electronics

Article

Training Vision Transformers in Federated Learning with Limited Edge-Device Resources

Jiang Tao [1], Zhen Gao [2,*] and Zhaohui Guo [1]

[1] School of Microelectronics, Tianjin University, Tianjin 300072, China
[2] School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China
* Correspondence: zgao@tju.edu.cn

Abstract: Vision transformers (ViTs) demonstrate exceptional performance in numerous computer vision tasks owing to their self-attention modules. Despite improved network performance, transformers frequently require significant computational resources. The increasing need for data privacy has encouraged the development of federated learning (FL). Traditional FL places a computing burden on edge devices. However, ViTs cannot be directly applied through FL on resource-constrained edge devices. To utilize the powerful ViT structure, we reformulated FL as a federated knowledge distillation training algorithm called FedVKD. FedVKD uses an alternating minimization strategy to train small convolutional neural networks on edge nodes and periodically transfers their knowledge to a large server-side transformer encoder via knowledge distillation. FedVKD affords the benefits of reduced edge-computing load and improved performance for vision tasks, while preserving FedGKT-like asynchronous training. We used four datasets and their non-IID variations to test the proposed FedVKD. When utilizing a larger dataset, FedVKD achieved higher accuracy than FedGKT and FedAvg.

Keywords: federated learning; vision transformer; split learning; knowledge distillation

check for updates

Citation: Tao, J.; Gao, Z.; Guo, Z. Training Vision Transformers in Federated Learning with Limited Edge-Device Resources. *Electronics* **2022**, *11*, 2638. https://doi.org/10.3390/electronics11172638

Academic Editor: Kah Phooi Seng

Received: 25 June 2022
Accepted: 29 July 2022
Published: 23 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Since the concept of a vision transformer (ViT) was proposed by Dosovitskiy et al. [1], ViT has demonstrated impressive performance in numerous machine vision tasks. Self-attention-based transformers are widely used in natural language processing (NLP) [2]. A multi-head self-attention transformer structure is used in these models which can attend flexibly to a sequence of visual patches to encode contextual information. Owing to the scalability and computational efficiency of transformers, it has become possible to train models of unprecedented size with over 100B parameters [1,3,4]. There is currently no indication of a saturating level of performance as the models and datasets grow. Although transformers have exerted significant influence in the computer vision (CV) and multi-model fields, they still face high storage and processing resource demands when deployed on edge devices, such as smartphones and AIoT devices. Moreover, an increasing number of people are concerned to address issues of data privacy [5,6]. As seen by the recent spike in interest in federated learning (FL) [7,8], the demand for edge-based training is growing. FL is a distributed learning paradigm which allows several edge devices to work together to build a global model without relying on a centralized dataset. When a single organization or user lacks sufficient or relevant data because of privacy legislation, FL can help improve the accuracy of a model. Many tech giants' FL services have been deployed commercially (e.g., WeBank's FATE and Google's TensorFlow Federated, TFF). Google has used FL to increase the accuracy of item rankings and language models on Android smartphones. When data centralization is undesirable or impossible, FL offers a viable alternative as an edge training framework. Traditional FL assumes that the client has sufficient computational power with GPUs to train a computation-intensive AI model. However, this assumption can be difficult to meet in practice.

Many model training technologies have been developed to train computation-intensive AI models in FL on large CNNs, such as split learning (SL) and knowledge distillation (KD). Model-parallelism-based SL [9,10] attempts to overcome edge computational limitations. SL partitions a large model and offloads a larger portion of the neural architecture to the cloud while maintaining a smaller portion of the local neural architecture. However, SL faces a complicated straggler problem because a single mini-batch iteration requires multiple rounds of communication between the server and edge devices. FedGKT [11] uses an alternating minimization strategy to train small CNNs on edge nodes and periodically transfers their knowledge to a large server-side CNN via KD [12,13].

To train ViTs on resource-constrained edge devices, many new transformer architectures and transformer pruning technologies have been proposed; however, smaller transformers often perform less well than larger ones. KD paves the way for improving the performance of ViTs. DeiT [14] introduced hard-label distillation and used a CNN network as the teacher. It utilized a distillation token to ensure that the student learned from the teacher through attention to it. Sun et al. proposed patient-knowledge distillation (Patient-KD) [15] for the BERT model. Patient-KD adopts a patient-learning mechanism: a student network learns from multiple intermediate layers of the teacher to extract internal knowledge. For multilayer KD, the patient learner has the benefit of distilling rich information through the deep structure of the teacher network. However, these methods do not consider the distributed characteristics of FL and can only transfer knowledge within a single dataset. Moreover, some hybrid models that combine CNNs and transformers aim to overcome the disadvantages of transformers that lack inductive biases. The authors of [16] minimally altered the early visual processing of ViTs by replacing their patchify stem with a standard convolutional stem consisting of approximately only five convolutions. Their results showed that a slight modification in early visual processing was beneficial for improving the accuracy of the final model.

On the one hand, traditional FL algorithms do not consider the features of transformers and thus cannot be directly used for ViTs; on the other hand, the existing transformer-based training technologies do not consider the distributed privacy of FL.Therefore, we propose a federated KD training algorithm (FedVKD) to train ViTs in FL with limited edge device resources. The contributions of this study can be summarized as follows.

(1) FedVKD places low-computing-demand convolutional networks at edge devices, while keeping high-computing-demand ViT at the cloud, as in SL. Our new FL paradigm transfers the computing pressure from edge devices to the cloud. In addition, our global model is a hybrid model consisting of a convolutional stem and transformer blocks. Therefore, our global model combines the local information perception ability of convolutional networks and the long-distance information capture capability of transformers into a single framework.

(2) FedVKD utilizes bidirectional KD to transfer knowledge between edge devices and the cloud. Considering the features of transformers, we make targeted improvements to bidirectional KD. We use a hard distillation strategy for transformer blocks on the server side, while using traditional KD for the CNN network on the edge side. Owing to the inherent inductive biases of CNNs (e.g., translation equivariance and locality), the CNN model on edge devices generalizes well when trained on insufficient amounts of local data. In the process of distilling knowledge from edge devices, the transformer learns from multiple CNNs as a student. Therefore, our hybrid model can obtain competitive results when trained on mid-sized datasets, such as ImageNet. Benefiting from bidirectional KD and a strong server-side model, the performance of the edge-side model is also improved.

