McCTTA: A Memory-Efficient Collaborative Continual Test-Time Adaptation Framework for Edge Devices

Haojie Bai, Aiguo Chen, Yijia Rong, Jiaxin Liu, Kexin Li, and Schahram Dustdar, *Fellow, IEEE*

***Abstract*—The exponential growth of data generated at the network edge has driven a paradigm shift from centralized cloud computing to local edge processing, accelerating the widespread adoption of edge computing across diverse applications. In this context, continual test-time adaptation (CTTA) on edge devices, which enables models to adapt to evolving target domains without access to source data or labeled samples, has become an emerging research focus due to its practical importance in dynamic environments with changing data distributions. However, limited computational and memory resources severely restrict CTTA on edge devices. Moreover, since adaptation relies on noisy unsupervised losses without access to labels, prolonged CTTA can lead to error accumulation. Additionally, the model is susceptible to catastrophic forgetting, an intrinsic challenge in continual adaptation. In this paper, we propose McCTTA, a memory-efficient collaborative continual test-time adaptation framework for edge devices. Specifically, McCTTA incorporates a generative model to synthesize images as a replacement for replay data on the cloud, and a lightweight side network attached to the frozen original network to reduce memory consumption during edge adaptation. We further introduce 2SR (Two-Stage Rehearsal), which decouples active forgetting and knowledge integration into two separate stages to address the plasticity–stability dilemma caused by distributional discrepancies between synthetic and real task data during continual adaptation. Finally, extensive experiments are conducted to evaluate the effectiveness of McCTTA. The results show that, compared with conventional TTA methods, McCTTA achieves superior accuracy and mitigates forgetting while requiring less memory.**

***Index Terms*—Edge computing, edge device, continual test-time adaptation, error accumulation, catastrophic forgetting.**

I. Introduction

THE remarkable advancements in Deep Neural Networks (DNNs) have significantly enhanced the performance of a wide array of artificial intelligence applications, particularly in computer vision, natural language processing, and robotics systems. With the exponential growth of mobile and Internet of Things (IoT) devices, in recent years, enormous amounts of data have been generated at the network edge. Considering factors such as high monetary costs, communication latency, and data privacy concerns, an increasing portion of data processing is being stored and performed locally rather than being transferred to centralized cloud data centers. This paradigm shift has fueled the rapid adoption of edge computing, which supports a broad spectrum of applications—from real-time video analytics and cognitive assistance to autonomous driving technologies and smart city systems.

Existing research has primarily concentrated on optimizing DNN inference for edge devices by employing model compression strategies, including pruning, quantization, and knowledge distillation. While these methods effectively reduce model complexity, they often lead to compromised inference performance or accuracy. To address this challenge, collaborative inference frameworks have gained traction, which strategically partition DNN computations between edge devices and cloud servers to leverage the cloud's abundant computational resources [1], [2], [3]. Concurrently, significant research efforts have explored on-device machine learning, aiming to enable complete model training and deployment directly on resource-constrained edge devices for fully localized execution of deep learning applications.

However, the continual adaptation and updating of models on edge devices remain underexplored, despite their practical significance in dynamic environments with evolving data distributions. A representative scenario is autonomous driving systems, where visual inputs from onboard cameras change continuously due to vehicle movement, varying surroundings, and dynamic lighting conditions. These dynamics inevitably result in domain shift—a divergence between the training and deployment data distributions. Deploying a pretrained model on edge devices without adaptation to the target domain often leads to substantial performance degradation. This necessitates the development of continual adaptation mechanisms deployed at the edge that can incrementally adjust models to accommodate previously unseen data distributions. This scenario accurately reflects a more demanding setting: continual test-time adaptation (CTTA) for edge devices, where models adapt to a continuously evolving target domain without access to source data or labeled target samples. This setting accurately captures real-world deployment scenarios, where models are required to exhibit autonomous

Received 2 September 2025; revised 26 March 2026; accepted 4 April 2026. Date of publication 7 April 2026; date of current version 12 June 2026. This work was supported in part by Sichuan Provincial Science and Technology Achievement Transformation Project under Grant 2024ZHCG0009 and in part by the Natural Science Foundation of Sichuan, China under Grant 2026NSFSC1458. *(Corresponding author: Kexin Li.)*

Haojie Bai, Aiguo Chen, Yijia Rong, Jiaxin Liu, and Kexin Li are with the Laboratory of Intelligent Collaborative Computing, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Trusted Cloud Computing and Big Data Key Laboratory of Sichuan Province, Chengdu 611731, China (e-mail: likx@uestc.edu.cn).

Schahram Dustdar is with Distributed Systems Group, TU Wien, 1040 Vienna, Austria, and also with the Catalan Institution for Research and Advanced Studies, 08010 Barcelona, Spain.

Digital Object Identifier 10.1109/TSC.2026.3681949

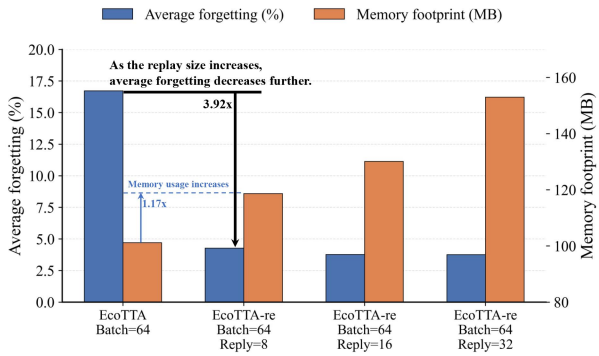


Fig. 1. EcoTTA-re with more replay samples has better performance on average forgetting, but needs more memory.

and incremental adaptation to changing environments, operating under minimal supervision and strictly limited computational resources.

Due to resource constraints, many on-device learning methods [4], [5], [6] introduce a lightweight side network that is attached to a pretrained backbone prior to deployment. During adaptation to new data, only the lightweight network is updated, allowing the device to cache reduced activations during backpropagation, thereby minimizing memory usage, making it more suitable for resource-constrained edge devices. Following this architectural paradigm, EcoTTA [6] extends the setting from single-step adaptation to continual test-time adaptation. Specifically, it introduces a self-distillation-based regularization strategy to mitigate error accumulation and catastrophic forgetting arising from prolonged unsupervised adaptation. In this strategy, the predictions of the lightweight network are aligned with those of the frozen backbone, effectively regularizing the adaptation process using knowledge distilled from the source domain. However, this mechanism only preserves knowledge from the source domain, rather than all previously encountered domains. As a result, it exhibits suboptimal average performance in mitigating catastrophic forgetting across multiple target domains.

In rehearsal-based continual learning methods [7], [8], [9], a memory buffer is maintained by sampling and storing a subset of the dataset after each task. During subsequent tasks, this buffer is replayed to reinforce prior knowledge. To evaluate the effectiveness of EcoTTA in alleviating catastrophic forgetting across sequential target domains, we perform a comprehensive analysis by introducing EcoTTA-re, an enhanced variant wherein the original self-distillation-based regularization term is replaced by a replay buffer strategy. This modification enables direct comparison of knowledge preservation approaches while maintaining the network architecture for continual adaptation. We quantify catastrophic forgetting through average forgetting rate (AFR), computed as the mean accuracy degradation between peak performance during adaptations and the final accuracy of each task. Memory consumption is measured following the protocol used in TinyTL [4]. Our comparative analysis examines: (1) memory usage (MB) and (2) average forgetting (%) between EcoTTA and EcoTTA-re under a training batch size of 64 and with varying replay buffer batch sizes. Experiments are conducted on CIFAR100-C using WideResNet-40. As illustrated in Fig. 1, EcoTTA-re achieves at least a 3.92× improvement in average forgetting, but requires 1.17× more memory compared to the original EcoTTA. Our experiments reveal a clear trade-off: expanding the replay buffer size yields progressively lower average forgetting, but at the cost of increased memory usage. This presents a fundamental limitation for edge deployment, as most edge devices possess severely constrained memory resources. Consequently, the development of memory-optimized continual adaptation frameworks becomes an imperative solution that must effectively adapt to new domains and mitigate catastrophic forgetting.

Building upon the idea that cloud–edge collaborative architectures can effectively compensate for the limitations of edge devices, we propose a novel Memory-efficient Collaborative Continual Test-Time Adaptation (MccTTA) framework. As illustrated in Fig. 2, this framework comprises two synergistic components: (1) a cloud-based generative surrogate that maintains a generative model to replace the traditional replay buffer, and (2) an edge-side adaptation engine featuring a lightweight neural attachment to a frozen backbone using both task-specific data and synthetic samples generated from the cloud. The generative model undergoes data-free training through an adversarial distillation process, where a dynamically updated discriminator provides quality control by enforcing sample relevance constraints. After each adaptation cycle, the updated edge model is uploaded to the cloud and used as the new discriminator to improve its judgment in future generations. This collaborative design enables memory-efficient and privacy-preserving adaptation on edge devices while obviating the need for original training data access. By incorporating synthetic samples generated across all previous tasks, the framework effectively mitigates catastrophic forgetting and ensures broader knowledge retention.

To further address the plasticity–stability dilemma that arises from the distributional gap between real-time data and synthetic samples, we propose a two-stage rehearsal-based adaptation algorithm, termed 2SR. In the first stage (rehearsal stage), the side network is adapted exclusively to new task data without considering forgetting, allowing for full plasticity. In the second stage (true learning stage), the model is fine-tuned using a combination of pseudo-labeled new data (predicted by the model from stage one) and synthetic replay samples, encouraging stable integration of both new and old knowledge. This decoupled learning strategy separates knowledge acquisition from forgetting mitigation, enabling the side network to learn effectively from knowledge-condensed data without being distracted by distributional interference from mixed real and synthetic samples.

