
    $\leftarrow \boldsymbol{DraftComputing}(Data_p, L_{lc_{supplement}})$;
    //Run on node i
  end
  $[D^e_1, D^e_2, D^e_3] \leftarrow \boldsymbol{DraftAggregation}(TL, D)$;
  for *each node* $i \in N_c$ *in parallel* **do**
    $\boldsymbol{LocalTraining}(Data_i)$; //Run on node i
    $\boldsymbol{DraftLearning}([D^e_1, D^e_{i\_2}, D^e_3], Data_p)$; //Run on node i
  end
end
DraftComputing$(Data_p, L_{lc_{supplement}})$**:**
  $D^e_{i\_1} \leftarrow \boldsymbol{Hook}(Data_p, L_1)$;
  $D^e_{i\_2} \leftarrow \boldsymbol{Hook}(Data_p, L_{lc})$;
  $D^e_{i\_3} \leftarrow \boldsymbol{Hook}(Data_p, L_{lc+lfc})$;
  for *each model-specific target layer* $L^j_{lc_{supplement}}$ **do**
    $D^{e\_j}_{i\_supplement} \leftarrow \boldsymbol{Hook}(Data_p, L^j_{lc_{supplement}})$
  end
end .;
DraftAggregation(TL, R)**:**
  $D^e_1 \leftarrow \boldsymbol{Avg}(D^e_{i\_1})$;
  $D^e_3 \leftarrow \boldsymbol{Avg}(D^e_{i\_3})$;
  for *each draft* $D^e_{i\_2}$ **do**
    draft_list = [ ];
    for *each draft* $D^{e\_j}_{supplement}$ *in* D **do**
      if $D^{e\_j}_{supplement}.lc$ == $D^e_{i\_2}.lc$ **then**
        draft_list.append($D^{e\_j}_{supplement}$);
      end
    end
    $D^e_{i\_2} \leftarrow sum(draft\_list)/len(draft\_list)$
  end
end .;
DraftLearning$([D_1, D_2, D_3], Data_p)$**:**
  $D_{1\_i} \leftarrow \boldsymbol{Hook}(Data_p, L_1)$;
  $D_{2\_i} \leftarrow \boldsymbol{Hook}(Data_p, L_{lc\_i})$;
  $D_{3\_i} \leftarrow \boldsymbol{Hook}(Data_p, L_{lc\_i+lfc\_i})$;
  Minimize $\lambda_1 \boldsymbol{loss_1}(D_{1\_i}, D_1) + \lambda_2 \boldsymbol{loss_1}(D_{2\_i}, D_2) + \lambda_3 \boldsymbol{loss_3}(D_{3\_i}, D_3)$
end .;

IV. EXPERIMENT AND ANALYSIS

A. Implementations Details

We implement the experiments in a pseudo-distributed setting on 6 × Nvidia RTX 2080 Ti GPU. We considered 1 parameter server and 20 devices.

1) Datasets

CIFAR10 [39] is an image classification data set with 10 labeled classes. It contains 50,000 training samples and 10,000 testing samples.

CIFAR100 [39] is like CIFAR10, except that it has 100 classes, each containing 600 images - 500 training images and 100 testing images.

FashionMNIST[40] is a dataset of 70,000 grayscale images of costume data, with 60,000 training examples and 10,000 testing examples

SVHN [41] is a real-world image data set that can be regarded as similar to the MNIST (e.g., the images are of small cropped digits). SVHN is obtained from the house number in the Google Street View image. It contains 73257 training samples and 26032 testing samples.

When the above data sets are used as the training data sets, we shuffle the training set and divide it evenly into 20 subsets, one of which is taken randomly by each client; when they are used as the source of global data, we select $|\boldsymbol{G}|$ images with $label \in [0, 9]$ as the global data sets.

2) Model Architectures

ResNetx [42] is a residual learning framework, which can effectively solve the problem of gradient disappearance/gradient explosion that often occurs during deep neural network training. It uses residual blocks to improve the performance of very deep neural networks. At present, ResNet has been widely used in many computer vision tasks.

TABLE I
TOTAL NUMBER OF PARAMETERS/DRAFTS TO BE COMMUNICATED IN HERMES AND DRAFTFED UNDER DIFFERENT COMPRESSION RATIO.

