FedKNOW

Federated Continual Learning with Signature Task Knowledge Integration at Edge

Luopan, Yaxin; Han, Rui; Zhang, Qinglong; Liu, Chi Harold; Wang, Guoren; Chen, Lydia Y.

DOI
10.1109/ICDE55515.2023.00033

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 IEEE 39th International Conference on Data Engineering (ICDE)

2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00033

FedKNOW: Federated Continual Learning with Signature Task Knowledge Integration at Edge

Yaxin Luopan, Rui Han, Qinglong Zhang, Chi Harold Liu, Guoren Wang
Beijing Institute of Technology, Beijing, China
Beijing, China
{1120181200, hanrui, 3120211050, chiliu, wanggr}@bit.edu.cn

Lydia Y. Chen
TU Delft
Delft, Netherlands
{lydiaychen@ieee.org

***Abstract*—Deep Neural Networks (DNNs) have been ubiquitously adopted in internet of things and are becoming an integral of our daily life. When tackling the evolving learning tasks in real world, such as classifying different types of objects, DNNs face the challenge to continually retrain themselves according to the tasks on different edge devices. Federated continual learning is a promising technique that offers partial solutions but yet to overcome the following difficulties: the significant accuracy loss due to the limited on-device processing, the negative knowledge transfer caused by the limited communication of non-IID data, and the limited scalability on the tasks and edge devices. In this paper, we propose FedKNOW, an accurate and scalable federated continual learning framework, via a novel concept of signature task knowledge. FedKNOW is a client side solution that continuously extracts and integrates the knowledge of signature tasks which are highly influenced by the current task. Each client of FedKNOW is composed of a knowledge extractor, a gradient restorer and, most importantly, a gradient integrator. Upon training for a new task, the gradient integrator ensures the prevention of *catastrophic forgetting* and mitigation of *negative knowledge transfer* by effectively combining signature tasks identified from the past local tasks and other clients' current tasks through the global model. We implement FedKNOW in PyTorch and extensively evaluate it against state-of-the-art techniques using popular federated continual learning benchmarks. Extensive evaluation results on heterogeneous edge devices show that FedKNOW improves model accuracy by 63.24% without increasing model training time, reduces communication cost by 34.28%, and achieves more improvements under difficult scenarios such as large numbers of tasks or clients, and training different complex networks.**

***Index Terms*—Federated learning, continual learning, deep neural networks, communication**

I. Introduction

Today, billions of mobile and Internet of Things (IoT) devices generate zillions bytes of data at the network edge, offering opportunities to deploy artificial intelligence (AI) locally on edge devices[1]. Such on-device AI applications, e.g. deep neural networks (DNNs), have the advantage of avoiding transmitting raw data and hence preserving data privacy [63]. At the same time, the arising new challenge is that the environment continuously evolves, requiring the DNN models to retrain and adapt to those changes [6]. For example, Figure 1 illustrates the DNN model in client 1 needs to handle a sequence of tasks (e.g. image classification or object detection) over time. Typically, a **task** is composed of multiple classes/objects (e.g. different animals or vehicles) and different features for each class [41].

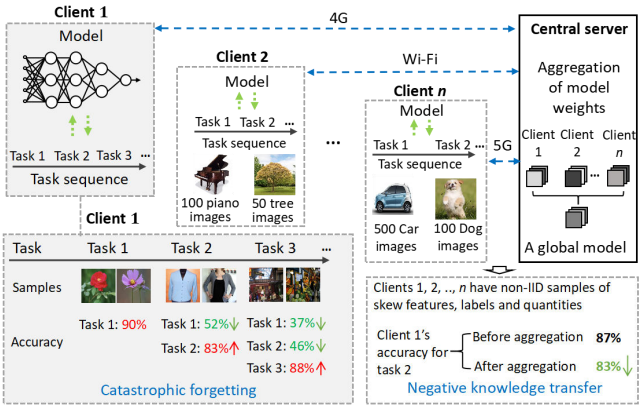

Fig. 1. An example scenario of federated continual learning with n clients

Federated continual learning (FCL). Continual learning is a prevalent technique that incrementally learns deep models from such a non-stationary data consisting of different tasks. Traditional continual learning only learns its models from the training samples on their hosted devices. In contrast, humans can learn from their own and others' past experiences through conversations, lectures, books and other means. Motivated by the intuition of learning from other clients' (indirect) experience, FCL combines continual learning in the federated learning framework such that the model in a client can continuously learn from its local data and the knowledge of the tasks in other clients [56], [58]. As shown in the exemplary scenario of Figure 1, a central server obtains the model weights/parameters locally trained in n clients, aggregates them into a global model, and sends it back to all clients. This allows each client to perform continual learning of its task sequence based on its local data, while learning from other clients by communicating their task-specific weights via the server. One major problem of performing continual learning in a client is **catastrophic forgetting**: when its model learns new tasks over

This work is supported by the National Natural Science Foundation of China (Grant No. 61872337, 62132019, 62272046) and Shandong Provincial Natural Science Foundation (Grant No. ZR2020MF034). Corresponding author: Rui Han.

[1]We also refer to such edge devices as clients.

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00033

time, it may forget previously learned task information and the model accuracy in these tasks degrades [8], [11], [44]. The challenge of coping with evolving task is further exacerbated when training DNN models according to evolving tasks on a large number of clients. In FCL, each client has its private sequence of tasks. Even for the same task, different clients host *non-IID datasets* whose distributions of classes, input data features, and numbers of samples vary [64]. This means although starting from the same global model, these clients have diverse models after local training. When aggregating a clients model with those of other clients, the model divergence may decrease the accuracy in local tasks, know as **negative knowledge transfer** [9], [21], [31], [62], [64].

Challenges of FCL at edge. Edge computing is developed to reduce communication costs to cloud servers and enhance data privacy via on-device data processing [47], [48]. Performing FCL at edge brings new problems such that the computation and communication costs in model training increase with the number of tasks and client, and conducting such expensive training on resource-constrained edge devices give rise to two technical challenges.

Limited computational resources lead to significant accuracy losses. Existing continual learning and FCL techniques are server-side solutions, which designed for powerful cloud servers and retain training samples or model weights of previous tasks from all clients to avoid catastrophic forgetting [2]–[4], [19], [24], [35], [42], [57], [58]. This means the learning process becomes longer when the number of tasks increases. For example, the training time of a ResNet-18 [15] increases by 50 times when the number of tasks increases from 1 to 80. When encountering resource constraints, these techniques can only retain a portion of samples and may incur large accuracy losses (20% to 50% losses) because most of important information in previous tasks is dropped. The **first challenge**, therefore, is to design a lightweight learning method that can keep extensive historical knowledge and take short model training time directly on resource-constrained edge devices.

Preventing negative knowledge transfer causes high communication costs and privacy leakage. Existing techniques rely on the central server which collects and keeps all clients' task models to prevent negative knowledge transfer [57], [58]. This mechanism causes high communication costs because: (i) the knowledge's size increases linearly with the number of clients; and (ii) all clients need to synchronize other clients' latest knowledge once any new task arrives. For instance, the communication traffic of FedWEIT [58] is eight times larger than that of the basic federate learning method when the client number is just 20. Maintaining the global knowledge among multiple clients also violates scalability and privacy enforcement of edge computing. The **second challenge** is how to develop a distributed method that can prevent negative knowledge transfer without increasing extra communications among clients.

In this paper, we depart from computationally and communication intensive FCL server-side approaches and propose FedKNOW, a lightweight client-side solution that integrates knowledge of signature tasks which encompass the relevant past and peer tasks. FedKNOW acts in each client and extracts compact and transferable knowledge (instead of data) – the critical subset of model weights. When learning a new task, FedKNOW integrates it with the knowledge of its **signature tasks**, which are the new task's most dissimilar tasks identified from local past tasks to prevent *catastrophic forgetting*, and the updated global model representing other clients' current tasks in preventing *negative knowledge transfer*. By completing knowledge integration with polynomial time complexity, FedKNOW addresses the limitations of existing techniques by providing both high model accuracy and low communication overhead at edge. In particular, the contributions of this paper are as follows:

Scalable client-side solution through the knowledge of signature tasks. In contrast the the prior art, FedKNOW is a client side method that acts on the knowledge of signature tasks, resulting into light-weight computation and communication for clients. FedKNOW extracts and retains each task's *knowledge* as a small proportion (e.g. 10%) of model weights with the highest degree of activation to the task.

High-accuracy model training via gradient integration. When learning a new task in a client, FedKNOW designs an optimization approach that integrates its gradient with gradients of previously experienced tasks, and integrates its gradients before and after global aggregation. Both integrations guarantee the acute angle between the integrated gradient and all other gradients. This gradient is then used in model updating to prevent catastrophic forgetting and negative knowledge transfer.