Our paper presents the following contributions:

- We propose MccTTA, a novel memory-efficient collaborative continual test-time adaptation framework tailored for edge computing environments. MccTTA enables more comprehensive mitigation of catastrophic forgetting by leveraging a cloud-based generative model while minimizing memory consumption on the edge devices. In addition, it operates without requiring raw data transmission to the cloud, thereby preserving user privacy.
- To address the plasticity–stability dilemma caused by the distributional discrepancy between synthetic and real task data during continual adaptation, we introduce 2SR

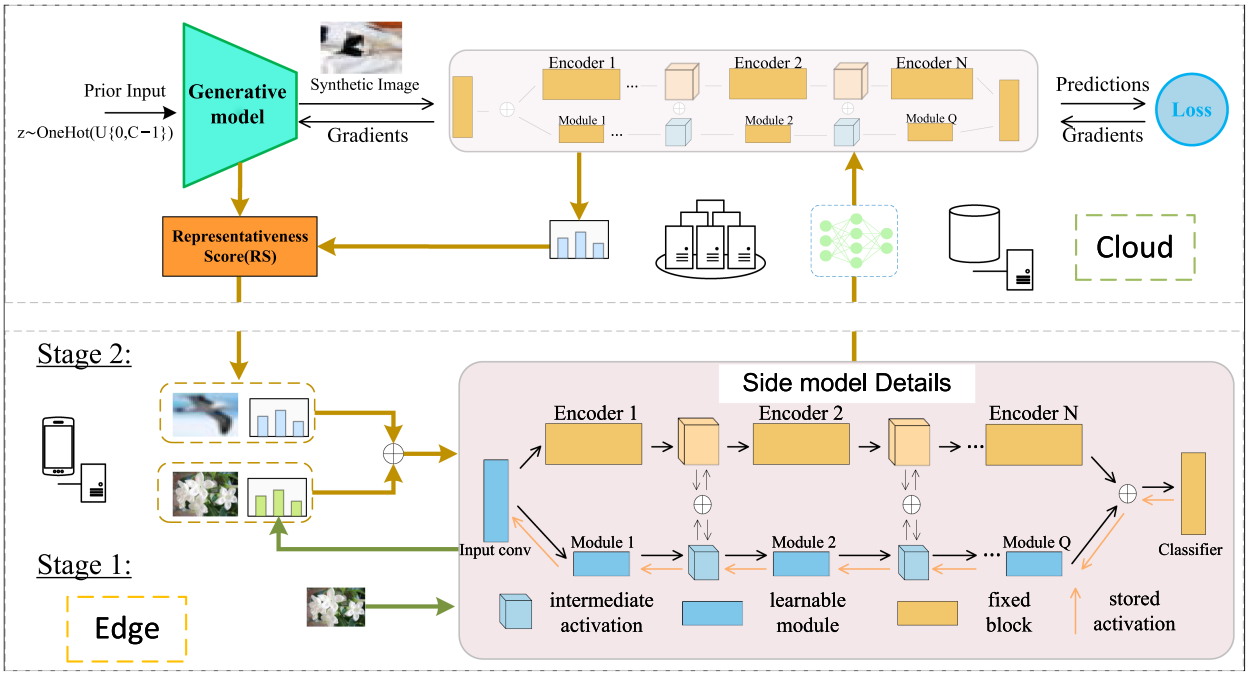


Fig. 2. Our memory-efficient collaborative continual test-time adaptation framework MccTTA.

(Two-Stage Rehearsal), which decouples active forgetting and knowledge integration into two separate stages. This separation enables the side network to adapt more effectively by first focusing on learning new information and then consolidating it alongside previous knowledge.

- We conduct comparative experiments against conventional TTA methods, showing that our MccTTA framework maintains high accuracy while requiring less memory. Moreover, our 2SR algorithm outperforms existing methods in both plasticity and stability. Extensive studies further validate the effectiveness of each component within our method. Importantly, we empirically demonstrate that our approach alleviates catastrophic forgetting and error accumulation, highlighting its suitability for realistic long-term adaptation scenarios.

II. Related Work

A. On-Device Learning

Numerous studies have explored enabling deep neural networks (DNNs) to learn from new data on edge devices. However, this remains a challenging task due to the limited computational and memory resources available on such devices. As a result, a considerable body of work has focused on developing memory-efficient learning strategies. MCUNetV2 [10] considers that the memory bottleneck of on-device learning is caused by uneven memory allocation in convolutional neural networks (CNNs). It finds that the memory usage of the first few blocks is an order of magnitude higher than that of the later layers. To address this, it proposes executing the initial memory-intensive phase in a patch-by-patch manner, while the remaining layers are processed in the standard layer-by-layer fashion. TinyTL [4] argues that the memory bottleneck in on-device training stems from activations rather than parameters. It proposes attaching a lightweight side network to a pre-trained backbone and updating only the side network during training. Building on this approach, [5], [6] adopt lightweight side networks to reduce activation storage and thereby lower memory requirements.

Other approaches aim to reduce activations directly without altering the model structure. [11] proposes asymmetric sparseness: pruning is applied to the activations used in backpropagation, while the forward activations remain dense. [12] proposes freezing and quantizing the bottom block of the model with 8-bit precision, updating only the top block on the target dataset. [13] proposes Sparse Update, which skips gradient computation for secondary layers and subtensors. It is the first solution capable of training CNNs on micro-devices with only 256 KB of SRAM and 1 MB of flash memory.

B. Test-Time Adaptation

Test-time adaptation (TTA) refers to adapting a source model to target data under distributional shift, without access to the original source data. Recent studies focus on entropy minimization and self-training methods to fine-tune source models [6], [9], [14], [15], [16], [17]. In particular, SHOT [14] updates only the feature extractor using information maximization loss and pseudo-labeling. TENT [15] adapts the batch normalization parameters through entropy minimization. Following TENT, [8] further explores the robustness of normalization layers in TTA. All of these methods aim to improve TTA performance under a specific target domain. In such setups, the adapted model is reset to the original source-pretrained model before adapting to a new target domain.

Continual test-time adaptation (CTTA) refers to a setting where the target domain changes continuously over time,

posing significant challenges to conventional TTA methods. The long-term unsupervised adaptation in CTTA makes the model vulnerable to compounding errors, as it relies solely on potentially noisy signals from unlabeled data. Furthermore, the model may excessively focus on new patterns in the target domain, causing it to forget previously acquired knowledge, which becomes particularly problematic when the model encounters test samples that resemble earlier data distributions. While various CTTA approaches attempt to address these issues, each comes with its own limitations. CoTTA [16] uses weight-averaged and augmentation-averaged predictions as pseudo labels, and randomly reverts a subset of neurons to their source-model states. Yet, this method requires processing many augmented versions of the input images simultaneously, which is challenging for resource-limited devices. EcoTTA [6] utilizes a self-distillation regularization term, where the side network is regularized by knowledge distilled from the original model. Nevertheless, it only considers introducing knowledge from the source domain and performs poorly in maintaining knowledge acquired during the adaptation process. F2L [9] separates active forgetting and knowledge integration to circumvent the plasticity–stability trade-off in CTTA. Despite its effectiveness, it requires two separate models and a replay buffer, making it unsuitable for deployment on edge devices.

C. Data-Free Knowledge Distillation

Knowledge distillation facilitates the transfer of information from a well-generalized teacher model to a compact student model. Conventional knowledge distillation methods assume access to the training data of the teacher model. However, in many real-world applications, access to the original data is restricted by privacy regulations or proprietary constraints. To overcome this challenge, data-free knowledge distillation (DFKD) [18] has emerged as a promising alternative. These methods typically employ generative networks to produce synthetic samples, which act as surrogates for the unavailable training set during distillation. They adopt joint adversarial training to align the synthetic distribution with the teacher model's knowledge, effectively guiding student model learning.

The DFKD paradigm has been effectively applied to model compression [18], enabling the transfer of knowledge from a full-precision teacher to a low-precision student using only synthetic data generated by a learned generator. It has also been leveraged in federated learning (FL) [19], [20], where it simultaneously mitigates data heterogeneity and addresses privacy constraints by eliminating the need to share raw data. In the context of continual learning (CL) [21], [22], [23], [24], DFKD provides a memory-efficient alternative to exemplar replay buffers, thereby reducing the risk of catastrophic forgetting. This suggests that DFKD holds promise for these CTTA methods, which require replay buffers.

III. Method

A. Preliminaries of Continual Test-Time Adaptation

Continual test-time adaptation (CTTA) aims to adapt models continually to a sequence of unlabeled target domains with shifting data distributions. Unlike conventional domain adaptation methods, which assume access to labeled source data (X^0, Y^0) for pretraining a source model, CTTA operates under a more challenging and realistic setting where neither source data nor task labels are available during adaptation.

At the t^{th} task, the model receives an unlabeled dataset denoted as:

$$X^t = \left(x^1, x^2 \ldots x^K\right) | x^i \overset{\text{iid}}{\sim} p_t(x), \forall i \tag{1}$$

where p_t is the target domain distribution at time step t, and K is the number of samples in each task, which remains fixed across all tasks. At any time t, we assume a distribution shift, i.e., $p_i \neq p_j$ when $i \neq j$. Moreover, during task t, the model has no access to data from previous tasks X^1, X^2,..., X^{t-1}, due to the memory constraints typical of edge devices. We denote the model adapted to task t as f_{Θ_t}, where Θ_t represents its parameters after adapting to distribution p_t.