Method	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Hermes	5,510,648	4,485,860	3,902,490	3,185,168	2,883,313	2,848,330	2,411,919	2,318,753
DraftFed	2,228,352							

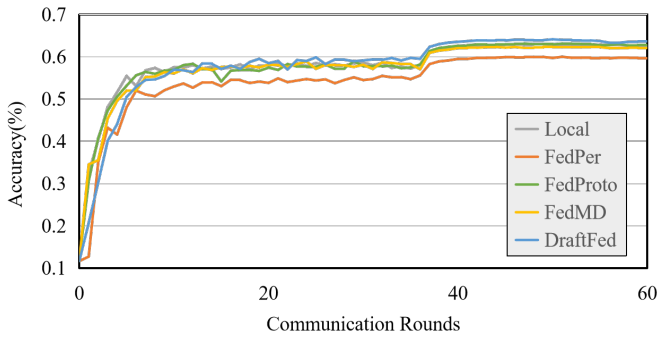


Fig. 6. Convergence Rate comparison (mixed ResNetx on local data sampled from CIFAR-10).

TABLE II
COMPARISON OF FL METHODS ON MIXED HETEROGENEOUS MODEL SCENARIOS OF TWO MODEL FAMILIES.

Model	Method	Accuracy
ResNet50, ResNet101 and ResNet152 Mixed	Local	62.76
	FedProto	63.27
	FedMD	63.44
	FedPer	59.51
	DraftFed	64.16
DenseNet121, DenseNet161, DenseNet169 and DenseNet201 Mixed	Local	68.55
	FedProto	
	FedMD	68.65
	FedPer	
	DraftFed	69.67

TABLE III
TOTAL NUMBER OF PARAMETERS/DRAFTS TO BE COMMUNICATED IN FEDAVG AND DRAFTFED.

Method	ResNetx	DenseNetx
FedAvg	762,925,852	323,815,780
DraftFed	60,817,408	2,228,352

TABLE IV
THE AVERAGE TRAINING COSTS OF LOCAL LEARNING, DRAFT PREPARATIONS AND DRAFT LEARNING (MIXED DENSENETX ON LOCAL DATA SAMPLED FROM CIFAR-10 AND GLOBAL DATA, WITH $|\boldsymbol{G}| = 512$, SAMPLED FROM FASHIONMNIST, SVHN AND CIFAR-100 RESPECTIVELY).

Local Training	Draft Computation	Draft Aggregation	Draft Learning
40.32s	2.42s	0.09s	2.33s

We adopted three of the models from the ResNet family - ResNet50, ResNet101 and ResNet152. In our settings, the local models of device $\#0 - \#7$ are ResNet50, Device $\#8 - \#15$'s are ResNet101, and the rest are ResNet152.

DenseNetx [43] is a network architecture where each layer is directly connected to every other layer in a feed-forward fashion (within each dense block). For each layer, the feature maps of all preceding layers are treated as separate inputs whereas its own feature maps are passed on as inputs to all subsequent layers.

We adopted DenseNet121, DenseNet161, DenseNet169 and DenseNet201 four types of models, where each of these models was chosen equally by the 20 devices.

3) Baselines

To make fair comparison, we compare DraftFed with four baselines:

Local is the no-federated-learning method where an individual model is trained for each client without any communication with others and the server.

Hermes[15] is a communication and inference-efficient FL framework under data heterogeneity. In Hermes, each device can learn a personalized and structured sparse deep neural network, which can run efficiently on devices.

FedPer [32] is a common *Base + Personalization Layers* approach commonly used in heterogeneous federated learning scenarios. The main idea of FerPer is to divide the model into two parts, the base layers and the personalization layers, and the server is only responsible for aggregating the model parameters of the base layers instead of the complete model.

FedMD [34] is a *knowledge-distillation* based method reduce the restrictions on the model structure. In contrast to exchanging model parameters, models in FedMD exchange the output of the fully-connected layers of the models to enable the collaborative learning.

FedProto [19] is a *prototype* FL framework where only prototypes are transmitted between the server and clients. The prototype is class-based which reduces the leakage of private local data to a certain extent.

