FedZKT: Zero-Shot Knowledge Transfer towards Resource-Constrained Federated Learning with Heterogeneous On-Device Models

Lan Zhang
Department of Electrical and Computer Engineering, Michigan Technological University
Houghton, MI USA
lanzhang@mtu.edu

Dapeng Wu
Department of Electrical and Computer Engineering, University of Florida
Gainesville, FL USA
dpwu@ufl.edu

Xiaoyong Yuan
College of Computing Michigan Technological University
Houghton, MI USA
xyyuan@mtu.edu

arXiv:2109.03775v2 [cs.LG] 5 Apr 2022

***Abstract*—Federated learning enables multiple distributed devices to collaboratively learn a shared prediction model without centralizing their on-device data. Most of the current algorithms require comparable individual efforts for local training with the same structure and size of on-device models, which, however, impedes participation from resource-constrained devices. Given the widespread yet heterogeneous devices nowadays, in this paper, we propose an innovative federated learning framework with heterogeneous on-device models through Zero-shot Knowledge Transfer, named by FedZKT. Specifically, FedZKT allows devices to independently determine the on-device models upon their local resources. To achieve knowledge transfer across these heterogeneous on-device models, a zero-shot distillation approach is designed without any prerequisites for private on-device data, which is contrary to certain prior research based on a public dataset or a pre-trained data generator. Moreover, this compute-intensive distillation task is assigned to the server to allow the participation of resource-constrained devices, where a generator is adversarially learned with the ensemble of collected on-device models. The distilled central knowledge is then sent back in the form of the corresponding on-device model parameters, which can be easily absorbed on the device side. Extensive experimental studies demonstrate the effectiveness and robustness of FedZKT towards on-device knowledge agnostic, on-device model heterogeneity, and other challenging federated learning scenarios, such as heterogeneous on-device data and straggler effects.**

***Index Terms*—Federated Learning, Model Heterogeneity, Resource Constraint, Data-Free, Knowledge Transfer**

I. Introduction

The demand for on-device training is recently increasing, as evinced by the surge of interest in federated learning [1]. Federated learning leverages on-device training at multiple distributed devices to obtain a knowledge-abundant global model without centralizing private on-device data [1], [2]. Classical federated learning algorithms, represented by FedAvg [2], require on-device training with the same model structure and size to perform the element-wise central average, which, however, impedes collaboration across heterogeneous hardware platforms. For instance, both wearable devices and smartphones are popular eHealth devices to monitor infectious diseases [3]. However, a typical wearable device is usually equipped with microcontroller units (MCU), whose on-chip memory is three orders of magnitude smaller than a smartphone due to the limited resource budget, especially the memory (SRAM) and storage (Flash) [4]. Hence, a wearable device can hardly run the same on-device model designed for a smartphone [5], resulting in the ineffectiveness of implementing classical federated learning. Given the widespread yet heterogeneous devices nowadays, it is vital to enable extensive participation in federated learning, especially with resource-constrained devices.

One promising solution is to allow federated learning with heterogeneous on-device models, which recently has attracted great attention. Diao *et al*. proposed HeteroFL to adaptively allocate a subset of global model parameters to an on-device model [6]. Inherently, HeteroFL assumes the architecture of a small model can be a subnetwork of a large one, which, however, is not always practical. For example, it is hard to find the architecture of MobileNet [7], *i.e*., a popular on-device model, as a subnetwork of other models, such as ShuffleNet [8] and ResNet [9]. Instead, some recent research enabled devices to design their on-device models independently based on federated distillation techniques [10]–[13]. Specifically, the logit information of on-device models is shared in FedMD [10], Cronus [11], and FedH2L [12] to achieve federated learning for personalization, security, and decentralization, respectively, and on-device model parameters are shared in FedDF [13] for robust model fusion. Although successful, all above algorithms rely on certain prerequisites of on-device knowledge, which leverage either a public dataset or a pre-trained data generator to extract and transfer knowledge. Consequently, the construction of such data-dependency requires careful deliberation and even prior knowledge of the private on-device data, making it infeasible in many applications in practice. Moreover, the impact of the quality of such prerequisites remains unclear on federated learning performance.

To tackle the above limitations of existing research, in this paper, we propose a Zero-shot Knowledge Transfer framework named by FedZKT for resource-constrained federated learning with heterogeneous on-device models in a data-free manner.

Specifically, in FedZKT, devices can design on-device models based on their heterogeneous local resources independently. To enable knowledge transfer across these on-device models, a zero-shot federated distillation approach is proposed without any prerequisite for private on-device data, which is contrary to the aforementioned prior research. This compute-intensive distillation task is assigned to the server to reduce the workload at devices, where the server constructs a generator to be adversarially trained with the ensemble of the collected on-device models. The distilled central knowledge is then sent back in the form of the corresponding on-device model parameters, which can be easily absorbed on the device side. In other words, FedZKT enables extensive participation, especially from the ubiquitous resource-constrained devices, who can easily contribute to federated learning by following the classical federated learning procedures with locally designed compact models. Overall, FedZKT consolidates several *advantages* into a single framework: independent on-device model design, extensive participation from resource-constrained and/or heterogeneous devices, and data-free knowledge transfer. Our main contributions are as follows:

- This paper introduces an innovative framework, FedZKT, for resource-constrained federated learning in a data-free manner, which performs zero-shot knowledge transfer across heterogeneous on-device models. Several key modules are designed to implement the lightweight and compute-intensive learning tasks on the device and server sides, respectively, which perfectly fits the unbalanced resources on both sides.
- Contrary to certain prior research based on either a public dataset or a pre-trained data generator, FedZKT provides an on-device knowledge agnostic approach without data-dependency concerns, where the server adversarially learns a generative model with the global model based on the ensemble of collected on-device models. A new loss function, softmax l_1 (SL) loss, is proposed to facilitate the zero-shot federated knowledge distillation.
- Extensive experimental results demonstrate the effectiveness of FedZKT on four popular datasets, with higher accuracy and better generalization performance compared to the state-of-the-art. FedZKT also performs robustness to challenging federated learning scenarios, such as non-iid data distribution and straggler effects.

II. Related Work

A. Heterogeneous Federated Learning.

Classical federated learning algorithms, represented by FedAvg [2], average the collected on-device model parameters to obtain a global model. Since the training mainly happens on the device side, the overall learning performance largely depends on participating devices. It has been shown that the statistical heterogeneity across devices, *i.e.*, non-iid on-device data, can lead to slow and unstable convergence [14], [15]. Such performance degradation has also been found when on-device resources, such as the local computing power or network connectivity, are heterogeneous [16]. Recent research has developed solutions to either address the "straggler effect" introduced by some poorly performed devices [2], [16]–[18] or reduce the local model size at all devices [19], [20]. However, most of these designs are still under the learning paradigm of FedAvg with homogeneous on-device models, *i.e.*, all devices need to run on-device models with the same structure and size.

