



FoCTTA: Low-Memory Continual Test-Time Adaptation with Focus

Youbing Hu¹, Yun Cheng², Zimu Zhou³, Anqi Lu¹, Zhiqiang Cao¹, Zhijun Li¹

¹Faculty of Computing, Harbin Institute of Technology

²Swiss Data Science Center, Zurich, Switzerland

³City University of Hong Kong

{youbing, luanqi, zhiqiang_cao}@stu.hit.edu.cn, yun.cheng@spsc.ethz.ch

zimuzhou@cityu.edu.hk, lizhijun_os@hit.edu.cn

Abstract—Continual adaptation to domain shifts at test time (CTTA) is crucial for enhancing the intelligence of deep learning enabled IoT applications. However, prevailing CTTA methods, which typically update all batch normalization (BN) layers, exhibit two memory inefficiencies. First, the reliance on BN layers for adaptation necessitates large batch sizes, leading to high memory usage. Second, updating all BN layers requires storing the activations of all BN layers for backpropagation, exacerbating the memory demand. Both factors lead to substantial memory costs, making existing solutions impractical for IoT devices. In this paper, we present FoCTTA, a low-memory CTTA strategy. The key is to automatically identify and adapt a few drift-sensitive representation layers, rather than blindly update all BN layers. The shift from BN to representation layers eliminates the need for large batch sizes. Also, by updating adaptation-critical layers only, FoCTTA avoids storing excessive activations. This focused adaptation approach ensures that FoCTTA is not only memory-efficient but also maintains effective adaptation. Evaluations show that FoCTTA improves the adaptation accuracy over the state-of-the-arts by 4.5%, 4.9%, and 14.8% on CIFAR10-C, CIFAR100-C, and ImageNet-C under the same memory constraints. Across various batch sizes, FoCTTA reduces the memory usage by 3-fold on average, while improving the accuracy by 8.1%, 3.6%, and 0.2%, respectively, on the three datasets.

Index Terms—continual test-time adaptation, adaptation-critical layers, distributional shift

I. INTRODUCTION

After deploying deep neural networks (DNNs) to IoT devices for real-world applications, the model accuracy often severely degrades when there is a notable shift between the source and the target domain [1]. In such cases, the pre-trained model should adapt to the target data distribution at test time without access to the target data labels, known as *Test-Time Adaptation* (TTA) [2]. As the target domain may evolve over time, continual test-time adaptation (CTTA) [3] is necessary, and has attracted increasing research interest [4].

As a continual unsupervised domain adaptation paradigm for IoT applications, CTTA must consider not only *effectiveness* (e.g., accuracy), but also *efficiency* (e.g., computation, memory, latency) when it comes to *what*, *when*, and *how* to adapt [5]–[7]. Due to the unsupervised nature, CTTA solutions [2] often minimize an entropy-based loss by updating certain

model parameters via standard mini-batch gradient descent. The adaption can be performed upon all test-time samples or selectively to reliable samples. For example, EATA [8] selects reliable samples through entropy filtering. Most CTTA methods adopt a *partial training* strategy, which only updates a small set of model parameters at test-time, for both effective adaptation and computation efficiency.

Although mainstream CTTA proposals [8] opt for *computation* efficiency, it does not easily translate into *memory* efficiency, which is crucial for IoT applications. Specifically, they update the affine parameters of all batch normalization (BN) layers to adapt to the target domain. Even though a small set of parameters are updated, this strategy demands substantial memory to function properly. (i) A large batch size is necessary to accurately estimate the current batch statistics. (ii) Updating the BN layers still involves storing activations for backpropagation, whose memory cost increases with the number of BN layers updated. Both factors lead to considerable memory overhead to maintain high adaptation accuracy. Fig. 1(a) shows the memory usage of TENT [2] with the increase of batch size. In Fig. 1(b), the BN-based method is sensitive to the batch size. In the target domain, it can only retain accuracy higher than the original pre-trained model (36.5MB) by consuming $8\times$ memory.

