FedViT

Federated continual learning of vision transformer at edge

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

DOI
10.1016/j.future.2023.11.038

Publication date
2024

Document Version
Final published version

Published in
Future Generation Computer Systems

Future Generation Computer Systems 154 (2024) 1–15

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

FedViT: Federated continual learning of vision transformer at edge

Xiaojiang Zuo [a], Yaxin Luopan [a], Rui Han [a,*], Qinglong Zhang [a], Chi Harold Liu [a], Guoren Wang [a], Lydia Y. Chen [b]

[a] Beijing Institute of Technology, Beijing, China
[b] TU Delft, Delft, Netherlands

ARTICLE INFO

Keywords:
Catastrophic forgetting
Continual learning
Edge computing
Federated learning
Knowledge transfer negative
Vision transformer

ABSTRACT

Deep Neural Networks (DNNs) have been ubiquitously adopted in internet of things and are becoming an integral part 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 (FCL) 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 (non-Independent and Identically Distributed) data, and the limited scalability on the tasks and edge devices. Moreover, existing FCL techniques are designed for convolutional neural networks (CNNs), which have not utilized the full potential of newly emerged powerful vision transformers (ViTs). Considering ViTs depend heavily on training data diversity and volume, we hypothesize ViTs are well-suited for FCL where data arrives continually. In this paper, we propose FedViT, an accurate and scalable federated continual learning framework for ViT models, via a novel concept of signature task knowledge. FedViT 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 FedViT 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 FedViT 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 FedViT improves model accuracy by 88.61% without increasing model training time, reduces communication cost by 61.55%, and achieves more improvements under difficult scenarios such as large numbers of tasks or clients, and training different complex ViT models.

1. Introduction

Today, billions of mobile and Internet of Things (IoT) [1] 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 [2]. 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 [3]. For example, Fig. 1 illustrates a prevalent DNN model – vision transformer (ViT) [4] – 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 musical instruments) and different features for each class [5].

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

* Corresponding author.
E-mail addresses: 691301967@qq.com (X. Zuo), 3120220955@bit.edu.cn (Y. Luopan), hanrui@bit.edu.cn (R. Han), 3120211050@bit.edu.cn (Q. Zhang), chiliu@bit.edu.cn (C.H. Liu), wanggr@bit.edu.cn (G. Wang), lydiaychen@ieee.org (L.Y. Chen).

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

https://doi.org/10.1016/j.future.2023.11.038
Received 1 May 2023; Received in revised form 21 November 2023; Accepted 22 November 2023
Available online 27 December 2023

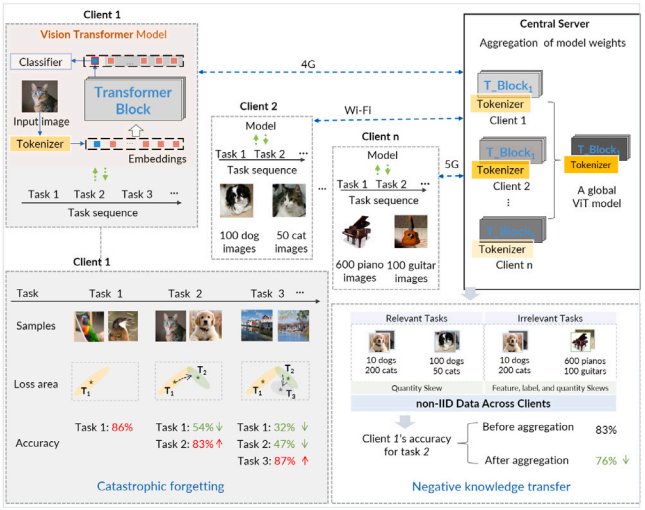


Fig. 1. An example scenario of federated continual learning with n clients.

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 [6,7]. As shown in the exemplary scenario of Fig. 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 time, it may forget previously learned task information and the model accuracy in these tasks degrades [8–10]. Taking the local learning process of client 1 in Fig. 1 as an illustration, once the model within client 1 completes the training for its initial task (i.e. task (1), its parameters converge to the optimal point of the task's loss area, producing an accuracy of 86%. However, when subsequently adapting to the second task (i.e. task (2) using the same model, the parameters gradually shift away from the task 1's optimal point and progressively approach the optimal state of task 2, due to the limited correlation between the two tasks. Consequently, despite achieving 83% accuracy on task 2, the model's classification accuracy for task 1 experiences a significant drop to 54%, exemplifying the phenomenon of catastrophic forgetting for the previous task. 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. When learning tasks are unrelated, their datasets are severely non-IID. Even for the same task, different clients also host *non-IID datasets* whose distributions of classes, input data features, and numbers of samples may vary [11]. This means that although starting from the same initialized global model, the local models on different clients diverge after separate local training on their distinct data distributions. This divergence presents a significant challenge when a client's model is aggregated with models from other clients in federated learning. The dissimilar knowledge in other clients' models may degrade the performance of the model on its specialized local data. This detrimental transfer of irrelevant knowledge from the global model is termed **negative knowledge transfer** [11–15], as shown in Fig. 1. Besides the two challenges, existing FCL works neglect the potential of using newly emerged powerful vision transformers (ViTs) [4] in this scenario. ViT is a transformer [16] model applied to the Computer Vision (CV) area, it exhibits strong robustness to data distribution drift [17] which is the key issue faced by continual leaning and federated learning. Moreover, the performance of ViT is highly dependent on the quality of training data [18], such as data volume and data diversity. In FCL, as the number of tasks increases, it naturally provides such data resource to ViT. Therefore, we focus on and investigate the performance of ViT models in FCL.

Challenges of ViT in FCL at edge. Edge computing is developed to reduce communication costs to cloud servers and enhance data privacy via on-device data processing [19–21]. Performing FCL at the edge brings new problems such that the computation and communication costs in model training increase with the number of tasks and clients, and conducting such expensive training on resource-constrained edge devices gives rise to two technical challenges.

Limited computational resources lead to significant accuracy losses. Existing continual learning and FCL techniques are server-side solutions, which are designed for powerful cloud servers and retain training samples or model weights of previous tasks from all clients to avoid catastrophic forgetting [7,22–29]. This means the learning process becomes longer when the number of tasks increases. For example, the training time of a ResNet-18 [30] 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 the important information in previous tasks is dropped. This phenomenon is more severe when using ViT, because ViT is a computation-intensive and data-hungry model [4,31]. Training ViT without sufficient iterations and data samples leads to quite low accuracy, making it unreliable to act as the task's knowledge. Moreover, storing a large number of samples or model weights significantly increases the training cost of ViT model on the client side. Therefore, the **first challenge**, 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 [7,29]. 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 [7] is eight times larger than that of the basic federated learning method when the client number is just 20. ViT's performance depends on the quality of training data and the number of model weights, therefore, frequent transmission with the central server leads to significant communication cost. 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 FedViT, a lightweight client-side solution that integrates knowledge of signature tasks which encompass the relevant past and peer tasks. FedViT acts in each client and extracts compact and transferable knowledge — the important data samples. When learning a new task, FedViT 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, FedViT 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 to the prior art, FedViT is a client side method that acts on the knowledge of signature tasks, resulting into light-weight computation and communication for clients.

- **High-accuracy model training via gradient integration**. When learning a new task in a client, FedViT 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 FedViT under the constraints of learning rates. We also fully implement FedViT 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 FedViT on 5 types of heterogeneous edge devices (Jetson TX2, Nano, Xavier NX, AGX, and Raspberry Pi). *(ii) High accuracy under resource-constrained continual learning*. We compared FedViT to 11 state-of-the-art techniques, and find that FedViT presents significant accuracy gaps among them, demonstrating its effectiveness to prevent negative knowledge transfer and catastrophic forgetting. *(iii) Communication-efficient federated model training*. To complete the same model training jobs, FedViT reduces communication time and volume compared to the latest FCL technique, especially FedWEIT. *(iv) Scalability*. 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 FedViT on 8 prevalent ViTs [32].

The remainder of this paper is organized as follows: Section 2 introduces the background and related work of this work, Section 3 explains our approach, Section 4 provides a convergence proof of our method, Section 5 evaluates it, and finally, Section 6 summarizes the work. Portions of this work appear in a previous conference paper [33] and we have largely extended the article, by discussing the advantages and key challenges when using ViT in FCL (see Section 1), giving a brief introduction to vision transformer (see Section 2.1), formulating the federated continual learning problem (see Section 2.2), expanding the related work with vision transformer and its applications in federated or continual learning (Section 2.3), developing new sample-based knowledge extraction (Section 3.2) method and gradient restore (Section 3.3) method to design a lightweight FCL approach for ViT, expanding the baselines and re-evaluating FedViT on all of the datasets and new ViT models (Section 5).

2. Background and related work

2.1. Vision transformer

Transformer [4,16,34] is an emerging neural network architecture based on self-attention mechanisms [16,35] to model global dependencies in sequential data. It captures long-range context by computing correlations between all input elements. The standard Transformer [16] consists of an encoder–decoder structure. The encoder maps an input sequence to a continuous representation through multi-head self-attention layers. The decoder generates the target sequence recursively from the encoder outputs. Transformer has shown powerful performance in Natural Language Processing (NLP) and Computer Vision (CV) areas. **Vision transformer** [4] is such a model applied in CV field. It splits an image into a sequence of tokens and captures the long-range dependencies among these tokens via self-attention mechanism. Besides, ViT is less constrained by "induction bias" [36], making it a flexible and powerful model with strong representation capabilities. Currently, ViTs have shown advantages in various computer vision tasks, such as image classification, object detection, and semantic segmentation.