B. Memory-Efficient Collaborative CTTA Framework

We assume that cloud resources are sufficient and that the edge device can perform adaptation during its idle time. Based on these assumptions, to address the proposed challenges above, we propose MccTTA, a novel cloud–edge collaborative continual test-time adaptation framework that reduces the reliance on edge device memory and computational capacity. As shown in Fig. 2, on the edge side, the device attaches a lightweight side network to a frozen pretrained backbone and updates only the lightweight network during adaptation, thereby minimizing memory consumption—similar to strategies employed by EcoTTA [6], TinyTL [4] do. On the cloud side, after each task, a generative model is trained using data-free knowledge distillation techniques to capture the knowledge of previously seen domains. This generative model then produces synthetic samples that are sent to the edge for replay, allowing the model to retain knowledge without requiring the edge device to store past data. The advantage of this framework is two-fold: on the one hand, it significantly alleviates the memory burden on edge devices during continual adaptation to new domains, while also achieving better retention of previously learned knowledge across multiple target distributions, with the help of synthetic data generated in the cloud; on the other hand, it safeguards privacy by eliminating the need to transmit raw data to the cloud, as the generative model is trained in a data-free manner.

1) Cloud Side. Data-Free Generative Model Training: The generative model is trained to synthesize images that imitate data from previous tasks, thereby eliminating the need to store real images. Since training such a generative model on the edge device is memory- and computation-intensive, we propose to perform the training on the cloud in a data-free manner, specifically using model inversion image synthesis techniques [21], [22]. Most model inversion approaches synthesize images by directly optimizing them under a discriminative model $f_{\Theta_{t-1}}$ (the model uploaded from the edge device at time $t-1$). Subsequently, the generative model produces synthetic images to serve as replay data for the t^{th} task on the edge device.

We adopt a ConvNet architecture G as the generative model. To approximate a uniform distribution over classes when generating synthetic samples, G takes noise input $z \sim$

$OneHot(U\{0, C-1\})$, where C denotes the total number of classes (constant across tasks), and outputs synthetic samples $\hat{x}$ matching the dimensionality of the original input images. Training the generative model involves multiple objectives to ensure that the synthesized samples both resemble past task data and exhibit sufficient diversity.

Cross entropy Loss: To encourage the synthetic samples $\hat{x}$ to mimic previously seen data, we require that they be correctly classified by the discriminative model $f_{\Theta_{t-1}}$. We achieve this by applying a cross-entropy loss between the predicted class probabilities and the class label corresponding to the original noise input z, which is sampled as a one-hot vector. The assigned label for each synthetic image is defined as $arg\max(z)$. The cross-entropy loss is formulated as:

$$L_{CE}(\hat{x}, z) = CE(arg\max(z), f_{\Theta_{t-1}}(\hat{x})) \tag{2}$$

Diversity Loss: To avoid mode collapse and encourage class diversity among the generated samples in a batch, we introduce a diversity loss by maximizing the information entropy of the mean class prediction over a batch of synthetic images. Specifically, given a batch size gb, the diversity loss is defined as:

$$L_{div}(\hat{x}) = -H\left(\frac{1}{gb}\sum_{i}^{gb} f_{\Theta_{t-1}}(\hat{x}_i)\right) \tag{3}$$

here, $H(p)$ denotes the entropy of a probability vector $p = [p_1, p_2, \ldots, p_C]$, computed as:

$$H(p) = -\sum_{i=1}^{C} p_i \log p_i \tag{4}$$

where C is the number of classes, and $f_{\Theta_{t-1}}(\hat{x}_i)$ denotes the predicted probability for a synthetic image from the discriminative model.

Layer-wise batch normalization (LBN) Loss: Previous research has shown that the distribution of synthetic images may significantly differ from the feature distribution learned by the previous model $f_{\Theta_{t-1}}$, resulting in unrealistic or out-of-distribution samples [21], [22]. To mitigate this issue, we adopt a Layer-wise batch normalization (LBN) loss to align the batch statistics of the synthetic images with those stored in the batch normalization layers of $f_{\Theta_{t-1}}$. Specifically, for each batch normalization layer, we aim to minimize the Kullback–Leibler (KL) divergence between the synthetic and real feature distributions. The LBN loss is defined as:

$$\begin{aligned} L_{\text{BN}}((\mu,\sigma),(\hat{\mu},\hat{\sigma})) &= \frac{1}{L}\sum_{i=1}^{L} \text{KL}\left(\mathcal{N}(\mu_i,\sigma_i^2),\mathcal{N}(\hat{\mu}_i,\hat{\sigma}_i^2)\right) \\ &= \log\frac{\hat{\sigma}_i}{\sigma_i} - \frac{1}{2}\left(1 - \frac{\sigma_i^2 + (\mu_i - \hat{\mu}_i)^2}{\hat{\sigma}_i^2}\right) \end{aligned} \tag{5}$$

L is the number of batch normalization layers, u_i and σ_i are the stored running mean and variance in the discriminator i^{th} batch normalization layer of $f_{\Theta_{t-1}}$ (estimated from real data), while $\hat{u}_i$ and $\hat{\sigma}_i$ are the batch statistics computed from synthetic images during training. This loss encourages the synthetic images to exhibit similar feature distributions as real images in intermediate layers, preventing the discriminator from trivially distinguishing synthetic from real samples based solely on batch normalization statistics.

In our experiments, we adopt standard CTTA benchmarks that include batch normalization layers. The batch statistics maintained by these layers during each adaptation task are uploaded to the cloud. In cases where the model does not contain batch normalization layers, the edge device can still compute feature statistics from selected intermediate layers and transmit them to the cloud. These statistics similarly guide the training of the generative model by providing a reference for aligning synthetic features with previously observed real data distributions.

Images smoothness Loss: We further introduce a prior that natural images tend to be locally smooth in pixel space and incorporate it as a regularization term L_{smooth} to stabilize the optimization process. This term minimizes the distance (such as L2 distance) between the generated image $\hat{x}$ and its smoothed counterpart obtained via Gaussian filtering:

$$L_{Smooth}(\hat{x}) = \|\hat{x} - Smooth(\hat{x})\|_2^2 \tag{6}$$

Finally, the training of the generator G is guided by a composite objective that balances multiple loss terms. We introduce weighting coefficients: w_{CE}, w_{div}, w_{BN} and w_{Smooth} to control the trade-offs among the components:

$$\begin{aligned} \min_{G} \quad & w_{\text{CE}} \cdot \mathcal{L}_{\text{CE}}(\hat{x}, z) + w_{\text{div}} \cdot \mathcal{L}_{\text{div}}(\hat{x}) \\ & + w_{\text{BN}} \cdot \mathcal{L}_{\text{BN}}((\mu,\sigma),(\hat{\mu},\hat{\sigma})) + w_{\text{Smooth}} \cdot \mathcal{L}_{\text{Smooth}}(\hat{x}) \end{aligned} \tag{7}$$

After training the generative model under the guidance of the discriminator $f_{\Theta_{t-1}}$, the generator produces synthetic images, which are then used as replay samples for adapting the edge model to the current task. To select high-quality synthetic images, we employ a filtering mechanism inspired by the Inception Score (IS) [25]. The original IS evaluates the quality of synthetic images using an Inception-V3 model pretrained on the ImageNet dataset. It measures both the confidence and diversity of the predicted class distributions. In contrast, we leverage the previous model $f_{\Theta_{t-1}}$ not only to guide the training of the generative model, but also to assess the quality of the generated samples, since high-quality synthetic images should resemble the distribution of past tasks. Specifically, each generated image $\hat{x}_i$ is passed through $f_{\Theta_{t-1}}$, and its representativeness score (RS) is computed using the Kullback-Leibler divergence:

$$RS(\hat{x}_i) = D_{KL}\left(f_{\Theta_{t-1}}(y|\hat{x}_i) \,\|\, \bar{p}(y)\right) \tag{8}$$

where $f_{\Theta_{t-1}}(y|\hat{x}_i)$ denotes the class probability distribution of each synthetic image as predicted by the discriminator $f_{\Theta_{t-1}}$. The mean prediction across a batch of gb synthetic images is computed as: $\bar{p}(y) = \frac{1}{gb}\sum_{i=1}^{gb} f_{\Theta_{t-1}}(y|\hat{x}_i)$.

The RS of a synthetic image measures the KL divergence between its predicted distribution and the average distribution across the batch. A higher RS suggests that the synthetic image is both confidently classified, with a prediction close to a one-hot vector, and distinctive, as its class distribution differs significantly from the batch average. These properties indicate that the image is of high quality and contributes to diversity, making it an informative candidate for replay.