A few pioneer CTTA schemes have been proposed to improve memory efficiency [9], [10]. Compared with standard methods that update all affine parameters in all BN layers [2], [8], these solutions either update shift-sensitive channels in the BN layers [9], [11], or completely freeze the original model and only update extra side-way prompt modules [10]. These schemes improve memory efficiency by reducing the number of adapted layers (channels), which saves the storage of activations during backpropagation. However, they still rely on large batch sizes to boost adaptation accuracy, making them still sub-optimal for low-memory adaptation.

In this paper, we aim at memory-efficient CTTA that functions with small batch sizes and low activation storage during backpropagation. Specifically, we choose to update the representation layers rather than the BN layers to be more resilient to batch sizes. In addition, observing the importance of representation layers varies for adaptation, we only update

 Corresponding Author

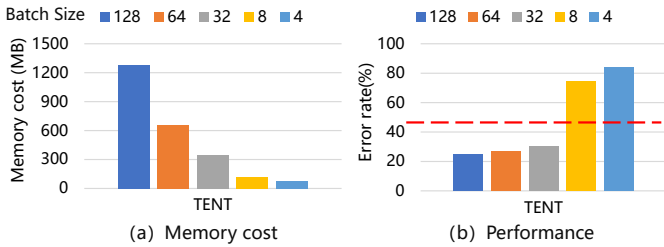


Fig. 1. Evaluate memory cost and performance across various batch sizes.

the top-K critical representation layers to reduce the storage of activations. By *focusing on adaptation-critical representation layers*, we not only ensure memory efficiency but also achieve high adaptation accuracy. This is implemented via an offline warm-up training stage after pre-training but before testing, where simulated unseen distribution shifts are used to identify shift-sensitive representation layers via a simple gradient-based importance metric. At test time, only the top-K critical representation layers are updated for adaptation. Experimental results show that our approach achieves state-of-the-art performance on standard benchmarks.

In summary, our contribution is threefold: (1) We leverage the representation layers rather than the BN layers in the pre-trained model for CTTA. This paradigm shift mitigates the reliance on large batch size, an obstacle towards memory-efficient CTTA. (2) We empirically show that the importance of representation layers differs for CTTA, and propose a simple metric to identify adaptation-critical representation layers. The layer-wise selective updating scheme notably reduces the storage of activations for CTTA. (3) We extensively validate the effectiveness of our solution on various models and datasets. Under the same memory constraints, we outperform the state-of-the-art CTTA methods SAR, ECoTTA, and EATA, showing accuracy improvements of 4.5%, 4.9%, and 14.8% on CIFAR10-C, CIFAR100-C, and ImageNet-C, respectively. Notably, we achieve a threefold reduction in average memory usage across different batch sizes, while boosting average accuracy by 8.1%, 3.6%, and 0.2% on the three datasets.

II. RELATED WORK

A. Test-Time Adaptation

Test-time adaptation (TTA) addresses shifts between source and target domains during testing without accessing source data [2]. For example, TENT [12] proposes an entropy minimization based unsupervised test-time objective, and adapts to the target domain by updating the affine transformations in the BN layers. Alternatively, SHOT [1] considers adaptation as source hypothesis transfer, and updates the feature representation layers to the target domain while keeping the classification layers unchanged. As an orthogonal solution, EATA [8] only selects reliable samples for adaptation. Continual TTA (CTTA) [3] extends the scope of TTA from a single target domain to a sequence of continuously changing domains. SWA [13]

explores safety supervision for CTTA. SAR [14] proposes a sharpness-aware and reliable entropy minimization to stabilize CTTA. LAW [4] leverages Fisher Information Matrix (FIM) to identify layers to keep or adapt.

Among the questions of *what*, *when*, and *how* to adapt in CTTA, we focus on *what* to update at low *memory* cost, and adopt standard approaches for entropy-based sample selection [8] and entropy minimization-based loss [12].

