Mixture-of-Experts as Continual Knowledge Adapter for Mobile Vision Understanding

Bicheng Guo, Conghao Zhou, *Member, IEEE*, Shibo He, *Senior Member, IEEE*, Jiming Chen, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—Continual machine learning in the context of limited computational resources and data availability is critical in the connected digital world. Current intelligent applications predominantly rely on deep learning models requiring labor/computation-intensive training. These models often struggle to adapt effectively to new data while preserving performance on previously learned knowledge. In this paper, we introduce a lightweight method for continual knowledge adaptation that can address these challenges. To prevent disruption of the existing services, we propose a Mixture-of-Experts (MoE) adapter that integrates seamlessly with the existing vision model to encode new data. The weights of the original model are kept fixed during the adaptation process, ensuring the preservation of previously learned knowledge. The MoE technique enables scaling up the parameters of the adapter while maintaining a relatively low computation, making it fit for constrained devices in mobile computation scenarios. Furthermore, to enhance learning efficiency and accelerate convergence with new data, we implement a knowledge fusion mechanism that facilitates interaction between the existing knowledge and the information extracted from new data. The timing of employing the fusion module is further investigated. We find that it is conducive in scenarios where the task's performance requirements are enhanced. The MoE adapter and knowledge fusion module are integrated at each stage with minimal trainable parameters, efficiently optimizing resource usage. Extensive experiments and ablation studies validate the effectiveness of the proposed method. Specifically, the proposed method prevents an accuracy drop of 43.02% on the previous data compared to the continual train method, while achieving an accuracy of 44.81% on the new data, which is even 0.34% higher than fully training a new model.**

***Index Terms*—Mixture-of-experts, continual knowledge adaption, mobile edge computing.**

Received 3 November 2024; revised 18 March 2025; accepted 22 April 2025. Date of publication 5 May 2025; date of current version 3 September 2025. This work was supported in part by the NSFC under Grant 62088101 Autonomous Intelligent Unmanned Systems and in part by the Key Research and Development Program of Zhejiang Program under Grant 2024C01065. An earlier version of this paper was presented at the IEEE Global Communications Conference (GLOBECOM) 2024. Recommended for acceptance by W. Wang. [DOI: 10.1109/GLOBECOM52923.2024.10901658]. *(Corresponding author: Shibo He.)*

Bicheng Guo, Shibo He, and Jiming Chen are with the College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: guobc@zju.edu.cn; s18he@zju.edu.cn; cjm@zju.edu.cn).

Conghao Zhou was with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. He is now with the School of Telecommunications Engineering, Xidian University, Xi'an 710126, China (e-mail: c89zhou@uwaterloo.ca).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TMC.2025.3567179

I. Introduction

VISION technologies play a pivotal role in advancing human understanding of the physical world, enabling the creation of intelligent digital ecosystems that transcend spatial and temporal constraints [1], [2], [3], [4]. These advancements are primarily driven by breakthroughs in deep learning vision models, e.g., ResNet [5] or Vision Transformer (ViT) [6], which have demonstrated unprecedented capabilities in interpreting visual information. The evolution of such models, in turn, hinges on two critical enablers: the availability of massive-scale datasets and the exponential growth in computational power [6]. Practical applications of these technologies permeate daily life, from autonomous vehicles synthesizing multi-camera feeds for collision avoidance [7], [8], to deep learning-powered medical imaging systems diagnosing cardiovascular diseases [9], and augmented reality applications seamlessly blending digital objects with real-world environments through smartphone cameras [10].

Training and deploying deep learning vision models require high-performance computing platforms, with powerful processors and large on-chip memory. However, most real-world devices are mobile edge devices, such as smartphones, tablets, and Internet of Things (IoT) devices [11], which are characterized by their constrained computing resources [12], [13]. In addition, the data-scarce issue further challenges the training of the deep learning vision models. Specifically, the training data for one specific application is usually hard to collect due to privacy issues or labor-intensive labeling processes. Thus, the pervasive deployment of vision algorithms for data-scarce tasks is largely refrained.

Given limited computing power and data availability, a key challenge in developing deep learning models is maintaining high performance amid a constant stream of new data [14]. Usually, the models need to be constantly updated to the time-varying and non-stationary data distributions in real-world applications [15]. For example, as shown in Fig. 1(a), the food delivery company initially developed an application to identify house numbers, speeding up deliveries. Later, there was a need to expand the app's capabilities to recognize food items during pickup at restaurants. Subject to the resource constraints of the devices, the promising solution is to adopt one single model to serve both new and old data simultaneously. Also, the model adaption process on new data should not interrupt the operation of the current service. In this way, we can maintain a low training budget for mobile devices.

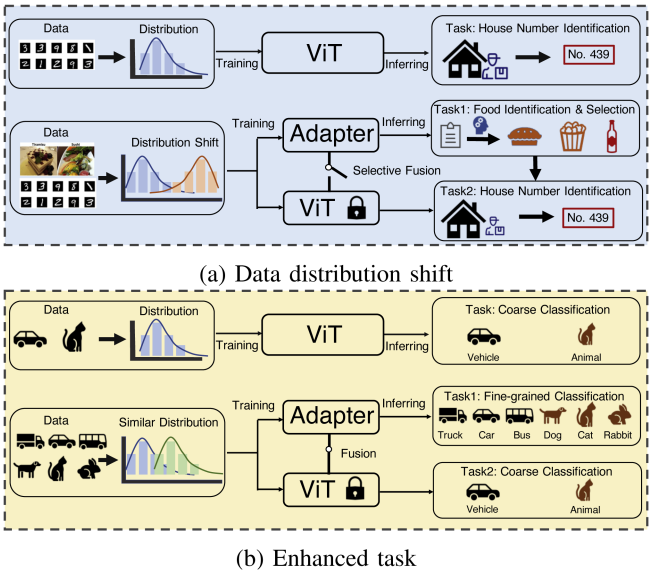


Fig. 1. An illustration of two scenarios for continual knowledge adaption for mobile vision understanding. (a) Data distribution shift. (b) Enhanced task.

Evolved with application operations, the data availability would get more comprehensive and the requirements of the original service will be enhanced over time, as shown in Fig. 1(b). For instance, in situations involving in-home health care, the data pertaining to patients can progressively become more comprehensive and intricate. Hence, the applied model requires enhancement to incorporate new symptoms, ensuring accurate and precise diagnoses for computation-limited home devices. To this end, one would naturally turn to training a new model or continuously training (CT) the existing one. This would also abandon the knowledge learned from the previous data, which, as a result, degrades its performance on the ongoing service. Yet, the additional costs associated with managing two separate models and the potential service disruption due to model training imply that the continual training of an entire model is not recommended.

Co-designing the model architecture and training strategy for deep learning vision models is challenging, especially when dealing with limited computing power and scarce data availability. Moreover, adapting the evolving data distributions complicates this endeavor. Previous works usually use a continual train strategy to update models with new data, but this can lead to catastrophic forgetting of previously learned knowledge [16]. The continual learning strategy sheds light on learning from the new data without forgetting the previous one. Nevertheless, this particular area of research tends to concentrate either on training models using powerful server-side computing resources [17] or on unsupervised/semi-supervised algorithms [18], overlooking the specific requirements of mobile continual knowledge learning.

In this paper, we tackle the above challenges by designing a Mixture-of-Experts (MoE) adapter and a knowledge fusion mechanism for continual adapting with limited computation resources and data availability, which is an extended version of our previous attention-based adapter [19]. Distinct from the traditional CT paradigm, the adapter features lightweight trainable components that are parallel to the existing models [20], [21]. This enables the model to learn new data distribution without interrupting the current service. Further, the sparsely gating mechanism of the MoE allows for an increase in the number of adapter parameters without adding to the computational burden. This benefits an enhanced representation learning ability for the adapter while not concerned with the limited on-chip requirements for edge devices [22]. During the adaptation process, the original model is frozen, in case of catastrophic forgetting of learned knowledge. At the same time, the trainable adapter encodes the new data, which consists of self-attention and MoE layers. The adapter features less trainable parameters and is efficient for training on mobile devices. A cross-attention block combines the learned old knowledge with new knowledge from new data, aiming at utilizing the previous knowledge to facilitate learning the new data, aiming at boosting the convergence speed and enhancing the performance. Extensive experiments on CIFAR-10, CIFAR-100, Food-101, and SVHN datasets show that the proposed method achieves a trade-off between the new and old data performance, as well as between the training process and the service continuity. Contributions of this work are summarized as follows:

- We explore a novel topic of continual knowledge adaption that adjusts the existing model on the multi-distribution data under constrained computation resources and data availability.
- We advance an MoE adapter and knowledge fusion module in a lightweight and efficient training style.
- Extensive experiments on four real-world datasets demonstrate that our method can enhance the model's learning ability and service continuity over new data distributions.

The rest of this paper is organized as follows. A brief review of the recent research work and advanced methods related to the adaption methods and the Mixture-of-Experts is presented in Section II. The description of the challenges in constrained continual adapting is discussed in Section III. The proposed method and algorithm implementation are explained in Section IV. The experimental results and ablation studies are reported in Section V. The conclusions follow in Section VI.

II. Related Work

A. Parameter-Efficient Fine-Tuning

In the field of Natural Language Processing (NLP), Large Language Models (LLMs) typically contain a vast number of parameters [23], enabling them to excel in representation learning from extensive training datasets. For example, GPT-3 [24] has 175 billion parameters and has been the foundation model for many NLP tasks. However, this magnitude leads to extremely prohibitive computation costs when fine-tuning LLMs for specific downstream tasks or adapting them to new datasets. To avoid such costs, many parameter-efficient fine-tuning methods are proposed [25], [26]. For example, Houlsby et al. [27] designed an auto-encoder-like adapter module, with

skip-connection bridging the input and out features. They inserted one adapter between the feed-forward layer and the layer normalization layer of the original Transformer block, while another adapter was added after the two feed-forward layers that follow the Transformer block. The LoRA method [20] is similar to the approach [27] for adapting to new data but with two key innovations. First, the weight increments from the new data adaptation are learned through low-rank decomposition matrices. Second, these low-rank matrices are added to the original weights during inference, ensuring no additional inference latency is introduced. Visual prompt tuning [28] introduces a far more efficient way to adapt to the new data. Different from the adapters, the append learnable tokens in the input space have less than 1% of the model parameters and are tuned with the task-specific head simultaneously. The approach of ViT-Adapter [21] is similar to ours in that it focuses on the image domain, but it emphasizes adapting a classification-specific ViT model for downstream dense prediction tasks. This contrasts with our approach to address data distribution shift issues. Specifically, ViT-Adapter introduces several components alongside the adapter and fusion module, including a Spatial Prior Module (SPM) that leverages the strong local spatial feature extraction capability of convolutional operators, as well as an injector that integrates the feature maps from the SPM into the ViT. Different from the aforementioned methods that focus on LLMs or vision models, we study scenarios involving mobile data distribution shifts. Our goal is to learn well from both new and existing data, which necessitates attention not only to adapter design but also to fusion mechanisms. Additionally, to deal with the computational constraints, we develop methods that decouple parameter growth from the increase in computation.