2) Edge-Side. Continual Domain Adaptation: For the edge-side model, we pay close attention to an important observation highlighted by TinyTL [4]: the primary memory bottleneck during on-device training is not the model parameters, but the

activations. This insight can be illustrated by analyzing the backpropagation process of a linear layer with weight W and bias b. Its forward computation is given by $f_{i+1} = f_i W + b$, and the corresponding backward gradients are:

$$\frac{\partial \mathcal{L}}{\partial f_i} = \frac{\partial \mathcal{L}}{\partial f_{i+1}} \frac{\partial f_{i+1}}{\partial f_i} = \frac{\partial \mathcal{L}}{\partial f_{i+1}} W^T, \quad \frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial f_{i+1}} f_i^T \tag{9}$$

Equation (9) reveals a fundamental difference in computational dependencies between trainable and frozen layers. Specifically, the gradient update for the weight W depends on the intermediate activation f_i, which must be stored during the forward pass if the layer is trainable. In contrast, for a frozen layer, the backpropagation is independent of f_i, depending exclusively on the fixed weights W, thereby significantly reducing memory consumption. Given any pretrained model, we attach a lightweight side network to the original heavyweight network. The original model is kept frozen, while the side network is responsible for adapting to new tasks. Specifically, we train a group of meta-networks that take as input the aggregated activations from their own layers and those of the frozen backbone. Following the design of EcoTTA [6], each meta-network consists of a batch normalization layer followed by a standard convolutional block. This separation of trainable parameters from the original model allows us to only store activations for the backpropagation of the side network, significantly reducing memory consumption in the adaptation process. During the t^{th} adaptation task, the model f_{Θ_t} is required to learn task-specific features from the unlabeled dataset X^t. To effectively utilize these unlabeled samples, we incorporate an entropy minimization loss, which encourages the model to produce high-confidence predictions by minimizing the uncertainty of its output distributions.

$$L_{ent}(x) = -\frac{1}{b} \sum_{i=1}^{b} \sum_{c=1}^{C} {f_{\Theta_t}}^c(y \mid x_i) \log {f_{\Theta_t}}^c(y \mid x_i) \tag{10}$$

where b denotes the batchsize of current images. To mitigate catastrophic forgetting, we incorporate a distillation regularization term using synthetic images:

$$L_{kd}(\hat{x}) = KL\left(f_{\Theta_{t-1}}(y|\hat{x}) \| f_{\Theta_t}(y|\hat{x})\right) \tag{11}$$

Since the synthetic images are generated to approximate replay data, they can be regarded as carriers of knowledge from past tasks. Therefore, the optimization of the current model f_{Θ_t} is regularized by the knowledge distilled from these synthetic samples. We note that the logit of $\hat{x}$ from $f_{\Theta_{t-1}}$ is computed in the cloud and transmitted together with $\hat{x}$ to the edge device. Finally, the total adaptation loss is defined as:

$$L_{total}(x, \hat{x}) = L_{ent}(x) + \lambda L_{kd}(\hat{x}) \tag{12}$$

After completing the t^{th} adaptation task, the updated model f_{Θ_t} is uploaded from the device to the cloud, preparing the system for the next task.

C. Novel Rehearsal-Based Two-Stage Algorithm

Furthermore, synthetic images generated to approximate previously seen data may exhibit a distributional shift relative to the current target domain. This domain discrepancy can introduce conflicting gradient directions between the synthetic and current real data during adaptation, potentially disturbing the learning dynamics. As a result, knowledge distilled from past tasks may interfere with the effective adaptation to the new domain, resulting in reduced model plasticity. It can be controlled by λ in (12) to balance them. However, choosing an appropriate λ is challenging, as it is difficult to achieve optimal performance in both plasticity and stability simultaneously. F2L [9] provides motivation for our design. It separates active forgetting from knowledge integration, aiming to achieve both plasticity and stability without forcing a trade-off between them. Specifically, F2L employs two types of models: a group of short-term models and a long-term model. Each short-term model actively learns new data without considering past knowledge and forgets after each new task. Its responsibility is to assign highly condensed pseudo-labels to the new data before being discarded. The long-term model, on the other hand, continuously learns from both new and replayed data using their pseudo labels. It consolidates the knowledge extracted by the short-term models, avoiding confusion in original new data and replay data, and thus maintains a stable knowledge base.

In the context of CTTA on edge devices, separating active forgetting and knowledge integration within a single model is a more practical design choice. Besides, our MccTTA framework replaces the traditional replay buffer with a generative model to minimize memory usage on edge devices. Based on this insight, we propose a novel rehearsal-based two-stage algorithm named 2SR, which enables efficient and memory-aware continual adaptation.

For each new task t, our method consists of two stages: a rehearsal stage, where the model learns from new data without considering forgetting, and a true learning stage, which follows afterward.

Rehearsal stage: Before the rehearsal stage begins, we save the current model state as $\Theta_{original}$, which will be reused in the true learning stage. In the rehearsal stage, we adopt a maximum information loss strategy, similar to SHOT [14]. This stage involves two loss terms: the entropy minimization loss $L_{ent}(x)$ and the diversity loss $L_{div}(x)$, which are designed to encourage confident and diverse predictions, respectively. Specifically, the diversity loss is defined as:

$$L_{\text{div}}(x) = -H\left(\frac{1}{b} \sum_{i=1}^{b} f_{\Theta_t}(x)\right) \tag{13}$$

$$L_{\text{IM}}(x) = L_{\text{ent}}(x) + L_{\text{div}}(x) \tag{14}$$

We note that $L_{ent}(x)$ is identical to the loss defined in (10), while $L_{div}(x)$ is similar to (3).

After fully learning from the new data, the model assigns pseudo labels $y = [\arg\max_c f^c_{\Theta_t}(c \mid x_i)]_{i=1}^K$, where $f^c_{\Theta_t}(c \mid x_i)$ denotes the predicted class probability of class c for sample x_i. We then transmit this model to the cloud to guide the training of the generative model and to filter synthetic images for the next task.

True learning stage: Once the pseudo-label assignment is completed, the model used in the rehearsal stage has fulfilled its role and is no longer used. During the true learning stage, the model is restored to its original state $\Theta_{original}$, so it can learn from both the new data and the replayed data using their

Algorithm 1: Rehearsal-based Two-Stage Algorithm (2SR) for MccTTA.

Input: Initial source model f_{Θ_0}, number of tasks N
Output: Final adapted model f_{Θ_N}
// Initialization on Server
Upload f_{Θ_0} to server; train initial generator G_0 using f_{Θ_0} as discriminator;
for $t = 1$ **to** N **do**
  Receive current task data x;
  Save initial model state: $\theta_{\text{origin}} = \theta_{t-1}$;
  // Stage 1: Rehearsal Stage
  Perform initial adaptation on x to obtain a temporary state $\tilde{\Theta}_t$ using Eq. (14);
  Assign pseudo-labels y to x using $f_{\tilde{\Theta}_t}$;
  // Stage 2: True Learning Stage
  Restored model to its initial state: $\theta_t \leftarrow \theta_{\text{origin}}$;
  Replay Generation: Generate synthetic samples $\hat{x}$ using G_{t-1}, and assign pseudo-labels $\hat{y}$ using the stabilized discriminator $f_{\Theta_{t-1}}$;
  Filtering: Select high-quality samples via RS filtering;
  Joint Training: Construct the combined dataset using Eq. (15);
  Update model to the final state for the current task by using Eq. (16);
  // Upload for $t+1$ Task
  Upload $f_{\tilde{\Theta}_t}$ to server as discriminator f_{Θ_t} to train generator G_t for next task;

pseudo labels, without being influenced by the unlabeled data seen in the rehearsal stage. The restored model is trained on the concatenation of new data and replayed synthetic data using a standard cross-entropy loss:

$$x_{\text{cat}} = \text{cat}(x, \hat{x}), \quad y_{\text{cat}} = \text{cat}(y, \hat{y}) \tag{15}$$

$$\mathcal{L}_{\text{CE}}(x_{\text{cat}}, y_{\text{cat}}) = \text{CE}(f_{\Theta_t}(x_{\text{cat}}), y_{\text{cat}}) \tag{16}$$

Here, x and y represent the new data and their corresponding pseudo labels, while $\hat{x}$ and $\hat{y}$ denote the replayed synthetic data, filtered by the RS and assigned pseudo labels by the cloud-side discriminator $f_{\Theta_{t-1}}$

To summarize the entire adaptation process, Algorithm 1 presents the 2SR algorithm applied over a continuum of tasks.

IV. Experiment

We conduct extensive experiments to evaluate the performance of MccTTA under a challenging continual test-time adaptation (CTTA) setting in this section.

A. Experimental Setup

Following the setup introduced in EcoTTA [6], we extend beyond the traditional single-domain TTA scenario by evaluating MccTTA on a continual TTA task, where the model must adapt to a sequence of diverse corruptions without resetting. This setting better reflects real-world deployment scenarios, where models continuously face distribution shifts over time.

1) Dataset: In line with prior TTA studies [6], [15], [16], we evaluate MccTTA on two benchmark pairs: CIFAR10, CIFAR100, ImageNet and CIFAR10-C, CIFAR100-C, ImageNet-C. The clean datasets (e.g., CIFAR10) serve as the source domains for pretraining, while the corresponding target domain datasets (e.g., CIFAR10-C) comprise 15 types of common corruptions (e.g., noise, blur, weather, and digital distortions), each with 5 severity levels. It is important to note that we only use the pretrained model obtained from the source domain and do not require access to the source dataset during adaptation.

Following the standard TTA evaluation protocol, we adopt the most challenging setting in our experiments by focusing on severity level 5 and evaluating performance across all 15 corruption types. For each corruption type, CIFAR10-C and CIFAR100-C contain 10,000 RGB images of size 32×32, corresponding to 10 and 100 classes, respectively, while ImageNet-C consists of 50,000 RGB images of size 224×224 across 1,000 classes.

2) Evaluation Metric: In all experiments, we report three key metrics to evaluate the performance of MccTTA: memory consumption, average accuracy, and average forgetting.

Memory Consumption: To assess memory efficiency, we follow the official implementation provided by TinyTL [4], which includes both model parameters and activation storage.

Average Accuracy: We report the model's accuracy on each corruption type after adaptation, and the average accuracy is computed as the mean across all corruption types.