B. Federated Distillation.

To allow federated learning with heterogeneous on-device models, federated distillation has attracted significant attention recently. Enlighten by the well-known knowledge distillation idea that transfers knowledge from a single or multiple teacher models to an empty student model [21]–[24], federated distillation learns a global model based on the collected on-device models. Since the data is stored locally and cannot be shared in federated settings, most federated distillation design is data-dependent. Specifically, by leveraging a pre-known public or surrogate dataset, federated distillation has been studied to handle heterogeneous on-device models for personalization [10], security [11], decentralization [12], and robustness [13], respectively. In addition, federated distillation has been used to improve federated learning performance, such as communication efficiency [25]–[29] and on-device privacy [11], [30], or address the aforementioned data heterogeneity challenges [31], [32]. However, the prerequisite of the data-dependency is not always available for federated distillation due to the unknown or confidential data distribution of on-device models. Moreover, the absence of a qualified public or surrogate dataset can return a poor approximation of teacher models in knowledge distillation [33], whose impact on federated learning still remains unclear. In this paper, we target a data-free federated distillation design without the data-dependency prerequisite for the resource-constrained federated learning with heterogeneous on-device models.

C. Data-Free Knowledge Distillation.

Recent efforts have been made on data-free knowledge distillation via zero-shot learning techniques [33]–[36]. Typically, a generative model is learned to synthesize the queries that the student makes to the teacher. Although this idea has been well studied in classical knowledge distillation, such as for model compression [34], little attention has been paid to distillation in federated settings. To the best of our knowledge, the data-free distillation design in FeDGen [37] is the closest setting to this work. Specifically, FeDGen targets the slow convergence issue due to data heterogeneity in federated learning, while this paper aims to enable resource-constrained federated learning with heterogeneous on-device models. In FeDGen, a generator is used on the device side to augment local knowledge for data-free distillation. Instead of distributing the generator to devices, this paper learns and keeps the generator at the server. Only the updated on-device model parameters will be sent back to devices, which can be easily absorbed locally. In this way, the compute-intensive data-free distillation task is assigned to the powerful server to enable extensive participation especially from resource-constrained devices.

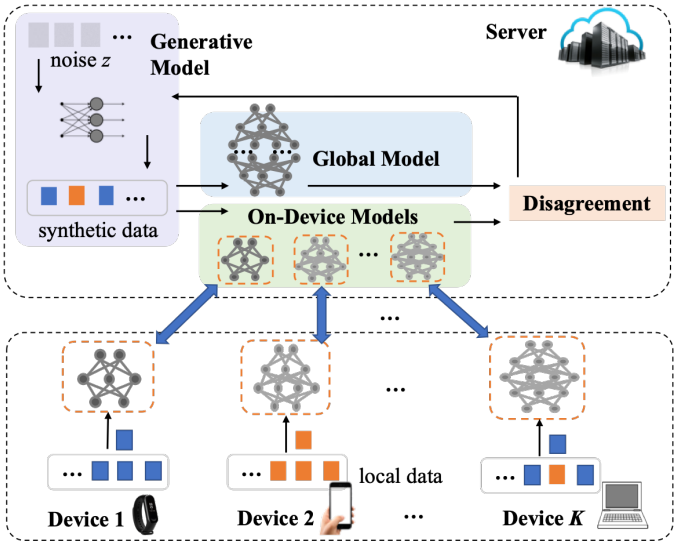


Fig. 1: **Overview of FedZKT: zero-shot knowledge transfer for federated learning with heterogeneous on-device models.** Most compute-intensive workloads of FedZKT are done at the server. Participating devices only need to update the on-device models based on their local data.

III. FedZKT: Federated Learning via Zero-shot Knowledge Transfer

This section presents the proposed FedZKT. We first describe the problem statement, followed by FedZKT design. Several key modules of FedZKT are detailed at both the server and device sides.

A. Problem Statement

As shown in Figure 1, we consider a federated learning task, *e.g.*, a supervised classification task, across K heterogeneous devices in set $\mathcal{K}$. Each device $k \in \mathcal{K}$ can independently design its on-device model f_k parameterized by w_k to "best" fit its own resources in computation, communication, storage, and power. In this case, the on-device models $\{f_k\}_{k\in\mathcal{K}}$ may have distinct model architectures. In addition, we assume each device owns a confidential dataset $\mathcal{D}_k$ that cannot be shared, and the server has no prior knowledge about the data at the device side, such as the data distribution. The server is assumed to be powerful, who coordinates all participating devices and aggregates their on-device knowledge. Hence, the main goal of the server is to extract distributed on-device knowledge from heterogeneous on-device models $\{f_k\}_{k\in\mathcal{K}}$ to obtain a knowledge-abundant global model F. Meanwhile, the well-trained on-device models $\{f_k\}_{k\in\mathcal{K}}$ will eventually contain the distilled knowledge from all peer devices.

B. FedZKT Design

To meet the unbalanced computing capabilities between the server and participating devices, the compute-intensive knowledge distillation task is assigned to the server. The procedures of the overall FedZKT are introduced in Algorithm 1. In the following, we will elaborate design principles of several key modules of FedZKT.

Algorithm 1 FedZKT

INPUT: global model parameters w, on-device model parameters $\{w_k\}_{k\in\mathcal{K}}$, generative model parameters θ, total communication rounds T, local training epochs T_l, distillation epochs n_D.

Initialize all models with random weights[1].
for each communication round $t = 1, 2, \cdots, T$ **do**
  $\mathcal{K}^t \leftarrow$ server selects a random subset devices from $\mathcal{K}$ as active devices
  // On-Device Update
  for each device $k \in \mathcal{K}^t$ **in parallel do**
    $\hat{w}_k \leftarrow$ **DeviceUpdate**$(w_k, \mathcal{D}_k, T_l, n_D)$
    Upload $\hat{w}_k$ to the server.
  end for
  // Server Update
  $\{w_k\}_{k\in\mathcal{K}}, w, \theta \leftarrow$ **ServerUpdate**$(\{\hat{w}_k\}_{k\in\mathcal{K}}, w, \theta)$.
  for each device $k \in \mathcal{K}$ **in parallel do**
    Transfer w_k to Device k.
  end for
end for

RETURN: w, $\{w_k\}_{k\in\mathcal{K}}$

Algorithm 2 FedZKT: **DeviceUpdate**

INPUT: on-device model f_k's parameters w_k, local dataset $\mathcal{D}_k$, local epochs T_l

for $t \in \{1, 2, ..., T_l\}$ **do**
  $\mathcal{L}_k \leftarrow \sum_{\{x,y\}\in\mathcal{D}_k} \mathcal{L}_{CE}(f_k(x; w_k), y)$
  $w_k \leftarrow w_k - \eta \frac{\partial \mathcal{L}_k}{\partial w_k}$
end for

RETURN: w_k

1) Zero-Shot Knowledge Distillation: As aforementioned, the goal of knowledge distillation at the server is to obtain the global model $\mathcal{F}(\cdot; w)$ parameterized by ω, which is expected to match the ensemble of on-device models $\{f_k\}_{k\in\mathcal{K}}$ that are trained respectively on distributed local knowledge domains $\{\mathcal{D}_k\}_{k\in\mathcal{K}}$. Since $\{\mathcal{D}_k\}_{k\in\mathcal{K}}$ are private and cannot be shared in federated settings, one intuitive idea is to leverage a synthetic dataset $\mathcal{D}_S$ to mimic the local knowledge in order to minimize the loss of disagreement between the teacher and the students. Thus, based on ensemble learning, we have

$$\min_{w} E_{x\sim\mathcal{D}_S}[\mathcal{L}(\mathcal{F}(x; w), f_{\text{ens}}(x))], \tag{1}$$

where f_{ens} denotes the ensemble of on-device models, *i.e.*, $f_{\text{ens}}(x) = \frac{1}{|\mathcal{K}|}\sum_k f_k(x; w_k)$, and $\mathcal{L}$ denotes the loss function to measure the disagreement between the global model $\mathcal{F}$ and f_{ens}. More discussions about $\mathcal{L}$ will be given in the next module.