Convergence proof and evaluation. We prove the convergence of FedKNOW under the constraints of learning rates. We also fully implement FedKNOW on top of PyTorch to support deep learning applications on edge devices, and conduct extensive evaluation against the state-of-the-art techniques, i.e. continual learning, federated learning and federated continual learning, using popular continual learning benchmarks.

Summary of experimental results. (i) Extensible in terms of architecture. We test FedKNOW on five types of heterogeneous edge devices (Jetson TX2, Nano, Xavier NX, AGX, and Raspberry Pi). *(ii) High-accuracy continual learning*. We extensively compare FedKNOW to the 11 state-of-the-art techniques, and find that it increases model accuracy by 63.24% using similar or less model training time, for extremely challenging scenarios of 80 tasks or 100 clients. Compared to standard continual learning that uses all the data sample, FedKNOW achieves higher accuracy because of effective avoidance of negative knowledge transfer. *(iii) Low-overhead federated model training*. To complete the same model training jobs, FedKNOW reduces communication time by 34.28% compared to the latest FCL technique, especially FedWEIT. *(iv) Applicability to different scenarios and DNNs.* We ensure that our approach works well when the numbers of tasks and clients increase, and the condition of communication changes. We also demonstrate the applicability of FedKNOW

on 8 prevalent DNNs [22].

II. Related Work

The major problem faced by federated continual learning is the catastrophic forgetting in neural networks when learning new tasks [44]. This problem is further complicated by the negative transfer in federated learning due to the Non-IID data sets in different clients [31]. Here are existing techniques designed to address these problems.

Continual learning. Mainstream techniques designed to address catastrophic forgetting can be divided into three categories: (1) *memory rehearsal* uses a memory cache to store the samples of previous tasks and use them in learning the new task to avoid forgetting. Hence their computational costs increase with the number of tasks [4], [35], [42]. (2) *Regularization-based techniques* estimate different parameters/weights' contributions to a model and maintain part of information in important weights when learning new tasks [2], [3], [19], [24]. (3) *Dynamic architectural techniques* design different models for different tasks and solve catastrophic forgetting by isolating part of the model parameters [32], [45], [59]. However, it is difficult to apply these techniques in federated learning, because they require massive retained samples or weights to increase model generalization/accuracy, while restricting to learn tasks from the local data without benefiting from other clients' knowledge.

Federated learning trains a global model using private data from multiple clients [28], [30]. In a client's model training, negative transfer from other clients (due to their Non-IID datasets) is a crucial problem that delays training convergency and degrades model accuracy [31]. *Personalized federated learning* is a major technique used to mitigate such negative transfer and it can be divided into four types: (1) Mixture model techniques such as adaptive personalized federated learning (APFL) [9] dynamically change the ratio of global and local models in training. (2) In local fine-tuning techniques, each client first accepts a global model, and then updates it using local data. Meta-learning [5], [10], [23] is increasingly employed to complete the update within a few iterations. (3) Contextualization aims to provide a different model for each context, e.g., character's context [53]. (4) Multi-task learning lets each client train a separate task [49] and further classifies clients into different groups according to their tasks [20]. Note that the last two types of techniques cannot be directly applied in continual learning, because they compound learning new tasks with contexts and multi-task learning.

FCL. Some initial technique proposes server-side solution that maintains some training samples in the server and uses them in global model updating to avoid catastrophic forgetting [57]. The effectiveness of this work depends on the amounts of maintained samples and it is impractical to share clients' local data in the server due to data privacy [38]. The latest FCL technique, FedWEIT [58] uses adaptive model weights to maintain previous tasks' knowledge in a client and retains all clients' adaptive weights in the central server. Whenever a client needs to learn a new task, it first obtains the server's adaptive weights and then uses them in model training to prevent both catastrophic forgetting and negative transfer. The main limitation of FedWEIT is the scalability with respect to the number of clients and tasks due to their communication overhead.

III. Design of FedKNOW

A. Overview

We design FedKNOW to continually train sequences of different learning tasks on federated clients. FedKNOW features on a novel concept of signature task knowledge which further enables lightweight computation and communication on resource-constrained edge devices. As shown in Figure 2, FedKNOW acts in each federated client and is composed of three components: knowledge extractor, gradient restorer and gradient integrator. In FedKNOW, each client has its private sequence of tasks. Upon receiving a new task t_{m+1} in client j, the client needs to train for multiple iterations locally and then send back the trained model to the central server for global model aggregation. Suppose the knowledge of m previously learned tasks are retained ($m \geq 1$), FedKNOW trains the DNN model using r aggregation rounds. Each round consists of two parts: local training with v training iterations and global aggregation with the central server. FedKNOW is designed with three objectives.

Lightweight and scalable method. In order to effectively training tasks on federated clients with a large number of tasks and clients, FedKNOW is a client-side method that exploits the limited resource of edge devices by extracting critical knowledge and integrating those of signature tasks. The **knowledge extractor** retains each task t_i's knowledge W_i, which corresponds to a set of the most important model weights. The **gradient restorer** transforms W_i into gradient g_i that represents the model update direction that maximizes task t_i's accuracy. The above two components employ the weight-based pruning technique [13], [14] for two reasons: (1) it can filter and remove most of unimportant model weights and the retained weights in knowledge W_i can precisely restore task t_i's the gradient information; (2) the pruning phase is independent of the network architecture and completes quickly on edge devices. It is feasible to extend the above knowledge extraction and restoring process with structured pruning techniques such as L1-norm or L2-norm filter pruning [29]

Catastrophic forgetting prevention. At each training iteration, the **gradient integrator** takes task t_{m+1}'s original gradient g_{m+1} and the k gradients ($k \leq m$) of t_{m+1}'s most dissimilar previous tasks as inputs, and outputs an integrated gradient g' that has an acute angle with all input gradients. In a geometric term, this means updating the model using gradient g' does not increase the loss (i.e. decreasing the accuracy) of the task represented by any input gradient [35]. Among all previous tasks, the k tasks that are most dissimilar with task t_{m+1} are considered because the included angles between their gradients and g_{m+1} are the largest. The integration process is

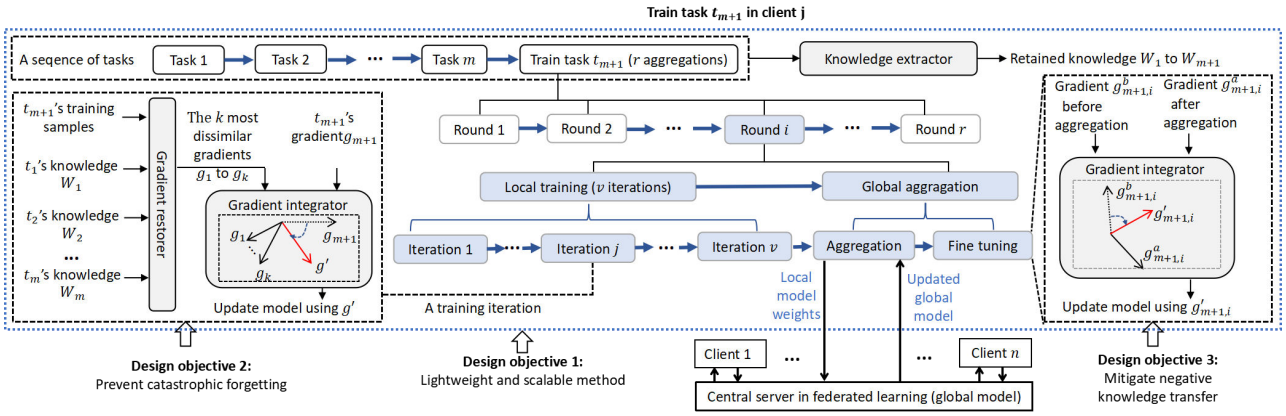

Fig. 2. FedKNOW process and its three design objectives

solved as an optimization problem that minimizes the rotated angle between g' and g_{m+1} and completes in polynomial time.

Negative knowledge transfer prevention. Following the standard federated learning setting [31], the global model in the central server starts from a random client's model. At aggregation round i, client j first uploads its local model weights to the server and obtains the updated global model after aggregation. After each global communication, FedKNOW fine tunes the model using one epoch of local samples. At each iteration, the **gradient integrator** takes the gradient $g^b_{m+1,i}$ before aggregation and $g^a_{m+1,i}$ after aggregation as inputs and outputs the integrated gradient $g'_{m+1,i}$ that has acute angles to both input gradients. Hence using $g'_{m+1,i}$ to update model can incorporate global information from other clients, while avoiding decreasing model accuracy in local data.