Standard ViT consists of three components: patch tokenizer, transformer encoder, and classifier (as is shown in Fig. 1). When an image $x \in R^{(C\times W\times H)}$ is arrived, where W, H and C are width, height, and channel number, the patch tokenizer splits the image into N patches and maps them into a sequence of d dimension patch embeddings $x \in R^{N\times d}$ by a trainable linear projection.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V \tag{1}$$

Then, the transformer encoder (composed of a stack of transformer blocks) transforms the embeddings x into three parts (matrixes): Q, K, and V, and then calculates their attention scores (correlations) via the formula (1) (according to [4]) to generate new embeddings that contain global information of x. Finally, the classifier predicts the input image's class according to the encoder's output vector (only the first element, i.e. CLS token [4]) via a commonly used linear projection and softmax operation. For simplicity, we omitted the introduction of other details of ViT, such as the CLS token, position embedding, layer normalization, etc.

2.2. Federated continual learning

Conventional federated learning (FL) is cast as an empirical risk minimization problem of the form:

$$\min_W Loss(W) = \sum_{n=1}^{N} \frac{|D^{(n)}|}{|D|} loss^{(n)}(W) \tag{2}$$

where W is the global model and $loss^{(n)}(W)$ is the nth device's local loss function, commonly defined by:

$$\frac{1}{|D^{(n)}|} \sum_{i=1}^{|D^{(n)}|} loss_{CE}(W; x_i, y_i) \tag{3}$$

where $D^{(n)} = \{(x_i, y_i)|0 \le i \le |D^{(n)}|\}$ is the data samples on nth device. In this setting, all data samples $D = \{D^{(1)}, D^{(2)}, \dots, D^{(N)}\}$ distributed among clients belong to a single and the same classification task.

While, **federated continual learning (FCL)** assumes that multiple clients are trained on a sequence of tasks from private data streams. In this setting, overall data samples D consists of a sequence of individual tasks. That is $D = \{D_1, D_2, \dots, D_T\}$, where T is the total number of tasks. As for a specific client n, its data samples $D^{(n)}$ may continuously come from different tasks, namely, $D^{(n)} = \{D_1^{(n)}, D_2^{(n)}, \dots, D_k^{(n)}\}$, where k is the identification of task. Then, the goal of FCL is to maximize each client's test accuracy on all of the past tasks' samples under the paradigm of federated learning.

2.3. Related work

The major problem faced by federated continual learning is the catastrophic forgetting in neural networks when learning new tasks [8]. This problem is further complicated by the negative transfer in federated learning due to the Non-IID datasets in different clients [15]. 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 [22–24]. (2) *Regularization-based techniques* estimate different parameters/weights' contributions to a model and maintain part of information in important weights when learning new tasks [25–28]. (3) *Dynamic architectural techniques* design different models for different tasks and solve catastrophic forgetting by isolating part of the model parameters [37–39]. However, it is difficult to apply these techniques in federated learning, because they require massive retained samples or weights to increase

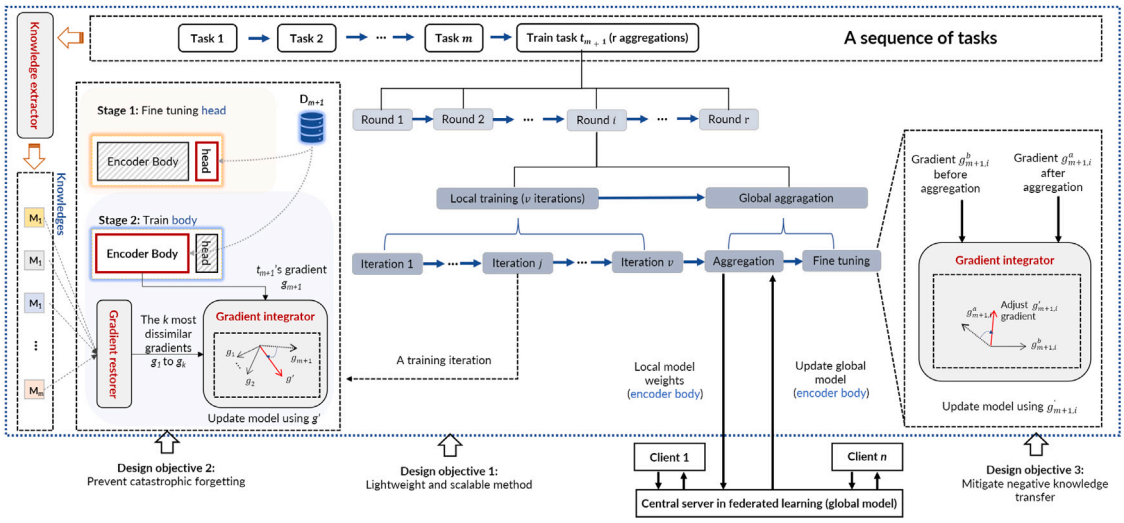


Fig. 2. FedViT process and its three design objectives.

model generalization, 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 [40,41]. This training paradigm utilizes distributed datasets to help DNN models, which are easily trapped in local optima due to limited training conditions [42] like insufficient samples, find the global optima parameters with stronger generalization. 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 [15]. *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) [13] 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 [43–45] 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 [46]. (4) Multi-task learning lets each client train a separate task [47] and further classifies clients into different groups according to their tasks [48]. 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.

Federated Continual Learning. 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 [29]. 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 [49]. The latest FCL technique, FedWEIT [7] 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.

Vision Transformer. The standard ViT [4] is the first transformer architecture in computer vision, but its performance depends on large scale training data and it has a heavy computational complexity due to the self-attention mechanism. Hence, the following directions of ViT are respectively *data efficient* ViTs [18,50–52] and *computational efficient* ViTs [53–58]. Data efficient ViTs either incorporate with convolutional operations to make ViTs training on small scale dataset [52] or distill knowledge from a pretrained CNN teacher [18]. Computational efficient ViTs adopt different space reduction methods to progressively shrink the dimension of features [54,55]. Besides the architecture design, ViT has attracted the attention of researchers in both the FL and continual learning fields. For example, ViT-FL [17] conducts a series of experiments and demonstrates that ViT is well-suited for non-iid FL setting. FESTA [59] splits the model into a head and a body, deploying the body on server to learn task-shared feature while deploying the head on each client to perform task-specific prediction tasks. In continual learning, Dytox [60] makes the body to capture task-agnostic common knowledge, while providing a unique task token for each newly added task. Similar to existing classifier ensemble strategies [61,62], Dytox concatenates all the task-specific classifiers to form a unified classifier capable of classifying multiple past tasks. LVT [31] and META [63] also explore the usage of ViT in different learning scenarios. However, these works either do not consider the catastrophic forgetting issue when learning stream of tasks on client or failed to leverage federated learning technology to transfer knowledge learned from other clients to improve the ViT's generalization capability.

3. Method

3.1. Overview

We design FedViT to continually train sequences of different learning tasks on federated clients. FedViT features on a novel concept of signature task knowledge which further enables lightweight computation and communication on resource-constrained edge devices. As shown in Fig. 2, FedViT acts in each federated client and is composed of three components: knowledge extractor, gradient restorer and gradient integrator. In FedViT, 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 ($m \geq 1$), FedViT trains ViT models using r aggregation rounds. Each round consists of two parts: local training with v training iterations and global aggregation with the central server. The v iterations contain two stages: (1) fine-tuning the ViT's *head* (classifier) weights by freezing the *body* (composed of tokenizer and transformer encoder), (2) training the *body* weights combined with m old tasks' knowledge. When aggregating global model on server and updating local model on client, only the ViT's body participates (as is shown in Fig. 1 where the *classifier* is not displayed in central server).

This training paradigm utilizes the properties of decomposability [59,60] of ViT and robustness [17] to distribution shift of its transformer encoder, making the body always learn cross-task information while the head only serves for task-specific information extraction.

In summary, FedViT is designed with three objectives.

Lightweight and scalable method. In order to effectively train tasks on federated clients with a large number of tasks and clients, FedViT is a client-side method that exploits the limited resources of edge devices by extracting critical knowledge and integrating those of signature tasks. The **knowledge extractor** stores the knowledge of each task, i.e., a subset of training samples for that task. The **gradient restorer** converts the knowledge of each task into a gradient g, which can help the ViT model quickly recover the knowledge retained from the previous task when learning new task. More specifically, We use a very small amount of locally trained samples as the knowledge learned by ViT. This is because ViT has a distinct "data hungry" characteristic, making its performance in resource-constrained edge computing environments more sensitive to the quality of training samples, especially when the amount of data for a single task is limited. Moreover, there are usually redundant samples in the original training set that may have a negative impact on recovering knowledge for a given task. For example, samples that cause a large inference loss for the ViT model after training. Given the limited storage space and computational overhead of edge devices, it is necessary to extract only a small amount (e.g. 10%) of samples for each task.

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 [22]. 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 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 [15], 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, FedViT 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.

3.2. Knowledge extractor

Knowledge extractor has two design purposes: to preserve knowledge that retains most of the information about the task and to make it easy to use this knowledge when learning new tasks. Since ViT is a highly data-dependent model, we aim to fully exploit the relationship between current task's training samples and the knowledge learned on it, making the ViT able to quickly and exactly recall the task's knowledge using only a small amount of the task's samples (e.g. 10%).

Specifically, we consider such task-related samples from two aspects: (1) The samples are well adapted (e.g. have lower loss values) to the trained ViT model on this task; (2) The class (category) distribution of these samples is consistent with that of the task's complete samples. We believe the samples extracted by this method are important and impressive for the model to recover the task's knowledge. Just like the way humans review old knowledge, using more important and impressive learning materials makes it easier to recall the memories about that knowledge. Similarly, using these samples will guide the optimization direction of ViT model to adapt to old tasks when learning new tasks.

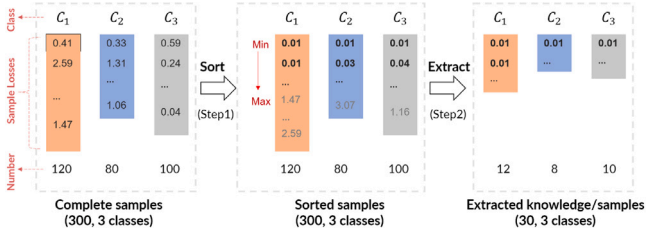


Fig. 3. Process of sample extraction: Upon completing a task, the knowledge extractor sorts samples within each class C_i in ascending order based on the loss values generated by the current model (**Step 1**), and then selects the lowest 10% (according to ρ) of loss values samples in each class to retain (**Step 2**). The ultimately extracted knowledge (data samples) maintains the same class distribution as the complete set of samples.