B. Results

1) Convergence Rate

First, we examine the convergence rate of DraftFed and other methods. Fig. 6 compares the inference accuracy curves w.r.t communication rounds between DraftFed (blue line) and other methods. In our experimental settings, the convergence rates of each method in Fig. 6 are similar, but our method achieves the best accuracy 64.16% compared to the other four methods.

2) Inference Accuracy

We tested the inference accuracy of our method and other SOTA works. Our experimental scenarios are divided into horizontal heterogeneity and integrated heterogeneity.

For horizontal heterogeneity, since it is not our major concern, we just select Hermes, one of the SOTA works, as our baseline without comparing other horizontal heterogeneity algorithms. The local data and global data are sampled from the same data sets and $|\boldsymbol{G}| = 512$. We prune the local models of the devices using the *Lottery Ticket hypothesis* [44], which is a model compression algorithm that seeks a sparser but essentially equivalent inference accuracy of the sub-network structure compared to the original model. Compression ratio is the sparsity of model, i.e., the number of non-zero elements of the model after pruning/ the model before pruning. We set pruning *percent* = 0.8, which means 20% of weights are pruned. When the current model compression rate is less than or equal to the pre-defined compression rate, the models stop pruning (model masks no longer change).

The average inference accuracy over all clients is reported in Fig. 7 and the corresponding communication costs are shown in Table 4. Because Hermes is parameter-fusion-based, it can achieve high accuracy when the model structures are similar. However, as shown in Fig. 7 and Table. 4, the overall accuracy of our method is just 0.08%-0.73% lower than Hermes on the FashionMNIST dataset, while reducing communication costs by 3.9%-59.6%; even on the CIFAR-10 dataset, DraftFed achieves higher accuracy than Hermes in most cases. In summary, compared to Hermes, DraftFed not only has equivalent inference accuracy while effectively reducing the pressure of storage and communication, but is applicable to a wider range of model heterogeneity scenarios.

For integrated heterogeneity, we tested the inference accuracy in two sets of experiments for the mixed heterogeneous model scenarios. The local data and global data are both sampled from CIFAR-10 and $|\boldsymbol{G}| = 512$.

The average inference accuracy over all clients is reported in Table 1. In the DenseNetx mixed case, the accuracy of FedPer and FedProto are missing. DenseNetx has a different structure of the model parameters from the first convolutional layer that aren't available for base layers selection, therefore we are unable to implement FerPer in this case. And for FedProto, the prototypes of DenseNetx models have different shapes that cannot be aggregated directly. In table 1, it can be seen that DraftFed achieves the highest accuracy.

3) Communication Efficiency

In DraftFed, only drafts are exchanged between the server and clients. In general, the sizes of the drafts are usually much smaller than that of the model parameters. As shown in Table 2, in the case of ResNetx, the communication volume of DraftFed is 8% of that of FedAvg (17% in the case of DenseNetx).

4) Training Costs

The training costs mainly consists of the time requiredfor local learning and draft learning, including the prep work such as draft computation and aggregation.As shown in Table 3, the total time for draft preparations (draft computation and aggregation) and draft learning is only 12% of that for the local learning.

TABLE V
AVERAGE INFERENCE ACCURACY OF FIG. 9 OVER THE THREE GLOBAL DATA SETS.

$loss_1$	$loss_2$	$loss_3$
68.49%	68.86%	68.84%
$loss_1 + loss_2$	$loss_1 + loss_3$	$loss_2 + loss_3$
68.86%	68.88%	68.91%
$loss_1 + loss_2 + loss_3$		
68.92%		

5) Privacy Preserving

The proposed DraftFed brings benefits to FL in terms of privacy preserving. First, in DraftFed, only the drafts are required to exchange rather than model parameters between the server and the clients. The drafts $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are the feature maps of inputs, while $\boldsymbol{D}_3$ is the embedding decision vector. Second, we propose to learn the drafts with global data is used as input, rather than local data of devices, further preventing the leakage of private data. Moreover, DraftFed can be integrated with various privacy-preserving techniques to further enhance the reliability of the system.