B. Knowledge Extractor

The knowledge extractor is designed with two purposes: the retained knowledge can keep most of the task's information and it can be quickly processed. The extraction process utilizes the weight-based pruning technique [13], [14] to remove most of the model weights whose absolute values are lower than a given threshold and retain the remaining ones (e.g. 10%) as the signature task knowledge. For example, this pruning technique can removes over 90% of the weights in VGG-16 [33] and causes negligible accuracy losses.

Formally, let W be the whole set of model weights for task t_i ($i \geq 1$), its knowledge W_i is defined as a proportion ρ of weights with the largest values: The knowledge extraction process has three steps: step 1 first trains the model until convergency; step 2 then selects a ratio ρ (e.g. 10%) of weights with the highest values to form t_i's knowledge W_i; finally, step 3 fine-tunes the weights in W_i while keeping other model weights unchanged.

$$W_i = \{w_i \mid w_i \in W \ and \ w_i \geq \rho\} \tag{1}$$

where ρ is a quantile that decides the ratio of top-ranked weights that are extracted as task t_i's knowledge.

C. Gradient Restorer

The knowledge restorer is designed for resource-constrained edge devices and it can produce a previous task t_i's gradient without storing its training samples. This component takes the retained knowledge $\{W_1, W_2,...,W_m\}$ of m tasks and the training samples (X_{m+1},Y_{m+1}) of current task t_{m+1} as inputs and outputs the gradients $\{g_1, g_2,...,g_m\}$ of the m tasks. Formally, let $loss()$ be the cross-entropy loss function (i.e. log loss function), task t_i's gradient is calculated as:

$$g_i = \nabla \ loss(f(W, X_{m+1}), f(W_i, X_{m+1})) \ (1 \leq i \leq m) \tag{2}$$

where $f(W, X_{m+1})$ represents the predicted labels of the current task, $f(W_i, X_{m+1})$ represents the predicted labels according to t_i's retained knowledge, and ∇ is the gradient operator of these two labels. In contrast, task t_{m+1}'s gradient is calculated using its ground truth labels Y_{m+1}: $g_{m+1} = \nabla \ loss(f(W, X_{m+1}), Y_{m+1})$.

With the increased number of tasks (that is, m is large), FedKNOW only selects the k gradients that are most dissimilar with task t_{m+1}'s gradient. That is, the distances (e.g. Wasserstein distance) between these gradients and g_{m+1} are the largest, hence these k gradients' corresponding tasks are mostly influenced by the model updating using gradient g_{m+1}. In training, only the selected k gradients are calculated to save computational costs. Note that in FedKNOW, parameters ρ and k are set according to hyperparameter search: a value is selected that it produces the highest model accuracy within certain memory or time constraint on edge devices. For the ratio ρ of retained weights, the constraint is the memory footprint of these weights. For the number k of gradients, the constraint is each task's computational time.

D. Gradient Integrator

The *gradient integrator* is developed to find a rotated gradient g' that decreases the loss of the current task t_{m+1} without increasing the losses of its signature tasks. These tasks are t_{m+1}'s k most dissimilar tasks in preventing catastrophic forgetting, and they are tasks from other clients in preventing

negative knowledge transfer. This requires the included angle between g' and any gradient g_i of these tasks being an acute angle [35], because these gradients decide the updating directions of model weights. If t_{m+1}'s original gradient g_{m+1} does not meet the above requirement, the integrator aims to minimize the rotation angle between g' and g_{m+1}, so as to maximize the learned knowledge of task t_{m+1}. Formally, let G=$\{g_1$ to $g_k\}$ be the set of previous gradients, the integrator employs the quadratic programming [35] to solve this optimization problem with polynomial time complexity:

$$\min_{g'} \ \frac{1}{2}||g_{m+1}, g'||_2^2 \quad s.t.\ Gg' \geq 0 \tag{3}$$

where $Gg' = |G||g|\cos\theta \geq 0$ means the included angle θ between any gradient in G and g' is an acute angle. In Equation (3), $\frac{1}{2}||g_{m+1}, g'||_2^2$=$\frac{1}{2}(g')^\top g' - g_{m+1}g' + \frac{1}{2}(g_{m+1})^\top g_{m+1}$, where $\frac{1}{2}(g_{m+1})^\top g_{m+1}$ is a constant and can be removed. Hence the gradient integrator solves the dual problem of Equation (3) as:

$$\min_{v} \ \frac{1}{2}v^\top GG^\top v + g^\top G^\top v \quad s.t.\ v \geq 0 \tag{4}$$

That is, the gradient integrator solves the dual optimization programming in Equation (4) to find v and calculates the integrated gradient as:

$$g' = G^\top v + g_{m+1} \tag{5}$$

E. Running Example

Figure 3 shows FedKNOW's model training process when a new task t_4 arrives, and the whole process has three global aggregation rounds and each round has three local training iterations. After learning each task, the *knowledge extractor* is applied to retain 10% of model weights as this task's knowledge. This examples selects one iteration and one aggregation round to illustrates how FedKNOW works.

At iteration 2 of round 2, FedKNOW prevents **catastrophic forgetting** based on the retained knowledge (W_1 to W_3) of the three previously learned tasks (t_1 to t_3). The *gradient restorer* first produces the labels of tasks t_1 to t_3 and uses these labels to compute gradients g_1 to g_3. It then calculates the Wasserstein distance between the gradient g_i $(1 \leq i \leq 3)$ and g_4, and selects the two most dissimilar gradients g_1 and g_2. Subsequently, the *gradient integrator* computes the included angles between g_4 and two selected gradients (g_1, g_2), and finds that angle between g_4 and g_1 is obtuse. This means directly updating the model weights according to gradient g_4 will increase the loss function of task t_1 and degrade the accuracy of this task. The integrator thus solves the quadratic programming problem to find the minimal rotation angle to adjust g_4. Finally, the adjusted gradient g is used in training.

After completing three training iterations of round 2, the local mode is uploaded to the central server for global aggregation. FedKNOW then performs a fine tuning of the updated global model. At each tuning iteration, the *gradient integrator* rotates each gradient before aggregation (e.g. $g^b_{4,2}$) such that it has an acute angle with the gradient after aggregation (e.g. $g^a_{4,2}$), and produces $g'_{4,2}$ that is used to update the model. This updating direction incorporates the global information from other clients, while avoid their **negative knowledge transfer** to the local model before aggregation.

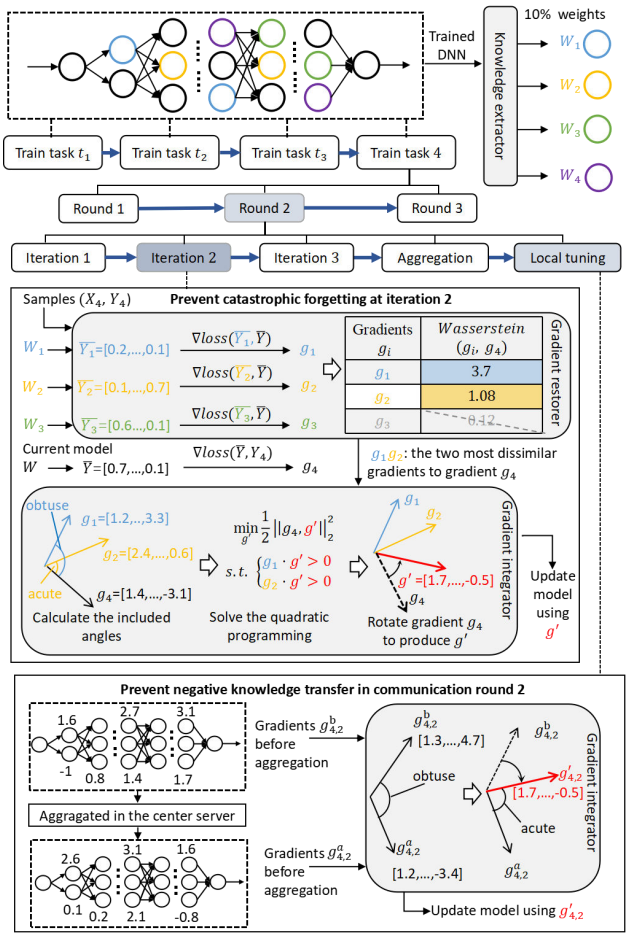

Fig. 3. Example model learning process with FedKNOW

IV. Proof of Convergency in FedKNOW

In this section, we prove the convergency of FedKNOW in the framework of federated learning and continual learning. Conceptually, the convergence means *the model weights/parameters can achieve the global optimum ones over the training process*. This work focuses on proving FedKNOW's convergency of model training in a client. For simplicity, we omit the index of the client in the following proof.