Formally, let $D_i = (X_i, Y_i)$ represent the complete training samples of task t_i on a client. After training the task t_i, the knowledge on task t_i can be denoted by $M_i = \{M_{i,j} | 1 \leq j \leq |c|\}$, where $|c|$ is the total number of classes in D_i; $M_{i,j}$ is the extracted samples belonging to the t_i task's jth class and it should satisfies the following conditions:

$$\begin{aligned} M_{i,j} &= (X'_{i,j}, Y'_{i,j}) \\ &= \{(x'_{i,j}, y'_{i,j}) | (x'_{i,j}, y'_{i,j}) \in D_{i,j}, loss(f(W_i, x'_{i,j}), y'_{i,j}) \leq \rho_j\} \end{aligned} \tag{4}$$

where W_i represents the model weights trained after the task t_i, $loss$ is the cross-entropy function, and ρ_j represents the loss value at a certain percentile (e.g. 10%) in the jth class.

Fig. 3 illustrates the process of knowledge extraction by a client after completing the task t_i: (1) the current trained model for task t_i is used to infer the loss values of all training samples; (2) all loss values are sorted in ascending order; (3) for each class C_j, the top 10% (according to the value of ρ) of samples based on their loss values are retained. This method constructs a sample subset (knowledge) whose class distribution is consistent with the original dataset (preserving the samples diversity of this task), while ensuring that each class contains samples that are best adapted to this task's model (ensuring the importance of the samples to this task).

3.3. Gradient restorer

The knowledge restorer is designed to utilize the extracted knowledge to maintain the model's accuracy on old tasks when learning new tasks. This component converts the saved samples (knowledge) $\{M_1, M_2..., M_m\}$ into gradients that are relevant to the m old tasks, i.e. $\{g_1, g_2, \ldots, g_m\}$. As these samples are most adapted to the old task models and maintain the class distribution of old tasks, the gradients computed on them can contain the hidden knowledge of these tasks. Therefore, combining these gradients to update new task's model can make it also adapt to the old task's samples.

Formally, we use cross-entropy as the loss function, and compute the gradients of current task's model W on the knowledge(samples) preserved from all the old task t_i:

$$g_i = \nabla loss(f(W, X'_i), Y'_i), (1 \leq i \leq m) \tag{5}$$

where ∇ is the gradient operator applied to the prediction $f(W, X'_i)$ and the ground truth label Y'_i for each old task. We then compute the gradient on the new task t_{m+1}:

$$g_{m+1} = \nabla loss(f(W, X_{m+1}), Y_{m+1}) \tag{6}$$

With the increased number of tasks (that is, m is large), FedViT only selects the k gradients that are most dissimilar with task t_{m+1}'s

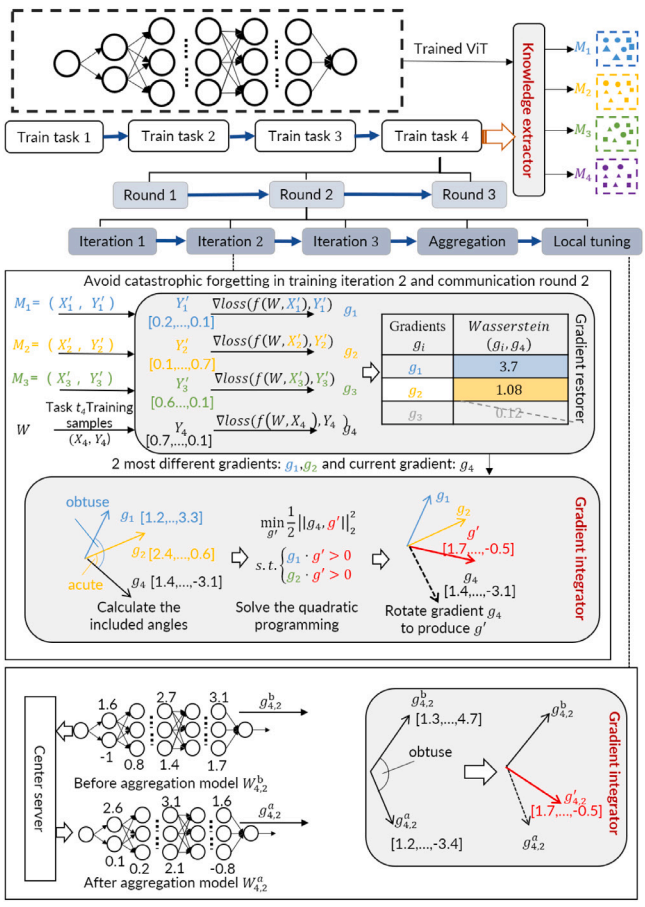


Fig. 4. Example model learning process with FedViT.

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 FedViT, 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 samples, the constraint is the memory footprint of these samples. For the number k of gradients, the constraint is each task's computational time.

3.4. 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 [22], 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 $G = \{g_1 \text{ to } g_k\}$ be the set of previous gradients, the integrator employs the quadratic programming [22] to solve this optimization problem with polynomial time complexity:

$$\min_{g'} \frac{1}{2}||g_{m+1}, g'||_2^2 \quad s.t.\ Gg' \geq 0 \tag{7}$$

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 Eq. (7), $\frac{1}{2}||g_{m+1}, g'||_2^2 = \frac{1}{2}(g')^\top g' - g_{m+1}g' + \frac{1}{2}(g_{m+1})^\top g_{m+1}$, 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 Eq. (7) as:

$$\min_{v} \frac{1}{2} v^\top GG^\top v + g^\top G^\top v \quad s.t.\ v \geq 0 \tag{8}$$

That is, the gradient integrator solves the dual optimization programming in Eq. (8) to find v and calculates the integrated gradient according to the following equation [22]:

$$g' = G^\top v + g_{m+1} \tag{9}$$

3.5. Running example

Fig. 4 shows FedViT'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 example selects one iteration and one aggregation round to illustrate how FedViT works.

At iteration 2 of round 2, FedViT prevents catastrophic forgetting based on the retained knowledge (M_1 to M_3) of the three previously learned tasks (t_1 to t_3). The gradient restorer computes gradients g_1 to g_3 using samples from every retained knowledge respectively. 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. FedViT 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 avoiding their **negative knowledge transfer** to the local model before aggregation.

4. Proof of convergence in FedViT

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

Definition of convergence. Let W be the set of the 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 $H(r)$ between these W_r and W^* is defined as:

$$H(r) = \sum_{i=1}^{r} f(W_r) - \min_{W} \sum_{i=1}^{r} f(W) \tag{10}$$

Given that r is usually a large number and the *training can converge* if $\frac{H(r)}{r}$ approaches 0. We convert Eq. (10) to:

$$\lim_{r\to\infty} \frac{H(r)}{r} = \lim_{r\to\infty} \mathbb{E}[f(W_r)] - f(W^*) = 0 \tag{11}$$

where $\mathbb{E}$ is the mathematical expectation. In convergence proof, we compute the upper bound of $\lim_{r\to\infty}\frac{H(r)}{r}$ and show that it approaches 0 under some constraints. The proof is based on the three assumptions in existing work [64,65]

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 [66] is given in Assumption 3 [15].

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

$$\mathbb{E}[f(W_r)] - f(W^*) \leq \frac{\tau}{\gamma + r - 1}\left(\frac{2B}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}||W_r - W^*||^2\right) \tag{12}$$

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 FedViT, suppose $W_r = W_r^G \cup W_r^L$ consists of global ViT weights W_r^G and local ViT 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 FedViT's convergence 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.

4.1. 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:*

$$\mathbb{E}[f(W_r^L)] - f(W^{L^*}) \leq \frac{D^2}{2\eta_r^L r} + \frac{\lambda^2\eta_j^L}{2} \tag{13}$$

Proof. . Let $g_r = \nabla f(W_r^L, \xi_r)$ be the gradient at iteration r. According to Eq. (10), the gap $H(r)$ between local weights W_r^L and W^{L^*} is:

$$\begin{aligned} H(r) &= \sum_{i=1}^{r} f(W_i^L) - \min_{W^L}\sum_{i=1}^{r} f(W_i^L) \\ &= \sum_{i=1}^{r}[f(W_i^L) - f(W^{L^*})] \end{aligned} \tag{14}$$

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

$$f(W^{L^*}) \geq f(W_r^L) + \langle g_r, W_r^L - W^{L^*}\rangle \tag{15}$$

In model updating, W_r^L satisfies:

$$\begin{aligned} & W_{r+1}^L = W_r^L - \eta_r^L g_r \\ \longrightarrow\ & W_{r+1}^L - W^{L^*} = W_r^L - W^{L^*} - \eta_r^L g_r \\ \longrightarrow\ & ||W_{r+1}^L - W^{L^*}||_2^2 = ||W_r^L - W^{L^*} - \eta_r^L g_r||_2^2 \\ \longrightarrow\ & \langle g_r, W_r^L - W^{L^*}\rangle = \frac{1}{2\eta_r^L}(||W_r^L - W^{L^*}||_2^2 - \\ & \quad ||W_{r+1}^L - W^{L^*}||_2^2) + \frac{\eta_r^L}{2}||g_r||_2^2 \end{aligned} \tag{16}$$

By combining Eqs. (14), (15), and (16) we have:

$$\begin{aligned} H(r) \leq & \sum_{j=1}^{r}\frac{1}{2\eta_j^L}(||W_j^L - W^{L^*}||_2^2 - ||W_{j+1}^L - W^{L^*}||_2^2) \\ & + \sum_{j=1}^{r}\frac{\eta_j^L}{2}||g_j||_2^2 \end{aligned} \tag{17}$$

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

$$\begin{aligned} & H(r) \leq D^2\frac{1}{2\eta_j^L} + \frac{\lambda^2}{2}\sum_{j=1}^{r}\eta_j^L \\ \longrightarrow\ & \mathbb{E}[f(W_r^L)] - f(W^{L^*}) \leq \frac{D^2}{2\eta_r^L r} + \frac{\lambda^2\eta_j^L}{2} \end{aligned} \tag{18}$$

4.2. 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:*

$$\begin{aligned} & \mathbb{E}[f(W_r^G)] - f(W^{G^*}) \leq \\ & \frac{\tau}{\gamma + r - 1}\left(\frac{2B}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}||W_r^G - W^{G^*}||^2\right) \\ & \textit{where } B = \sum_{i=1}^{N} p_i^2\sigma_i^2 + 6L\Omega + 8(r-1)^2(g')^2. \end{aligned} \tag{19}$$

Proof. FedViT employs FedAvg [15] and FedKNOW [33] as the global parameter aggregation algorithm. Hence if its gradient g' follows Assumptions 1 and 2, its training of global weights can be bounded by Eq. (19) according to Assumption 3. We now proves the boundedness of g'.