(3) We performed suitable experiments to verify the performance of the FedVKD framework and demonstrated a new method for training ViTs in FL with limited edge-device resources.

The rest of this paper is organized as follows: Essential concepts and background are introduced in Section 2. The system architecture is presented in Section 3. The experiments

performed are detailed in Section 4. Section 5 discusses and concludes the paper. Section 6 considers limitations of the study.

2. Preliminaries

2.1. Vision Transformer (ViT)

By providing scalable training, transformers have transformed NLP. Transformers use multiheaded self-attention, which is more general than convolution, and perform global information processing. The authors of [17] observed that single-head self-attention is a type of non-local method. A transformer encoder was used in [1] to classify images with minor vision-specific changes. First, the input images were separated into a series of patches in a ViT; subsequently, the transformer network was used to extract image features for visual recognition. By evenly splitting an input image, p patches can be obtained. If the image resolution is 224×224 and the patch size is set to 16×16, the image is divided into 196 patches. A linear layer flattens and projects these patches onto patch embeddings. We obtained $n \times p$ patch embeddings, $X \in \mathbb{R}^{n \times p \times c}$, for a batch of n images, where c was the embedding dimension. For relationship modeling and feature aggregation, these patch embeddings were input into the transformer network. The structure of the ViT comprises position encoding, a multi-head self-attention (MSA) block, and a multi-layer perceptron (MLP) block. The information flow is expressed as follows.

$$X \leftarrow MSA(LN(X)) + X \tag{1}$$

$$X \leftarrow MLP(LN(X)) + X \tag{2}$$

The input is added with the position encoding before the first MSA, and LN is the layer-normalization layer. The MSA mechanism can be formulated as follows.

$$MSA(X) = FC_{\text{out}}\left(\text{Attention}\left(FC_q(X), FC_k(X), FC_v(X)\right)\right) \tag{3}$$

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{4}$$

When the hidden dimension of MLP is 4c by default, the floating-point operations per second (FLOPs) of MSA and MLP are $4nc^2 + 2n^2c$ and $8nc^2$, respectively. The ViT with L blocks has $L(12nc^2 + 2n^2c)$ FLOPs.

The computational cost of ViTs is always high because of the large values of n and d (typically in the several hundreds). To process a 224×224 input image, the base version of the ViT requires 17.6B FLOPs. For practical deployment on edge devices, a lightweight ViT with fewer FLOPs is preferred. In contrast, smaller ViTs usually perform more poorly than larger ones.

2.2. Knowledge Distillation

KD was first defined in [12] and generalized by Hinton et al. [13]. Model compression and knowledge transfer are two fields in which KD is widely employed. A student model was trained to mimic a teacher model or an ensemble of models for model compression. Although several types of KD are defined depending on the objective, one common feature of any KD is represented by its S-T (student-teacher) framework, wherein the model delivering information is referred to as the teacher and the model gaining knowledge is referred to as the student. Irrespective of the structural differences between the teacher and student networks, ref. [13] demonstrated transfer of knowledge from the teacher model to the student model by minimizing the difference between the logits (inputs to the final softmax) provided by the teacher model and those produced by the student model.

However, the output of the softmax function on the teacher's logits indicated that the correct class had a very high probability, with all other class probabilities being extremely close to zero. The function does not provide much information beyond the ground-truth labels already provided in the dataset in this case.To address this issue, the authors of [13]

proposed the concept of softmax temperature, which may be used to soften the target. The class probability q_i of an objective is determined using the logits z_i from a network as follows:

$$q_i = \frac{\exp\left(\frac{Z_i}{T}\right)}{\sum_j \exp\left(\frac{Z_j}{T}\right)} \tag{5}$$

where T is the temperature parameter. We obtain the standard softmax function when $T = 1$. The probability distribution generated by the softmax function becomes softer as this increases, thereby offering more information about the classes that the teacher finds to be more similar to the predicted class. Knowledge is transferred to the distilled model in the simplest form by training it on a transfer set and using a soft target distribution for each case in the transfer set. This is generated by employing a ponderous model with a high temperature in its softmax function. When training the distilled model, the same high temperature is employed. However, the model uses a temperature of 1 after it has been trained.

In [13], The Kullback—Leibler divergence between the softmax of the student model and the softmax of the teacher model was minimized via soft distillation. The goal of distillation is as follows.

$$\ell_{\text{global}} = (1-\lambda)\ell_{CE}(\psi(Z_s), y) + \lambda T^2 \ell_{KL}\left(\psi\left(\frac{Z_s}{T}\right), \psi\left(\frac{Z_t}{T}\right)\right) \tag{6}$$

where λ is the coefficient balancing the Kullback—Leibler divergence loss (ℓ_{KL}) and cross-entropy (ℓ_{CE}) on ground truth labels y; ψ is the softmax function; and Z_s and Z_t are the student model logits and teacher model logits, respectively.

The authors of [14] proposed hard-label distillation. They consider the teacher's hard decision to be a true label and take $y_t = argmax_c Z_t(c)$ as the teacher's hard decision. The objective of this hard-label distillation is as follows.

$$\ell_{\text{global}}^{\text{hardDistill}} = \frac{1}{2}\ell_{CE}(\psi(Z_s), y) + \frac{1}{2}\ell_{KL}(\psi(Z_s), y_t) \tag{7}$$

When a specific data augmentation occurs, the hard label of the teacher may change for a given image. This variant of distillation is conceptually simpler and parameter-free, and is superior to the traditional method. In addition, in [14], it was reported that the teacher prediction y_t plays the same role as the true label y.

2.3. Split Learning

SL [9] is another type of distributed collaborative machine learning (DCML). Unlike FL, SL divides a deep learning network W into several parts, each of which is processed and computed on a separate device. In a basic configuration, W is divided into two parts, W_c and W_s, which are referred to as the client-side and server-side networks, respectively.

The complete model is trained and tested by running sequential (forward/backward) propagation between the client and server. In its most basic form, forward propagation occurs as follows: A client forward propagates until a certain network layer called the cut layer is reached. Then, the cut layer's activations are referred to as smashed data and are relayed to the server. Subsequently, the server treats the crushed data (received from the client) as input and executes forward propagation on the next layer. Thus far, a single forward propagation has been achieved on the complete model. Back-propagation works as follows: After calculating the loss, the server performs back-propagation, which involves computing weight gradients and layer activations until the cut layer. Subsequently, the server sends the crushed data's gradients back to the client. The client performs back-propagation on its client-side network using the received gradients. Up to this point, a single pass of back-propagation between a client and server has been completed. The (forward and reverse) propagation in the ML model training continues until the model

is trained on all participating clients and reaches a reasonable convergence point (e.g., high prediction accuracy).