B. *Mixture-of-Experts*

The large number of parameters in attention-based models is crucial for effectively learning knowledge from data samples, as highlighted by the scaling law [29], provided with a sufficient computation budget and good training samples. While the concept of conditional computation can encourage the model to enlarge the number of parameters without increasing the computation amount [30]. This is achieved by activating only a partial of the parameters determined by the input tokens' choice. Shazeer et al. [31] introduced sparsely-gated MoE layer as the component of an LSTM language model, from when the MoE technique becomes the de-facto method to scale the LLMs, e.g., Mixtral 8×7B [32]. The MoE consists of two main components: multiple experts with separate parameters and a routing/gating network. An input token passes through the gating network, which determines the probabilities of selecting the specific experts. The chosen experts then process the token, and the outputs are aggregated, and weighted by the routing probabilities. For the implementation details, GShard [33] is the first trail to implement MoE into the Transformer. It replaces the feed-forward layer in every two encoder blocks with an MoE layer and employs Top-2 routing, as shown in Fig. 2. Switch Transformer [34] further lowers the computation and communication costs by selecting only one expert for each token, and

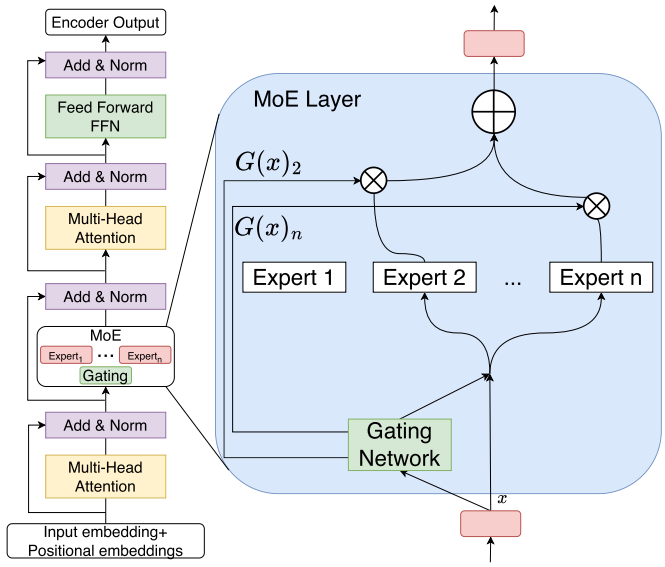


Fig. 2. The illustration of a mixture-of-experts transformer.

increases the stability by introducing an expert capacity and a load balance auxiliary loss. Zhou et al. [35] discovered that selecting more complex block configurations, rather than adhering to a fixed pattern of alternating sparse and dense layers, proves to be more efficient. Additionally, optimizing the architecture, sparsity, and routing mechanisms within the sparse layers is crucial for achieving optimal performance. To this end, they developed a block-wise search space that encompasses factors such as layer types (Attention, MoE, or feed-forward networks), model dimensions, and the number of attention heads, among others. They then employed a regularized evolutionary search algorithm to identify a promising block configuration, which can be stacked and scaled to construct an MoE Transformer. Mobile V-MoEs [36] tries to scale down the Vision Transformers to fit the low computing-power mobile devices. It routes the entire image to the selected images rather than a per-token routing strategy for saving the computation. However, additional super-class information is required to supervise the training to ensure performance. The aforementioned methods only focus on optimizing the design of MoE Transformers, while the work [37] explored the potential of MoE in the domain of continual learning of Vision-Language models like CLIP, aiming at remedying the issues of losing the zero-shot transfer ability of other continual learning methods. Different from the perspective of designing dedicated MoE Transformers and enhancing the CLIP zero-shot transfer ability on multi-task learning, we focus on applying the MoE to constrained continual knowledge adaption, that tackles the data distribution shift and satisfies the enhanced performance requirements for an existing model.

We have further made a detailed comparison between the two aforementioned techniques in Table I. We can see that both methods are not suitable for our research scope in these aspects. For the LoRA method, though it has great parameter training efficiency, its motivation is located in adapting the existing large models to the downstream task, which is more

TABLE I
LoRA vs. MoE: Key Comparisons

Aspect	LoRA	MoE
Parameter Efficiency	Low-rank	Sparse experts
Compute Overhead	Per rank	Per activated experts
Adaptation Scope	Task-specific	General-purpose
Input-Dependent Routing	None	Gating
Scalability	Fixed trainable parameters	Expert scaling

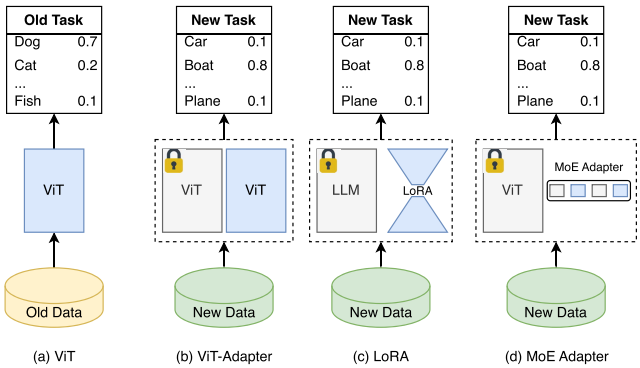


Fig. 3. The illustration of the normal adapter, LoRA method, and the proposed MoE adapter. The grey color and the lock icon indicate the parameter is not updated.

like a one-time post-train paradigm. However, in our continual knowledge adaptation scenario, we have to learn the model under constantly changing data. For the MoE, it is a modification to the data encoder part of the model by replacing the dense feed-forward layer with a sparsely activated experts layer. And this change would scale up the parameters while keeping the computation low by setting the number of activated experts. However, modifying the original data encoder would drop the learned knowledge on the previous data, which is not suited for edge continual knowledge adaption. Differently, in this paper, we adopted an adapter design with MoE architecture that runs along with the original model and retains the learned knowledge, while at the same, the MoE-adapter enables large parameters to learn on the new data and not trigger large computations. An illustration regarding the normal adapter, LoRA method, and our MoE adapter is presented in Fig. 3.

III. Challenges in Constrained Continual Knowledge Adaption

In this section, we outline the challenges of mobile environments with constrained computational resources.

Constrained Computation: In real-world applications, deep learning vision models are usually deployed on mobile devices with limited computation resources. Most of the time, users engage with computing applications, such as face recognition and Optical Character Recognition (OCR) translation, through computation-constrained electrical gadgets like smartphones or tablets [38]. Even the powerful computing platform of autonomous driving sensing, HW 3.0 from Tesla, has only 72 TOPS of computing power and 512 MB of on-chip memory, which are respectively 3/10 and 1/24 of a desktop RTX 3080 GPU. Thus, the limited computation resources on those mobile devices present a significant challenge for running the vision applications. Furthermore, the full deployment process for such resource-intensive services typically consists of two phases: training and inference. Inference involves only storing the model weights and executing the forward pass, whereas training is much more resource-intensive. This is because it requires storing the gradients and performing the backward pass to optimize the model. Due to the challenges of limited computational resources at the edge, the training process is typically conducted on cloud servers, where ample resources are available. Once training is complete, the model is distributed to mobile devices on the user side for local inference using on-device data. Alternatively, users' data can be collected and sent to the cloud for model inference.

Because mobile devices are typically near the data source, training models on them can effectively reduce communication costs of data transmission and lower the risk of privacy breaches. Moreover, performing computation close to the data source enables the detection of data distribution shifts and the timely update of model weights when new data arrives. As a result, it is desirable to accommodate the training strategy to fit the constrained computation resources on the edge side, utilizing the benefits of near-data source computing.

Limited Data Availability: A shared issue in real-world applications is the difficulty in collecting sufficient and high-quality data for training a well-performed deep learning model, including raw data and corresponding labels. For one aspect, gathering raw data and transforming it into useful and trainable data is both laborious and time-consuming. Moreover, there are privacy concerns, with users being unwilling to share their data. All of these makes the preparation of a wealthy dataset much harder. Another key aspect is that most of the applications are on a small scale in the form of a limited number of users and unique knowledge that is specific to the domain. For example, the healthcare services are dedicated to certain groups of patients with unique symptoms and medical knowledge. In the industry digital twin system, the collected data is unique to the specific production process and the number of the data is on a per-plant scale. This could further complicate the collection and limit the data quantity. Thus, it is necessary to develop training strategies under the limited data availability.

Continual Adaption: The available data is non-stationary and could vary over time. Thus, the model needs to be continuously updated to ensure the performance of the model over new data.