B. Efficient On-device Model Adaptation

There is an increasing interest to enable model adaptation on memory-limited platforms, where the bottleneck lies in the storage of activations and the use of large batch size for effective backpropagation. As a special model adaptation problem (unsupervised continual domain adaptation), CTTA faces the same memory bottleneck, and a few pioneer studies have explored memory-efficient CTTA. For example, TENT [2] reduces activation during adaptation by exclusively updating BN layers. MECTA [9] further prunes activations of cached BN layers during backpropagation. EcoTTA [10] keeps the entire model parameters frozen and only updates a small set of extra meta layers. Although these approaches significantly reduce the memory consumption by saving storage of activations, they still depend on large batch sizes. To operate with small batch sizes, SAR [14] replaces the BN with layer normalization (group normalization). TTN [11] adjusts the weight of the BN layers updated by the test-time batch according to the domain offset sensitivity of each BN layer.

Unlike the solutions above, we target at memory-efficient CTTA with both reduced activation size and batch size. The key idea is to update only a few representation layers sensitive to distribution shifts.

III. PROBLEM STATEMENT

a) Continual TTA: We consider the standard continual test-time adaptation (CTTA) setup [2]. A model $f_{\theta}(y|x)$ with parameters θ is pre-trained on source data $D_s = (X^S, Y^S) = \{(x, y) \sim p_s(x, y)\}$, where $x \in X^S$ is e.g., an image and $y \in Y^S$ is its associated label from the source class set Y^S . The target data are *unlabeled* and sampled from an arbitrary target distribution $D_t = \{x \sim p_t(x)\}$, where $p_t(x)$ undergoes *continual* changes over time t . Following the covariate shift assumption, $p_s(y|x) = p_t(y|x)$ and $p_s(x) \neq p_t(x)$. As the target distribution gradually shifts from the source, $f_{\theta}(y|x)$ no longer approximates $p(y|x)$, and needs adaptation to retain accuracy. Unique in CTTA, θ_t should make predictions online when source data is absent and adapt themselves into θ_{t+1} for the subsequent input, given access to only $p_t(x)$ at step t .

b) Memory Footprint of CTTA: In CTTA, the model f_{θ} is often a neural network $f_{\theta}(\cdot) = f_{\theta_L}(f_{\theta_{L-1}}(\dots(f_{\theta_1}(\cdot))\dots))$, with parameters θ_l in layer l . Assume the parameter θ_l consists of weights \mathbf{W}_l and bias \mathbf{b}_l , and the input and output features of this layer are \mathbf{a}_l and \mathbf{a}_{l+1} , respectively. Given the forward pass $\mathbf{a}_{l+1} = \mathbf{a}_l \mathbf{W}_l + \mathbf{b}_l$, the corresponding backward pass with batch size 1 is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_{l+1}} \mathbf{W}_l^T, \frac{\partial \mathcal{L}}{\partial \mathbf{W}_l} = \mathbf{a}_l^T \frac{\partial \mathcal{L}}{\partial \mathbf{a}_{l+1}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_{l+1}} \quad (1)$$

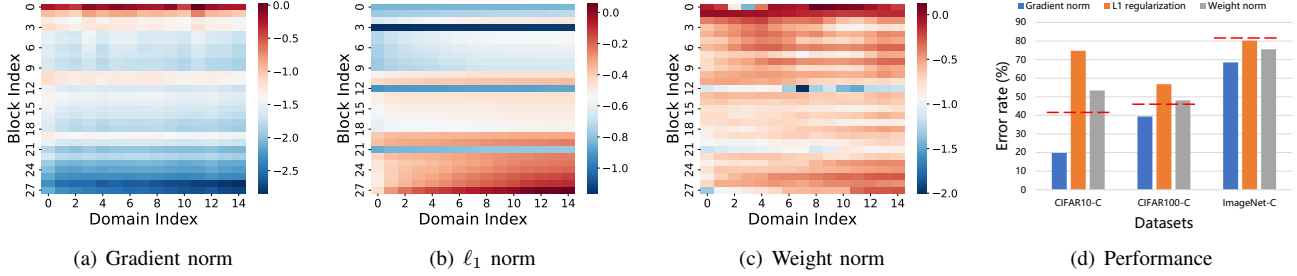


Fig. 2. We evaluated the performance of selecting the top-K layers of the adaptation model using various metrics on three commonly used CTTA benchmarks.