In SL, the learning process is synchronized with numerous clients, either in a centralized or peer-to-peer manner, but only one client engages with the server in one instance.

3. System Architecture

An overview of the proposed FedVKD framework is shown in Figure 1. The training process of the FedVKD framework and the problem formation and framework details of the proposed FedVKD are introduced in this section.

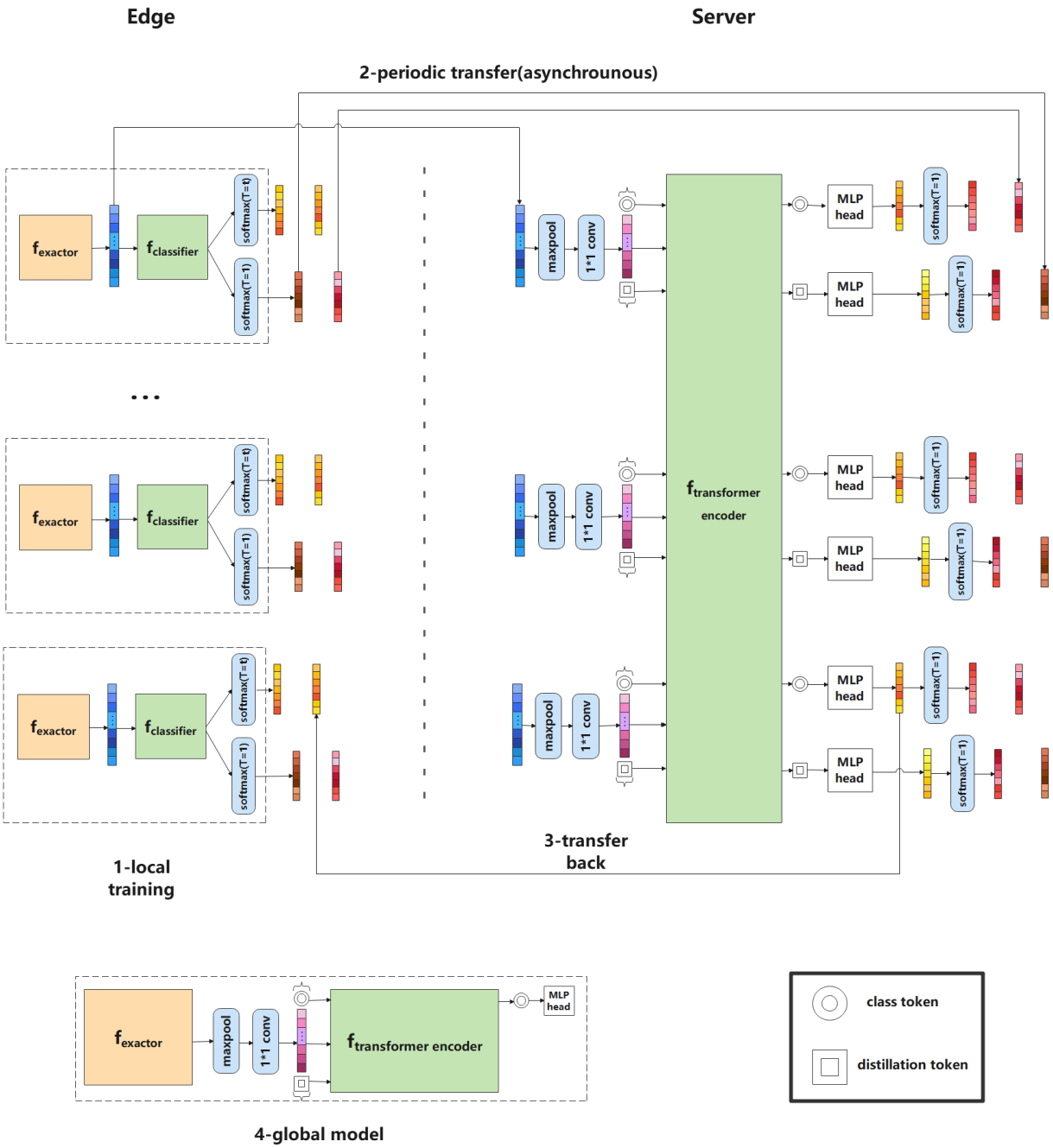

Figure 1. Training process of the FedVKD framework.

3.1. FedVKD Framework

FedVKD is an alternating minimization technique that optimizes two random variables (the edge model and the server model) by alternately fixing one and optimizing the other. FedVKD offloads the computing pressure onto the server and improves the performance of the edge and server models.

The model's training processes can be divided into four parts: (1) Local training: the CNN on the edge device is composed of a lightweight feature extractor and classifier and can be trained efficiently with its own data; (2) Periodic transfer: following local training, all edge nodes agree to generate the same tensor dimensions as the feature extractor output; (3) Transfer back: the large server model transformer encoder is trained by taking features derived from the edge-side extractor. Subsequently, a loss function of hard KD is used to minimize the gap between the prediction and the hard label predicted by the edge-side model and the gap between the prediction and ground truth. To boost the edge model, the server transmits its predicted soft labels to the edge model. Then, the edge model uses the server-side soft labels to train its local dataset with a loss function of soft KD; (4) Edge-sided model: consequently, the knowledge transferred from the edge models and vice versa significantly improves the server's performance. When training is concluded, the final model is a combination of the local feature extractor and the shared transformer encoder.