Definition of convergency. Let W be the set of model weights in a client, W^* be the optimal weights, and W_r be weights at iteration r $(r \geq 1)$, and $f(\cdot)$ be the label prediction

function of the model. During the iterative training process, the gap $H(r)$ at between these W_r and W^* is defined as:

$$H(r) = \sum_{i=1}^{r} f(W_r) - \min_{W} \sum_{i=1}^{r} f(W) \tag{6}$$

Given that r is usually a large number and the *training can converge* if $\frac{H(r)}{r}$ approaches 0. We convert Equation (6) as:

$$\lim_{r\to\infty} \frac{H(r)}{r} = \lim_{r\to\infty} \mathbb{E}[f(W_r)] - f(W^*) = 0 \tag{7}$$

where $\mathbb{E}$ is the mathematical expectation. In convergency proof, we compute the upper bound of $\lim_{r\to\infty} \frac{H(r)}{r}$ and shows that it approaches 0 under some constraints. The proof is based on the three assumptions in existing work [50], [61]

Assumption 1. *The expected squared norm of stochastic gradients is uniformly bounded, i.e.* $\mathbb{E}\|\nabla f(W_r, \xi_r)\|^2 \leq \lambda$, *where* ξ_r *is batch of training samples and* λ *is a constant*

Assumption 2. *The update of model parameters is bounded by a constant D:* $\|W_r - W_{r+1}\|_2 \leq D$.

In federated learning, suppose gradients follow assumptions 1 and 2, the upper bound of FedAvg [37] is given in assumption 3 [31].

Assumption 3. *In FedAvg, the training is bounded by:*

$$\mathbb{E}[f(W_r)] - f(W^*) \leq \frac{\tau}{\gamma + r - 1}\left(\frac{2B}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}||W_r - W^*||^2\right) \tag{8}$$

where $B = \sum_{i=1}^{N} p_i^2\sigma_i^2 + 6L\Omega + 8(r-1)^2\lambda^2$, *L,* μ *are constant,* σ_i *is the upper bound of gradient* g_r*'s variance,* λ *is the upper bound of* $(g_r)^2$, $\tau = \frac{L}{\mu}$, $\gamma = max\{8\tau, r\}$, p_i *denotes the weights of client i, and* $\Omega = f^*(W) - \sum_{i=1}^{n} p_i f_i(W^*)$ *denotes training data's degree of severity in terms of Non-IID.*

Proof steps. In FedKNOW, suppose $W_r = W_r^G \cup W_r^L$ consists of global weights W_r^G and local weights W_r^L, At iteration r, let η_r^G and η_r^L be the learning rates used in training weights W_r^G and W_r^L, respectively. The proof of FedKNOW's convergency has three steps: **Lemma 1** proves the upper bound of training W^G; **Lemma 2** proves the upper bound of training W^L; and finally **Theorem 1** proves the convergence of training the whole model W under the constraints of two learning rates η_r^G and η_r^L.

A. Upper bound of Training Local Weights W^L

Lemma 1. *Let* W_r^L *be the client's local weights at iteration* r *and* W^{L^*} *be the optimal local weights, the training of* W_r^L *is bounded by:*

$$\mathbb{E}[f(W_r^L)] - f(W^{L^*}) \leq \frac{D^2}{2\eta_r^L r} + \frac{\lambda^2\eta_j^L}{2} \tag{9}$$

Proof. Let $g_r = \nabla f(W_r^L, \xi_r)$ be the gradient at iteration r. According to Equation (6), the gap $H(r)$ between local weights W_r^L and W^{L^*} is:

$$\begin{aligned} H(r) &= \sum_{i=1}^{r} f(W_i^L) - \min_{W^L} \sum_{i=1}^{r} f(W_i^L) \\ &= \sum_{i=1}^{r} [f(W_i^L) - f(W^{L^*})] \end{aligned} \tag{10}$$

Suppose that $f(\cdot)$ is convex, we have:

$$f(W^{L^*}) \geq f(W_r^L) + \langle g_r, W_r^L - W^{L^*}\rangle \tag{11}$$

In model updating, W_r^L satisfies:

$$\begin{aligned} & W_{r+1}^L = W_r^L - \eta_r^L g_r \\ \longrightarrow\ & W_{r+1}^L - W^{L^*} = W_r^L - W^{L^*} - \eta_r^L g_r \\ \longrightarrow\ & ||W_{r+1}^L - W^{L^*}||_2^2 = ||W_r^L - W^{L^*} - \eta_r^L g_r||_2^2 \\ \longrightarrow\ & \langle g_r, W_r^L - W^{L^*}\rangle = \frac{1}{2\eta_r^L}(||W_r^L - W^{L^*}||_2^2 - \\ & \qquad ||W_{r+1}^L - W^{L^*}||_2^2) + \frac{\eta_r^L}{2}||g_r||_2^2 \end{aligned} \tag{12}$$

By combining Equations (10), (11), and (12) we have:

$$\begin{aligned} H(r) \leq & \sum_{j=1}^{r} \frac{1}{2\eta_j^L}(||W_j^L - W^{L^*}||_2^2 - ||W_{j+1}^L - W^{L^*}||_2^2) \\ & + \sum_{j=1}^{r} \frac{\eta_j^L}{2}||g_j||_2^2 \end{aligned} \tag{13}$$

We further lower the upper bound $H(r)$ of local weights W^L based on the Assumptions 1 and 2 and scale $H(r)$ as:

$$\begin{aligned} & H(r) \leq D^2\frac{1}{2\eta_j^L} + \frac{\lambda^2}{2}\sum_{j=1}^{r}\eta_j^L \\ \longrightarrow\ & \mathbb{E}[f(W_r^L)] - f(W^{L^*}) \leq \frac{D^2}{2\eta_r^L r} + \frac{\lambda^2\eta_j^L}{2} \end{aligned} \tag{14}$$

B. Upper bound of Training Global Weights W^G

Lemma 2. *Let* W_r^G *be the client's global weights at iteration* r *and* W^{G^*} *be the optimal global weights, the training of* W_r^G *is bounded by:*

$$\begin{aligned} & \mathbb{E}[f(W_r^G)] - f(W^{G^*}) \leq \\ & \frac{\tau}{\gamma + r - 1}\left(\frac{2B}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}||W_r^G - W^{G^*}||^2\right) \end{aligned} \tag{15}$$

where $B = \sum_{i=1}^{N} p_i^2\sigma_i^2 + 6L\Omega + 8(r-1)^2(g')^2$.

Proof. FedKNOW employs FedAvg [31] as the global parameter aggregation algorithm. Hence if its gradient g' follows the Assumptions 1 and 2, its training of global weights can be bounded by Equation (15) according to Assumption 3. We now proves the boundedness of g'.

In FedKNOW, g' is an integrated gradient of k previous gradients (G=$\{g_1$ to $g_k\}$) and the gradient g_r calculated using

the current task's training samples. That is, $g' = G^\top v + g_r$ ($v \geq 0$) according to Equation (5). According to Assumption 1, all gradients in G are bounded and g_r is a constant, we have $||g'||^2$ in Equation (15) is bounded:

$$\begin{aligned} ||g'||^2 &= max(||G^\top v + g_r||^2) \\ &= max(g_r^\top g_r + 2v^\top G g_r + 1) \end{aligned} \tag{16}$$

C. Convergence of Overall Model

Theorem 1. *In a client, FedKNOW can converge under two constraints: (1) its local weights's learning rate η^L decreases at the rate of $\mathcal{O}(r^{-\frac{1}{2}})$; and (2) its global weights' learning rate $\eta^G \leq \frac{2}{\mu(\gamma+r)}$ and it decreases at the rate of $\mathcal{O}(r^{-1})$:*

$$\lim_{r\to\infty} \mathbb{E}[f(W_r^L \cup W_r^G)] - f(W^*) = 0 \tag{17}$$

Proof. Let $W^* = W^{L^*} \cup W^{G^*}$, we convert Equation (17) as:

$$\begin{aligned} &\lim_{r\to\infty} \mathbb{E}[f(W_r^L \cup W_r^G)] - f(W^{L^*} \cup W^{G^*}) = 0 \\ \longrightarrow &\lim_{r\to\infty} (\mathbb{E}[f(W_r^L] - f(W^{L^*}) \cup (\mathbb{E}[f(W_r^G] - f(W^{G^*}) = 0 \end{aligned} \tag{18}$$

According to Lemma 1, $\lim_{r\to\infty} \mathbb{E}[f(W_r^L)] - f(W^{L^*})$ is bounded (Equation (9)) and this bound approaches to 0 if the learning rate η^L decreases at the rate of $\mathcal{O}(r^{-\frac{1}{2}})$ [27], [65]. Similarity, Lemma 2 states that $\lim_{r\to\infty} \mathbb{E}[f(W_r^G)] - f(W^{G^*})$ is bounded (Equation (15)) and this bound approaches to 0 if learning rate $\eta_r^G \leq \frac{2}{\mu(\gamma+r)}$ and decreases at the rate of $\mathcal{O}(r^{-1})$ [31]. ■

V. Evaluation

In this section, we evaluate the full implementation of FedKNOW on top of PyTorch [1] on exhaustive experimental scenarios against a wide set of data benchmarks and DNNs.