C. Research Questions

In addition to what we have described about DraftFed, there may be several questions that exist. We answer the following research questions that may be pop in your head:

(RQ1) Does the selection of public set affect the average accuracy?

To answer this question, we tested three cases where the public sets are sampled from CIFAR-100, SVHN and FashionMNIST datasets, respectively, and the local training sets are CIFAR-10 with equally split over clients. The local models are ResNetx mixed. We compare DraftFed with other SOTA methods under the same settings. Since in FedProto, the prototype is based on the local data of the devices, in order to evaluate its performance on different public data sets, we slightly adjust its code to make sure the computation of the prototype is based on the global data. It can be seen in Fig. 8 that DraftFed achieves the highest accuracy in most cases, ensuring that DraftFed can accommodate different public data options.

(RQ2) How different loss combinations affect model accuracy?

To explore the impact of different loss combinations on the inference accuracy, we conduct the ablation experiments. Fig. 9 shows the inference accuracy for 7 combinations of losses with different global data, where $loss_1 + loss_2 + loss_3$ is the joint loss chosen by as the setting for the previous DraftFed experiments. Table 4 is the average inference accuracy of fig. 9 over the three global data sets, from which it can be seen that DraftFed achieves the highest accuracy.

Overall, the triple-loss combination achieved the highest accuracy rate, then is the dual-loss combinations, and the single-losses have the lowest accuracy. Among the three single-loss cases, $loss_3$ is more suitable for the case where the global data is homogeneous with the local data (FashionMNIST: 68.42%,

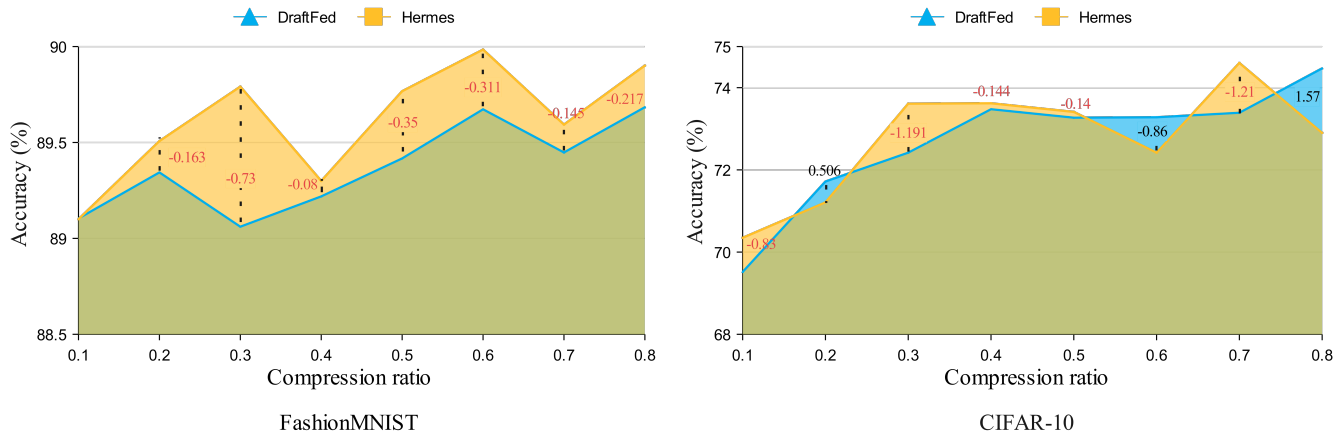


Fig. 7. Average inference accuracy of Hermes and DraftFed under different compression ratio (mixed DenseNetx).