In FedViT, we select k past tasks from the stored samples $\{M_1, M_2, \ldots, M_m\}$, and use the ViT model W^G to compute the past gradients g_1 to g_k. Then we use Gradient Integrator and the current task's training samples to get g'. That is, $g' = G^\top v + g_r$ $(v \geq 0)$ according to Eq. (9). According to Assumption 1, all gradients in G are bounded and g_r is a constant, we have $||g'||^2$ in Eq. (19) is bounded:

$$\begin{aligned} ||g'||^2 &= max(||G^\top v + g_r||^2) \\ &= max(g_r^\top g_r + 2v^\top G g_r + 1) \end{aligned} \tag{20}$$

4.3. Convergence of overall model

Theorem 1. *In a client, FedViT 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 $\mathcal{O}(r^{-1})$:*

$$\lim_{r\to\infty}\mathbb{E}[f(W_r^L \cup W_r^G)] - f(W^*) = 0 \tag{21}$$

Proof. Let $W^* = W^{L^*} \cup W^{G^*}$, we convert Eq. (21) as:

$$\begin{aligned} & \lim_{r\to\infty}\mathbb{E}[f(W_r^L \cup W_r^G)] - f(W^{L^*} \cup W^{G^*}) = 0 \\ \longrightarrow\ & \lim_{r\to\infty}\mathbb{E}[fW_r^L] - f(W^{L^*}) \cup \mathbb{E}[fW_r^G] - f(W^{G^*}) = 0 \end{aligned} \tag{22}$$

According to Lemma 1, $\lim_{r\to\infty}\mathbb{E}[f(W_r^L)] - f(W^{L^*})$ is bounded (Eq. (13)) and this bound approaches to 0 if the learning rate η^L decreases at the rate of $\mathcal{O}(r^{-\frac{1}{2}})$ [67,68]. Similarity, Lemma 2 states that $\lim_{r\to\infty}\mathbb{E}[f(W_r^G)] - f(W^{G^*})$ is bounded (Eq. (19)) and this bound approaches to 0 if learning rate $\eta_r^G \leq \frac{2}{\mu(\gamma+r)}$ and decreases at the rate of $\mathcal{O}(r^{-1})$ [15]. ■

5. Evaluation

In this section, we evaluate the full implementation of FedViT on top of PyTorch [69] in exhaustive experimental scenarios against a wide set of data benchmarks and ViTs.

5.1. Experimental settings

Testbed. We choose four types of heterogeneous edge platforms imposing different architectures to showcase FedVIT'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 4 GB 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 ViTs in PyTorch 1.9.0 (Python 3.6.9).

Datasets. We select five representative federated and continual datasets to evaluate FedViT. In these datasets, a *task* refers to an image classification task for a given set of objects. Following the setting of classic continual learning methods [70], we divided the dataset (training/test sets) into tasks by classes.

- CIFAR100 [71] and FC100 [72]: They 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.
- COR50 [73]: This 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 [74]: This 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.
- TiniImageNet [75]: This dataset has 100k data points from 200 classes and 10k testing points (50 ones per class). We divided this dataset into 50 tasks (4 classes per task) to evaluate the impact of task number to FedViT's performance.

These datasets were trained on a 6-layer ViT model, which has the same architecture design as the standard ViT model [4]. Furthermore, to evaluate the FedViT's generalization capability on different model architectures, we also tested it on 8 state-of-the-art ViTs [32]. Considering evaluation budget, we use the tiny version of the 8 ViT models and modify their configurations to adapt to the benchmarks. For example, to train DeiT [18] model on CIFAR-100, we modified the model's image size parameter from 224 to 32 and the patch size parameter from 16 to 4. With these changes, DeiT can split a CIFAR-100 image into the same number of tokens (i.e. 64) as that of origin DeiT, enabling it to capture the global inter-dependencies among them.

Task and dataset assignment in federated setting. We followed the setup of FedRep [76] to implement the federated continual learning dataset distribution. Each client has the complete dataset and a distinct task sequence. To build non-IID federated learning scenarios with varying degrees of heterogeneity, we randomly assign 2 to 5 classes of data in a task to different clients. For each class, we randomly select 5% to 10% of the training samples to assign.

Compared Baselines. We implemented and compared the performance of our method FedViT with 11 state-of-the-art baselines, including 6 continual learning methods, 3 federated learning methods, and 2 federated continual learning methods.

- *Continual learning methods:* (1) gradient episodic memory (GEM) for continual learning [22,23] calculates previous gradients and uses the included angle between them and the current gradient in model training; (2) Balanced Continual Learning (BCN) [24] retains the previous training samples and uses them to maximize the data distribution among different tasks and minimize the model training errors; (3) Elastic Weight Consolidation(EWC) [26] 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; (4) Memory aware synapses(MAS) [27] improves this approach by estimating each weight's importance according to the output's sensitivity to this weight. (5) Dytox [60] is a recent continual learning method specifically designed for ViT. It utilizes the robustness to distribution drift of ViT's transformer encoder to obtain common features among tasks, and then learns an extended token for each incoming task, thereby preventing catastrophic forgetting. 6) LVT [31] is also designed for ViT. It replaces the K matrix generated by ViT with learnable weights, when learning a new task, it constrains the distance between old task's K weights and new task's K weights to prevent forgetting old knowledge.
- *Federated learning methods:* (1) FedAvg [66] 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 [13] dynamically changes the ratio of global and local models in training. (3) FedRep [76] 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.
- *Federated continual learning methods*: (1)FedWEIT [7] 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; (2)FedKNOW [33] is the latest FCL method that retains task-specific model weights for each old task. When learning a new task, it updates the model by integrating all gradients restored from old tasks' weights, thus preserving old knowledge. Although both FedKNOW and FedWEIT transfer knowledge through model weights, FedKNOW only constructs the knowledge base on client site, hence significantly reducing communication cost.

Evaluation Metrics. We consider both model accuracy and training time (hour) in evaluation. The accuracy refers to the top-1 accuracy of the model on test data points: the top predicted class (the one with the largest probability) is the same as actual class label. In the continual learning scenario, the accuracy of task t_m refers to the average accuracy of the model on all m learned tasks, namely $A_m = \frac{1}{m}\sum_{i=1}^{m} a_{m,i}$, where $a_{m,i}$ represents the accuracy of task t_m's model on task t_i's test dataset.

5.2. Comparative evaluations under different FCL scenarios

This section's evaluation compares model accuracy and training time between FedViT and 11 baseline 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.

Hyperparameters Settings. We train all ViT models using the AdamW optimizer [77], where the learning rate and weight decay are set to 0.0005 and 0.05, respectively. These values are chosen from the hyperparameter search space of DeiT [18] and provide a relatively high training accuracy. We perform grid search to optimize federated learning hyperparameters using the FedAvg algorithm [15], with the following parameter spaces: client sampling rate {0.1, 0.2, 0.4, 0.6}, communication rounds per task {5, 10, 15}, and local iterations (epochs) {2, 6, 10, 15}. To fairly compare all techniques under edge constrains, we set 10 communication rounds per task, 0.4 client sampling rate, and 6 local iterations. This configuration balances the accuracy and evaluation overhead across all baselines. For hyperparameters brought by some specific baselines, we set search space bounds to 1/2 and 2 times their original values. Particularly, memory rate is set to 10% for memory-based methods (GEM, BCN); regularization weights are respectively set to 40000 and 100 for EWC and MAS, which are regularization-based methods; knowledge storage rate ρ is set to 10% and the selected gradient number k is set to 10 for the gradient integration methods (FedKNOW, FedViT). The 10% memory rate enables fair comparison with memory-based methods, and 10 selected gradient number achieves the highest accuracy under time constraints.

Comparison Results. Figure 5 displays the comparison results of the 12 techniques and we have three key observations:

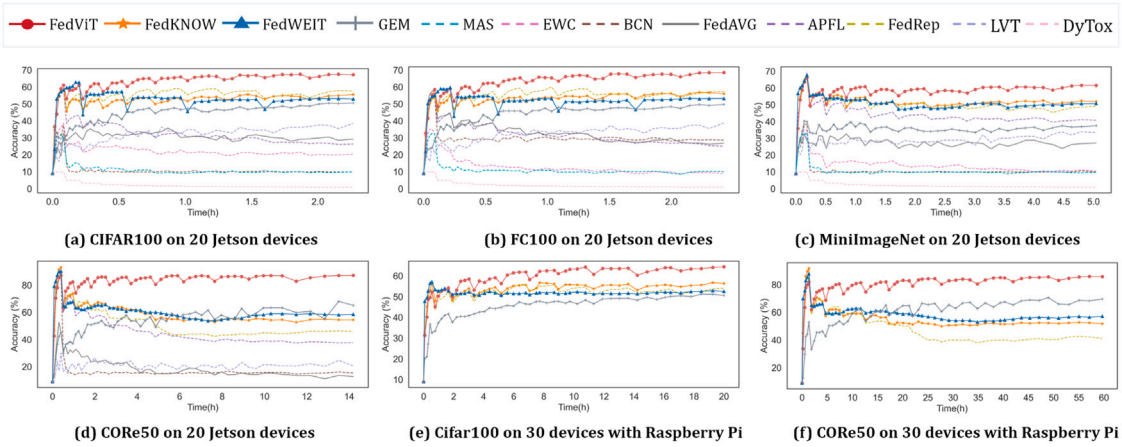


Fig. 5. Comparison of model accuracy and training time between FedViT and 11 baseline methods.