[1]This work uses Glorot initialization [38]. The same initialization is not required for on-device models.

Algorithm 3 FedZKT: **ServerUpdate**

INPUT: global model $\mathcal{F}$'s parameters w, on-device model f_k's parameters $\{w_k\}_{k\in\mathcal{K}}$, generator G's parameters θ, distillation epochs n_D.

// Transfer knowledge from on-device to the global model
for $n \in \{1, 2, ..., n_D\}$ **do**
  // Generator Update
  $z \sim \mathcal{N}(0, \mathbf{I})$
  $x \leftarrow G(z; \theta)$
  $\mathcal{L}_G \leftarrow -\mathcal{L}(\mathcal{F}(x; w), \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} f_k(x; w_k))$
  $\theta \leftarrow \theta - \eta_G \frac{\partial \mathcal{L}_G}{\partial \theta}$
  // Global Model Update
  $z \sim \mathcal{N}(0, \mathbf{I})$
  $x \leftarrow G(z; \theta)$
  $\mathcal{L}_S \leftarrow \mathcal{L}(\mathcal{F}(x; w), \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} f_k(x; w_k))$
  $w \leftarrow w - \eta_S \frac{\partial \mathcal{L}_S}{\partial w}$
end for
// Transfer knowledge from global to on-device models
for $n \in \{1, 2, ..., n_D\}$ **do**
  $z \sim \mathcal{N}(0, \mathbf{I})$
  $x \leftarrow G(z; \theta)$
  for each device $k \in \mathcal{K}$ **do**
    $\mathcal{L}_k \leftarrow \mathcal{L}(\mathcal{F}(x; w), f_k(x; w_k))$
    $w_k \leftarrow w_k - \eta_S \frac{\partial \mathcal{L}_k}{\partial w_k}$
  end for
end for

RETURN: $\{w_k\}_{k\in\mathcal{K}}$, w, θ

Since $\mathcal{D}_S$ is expected to synthesize the private local knowledge in a data-free manner, instead of assuming a pre-known $\mathcal{D}_S$, we introduce a generative model G to distill local knowledge in a zero-shot manner. Enlighten by the well-known idea of Generative Adversarial Networks (GAN) [33], G is responsible to provide difficult inputs for the training of $\mathcal{F}$, which maximizes the disagreement between the current global and on-device models. Meanwhile the generated inputs also need to perform well in (1) to enable knowledge matching between the global and on-device models. Hence, the goals of G and $\mathcal{F}$ are to maximize and minimize the disagreement between $\mathcal{F}$ and $\{f_k\}_{k\in\mathcal{K}}$, respectively, where the adversarial game can be given by

$$\min_{\mathcal{F}} \max_{G} E_{z\sim\mathcal{N}(0,1)}[\mathcal{L}(\mathcal{F}(G(z)), f_{\text{ens}}(G(z)))], \tag{2}$$

where z is the noise following Gaussian distribution $\mathcal{N}(0, 1)$. Therefore, the server will alternatively train the generative model G and global model $\mathcal{F}$ in (2). It should be mentioned that most prior knowledge distillation approaches perform well when extracting a small student model from a large teacher model, while some recent research [33], [35] has shown that high distillation accuracy can be achieved even when the teacher model has a smaller and different architecture than the student's. Along this line, this work is further motivated to transfer knowledge between the powerful global model and the heterogeneous and potentially compact on-device models.

2) Loss Function Design: This module discusses the loss function $\mathcal{L}$ in (2) that measures the disagreement between the global model $\mathcal{F}$ and the on-device model ensemble f_{ens}, which is used to train $\mathcal{F}$ and the generative model G simultaneously. The loss function design is key to the distillation performance since the gradients computed through $\mathcal{F}$ and f_{ens} can easily impede the convergence of the optimizer, such as leading to gradient vanishing [34] when the wrong loss function is used.

Most prior research of knowledge distillation measures the model disagreement between the teacher and the student by *Kullback–Leibler (KL) divergence* [23], [24]. Hence, the KL divergence between the outputs of the global model $\mathcal{F}$ and the ensemble of on-device models f_{ens} after the softmax function becomes a candidate for the loss function, where the KL-divergence loss function can be given by

$$\mathcal{L}_{\text{KL}}(x) = \sum \mathcal{F}(x) \log \frac{\mathcal{F}(x)}{f_{\text{ens}}(x)}. \tag{3}$$

However, the KL-divergence loss tends to suffer from gradient vanishing [34] with respect to input data x when the student model $\mathcal{F}$ converges to the teacher model f_{ens}. The problem becomes even more serious in zero-shot distillation settings, since the gradient vanishing will further affect the training of the generative model G.

In view of this, recent zero-shot distillation research [33], [34] introduces ℓ_1 *norm loss*, which compares the logit outputs (model outputs before the softmax layer) between the teacher and student models:

$$\mathcal{L}_{\ell_1}(x) = ||u(x) - \frac{1}{|\mathcal{K}|}\sum_k v_k(x)||_1, \tag{4}$$

where u and v_k denote the logit outputs of the global model (student) and the kth on-device model (teacher), respectively. However, given the diverse on-device model parameters in our heterogeneous federated learning, the ℓ_1 norm loss may lead to the unstable training due to the large gradients. Specifically, federated learning requires aggregating distributed knowledge from participating devices. However, averaging the logit values over on-device models increases the gradients, making the whole learning process unstable.

To address the above challenges for zero-shot distillation in federated learning, we propose a new loss function named by *Softmax ℓ_1 (SL) loss*, which applies the softmax output to the ℓ_1 norm loss:

$$\mathcal{L}_{\text{SL}}(x) = ||\mathcal{F}(x) - f_{\text{ens}}(x)||_1. \tag{5}$$

The SL loss is designed to overcome the drawbacks of using KL-divergence loss and ℓ_1 norm loss. Two hypotheses for this design are provided below. Specifically, Hypothesis 1 suggests that the SL loss can reduce the gradient vanishing effect than the KL-divergence loss for better convergence in zero-shot distillation; Hypothesis 2 suggests that the SL loss can make the training more stable compared to the ℓ_1 norm loss. Details are given in the Appendix for justification. In addition,

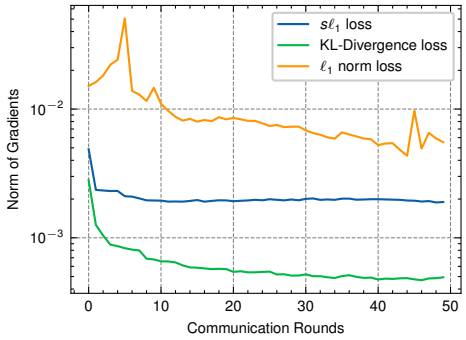


Fig. 2: **Norm of gradients w.r.t input data (MNIST, IID).** The gradients for the KL-divergence loss tend to vanish, while the gradients for the ℓ_1 norm loss are much larger and unstable during the learning process. The proposed SL loss overcomes both problems in the federated learning.