3.2. Problem Formation

We aim to collaboratively train the ViTs in FL, wherein many resource-constrained edge devices are not equipped with GPU accelerators. In particular, we consider supervised learning with C categories for a dataset $\mathcal{D}$. K-edge devices are used in the FL system. The k-th edge device has its own dataset $\mathcal{D}_k := \left\{\left(X_i^k, y_i\right)\right\}_{i=1}^{N^{(k)}}$, $N = \sum_{k=1}^{K} N^{(k)}$, $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \cdots, \mathcal{D}_k\}$. Abbreviations lists the primary notation used throughout this paper. Following earlier work, we formulate FL as a distributed optimization problem:

$$\min_{\boldsymbol{W}} F(\boldsymbol{W}) \stackrel{\text{def}}{=} \min_{\boldsymbol{W}} \sum_{k=1}^{K} \frac{N^{(k)}}{N} \cdot f^{(k)}(\boldsymbol{W}), \text{ where } f^{(k)}(\boldsymbol{W}) = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} \ell(\boldsymbol{W}; \boldsymbol{X}_i, y_i) \tag{8}$$

3.3. Training ViT in FL with KD

As mentioned above, traditional FL uses FedAvg to solve objective Equation (8) locally. However, in practice, the resource-constrained edge devices cannot train transformers owing to the lack of GPU accelerators and sufficient memory. Inspired by SL, we split ViTs (in particular, our ViT is a hybrid model equipped with a CNN stem) into two portions and offloaded the computing-intensive transformer block to the server side.

We divided the global ViT in Equation (8) into two partitions: a small feature extractor model W_e and a large-scale server-side model W_t, which are placed on the edge and server, respectively. Additionally, we added a classifier W_c for W_e to create a small, but fully trainable, model on the edge device. W_e was used to extract feature maps or patch input images. Subsequently, a single global model optimization was reformulated into a non-convex optimization problem that needs to solve the F_t and the F_c simultaneously, as follows.

$$\underset{\boldsymbol{W}_t}{\operatorname{argmin}} F_t(\boldsymbol{W}_t, \boldsymbol{W}_e^*) = \underset{\boldsymbol{W}_t}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} \boldsymbol{\ell}_t\left(f_t\left(\boldsymbol{W}_t; \boldsymbol{H}_i^{(k)}\right), y_i^{(k)}\right) \tag{9}$$

$$\text{subject to: } \boldsymbol{H}_i^{(k)} = f_e^{(k)}\left(\boldsymbol{W}_e^{(k)}; \boldsymbol{X}_i^{(k)}\right) \tag{10}$$

$$\underset{\left(\boldsymbol{W}_e^{(k)}, \boldsymbol{W}_c^{(k)}\right)}{\operatorname{argmin}} F_c\left(\boldsymbol{W}_e^{(k)}, \boldsymbol{W}_c^{(k)}\right) = \underset{\left(\boldsymbol{W}_e^{(k)}, \boldsymbol{W}_c^{(k)}\right)}{\operatorname{argmin}} \sum_{i=1}^{N^{(k)}} \ell_c\left(f^{(k)}\left(\left(\boldsymbol{W}_e^{(k)}, \boldsymbol{W}_c^{(k)}\right); \boldsymbol{X}_i^{(k)}\right), y_i^{(k)}\right) \tag{11}$$

$$= \underset{\left(W_e^{(k)}, \boldsymbol{W}_c^{(k)}\right)}{\operatorname{argmin}} \sum_{i=1}^{N^{(k)}} \ell_c \left(f_c^{(k)}(\boldsymbol{W}_c^{(k)}; \underbrace{f_e^{(k)}\left(\boldsymbol{W}_e^{(k)}; \boldsymbol{X}_i^{(k)}\right)}_{\boldsymbol{H}_i^{(k)}}), y_i^{(k)} \right) \tag{12}$$

Owing to the structural differences between the server model f_t and edge model f_c, we incorporated KD loss into the optimization equations to circumvent the optimization difficulty. Moreover, the knowledge transferred from the server model can boost the optimization on the edge. Inspired by DeiT, we adopted a hard distillation strategy for transformer blocks on the server side, while using traditional KD for the CNN network on the edge devices. We added a distillation token to the input of the transformer. The distillation token played a role similar to that of a class token. It interacted with other inputs via self-attention and was output by the network after the last layer. The distillation token ensured that the transformer blocks learn from the CNN through attention. $\ell_t^{harddistill}$ and $\ell_c^{(k)softdistill}$ are formulated as follows.

$$\ell_t^{\text{harddistill}} = \ell_{CE}\left(\psi(\mathbf{z}_t), y_i^{(k)}\right) + \sum_{k=1}^{K} \ell_{CE}\left(\psi(\mathbf{z}_t), y_c^{(k)}\right) \tag{13}$$

$$\ell_c^{(k)\ \text{softdistill}} = \ell_{CE}\left(\psi\left(\mathbf{z}_C^{(k)}\right), y_i^{(k)}\right) + \ell_{KD}\left(\mathbf{z}_t, \mathbf{z}_c^{(k)}\right) = \ell_{CE}\left(\psi\left(\mathbf{z}_C^{(k)}\right), y_i^{(k)}\right) + D_{KL}(\boldsymbol{p}_t \| \boldsymbol{p}_k) \tag{14}$$

where $\boldsymbol{p}_k^i = \frac{\exp\left(z_c^{(k,i)}/T\right)}{\sum_{i=1}^{C}\exp\left(z_c^{(k,i)}/T\right)}$ and $\boldsymbol{p}_t^i = \frac{\exp\left(z_t^i/T\right)}{\sum_{i=1}^{C}\exp\left(z_t^i/T\right)}$ are the probabilistic predictions of the edge-side model $f^{(k)}$ and server-side model f_t, respectively. After substituting Equations (13) and (14) into Equations (9) and (12), respectively, the optimization problem can be reformulated as follows.

$$\begin{aligned} \underset{\boldsymbol{W}_t}{\operatorname{argmin}} F_t\left(\boldsymbol{W}_t, \boldsymbol{W}_e^{(k)*}\right) = \underset{\boldsymbol{W}_t}{\operatorname{argmin}} & \sum_{k=1}^{K}\sum_{i=1}^{N^{(k)}} \ell_{CE}\left(f_t\left(\boldsymbol{W}_t; \boldsymbol{H}_i^{(k)}; X_{\text{class}}\right), y_i^{(k)}\right) \\ & + \sum_{k=1}^{K}\sum_{i=1}^{N^{(k)}} \ell_{CE}\left(f_t\left(\boldsymbol{W}_t; \boldsymbol{H}_i^{(k)}; X_{\text{distill}}\right), y_c^{(k)*}\right) \end{aligned} \tag{15}$$