Average Forgetting: We define the forgetting score for each corruption type as the difference between the highest accuracy the model has ever achieved on that type and its final accuracy after adapting to all corruption types. The average forgetting is then calculated as the mean forgetting score across all corruption types.

3) Implementation Details: Following the experimental setup of EcoTTA [6], we employ commonly used pretrained backbones for the edge-side. Specifically, for CIFAR10-C, we adopt two backbone models: WideResNet-28 provided by torchvision and WideResNet-40 sourced from RobustBench [33]. For CIFAR100-C, we employ WideResNet-40 from RobustBench. For ImageNet, we use both the ResNet-50 model from RobustBench and the standard ResNet-50 implementation provided by torchvision. Furthermore, following the design of EcoTTA [6], a lightweight side network is attached to each backbone model.

For the cloud-side generative model, we adopt the architecture utilized in MFCL [34]. Specifically, the generator consists of a linear projection layer followed by a sequence of upsampling modules. These modules integrate interpolation with convolutional layers, Batch Normalization, and LeakyReLU activations to progressively recover the image resolution.

In terms of optimization, the edge-side model uses a standard SGD optimizer with a momentum of 0.9 and a learning rate of 0.08 in the rehearsal stage. In the true learning stage, the same optimizer is employed, but the learning rate is reduced to 0.04. The batch size b is set to 128 for the current test stream,

TABLE I
COMPARISON OF AVERAGE ACCURACY AND MEMORY CONSUMPTION OF MCCTTA AND REPRESENTATIVE TTA METHODS ACROSS MULTIPLE CORRUPTIONS ON CIFAR10-C AND CIFAR100-C

Method	WideResNet-40 (AugMix) CIFAR10-C		WideResNet-28 (Standard) CIFAR10-C		WideResNet-40 (AugMix) CIFAR100-C	
	Avg. Acc. (%) ↑	Mem. (MB) ↓	Avg. Acc. (%) ↑	Mem. (MB) ↓	Avg. Acc. (%) ↑	Mem. (MB) ↓
Source	63.3	24.8	56.5	130.5	30.3	24.8
BN Stats Adapt [26]	84.6	24.8	79.1	130.5	58.9	24.8
Single do. TENT [15]	87.3	423.0	80.8	1453.5	63.3	423.0
Continual TENT	86.7	423.0	80.0	1453.5	61.7	423.0
TTT++ [27]	85.4	878.3	79.7	3161.3	59.0	879.8
SWR&NSP [28]	87.9	900.0	82.8	3489.8	63.4	900.0
NOTE [29]	86.6	423.0	79.8	1453.5	57.2	423.0
EATA [30]	87.0	423.0	81.4	1453.5	62.9	423.0
CoTTA [16]	86.0	920.3	83.0	3818.3	61.9	920.3
PALM [31]	86.3	855.4	83.3	2936.4	63.7	855.4
SURGEON [32]	87.8	118.4	83.1	229.1	63.6	118.4
Ours	**88.1**	**215.6**	**83.8**	**208.3**	**64.1**	**215.6**

Fig. 3. Edge devices in the experiments.

and a small batch of 16 synthetic images is sampled from the cloud-side generative model. The cloud-side generative model, on the other hand, is optimized using the Adam optimizer with a learning rate of 0.001. The weights w_{CE}, w_{div}, w_{BN}, and w_{Smooth} are all set to 1, and gb is set to 128, following the MFCL [34].

Regarding hardware configuration, the cloud server is equipped with an NVIDIA 4090 GPU with 24 GB of memory, while the edge device is an NVIDIA Jetson Xavier NX with 8 GB of memory, as illustrated in Fig. 3. Communication between the two devices during the adaptation process is established via a Wi-Fi connection, with data exchange implemented using socket programming.

B. Comparisons With TTA Methods

We conduct a systematic comparison between the proposed MccTTA framework and several representative test-time adaptation (TTA) methods on widely used benchmarks, including CIFAR10-C, CIFAR100-C, and ImageNet-C, across multiple pretrained models. As shown in Table I, MccTTA achieves the best adaptation performance among competing methods while maintaining competitive memory consumption. In terms of average accuracy, our method outperforms all baselines. We regard source-only inference as a baseline, where the pretrained model is directly applied without adaptation. As expected, this setting yields the lowest accuracy, highlighting the necessity of adaptation under distribution shifts. The BN Stats Adapt method [26], which freezes model parameters and updates only Batch Normalization statistics using test data, improves performance over the source-only baseline.

Single-domain TTA methods [15], [27], [28] demonstrate certain improvements but suffer from instability under continual test-time adaptation due to error accumulation. Continual-domain TTA methods [16], [29], [30], [31], [32] are specifically designed to address sequential distribution shifts and demonstrate competitive performance. Nevertheless, MccTTA consistently surpasses these methods and achieves the best overall results. We attribute this advantage to the 2SR design of MccTTA, which allows the model to leverage pseudo-labels generated under the guidance of active forgetting, thereby achieving more effective adaptation.

We further report results on the large-scale ImageNet-C benchmark in Table II. The results demonstrate that MccTTA consistently maintains superior adaptation performance, achieving the highest average accuracy under both AugMix and standard pretrained settings. Regarding memory consumption, our framework achieves a competitive overhead among all TTA methods, as only a lightweight side network is adapted. It is worth noting that, although source-only and BN Stats Adapt require no additional training and therefore consume even less memory, they deliver inferior adaptation performance compared with TTA methods.

To further analyze the stability of continual adaptation, we compare the average accuracy and average forgetting between MccTTA and the representative continual TTA method EcoTTA across CIFAR10-C, CIFAR100-C, and ImageNet-C (Table III). The results show that MccTTA not only achieves higher accuracy but also alleviates catastrophic forgetting more effectively. This demonstrates that MccTTA maintains a favorable balance between plasticity and stability. We attribute this advantage to the decoupling of active forgetting and knowledge integration, which enables the side network to effectively adapt to new domains while persistently retaining knowledge from past tasks.

C. Ablation Studies

1) Ablation Studies of Data-Free Generative Model Training: To better understand the role of each component in our data-free generative model training method, we carry out an ablation

TABLE II
COMPARISON OF AVERAGE ACCURACY AND MEMORY CONSUMPTION OF MCCTTA AND REPRESENTATIVE TTA METHODS ACROSS MULTIPLE CORRUPTIONS ON IMAGENET-C

Method	ResNet-50 (AugMix) Avg. Acc. (%) ↑	ResNet-50 (Standard) Avg. Acc. (%) ↑	(MB) Total Mem. ↓
Source	25.6	17.7	91
BN Stats Adapt [26]	42.1	27.8	91
Continual TENT [15]	43.9	33.8	1486
EATA [30]	45.1	36.2	1486
CoTTA [16]	45.4	37.4	3132
PALM [31]	44.9	36.1	12645
Ours	**46.3**	**36.8**	**747**

TABLE III
COMPARISON OF AVERAGE ACCURACY AND AVERAGE FORGETTING OF MCCTTA AND ECOTTA ACROSS MULTIPLE CORRUPTIONS ON CIFAR10-C, CIFAR100-C, AND IMAGENET-C

Dataset	Method	WideResNet-40 (AugMix) Avg. Acc. (%) ↑	WideResNet-40 (AugMix) Avg. Forget (%) ↓	WideResNet-28 (Standard) Avg. Acc. (%) ↑	WideResNet-28 (Standard) Avg. Forget (%) ↓
CIFAR10-C	EcoTTA	87.72	8.66	83.44	18.81
	Ours	88.10	6.22	83.80	17.77
CIFAR100-C	EcoTTA [6]	63.63	16.52	-	-
	Ours	64.10	13.61	-	-

Dataset	Method	ResNet50 (AugMix) Avg. Acc. (%) ↑	ResNet50 (AugMix) Avg. Forget (%) ↓	ResNet-50 (Standard) Avg. Acc. (%) ↑	ResNet-50 (Standard) Avg. Forget (%) ↓
ImageNet-C	EcoTTA	45.60	24.58	36.60	20.54
	Ours	46.30	20.41	36.81	17.12

TABLE IV
ABLATION STUDY OF THE 2SR ALGORITHM ON CIFAR10-C AND CIFAR100-C

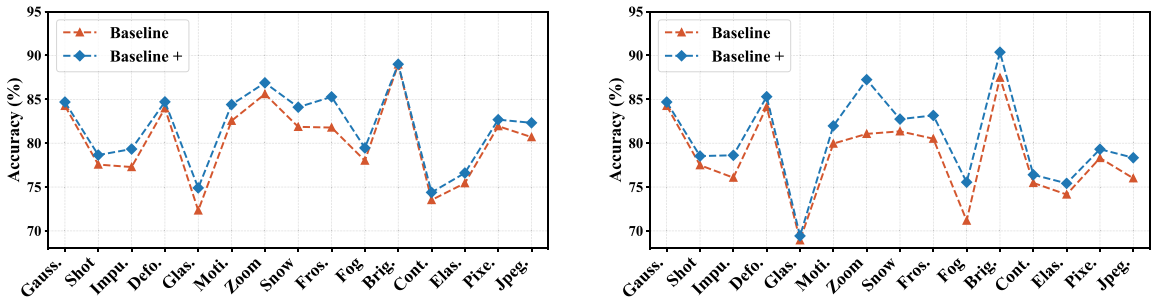
Method	WideResNet-40 (AugMix) CIFAR10-C Avg. Acc. (%) ↑	WideResNet-40 (AugMix) CIFAR10-C Avg. Forget (%) ↓	WideResNet-28 (Standard) CIFAR10-C Avg. Acc. (%) ↑	WideResNet-28 (Standard) CIFAR10-C Avg. Forget (%) ↓	WideResNet-40 (AugMix) CIFAR100-C Avg. Acc. (%) ↑	WideResNet-40 (AugMix) CIFAR100-C Avg. Forget (%) ↓
Source	63.30	0.00	56.50	0.00	30.30	0.00
w Rehearsal	87.37	8.01	76.35	20.22	63.96	18.11
w True learning	55.92	30.25	66.90	26.75	56.35	19.85
Ours (full)	**88.10**	**6.22**	**83.80**	**17.77**	**64.10**	**13.61**

TABLE V
ABLATION STUDY OF THE DATA-FREE GENERATIVE MODEL TRAINING METHOD ON CIFAR10-C WITH WIDERESNET-40

Method	Avg. Acc. (%) ↑	Avg. Forget (%) ↓
Ours (full)	**88.10**	**6.22**
w/o L_{CE}	87.48	7.33
w/o L_{div}	87.38	6.94
w/o L_{BN}	87.33	7.79
w/o L_{Smooth}	87.40	7.34

analysis on CIFAR10-C with WideResNet-40, removing one loss in (7) at a time. Table V presents the impact of each ablated component in terms of task accuracies, average accuracy, and average forgetting, as well as the performance change compared to our complete method. The results indicate that every loss component in (7) is instrumental in generating representative replay samples, with each component substantially enhancing the model's performance in both average accuracy and average forgetting.