Figure 2 shows the norm of gradients for KL-divergence, ℓ_1 norm, and the proposed SL norm losses, respectively, where the gradients for the KL-divergence loss tend to vanish, while the gradients for the ℓ_1 norm loss are much larger and unstable during the learning process. More empirical evaluation results are given in Section IV-C1.

Hypothesis 1. *When the global model F converges to the ensemble of on-device models f_{ens}, the gradients of KL divergence loss with respect to the input data x are smaller than those of the SL loss:*

$$||\nabla_x \mathcal{L}_{\text{KL}}(x)|| \underset{F \to f_{\text{ens}}}{\leq} ||\nabla_x \mathcal{L}_{\text{SL}}(x)||. \tag{6}$$

Hypothesis 2. *When the global model F converges to the ensemble of on-device models f_{ens}, the gradients of the ℓ_1 norm loss with respect to the input data x are greater than those of the SL loss:*

SL loss的梯度比KL大，比L1小

$$||\nabla_x \mathcal{L}_{\ell_1}(x)|| \underset{F \to f_{\text{ens}}}{\geq} ||\nabla_x \mathcal{L}_{\text{SL}}(x)||. \tag{7}$$

3) Bidirectional Knowledge Transfer: The above two modules enable the knowledge transfer from devices to the server. After that, the aggregated central knowledge needs to transfer back for the next learning iteration. One intuitive approach is to broadcast the updated global model $\mathcal{F}$, based on which, device k can use its data $\mathcal{D}_k$ to distill an updated on-device model, $\min_{w_k} E_{x\sim\mathcal{D}_k}[\mathcal{L}(\mathcal{F}(x), f_k(x; w_k))]$, similar to our discussion in the above module based on (1). However, since our design aims to enable resource-constrained federated learning, to utmostly reduce the workload at devices, we run the compute-intensive distillation task to transfer knowledge from the global model to on-device models at the server. Since the above module learns the generator G to produce difficult data that maximizes the disagreement between global and on-device models, we will reuse this well-learned generator G to provide input data to distill the updated on-device model $\{f'_k\}_{k\in\mathcal{K}}$. Hence, the objective function for knowledge transfer from the global model $\mathcal{F}$ to the on-device model f'_k, $k \in \mathcal{K}$ can be given by

$$\min_{f'_k} E_{z\sim\mathcal{N}(0,1)}[\mathcal{L}(\mathcal{F}(G(z)), f'_k(G(z)))]. \tag{8}$$

Different from the distillation of module III-B1 designed for a zero-shot setting, we adopt the KL-divergence loss $\mathcal{L}_{\text{KL}}$ here for distillation with a pre-trained data generator G. The updated on-device model f'_k will be sent back to device k, which obtains the central knowledge learned by global model $\mathcal{F}$. This completes the one-round bidirectional knowledge transfer. Since the computation of (2) and (8) is performed at the server, FedZKT follows the same on-device learning mode as classical federated learning [2], while allowing participation with independently designed compact on-device models.

4) ℓ_2 Regularization for Non-IID Data Distribution: In addition to tackling the model architecture heterogeneity, FedZKT is expected to handle data heterogeneity in federated learning. To do so, we limit the update of on-device models when training on their local datasets in Algorithm 2. Specifically, the ℓ_2 regularization is added to the loss function of the on-device update, which is given by

$$\min_{w_k^t} \sum_{\{x,y\}\in\mathcal{D}_k} \mathcal{L}_{CE}(f_k(x; w_k^t), y) + ||w_k^t - w_k^{t-1}||_2^2, \tag{9}$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss for classification tasks, and w_k^{t-1} is the parameter set transferred from the server in the last iteration $t-1$. The ℓ_2 regularization has been used to tackle the non-iid data distribution with homogeneous model architectures in FedProx [39]. Compared with FedProx, due to model heterogeneity, FedZKT uses the received local model parameter set w_k^{t-1} rather than the global model parameter set w^{t-1}. Empirical evaluation in Section IV-C4 further demonstrates the improvement of adding ℓ_2 regularization for non-iid data distributions.

IV. Experimental Validation

This section conducts extensive experiments to evaluate the proposed FedZKT. We first introduce the experimental setup, followed by the experimental results and ablation studies.

A. Experimental Setup

1) Dataset: The experiments are conducted on four widely used image datasets: MNIST [40], KMNIST [41], FASHION-MNIST [42] (FASHION in short in this paper), and CIFAR-10 [43].

2) On-Device Model Heterogeneity: Five different neural network architectures are considered for each dataset. For small datasets, *i.e.*, MNIST, KMNIST, and FASHION, we deploy a CNN model, a Fully-Connected Model, and three LeNet-like models with different channel sizes and numbers of layers. For CIFAR-10 dataset, we deploy two ShuffleNetV2 models [8], two MobileNetV2 models [7], and a LeNet-like model. Specifically, to support resource-constrained federated learning, ShuffleNetV2 and MobileNetV2 are adopted, which

are popular neural architectures designed towards low-end devices. LeNet is a simple neural network architecture consisting of two convolutional layers and three fully connected layers. We use different channel sizes for each ShuffleNetV2 and MobileNetV2 model, so as to increase the heterogeneity of on-device models in the evaluation.

3) Federated Learning Settings: The experiments are conducted with multiple devices $K \in \{5, 10, 15, 20\}$ (by default k=10). For the small datasets, *i.e.*, MNIST, KMNIST, FASHION, we conduct 50 communication rounds ($T = 50$); in each round, each device trains the on-device models for 5 epochs. For the CIFAR-10 dataset, we conduct 100 communication rounds ($T = 100$); in each round, each device trains the on-device model for 10 epochs. We use the stochastic gradient descent (SGD) for both the on-device and global training, where the learning rate is set to be 0.01. Besides, we train the server model and the generator for 200 iterations for the small datasets ($n_G = n_s = 200$) and 500 iterations for the CIFAR-10 dataset ($n_G = n_s = 500$). The generator is trained using Adam optimizer with a batch size of 256 and a learning rate of 0.001. The learning rates for both the server model and the generator are reduced by 0.3 at the half and 3/4 of the total iterations. The batch size for all model training is 256.

4) Data Heterogeneity: The experiments will be conducted on both iid and non-iid on-device data distributions. In the iid setting, on-device data is randomly drawn from the dataset. In the non-iid setting, two scenarios of label distribution skew are adopted based on the common experimental settings in recent federated learning research [44], [45]: 1) quantity-based label imbalance, where each device owns data consisting of a specific number of classes; 2) distribution-based label imbalance, where each device owns a proportion of the labels following a Dirichlet distribution. We sample $p_k = \{p_{kj}\}$ from $Dir_N(\beta)$, where p_{kj} denotes the proportion of data in class k owned by device j, and β is a concentration parameter of the Dirichlet distribution. A small value of β suggests a more imbalanced distribution of labels.

5) Baseline Approach: As aforementioned in Section II-B and II-C, existing federated learning designs to support heterogeneous on-device models are data-dependent, *i.e.*, based on either a public dataset or a pre-trained generator, while the proposed FedZKT is based on a data-free manner. Hence, the baseline approach for FedZKT adopts the most representative data-dependent algorithm, FedMD [10]. Similar to FedZKT, FedMD allows devices to independently design their on-device models, which provides extensive experimental results for on-device model heterogeneity but requiring a public dataset. To learn from MNIST, FASHION, and KMNIST, we select public datasets as FASHION, MNIST, and FASHION, respectively. To explore the impact of data dependency during knowledge transfer, we select two different public datasets, *i.e.*, CIFAR-100 and SVHN, to learn CIFAR-10.