A. Experimental Settings

Testbed. We choose four types of heterogeneous edge platforms imposing different architectures to showcase FedKNOW's cross-platform nature when it comes to hardware: Jetson TX2 has 256-core NVIDIA Pascal GPU and 8 GB memory; Jetson Nano has NVIDIA Maxwell architecture with 128 NVIDIA CUDA cores and 4GB memory; Jetson Xavier NX has 384-core NVIDIA Volta GPU with 48 Tensor Cores and 16 GB memory; and Jetson AGX has 512-core Volta GPU with Tensor Cores and 32 GB memory. All Jetson platforms run Ubuntu 18.04.5 LTS and support DNNs in PyTorch 1.9.0 (Python 3.6.9).

Datasets and DNN models. We select five representative federated and continual datasets to evaluate FedKNOW. In these datasets, a *task* refers to an image classification task for a given set of objects. Following the setting of typical continual learning methods [43], the training/test points in each dataset are equally splited into each task and each class.

- *Cifar100 [25]* and *FC100 [40]* datasets both have 50k data points (training samples) from 100 classes and 10k testing points (100 ones per class). In continual learning, these data points belong to 10 tasks and each task has 10 classes.
- *CORe50 [34]* dataset has 165k data points from 550 classes and 55k testing points (100 ones per class). These data points belong to 11 tasks and each task has 50 classes.
- *MiniImageNet [52]* dataset has 50k data points from 100 classes and 10k testing points (100 ones per class). These data points belong to 10 tasks and each task has 10 classes.
- *TinyImageNet [26]* dataset has 100k data points from 200 classes and 10k testing points (50 ones per class). These data points belong to 20 tasks and each task has 10 classes.

In evaluation, the first three datasets are trained with a 6-layer CNN model [19] and the last two datasets are trained with the ResNet-18 model [15]. To evaluate FedKNOW's generalization capability on different network architectures, we also test it using 8 state-of-the-art DNNs with different depths, widths, multi-path, feature map exploitation mechanisms [22].

Task and dataset assignment in federated setting. We set the distributions of tasks and datasets following the setting of FedRep [7]. Each client has all tasks of a dataset and its distinct task sequence. To guarantees the data heterogeneously (non-IID) among different clients, we randomly allocates 2 to 5 of each task's classes to each client. For each class, we randomly selects 5% to 10% of the training samples in allocation.

Compared baselines. We implement and compare our approach with 11 state-of-the art techniques that can be divided into three categories

- Six *continual learning* methods: (1) gradient episodic memory (GEM) for continual learning [4], [35] calculates previous gradients and uses the included angle between them and the current gradient in model training. (2) Balanced Continual Learning (BCN) [42] retains the previous training samples and uses them to maximize the data distribution among different tasks and minimize the model training errors. (3) Contrastive Continual Learning(Co^2L) [3] focuses on feature transfer and maintains contrastive learned representations to mitigate catastrophic forgetting. (4) Elastic Weight Consolidation(EWC) [24] initially proposes the idea of regularization. It uses the Fisher information matrix to calculate the changes of model weights in different tasks and avoid drastic changes in weights. (5) Memory aware synapses(MAS) [2] improves this approach by estimating each weight's importance according to the output's sensitivity to this weight. And (6) adaptive Group Sparsity Based Continual Learning (AGS-CL) [19] applies different learning strategies for different tasks.
- Three *federated learning* methods: (1) FedAvg [37] is a typical approach that calculates each client's weight factor according to its number of training samples and uses these factors to aggregate the models of all clients. (2) APFL [9] dynamically changes the ratio of global and local models in training. And (3) FedRep [7] divides a model into presentation layers and head layers, and only communicates

presentation layers in federated learning, while adaptively training model weights in each client.

- Two *Federated continual learning* methods: (1) federate learning with continual local training (FLCN) maintains some training samples in the server and uses them in global model updating to avoid catastrophic forgetting [57]. (2) FedWEIT [58] is the latest technique that uses masks to divide model weights into base ones and adaptive ones, and maintains the adaptive weights of all clients and tasks as the previous knowledge. In each client, it obtains all clients' adaptive parameters and trains them together with the new task's weights based on the regularization method.

Evaluation Metrics. We consider both model accuracy and training time (hour) in evaluation. The accuracy metric is the top-1 accuracy on test data points: the top predicted class (the one with the largest probability) is the same as the actual class label. In a continual learning scenario, the reported accuracy of task t_m is the average accuracy of all m learned tasks.

B. Comparative Evaluations under Different FCL Scenarios

This section's evaluation compares *model accuracy* and *training time* between FedKNOW and 11 baseline techniques. To make our comparisons fair and avoid leakage of test data, we employ the prevalent benchmarking method [12] that searches hyperparameter using an additional test dataset (that is, SVHN [39] with two tasks and each class has 5 classes). For each dataset, this method searches the optimal hyperparameters that produce the highest accuracy on the SVHN dataset. The model training settings include two parts:

- **Common training settings for all techniques**. In comparison, the model is trained using the same initial weights, training samples, hyperparameters, and a cluster of 20 heterogeneous edge clients, including 2 Jetson AGX, 2 Jetson TX2, 8 Jetson Xavier NX, and 8 Jetson Nano platforms. In hyperparameter search, the search scopes of aggregation rounds and training iterations are 5 to 15 and 5 to 150, respectively. The search scopes of learning rates and decreases rates are {0.0005,0.0008,0.001,0.005} and {1e-6,1e-5,1e-4}. The scopes of both hyperparameters satisfy the condition of convergency in Section IV. For Cifar100, FC100, CORe50, MinyImageNet, and TinyImageNet workloads, the numbers of global aggregation rounds are set to 15, 15, 15, 10, and 5, respectively. Each round consists of 25 local local training iterations (i.e. 5 epoches). In these five workloads, the learning rates are set to 0.001, 0.001, 0.001, 0.0008, and 0.0008, and their decrease rates are set to 1e-4, 1e-4, 1e-4, 1e-5, and 1e-5, respectively.
- **Specific settings for some techniques**. In hyperparameter search of each baseline method, we set the lower and upper bounds of search space as 1/2 and 2 of the parameter value in its original evaluation. In memory-based continual learning methods(GEM, BCN, Co^2L), 10% of training samples are retained to avoid catastrophic forgetting. In regularization-based continual learning methods, the regularization hyperparameters are 40000 and 100 for EWC and MAS. In FLCN, we select 10% of training samples randomly to the server for updating the regularization parameters. In FedKNOW, the search space of ratio ρ of retained weights and number k of selected gradients in integration are $\{5\%, 10\%, 20\%\}$, and $\{5, 10, 20\}$, respectively. Ratio ρ is set to 10% because 20% of weights exceed the memory constraint of 4 GB and 10% produces a better accuracy than that of 5%. Gradient number k is set to 10 because it produces the highest accuracy within the time constraint (that is, each task's computational time is smaller than 20 minutes).

Comparison results. Figure 4 displays the comparison results of 12 techniques and we have three key observations.

Impact of catastrophic forgetting. Three federated learning baselines take less time to converge because these methods donot consider previous task information in model training, hence their model accuracies are lower than most of the other techniques due to catastrophic forgetting. This also explains the results that when the number of task increases, the accuracies of all techniques decrease. FedKNOW suffers least from the accuracy depredation because when learning a new task, it integrates its knowledge with the seen tasks that are most dissimilar from the current task model. In contrast, FedWEIT uses the maintained knowledge of all tasks (stored at the server) and may lower the influence of important tasks.

Impact of negative knowledge transfer. In a federated learning environment, the non-IID datasets in different clients also considerably influence model accuracy. The five continual learning baselines well address catastrophic forgetting, but suffer from negative knowledge transfer from other clients. For example, AGS-CL's loss function considers the changes in model weights. Hence the large changes in global model weights cause non-convergence in CORe50, MiniImageNet, and TinyImageNet datasets, and this observation is no observed in Figures 4(c), (g), and (h). In addition, FedWEIT has higher accuracies than other baseline methods in the first three datasets. However, its parameter decomposition strategy may harm the functionalities of some particular layers (downsample in ResNet) and thus its accuracies are lower in this DNN model (Figures 4 (g) and (h)). In contrast, FedKNOW achieves the highest accuracies in all settings thanks to its reliable gradient integration mechanism.