2) Ablation Studies of 2SR Algorithm: To assess the contribution of each stage in the 2SR algorithm, we perform an ablation study on CIFAR10-C and CIFAR100-C in Table IV. The introduction of a rehearsal stage to the source model directly utilizes entropy minimization while bypassing knowledge consolidation and replay mechanisms, significantly enhancing the model's performance. However, this improvement comes at the cost of increased average forgetting. Conversely, incorporating the true learning stage into the source model, which uses pseudo-labeling from the source model, results in a decline in performance due to inadequate learning. Our method, which combines both stages, consistently achieves the highest average accuracy and the lowest forgetting rates across all benchmarks. The synergy between these two stages enables 2SR to outperform all baseline configurations, highlighting the critical role of the stage-wise design for robust CTTA.



(a) $\lambda = 0.1$ (b) $\lambda = 0.3$

Fig. 4. Comparison of task accuracies between baseline and *baseline+* when adapting to different types of corruption on CIFAR10-C using WideResNet-40. Subfigure (a) reports results under $\lambda = 0.1$, while subfigure (b) shows results under $\lambda = 0.3$.

D. Evaluation of the Plasticity–Stability Trade-Off

Table III demonstrates the superiority of our method's 2SR algorithm, which achieves strong plasticity while maintaining optimal stability. In the following section, we further analyze the two stages of 2SR and examine their individual contributions to the algorithm's consistently excellent performance.

The rehearsal stage constitutes a crucial element of the 2SR algorithm. By enhancing the accuracy of pseudo-labels, this mechanism substantially improves the plasticity of the adapted model. To validate the effectiveness of rehearsal, we implement (12) as a baseline and extend it with the 2SR algorithm, yielding a variant termed *baseline+*. In this variant, the first stage emphasizes thorough learning without forgetting, and the second stage exploits the pseudo-labels generated in the first stage to calculate the loss L_{ent}. We compare the accuracy of the baseline and *baseline+* when adapting to different types of corruption. The experiments are conducted on CIFAR10-C using WideResNet-40, and the results are presented in Fig. 4(a). It reveals that *baseline+* achieves markedly higher accuracy, highlighting that the integration of pseudo-labels leads to a substantial improvement in plasticity.

Active forgetting facilitates enhanced plasticity through pseudo-labels; however, it alone cannot eliminate the intrinsic conflict between L_{ent} (the plasticity gradient) and L_{kd} (the stability gradient). To examine this phenomenon, we raise the value of λ from 0.1 to 0.3 and repeated the corresponding experiments. The results of $\lambda = 0.3$ are presented in Fig. 4(b). The results demonstrate that, although *baseline+* boosts plasticity, the use of pseudo-labels alone cannot ensure consistently high accuracy in later domains. This finding underscores that improving pseudo-label accuracy alone is not enough.

In our 2SR algorithm, the traditional trade-off between L_{ent} and L_{kd} is replaced by a knowledge integration mechanism. By leveraging the precise pseudo-labels obtained during the first stage for both current and replayed data, this mechanism enables unified, conflict-free training, which is essential for simultaneously maintaining stability and enhancing plasticity. To substantiate this claim, we compare the average accuracy and average forgetting of $baseline+$ and our method on CIFAR10-C using WideResNet-40. The performance comparison presented in Table VI confirms the effectiveness of the knowledge integration mechanism.

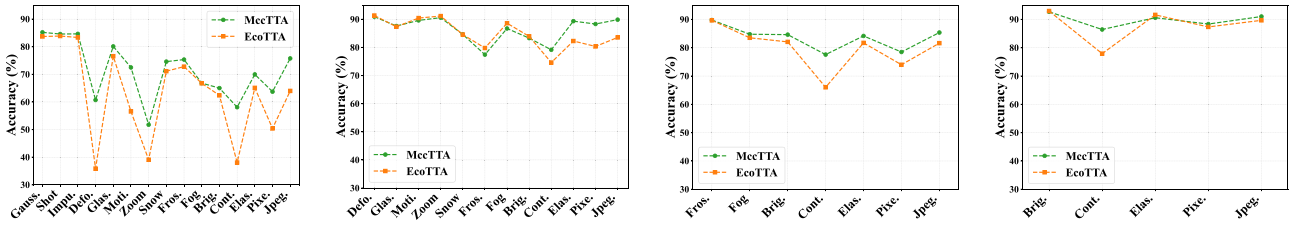
TABLE VI
AVERAGE ACCURACY AND FORGETTING OF BASELINE BASELINE+, AND MCCTTA ON CIFAR10-C WITH WIDERESNET-40

Method		Avg. Acc. (%) ↑	Avg. Forget (%) ↓
baseline	$\lambda = 0.1$	80.38	7.62
baseline+		81.63	6.82
baseline	$\lambda = 0.3$	78.42	7.16
baseline+		80.27	5.94
Ours		**88.10**	**6.22**

E. Evaluation of Catastrophic Forgetting and Error Accumulation Mitigation

1) Catastrophic Forgetting: To evaluate the effectiveness of our method in alleviating catastrophic forgetting, we conducted experiments following a broader evaluation setting. Unlike EcoTTA, which only reports accuracy on clean target data (i.e., test-set of CIFAR10 dataset) after each adaptation step, we assess forgetting across all previously encountered corruptions. Specifically, we measure the accuracy on all corruption target data of CIFAR10-C (i.e., the test set of 15 corruption types) using WideResNet-40. The 15 corruption types in CIFAR10-C can be grouped into four categories: noise, blur, weather, and digital distortions. For analysis, we select one representative corruption type from each category, and the results are presented in Fig. 5. For EcoTTA, its regularization is primarily designed to preserve knowledge of clean target data, leading to a gradual accuracy drop on target corruption data, which reflects the effect of catastrophic forgetting. This trend is particularly evident in Fig. 5(a), where earlier corruptions suffer the most severe degradation over time. In contrast, our method consistently achieves higher accuracy across multiple target domains compared to EcoTTA; more importantly, it maintains stable performance without a pronounced drop during continual adaptation. This demonstrates that our approach effectively mitigates catastrophic forgetting, enabling reliable deployment on diverse target domains rather than being restricted to clean data alone.

2) Error Accumulation: To evaluate the effectiveness of our method in mitigating error accumulation, we simulate long-term adaptation by repeatedly performing adaptation tasks on all 15 corruption types for 100 rounds. As shown in Fig. 6, TENT exhibits a steadily declining accuracy trend over successive



(a) Noise (b) Blur (c) Weather (d) Digital

Fig. 5. Continual adaptation performance on representative corruption types in CIFAR10-C with WideResNet-40. Subfigure (a)– (d) show accuracy over adaptation rounds for noise, blur, weather, and digital distortions, respectively. Compared to EcoTTA, our approach maintains higher and more stable accuracy, effectively mitigating catastrophic forgetting.

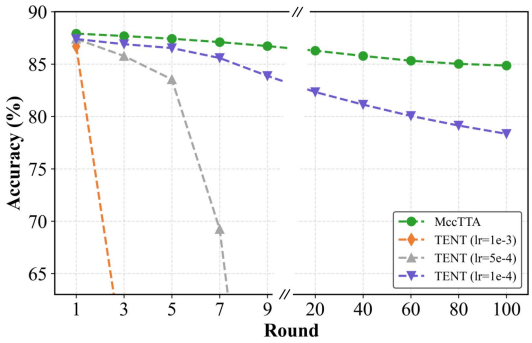


Fig. 6. Evaluation of MccTTA and TENT on Long-Term Continual Adaptation: Accuracy over 100 Rounds on CIFAR10-C with WideResNet-40.

rounds. In particular, when the learning rate is set to 1e-3, its accuracy drops rapidly within only 3 rounds. This indicates that, for conventional TTA methods, the inherent noise in unlabeled data can accumulate over prolonged adaptation, resulting in performance degradation. In contrast, our approach, leveraging accurate pseudo-labels, maintains stable adaptation performance throughout the 100 rounds, effectively preventing error accumulation and ensuring reliable results even under extended continual adaptation.