B. Experimental Results

1) IID Data Distribution: We first report the average accuracy of the global model eventually learned by the proposed FedZKT and the baseline approach FedMD under the iid on-device data distribution. As illustrated in Table I, FedZKT achieves higher accuracy than FedMD in most cases, which indicates the effectiveness of FedZKT. Besides, we observe that the performance of FedMD depends on the selection of the public dataset. Specifically, FedMD uses two different public datasets, CIFAR-100 and SVHN, to train on the CIFAR-10 dataset, respectively. When the public dataset (CIFAR-100) has similar distribution as the on-device dataset (CIFAR-10), FedMD achieves a higher accuracy; while when the public dataset (SVHN) is quite different from the on-device dataset (CIFAR-10), the performance of FedMD drops significantly. Thus, it is critical for FedMD to select a proper public dataset at the server, which unfortunately is extremely challenging in practice since the server may have no access to the private on-device dataset. Instead, FedZKT provides a data-free approach with even higher accuracy performance.

On-Device Dataset	FedMD		FedZKT
	Public Dataset	Average Accuracy	Average Accuracy
MNIST	FASHION	96.69%	**97.76%**
FASHION	MNIST	**85.83%**	84.42%
KMNIST	FASHION	84.02%	**86.43%**
CIFAR-10	CIFAR-100	67.34%	**78.02%**
CIFAR-10	SVHN	20.38%	

TABLE I: Performance of FedZKT and FedMD under IID on-device data distribution.

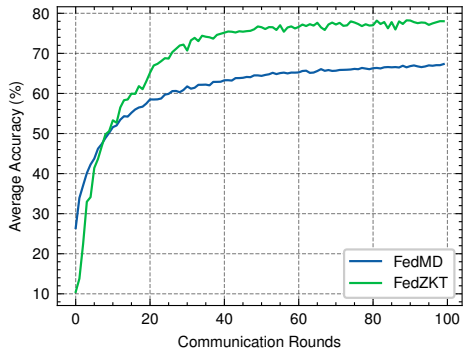
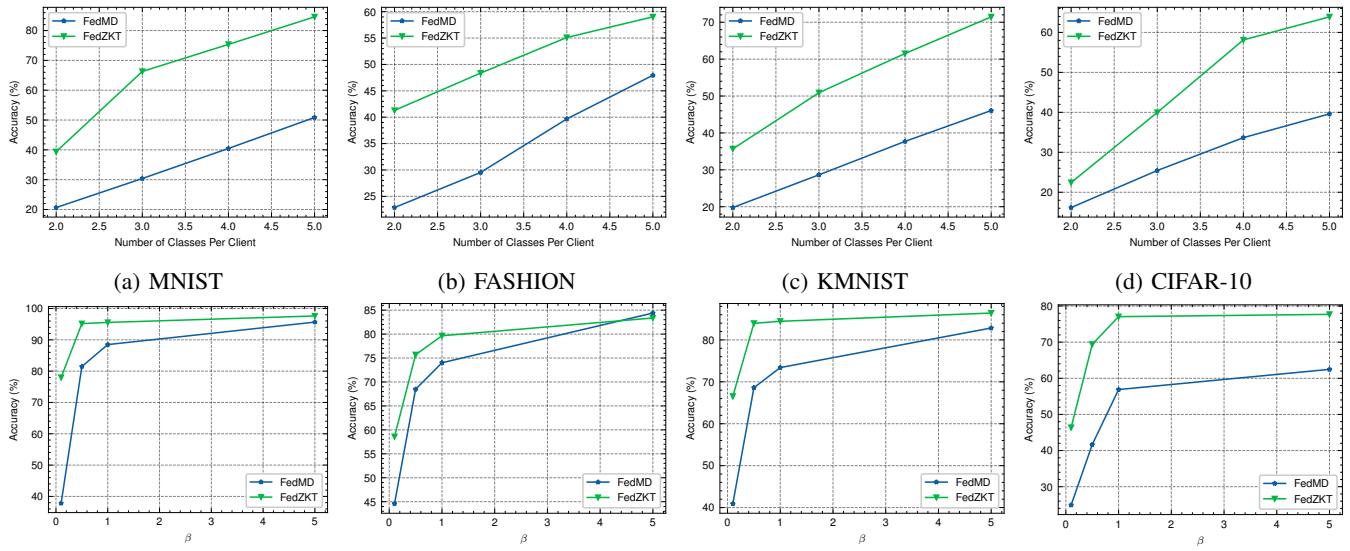


Fig. 3: Learning curves of FedZKT and FedMD (CIFAR-10, IID).

We then evaluate learning curves of FedZKT and FedMD on CIFAR-10 dataset under the iid on-device data distribution, where FedMD uses CIFAR-100 dataset as the public dataset. As shown in Figure 3, FedMD performs better than FedZKT at the beginning of learning. Since the distribution of the public dataset (CIFAR-100) is close to that of the on-device dataset (CIFAR-10), FedMD can quickly absorb knowledge by using the public dataset at the beginning. However, we observe that with the increase of learning rounds, FedZKT eventually outperforms FedMD. This is because FedZKT can iteratively learn from on-device models to improve the generator and thus produces more representative samples, while FedMD can only



(a) MNIST (b) FASHION (c) KMNIST (d) CIFAR-10

(e) MNIST (f) FASHION (g) KMNIST (h) CIFAR-10

Fig. 4: Performance of FedZKT and FedMD under non-IID on-device data distribution: Quantity-based label imbalance (a)-(d), Distribution-based label imbalance (e)-(h).

Non-IID scenario	KL-divergence	ℓ_1 norm	SL loss
$C = 5$	48.23%	14.60%	**63.89%**
$\beta = 0.5$	66.17%	16.34%	**69.39%**

TABLE II: Effect of loss functions for zero-shot knowledge distillation in FedZKT (CIFAR-10, Non-IID).

use existing samples from the public dataset that cannot be improved during learning, which further indicates the benefits of our data-free design.

2) Non-IID Data Distribution: The accuracy performance of FedZKT and FedMD is also evaluated under the non-iid on-device data distribution. As aforementioned in Section IV-A4, two data skew scenarios are considered for the non-iid setting: in the quantity-based label imbalance scenario, the values of the number of classes per device are set to be $c \in \{2, 3, 4, 5\}$; in the distribution-based label imbalance scenario, four values of the concentration parameter $\beta \in \{0.1, 0.5, 1, 5\}$ are considered. Figure 4 illustrates the accuracy performance under the above two non-iid data scenarios. We observe that FedZKT outperforms FedMD in almost all non-iid environments, which indicates the robustness of FedZKT in handling non-iid on-device data scenarios.

C. Ablation Studies

This section performs ablation studies to evaluate the impact of the key modules/considerations for FedZKT under challenging federated learning environments.

1) Effects of Loss Function Design: We first evaluate the loss function design for zero-shot federated knowledge distillation in Section III-B2. We consider the more challenging non-iid on-device data scenarios: the distribution-based label imbalance with $\beta = 0.5$ and the quantity-based label imbalance with $c = 5$. As illustrated in Table II, the proposed SL loss achieves better accuracy performance than the KL-divergence loss and the ℓ_1 norm loss in the two non-iid scenarios. In addition, our results show that ℓ_1 norm loss is not suitable for zero-shot federated distillation under non-iid settings due to the unstable learning performance, although it can avoid the gradient vanishing in zero-shot distillation.