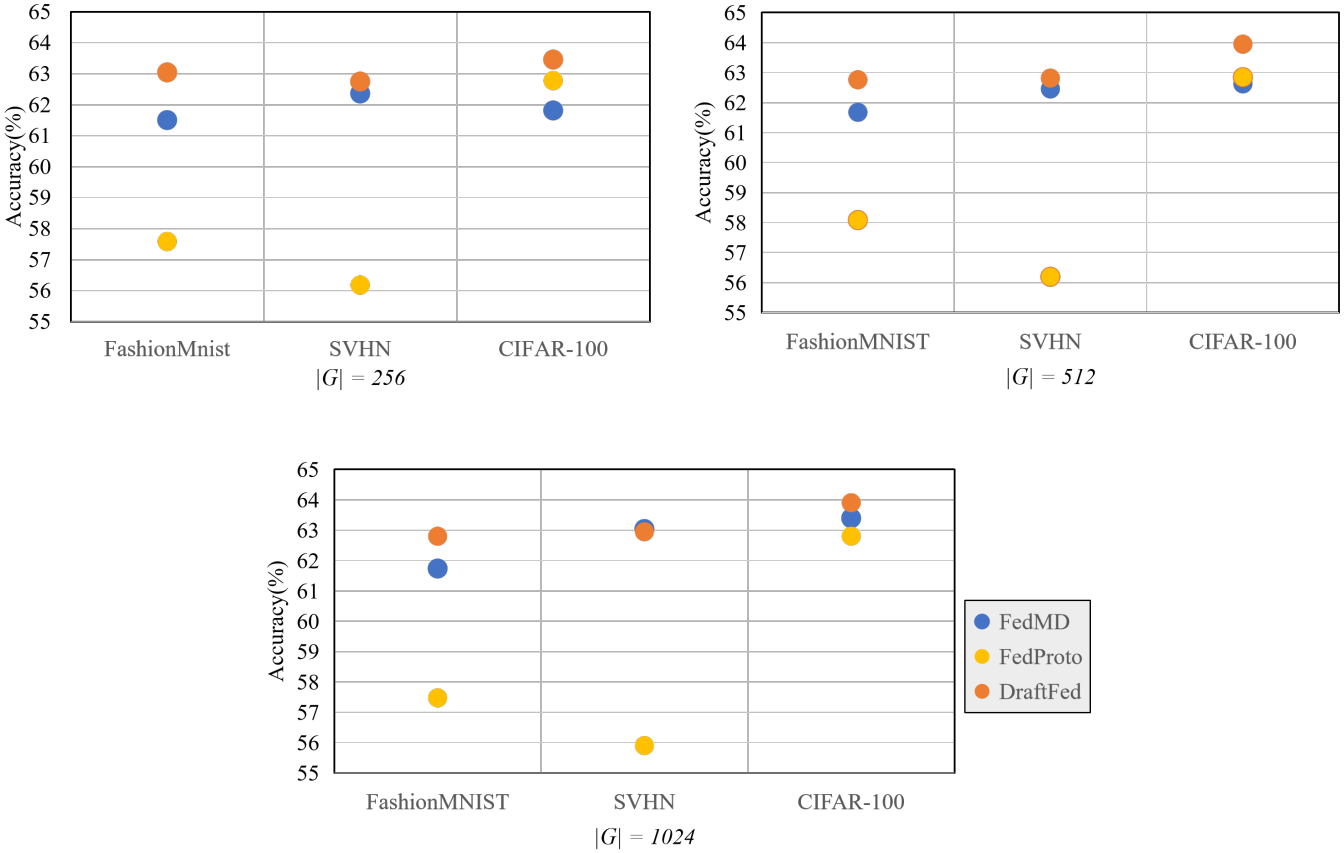


Fig. 8. Average inference accuracy of FedMD, FedProto and DraftFed on different global data with varying number of samples (mixed ResNetx on local data sampled from CIFAR-10).

SVHN: 68.83%, CIFAR-100: 69.27%), while $loss_2$ has a relatively balanced performance (FashionMNIST: 68.49%, SVHN: 69.01%, CIFAR-100: 69.07%).

The drafts are considered as fuzzy images of the data extracted by models. Different drafts reflect different characteristics of the data. The $\boldsymbol{D}_1$, $\boldsymbol{D}_2$ and $\boldsymbol{D}_3$ in DraftFed are the outputs of the first convolutional layer, the last convolutional layer and the entire model respectively. In the task of image classification, $\boldsymbol{D}_1$ represents the shallow image features of the data, such as straight edge, color, texture, while $\boldsymbol{D}_2$ represents the more detailed abstract features. $\boldsymbol{D}_3$ is the soft label of the data, which reflects the statistical properties of the data. $Loss_3$ provides a more direct direction for model learning, while it is more susceptible to the source of global data than $loss_3$. Compared to $loss_2$ and $loss_3$, $loss_1$ is more like a regular term to prevent over-fitting.