$$\text{where } y_c^{(k)*} = f_c^{(k)}\left(\boldsymbol{W}_c^{(k)}; \boldsymbol{H}_i^{(k)}\right), \text{ and } \boldsymbol{H}_i^{(k)} = f_e^{(k)}\left(\boldsymbol{W}_e^{(k)*}; \boldsymbol{X}_i^{(k)}\right) \tag{16}$$

$$\underset{W^{(k)}}{\operatorname{argmin}} F_c\left(\boldsymbol{W}_t^*, \boldsymbol{W}^{(k)}\right) = \underset{W^{(k)}}{\operatorname{argmin}} \sum_{i=1}^{N^{(k)}} \ell_{CE}\left(\mathbf{z}_C^{(k)}, y_i^{(k)}\right) + \ell_{KD}\left(\mathbf{z}_t^*, \mathbf{z}_C^{(k)}\right) \tag{17}$$

$$\text{where } \mathbf{z}_c^{(k)} = f_c^{(k)}(\boldsymbol{W}_c^{(k)}; \underbrace{f_e^{(k)}\left(\boldsymbol{W}_e^{(k)}; \boldsymbol{X}_i^{(k)}\right)}_{\boldsymbol{H}_i^{(k)}}), \text{ and } \boldsymbol{z}_t^* = f_t\left(\boldsymbol{W}_t^*; \boldsymbol{H}_i^{(k)}\right) \tag{18}$$

where the * notation indicates that the related random variables are fixed during optimization. This optimization occurs across several rounds between Equations (15) and (17) until a convergence state is reached.

4. Experiments

4.1. Experimental Setup

Our FedVKD training framework was developed based on FedML [18]. FedML is an open-source federated learning research library that streamlines the development of novel algorithms and deploys them in a distributed computing environment. In resource-constrained experiments, our server node was equipped with four NVIDIA RTX 3080Ti GPUs, each with sufficient GPU memory to train large models. We used several CPU-based nodes as edge devices to train the CNNs. In resource-sufficient experiments for traditional

FL, our server node was equipped with four NVIDIA RTX 3080Ti GPUs, and each edge device was equipped with two NVIDIA RTX 2080Ti GPUs.

For our training task, we used image classification on CIFAR-10, CIFAR-100, and ImageNet. To distribute the training examples in a dataset among nodes, we followed earlier work [19]. If a dataset contains M classes, we divided the nodes into M groups at random. A training example with label l was allocated to group l with probability $p > 0$ and with probability $\frac{1-p}{M-1}$ to any other group. Data were provided uniformly to each node within the same group. The distribution difference of the nodes' local training data was controlled by p. If $p = \frac{1}{M}$, the nodes' local training data were independent and identically distributed (IID), otherwise they were non-independent (non-IID). Furthermore, a greater p suggests increased non-IID in the nodes' local training data. The fact that DCMLs frequently have non-IID local training data is one of its distinguishing features [7,20]. Consequently, we selected $p > \frac{1}{M}$ as the default to imitate the non-IID settings. We split the dataset into 10 groups according to their categories (e.g., each group has 10 categories in CIFAR-100) and set $p = \frac{1}{5}$ to simulate non-IID local training data.

FedVKD was compared to FedAvg [7], a state-of-the-art FL method, FedAUX, a federated distillation method with leveraging unlabeled auxiliary data [21]. and FedGKT, a CNN-based SL method [11]. After each round, test images were used for the global test. We used the top one percent test accuracy as a criterion to compare the model performance for different techniques. As shown in Figure 2, a hybrid ViT model was used as the global model in our framework. Inspired by the work of Xiao et al. [16], we replaced ViT-B's patchify stem with a shallow network of ResNet-18. The shallow network of ResNet-18 was placed on the edge side and served as the extractor of the edge-side CNN network. As shown in Figure 3, for FedGKT, the edge side was the same as that in our study. The server-side model architecture refers to FedGKT's ResNet-109, which comprises the global ResNet-113 with the extractor of the edge-side CNN network. However, the baseline FedAvg and FedAUX require all edge nodes to be trained using these two global models. Moreover, FedAUX needs to add subsets of ImageNet-21K as auxiliary data. We used an experimental setup as described in [21] for FedAUX training.

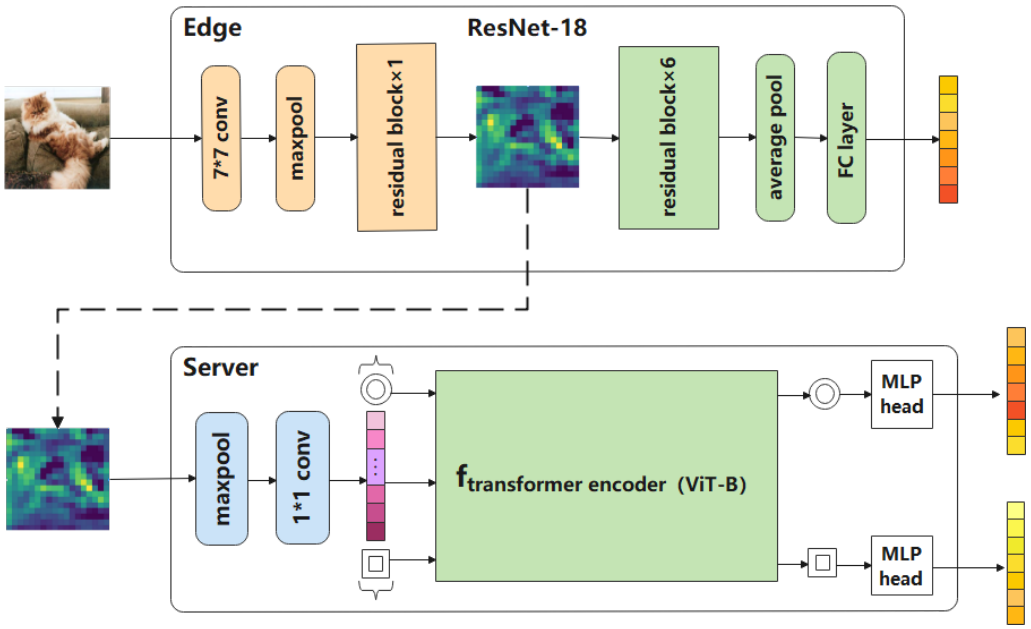

Figure 2. ViT architectures on the edge and server.