2) Effects of On-Device Model Architectures: One key consideration of FedZKT is to enable each devices, especially the resource-constrained one, to independently design the on-device model. Hence, Figure 5 evaluates the impact of the on-device model architecture on the learning curve of each device. Specifically, we consider ten devices under the same model architecture configuration as Table III. Detailed model settings are provided in the Appendix. Since Device 5 and Device 10 use Model E, *i.e.*, a simple LeNet-like neural network, their on-device learning performance is lower than those using the ShuffleNetV2 and MobileNetV2 models. In addition, we present the lower bound and upper bound performance of on-device training in Table III, where the lower bound considers the on-device model is trained on its own data only; the upper bound assumes the on-device model can access others' local data. By comparing Table III with the eventual accuracy performance in Figure 5, we observe that the performance of FedZKT is very close to the upper-bound values, which indicates the effectiveness of FedZKT in handling federated learning with heterogeneous on-device models.

3) Effects of Stragglers: Straggler effect has been one major concern in federated learning, where one or several devices cannot timely participate in training due to the unstable on-device conditions, such as the poor networking and the low battery conditions. Hence, we evaluate the effect of stragglers

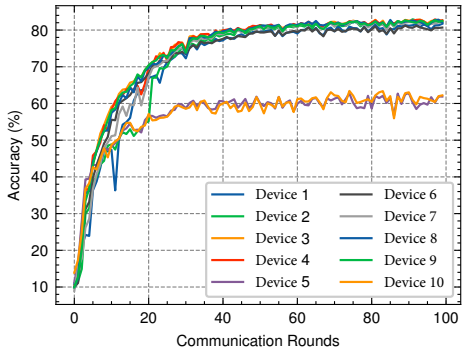
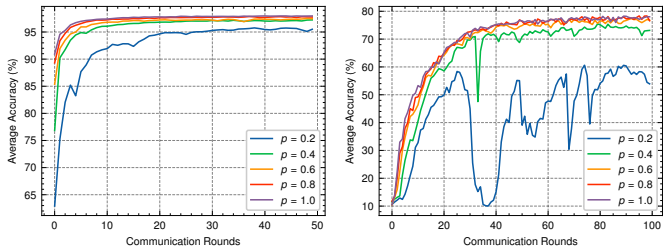


Fig. 5: Effect of on-device model architecture: learning curves of devices in FedZKT with different models (CIFAR-10, IID).

Model Architecture	Upper Bound	Lower Bound
Device 1: Model A	84.18%	50.17%
Device 2: Model B	86.98%	54.08%
Device 3: Model C	88.63%	58.56%
Device 4: Model D	87.36%	57.84%
Device 5: Model E	70.77%	58.90%
Device 6: Model A	84.39%	50.13%
Device 7: Model B	85.99%	51.07%
Device 8: Model C	88.34%	62.15%
Device 9: Model D	88.87%	59.88%
Device 10: Model E	70.65%	54.81%

TABLE III: Effect of on-device model architecture: lower and upper bound of on-device performance (CIFAR-10, IID).



(a) MNIST, IID. (b) CIFAR-10, IID.

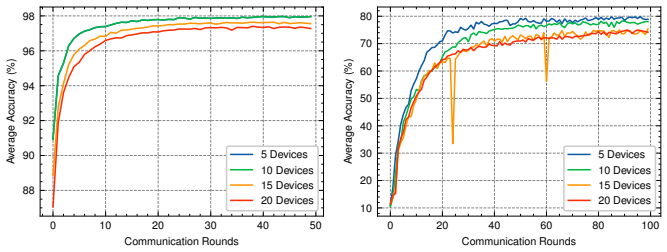
Fig. 6: Effect of stragglers: average accuracy of FedZKT when p portion of devices are trained in each round.

in FedZKT. Specifically, in each communication round, we randomly select a portion p of devices as the active ones who participate in federated learning, $p \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, and the rest devices are inactive that cannot contribute to the learning round. The learning procedures follow the Algorithm 1 but do not update the parameters for inactive devices. As shown in Figure 6, the performance of FedZKT is stable during the training in most cases. The unstable training only occurs when a very small portion of devices, *i.e.*, $p = 0.2$, participates, slowing down the process of training. As long as the majority of devices can participate in the training, the stragglers do not have a significant impact on FedZKT.

4) Effects of ℓ_2 Regularization: Experiments are conducted to evaluate the performance of the ℓ_2 regularization for on-device training against the non-iid data distribution in FedZKT (Section III-B4). As illustrated in Table IV, the performance of using ℓ_2 regularization is better than that without the regularization in both quantity-based (C = 5) and distribution-based (β = 0.5) label imbalance non-iid scenarios, which demonstrates the effectiveness of the ℓ_2 regularization.

Non-IID scenario	no regularization	ℓ_2 regularization
$C = 5$	56.58%	**63.89%**
$\beta = 0.5$	66.17%	**69.39%**

TABLE IV: Effect of ℓ_2 regularization in FedZKT (CIFAR-10, Non-IID).



(a) MNIST, IID. (b) CIFAR-10, IID.

Fig. 7: Effects of device number: average accuracy of K on-devices models participated in FedZKT ($K \in \{5, 10, 15, 20\}$).

5) Effects of Device Number: We finally evaluate the effect of device number K on FedZKT, where $K \in \{5, 10, 15, 20\}$. Figure 7 illustrates the learning curves of FedZKT for MINIST and CIFAR-10 datasets under iid conditions. We observe that the number of devices does not affect the overall training too much. Although a smaller number of devices, *e.g.*, $K = 5$, produce higher average accuracy, FedZKT achieves good performance with more devices as well. As observed from the experiments, the effect of device numbers on FedZKT is subtle, *i.e.*, around $\pm 2\%$ in terms of the average accuracy.

V. Conclusions

This paper proposed an innovative federated learning framework for resource-constrained and heterogeneous devices via zero-shot knowledge transfer, named by FedZKT. FedZKT allows devices, especially the resource-constrained ones, to determine their on-device models independently. Unlike certain prior research relied on the prerequisite for private on-device knowledge, FedZKT enables knowledge transfer across heterogeneous on-device models in a data-free manner, where a new loss function, SL loss, is proposed to facilitate the zero-shot federated knowledge distillation. Moreover, to meet the unbalanced resources between the server and device sides, FedZKT assigns the compute-intensive distillation task to the server. The distilled central knowledge is then sent back in the form of on-device model parameters, which can be easily absorbed at devices. Extensive experiments demonstrate the effectiveness and the robustness of FedZKT towards on-device knowledge agnostic (data-free), on-device model heterogene-

ity, and other challenging federated learning scenarios, such as heterogeneous on-device data and straggler effects.

ACKNOWLEDGMENT

The authors thank all the anonymous reviewers for their insightful feedback. This work was supported by the National Science Foundation under Grant No. 2106754

REFERENCES

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[3] X. Chen, G. Zhu, L. Zhang, Y. Fang, L. Guo, and X. Chen, "Age-stratified covid-19 spread analysis and vaccination: A multitype random network approach," *IEEE Transactions on Network Science and Engineering*, 2021.

[4] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "Mcunet: Tiny deep learning on iot devices," *arXiv preprint arXiv:2007.10319*, 2020.