Impact of heterogeneous edge devices. We extend the above evaluation by adding 10 CPU-based devices (Raspberry Pi 4B) to the cluster with 20 Jetson devices. The Raspberry Pi devices consist of one with 2 GB memory, five ones with 4 GB memory, and four ones with 8 GB memory. Using the Cifar100, FC100, and CORe50 datasets, this evaluation compares the three techniques (GEM, FedWEIT, and FedKNOW) that produce the highest accuracies among all techniques. Figures 4(d), (e), and (f) show that: (i) training in resource-limited Raspberry Pi devices considerably delays the training time of all techniques by an average of 12 times. In particular, FedWEIT has the largest increase in training time because its global knowledge becomes larger with more tasks and clients. For example, FedWEIT's training time of the last task is 1 hour longer than the other two techniques. (ii) Resource

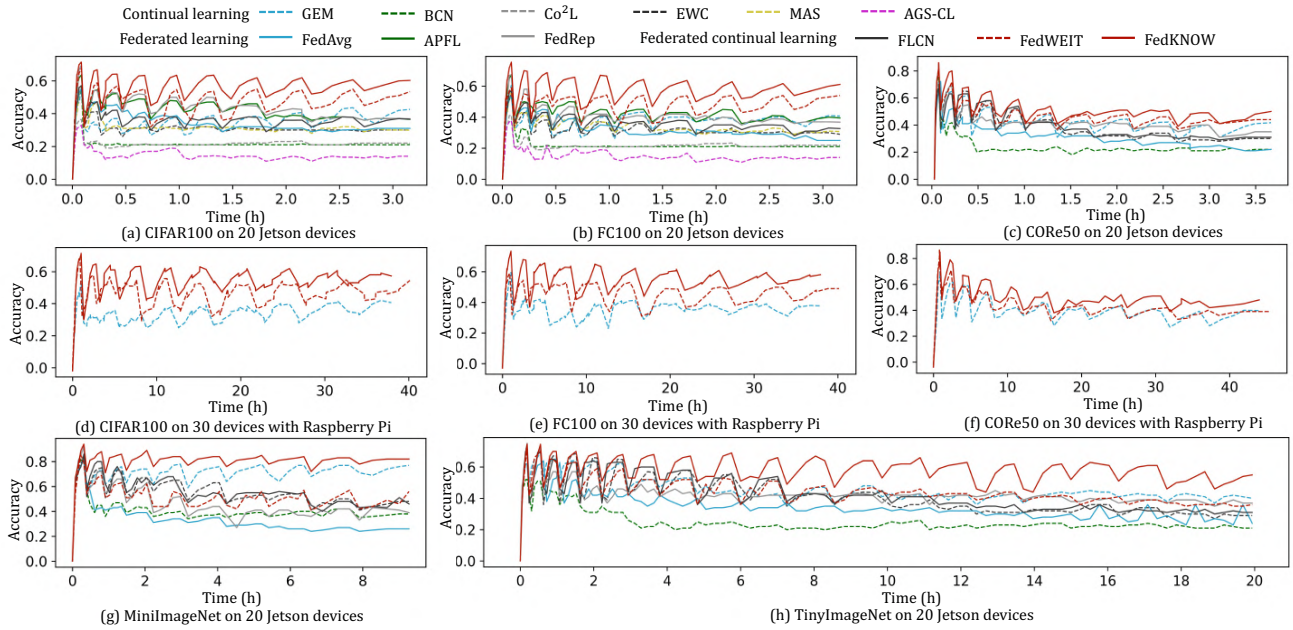

Fig. 4. Comparison of model accuracy and training time between FedKNOW and 11 baseline methods

heterogeneity decreases accuracies for all three techniques by 3% to 5%. The results show FedKNOW still achieves the highest accuracies because it is lightweight and integrates task knowledge locally. In contrast, FedWEIT requires each client using the heavyweight global knowledge, which makes the Raspberry Pi of 2 GB memory out of memory after learning 7 tasks and cannot participate in the following federated learning.

Table I summarizes the percentages of increase in the average accuracy, when comparing the accuracy of FedKNOW against the average accuracy of all 11 baselines techniques across 5 different datasets. For each dataset, the increased accuracy of each task is reported. We can see that when the task number increases, the percentage accuracy improvement increases from 10.21% to 98.72%. Overall, when considering all evaluation cases, our approach improves the accuracy by an average of 77.35%, 33.26%, and 31.27% compared to continual learning, federated learning, and FCL baselines, respectively.

C. Evaluation of Communication Cost

Following the evaluation settings of the previous section, this section's evaluation focuses on communication cost in model training. We compare our approach with FedWEIT because in this method, each client needs to obtain the retained adaptive weights of all other clients before learning a new task. Although these weights bring higher accuracies, they also incur large communication traffic that increases with the number of clients. In contrast, both our approach and other baseline methods employs the standard FedAvg method in federated learning and have the same communication cost.

TABLE I
A SUMMARY OF AVERAGE PERCENTAGE ACCURACY IMPROVEMENT

	CIFAR100	FC100	Corn50	Mini Imagenet	Tiny Imagenet		
Task1	36.52%	36.16%	17.31%	10.21%	16.00%	**Task11**	64.00%
Task2	74.74%	63.62%	38.80%	20.39%	18.31%	**Task12**	67.98%
Task3	82.58%	74.89%	32.54%	37.04%	20.62%	**Task13**	73.00%
Task4	84.84%	79.22%	46.21%	70.30%	28.22%	**Task14**	80.27%
Task5	88.69%	86.83%	45.39%	68.12%	33.00%	**Task15**	84.00%
Task6	94.87%	86.15%	37.27%	70.57%	35.09%	**Task16**	93.50%
Task7	92.40%	88.81%	37.06%	65.61%	45.00%	**Task17**	95.00%
Task8	95.52%	85.50%	51.84%	67.98%	54.46%	**Task18**	97.78%
Task9	98.72%	87.57%	56.65%	70.34%	52.00%	**Task19**	91.00%
Task10	97.75%	90.49%	54.72%	72.18%	61.49%	**Task20**	87.57%
Task11			63.22%				

Evaluation of different workloads. In federated learning, the communication cost among clients and the central server considerably impacts the model training performance (communication time takes about 10% to 30% of model training time). As shown in 5, FedKNOW takes much less communication cost when performing the same model training task. This is because our approach employs a distributed knowledge retaining mechanism that each client only uses its own knowledge to retain previous tasks. By contrast, FedWEIT applies a centralized mechanism that aggregates all clients' adaptive weights in the server and uses these as the knowledge. This means each time a client learns a new task, it needs to send its latest model weights to the server and all other clients need to obtain these weights from the server. We note that in FedWEIT, a client's own adaptive weights cannot represent its previous tasks. This is these parameters

are generated using regularization techniques and in a client, different tasks' adaptive weights have small differences. Hence FedWEIT needs to use other clients' adaptive weights to increase the model generalization in continual learning. Overall, our approach reduces communication cost by 34.28%.

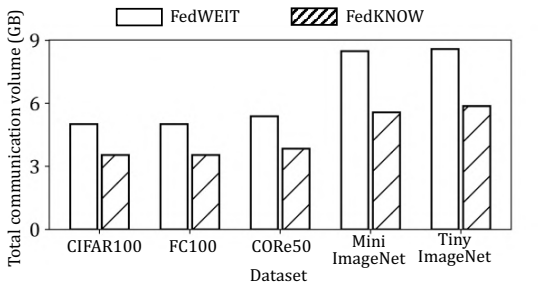

Fig. 5. Comparison of communication time under different workloads

Evaluation under different network bandwidths. In a distributed edge computing environment, network bandwidth is a key factor that influences communication time. In the previous evaluation, the network bandwidth limitation is 1 MB/second. We extended this evaluation be test 8 different network bandwidths, ranging from 50 KB per second to 10 MB per second, in each client. Figure 6 shows the communication time of two DNN models under different bandwidths. We can see that our approach consistently takes less communication time than FedWEIT. As expected, the communication time becomes longer when the network bandwidth decreases and our approach can save more communication time under tenser network conditions (up to 10 hours when the network bandwidth is 50 KB per second).