Impact of catastrophic forgetting. In the federated learning methods, both FedAvg and APFL incur more catastrophic forgetting than most of other baselines (especially in Fig. 5(d)) because they do not retain any old tasks' knowledge. On the other hand, the FL method FedRep produces higher accuracies than most of the other FL methods. This is because FedRep only aggregates the encoder, which is robust to distribution shift, of ViT to obtain cross-task knowledge for each client, thus maintaining the accuracies as the tasks evolve. However, in more challenging scenarios (e.g., CORe50), FedRep also causes obvious decrease of accuracy (Fig. 5(d)). Instead, our method FedViT not only leverages the robustness of ViT's encoder but also injects the most important old tasks' knowledge into the new task's learning via the gradient integration, thereby suffering from the least catastrophic forgetting over all the evaluated baselines. In the continual learning methods, GEM produces the highest accuracies by replaying old tasks' training samples, while the regularization based methods such as MAS and EWC show sharp declines. This suggests that ViT can benefit more from data samples in FCL, in contrast, preventing catastrophic forgetting by only constraining the model's parameter space on each client may lead to a decreased performance of global model after aggregation. In federated continual learning methods, FedKNOW and FedWEIT effectively avoid catastrophic forgetting but still produce lower accuracies than FedViT by an average of 13.5 percent. This is because they take model weights as knowledge to prevent forgetting, but ViT is difficult to achieve a desired accuracy in resource-constrained FCL, decreasing the knowledge's reliability.

Impact of negative knowledge transfer. In a federated learning environment, the non-IID datasets in different clients also considerably influence model accuracy. The six continual learning baselines well address catastrophic forgetting, but suffer from negative knowledge transfer from other clients. For example, DyTox extends task-specific tokens for each client, however, due to the client sampling in FL, newly participated clients directly initialize their tokens through the global model that injects unrelated task tokens from other clients, thus incurring the most negative knowledge transfer as shown in Fig. 5. In addition, FedWEIT and FedKNOW have higher accuracies than other baselines. However, their parameter decomposition/pruning strategies are only designed for CNN and may harm the functionalities of some particular layers (e.g., the structure of multi-head self-attention) of ViT, resulting in lower accuracies than FedViT.

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. In this section, We chose CIFAR100 and CORe50 as the datasets, because they have different numbers of training samples. In addition, the evaluation only compares the five representative algorithms: FedViT, FedKNOW, FedWEIT, GEM, and FedRep. This is because these algorithms show superior performance compared to other techniques, and represent the three categories of evaluated method in this paper. Figs. 5(e) and (f) show that training in resource-limited Raspberry Pi devices considerably delays the training time of all techniques by an average of 6.5 times, and there are not obvious accuracies among all the methods thanks to the ViT's robustness property. Overall, The results show FedViT always achieves the highest accuracies because it is lightweight and integrates task knowledge locally.

Table 1
A summary of average percentage accuracy improvement. Note that only CORe50 dataset consists of 11 learning tasks, so there are blank entries in the last row.

	Cifar100	FC100	MiniImageNet	CORe50
Task 1	64.6%	35.95%	22.19%	14.43%
Task 2	74.22%	47.3%	60.95%	40.35%
Task 3	88.3%	75.37%	76.53%	66.74%
Task 4	87.5%	77.51%	80.1%	76.54%
Task 5	95.0%	92.66%	93.93%	87.33%
Task 6	103.03%	98.37%	97.92%	92.02%
Task 7	107.69%	99.47%	101.17%	111.57%
Task 8	106.81%	103.38%	101.18%	115.13%
Task 9	113.74%	114.13%	109.92%	127.49%
Task 10	110.66%	110.33%	106.97%	125.14%
Task 11				119.67%

Table 1 summarizes the percentages of increase in the average accuracy, when comparing the accuracy of FedViT against the average accuracy of all 11 baselines techniques across 4 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 14.43% to 127.49%.

5.3. 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.

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

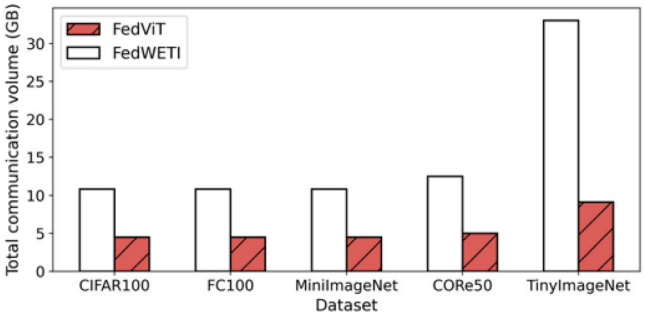


Fig. 6. Comparison of communication volume under different workloads.

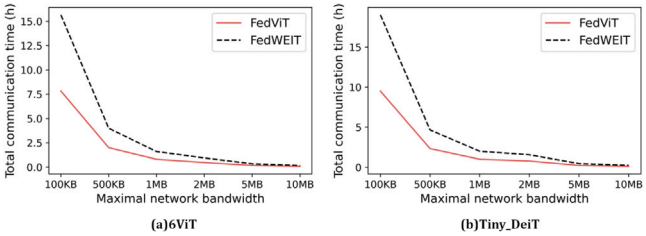


Fig. 7. Comparison of communication time under different bandwidths.

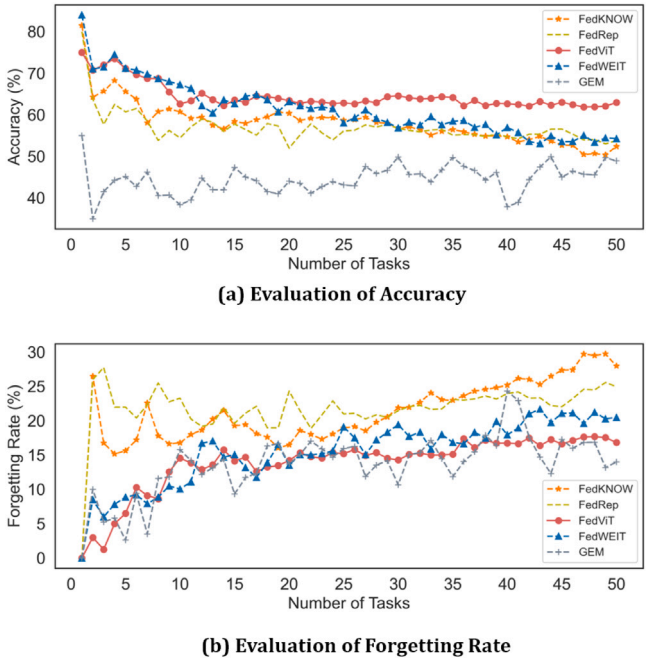


Fig. 8. Evaluations of 50 tasks.

about 10% to 30% of model training time). As shown in Fig. 6, FedViT 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 because 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 61.55%.

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 to test 6 different network bandwidths, ranging from 100 KB per second to 10 MB per second, in each client. Fig. 7 shows the communication time of two ViT 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 9.5 h when the network bandwidth is 100 KB per second).

5.4. Discussion of task and client numbers

In this section, we extend our experiments to further discuss two key factors that affect the performance of FCL: the number of clients and the number of tasks. In evaluations, we report *average accuracy* and *average forgetting rate* as metrics. Suppose m tasks are learned, the definition of *average accuracy* is identical to Section 5.1, and the *forgetting rate* is defined by $F_m = \frac{1}{m-1}\sum_{i=1}^{m-1} max_{k\in\{1,\ldots,m-1\}}(a_{k,i} - a_{m,i})$ [23,31,78], where $a_{m,i}$ is the test accuracy on task t_i when the model learned the task t_m.

Number of Tasks. In this section, we use TinyImageNet dataset that consists of 200 classes data samples to construct a multi-task dataset. In particular, we split the dataset into 50 tasks, where each task contains 4 classes of data samples. To ensure the data heterogeneity for each task, we partition the dataset into 10 clients and each client contains at most 2 classes of data samples. In addition, we still use 6-layer ViT to evaluate. Fig. 8 shows the fluctuations of accuracies and forgetting rates when the number of tasks increases from 1 to 50. We can see that FedViT provides the highest accuracy not only at the final task but also the most training stages, meanwhile, the forgetting rate of FedViT is just higher than the lowest (GEM) within 3 percent, verifying its effectiveness of retaining old tasks' knowledge among the four methods. Although GEM produces the lowest forgetting rete, it still results in the lowest accuracies at all the training stages due to the lack of a mechanism to mitigate negative knowledge transfer caused by model aggregation, especially when facing massive tasks.

Number of Clients. Two cluster scales are considered in this evaluations: 50 and 100 clients. This is because when the number of clients is 100, each client can only be assigned a very small amount (50 training samples per task) of CIFAR100 data samples, making it a highly heterogeneous federated learning scenario.