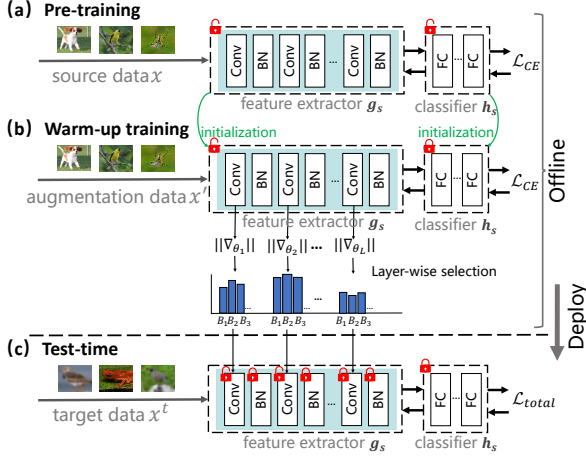


Fig. 3. The pipeline of our FoCTTA framework.

Eq. 1 shows that to update a learnable layer with weight \mathbf{W}_l , one must store all \mathbf{a}_l to compute the gradients. Hence, for a model with L layers, the memory cost during backpropagation given batch size B can be estimated as

$$m(\text{cost}) = \sum_{l=1}^L (m(\theta_l) + m(\mathbf{a}_l) \cdot B) \quad (2)$$

where $m(\cdot)$ denotes the memory requirements. From Eq. 2, the memory cost of adaptation (via gradient descent) increases with # layers L updated and the batch size B . Note that memory of the weights θ are constant during adaptation.

c) *Parameters to Update in CTTA*: Existing CTTA methods update the pre-trained model $f_{\theta}(y|x)$ at test-time to better approximate $p(y|x)$. The parameters θ are often divided into adaptable weights θ^a and frozen weights θ^f , where the adaptable weights are updated by minimizing an unsupervised loss $\mathcal{L}(x; \theta^a \cup \theta^f)$, $x \sim p_t(x)$ w.r.t. θ^a . CTTA methods differ in their choices of the parameter partition $\{\theta^a, \theta^f\}$ and loss function \mathcal{L} , where the choice of adaptable weights θ^a affects the memory cost. Most CTTA proposals opt for *computation efficiency*, which does not translate into *memory efficiency* due to the storage of substantial activations and the use of large batch sizes. Therefore, we focus on identifying adaptable weights θ^a that yield low memory cost without compromising adaptation accuracy.

IV. METHOD

We propose FoCTTA, a memory-efficient CTTA scheme that focuses on adaptation-critical representation layers. FoCTTA updates representation layers rather than BN layers to mitigate dependence on a large batch size B . It also selectively updates the representation layers to reduce the number of adaptable layers L . From Eq. 2, the reduction of B and L decreases the memory usage during mini-batch gradient descent, the key memory bottleneck in CTTA. Fig. 3 illustrates the workflow of FoCTTA.

A. Updating Representation Layers for CTTA

As mentioned, a core issue in CTTA is to decide the adaptable parameters θ^a . Inspired by the concept of source hypothesis transfer [1], we focus on representation layers for CTTA. According to [1], $f_{\theta}(y|x)$ can be decomposed into a feature extractor g_s and a classifier h_s , where $f_{\theta}(y|x) = h_s(g_s(y|x))$. It suffices to update g_s for CTTA. Since adapting the feature extractor reduces the number of adaptable layers, it holds potential for memory-efficient CTTA. Furthermore, as shown in V-B, updating the representation layers works with small batch sizes, an essential advantage over BN layers.

We push the idea one step forward by asking the following: can we achieve high CTTA accuracy by *selectively* updating the representation layers? That is, we hypothesize that the *importance* of representation layers varies for CTTA, thereby only the critical ones needs updating. Although layer importance and layer-wise fine-tuning have been extensively explored in network pruning [15], their observations were intended for supervised learning in the same domain. It is unclear whether similar observations and hypothesis hold for unsupervised adaptation to domains with shifts.