(RQ3) How to choose the $\boldsymbol{\lambda_1}$, $\boldsymbol{\lambda_2}$ and $\boldsymbol{\lambda_3}$?

To answer this question, we perform a grid searching for

TABLE VI
AVERAGE INFERENCE ACCURACY OF DIFFERENT $[\lambda_1, \lambda_2, \lambda_3]$ COMBINATIONS OVER THE THREE GLOBAL DATA SETS (MIXED DENSENETX ON LOCAL DATA SAMPLED FROM CIFAR-10). THE BEST RESULTS ARE IN **BOLD.**

	0.1-1-1	0.1-0.1-1	0.1-1-0.1	0.01-1-1	0.01-1-0.1	0.01-1-0.01	0.01-0.1-1	0.01-0.01-1
FasionMNIST	68.53%	68.51%	68.58%	68.5%	68.55%	**68.67%**	68.43%	68.62%
SVHN	68.81%	68.86%	68.86%	**68.93%**	68.92%	68.7%	68.77%	68.86%
CIFAR-100	69.32%	69.52%	**69.59%**	69.46%	69.41%	69.25%	69.33%	69.40%
Average	68.89%	68.96%	**69.01%**	68.96%	68.96%	68.87%	68.84%	68.96%

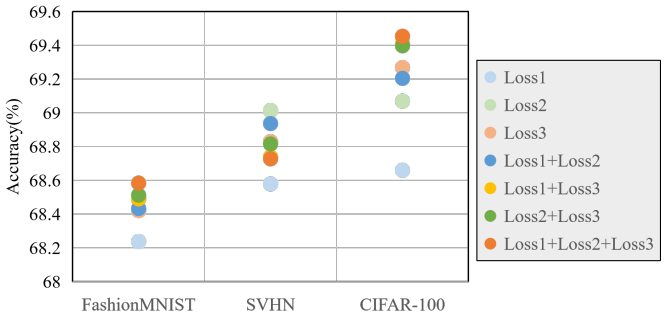


Fig. 9. Average inference accuracy for seven combinations of losses with different global data with number of samples $|\boldsymbol{G}| = 512$ (mixed DenseNetx on local data sampled from CIFAR-10).

the values of λ_1, λ_2 and λ_3 in $[1, 0.1, 0.01]$. Based on the observation in **RQ2**, we set $\lambda_1 = min(\lambda_1, \lambda_2, \lambda_3)$. After removing the redundant cases of isometric scaling, the remaining results are shown in Table 5. Among them, $\lambda_1 = 0.1$, $\lambda_2 = 1$, $\lambda_3 = 0.1$ achieves the highest average accuracy. We could not find a set of $[\lambda_1, \lambda_2, \lambda_3]$ achieving the best inference accuracy in the three sets of global data at the same time, but one commonality is noticed among the three cases achieving the highest accuracy: $\lambda_2 = 1$. Therefore, for realistic applications in other scenarios, $\lambda_1 = 0.1$, $\lambda_2 = 1$, $\lambda_3 = 0.1$ can be considered when setting the hyper-parameters in the initial stage. And we highly recommend to set $\lambda_2 = 1$.

V. CONCLUSION

Traditional federated learning requires all local nodes to train network models with the same structure, which is difficult to satisfy in IoT application scenarios. Secondly, previous studies have shown, even the proponents of federated learning also admit, that the transmission of complete model information in traditional federated learning has the risk of privacy leakage. At the same time, most of the existing studies about federated learning of heterogeneous models are limited to horizontal heterogeneity which share a highly homogeneous vertical structure. Little work has been done on vertical heterogeneity such as models with different number of functional layers or different connection orders, not to mention the integrated heterogeneity scenarios. In order to tackle the challenges mentioned above, we propose a novel draft learning method to implement personalized federated learning for integrated heterogeneous models.

DraftFed can effectively address the federated learning for integrated heterogeneous models. In DraftFed, only the drafts are transmitted between the server and devices. We set up a global data set as the input of draft learning. Each device learns both the local training samples and the aggregated drafts.