The change in the data distribution is usually presented in two forms. One is that the new data distribution could be different from the previously learned one. This is due to the evolution of the data source or the change of the application environment throughout the model service cycle. For example, the new demand which is stemmed from the continuously growing business, or the patient's symptoms are varying over time. As a result, the model should adapt to the new data distribution, at the same time maintain the ability to serve the previous data. The other is the requirement of more comprehensive knowledge

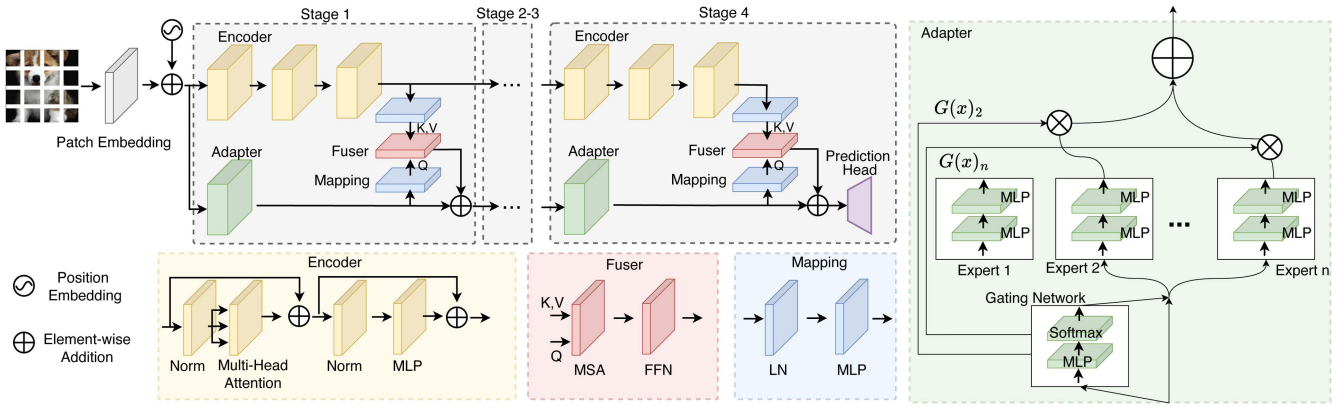


Fig. 4. The architecture illustration of the proposed framework. The adaption process starts by dividing the pre-trained ViT-Tiny model into four stages. For each stage, a mixture-of-experts (MoE) adapter is employed to adapt on the new data, while the weights of the pre-trained ViT encoder block are fixed, and it only performs inference on the new data. At the end of each stage, an optional cross-attention-based feature fusion block is employed to facilitate the new data learning by utilizing the learned knowledge from the ViT-Tiny model. The features from the two branches are summed element-wisely.

of an enhanced task, as is shown in Fig. 1(b). For example, after a long time of running period, the data collected from the same assembly line is much richer than it was put into production. Also, the quality demands of the production process are enhanced in the form of requiring the model to have higher accuracy or identify more categories. In this case, the model needs to be updated to grab more comprehensive knowledge.

For both forms, the previously learned knowledge is still valuable. For the first one, though the new data distribution is totally distinct from the previous one, the model still needs to function well on both the new and old data distributions, especially the scenarios in which data are changing in cycles. For the second, the learned old knowledge could be also useful to facilitate the learning of the new data, because the new one is still related to the previous one in some patterns and semantics.

IV. Mixture-of-Experts as Continual Knowledge Adapter

In this section, we address the above challenges by embracing an adapter-based method for the efficient continual adapting of mobile vision tasks, which features constrained computing resources and small-scale datasets. The core design is to build the adapters based on the MoE [31] technique, which is a promising solution for enlarging the number of parameters to boost the data learning capacity while not incurring large computations for mobile devices. The data adaption procedure is based on a well-trained vision model on one specific dataset, denoted as the old dataset. Then, it starts with learning the new coming data by the trainable adapters along with the trained models, whose parameters are fixed to retain the learned old data knowledge. During the adaption, the previous well-trained vision models can still function well for the old data distribution. Further, a knowledge fusion module is employed to utilize the old knowledge from the well-trained model, which may fasten the convergence of the adaption process. The overall architecture of the proposed framework is illustrated in Fig. 4.

A. Vision Transformer for Mobile Computing

Vision Transformer (ViT) [6] is an attention-based vision model, which has achieved impressive representation ability and astonishing performance on various vision tasks [39], [40], and outperforms the convolutional counterparts [5]. Such success is attributed to the attention mechanism, whose enhanced representation capacity can model the global relations between patch tokens from out of an image. Usually, The ViT has a dense parameter count and requires a large quantity of data samples to be fully trained from scratch. However, we can fit the dense ViT model on constrained computing resources, such as mobile devices, and train it with the small-scale dataset, by adjusting the number of embedding dimensions and the number of heads for each attention block. In this section, we first adopt the minimum parameter version of the ViT model, i.e., ViT-Tiny, to learn the knowledge from a specific dataset.

The procedure starts with dividing the input image into patch tokens. First, the input 2D image $\boldsymbol{x} \in \mathbb{R}^{H\times W\times C}$ is reshaped into a sequence of patches of $\boldsymbol{x}_{\mathrm{p}} \in \mathbb{R}^{N\times(P^2\cdot C)}$, where (H, W) is the image width and height, C is the number of channels, and (P, P) is the patch size. The number of patches is calculated by $N = HW/P^2$. Then, the sequence of patches is mapped into a fixed D dimension by using a linear projection layer, parameterized by $\boldsymbol{W}$, which is concatenated with a learnable `[class]` embedding $\boldsymbol{x}_{\text{class}} \in \mathbb{R}^D$ at the head of the sequence later. The `[class]` embedding is used to attend to the semantics of various patches and calculate the final categories probabilities. Subsequently, the sequence is added with a positional encoding to retain the spatial information and we obtain the initial input $\boldsymbol{z}_0$. The above process follows,

$$\boldsymbol{z}_0 = \text{Concat}(\boldsymbol{x}_{\text{class}}; \boldsymbol{x}_{\mathrm{p}}\boldsymbol{W}) + \boldsymbol{E}_{\text{pos}}, \tag{1}$$

of which $\boldsymbol{W} \in \mathbb{R}^{(P^2\cdot C)\times D}$ denotes the weight matrix of the linear projection layer where the bias parameters are omitted for brevity; $\boldsymbol{E}_{\text{pos}} \in \mathbb{R}^{(N+1)\times D}$ is the 1-D learnable positional encoding matrix. The weights of the linear projection layer and the positional encoding are randomly initialized.

The basic component of the ViT model is the Transformer encoder that consists of alternating layers of the Multi-head Self-attention layer (MSA) and the Feed Forward Network (FFN). LayerNorm(LN) as well as the residual connection is applied for each layer. Specifically, for a single-head attention mechanism, it adopts the scaled dot-product attention following,

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{2}$$

where the $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are the `query`, `key` matrices of dimension d_k, and `value` matrix of dimension d_v, respectively. They are yield by linearly projecting the input $\boldsymbol{e}$ of dimension d_{model} as follows,

$$\boldsymbol{Q} = \boldsymbol{e}\boldsymbol{W}^Q, \tag{3}$$

$$\boldsymbol{K} = \boldsymbol{e}\boldsymbol{W}^K, \tag{4}$$

$$\boldsymbol{V} = \boldsymbol{e}\boldsymbol{W}^V, \tag{5}$$

where the projecting parameters are $\boldsymbol{W}^Q \in \mathbb{R}^{d_{model}\times d_k}$, $\boldsymbol{W}^K \in \mathbb{R}^{d_{model}\times d_k}$, and $\boldsymbol{W}^V \in \mathbb{R}^{d_{model}\times d_v}$, respectively. For single-head attention, the dimensions for `query`, `key` and `value` matrices are the same as the input, i.e., $d_k = d_v = d_{model}$. To capture the information from various representation subspaces, we repeat the attention process in (2) for h times with different learnable query, key and value matrices in parallel. After obtaining all of h outputs, dubbed as head, they are concatenated and projected to form the final outputs, which are the multi-head attention as follows,[1]

$$\text{MSA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\boldsymbol{W}_o, \tag{6}$$

$$where\, \text{head}_i(\boldsymbol{e}) = \text{Attention}(\boldsymbol{e}\boldsymbol{W}_i^Q, \boldsymbol{e}\boldsymbol{W}_i^K, \boldsymbol{e}\boldsymbol{W}_i^V). \tag{7}$$

The weight of the final projection layer is $\boldsymbol{W}_o \in \mathbb{R}^{hd_v\times d_{model}}$.

The fully connected FFN contains two layers of linear projection with a ReLU activation function in between:

$$\text{FFN}(\boldsymbol{x}) = \text{Max}(0, \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2. \tag{8}$$

Before input into the MSA and FFN layers for each block, we apply Layer Normalization(LN) to the inputs $\boldsymbol{x}$ follows,

$$\text{LN}(\boldsymbol{x}) = \frac{\boldsymbol{x} - \text{Mean}(\boldsymbol{x})}{\sqrt{\text{Var}(\boldsymbol{x}) + \epsilon}} * \gamma + \beta, \tag{9}$$

where both γ and the β are the learnable affine transform parameters.

After obtaining the input embedding $\boldsymbol{z}_0$ following (1), it is then fed into a stack of L encoder sub-blocks to iteratively extract the latent representation of the input. The encoder sub-block is consisted by the following two blocks:

$$\boldsymbol{z}'_l = \text{MSA}(\text{LN}(\boldsymbol{z}_{l-1})) + \boldsymbol{z}_{l-1}, \qquad l = 1, \ldots, L, \tag{10}$$

$$\boldsymbol{z}_l = \text{FFN}(\text{LN}(\boldsymbol{z}'_l)) + \boldsymbol{z}'_l, \qquad l = 1, \ldots, L. \tag{11}$$

Finally, the processed `[class]` token out of the output embedding $\boldsymbol{z}_L$ from the last encoder sub-block is fed into a class-specific prediction multi-layer perceptron head to obtain the final prediction $\hat{y}$. This token now encompasses the selected information from all of the patches, known as the attending process, which is used to calculate the category probabilities for the input.

$$\hat{y} = \text{Softmax}(\text{LN}(\boldsymbol{x}_L[0])\boldsymbol{W}_c), \tag{12}$$

where $\boldsymbol{x}_L[0]$ refers to the processed `[class]` token $\boldsymbol{x}_{\text{class}}$. We focus on the image classification task and adopt the soft cross-entropy loss to supervise the training process following,

$$l_n = -y_n \log \frac{\exp(\hat{y}_n)}{\sum_{c=1}^{C} \hat{y}_{n,c}}, n = 1, \ldots, N. \tag{13}$$

where C is the number of the categories given a specific image classification task. The final loss is obtained by calculating the mean value for all l_n from a mini-batch of size N, $\mathcal{L}_{cls} = \text{mean}(\sum_{n=1}^{N} l_n)$.

The flexibility to tune the number of heads in (10) and the hidden dimension D in (11) makes it possible to scale down the ViT model to fit the constrained mobile devices, as well as preventing over-fitting on the small-scale dataset. As a result, we adopt the tiny version of the ViT model as the backbone of our work, and make it learn the knowledge for specific datasets at first. The model specifications of ViT-Tiny are shown in the left column in Table II, as well as the standard base version in the right column. Specifically, we set the number of heads to 3, and the hidden dimension $D = 192$. Given the FFN embedding ratio is 4, the dimension of the latent feature for (8) is $4 \times 192 = 768$. In our work The ViT-Tiny is first trained on the existing dataset to serve the requirements of service. When the new data comes, the parameters of ViT-Tiny are fixed and we apply the adapter to learn knowledge from the new data.