F. Computational and Communication Overhead Analysis

Since MccTTA adopts a collaborative cloud–edge design, its practical deployment depends not only on adaptation accuracy but also on overall computational and communication efficiency during continual adaptation. Therefore, we further analyze the system overhead introduced by the proposed MccTTA framework.

1) Cloud-Side Generative Model Training Overhead: We report the computational overhead of the cloud-side generative model in terms of execution time and GPU memory consumption. The generator is optimized after each task using the uploaded edge model as a discriminator to synthesize replay samples for subsequent adaptation stages. As shown in Table VII, the resulting overhead remains relatively small compared with the overall adaptation process. This is mainly because the generative model is lightweight and trained using data-free distillation objectives without requiring access to real datasets. These results indicate that the cloud-side computational cost is modest and well-suited for practical deployment, particularly since cloud servers typically possess sufficient computational resources. Importantly, this overhead is fully offloaded from the edge device, ensuring that edge-side adaptation remains lightweight.

TABLE VII
CLOUD-SIDE GENERATIVE MODEL TRAINING OVERHEAD IN TERMS OF TIME AND GPU MEMORY CONSUMPTION

Dataset	WideResNet-40 (AugMix) Time (s)	WideResNet-40 (AugMix) Memory (GB)	WideResNet-28 (Standard) Time (s)	WideResNet-28 (Standard) Memory (GB)
CIFAR10-C	17.24	1.44	213.14	2.25
CIFAR100-C	8.19	1.55	-	-

TABLE VIII
CLOUD–EDGE COMMUNICATION OVERHEAD FOR MODEL UPLOAD AND REPLAY SAMPLE TRANSMISSION

Backbone	Metric Type	Size (MB)	Latency (s)
WideResNet-40 (AugMix)	Model Upload	9.13	0.730
	Replay Sample	0.1875	0.015
WideResNet-28 (Standard)	Model Upload	152.55	12.204
	Replay Sample	0.1875	0.015

2) Cloud–Edge Communication Overhead: Table VIII presents the communication overhead between the edge device and the cloud server in MccTTA, including both model uploading for generator training and replay sample transmission for continual adaptation. The results show that replay sample transmission introduces negligible communication cost, where the size of 16 replay samples in each batch is only 0.1875 MB, resulting in a latency of approximately 0.015 seconds. In contrast, the model upload cost depends on the backbone size. Nevertheless, model uploading is performed only once after each task and therefore does not affect per-batch adaptation latency, making the overall communication overhead acceptable in realistic cloud–edge collaborative scenarios. Overall, these results validate that MccTTA achieves memory-efficient knowledge replay without introducing a significant communication burden.

3) End-to-End Adaptation Latency: To further evaluate practical efficiency, Table IX compares the end-to-end latency of MccTTA with representative TTA methods, including PALM and SURGEON. The reported latency covers the complete adaptation process across all tasks. The Source setting denotes the inference time without any adaptation, including data loading and

TABLE IX
END-TO-END LATENCY (S) OF MCCTTA AND REPRESENTATIVE TTA METHODS. WRN-40 DENOTES WIDERESNET-40 (AUGMIX) AND WRN-28 DENOTES WIDERESNET-28 (STANDARD).

Dataset	Backbone	Source	PALM [31]	SURGEON [32]	Ours
CIFAR10-C	WRN-40	46.15	1318.81	3476.37	421.69
	WRN-28	283.81	3083.71	3748.37	1452.77
CIFAR100-C	WRN-40	45.92	1303.87	3206.02	288.75

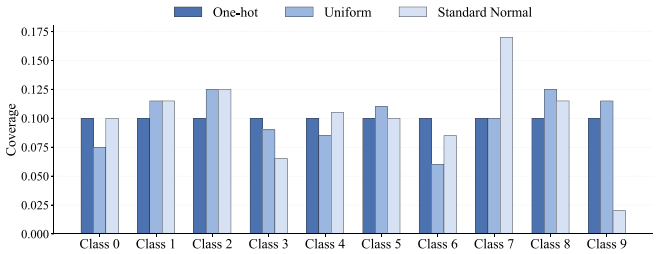


Fig. 7. Class coverage of generated samples under one-hot, uniform, and standard normal noise priors.

forwarding overhead. Compared with PALM, which requires updating heavy model layers and thus incurs substantial adaptation time, MccTTA only updates a lightweight side network, resulting in significantly lower latency. For SURGEON, it requires an additional forward-backward process before adapting each data batch to calculate the weight gradient importance of each layer, which is used to dynamically adjust layer-specific activation pruning ratios, leading to higher overall end-to-end adaptation latency. Compared with SURGEON, our method introduces acceptable additional overhead due to cloud collaboration and replay integration, while achieving improved stability as demonstrated in previous experiments.

These results demonstrate that the proposed two-stage rehearsal strategy introduces only limited computational overhead while providing more effective continual adaptation behavior.

G. Evaluation of Noise Prior and Sampling

1) Evaluation of Noise Prior: Maintaining informative replay samples is essential for preserving previously learned knowledge. To assess the impact of the noise prior to the cloud-side generator, we train with inputs sampled from three distributions: one-hot, uniform, and standard normal, and evaluate the diversity of synthesized data. Specifically, after data-free training, we draw 200 samples and compute class coverage, defined as the proportion of generated samples belonging to each class (e.g., the fraction of samples assigned to class i among 200 generated samples). As shown in Fig. 7 , training with a one-hot noise prior yields the most balanced class coverage among the alternatives. We also examine the impact of the noise prior through the RS metric, adapted from the Inception Score by substituting the Inception-V3 network with an adapted model $f_{\Theta_{t-1}}$ as the discriminator. The RS values for each noise prior are summarized in Table XI. The results corroborate that the one-hot noise prior effectively guides the generator to produce synthetic samples that are both diverse and representative.

2) Evaluation of Sampling Method: The selection of synthetic samples also plays a key role in retaining the previously learned knowledge, while representative and diverse samples support both stability and plasticity. To validate the effectiveness of the RS-based sampling strategy proposed in our method, we evaluate its performance in terms of average accuracy and average forgetting on CIFAR10-C, using WideResNet-40 and WideResNet-28. The method is compared against established sampling approaches, including Random, Entropy-based, and Confidence-based sampling. As shown in Table X, RS-based sampling achieves superior performance on both metrics, demonstrating that it selects highly representative samples that effectively encode prior task knowledge.

H. Robustness of Key Hyper-Parameter

To evaluate the robustness of MccTTA with respect to key hyperparameters, we conduct sensitivity analyses on two important factors in the proposed framework, namely the learning rate in the true learning stage and the number of replay samples used for rehearsal. The evaluation is performed in terms of average accuracy and average forgetting across all tasks.

1) Learning Rate in the True Learning Stage: The learning rate used in the true learning stage plays an important role in MccTTA, as this stage performs the actual model adaptation through real and synthesized replay samples, enabling the model to consolidate previously acquired knowledge while adapting to new distributions. Table XII presents the sensitivity analysis with respect to this learning rate. As observed from the results, MccTTA maintains stable performance within a reasonable range of learning rates, demonstrating robustness to this hyperparameter, while performance degradation is only observed under an excessive learning rate (e.g., 0.4).

2) Number of Replay Samples: Table XIII shows the sensitivity analysis with respect to the number of replay samples. Increasing the number of replay samples generally improves forgetting mitigation, as more replay information helps preserve previously acquired knowledge during continual adaptation. The best performance is achieved at 16, which provides the highest average accuracy and the lowest average forgetting. Overall, the results suggest that MccTTA is not overly sensitive to the replay sample number and achieves stable performance across a moderate range of values.

I. Visual Comparison of Real Data Versus Synthetic Replay Data

In our MccTTA framework, the cloud-side generator is trained subsequent to the completion of adaptation on a target domain. Fig. 8 presents a comparison between real images randomly sampled from CIFAR-10 and synthetic replay samples. Images in the same column belong to the same category. It can be observed that the synthetic images successfully capture the same class-specific semantic information as the real images without replicating any specific real samples, thereby ensuring privacy preservation. Consequently, these synthetic samples can effectively aid in mitigating catastrophic forgetting.

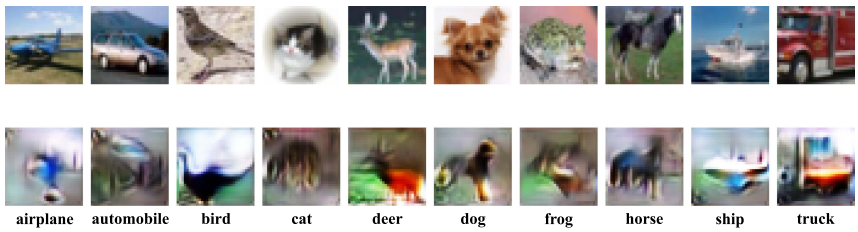


Fig. 8. Real data vs synthetic data generated by the generative model for the CIFAR-10 dataset.