Table 2 lists the accuracies and forgetting rates of the five methods with 50 and 100 clients. We can see that most of the methods result in accuracies decrease as the client number increase. However, our approach FedViT still produces the highest final accuracies in both 50 and 100 scenarios, meanwhile, the forgetting rates are close to that of GEM which produces the lowest. This is because FedViT only stores the most important data samples of learned tasks and their contribution becomes larger when training samples are insufficient. Meanwhile, FedViT optimizes the gradient integration process to handle the influence of negative transfer caused by client number increase. In contrast, FedRep forgets the most for it does not consider exploiting the information of previous tasks. However, its training strategy facilitate the learning of new tasks even under the severe negative knowledge transfer scenario, thus achieving higher accuracies than GEM. On the other hand, FedKNOW and FedWEIT also achieve higher accuracies than GEM thanks to their mechanisms to address negative knowledge transfer. However, they still produce lower accuracies and

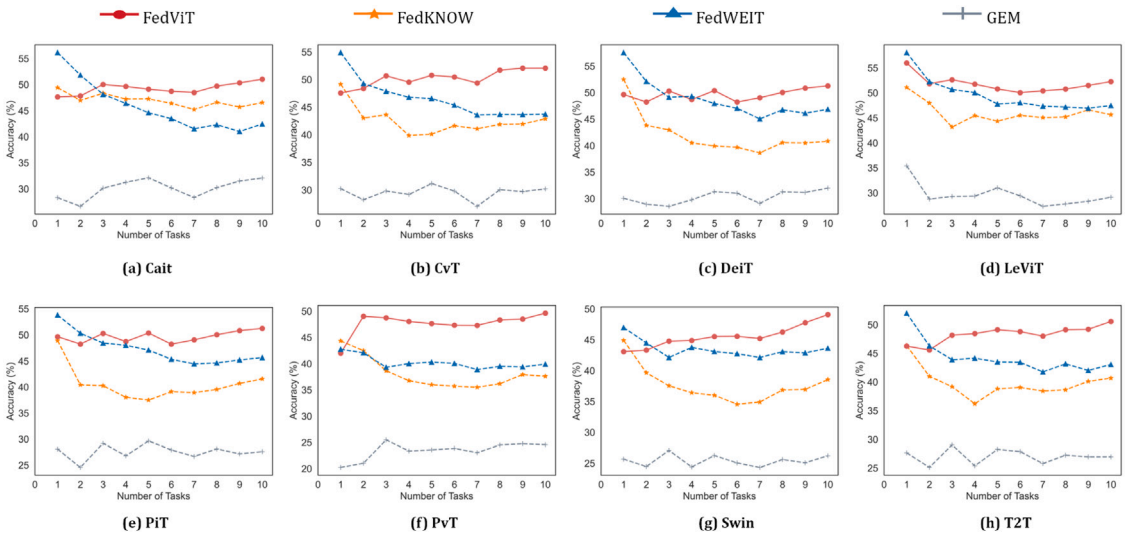


Fig. 9. Applicability of FedViT to five categories of ViTs.

Table 2
Comparisons of accuracy and forgetting rate under different numbers of clients.

Method	50 clients		100 clients	
	Acc.	Forgetting rate	Acc.	Forgetting rate
FedRep	50.33	18.50	50.40	15.69
GEM	46.30	**1.23**	45.54	**0.24**
FedWEIT	53.02	6.51	50.20	5.22
FedKNOW	59.26	11.30	53.18	7.30
FedViT(ours)	**62.08**	1.70	**54.25**	4.23

higher forgetting rats than that of FedViT, indicating that storing small amount of samples rather than model weights may still be the best choice when training on limited number of samples.

In summary, FedViT leverages a small amount of high-quality samples as old task knowledge, enabling it to accurately recover the old task knowledge even in the scenarios with extremely limited training samples. Additionally, the knowledge restorer based on gradient integration effectively addresses the severe negative knowledge transfer problem caused by the increasing number of clients or tasks, making it a potential candidate for practical federated learning applications.

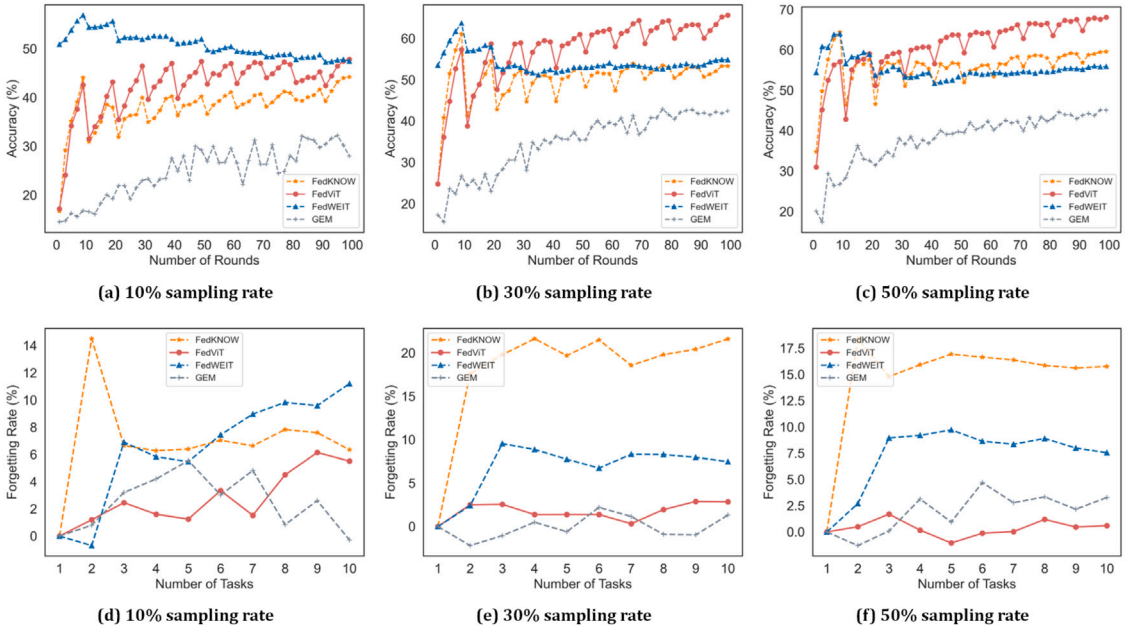
5.5. Applicability in different ViTs and settings

Applicability of FedViT to ViTs. FedViT represents the first framework that supports knowledge-level federated continual learning of ViTs for edge-based environment. FedViT can be generalized to support most of state-of-the-art ViT models. To support this claim, we implemented and tested 8 ViTs belonging to 5 categories: (1) pure transformer architecture design (DeiT [18] or standard ViT [4]); (2) deep-layer ViT (Cait [79]); (3) introducing convolutional operation (CvT [52] and LeViT [58]); (4) reducing feature scale (PvT [54] and PiT [17]); (5) fusing local and global features (Swin [56] and T2T [57]). In the evaluation, we set each task to 5 rounds, with 4 epochs per round, and keep the other settings the same as Section 5.1. The reduction of training budget is because we only focus on the accuracy gaps among these FCL techniques.

Figure 9 shows a comparison of training different ViT models on the 10-task MiniImageNet dataset using FedViT, FedKNOW, FedWEIT and GEM. It can be observed that FedViT displays obvious superiority over all the other three baselines by producing much higher accuracies. This is because all the ViT variants need rich training data samples to better understand the global information of the dataset, thus making the sample-based method FedViT architecture-agnostic in training ViT in FCL. In the other methods, GEM still achieves the lowest accuracy (under 35%) among all ViTs due to the negative knowledge transfer caused by non-IID. FedKNOW and FedWEIT, on the other hand, suffer from obvious knowledge forgetting problems for different ViTs. This is because they apply the simplest model decomposing methods (e.g., the L1 pruning method in FedKNOW) designed for CNNs to ViT models, while the importance of weights in different ViTs varies, thus decreasing the accuracy of ViT.

Comparisons under different Client Sampling Rates. In this section, we investigate the impact of client sampling rate on FedViT, FedKNOW, and GEM. Figure 10 shows average accuracies and forgetting rates under different client sampling rates. Our experimental results reveal that, despite performing well when sampling rate is 10% (Fig. 10(a)), FedWEIT's overall accuracy shows a decreasing trend as the number of tasks increases, making it unsuitable for edge FCL scenarios with large number of tasks and participated clients. Moreover, as the sampling rate increases, the accuracy of all the four methods improves (Fig. 10(b) and (c)) while forgetting rate decreases Fig. 10(e) and (f). This is because the global ViT model sees more data samples from different clients, thus improving the model's generalization. Besides, as the sampling rate increases, FedViT always achieves the highest final accuracies than others and results in considerably lower forgetting rates in all the scenarios.

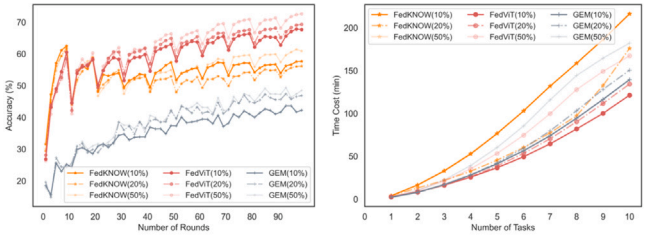
Comparisons under different Knowledge Storage Rates. In this section, we investigate the impact of knowledge storage rate on FedViT, FedKNOW, and GEM in terms of average accuracy and task-specific time cost. Fig. 11 shows the comparison results, and we make the following observations: First, FedViT achieves the best results in both accuracy and time overhead in all the different settings. Specifically, FedViT's performance is less sensitive to the variation of knowledge storage rates than the other two methods. Second, as the knowledge storage rate increases, all the methods improve their accuracies and FedViT shows the highest accuracies over others. Moreover, FedViT outperforms the other two methods in terms of the extent of improvement, with a larger accuracy improvement rate (7%) than GEM (improvement rate 6.5%). Second, FedKNOW exhibits a more complex pattern in terms of its accuracy variation with knowledge storage rates. The minimum accuracy occurs at a 20% storage ratio, while the highest accuracy occurs at 50%. This indicates that the pruning strategy in FedKNOW for ViT models might disrupt the model structure. For



(a) 10% sampling rate (b) 30% sampling rate (c) 50% sampling rate

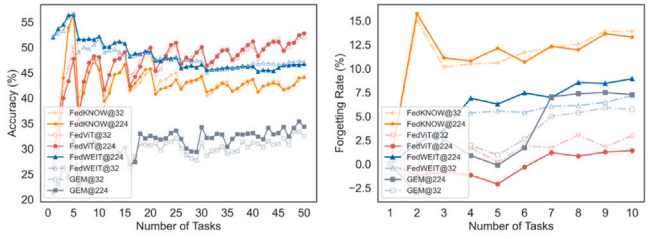
(d) 10% sampling rate (e) 30% sampling rate (f) 50% sampling rate

Fig. 10. Performance under different client sampling rates.