TABLE II
VIT VARIANTS SPECIFICATIONS

Terms	Values	
	ViT-Tiny	ViT-Base
Attention heads	3	12
Embedding dimension	192	768
FFN embedding ratio	4	4
Depths	12	12
Input Image Resolution	224	224
Patch Size	16	16
Number of Parameters	5.52M	85.78M

B. Attention-Based Adapter

As discussed in Section III, we should update the existing model to persist the previously learned knowledge while learning well the new knowledge. To efficiently learn new knowledge without catastrophically forgetting the previously learned one, as well as not disrupting the deployed services, we propose a co-design of the model architecture and learning strategy. The pre-processing of the input image strictly follows the steps in Section IV-A, that image sampled from the new dataset is reshaped into a sequence of patches, $\boldsymbol{x}_{\text{p}}^{new} \in \mathbb{R}^{N\times(P^2\cdot C)}$. Then it is mapped to a fixed dimension of D with the same projection layer in (1) and added with positional encoding. Specifically, the

[1]For the multi-head attention, $d_k = d_v = d_{model}/h$

weight $\boldsymbol{W}$ of the projection layer is directly inherited from the previous learning process and is fine-tuned during the adaption process. Since the adaption process aims to perform on the new coming data, we replace the previous learnable `[class]` (as (1)) token with a new one with randomly initialized parameters, and concatenate it at the beginning of the new data input sequence to form the input embedding $\boldsymbol{z}_0^{new}$.

Then, we designed a lightweight trainable adapter that is parallel to the existing ViT-Tiny model. Specifically, we divide the stacked ViT-Tiny encoder sub-blocks into four stages so that each stage consists of $L/4$ encoder sub-blocks. During adapting to the new data, the weights of the existing encoder sub-blocks are fixed so that the learned knowledge is retained. Along with the sub-blocks in each stage, We added only one adapter block with trainable parameters to learn from the new data. The architecture of the adapter block includes an MSA layer and an FFN layer, following,

$$\boldsymbol{z}_s'^{new} = \mathrm{MSA}(\mathrm{LN}(\boldsymbol{z}_{(s-1)}^{new})) + \boldsymbol{z}_{(s-1)}^{new},\ s = 1, \ldots, 4, \quad (14)$$

$$\boldsymbol{z}_s''^{new} = \mathrm{FFN}(\mathrm{LN}(\boldsymbol{z}_s'^{new})) + \boldsymbol{z}_s'^{new},\ s = 1, \ldots, 4, \quad (15)$$

where the subscript s is the stage indicator. The $\boldsymbol{z}_s''^{new}$ is the output from the adapter block at the s_{th} stage, which is fused with the output from the $\frac{L}{4}s_{\mathrm{th}}$ encoder block. The parameters of the adapter are trainable. For each stage, the existing encoder sub-blocks perform inference layer by layer. The output $\boldsymbol{z}_{\frac{L}{4}s}$ at each end of the normal encoder stage is fused with $\boldsymbol{z}_s''^{new}$:

$$\boldsymbol{z}_s^{new} = \mathrm{Fusion}(\boldsymbol{z}_s''^{new}, \boldsymbol{z}_{\frac{L}{4}s}),\ s = 1, \ldots, 4, \quad (16)$$

where the Fusion module will be discussed in Section IV-D.

The adapter module is designed to be lightweight. Compared to training the full twelve encoder blocks of the ViT-Tiny, the four adapter blocks require fewer computation resources, making it easy to integrate on constrained mobile devices. Further, modular design enables the adapter to be an add-on to the existing model in a distributed manner. The adapter can be deployed to other devices, with the intermediate features transferred via communication. This not only enables more flexibility in the deployment but also avoids the disrupting of the training on the existing service.

C. MoE-Based Adapter

The model size, specifically the sufficiently large number of parameters, is central to the ability to learn well the knowledge from the data samples, as is pointed out in scaling law [29]. However, the constrained resource of the mobile edge devices limits the deployment of models with dense parameters. For example, in Section IV-B, the number of the newly added attention-based adapter blocks is only one for each stage. This means the entire model only has 1/4 of the learnable parameters of the original ViT-Tiny model, which potentially limits the performance of new knowledge adaptation. Due to the constrained computing power and demands for enlarging the parameters of the adapter, a solution is required to decouple the computing and the parameters.

Conditional computing sheds light on fulfilling such a demand that it can increase the model parameters without a proportional increase in computational costs [31]. Specifically, the idea is that parts of the model are activated on a per-example basis. To achieve this, we replace the FFN layer in the attention blocks with a sparsely activated MoE layer. The MoE layer consists of M experts, $E_1, \ldots, E_M$, where each expert is modeled by an FFN and equipped with separate parameters. The expert handles the selected patch tokens out of an image. This is achieved by routing the token to the specified expert according to its endogenous local semantic information by a self-gating mechanism. The gating network takes as input the patch token and then outputs a sparse M-dimensional vector $G(\boldsymbol{x})$, indicating the selection of the experts. Specifically, inside the MoE layer, denote the output from m_{th} expert by $E_m(\boldsymbol{x})$, the output of the MoE layer is yielded as:

$$\boldsymbol{z} = \sum_{m=1}^{M} G(\boldsymbol{x})_m E_m(\boldsymbol{x}). \quad (17)$$

In our implementation, the number of experts m is set to 16. And we substitute the FFN layer (15) with the MoE layer (17).

For the self-gating design, a plain and non-sparse implementation is to multiply the input token with trainable weights $\boldsymbol{W}_g$ and be applied with a Softmax function following:

$$G(\boldsymbol{x}) = \mathrm{Softmax}(\boldsymbol{x}\boldsymbol{W}_g). \quad (18)$$

Following the implementation of noisy top-k gating [31], two elements, sparsity, and noise, are added to the gating network in (18). The sparsity limits the computation to the selected experts, which is achieved by the top-k gating function. And the injected noise can bring smoothness and enable back-propagating of the load-balancing regulation term. Specifically, first, we add Gaussian noise whose magnitude is tunable and adjusted according to the input token itself:

$$\boldsymbol{x}_{noise} = \boldsymbol{x}\boldsymbol{W}_g + \mathrm{GaussianNoise} \cdot \mathrm{Softplus}(\boldsymbol{x}\boldsymbol{W}_{noise}). \quad (19)$$

Then, we select the top k logits according to the noisy input and apply the Softmax function to obtain the final expert selection probability:

$$G(\boldsymbol{x}) = \mathrm{Softmax}(\mathrm{Top}k(x_1), \ldots, \mathrm{Top}k(x_M)), \quad (20)$$

$$\mathrm{Top}k(x_m) = \begin{cases} x_m, & \text{if } x_m \text{ is in the top } k \text{ elements of } \boldsymbol{x}_{noise}, \\ 0, & \text{otherwise.} \end{cases} \quad (21)$$

To prevent the gating network from favoring the same few experts, which is due to the self-reinforcing mechanism that favored experts are trained more quickly and the gating network becomes biased towards them, we apply constraints by using the expert importance regulation termed as $\mathcal{L}_{imp}$. First, an expert's importance for a batch of training examples is measured by the total gate values assigned to it across the batch $\mathcal{X}$, following:

$$\mathrm{Imp}(\mathcal{X})_m = \sum_{\boldsymbol{x} \in \mathcal{X}} G(\boldsymbol{x})_m. \quad (22)$$

Then, the importance regulation loss $\mathcal{L}_{imp}$ is obtained by calculating the square of the coefficient of variation of the importance values for a set of m experts:

$$\mathcal{L}_{imp} = \mathrm{CV}(\mathrm{Imp}(\mathcal{X})_1, \ldots, \mathrm{Imp}(\mathcal{X})_M)^2. \tag{23}$$

From the experts' perspective, the distribution of training examples can vary greatly. Some experts may be assigned a small number of highly weighted examples, whereas others might receive many examples with minimal weight. Following [31] we apply load balance regulation termed as $\mathcal{L}_{loads}$ to encourage each expert to receive roughly the same number of tokens. Unfortunately, the load term for each expert is non-differentiable due to the sum of gate values exceeding zero following:

$$\mathrm{Load}(\mathcal{X})_m = \sum_{\boldsymbol{x}\in\mathcal{X}} \mathrm{Count}(G(\boldsymbol{x})_m > 0). \tag{24}$$

To enable the back-propagation of load term, we first compute the probability that a gate value is in the top k given different random noise, following

$$P(\boldsymbol{x}, m) = Pr((\boldsymbol{x}_{noise})_m > k_{\mathrm{th}}(\boldsymbol{x}_{noise})), \tag{25}$$

where $(\boldsymbol{x}_{noise})_m$ is the noisy gate logits of m_{th} expert; $k_{\mathrm{th}}(\boldsymbol{x}_{noise})$ denotes the k highest value of $\boldsymbol{x}_{noise}$. This is equivalent to calculating the probability of m_{th} expert being selected for one example, which is calculated by:

$$P(\boldsymbol{x}, m) = \Phi\left(\frac{(\boldsymbol{x}\boldsymbol{W}_g)_m - k_{\mathrm{th}}(\boldsymbol{x}_{noise})}{\mathrm{Softplus}((\boldsymbol{x}\boldsymbol{W}_{noise})_m)}\right), \tag{26}$$

where Φ is the standard normal distribution.

Then we can approximate the load of the m_{th} expert on a batch of $\mathcal{X}$ by calculating the sum of the probabilities of selecting that m_{th} expert for each example in the batch, instead of simply counting the number of examples assigned to it:

$$\mathrm{Load}(\mathcal{X})_m = \sum_{\boldsymbol{x}\in\mathcal{X}} P(\boldsymbol{x}, m). \tag{27}$$

The final load balance regulation term is defined as the coefficient of variation of loads from all of m experts, following:

$$\mathcal{L}_{loads} = \mathrm{CV}(\mathrm{Load}(\mathcal{X})_1, \ldots, \mathrm{Load}(\mathcal{X})_M). \tag{28}$$

Finally, the task loss, as well as the two regulation terms, are combined to supervise the training of the adapter jointly:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda(\mathcal{L}_{imp} + \mathcal{L}_{loads}), \tag{29}$$

where the loss coefficient λ is set to 0.01.

D. Knowledge Fusion

In some circumstances, the previously acquired knowledge may present helpful information for the new data adapting. Even though the data distribution is different, the trained model may have a basic understanding of recognizing the colors, shapes, and textures of the objects. To fully exploit the learned knowledge and facilitate the learning of the new data, we design a cross-attention-based knowledge fusion mechanism to combine the output features from each stage of the backbone and the adapter.