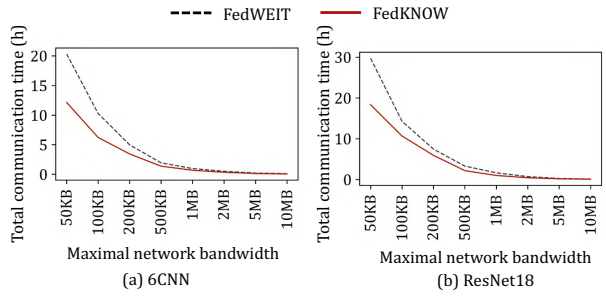

Fig. 6. Comparison of communication time under different network bandwidths

D. Discussion of Task and Client Numbers

In this section, we extended the previous evaluations to discuss two key factors in federated and continual learning scenarios: the number of task and the number of client. By taking FedWEIT, GEM, and FedKNOW as examples, our discussion demonstrates how these two factors effect the learning effectiveness. This is because FedWEIT and GEM perform best in all baselines: FedWEIT is developed specifically for the continual setting and GEM stores the past samples directly which sacrifices the efficiency to mitigate catastrophic forgetting. In evaluation, we report two metrics: *average accuracy* and *average forgetting rate* in all learned tasks. Suppose m tasks are learned, the forgetting rate of the kth task ($1 \leq k \leq m$) is defined as: the difference between t_k's accuracies after learning k and m tasks, divided by the former accuracy. We report the forgetting rate after learning a new task and its value ranges between 0 and 1.

Number of tasks. In this evaluation, we combine the tasks in MiniImageNet, Cifar100, and TinyImage workloads, and construct a dataset with 80 tasks. We still use FedRep's method [7] to distribute this dataset to 20 clients to guarantee data heterogeneity, and use ResNet-18 to learn these tasks. Figure 7 shows the fluctuations of accuracies and forgetting rates when the number of tasks increases from 1 to 80. The comparison results show that: (i) FedKNOW consistently provides higher latencies. This result verifies that our approach maintains the most of previous tasks' knowledge among three methods. (ii) When the job number increases, the model accuracies have apparent decreases in all methods. This result can be explained by these methods' forgetting rates: ResNet-18 has a limited generalization capacity and hence its forgetting rate continuously increases when more task knowledge is learned.

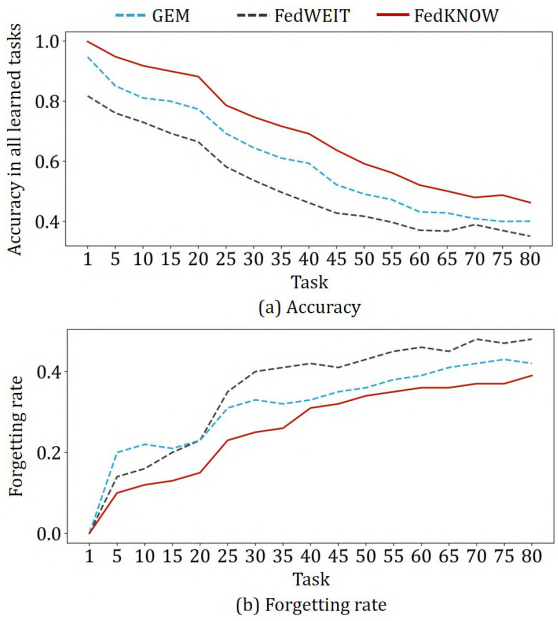

Fig. 7. Discussion of model accuracies under different numbers of tasks

Number of clients. Two cluster scales are considered in this evaluations: 50 and 100 clients. This is because when distributing the MiniImageNet dataset to these clients. Each client only has a small number of training samples when the client number is 100. Figure 8 displays the comparison of three methods. We can see that for both client numbers, FedKNOW can achieve the highest accuracies (Figure 8(a)) and our approach

has the lowest forgetting rates (Figure 8(b)). This is because when the client number becomes larger and the training samples among different clients become more heterogeneous, the negative knowledge transfer becomes more severe. Our approach optimizes the gradient integration process and tries to minimize the influence of such negative transfer when learning new tasks, thus gaining more accuracy improvement when the degree of data Non-IID increases. In most of the cases, GEM has higher accuracies than FedWEIT. This is because FedWEIT decomposes the weights of each layer into adaptive and base ones. However, ResNet-18 has several downsample layers with few but important weights, and the decomposition these layers degrades the model accuracy. In contrast, GEM maintains the architecture of the model and uses retained samples that provide higher accuracies than FedWEIT.

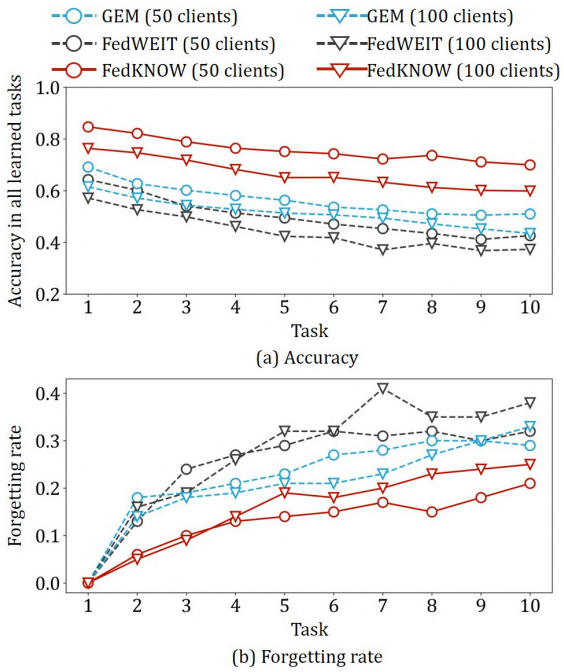

Fig. 8. Discussion of model accuracies under different numbers of clients

E. Applicability in Different DNNs and Settings

Applicability of FedKNOW to DNNs. FedKNOW represents the first framework that supports knowledge-level federated continual learning of DNNs for edge-based environment. FedKNOW can be generalized to support most of state-of-the-art DNNs [22]. To support this claim, we implemented and tested 8 DNNs beloning to six typical categories: (1) depth (ResNet-152 [16]); (2) multi-path (DenseNet [18]); (3) width (InceptionV3 [51], ResNeXt [55], and WideResNet [60]); (4) feature map exploitation (SENet18 [17]); (5) attention (ResNet-152 [54] and SENet18 [17]); and (6) lightweight DNN (MobileNetV2 whose width multiplier is 1.0 and width multiplier is 2.0 [46] and ShuffleNetV2 [36]). The evaluation settings of these models follow Section V-B.

Figure 9 illustrates the comparison results when applying GEM, FedWEIT, and FedKNOW to learn the 10 tasks in MiniImageNet. The results show that when a new task comes, our method can re-train all these DNN models to maintain high accuracies for all learned tasks. Although the accuracies in all three methods decrease when the task number increases, our method provides the highest accuracies in all cases. This is because our knowledge extraction and integration mechanism maintains and restores the most task knowledge in the continual learning process. In addition, we can see different models have different generalization capabilities and thus have different accuracies when learning the same tasks. For example, when comparing to the ResNet-18 model used in previous evaluations (Figure 4(d)), small models such as MobileNetV2 have lower accuracies (Figure 9(e)), while the large models such as WideResNet50, ResNeXt50, ResNet-152, and SENet18 ((Figures 9(a) to (d))) have considerable higher accuracies. The wide applicability of FedKNOW makes it possible to use large models on small edge devices to deliver high accuracies.

In contrast, FedWEIT has the lowest accuracies when training these models. The main reason comes from the design of its loss function, which consists of three components: (1) the task loss; (2) the sparsity regularization penalty term for task-adaptive weights and masks; (3) the difference between the weights of two consecutive time steps. In training, two hyper-parameters are used to balance these components. However, in this evaluation, different models have considerably different architectures and numbers of weights (for example, the size of ResNet152 is 117 MB while the size of MobileNetV2 is only 3.5 MB). This means the two hyper-parameters have to be carefully tuned for each model in its training. Moreover, for compact networks such as MobileNetV2, ShuffleNetV2, and ResNeXt, sparse model weights may have no volume to store much knowledge. This further degrades the accuracy of FedWEIT.

Discussion of different parameter settings. In this evaluation, we discuss three typical methods to retain previous knowledge in continual learning and show the impact of different parameter settings in these methods: (1) GEM stores 10%, 20%, 50%, 100% of each task's training samples and uses them when learning new tasks. (2) FedWEIT decomposes and regularizes adaptive weights from each task's learned model. In evaluation, we consider two settings: FedWEIT uses adaptive weights from all clients (the original setting) and it only uses adaptive weights from its own tasks in one client. (3) FedKNOW extracts the most important weights of each task as its knowledge. Three percentages of weights with the largest values are tested: 5%, 10%, and 20%. MiniImageNet and ResNet-18 are tested here. Figure 10 shows the model accuracies and training time of the three methods under different settings and we have two key observations.