TABLE X
COMPARISON OF AVERAGE ACCURACY AND AVERAGE FORGETTING OF RS-BASED SAMPLING AND OTHER SAMPLING METHOD ON CIFAR10-C WITH WIDERESNET-40

Sampling Method	WideResNet-40 (AugMix)		WideResNet-28 (Standard)	
	Avg. Acc. (%) ↑	Avg. Forget (%) ↓	Avg. Acc. (%) ↑	Avg. Forget (%) ↓
Random sampling	87.51	8.00	83.49	22.39
Entropy-based sampling	87.57	7.84	82.96	19.89
Confidence-based sampling	87.73	6.80	83.16	21.14
RS-based sampling	**87.56**	**5.59**	**83.80**	**17.77**

TABLE XI
RS OF GENERATED SAMPLES UNDER ONE-HOT, UNIFORM, AND STANDARD NORMAL NOISE PRIORS

Noise prior	RS value ↑
Uniform	4.2913 ± 0.18
Standard normal	4.2027 ± 0.12
One-hot	**9.9999 ± 0.02**

TABLE XII
SENSITIVITY ANALYSIS OF THE LEARNING RATE OF THE TRUE LEARNING STAGE

Learning Rate	Avg. Acc. (%) ↑	Avg. Forget (%) ↓
0.4	84.02	10.99
0.1	86.64	7.87
0.04	88.10	6.22
0.01	87.03	7.00
0.004	86.70	7.05

TABLE XIII
SENSITIVITY ANALYSIS OF THE NUMBER OF REPLY SAMPLES

Replay Samples	Avg. Acc. (%) ↑	Avg. Forget (%) ↓
1	87.96	8.40
8	87.77	7.40
16	88.10	6.22
24	86.92	6.82

V. CONCLUSION

In this paper, we address a practical yet underexplored problem: the continual adaptation and updating of models on edge devices, also known as continual test-time adaptation (CTTA). To address this, we propose MccTTA, a memory-efficient collaborative continual test-time adaptation framework for edge devices, enabling edge models to continuously adapt to evolving target domains while maintaining low memory consumption. During adaptation, a generative model is trained on the cloud in a data-free manner to replace the traditional replay buffer. By leveraging this generative model, MccTTA more effectively mitigates catastrophic forgetting. For edge models, we further employ a memory-efficient architecture in which a lightweight side network, attached to a frozen backbone, adapts using task-specific and cloud-generated synthetic data, reducing memory consumption while preserving knowledge from both past and current tasks. To address the plasticity–stability dilemma caused by distributional discrepancies between synthetic and real task data during continual adaptation, we introduce 2SR (Two-Stage Rehearsal). This method decouples active forgetting and knowledge integration into two separate stages, allowing the side network to first focus on acquiring new knowledge and then consolidate it alongside previously learned information. We expect that our contributions will facilitate further research in this area and make continual model updating and adaptation on edge devices practical.

REFERENCES

[1] M. Li, Y. Li, Y. Tian, L. Jiang, and Q. Xu, "AppealNet: An efficient and highly-accurate edge/cloud collaborative architecture for DNN inference," in *Proc. 58th ACM/IEEE Des. Automat. Conf.*, 2021, pp. 409–414.

[2] C. Luo, J. Chen, X. Feng, J. Zhang, and J. Li, "BSL: Sustainable collaborative inference in intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15995–16005, Dec. 2023.

[3] W. Jiang, H. Han, D. Feng, L. Qian, Q. Wang, and X.-G. Xia, "Energy-efficient and accuracy-aware DNN inference with IoT device-edge collaboration," *IEEE Trans. Serv. Comput.*, vol. 18, no. 2, pp. 784–797, Mar./Apr. 2025.

[4] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce memory, not parameters for efficient on-device learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 11285–11297.

[5] L. Yang, A. S. Rakin, and D. Fan, "Rep-Net: Efficient on-device learning via feature reprogramming," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12277–12286.

[6] J. Song, J. Lee, I. S. Kweon, and S. Choi, "EcoTTA: Memory-efficient continual test-time adaptation via self-distilled regularization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 11920–11929.

[7] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: A strong, simple baseline," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 15920–15930.

[8] L. Kumari, S. Wang, T. Zhou, and J. A. Bilmes, "Retrospective adversarial replay for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 28530–28544, 2022.

[9] M. A. Hassan and C.-G. Lee, "Forget to learn (F2L): Circumventing plasticity–stability trade-off in continuous unsupervised domain adaptation," *Pattern Recognit.*, vol. 159, 2025, Art no. 111139.

[10] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. Han, "Memory-efficient patch-based inference for tiny deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 2346–2358.

[11] Z. Jiang, X. Chen, X. Huang, X. Du, D. Zhou, and Z. Wang, "Back razor: Memory-efficient transfer learning by self-sparsified backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 29248–29261, 2022.

[12] H.-Y. Chiang, N. Frumkin, F. Liang, and D. Marculescu, " MobilETL: On-device transfer learning with inverted residual blocks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 6, 2023, pp. 7166–7174.

[13] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256 kb memory," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 22941–22954.

[14] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation," in*Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6028–6039.

[15] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=uXl3bZLkr3c

[16] Q. Wang, O. Fink, L. Van Gool, and D. Dai, "Continual test-time domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 7201–7211.

[17] M. Ramazanova, A. Pardo, B. Ghanem, and M. Alfarra, "Test-time adaptation for combating missing modalities in egocentric videos," in *Proc. 13th Int. Conf. Learn. Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=1L52bHEL5d

[18] M. Haroush, I. Hubara, E. Hoffer, and D. Soudry, "The knowledge within: Methods for data-free model compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8494–8502.

[19] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 12878–12889.

[20] M.-T. Tran, T. Le, X.-M. Le, M. Harandi, and D. Phung, "Text-enhanced data-free approach for federated class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 23870–23880.

[21] H. Yin et al., "Dreaming to distill: Data-free knowledge transfer via deepinversion," in *Proc. IEEE/CVF Conf. Comput. Vis. pattern Recognit.*, 2020, pp. 8715–8724.

[22] J. Smith, Y.-C. Hsu, J. Balloch, Y. Shen, H. Jin, and Z. Kira, "Always be dreaming: A new approach for data-free class-incremental learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9374–9384.

[23] Q. Gao, C. Zhao, B. Ghanem, and J. Zhang, "R-DFCIL: Relation-guided representation learning for data-free class incremental learning," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 423–439.

[24] G. Rypeść, S. Cygert, T. Trzciński, and B. Twardowski, "Task-recency bias strikes back: Adapting covariances in exemplar-free class incremental learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 37, pp. 63268–63289.

[25] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. 2016, vol. 29, pp. 2226–2234.

[26] Z. Nado, S. Padhy, D. Sculley, A. D'Amour, B. Lakshminarayanan, and J. Snoek, "Evaluating prediction-time batch normalization for robustness under covariate shift," 2020, *arXiv:2006.10963*.

[27] Y. Liu, P. Kothari, B. Van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi, "TTT: When does self-supervised test-time training fail or thrive?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 21808–21820.

[28] S. Choi, S. Yang, S. Choi, and S. Yun, "Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 440–458.

[29] T. Gong, J. Jeong, T. Kim, Y. Kim, J. Shin, and S.-J. Lee, "Robust continual test-time adaptation: Instance-aware BN and prediction-balanced memory," 2022, *arXiv:2208.05117*.

[30] S. Niu et al., "Efficient test-time model adaptation without forgetting," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2022, pp. 16888–16905.

[31] S. K. Maharana, B. Zhang, and Y. Guo, "Palm: Pushing adaptive learning rate mechanisms for continual test-time adaptation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 39, no. 18, 2025, pp. 19378–19386.

[32] K. Ma et al., "Surgeon: Memory-adaptive fully test-time adaptation via dynamic activation sparsity," in *Proc. Comput. Vis. Pattern Recognit. Conf.*, 2025, pp. 30514–30523.

[33] F. Croce et al., "RobustBench: A standardized adversarial robustness benchmark," in *Proc. Neural Inform. Process. Syst. Track Datasets Benchmarks*, J. Vanschoren and S. Yeung, Eds. 2021, vol. 1.

[34] H. Chen et al., "Data-free learning of student networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3514–3522.

Haojie Bai received the BE degree in 2024 from the University of Electronic Science and Technology of China, Chengdu, China, where he is currently working toward the PhD degree with the Laboratory of Intelligent Collaborative Computing. His research interests include edge computing and computer vision.

Aiguo Chen received the PhD degree in signal and information processing from the Beijing University of Posts and Telecommunications, China, in 2009. From 2013 to 2014, he was a visiting scholar with Arizona State University, USA. He is currently a professor with the University of Electronic Science and Technology of China. His research interests include cloud and edge computing, blockchain, Big Data analytics, and privacy.

Yijia Rong received the BE degree in 2024 from the University of Electronic Science and Technology of China, Chengdu, China. He is currently working toward the MSc degree with the Laboratory of Intelligent Collaborative Computing, University of Electronic Science and Technology of China. His research interests include collaborative inference and edge computing.

Jiaxin Liu received the BE degree in 2024 from the University of Electronic Science and Technology of China, Chengdu, China, where he is currently working toward the PhD degree with the Laboratory of Intelligent Collaborative Computing. His research interests include computer vision and edge computing.

Kexin Li received the PhD degree in 2023 from Northeastern University, Shenyang, China. She is currently a lecturer with the University of Electronic Science and Technology of China. Her research interests include software-defined networking, edge computing, and machine learning.

Schahram Dustdar (Fellow, IEEE) is currently a full professor of computer science (informatics) with a focus on Internet Technologies heading the Distributed Systems Group, TU Wien. He was the recipient of the ACM Distinguished Scientist Award in 2009 and the IBM Faculty Award in 2012. He is also an associate editor for *IEEE Transactions on Services Computing*, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*, and on the editorial board of *IEEE Internet Computing*. He is the editor-in-chief of *Computing* (an SCI-ranked journal of Springer). Since Dec. 2016, he has been the chair with Informatics Section, Academia Europaea. Since 2016, he has been a member of the IEEE Conference Activities Committee (CAC), the Section Committee of Informatics of the Academia Europaea since 2015, a member of the Academia Europaea: The Academy of Europe, Informatics Section since 2013.