Specifically, we use the features of the adapter block as the `query` embeddings $\boldsymbol{q}$, and the features of the encoder block at the end of each stage as the `key` and `value` embeddings, $\boldsymbol{k}$ and $\boldsymbol{v}$, respectively. Notice, the weights of the encoder blocks are fixed during the training of the adapter. So fusing the intermediate features of the encoder equals applying the pre-trained knowledge to the new coming data. The cross-attention mechanism is to compute the attention scores between the `query` and the `key`, so that to obtain the information on which part of the old knowledge is important to the new data. Then the attention scores are used to weight the `value` and the weighted value matrices are added to the adapter's output to obtain the fused feature. Finally, the fused feature is fed into an FFN layer to obtain the final output $\boldsymbol{e}_s$ for s_{th} stage. The Fusion block in (16) is formulated as:

$$\boldsymbol{e}'_s = \mathrm{MSA}(\boldsymbol{q}_s, \boldsymbol{k}_s, \boldsymbol{v}_s) + \boldsymbol{q}_s,\ s = 1, \ldots, 4, \tag{30}$$

$$\boldsymbol{e}_s = \mathrm{FFN}(\mathrm{LN}(\boldsymbol{e}'_s)) + \boldsymbol{e}'_s,\ s = 1, \ldots, 4. \tag{31}$$

The output of the encoder block at the end of the s_{th} stage (the $\frac{L}{4}s_{\mathrm{th}}$ layer), $\boldsymbol{z}_{\frac{L}{4}s}$ (11), first goes through an LN layer and then a linear projection layer to obtain the `key` and `value` embeddings $\boldsymbol{k}_s$ and $\boldsymbol{v}_s$. The `query` $\boldsymbol{q}_s$ comes from the adapter output $\boldsymbol{z}_s^{\prime\prime new}$, processed by an LN layer and a linear projection layer, either.

For each stage, the obtained features $\boldsymbol{e}_s$ are fused to the adapter output $\boldsymbol{z}_s^{\prime\prime new}$ by point-wise addition. Once obtained the features in the final stage, the new `[class]` token out of it are fed into a new prediction MLP head to obtain the final prediction, resembling the (12). In some cases, the fusion module is optional to avoid slight performance degradation on the new data, especially when the model is first pre-trained on a tougher task and then transferred to a simple one.

V. Performance Evaluation

In this section, we evaluate the two proposed adapter methods under two scenarios, i.e., the dataset distribution shift, and the enhanced task requirements on the more comprehensive dataset, as discussed in Section III respectively. We also give ablation studies of the key components and design insights. In detail, the experiments are performed on four datasets including CIFAR-10, CIFAR-100, SVHN, and Food-101. All of the experiments are implemented on one computing workstation with one AMD EPYC 7402 processor and one NVIDIA RTX 3090 GPU, under the PyTorch 1.8.1 and CUDA 11.1 environment.

A. Scenarios and Dataset Settings

As the time evolves, the data distribution and service requirements to which the trained model is deployed may present non-stationary nature. But for the application and its accompanying model's development cycle, to tackle such dynamics, usually, the application would first publish one version of the model with certain performance, which is trained on the currently available data. The application with this model would run for a certain time to serve the user normally. While at the same time, the application developer would collect new data and continually

TABLE III
SVHN AND FOOD-101 DATASETS STATISTICS

Attributes		SVHN	Food-101
Number of Categories		10	101
Number of Images	Training	73,757	75,750
	Testing	26,032	25,250
Channel Distributions	Red	0.4377 ± 0.1980	0.5458 ± 0.2717
	Green	0.4438 ± 0.2010	0.4443 ± 0.2750
	Blue	0.4728 ± 0.1970	0.3443 ± 0.2796
Image Size		32 pixels ×32 pixels	maximum side of 512 pixels
Category Examples		"0", "1", "2", "3", "4", ..., "8", "9"	"pizza", "miso_soup", "chocolate_cake" ..., "pad_thai", "waffles"

TABLE IV
CIFAR-10 AND CIFAR-100 DATASETS STATISTICS

Attributes		CIFAR-10	CIFAR-100
Number of Categories		10	100
Number of Images	Training	50,000	50,000
	Testing	10,000	10,000
Channel Distributions	Red	0.4914 ± 0.2471	0.5071 ± 0.2675
	Green	0.4822 ± 0.2435	0.4867 ± 0.2565
	Blue	0.4465 ± 0.2616	0.4408 ± 0.2761
Image Size		32 pixels ×32 pixels	32 pixels ×32 pixels
Category Examples		"airplane", "automobile", "bird", ..., "cat", "dog"	"beaver", "orchids", "bee" ..., "bicycle", "tractor"

train the model offline to update its knowledge. After training the model and testing its robustness, the application publishes this model to serve the users further.

So, considering the data dynamics as well as the periodic model development cycle, we adopt two datasets with distinctively different distributions, i.e., Street View House Numbers (SVHN) and Food-101. The SVHN is a real-world dataset of house numbers cropped from pictures of house number plates. It has 73,757 and 26,032 32×32 RGB images of printed digits labeled from 0 to 9, for training and testing respectively. The Food-101 dataset comprises 101 distinct food categories, each containing 750 training images and 250 test images, resulting in a total of 101,000 images. The two datasets have distinctively different data distributions in that the color space and the object categories are entirely different. For SVHN, the mean values and standard variations of color digits for each RGB channel are 0.4377 ± 0.1980, 0.4438 ± 0.2010, and 0.4728 ± 0.1970, respectively. While for Food-101, the mean and standard variations for the RGB channel are 0.5458 ± 0.2717, 0.4443 ± 0.2750, and 0.3443 ± 0.2796, respectively. Besides, the labels are also different in not only the number of classes (10 for SVHN and 101 for Food-101) but also the semantic meanings: the SVHN merely contains colorful digits of the real-world house plate, while Food-101 contains a large number of daily food categories. The difference between the two datasets is listed in Table III for a clear comparison.

This distribution shift between the two datasets allows us to simulate the scenarios of the data distribution changing. For instance, imagine a scenario where a food delivery company initially deploys a model trained on the SVHN dataset to recognize house numbers, enabling delivery drivers to locate the correct address within a community efficiently. Over time, the company may expand this system to include food recognition capabilities, assisting drivers in accurately identifying orders when collecting them from restaurants. Further, considering the requirement that the deployment of the new function could not interfere with the existing one, and the fact that constrained computation power could not support training a new model for the dataset with a new distribution. As a result, the most efficient way is to adapt the existing SVHN model to the new distribution of the food images and let it perform well on both the dataset, instead of training a new model from scratch.

To simulate the scenarios regarding the enhanced task performance on more comprehensive datasets, we adopt CIFAR-10 and CIFAR-100 where the channel distribution of the data is quite similar, 0.4914 ± 0.2471, 0.4822 ± 0.2435, 0.4465 ± 0.2616 for the former, and 0.5071 ± 0.2675, 0.4867 ± 0.2565, 0.4408 ± 0.2761 for the latter. The comparison between the two datasets is shown in the Table IV. The two datasets also share some universal categories such as automobile, truck, cat, and dog for the CIFAR-10, as well as the mammals and vehicles for CIFAR-100. But CIFAR-100 is more comprehensive which constitutes a challenging task. For example, it contains many more numbers and diverse visual concepts, spanning from living animals, daily objects, and plants, presented by 20 superclasses. For each superclass, it has five fine-grained categories, which constitute a total of 100 categories than CIFAR-10's 10 categories, The partially shared visual concepts of the two datasets and the more rich categories enables us to simulate the scenarios of adapting the existing model for enhanced task performance with more fine-grained categories. In the initial stage of deployment, models are often trained on limited, small datasets that feature coarse-grained categories due to data constraints. As the service operates and more data becomes available, alongside increasing performance requirements, the provider may seek to enhance the model's performance by incorporating the recognition of more fine-grained categories. Since the model is already equipped with knowledge from the previous dataset, the design for adapting to an enhanced dataset should leverage this prior information to streamline the learning process. Our approach enables the model to adapt to new data by utilizing previously acquired knowledge, without downgrading the performance of the ongoing service.

B. *Performance on Datasets With Distributions Shift*

Scenarios and training settings: We adopt the SVHN and Food-101 datasets to evaluate the performance of the proposed adapter design under the data distribution shifting scenarios. Specifically, we first pre-train a ViT-Tiny model on the SVHN dataset, which we view as the old data. For the training details, we adopt AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, weight decay is set to 0.05. The batch size is 128. The learning rate is set to 1.25×10^{-4} and the model is trained for 100 epochs with the beginning 20 epochs as warm-up. When the model is fully trained on the training split of the SVHN dataset, it is then deployed and we use the Top-1 accuracy on the validation split as the performance metric for this time. Over time, the model is required to handle a new dataset with a distinct distribution

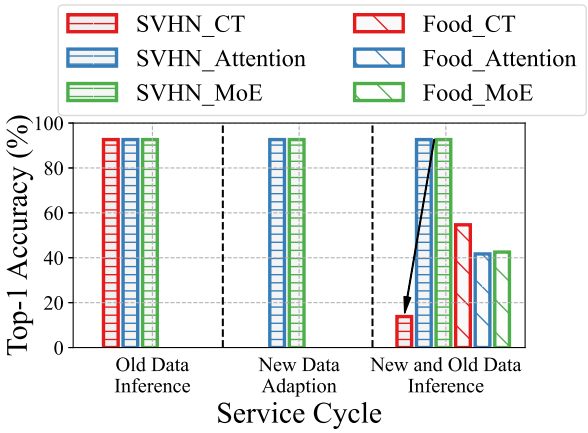


Fig. 5. Performance comparison for MoE (green) and attention-based (blue) adapters. Here, the SVHN (horizontal lines) is the old dataset while the food-101 (diagonal lines) is the new. The traditional model continual train (red) strategy is adopted as a comparison. The mean top-1 classification accuracy is reported.