Model accuracy. The results in Figure 10(a) show that retaining less information (samples or model weights) indeed causes lower accuracies in each method. We can see that even when GEM uses 100% of previous training samples, it still

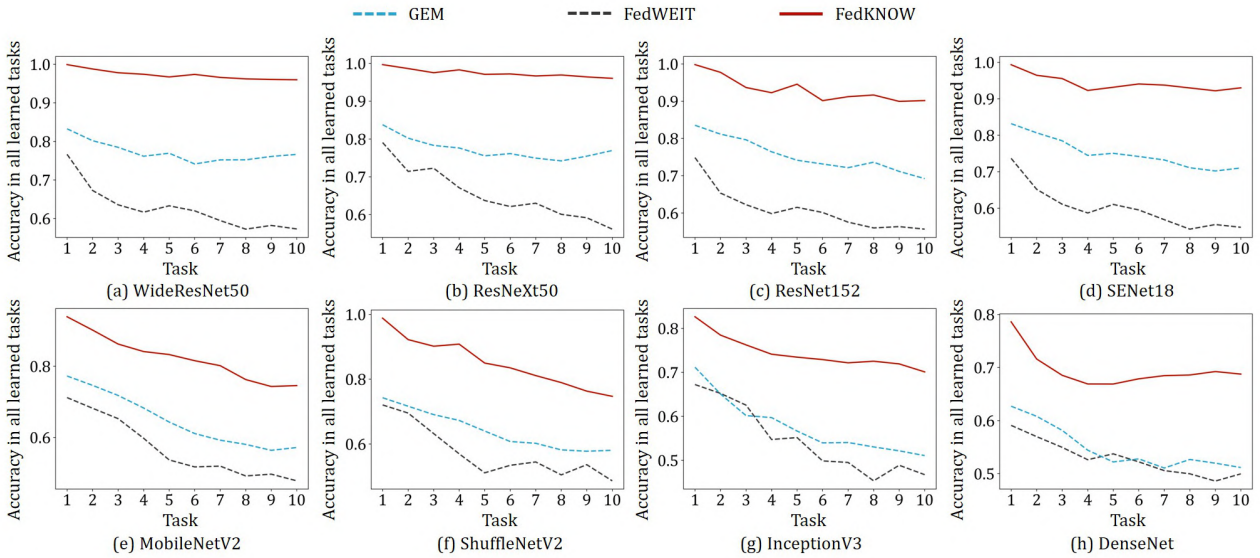

Fig. 9. Applicability of FedKNOW to six categories of DNNs

provides lower accuracies than FedKNOW in most of the cases. This is because GEM does not consider negative knowledge transfer from other clients' Non-IID data. In contrast, FedKNOW optimizes the gradient integration to minimizes the influence of negative transfer while preventing forgetting previous tasks' knowledge. This allows our approach to avoid the influence of uncertain knowledge transfer from other clients in real scenarios, in which the clients participating in learning the model may dynamically change over time.

Model training time. As expected, Figure 10(b) shows retaining and processing more previous information takes longer training time. FedWEIT only using one client's adaptive weights takes the shortest time because it processes the least information, which is insufficiently to maintain previous knowledge and hence this method has the lowest accuracy. GEM suffers most from processing more training samples. For example, the model training time on 100% previous samples is 2 to 3 times longer when only 10$ of previous samples are used. In contrast, the differences of model training time among three knowledge sizes are much smaller in FedKNOW. This allows our approach to use larger knowledge to increase model accuracy.

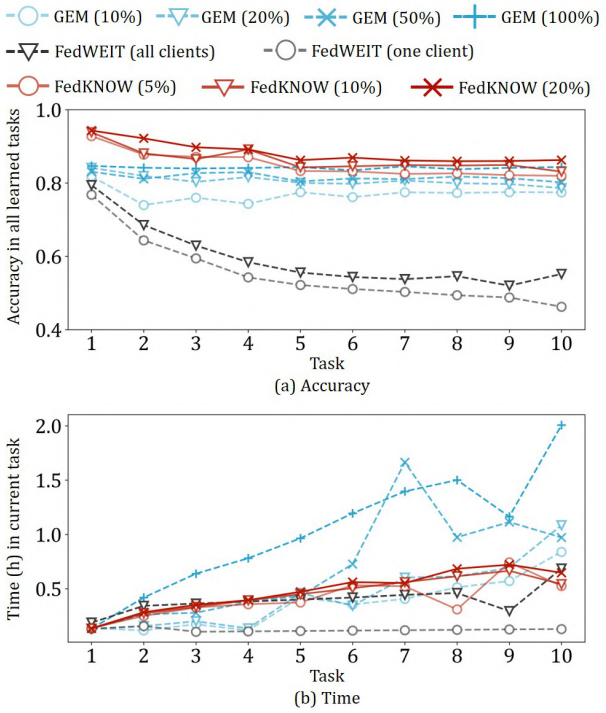

Fig. 10. Discussion of different parameter settings

VI. Conclusion

This paper presents the design, implementation and evaluation of FedKNOW, a framework that enables accurate and communication-efficient federated continual learning on distributed edge devices. FedKNOW is based on a novel optimization to integrate signature task knowledge. Our approach extracts and retains task knowledge from all learned tasks, while optimally assembling the most important knowledge to adapt to a new task with high accuracy and low communication overheads. Extensive evaluations in real scenarios against latest federated continual learning techniques strongly prove the efficacy and practicality of FedKNOW, especially for challenging learning scenarios of 80 different tasks and 100 clients.

REFERENCES

[1] Pytorch. https://pytorch.org/, 2022.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV'2018*, pages 139–154, 2018.

[3] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *ICCV'2021*, pages 9516–9525, 2021.

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[5] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*, 2018.

[6] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[7] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *ICML'2021*, pages 2089–2099. PMLR, 2021.

[8] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[9] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML'2017*, pages 1126–1135. PMLR, 2017.

[11] Stephen T Grossberg. *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, volume 70. Springer Science & Business Media, 2012.

[12] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

[13] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*, 2016.

[14] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR'2016*, pages 770–778, 2016.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR'16*, pages 770–778. IEEE Computer Society, 2016.

[17] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(8):2011–2023, 2020.

[18] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR'17*, pages 2261–2269. IEEE Computer Society, 2017.

[19] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. *Advances in Neural Information Processing Systems*, 33:3647–3658, 2020.

[20] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[21] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for on-device federated learning. 2019.

[22] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, 2020.

[23] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. *Advances in Neural Information Processing Systems*, 32, 2019.

[24] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[26] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[28] Anran Li, Lan Zhang, Junhao Wang, Juntao Tan, Feng Han, Yaxuan Qin, Nikolaos M Freris, and Xiang-Yang Li. Efficient federated-learning model debugging. In *ICDE'2021*, pages 372–383. IEEE, 2021.

[29] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR'17*, 2016.

[30] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[31] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *ICML'19*, 2019.

[32] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[33] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *ACPR'15*, pages 730–734, 2015.

[34] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *CoRL'2017*, pages 17–26. PMLR, 2017.

[35] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[36] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: practical guidelines for efficient CNN architecture design. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *ECCV'18*, volume 11218 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2018.

[37] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AIR'2017*, pages 1273–1282. PMLR, 2017.

[38] H Brendan Mcmahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera Y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS'17*, pages 1273–1282, 2017.

[39] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[40] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*, 31, 2018.

[41] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[42] Krishnan Raghavan and Prasanna Balaprakash. Formalizing the generalization-forgetting trade-off in continual learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[43] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *ICCV'17*, pages 2001–2010, 2017.

[44] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[45] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[46] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR'18*, pages 4510–4520, 2018.

[47] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[48] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[49] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

[50] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR'16*, pages 2818–2826. IEEE Computer Society, 2016.

[52] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[53] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

[54] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *ECCV'18*, volume 11211 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2018.

[55] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR'17*, pages 5987–5995. IEEE Computer Society, 2017.

[56] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):12, 2019.

[57] Xin Yao and Lifeng Sun. Continual local training for better initialization of federated models. In *ICIP'2020*, pages 1736–1740. IEEE, 2020.

[58] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *ICML'2021*, pages 12073–12086. PMLR, 2021.

[59] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[60] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *BMVC'16*, pages 87.1–87.12. BMVA Press, September 2016.

[61] Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. *Advances in neural information processing systems*, 25, 2012.

[62] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[63] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[64] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

[65] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML'03*, pages 928–936, 2003.