In addition, in future work, we will explore how to aggregate drafts with extremely mismatched shapes while not losing information behind them. And we need to further consider the extreme non-IID scenarios.

ACKNOWLEDGMENTS

This work is supported by the Key R\&D Program of Guangdong Province No.2019B010136003 and the National Natural Science Foundation of China No. 62172428, 61732004, 61732022.

REFERENCES

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016. [Online]. Available: http://arxiv.org/abs/1610.05492

[2] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7

[3] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, A. Singh and X. J. Zhu, Eds., vol. 54. PMLR, 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[5] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 12:1–12:19, 2019. [Online]. Available: https://doi.org/10.1145/3298981

[6] Apple, "Compare iphone models," Website, 2023, https://www.apple.com/.

[7] NanoReview, "Smartphone processors ranking," Website, 2023, https://nanoreview.net/en/soc-list/rating.

[8] gwi, "The smartphone marketplace," Website, 2019, https://www.gwi.com/hubfs/Downloads/The_Smartphone_Marketplace.pdf.

[9] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. V. Gool, "AI benchmark: Running deep neural networks on android smartphones," in *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part V*, ser. Lecture Notes in Computer Science, L. Leal-Taixé and S. Roth, Eds., vol. 11133. Springer, 2018, pp. 288–314. [Online]. Available: https://doi.org/10.1007/978-3-030-11021-5_19

[10] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, Eds. ACM / IW3C2, 2021, pp. 935–946. [Online]. Available: https://doi.org/10.1145/3442381.3449851

[11] C. Yang, M. Xu, Q. Wang, Z. Chen, K. Huang, Y. Ma, K. Bian, G. Huang, Y. Liu, X. Jin, and X. Liu, "Flash: Heterogeneity-aware federated learning at scale," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2022.

[12] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 131:1–131:36, 2022. [Online]. Available: https://doi.org/10.1145/3460427

[13] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 2020, pp. 2938–2948. [Online]. Available: http://proceedings.mlr.press/v108/bagdasaryan20a.html

[14] C. Zhao, Y. Wen, S. Li, F. Liu, and D. Meng, "Federatedreverse: A detection and defense method against backdoor attacks in federated learning," in *IH&MMSec '21: ACM Workshop on Information Hiding and Multimedia Security, Virtual Event, Belgium, June, 22-25, 2021*, D. Borghys, P. Bas, L. Verdoliva, T. Pevný, B. Li, and J. Newman, Eds. ACM, 2021, pp. 51–62. [Online]. Available: https://doi.org/10.1145/3437880.3460403

[15] A. Li, J. Sun, P. Li, Y. Pu, H. Li, and Y. Chen, "Hermes: an efficient federated learning framework for heterogeneous mobile clients," in *ACM MobiCom '21: The 27th Annual International Conference on Mobile Computing and Networking, New Orleans, Louisiana, USA, October 25-29, 2021*. ACM, 2021, pp. 420–437. [Online]. Available: https://doi.org/10.1145/3447993.3483278

[16] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=TNkPBBYFkXg

[17] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "Lotteryfl: Empower edge intelligence with personalized and communication-efficient federated learning," in *6th IEEE/ACM Symposium on Edge Computing, SEC 2021, San Jose, CA, USA, December 14-17, 2021*. IEEE, 2021, pp. 68–79. [Online]. Available: https://doi.org/10.1145/3453142.3492909

[18] X. Lu, Y. Liao, C. Liu, P. Liò, and P. Hui, "Heterogeneous model fusion federated learning mechanism based on model mapping," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 6058–6068, 2022. [Online]. Available: https://doi.org/10.1109/JIOT.2021.3110908

[19] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "Fedproto: Federated prototype learning across heterogeneous clients," 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*. AAAI Press, 2022, pp. 8432–8440. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20819

[20] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent iot applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020. [Online]. Available: https://doi.org/10.1109/OJCS.2020.2993259

[21] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/24389bfe4fe2eba8bf9aa9203a44cdad-Abstract.html

[22] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1-2, pp. 1–210, 2021. [Online]. Available: https://doi.org/10.1561/2200000083