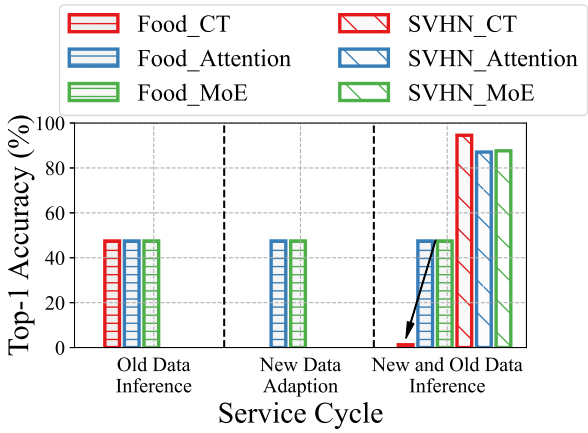


Fig. 6. Performance comparison for MoE (green) and attention-based (blue) adapters. Here, the food-101 (horizontal lines) is the old dataset while the SVHN (diagonal lines) is the new. The traditional model continual train (red) strategy is adopted as a comparison. The mean top-1 classification accuracy is reported.

such as Food-101. To maintain the performance of the previous SVHN and to continually learn on the Food-101, we apply our attention-based adapter and an enhanced version of the MoE-based adapter to encode the new data. Specifically, to avoid catastrophically forgetting, we fix the parameters of all twelve encoder blocks of (10) and (11), and only train the four newly added adapter blocks of (14) and (15) for Attention-based adapter; only train the (14) and (17) for MoE-based adapter, where both of the two adapters' parameters are initialized randomly. Further, the trained parameters of the patch projection layer of (1) are directly inherited from the pre-trained model and further tuned during the adaption process. The adaption process follows the hyper-parameters of the training process. After the new dataset adaption, we evaluate the performance of the adapted model on both the new and the old data.

The model Continual Train (CT) strategy is employed as a comparison, which is widely adopted when fine-tuning a pre-trained model to downstream tasks or data with new distribution. To be specific, we fully tune a pre-trained SVHN model on the Food-101 dataset with the same hyper-parameters of the adaption settings to constitute a fair comparison. Then we use the weights from the continual training process to evaluate their validation Top-1 accuracy on both the SVHN (Old) and the Food-101 dataset (New).

Due to the significant differences in distribution and magnitude between the two datasets, To investigate the impact of the order of the pre-training dataset on adapter design, we set the old dataset as the Food-101 and the new dataset as the SVHN. All experiments are performed five times independently with different random seeds to eliminate the randomness.

Evaluations and discussions: We first illustrate the model's entire service cycle as well as its performance at each stage, measured by the Top-1 classification accuracy, as shown in Figs. 5 and 6 for two different training orders, respectively. To be specific, for all the methods, we report the aforementioned three phases for the entire service cycle: first, the pre-trained model performs inference on the old data, reporting its Top-1 accuracy on the validation set of the old data; second, adapt the trainable adapters on the new dataset and continually train the pre-trained model on the new data, during which the model's Top-1 accuracy on the old dataset is reported; finally, the adapted or the continually trained models perform inference on both the new and old data, reported by the Top-1 accuracy on the validation sets of two datasets.

When the pre-train dataset is the SVHN (horizontal lines) and the new dataset is Food-101 (diagonal lines), the performance over service time is shown in Fig. 5. For the first phase shown in the left sub-block, since both the continually trained (red bar) and the adapter methods (blue and green indicate the Attention-based and the MoE-based adapter respectively) start with the same ViT-Tiny model pre-trained on SVHN, all of the three methods reach an average of 92.64% Top-1 accuracy on the validation split of SVHN. During the second phase in the middle sub-block, the new dataset Food-101 comes in, and the model begins to learn from it. For the continually trained strategy (red bar), the model applies the pre-trained weights to be tuned on the new data, so the model has to stop serving the old data and performing inference on the validation set of the old data. This downgrades the continuity of the service and users' satisfaction. However, for the adapters (blue and green bars), the original model does not need to stop inference and its performance on the old dataset will be consistent with the previous phase. This is because 1) the adaption only requires the original model to perform inference on the new dataset, the inference on the previous dataset will not be interfered; 2) only the parameters from adapters and the fusion module will be tuned to retain the model's knowledge of the previous data. As a result, the proposed two adapter designs can still maintain the previous performance on the old data of 92.64%, at the same time, they begin training on the new data. In the third phase, the continually trained method finishes training the full parameters and restarts to serve the old and new data. It achieves an accuracy of 54.68% on the new data, which is promising for the new task but at the cost of training the entire model weights of 5.52 M. Further, there

TABLE V
COMPARISON RESULTS ON THE SVHN AND FOOD-101 DATASETS

Method	New Food-101 (%)	Old SVHN (%)	New SVHN (%)	Old Food-101 (%)
ViT-Tiny/16@224px				
Train from the scratch	47.48 ± 0.49	92.64 ± 0.21	92.64 ± 0.21	47.48 ± 0.49
Continual train	54.68 ± 0.53 (↑ 7.2)	13.81 ± 5.90 (↓ 78.83)	94.57 ± 0.11 (↑ 1.93)	1.24 ± 0.13 (↓ 46.24)
Adapter				
Attention	41.70 ± 0.72 (↓ 5.78)	92.64 ± 0.21 (-)	87.08 ± 0.57 (↓ 5.56)†	47.48 ± 0.49 (-)
MoE	42.53 ± 0.32 (↓ 4.95)	92.64 ± 0.21 (-)	87.64 ± 0.43 (↓ 5.00)†	47.48 ± 0.49 (-)

All experiments are independently performed five times with different seeds. The mean and standard variation of Top-1 classification accuracy are reported. † indicates that the adaptation process does not utilize the Fusion module.

exists a catastrophic forgetting issue on the old data, that the accuracy drops from 92.64% to 13.81%, which is a significant magnitude of 78.83% (shown as the black arrow in Fig. 5). This drop is not acceptable for both the service user and the provider, which obviously deteriorates the model's performance. On the contrary, the adapter design not only remembers the old data, shown by maintaining the same high accuracy at all times, but also achieves an acceptable accuracy of 41.70% (blue diagonal bar) on the new data with only 2.58 M trainable parameters. The performance lag is probably due to we have only half of the trainable weights, and also the adapter is trained from scratch. While the continual train method inherits the weights that are already equipped with the knowledge of basic colors, shapes, and textures, making it easier to learn from the data. Also, we do not particularly tune the hyper-parameters but just keep the same settings with the continual train method, to constitute a fair comparison. With a more advanced MoE design, the Top-1 accuracy on Food-101 reaches 42.53% (green diagonal bar). This is brought by the expansion of the trainable parameters (8.35 M more) but still the same computation power as Attention-based counterpart. As is discussed in the Section IV, the MoE layer enables the decoupling between enlarging the number of parameters and increasing computation power, which paves the way for enhancing the learning ability when facing the data shift issue.

We further look at the effect of swapping the order of the pre-training dataset. In Fig. 6, where the Food-101 is the old data and SVHN is new, the conclusions are consistent with the observation from Fig. 5. The catastrophic forgetting is much more severe: after the continual training, the performance on the previous dataset drops to 1.24%, which represents a substantial decline of 46.25%. Despite the performance on new data being promising at 94.57%, the symptom of performance dropping to almost zero is unacceptable to the service providers and the users. On the contrary, our attention-based adapter maintains the Top-1 accuracy on the old data at 47.48% while achieving an acceptable accuracy of 87.08% on the new data. With a more advanced MoE design, the performance reaches a new high at 87.64% without incurring any more computation, demonstrating the efficiency of the MoE adapter. The adapter design can normally serve the old data no matter if it is in the phase of adaptation.

The empirical comparison results are shown in Table V. We further find that the order of the pre-trained dataset would affect the decline in magnitude for the continual train method. When the new dataset is much harder than the pre-trained dataset (from SVHN to Food-101), the performance drop is smaller than the situation when the new dataset is easier (from Food-101 to SVHN), the decline percentage is 85.09% vs. 97.38%, respectively. That indicates the degree of catastrophic forgetting is determined by the relative difficulty between the pre-trained dataset and the target dataset. Our adapter design can avoid such issues and maintain high performance when facing distribution shifts.

C. Performance on the Enhanced Task

Scenarios and training settings: To verify the performance of the adapter design in the scenario that a more comprehensive dataset is offered with enhanced performance requirements, we experiment with the two adapters on the CIFAR-10 and CIFAR-100 datasets, where the continual training method is adopted as a comparison.

Specifically, we first pre-train a ViT-Tiny model on the CIFAR-10 dataset which is viewed as the old data. The training hyper-parameters are the same with SVHN and Food-101 datasets, that the optimizer is AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay is set to 0.05. The learning rate is 1.25×10^{-4} while the batch size is 128. The model is trained for 100 epochs with the first 20 epochs as warm-up. Then for the adaption process, the parameters from the ViT-Tiny part are frozen to preserve the knowledge on CIFAR-10, while we train only the adapter part on the CIFAR-100 dataset for 100 epochs with the same hyper-parameter settings. Since the number of categories for two datasets is different, i.e., 10 and 100, the adaption process additionally trains a task-specific head for CIFAR-100. Further, due to the distribution shift between the two datasets is negligible which is shown in Table IV, we assume that the knowledge gained from CIFAR-10 would benefit the learning process of CIFAR-100, in a way that leverages the trained model's knowledge of recognizing colors, shapes, and textures of objects from datasets with similar distributions. To utilize this knowledge, we design a fusion module to fuse the knowledge from the previous and the new dataset, as is discussed in Section IV-D. The fusion mechanism includes four blocks that are deployed at the end of each stage, each block contains an MSA layer (30) and a FFN layer (30). Thus, during the adaptation process, we train a total of eight layers of parameters.

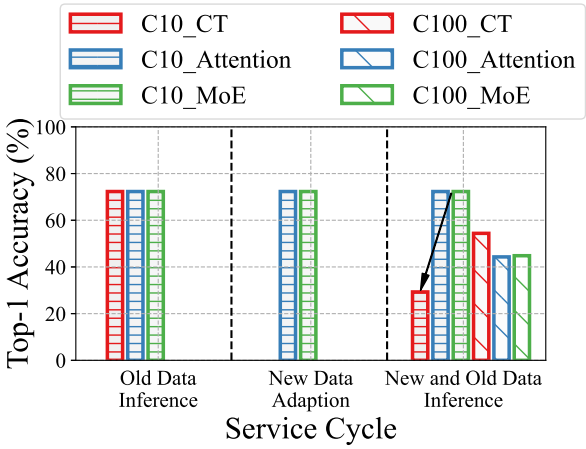


Fig. 7. Performance comparison for MoE (green) and attention-based (blue) adapters. Here, the CIFAR-10 (horizontal lines) is the old dataset while the CIFAR-100 (diagonal lines) is the new. The traditional model continual train (red) strategy is adopted as a comparison. The mean top-1 classification accuracy is reported.