[5] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," *arXiv preprint arXiv:1908.09791*, 2019.

[6] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," *arXiv preprint arXiv:2010.01264*, 2020.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[8] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[11] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr, "Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer," *arXiv preprint arXiv:1912.11279*, 2019.

[12] Y. Li, W. Zhou, H. Wang, H. Mi, and T. M. Hospedales, "Fedh2l: Federated learning with model and statistical heterogeneity," *arXiv preprint arXiv:2101.11296*, 2021.

[13] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *arXiv preprint arXiv:2006.07242*, 2020.

[14] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[15] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[16] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[17] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.

[18] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[19] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[20] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large cnns at the edge," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[21] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[22] L. J. Ba and R. Caruana, "Do deep nets really need to be deep?" *arXiv preprint arXiv:1312.6184*, 2013.

[23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[24] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4794–4802.

[25] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[26] N. Guha, A. Talwalkar, and V. Smith, "One-shot federated learning," *arXiv preprint arXiv:1902.11175*, 2019.

[27] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data," *arXiv preprint arXiv:2008.06180*, 2020.

[28] F. Sattler, A. Marban, R. Rischke, and W. Samek, "Communication-efficient federated distillation," *arXiv preprint arXiv:2012.00632*, 2020.

[29] H. Seo, J. Park, S. Oh, M. Bennis, and S.-L. Kim, "Federated knowledge distillation," *arXiv preprint arXiv:2011.02367*, 2020.

[30] L. Sun and L. Lyu, "Federated model distillation with noise-free differential privacy," *arXiv preprint arXiv:2009.05537*, 2020.

[31] H.-Y. Chen and W.-L. Chao, "Feddistill: Making bayesian model ensemble applicable to federated learning," *arXiv e-prints*, pp. arXiv–2009, 2020.

[32] F. Sattler, T. Korjakow, R. Rischke, and W. Samek, "Fedaux: Leveraging unlabeled auxiliary data in federated learning," *arXiv preprint arXiv:2102.02514*, 2021.

[33] J.-B. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4771–4780.

[34] G. Fang, J. Song, C. Shen, X. Wang, D. Chen, and M. Song, "Data-free adversarial distillation," *arXiv preprint arXiv:1912.11006*, 2019.

[35] P. Micaelli and A. Storkey, "Zero-shot knowledge transfer via adversarial belief matching," *arXiv preprint arXiv:1905.09768*, 2019.

[36] Y. Choi, J. Choi, M. El-Khamy, and J. Lee, "Data-free network quantization with adversarial knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 710–711.

[37] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," *arXiv preprint arXiv:2105.10056*, 2021.

[38] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[39] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[40] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[41] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," 2018.

[42] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[43] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.

[44] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[45] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," *arXiv preprint arXiv:2102.02079*, 2021.

APPENDIX: LOSS FUNCTION DESIGN

This section justifies the two hypotheses proposed for the loss function design of zero-shot knowledge distillation in section III-B2. The goal of this design is to select the loss function in (2), which measures the disagreement between the global model $\mathcal{F}$ and the ensemble of on-device models f_{ens}, *i.e.*, $f_{\text{ens}}(x) = \frac{1}{|\mathcal{K}|}\sum_k f_k(x; w_k)$, in a zero-shot approach.

Given the input data x, we define the outputs of $\mathcal{F}$ after the softmax layer as $U = \mathcal{F}(x)$, and the logit outputs of the on-device model f_k, $k \in \mathcal{K}$ as $V_k = f_k(x)$. In addition, define the logit output before the softmax layer as u and v_k for $\mathcal{F}$ and f_k, $k \in \mathcal{K}$, respectively. Thus, we have $U = \mathrm{softmax}(u)$ and $V_k = \mathrm{softmax}(v_k)$. Based on the above definitions, in the following, we first provide the norm of gradients for ℓ_1 norm loss, KL-divergence loss, and SL loss, respectively, followed by the hypotheses justification.

A. Norm of gradients for ℓ_1 norm loss

Given the ℓ_1 norm loss in our setting can be defined by

$$\mathcal{L}_{\ell_1} = ||u - \frac{1}{|\mathcal{K}|}\sum_k v_k||_1,$$

we have the gradients of the ℓ_1 norm loss with respect to input data x by

$$\begin{aligned}\nabla_x \mathcal{L}_{\ell_1} =& \sum_i \mathrm{sign}(u^i - \frac{1}{|\mathcal{K}|}\sum_j v_j^i)(\frac{\partial u^i}{\partial x} - \frac{\partial(\frac{1}{|\mathcal{K}|}\sum_k v_k^i)}{\partial x}),\\ =& \sum_i \mathrm{sign}(u^i - \frac{1}{|\mathcal{K}|}\sum_j v_j^i)\frac{1}{|\mathcal{K}|}\sum_k(\frac{\partial u^i}{\partial x} - \frac{\partial v_k^i}{\partial x}),\\ =& \frac{1}{|\mathcal{K}|}\sum_i\sum_k \mathrm{sign}(u^i - \frac{1}{|\mathcal{K}|}\sum_j v_j^i)(\frac{\partial u^i}{\partial x} - \frac{\partial v_k^i}{\partial x}),\end{aligned}$$

where i denotes the dimension of output. Hence, the norm of gradients of ℓ_1 norm loss can be given by

$$\begin{aligned}&||\nabla_x \mathcal{L}_{\ell_1}|| =\\ &\frac{1}{|\mathcal{K}|}||\sum_i\sum_k \mathrm{sign}(u^i - \frac{1}{|\mathcal{K}|}\sum v_k^i)(\frac{\partial u^i}{\partial x} - \frac{\partial v_k^i}{\partial x})||.\end{aligned} \tag{10}$$

B. Norm of gradients for KL-divergence loss

Given the KL-divergence loss in our setting defined by

$$\mathcal{L}_{\mathrm{KL}} = \sum_i U^i \log\frac{U^i}{\frac{1}{|\mathcal{K}|}\sum_k V_k^i},$$

we have the gradients of KL-divergence loss with respect to input x, which can be given by

$$\begin{aligned}&\nabla_x \mathcal{L}_{\mathrm{KL}} =\\ &\sum_i \frac{\partial U^i}{\partial x}\log\frac{U^i}{\frac{1}{|\mathcal{K}|}\sum_k V_k^i} - \frac{\partial(\frac{1}{|\mathcal{K}|}\sum_k V_k^i)}{\partial x}\frac{U^i}{\frac{1}{|\mathcal{K}|}\sum_k V_k^i}.\end{aligned} \tag{11}$$

When U converges to the ensemble of V_k, we have

$$U(x) = \frac{1}{|\mathcal{K}|}\sum_k V_k(x)(1+\delta(x)), \tag{12}$$

where $\delta(x)\to 0$ when $F \to f_{\mathrm{ens}}$. Thus, the gradients of KL-divergence loss in Eq. 12 can be given by

$$\begin{aligned}\nabla_x \mathcal{L}_{\mathrm{KL}} \approx& \sum_i \frac{\partial U^i}{\partial x}\delta_i - \frac{\partial(\frac{1}{|\mathcal{K}|}\sum_k V_k^i)}{\partial x}(1+\delta_i) \qquad (13)\\ &(\mathrm{since}\log(1+\delta) \underset{\delta\to 0}{\to} \delta)\\ =& \sum_i \delta_i(\frac{\partial U^i}{\partial x} - \frac{\partial(\frac{1}{|\mathcal{K}|}\sum_k V_k^i)}{\partial x}) \qquad (14)\\ &(\forall k, \sum_i \frac{\partial V_k^i}{\partial x} = 0, \mathrm{since}\sum_i V_k^i = 1),\\ =& \frac{1}{|\mathcal{K}|}\sum_i\sum_k \delta_i(\frac{\partial U^i}{\partial x} - \frac{\partial V_k^i}{\partial x}) \qquad (15)\end{aligned}$$