[23] W. Y. B. Lim, S. Garg, Z. Xiong, D. Niyato, C. Leung, C. Miao, and M. Guizani, "Dynamic contract design for federated learning in smart healthcare applications," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 16 853–16 862, 2021. [Online]. Available: https://doi.org/10.1109/JIOT.2020.3033806

[24] P. Pace, G. Aloi, R. Gravina, G. Caliciuri, G. Fortino, and A. Liotta, "An edge-based architecture to support efficient applications for healthcare industry 4.0," *IEEE Trans. Ind. Informatics*, vol. 15, no. 1, pp. 481–489, 2019. [Online]. Available: https://doi.org/10.1109/TII.2018.2843169

[25] Z. Yan, Y. Z. Yi, J. Zhang, N. Zhao, Y. Ren, J. Wan, and J. Yu, "Federated learning model training method based on data features perception aggregation," in *94th IEEE Vehicular Technology Conference, VTC Fall 2021, Norman, OK, USA, September 27-30, 2021*. IEEE, 2021, pp. 1–7. [Online]. Available: https://doi.org/10.1109/VTC2021-Fall52928.2021.9625291

[26] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *CoRR*, vol. abs/1802.07876, 2018. [Online]. Available: http://arxiv.org/abs/1802.07876

[27] L. Wang, Y. Wang, Y. Bai, P. Liu, and X. Li, "POI recommendation with federated learning and privacy preserving in cross domain recommendation," in *2021 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2021, Vancouver, BC, Canada, May 10-13, 2021*. IEEE, 2021, pp. 1–6. [Online]. Available: https://doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484510

[28] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized collaborative learning of personalized models over networks," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, A. Singh and X. J. Zhu, Eds., vol. 54. PMLR, 2017, pp. 509–517. [Online]. Available: http://proceedings.mlr.press/v54/vanhaesebrouck17a.html

[29] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," in *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, ser. Proceedings of Machine Learning Research, A. J. Storkey and F. Pérez-Cruz, Eds., vol. 84. PMLR, 2018, pp. 473–481. [Online]. Available: http://proceedings.mlr.press/v84/bellet18a.html

[30] Q. Yang, J. Zhang, W. Hao, G. P. Spell, and L. Carin, "FLOP: federated learning on medical datasets using partial networks," in *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, F. Zhu, B. C. Ooi, and C. Miao, Eds. ACM, 2021, pp. 3845–3853. [Online]. Available: https://doi.org/10.1145/3447548.3467185

[31] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 2089–2099. [Online]. Available: http://proceedings.mlr.press/v139/collins21a.html

[32] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *CoRR*, vol. abs/1912.00818, 2019. [Online]. Available: http://arxiv.org/abs/1912.00818

[33] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/18df51b97ccd68128e994804f3eccc87-Abstract.html

[34] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *CoRR*, vol. abs/1910.03581, 2019. [Online]. Available: http://arxiv.org/abs/1910.03581

[35] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[36] I. Bistritz, A. J. Mann, and N. Bambos, "Distributed distillation for on-device learning," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing*

Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/fef6f971605336724b5e6c0c12dc2534-Abstract.html

[37] X. Gong, A. Sharma, S. Karanam, Z. Wu, T. Chen, D. S. Doermann, and A. Innanje, "Ensemble attention distillation for privacy-preserving federated learning," in *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 15 056–15 066. [Online]. Available: https://doi.org/10.1109/ICCV48922.2021.01480

[38] C. Wu, F. Wu, R. Liu, L. Lyu, Y. Huang, and X. Xie, "Fedkd: Communication efficient federated learning via knowledge distillation," *CoRR*, vol. abs/2108.13323, 2021. [Online]. Available: https://arxiv.org/abs/2108.13323

[39] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Handbook of Systemic Autoimmune Diseases*, vol. 1, no. 4, 2009.

[40] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[41] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[42] T. Q. Tran and J. Sakuma, "Seasonal-adjustment based feature selection method for predicting epidemic with large-scale search engine logs," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds. ACM, 2019, pp. 2857–2866. [Online]. Available: https://doi.org/10.1145/3292500.3330766

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[44] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 2261–2269. [Online]. Available: https://doi.org/10.1109/CVPR.2017.243