The model continual train strategy is adopted as a comparison with the same training settings as the adaption process for a fair comparison. It simply fine-tunes the full parameters of the ViT-Tiny model and new 100-way classification head on the CIFAR-100 dataset with the same training settings and epochs. All the experiments for continually training and two adapters are performed five times with different random seeds.

Evaluations and discussions: The performance across various stages of the service cycle is shown in Fig. 7, measured by the model's top-1 accuracy on two datasets, where we observe the same phenomenon as the previous experiments. Specifically, for the first phase shown in the left sub-block, all methods start from the pre-trained ViT-Tiny model that performs inference on the CIFAR-10 (old dataset, horizontal lines), yielding an average accuracy of 72.31%. With CIFAR-100 coming in, all three models step into the phase of new data adaption phase in the middle sub-block. For the continual training method, since the model weights are adjusted based on the new dataset, it is no longer capable of handling the old data, which is why the red bar is not displayed. In contrast, because the weights of the ViT-Tiny model remain unchanged, both adapter designs (blue and green bar) can still perform inference on the old data, preserving the previous high accuracy of 72.31%, while at the same time training the adapter weights on the new data. After finishing the adaption, all three models restart to serve both the new and the old data, as shown in the right sub-block in Fig. 7. For the continual training method, though the performance on the new data reaches a promising mean one of 54.39%, the catastrophic forgetting issue downgrades its performance on the old data from 72.31% to 29.29%, which has reduced the original performance by more than half and is not acceptable. However, our method can achieve the advantage of maintaining high performance on both datasets. For the attention-based adapter, it reaches the accuracy on CIFAR-100 of 44.30%, which is almost on par with the pre-trained model with only 2.58 M trainable parameters compared to the 5.52 M of the full ViT-Tiny model.

TABLE VI
COMPARISON RESULTS ON THE CIFAR-10 AND CIFAR-100 DATASETS

Method	New CIFAR-100 (%)	Old CIFAR-10 (%)
ViT-Tiny/16@224px		
Train from the scratch	44.47 ± 0.46	72.31 ± 0.32
Continual train	54.39 ± 0.40 (↑ 9.92)	29.29 ± 1.56 (↓ 43.02)
Adapter		
Attention	44.30 ± 0.41 (↓ 0.17)	72.31 ± 0.32 (-)
MoE	44.81 ± 0.48 (↑ 0.34)	72.31 ± 0.32 (-)

All experiments are independently performed five times with different seeds. The mean and standard variation of Top-1 classification accuracy are reported.

With a more advanced MoE adapter design, the performance on CIFAR-100 reaches 44.81% even surpassing 44.47% of the pre-trained model. This demonstrates that the MoE architecture, combined with the fusion design, provides a powerful distribution learning capability and enables more effective utilization of the knowledge learned from the pre-trained ViT-Tiny. Notice that we made minimal effort to tune the hyper-parameters, simply adhering to a uniform parameter configuration. We believe that applying specialized parameter tuning for the MoE adapter will further enhance its potential learning capability. The detailed empirical comparison results are shown in the Table VI.

D. Ablation Study on Fusion Mechanism

The utilization and efficacy of the fusion module are contingent upon the specific configurations employed. In this section, we conduct a comprehensive ablation study to elucidate its practical applications, systematically examining its merits, limitations, and performance across diverse scenarios. The experimental evaluation is performed on all four benchmark datasets, maintaining consistent training protocols with our prior experiments to ensure comparability and reproducibility.

The comparative analysis presented in Table III reveals significant complexity differences between the SVHN and Food-101 datasets, both in terms of semantic content and task difficulty. Specifically, the SVHN dataset exhibits substantially lower complexity, as evidenced by two key observations: (1) the color channel distributions in SVHN images demonstrate high similarity, whereas Food-101 images exhibit pronounced inter-channel variations; (2) the classification task for SVHN involves only 10 categories, representing a significantly less challenging problem space compared to the 101-way classification required for Food-101.

First, we examine the distribution shift scenario where a ViT-Tiny model, pre-trained on a complex dataset for a challenging task, undergoes adapter training on a simpler dataset with reduced task complexity (e.g., adapting a Food-101 model to the SVHN dataset). As illustrated in Fig. 8, the integration of fusion blocks demonstrates contrasting effects for attention-based and MoE-based adapter architectures. Empirical results reveal that incorporating the fusion module consistently degrades the average top-1 accuracy on the target dataset across both adapter designs. Specifically, the baseline configurations (without fusion) achieve mean accuracies of 87.09% and 87.64%

TABLE VII
ABLATION RESULTS ON FUSION MECHANISM

Method	Accuracy		
	SVHN(%)	Food-101 (%)	CIFAR-100 (%)
ViT-Tiny/16@224px			
Train from the scratch	92.64 ± 0.21	47.48 ± 0.49	44.47± 0.46
Attention-based Adapter			
w/o fusion	87.08 ± 0.57 (↓ 5.56)	38.39 ± 0.19 (↓ 9.09)	38.30 ± 0.36 (↓ 6.17)
w fusion	86.03 ± 1.01 (↓ 6.61)	41.70 ± 0.72 (↓ 5.78)	44.30 ± 0.41 (↓ 0.17)
MoE-based Adapter			
w/o fusion	87.64 ± 0.43 (↓ 5.00)	40.10 ± 0.45 (↓ 7.38)	40.08 ± 0.31 (↓ 4.39)
w fusion	86.57 ± 0.40 (↓ 6.07)	42.53 ± 0.32 (↓ 4.95)	44.81 ± 0.48 (↑ 0.34)

All experiments are independently performed five times with different seeds. The mean and standard variation of Top-1 classification accuracy are reported.

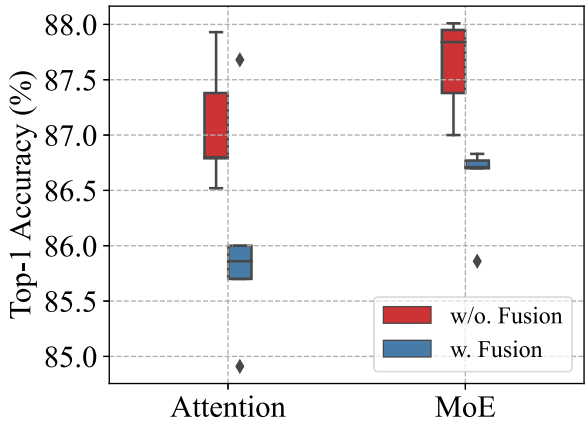


Fig. 8. Ablation results regarding the impact of the fusion module. The MoE and attention-based adapters are trained on the SVHN dataset for five independent runs with various seeds.

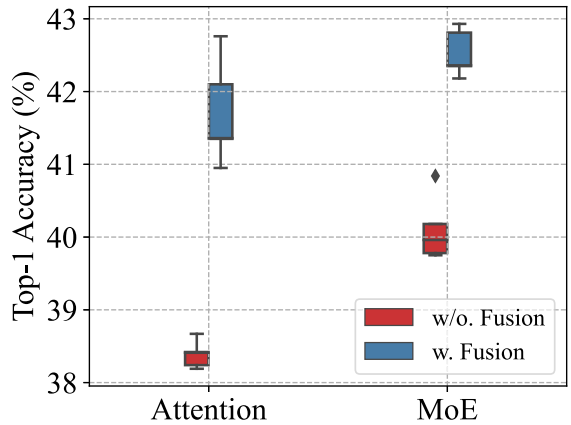


Fig. 9. Ablation results regarding the impact of the fusion module. The MoE and attention-based adapters are trained on the food-101 dataset for five independent runs with various seeds.

for attention-based and MoE-based adapters, respectively (denoted by red boxes). In contrast, the fusion-enabled counterparts exhibit reduced performance, attaining 86.03% and 86.57% for the respective architectures (denoted by blue boxes).

Furthermore, statistical analysis across five independent experimental trials indicates that the accuracy distribution of fusion-enabled models is systematically lower than their non-fusion counterparts. Notably, the upper confidence bound of fusion-enabled models fails to surpass the lower bound of baseline models, a consistent trend observed across both architectural variants. This systematic performance degradation suggests that task-specific knowledge acquired from complex source domains may lack sufficient generality, thereby adversely impacting adaptation to simpler target tasks. These findings collectively demonstrate that fusion modules may provide limited benefits, or even prove detrimental when adapting models to target domains that are substantially simpler than the source domain in terms of either task complexity or dataset scale.

Conversely, when reversing the training paradigm—specifically, pre-training on a simpler dataset followed by adapter fine-tuning on a more complex one—we observe a markedly different performance trend. As evidenced by the experimental results in Fig. 9, the integration of the fusion module yields substantial performance improvements, enhancing accuracy by 3.31% and 2.43% for attention-based and MoE-based adapters, respectively. This empirical evidence

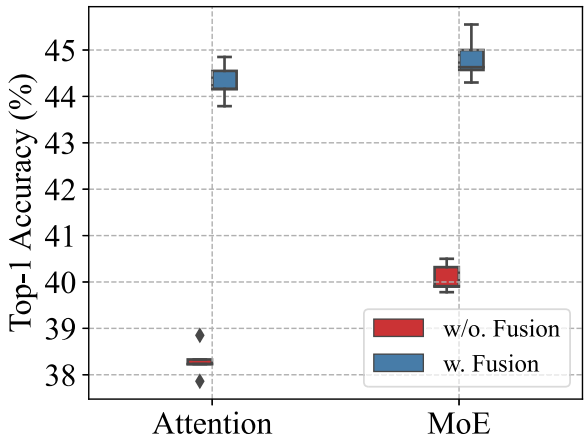


Fig. 10. Ablation results regarding the impact of the fusion module. The MoE and attention-based adapters are trained on the CIFAR-100 dataset for five independent runs with various seeds.

suggests that feature representations and knowledge acquired from simpler tasks possess greater transferability, effectively facilitating model adaptation to more complex downstream tasks. The observed performance gains underscore the potential benefits of leveraging hierarchical feature learning, where foundational knowledge from simpler domains can serve as a robust basis for tackling more challenging problem spaces.