(a)Comparison of accuracy (b)Comparison of training time

Fig. 11. Performance under different knowledge rates.



(a)Comparison of accuracy (b)Comparison of forgetting rate

Fig. 12. Performance under different input image sizes.

example, the pruning method may remove the important weights of a attention head that is critical to a specific task. Hence, FedKNOW is more sensitive to the choice of knowledge storage ratios when using ViT models. Last, the training time cost increases as the knowledge storage rate increases and FedViT still shows the lowest training time cost for it freezes a portion of the model parameters during training.

Impact of Image Resolution. Considering the budget constraints of our experimental evaluations, we uniformly resized the input images to 32×32 pixels in previous evaluations to obtain results more quickly. However, reducing the size of the images may result in loss of other information contained in original images, thereby affecting the accuracy of model's training. We claim that FedViT's advantage will not be affected by the resize operation. To illustrate this, we evaluate the impact of input image size to the model's accuracy and forgetting rate using both 32×32 resolution and 224×224 resolution input images. Fig. 12 shows the evaluation results of Tiny DeiT model trained on MiniImageNet dataset with the both two resolutions. We can see that FedViT still shows obvious superiority by causing the highest accuracies and lowest forgetting rates under both of the resolution of 32 and 224. Moreover, in all tests, the accuracies and forgetting rates during training have no obvious gaps between different resolutions. This is because although with different input sizes, ViT projects each token into a embedding with the same dimension (e.g., 192 dimension), and extracts global features on this embedding rather than the actual values of input image, hence when the number of tokens are close, the classification performance will not significant decrease by the change of input size.

Table 3
Performance of FedViT under different rates of knowledge storage.

Items	$k = 10$				
	$\rho = 0.1$	$\rho = 0.2$	$\rho = 0.4$	$\rho = 0.6$	$\rho = 0.8$
Acc.	68.44	70.54	73.25	73.95	**75.26**
Forgettting Rate	2.22	1.08	0.39	0.19	**−0.57**

Table 4
Performance of FedViT under different numbers of selected gradients.

Items	$\rho = 0.4$				
	$k = 1$	$k = 2$	$k = 4$	$k = 6$	$k = 8$
Acc.	68.51	71.68	72.78	73.04	**73.18**
Forgetting Rate	2.65	0.76	0.44	0.21	**−0.12**

5.6. Effect of hyperparameters

In this section, we explore the impact of FedViT's hyperparameters on its performance. Except for the knowledge retention rate ρ and the selected gradient number k, most evaluation settings follow Section 5.2.

Effect of Knowledge Storage Rate ρ. To evaluate how the knowledge storage rate ρ affects FedViT's performance, we fixed the number of selected gradients k to 10 by default. As shown in Table 3, when ρ increases from 0.1 to 0.8, the accuracy improvement diminishes while the forgetting rate approaches 0. This suggests that only extracting and retaining a small subset of key samples as knowledge for each task is effective. These samples contain rich information about the task and

can help FedViT achieve good performance even with limited storage overhead. Storing excessive samples provides diminishing benefits but higher overhead.

Effect of Number of Selected Gradients k. To evaluate how the number of selected gradients k affects FedViT's performance, we fixed the knowledge storage rate ρ to 0.4. We choose $\rho = 0.4$ because it achieves a good balance between accuracy and storage overhead as shown in the analysis above. As shown in Table 4, the accuracy and forgetting rate improve quickly when k increases from 1 to 4, and then stabilize with larger k. This suggests integrating gradients from a small number of key previous tasks is sufficient to prevent catastrophic forgetting. Increasing k provides diminishing benefits but higher overhead.

6. Conclusion

This paper presents the design, implementation and evaluation of FedViT, a framework designed for transformer-based computer vision models on distributed edge devices. FedViT innovates in three aspects to tackle key challenges in federated continual learning at edge. First, lightweight sample-based knowledge extraction reduces device overhead while retaining crucial past task information, enhancing the continual learning accuracy under constraints. Second, gradient integration eliminates negative transfer and facilitates assimilation of old and new knowledge, improving model generalization. Finally, localized integration of signature task knowledge achieves scalability as tasks and clients grow without extra communication costs. Extensive experiments demonstrate FedViT markedly reduces communication overhead while ensuring accuracy, and exhibits strong scalability.

While this paper focuses on applying FedViT to vision transformers in computer vision, we believe its core concepts could generalize to other areas. First, for NLP applications, the core ideas like compact sample-based knowledge extraction and gradient integration can be adapted by extracting representative text segments as knowledge and performing optimization on text embeddings to mitigate negative transfer. Second, for Graph Neural Network (GNN) applications [80], we consider the unique connectivity of graphs. Due to the correlation between different nodes in the graph structure, FedViT preserves some subgraphs while preserving historical nodes to ensure the restoration of most of the features of historical tasks. We hope FedViT provides a framework to inspire federated continual learning research for transformers across areas.

CRediT authorship contribution statement

Xiaojiang Zuo: Conceptualization, Methodology, Software, Writing – original draft. **Yaxin Luopan:** Data curation, Formal analysis. **Rui Han:** Funding acquisition, Writing – review & editing. **Qinglong Zhang:** Investigation, Visualization. **Chi Harold Liu:** Supervision, Project administration. **Guoren Wang:** Writing – review & editing. **Lydia Y. Chen:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We will open our code and dataset soon through the github.

Acknowledgments

This paper was supported by the National Key R&D Program of China (No. 2021YFB3301503), and the National Natural Science Foundation of China (Grant No. 62272046, 62132019, 61872337).

References

[1] A. Ghasempour, Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges, Inventions 4 (1) (2019) 22.
[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge intelligence: Paving the last mile of artificial intelligence with edge computing, Proc. IEEE 107 (8) (2019) 1738–1762.
[3] J. Chen, X. Ran, Deep learning with edge computing: A review, Proc. IEEE 107 (8) (2019) 1655–1674.
[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16 × 16 words: Transformers for image recognition at scale, 2020, arXiv preprint arXiv:2010.11929.
[5] G.I. Parisi, R. Kemker, J.L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Netw. 113 (2019) 54–71.
[6] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 1–19.
[7] J. Yoon, W. Jeong, G. Lee, E. Yang, S.J. Hwang, Federated continual learning with weighted inter-client transfer, in: ICML'2021, PMLR, 2021, pp. 12073–12086.
[8] A. Robins, Catastrophic forgetting, rehearsal and pseudorehearsal, Connect. Sci. 7 (2) (1995) 123–146.
[9] S.T. Grossberg, Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control, Vol. 70, Springer Science & Business Media, 2012.
[10] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, T. Tuytelaars, A continual learning survey: Defying forgetting in classification tasks, IEEE Trans. Pattern Anal. Mach. Intell. 44 (7) (2021) 3366–3385.
[11] H. Zhu, J. Xu, S. Liu, Y. Jin, Federated learning on non-IID data: A survey, Neurocomputing 465 (2021) 371–390.
[12] S.P. Karimireddy, S. Kale, M. Mohri, S.J. Reddi, S.U. Stich, A.T. Suresh, SCAFFOLD: Stochastic controlled averaging for on-device federated learning, 2019.
[13] Y. Deng, M.M. Kamani, M. Mahdavi, Adaptive personalized federated learning, 2020, arXiv preprint arXiv:2003.13461.
[14] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, 2018, arXiv preprint arXiv:1806.00582.
[15] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of FedAvg on non-IID data, in: ICML'19, 2019.
[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[17] L. Qu, Y. Zhou, P.P. Liang, Y. Xia, F. Wang, E. Adeli, L. Fei-Fei, D. Rubin, Rethinking architecture design for tackling data heterogeneity in federated learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10061–10071.
[18] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, in: International Conference on Machine Learning, PMLR, 2021, pp. 10347–10357.
[19] M. Satyanarayanan, The emergence of edge computing, Computer 50 (1) (2017) 30–39.
[20] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J 3 (5) (2016) 637–646.
[21] O.A. Alzubi, J.A. Alzubi, M. Alazab, A. Alrabea, A. Awajan, I. Qiqieh, Optimized machine learning-based intrusion detection system for fog and edge computing environment, Electronics 11 (19) (2022) 3007.
[22] D. Lopez-Paz, M. Ranzato, Gradient episodic memory for continual learning, Adv. Neural Inf. Process. Syst. 30 (2017).
[23] A. Chaudhry, M. Ranzato, M. Rohrbach, M. Elhoseiny, Efficient lifelong learning with a-gem, 2018, arXiv preprint arXiv:1812.00420.
[24] K. Raghavan, P. Balaprakash, Formalizing the generalization-forgetting trade-off in continual learning, Adv. Neural Inf. Process. Syst. 34 (2021).
[25] H. Cha, J. Lee, J. Shin, Co2l: Contrastive continual learning, in: ICCV'2021, 2021, pp. 9516–9525.
[26] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proc. Natl. Acad. Sci. 114 (13) (2017) 3521–3526.
[27] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, T. Tuytelaars, Memory aware synapses: Learning what (not) to forget, in: ECCV'2018, 2018, pp. 139–154.
[28] S. Jung, H. Ahn, S. Cha, T. Moon, Continual learning with node-importance based adaptive group sparse regularization, Adv. Neural Inf. Process. Syst. 33 (2020) 3647–3658.
[29] X. Yao, L. Sun, Continual local training for better initialization of federated models, in: ICIP'2020, IEEE, 2020, pp. 1736–1740.
[30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR'2016, 2016, pp. 770–778.
[31] Z. Wang, L. Liu, Y. Duan, Y. Kong, D. Tao, Continual learning with lifelong vision transformer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 171–181.