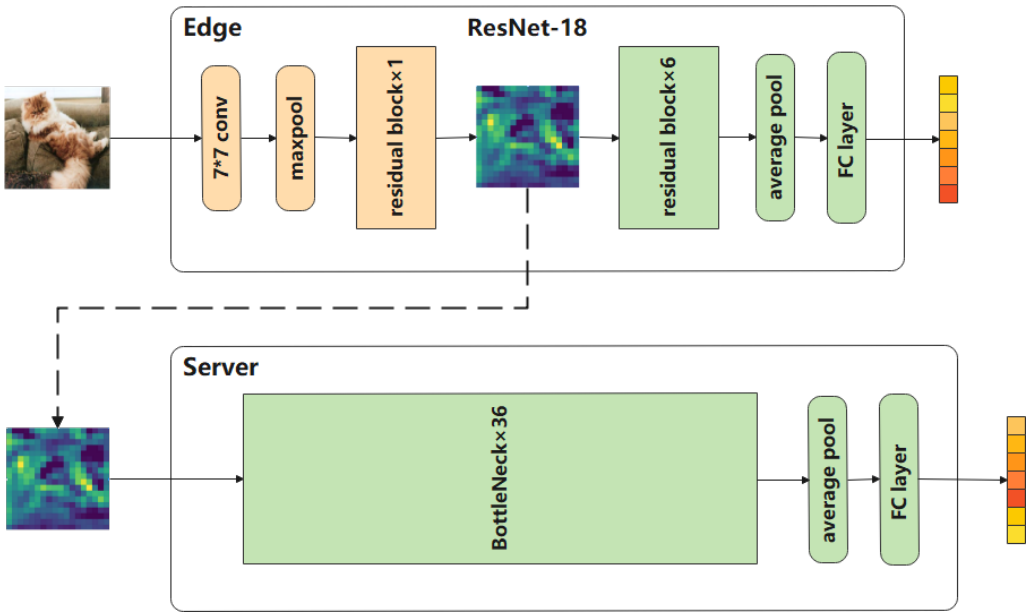

Figure 3. CNN architectures on the edge and server.

4.2. Experimental Results

In our experiment, 10 edge nodes and a server were run for all datasets and models. Figure 4 depicts the test accuracy curves during training under the IID setting with the four datasets. This includes the results for FedGKT (CNN) [11], FedAUX (CNN/ViT) [21], FedAvg (CNN/ViT) [7], and FedVKD (ViT). As shown in the figure, the ViT-based frameworks performed worse than the CNN-based frameworks in the case of a small dataset. This seemingly discouraging outcome may be because CNN-based frameworks benefit from the inductive biases inherent in CNNs when facing insufficient amounts of data. When the size of the datasets increased, the ViT-based frameworks began to outperform the CNN-based frameworks. As shown in Figure 5, the larger the dataset, the better the performance of the ViT-based frameworks. At best, the experimental results on the dataset of ImageNet showed that the test accuracy of the proposed fedVKD (ViT) was 5.71% higher than the FedAUX (CNN). When we focus only on ViT-based frameworks, we find that frameworks with KD performed better than the traditional method. Compared with the FedAUX (ViT) on the ImageNet, the test accuracy of the proposed fedVKD (ViT) was improved by 4.12%. We found that our KD strategy could aggregate more knowledge from edge devices. In particular, the edge-side CNN networks generalized better when facing insufficient amounts of data, and our method was able to aggregate such generalization using KD.

In Table 1, we summarize all the numerical results for our method and the baselines in both IID and non-IID settings. As confirmed by this study, as well as earlier FL work [11,22,23], it is common for the test accuracy under non-IID to be lower than that under IID. As well as benefiting from KD and transformer structure, the fedvkd (ViT) with released performance also exhibited better stability in the test accuracy in the face of non-IID datasets.On the ImageNet, the test accuracy of the fedvkd (ViT) only decreased by 2.79%, but the FedAUX (ViT), the FedGKT (CNN) and the FedAvg (ViT) decreased by 4.47%, 7.31% and 3.79%, respectively.

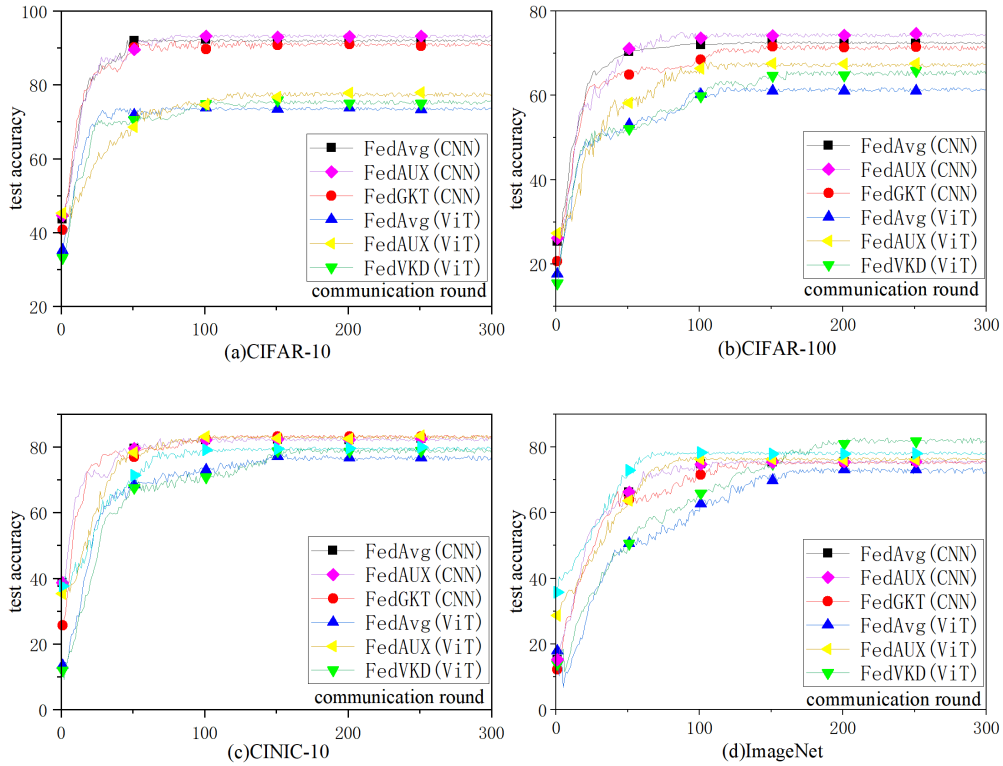

Figure 4. Test accuracy curves of the four frameworks in the four datasets of CIFAR-10, CIFAR-100, CINIC-10, ImagrNet.