This finding is further substantiated through experiments involving fine-tuning from a CIFAR-10 pre-trained model to CIFAR-100, where the fusion module delivers significant performance gains. Specifically, the accuracy improvements amount to 6.00% and 4.73% for attention-based and MoE-based adapters, respectively. Notably, the MoE adapter achieves a 0.34% accuracy advantage over the fully-trained model, underscoring both the effectiveness of the fusion module and the inherent potential of the MoE architectural paradigm. Comprehensive empirical results from this comparative analysis are systematically presented in Table VII.

Furthermore, our experiments reveal a consistent performance trend: across all configurations—irrespective of the inclusion of the fusion module or the specific dataset employed—the MoE adapter demonstrates superior accuracy compared to its attention-based counterpart. This persistent performance advantage highlights the architectural robustness and generalization ability of the MoE framework, suggesting its intrinsic capability to better capture and leverage task-relevant features across diverse adaptation scenarios.

VI. Conclusion and Future Work

In this paper, we studied continual knowledge adaption for multi-distribution data under constrained computing resources and scarce data. By leveraging the sparse routing mechanism of the Mixture-of-Experts, we have designed a lightweight MoE adapter that enables the model to scale its parameter size for stronger representation without increasing computational cost. The MoE adapter avoids interrupting the trained model and ensures the performance of both new and old distributions. In addition, we designed a knowledge fusion mechanism to exploit the useful information from learned knowledge, which enhances learning and accelerates the convergence of the adaption process. Our insights and design can shed light on making progress in constrained continual learning and benefit the knowledge utilization in mobile edge computing systems. In the future work, we will explore leveraging the strong representation learning capability of MoE to design an efficient model for multimodal data continually understanding.

References

[1] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tut.*, vol. 24, no. 1, pp. 1–30, Firstquarter 2022.

[2] D. Wu, D. Zhang, M. Zhang, R. Zhang, F. Wang, and S. Cui, "ILCAS: Imitation learning-based configuration-adaptive streaming for live video analytics with cross-camera collaboration," *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 6743–6757, Jun. 2024.

[3] T. Brown, B. Mann, and N. Ryder, "Video generation models as world simulators," Tech. Rep., 2024. [Online]. Available: https://openai.com/index/video-generation-models-as-world-simulators/

[4] Z. Yan et al., "Contrastive open-set active learning-based sample selection for image classification," *IEEE Trans. Image Process.*, vol. 33, pp. 5525–5537, 2024.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[6] A. Dosovitskiy et al., "An image is worth 16 × 16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2021.

[7] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289–299, Aug. 2014.

[8] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 10164–10183, Dec. 2024.

[9] D. Qiu, Y. Cheng, K. K. L. Wong, W. Zhang, Z. Yi, and X. Wang, "DBSR: Quadratic conditional diffusion model for blind cardiac MRI super-resolution," *IEEE Trans. Multimedia*, vol. 26, pp. 11358–11371, 2024.

[10] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Surveys Tut.*, vol. 23, no. 2, pp. 1160–1192, Secondquarter 2021.

[11] C. Zhou, J. Gao, M. Li, N. Cheng, X. S. Shen, and W. Zhuang, "Digital-twin-based 3-D map management for edge-assisted device pose tracking in mobile AR," *IEEE Internet Things J.*, vol. 11, no. 10, pp. 17812–17826, May 2024.

[12] Z. Cao et al., "Patching in order: Efficient on-device model fine-tuning for multi-DNN vision applications," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 14484–14501, Dec. 2024.

[13] H. Li, X. Li, Q. Fan, Q. He, X. Wang, and V. C. M. Leung, "Distributed DNN inference with fine-grained model partitioning in mobile edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 9060–9074, Oct. 2024.

[14] D. Wu et al., "NetLLM: Adapting large language models for networking," in *Proc. ACM SIGCOMM Conf.*, 2024, pp. 661–678.

[15] Z. Zhang, B. Guo, W. Sun, Y. Liu, and Z. Yu, "Cross-FCL: Toward a cross-edge federated continual learning framework in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 313–326, Jan. 2024.

[16] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," 2013, *arXiv:1312.6211*.

[17] W. Chen et al., "Lifelong language pretraining with distribution-specialized experts," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 5383–5395.

[18] X. Yu, A. Thomas, I. G. Moreno, L. Gutierrez, and T. Rosing, "Intelligence beyond the edge using hyperdimensional computing," in *Proc. ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2024, pp. 1–13, doi: 10.1109/IPSN61024.2024.00005.

[19] B. Guo, C. Zhou, H. Liu, S. He, J. Chen, and X. S. Shen, "Attention-based vision knowledge adaptation for constrained continual learning," in *Proc. IEEE Glob. Commun. Conf.*, 2024, pp. 2371–2376.

[20] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Representations*, 2022.

[21] Z. Chen et al., "Vision transformer adapter for dense predictions," in *Proc. Int. Conf. Learn. Representations*, 2023.

[22] P. Wu, K. Li, T. Wang, Y. Dong, V. C. Leung, and F. Wang, "FedFMSL: Federated learning of foundation models with sparsely activated LoRA," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 15167–15181, Dec. 2024.

[23] X. Ma, L. Luo, and Q. Zeng, "From one thousand pages of specification to unveiling hidden bugs: Large language model assisted fuzzing of matter IoT devices," in *Proc. USENIX Secur. Symp.*, 2024, pp. 4783–4800.

[24] T. Brown et al., "Language models are few-shot learners," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 1877–1901.

[25] R. K. Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 1022–1035.

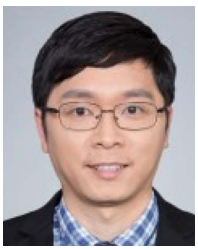
[26] T.-Y. Wu et al., "Class-incremental learning with strong pre-trained models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9591–9600, doi: 10.1109/CVPR52688.2022.00938.

[27] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2019, vol. 97, pp. 2790–2799.

[28] M. Jia et al., "Visual prompt tuning," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 709–727.
[29] J. Kaplan et al., "Scaling laws for neural language models," 2020, *arXiv:2001.08361*.
[30] K. Cho and Y. Bengio, "Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning," 2014, *arXiv:1406.7362*.
[31] N. Shazeer et al., "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," 2017, *arXiv:1701.06538*.
[32] A. Q. Jiang et al., "Mixtral of experts," 2024, *arXiv:2401.04088*.
[33] D. Lepikhin et al., "GShard: Scaling giant models with conditional computation and automatic sharding," in *Proc. Int. Conf. Learn. Representations*, 2021.
[34] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, no. 1, pp. 1–39, Jan. 2022.
[35] Y. Zhou et al., "Brainformers: Trading simplicity for efficiency," in *Proc. Int. Conf. Learn. Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=w5q6tHO1dl1
[36] E. Daxberger et al., "Mobile V-MoEs: Scaling down vision transformers via sparse Mixture-of-Experts," 2023, *arXiv:2309.04354*.
[37] J. Yu et al., "Boosting continual learning of vision-language models via mixture-of-experts adapters," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 23219–23230.
[38] Y. Chen et al., "AccEPT: An acceleration scheme for speeding up edge pipeline-parallel training," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 10938–10951, Dec. 2024.
[39] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9992–10002.
[40] A. Kirillov et al., "Segment anything," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 3992–4003.

Bicheng Guo received the BEng degree from Chongqing University, Chongqing, in 2017, the MSc degree from Wuhan University, Wuhan, and the PhD degree in control science and engineering from Zhejiang University, Hangzhou, in 2025. From 2023 to 2024, he was a visiting student with the University of Waterloo, Waterloo. His research interests include pattern recognition, neural architecture search, and large language modeling.

Conghao Zhou (Member, IEEE) received the BEng degree from Northeastern University, Shenyang, China, the MSc degree from the University of Illinois Chicago, Chicago, IL, USA, and the PhD degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada. He is currently a professor with the School of Telecommunications Engineering, Xidian University, China. His research interests include space-air-ground integrated networks, AI for networking, and immersive communications.



Shibo He (Senior Member, IEEE) received the PhD degree in control science and engineering from Zhejiang University, Hangzhou, in 2012. He was an associate research scientist from March 2014 to May 2014, and a postdoctoral scholar from 2012 to 2014, with Arizona State University, Tempe. From 2010 to 2011, he was a visiting scholar with the University of Waterloo, Waterloo. He is currently a professor with Zhejiang University. His research interests include the IoT, crowdsensing, and Big Data analysis. Dr. He is on the editorial board of several journals including *IEEE Transactions on Network Science and Engineering*. He was the symposium co-chiar for IEEE ICC 2017, finance registration chair of ACM MobiHoc 2015, and the technical program committee co-chair for IEEE ScalCom 2014.

Jiming Chen (Fellow, IEEE) received the PhD degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2005. He is currently a professor with the Department of Control Science and Engineering, deputy director of the State Key Laboratory of Industrial Control Technology, director of Institute of Industrial Process Control, Zhejiang University, and the president of Hangzhou Dianzi University. His research interests include the Internet of Things, networked control, and wireless networks. Dr. Chen is a fellow of the CAA. He was the recipient of the 7th IEEE ComSoc Asia/Pacific Outstanding Paper Award, JSPS Invitation Fellowship, and the *IEEE ComSoc* AP Outstanding Young Researcher Award. He is the general co-chairs for the IEEE RTCSA'19, IEEE Datacom'19, and the IEEE PST'20. He is an IEEE VTS distinguished lecturer. He is on the editorial boards of multiple *IEEE Transactions*.

Xuemin (Sherman) Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a university professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, Canadian Academy of Engineering fellow, Royal Society of Canada fellow, Chinese Academy of Engineering Foreign member, and an International fellow of the Engineering Academy of Japan. Dr. Shen was the recipient of the "West Lake Friendship Award" from Zhejiang Province in 2023, President's Excellence in Research from the University of Waterloo in 2022, Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society (ComSoc), Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013), Excellent Graduate Supervision Award in 2006 from the University of Waterloo, and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He is/was the general chair for the 6G Global Conference'23, and ACM Mobihoc'15, Technical Program Committee chair/co-chair for IEEE Globecom'24, 16 and 07, IEEE Infocom'14, IEEE VTC'10 Fall, and the chair for the IEEE ComSoc Technical Committee on Wireless Communications. He was the former president of the IEEE ComSoc, vice president for Technical & Educational Activities, vice president for Publications, member-at-Large on the Board of Governors, chair of the Distinguished Lecturer Selection Committee, and a member of IEEE Fellow Selection Committee of the ComSoc. He the editor-in-chief of the *IEEE Internet of Things Journal*, *IEEE Network*, and *IET Communications*.