Thus, the norm of the gradients of KL-divergence loss can be given by

$$||\nabla_x \mathcal{L}_{\mathrm{KL}}|| \approx \frac{1}{|\mathcal{K}|}||\sum_i\sum_k \delta_i(\frac{\partial U^i}{\partial x} - \frac{\partial V_k^i}{\partial x})||. \tag{16}$$

C. Norm of gradients for SL Loss

The proposed SL loss is defined by

$$\mathcal{L}_{\mathrm{SL}} = ||U - \frac{1}{|\mathcal{K}|}\sum_k V_k||_1.$$

Similarly to the ℓ_1 norm loss, the gradients of SL loss with respect to input x can be given by

$$\begin{aligned}&\nabla_x \mathcal{L}_{\mathrm{SL}} =\\ &\frac{1}{|\mathcal{K}|}\sum_i\sum_k \mathrm{sign}(U^i - \frac{1}{|\mathcal{K}|}\sum_j V_j^i)(\frac{\partial U^i}{\partial x} - \frac{\partial V_k^i}{\partial x}).\end{aligned} \tag{17}$$

Thus, the norm of gradients of the SL loss can be given by

$$\begin{aligned}&||\nabla_x \mathcal{L}_{\mathrm{SL}}|| =\\ &\frac{1}{|\mathcal{K}|}||\sum_i\sum_k \mathrm{sign}(U^i - \frac{1}{|\mathcal{K}|}\sum_j V_j^i)(\frac{\partial U^i}{\partial x} - \frac{\partial V_k^i}{\partial x})||.\end{aligned} \tag{18}$$

D. Justification of Two Hypotheses

Based on the above norm of gradients, this section justifies the Hypothesis 1 and Hypothesis 2. We first introduce the two Lemmas provided in [33].

Lemma 1. *[Lemma 2 in [33]] If $S(x) \in (0,1)^K$ is the softmax output of a differentiable function (e.g., a neural network) on an input x, s is the corresponding logits vector, and J is the Jacobian matrix $\frac{\partial S}{\partial s}$, then for any vector z, we have:*

$$\forall z, \quad ||Jz|| \le ||z||. \tag{19}$$

Lemma 2. *[Lemma 3 in [33]] Let $U(x)$ and $V(x)$ be the softmax output of two differential functions (e.g., neural networks) on input x, with respective logits $u(x)$ and $v(x)$. When U converges to V then $\frac{\partial U}{\partial u}$ converges to $\frac{\partial V}{\partial v}$.*

Based on the above two Lemmas, we then reformulate the derived norm of gradients of the three loss functions, and

Model A		Model B		Model C		Model D		Model E
Arch	Parameter	Arch	Parameter	Arch	Parameter	Arch	Parameter	Arch
ShuffleNetV2	net size 0.5	ShuffleNetV2	net size 1.0	MobileNetV2	width multiplier 0.8	MobileNetV2	width multiplier 0.6	LeNet

TABLE V: Model Architecture for the CIFAR-10 dataset.

finally derive the following two hypotheses. Their proofs are given below.

Hypothesis 3. *When the global model F converges to the ensemble of on-device models f_{ens}, the gradients of KL divergence loss with respect to the input data x are smaller than those of the SL loss:*

$$||\nabla_x \mathcal{L}_{\text{KL}}(x)|| \underset{F \to f_{\text{ens}}}{\leq} ||\nabla_x \mathcal{L}_{\text{SL}}(x)||. \tag{20}$$

Hypothesis 4. *When the global model F converges to the ensemble of on-device models f_{ens}, the gradients of the ℓ_1 norm loss with respect to the input data x are greater than those of the SL loss:*

$$||\nabla_x \mathcal{L}_{\ell_1}(x)|| \underset{F \to f_{\text{ens}}}{\geq} ||\nabla_x \mathcal{L}_{\text{SL}}(x)||. \tag{21}$$

Proof. Each term in the norm of gradients of ℓ_1 norm loss in Eq. 10 can be given by

$$\forall i,k \quad ||\text{sign}(u^i - \frac{1}{|\mathcal{K}|}\sum_j v_j^i)(\frac{\partial u^i}{\partial x} - \frac{\partial v_k^i}{\partial x})|| = ||\frac{\partial u^i}{\partial x} - \frac{\partial v_k^i}{\partial x}||. \tag{22}$$

For the SL loss, when the global model converges to the ensemble of the on-device models, each term in the norm of gradients in Eq. 18 can be given by

$$\begin{aligned}
\forall i,k \quad & ||\text{sign}(U^i - \frac{1}{|\mathcal{K}|}\sum_j V_j^i)(\frac{\partial U^i}{\partial x} - \frac{\partial V_k^i}{\partial x})|| && (23)\\
= \quad & ||\frac{\partial U^i}{\partial x} - \frac{\partial V_k^i}{\partial x}|| && (24)\\
= \quad & ||\frac{\partial U^i}{\partial u}\frac{\partial u^i}{\partial x} - \frac{\partial V_k^i}{\partial v}\frac{\partial v_k^i}{\partial x}||, \\
\approx \quad & ||\frac{\partial U^i}{\partial u}(\frac{\partial u^i}{\partial x} - \frac{\partial v_k^i}{\partial x})|| \quad (\text{Lemma 2}), \\
\leq \quad & ||\frac{\partial u^i}{\partial x} - \frac{\partial v_k^i}{\partial x}|| \quad (\text{Lemma 1}), && (25)
\end{aligned}$$

where the last two derivations follow the Lemma 2 and Lemma 1, respectively. Thus, by summing up the terms over i and k, we can expect the gradients of the SL loss is smaller than the ℓ_1 norm loss. This completes the justification for Hypothesis 2. However, to rigorously proof this, we need further assumptions on the data distribution and models.

For the KL-divergence loss, when the global model converges to the ensemble of the on-device models, each term in the norm of gradients in Eq. 13 can be given by

$$\forall i,k \quad ||\delta_i(\frac{\partial U^i}{\partial x} - \frac{\partial V_k^i}{\partial x})||. \tag{26}$$

Thus, by comparing Eq. 24 and Eq. 26, we can expect the gradients of KL-divergence loss is smaller than that of the SL loss This completes the justification for Hypothesis 1. □

APPENDIX: ON-DEVICE MODEL ARCHITECTURES

To make the experimental validation concise and easy to follow, we detail the setting of on-device model architectures in Section IV-C2 here.

For the ten devices trained with the CIFAR-10 dataset, we use ShuffleNetV2 and MobileNetV2 with different numbers of filters and a LeNet-like model architectures to meet diverse device capacity for on-device models[2]. Table V presents the detailed model architectures. We use SGD with a learning rate 0.01 and a weight decay of 0.0005. The batch size and the total training epoch are set to be 256 and 100, respectively.

[2]We implement the models based on https://github.com/kuangliu/pytorch-cifar.