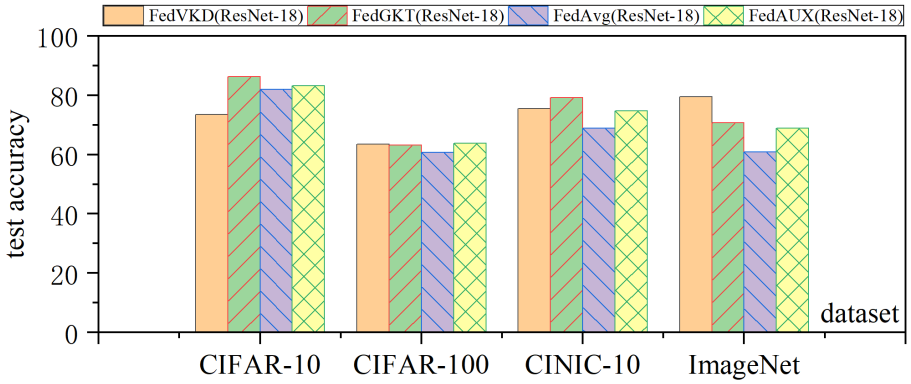

Figure 5. Comparison of the test accuracy of the edge-side model over the four datasets of CIFAR-10, CIFAR-100, CINIC-10, ImagrNet.

We tested the performance of small CNN networks on the edge side under three different frameworks. As shown in Figure 5, the performance of ResNet-18 with the KD strategy was better than that with FedAvg. This means that a small network can acquire more knowledge from a large network using KD. Our ViT-based distillation method outperformed the CNN-based distillation method. Therefore, a stronger and more studious student network can teach teacher networks. We believe that this finding is meaningful. The use of ViT-based KD in FL not only boosts the performance of the global model but also the edge-side model. An efficient and generalized small model is very useful when the deployment conditions of the model are resource constrained.

Table 1. Comparison of the test accuracy of our method and several state-of-the-art methods over four datasets in IID and non-IID settings.

Method	CIFAR-10		CIFAR-100		CINIC-10		ImageNet	
	IID	Non-IID	IID	Non-IID	IID	Non-IID	IID	Non-IID
FedVKD (ViT)	75.88	69.93	65.86	60.79	79.74	76.18	82.72	80.41
FedAUX (ViT)	77.99	74.17	68.04	62.44	80.04	77.46	78.6	75.09
FedAvg (ViT)	73.91	68.10	61.87	57.84	77.77	74.24	73.79	70.99
FedGKT (CNN)	91.96	85.17	71.86	65.31	83.62	77.27	75.78	70.24
FedAUX (CNN)	93.6	90.05	74.98	70.56	83.5	78.81	77.01	74.31
FedAvg (CNN)	92.74	84.36	72.71	66.44	82.75	75.18	75.86	69.83

5. Discussion and Conclusions

In this paper, we have presented a framework (FedVKD) to train ViTs in resource-constrained FL. In the process of periodic transfer, the server-side transformer encoder learns from multiple small edge-side CNNs using KD. This strategy offloads computationally intensive tasks to the server and takes full advantage of the inductive biases of CNNs. Moreover, the small edge-side CNNs learn more from the server-side transformer encoder. Essentially, our FedVKD is a process of learning and meriting from each other and makes full use of the unique advantages of the models.

Currently, data have become a basic strategic resource; however, data always come from resource-constrained edge devices (such as smartphones, smartwatches, IoT devices, and personal computers). Therefore, as demonstrated by this study, it is meaningful to introduce a stronger model to edge devices while maintaining data privacy. Our current study focuses on CV. In addition, the transformer structure provides a very good bridge between CV and NLP. Our future work will expand the scale of the experiments and explore the performance of our framework when applied to the field of multimodal learning.

6. Limitations of the Study

We have sought to make our analyses as comprehensive as possible, but FL is an art of trade-offs among many factors. We acknowledge that it is a challenge to design a universal system that can solve all problems, thus we discuss some limitations of our framework.

Byzantine robustness: Due to its distributed characteristic, FL is vulnerable to hostile operations on malicious edges, which could be fake edges injected by an attacker, or genuine edges invaded by an attacker. Traditionally, malicious edges can poison the global model by sending poisoned local model updates to the server (called local model poisoning attacks) or poisoning their local training data (known as data poisoning attacks). In our framework, malicious edges can corrupt the global model by poisoning the local feature map sent to the server.

Privacy: As [11] points out, existing technologies, such as multi-party computation (MPC) and differential privacy (DP), are capable of defending data privacy against a hidden vector reconstruction attack; exchanging hidden feature maps appears to be a safer option than exchanging the model or gradient. However, the lack of analysis and comparison of the degree of privacy leakages between these three settings (model, gradient, and hidden map) represents a further limitation of our work.

Author Contributions: Conceptualization, J.T.; methodology, J.T.; software, J.T.; validation, J.T., Z.G. (Zhen Gao), Z.G. (Zhaohui Guo); formal analysis, J.T.; investigation, J.T.; resources, J.T.; data curation, J.T.; writing—original draft preparation, J.T.; writing—review and editing, Z.G. (Zhen Gao), Z.G. (Zhaohui Guo); visualization, J.T.; supervision, Z.G. (Zhen Gao); project administration, Z.G. (Zhen Gao); funding acquisition, Z.G. (Zhen Gao). All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key Research and Development Program of China under Grant 2021YFE0205300.