[32] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, et al., A survey on vision transformer, IEEE Trans. Pattern Anal. Mach. Intell. 45 (1) (2022) 87–110.
[33] Y. Luopan, R. Han, Q. Zhang, C.H. Liu, G. Wang, FedKNOW: Federated continual learning with signature task knowledge integration at edge, 2022, arXiv preprint arXiv:2212.01738.
[34] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 38–45.
[35] H. Zhao, J. Jia, V. Koltun, Exploring self-attention for image recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10076–10085.
[36] Y. Xu, Q. Zhang, J. Zhang, D. Tao, Vitae: Vision transformer advanced by exploring intrinsic inductive bias, Adv. Neural Inf. Process. Syst. 34 (2021) 28522–28535.
[37] A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive neural networks, 2016, arXiv preprint arXiv:1606.04671.
[38] J. Yoon, E. Yang, J. Lee, S.J. Hwang, Lifelong learning with dynamically expandable networks, 2017, arXiv preprint arXiv:1708.01547.
[39] Z. Li, D. Hoiem, Learning without forgetting, IEEE Trans. Pattern Anal. Mach. Intell. 40 (12) (2017) 2935–2947.
[40] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proc. Mach. Learn. Syst. 2 (2020) 429–450.
[41] A. Li, L. Zhang, J. Wang, J. Tan, F. Han, Y. Qin, N.M. Freris, X.-Y. Li, Efficient federated-learning model debugging, in: ICDE'2021, IEEE, 2021, pp. 372–383.
[42] A.A. Movassagh, J.A. Alzubi, M. Gheisari, M. Rahimi, S. Mohan, A.A. Abbasi, N. Nabipour, Artificial neural networks training algorithm integrating invasive weed optimization with differential evolutionary model, J. Ambient Intell. Humaniz. Comput. (2021) 1–9.
[43] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: ICML'2017, PMLR, 2017, pp. 1126–1135.
[44] F. Chen, M. Luo, Z. Dong, Z. Li, X. He, Federated meta-learning with fast convergence and efficient communication, 2018, arXiv preprint arXiv:1802.07876.
[45] M. Khodak, M.-F.F. Balcan, A.S. Talwalkar, Adaptive gradient-based meta-learning methods, Adv. Neural Inf. Process. Syst. 32 (2019).
[46] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, D. Ramage, Federated evaluation of on-device personalization, 2019, arXiv preprint arXiv:1910.10252.
[47] V. Smith, C.-K. Chiang, M. Sanjabi, A.S. Talwalkar, Federated multi-task learning, Adv. Neural Inf. Process. Syst. 30 (2017).
[48] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Found. Trends Mach. Learn. 14 (1–2) (2021) 1–210.
[49] H.B. Mcmahan, E. Moore, D. Ramage, S. Hampson, B.A.Y. Arcas, Communication-efficient learning of deep networks from decentralized data, in: AISTATS'17, 2017, pp. 1273–1282.
[50] S. d'Ascoli, H. Touvron, M.L. Leavitt, A.S. Morcos, G. Biroli, L. Sagun, Convit: Improving vision transformers with soft convolutional inductive biases, in: International Conference on Machine Learning, PMLR, 2021, pp. 2286–2296.
[51] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, H. Shi, Escaping the big data paradigm with compact transformers, 2021, arXiv preprint arXiv:2104.05704.
[52] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, L. Zhang, Cvt: Introducing convolutions to vision transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 22–31.
[53] S. Mehta, M. Rastegari, Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer, 2021, arXiv preprint arXiv:2110.02178.
[54] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, L. Shao, Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 568–578.
[55] B. Heo, S. Yun, D. Han, S. Chun, J. Choe, S.J. Oh, Rethinking spatial dimensions of vision transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 11936–11945.
[56] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10012–10022.
[57] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F.E. Tay, J. Feng, S. Yan, Tokens-to-token vit: Training vision transformers from scratch on imagenet, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 558–567.
[58] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, M. Douze, Levit: a vision transformer in convnet's clothing for faster inference, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 12259–12269.
[59] S. Park, G. Kim, J. Kim, B. Kim, J.C. Ye, Federated split task-agnostic vision transformer for COVID-19 cxr diagnosis, Adv. Neural Inf. Process. Syst. 34 (2021) 24617–24630.
[60] A. Douillard, A. Ramé, G. Couairon, M. Cord, Dytox: Transformers for continual learning with dynamic token expansion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 9285–9295.
[61] O.A. Alzubi, J.A. Alzubi, M. Alweshah, I. Qiqieh, S. Al-Shami, M. Ramachandran, An optimal pruning algorithm of classifier ensembles: dynamic programming approach, Neural Comput. Appl. 32 (2020) 16091–16107.
[62] J.A. Alzubi, Diversity-based boosting algorithm, Int. J. Adv. Comput. Sci. Appl. 7 (5) (2016).
[63] M. Xue, H. Zhang, J. Song, M. Song, Meta-attention for vit-backed continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 150–159.
[64] Y. Zhang, M.J. Wainwright, J.C. Duchi, Communication-efficient algorithms for statistical optimization, Adv. Neural Inf. Process. Syst. 25 (2012).
[65] S.U. Stich, Local SGD converges fast and communicates little, 2018, arXiv preprint arXiv:1805.09767.
[66] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: AIR'2017, PMLR, 2017, pp. 1273–1282.
[67] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436–444.
[68] M. Zinkevich, Online convex programming and generalized infinitesimal gradient ascent, in: ICML'03, 2003, pp. 928–936.
[69] Pytorch, 2022, https://pytorch.org/.
[70] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert, icarl: Incremental classifier and representation learning, in: ICCV'17, 2017, pp. 2001–2010.
[71] A. Krizhevsky, G. Hinton, et al., Learning Multiple Layers of Features from Tiny Images, Citeseer, 2009.
[72] B. Oreshkin, P. Rodríguez López, A. Lacoste, Tadam: Task dependent adaptive metric for improved few-shot learning, Adv. Neural Inf. Process. Syst. 31 (2018).
[73] V. Lomonaco, D. Maltoni, Core50: a new dataset and benchmark for continuous object recognition, in: CoRL'2017, PMLR, 2017, pp. 17–26.
[74] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, Adv. Neural Inf. Process. Syst. 29 (2016).
[75] Y. Le, X. Yang, Tiny imagenet visual recognition challenge, CS 231N 7 (7) (2015) 3.
[76] L. Collins, H. Hassani, A. Mokhtari, S. Shakkottai, Exploiting shared representations for personalized federated learning, in: ICML'2021, PMLR, 2021, pp. 2089–2099.
[77] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, 2017, arXiv preprint arXiv:1711.05101.
[78] A. Chaudhry, P.K. Dokania, T. Ajanthan, P.H. Torr, Riemannian walk for incremental learning: Understanding forgetting and intransigence, in: ECCV'2018, 2018, pp. 532–547.
[79] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, H. Jégou, Going deeper with image transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 32–42.
[80] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.

Xiaojiang Zuo is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Beijing Institution of Technology, Beijing, China. His research interests include federated learning and edge computing.

Yaxin Luopan is a Master student at the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include edge intelligence and federated learning.

Rui Han is an Associate Professor at the School of Computer Science and Technology, Beijing Institute of Technology, China. Before joining BIT, He received M.Sc. with honor in 2010 from Tsinghua University, China, and obtained his Ph.D. degree in 2014 from Imperial College London, UK. His research interests are system optimization for cloud data center workloads (in particular highly parallel services and deep learning applications). He has over 40 publications in these areas, including papers at MobiCOM, TPDS, TC, TKDE, INFOCOM, and ICDCS.

Qinglong Zhang is a Master student at the School of Computer Science and Technology, Beijing Institute of Technology. His work focuses on edge intelligence and deep learning applications.

Chi Harold Liu (SM'15) received the B.Eng. degree from Tsinghua University, Beijing, China, and the Ph.D. degree from the Imperial College London, London, U.K. He is currently a Full Professor and the Vice Dean with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing. Before that, he worked for IBM Research — China and Deutsche Telekom Laboratories, Berlin, Germany, and IBM T. J. Watson Research Center, USA. He is now an Associate Editor for IEEE Trans. Network Science and Engineering. His current research interests include the big data analytics, mobile computing, and deep learning. Dr. Liu is a fellow of IET, and a fellow of Royal Society of the Arts.

Guoren Wang received the B.Sc., M.Sc., and Ph.D. degrees from the Department of Computer Science, Northeastern University, Shenyang, China, in 1988, 1991, and 1996, respectively. He is now a Full Professor and Director of Institute of Data Science and Knowledge Engineering at the Department of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. He was an Assistant President for Northeastern University, China. His research interests include XML data management, query processing and optimization, bioinformatics, high dimensional indexing, parallel database systems, and cloud data management. He has published more than 150 research papers in top conferences and journals like IEEE TKDE, ICDE, SIGMOD, VLDB, etc.

Lydia Y. Chen is an Associate Professor in the Department of Computer Science at the Technology University Delft. Prior to joining TU Delft, she was a research staff member at the IBM Zurich Research Lab from 2007 to 2018. She received Ph.D. from the Pennsylvania State University and B.A from National Taiwan University Her research interests center around dependability management, resource allocation and privacy enhancement for large scale data processing systems and services. She has published more than 80 papers in journals, e.g., IEEE Transactions on Distributed Systems, IEEE Transactions on Service Computing, and conference proceedings, e.g., INFOCOM, Sigmetrics, DSN, and Eurosys. She was a co-recipient of the best paper awards at CCgrid'15 and eEnergy'15. She is a senior IEEE member.