Data Availability Statement: The CIFAR-10 and CIFAR-100 dataset referred to in this study are openly available in "Learning multiple layers of features from tiny images" at https://www.cs.toronto.edu/~kriz/cifar.html, accessed on 8 February 2022; the CINIC-10 dataset referred to in this study is openly available in "CINIC-10 is not ImageNet or CIFAR-10" at DOI:10.7488/DS/2448, accessed on 8 February 2022; the ImageNet dataset referred to in this study is openly available in "ImageNet: A large-scale hierarchical image database" at DOI:10.1109/CVPR.2009.5206848, accessed on 8 February 2022; the ImageNet-21K dataset referred to in this study is openly available in "ImageNet-21K Pretraining for the Masses" at https://github.com/Alibaba-MIIL/ImageNet21K, accessed on 8 July 2022.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

Notations	Meanings
$\boldsymbol{X}_i$	i-th training sample
y_i	Corresponding label of X_i, $y_i \in \{1, 2, \ldots, C\}$
$N^{(k)}$	Sample number in dataset $\mathcal{D}_k$
$\boldsymbol{W}$	Network weights of a global model
$f^{(k)}(\boldsymbol{W})$	i-th edge's local objective function
ℓ	Loss function of the global model
ℓ_t	General loss functions for the server-side model
ℓ_c	General loss functions for the edge-side model
f_t	Server-side model
$f_e^{(k)}$	i-th edge's feature extractor
$f_C^{(k)}$	i-th edge's classifier
$f^{(k)}$	Edge-side model including $f_e^{(k)}$ followed by $f_C^{(k)}$
$\boldsymbol{W}_t$	Network weights of f_t
$\boldsymbol{W}_e^{(k)}$	Network weights of $f_e^{(k)}$
$\boldsymbol{W}_c^{(k)}$	Network weights of $f_C^{(k)}$
$\boldsymbol{W}^{(k)}$	Combination of $\boldsymbol{W}_e^{(k)}$ and $\boldsymbol{W}_c^{(k)}$
$\boldsymbol{H}_i^{(k)}$	i-th sample's feature map (a hidden vector or tensor)
ℓ_{CE}	Cross-entropy loss between the predicted values and ground truth labels
ℓ_{KD}	Kullback–Leibler (KL) divergence function
z_F	Output of the last fully connected layer in the server-side model
$z_c^{(k)}$	Output of the last fully connected layer in the edge-side model
T	Temperature hyperparameter of the softmax function
X_{class}	Class token
$X_{distill}$	Distillation token

References

1. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S. An image is worth 16 × 16 words: Transformers for image recognition at scale. In Proceedings of the 9th International Conference on Learning Representations (ICLR 2021), Virtual Event, 3–7 May 2021.
2. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
3. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Online, 6–12 December 2020; pp. 1877–1901.
4. Lepikhin, D.; Lee, H.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. In Proceedings of the 9th International Conference on Learning Representations (ICLR 2021), Virtual Event, 3–7 May 2021.
5. Warnat-Herresthal, S.; Schultze, H.; Shastry, K.L.; Manamohan, S.; Mukherjee, S.; Garg, V.; Sarveswara, R.; Händler, K.; Pickkers, P.; Aziz, N.A.; et al. Swarm Learning for decentralized and confidential clinical machine learning. *Nature* **2021**, *594*, 265–270. [CrossRef] [PubMed]

6. Froelicher, D.; Troncoso-Pastoriza, J.R.; Raisaro, J.L.; Cuendet, M.A.; Sousa, J.S.; Cho, H.; Berger, B.; Fellay, J.; Hubaux, J.-P. Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption. *Nat. Commun.* **2021**, *12*, 5910. [CrossRef]
7. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.Y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017), Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
8. Peter, K.; McMahan, H.B.; Brendan, A.; Aurélien, B.; Mehdi, B.; Arjun Nitin, B.; Kallista, B.; Zachary, C.; Graham, C.; Rachel, C.; et al. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
9. Gupta, O.; Raskar, R. Distributed learning of deep neural network over multiple agents. *J. Netw. Comput. Appl.* **2018**, *116*, 1–8. [CrossRef]
10. Vepakomma, P.; Gupta, O.; Swedish, T.; Raskar, R. Split Learning for Health: Distributed Deep Learning without Sharing Raw Patient Data. ICLR AI for Social Good Workshop 2019. Available online: https://aiforsocialgood.github.io/iclr2019/accepted/track1/pdfs/31_aisg_iclr2019.pdf (accessed on 1 October 2021).
11. He, C.; Annavaram, M.; Avestimehr, S. Group knowledge transfer: Federated learning of large cnns at the edge. In Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NeurIPS 2020), Online, 6–12 December 2020; pp. 14068–14080.
12. Buciluǎ, C.; Caruana, R.; Niculescu-Mizil, A. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2016; pp. 535–541.
13. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531v1.
14. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training data-efficient image transformers & distillation through attention. In Proceedings of the 38th International Conference on Machine Learning, Virtual Event, 18–24 July 2021; pp. 10347–10357.
15. Sun, S.; Cheng, Y.; Gan, Z.; Liu, J. Patient knowledge distillation for bert model compression. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019; pp. 4323–4332.
16. Xiao, T.; Singh, M.; Mintun, E.; Darrell, T.; Dollár, P.; Girshick, R. Early convolutions help transformers see better. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Virtual Event, 6–14 December 2021; pp. 30392–30400.
17. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local Neural Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7794–7803. [CrossRef]
18. He, C.; Li, S.; So, J.; Zeng, X.; Zhang, M.; Wang, H.; Wang, X.; Vepakomma, P.; Singh, A.; Qiu, H. Fedml: A research library and benchmark for federated machine learning. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020 SpicyFL Workshop), Online, 6–12 December 2020.
19. Fang, M.; Cao, X.; Jia, J.; Gong, N. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Boston, MA, USA, 12–14 August 2020; pp. 1605–1622.
20. Konečný, J.; McMahan, H.B.; Felix, X.Y.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated learning: Strategies for improving communication efficiency. In Proceedings of the 6th International Conference on Learning Representations (ICLR 2018), Vancouver, BC, Canada, 30 April–3 May 2018.
21. Sattler, F.; Korjakow, T.; Rischke, R.; Samek, W. FedAUX: Leveraging Unlabeled Auxiliary Data in Federated Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**. [CrossRef] [PubMed]
22. Hsieh, K.; Phanishayee, A.; Mutlu, O.; Gibbons, P. The non-iid data quagmire of decentralized machine learning. In Proceedings of the 37th International Conference on Machine Learning, Virtual Event, 13–18 July 2020; pp. 4387–4398.
23. Reddi, S.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečný, J.; Kumar, S.; McMahan, H.B. Adaptive federated optimization. In Proceedings of the 9th International Conference on Learning Representations (ICLR 2021), Virtual Event, 3–7 May 2021.