To this end, we conduct an empirical study to understand the importance of individual representation layers to CTTA. Specifically, we pick three layer importance metrics: gradient norm [16], ℓ_1 norm [17], and weight norm [18] commonly used for supervised training without domain shift, and select the top-K important layers ($K = 5$) for adaptation. We tested various models and datasets in standard CTTA benchmarks. Fig. 2 shows the results. More experimental details and results are shown in the supplementary material Sec. A. We make the following observations.

- *Importance of representation layers differs for CTTA.* The importance indicated by different metrics varies across layers. For instance, the gradient norm suggests important representation layers are the shallow layers, the ℓ_1 norm shows deeper layers are more important, while the weight norm indicates the critical representation layers are distributed throughout the model.
- *Gradient norm indicates layer importance for CTTA.* Across three datasets, we consistently observe that important representation layers selected by the gradient norm achieve the highest accuracy. The important layers identified by the other two metrics even yield performance lower than the model without adaptation.

These observations validate the hypothesis that layer importance also varies in the context of CTTA and show the gradient norm as an effective importance metric for CTTA.

B. Identifying Critical Representation Layers

To identify important representation layers sensitive to domain shifts, we leverage an additional *warm-up training* phase after pre-training but before testing. It simulates domain shifts by augmenting the original training data.

Specifically, we take each original data point x and create an augmented counterpart x' that shares the same semantic information. As shown in Fig. 3, in the warm-up training phase, we freeze the classifier h_s of the pre-trained model, optimize the feature extractor g_s with cross-entropy loss using the augmented data x' as input, and collect the gradient norms of each layer of g_s in each batch.

The average gradient norm of each layer quantifies the layer’s importance:

$$s = \lceil \log \frac{1}{B_N} \sum_{b=1}^{B_N} \|\nabla_{\theta_l}\| \rceil_{l=1}^L \quad (3)$$

where B_N , θ_l and ∇_{θ_l} are the N -th batch size, parameters of layer l , gradients of layer l , respectively. The vector s of length L stores the importance of all layers in the pre-trained model. We then sort s from high to low. The layers with the largest average gradient norms are identified as important for CTTA. In practice, the selection is conducted by $\alpha \|s\|$, where α is a tunable hyperparameter to balance memory cost and accuracy. Note that the warm-up training is performed before test time, which is common in other CTTA solutions [8], [10]. Also, it does not require access to the source dataset (X^S, Y^S) during test-time, and is agnostic to the architecture and pre-training method of the original model.

C. CTTA Objective

During test-time adaptation, FoCTTA exclusively optimizes adaptation-crucial layers for the target domains, maintaining the other layers unchanged. In line with [8], we utilize the entropy predicted by the adaptation model to identify reliable samples for subsequent model optimization. Consequently, the online adaptation loss function is formulated as

$$\mathcal{L}_{ent} = \mathbb{I}_{\{H(\hat{y}) < H_0\}} \cdot H(\hat{y}), H(\hat{y}) = - \sum_c p(\hat{y}) \log p(\hat{y}) \quad (4)$$

where \hat{y} is the prediction output of a test image, and $p(\cdot)$ denotes the softmax function. The symbol $\mathbb{I}_{\{\cdot\}}$ represents an indicator function, and H_0 is a predefined hyperparameter.

In addition, to prevent catastrophic forgetting [3], [8] and error accumulation [19] due to long-term continuous adaptation, we add a regularization term to the loss function when optimizing Eq. 4. The final loss function is

$$\mathcal{L}_{total} = \mathcal{L}_{ent} + \lambda \sum_{m=1}^M \|\tilde{x}_m - x_m\|_1 \quad (5)$$

where λ is a positive scalar to control the ratio between two terms in the loss function. $M = \alpha \|s\|$ denotes the number of layers to be updated. The terms \tilde{x}_m and x_m represent the m -th output of the adapted model and the original model, respectively. Our evaluations show that a small portion of (1.0%) representation layers need to be updated.

V. EXPERIMENTS

A. Experimental Setup

We use CIFAR10, CIFAR100 [20], and ImageNet [21] as the source domain datasets, while CIFAR10-C, CIFAR100-C, and ImageNet-C as the corresponding target domain datasets.

All experiments use the PyTorch. We use identical pre-trained models: WideResNet-28 and WideResNet-40 from RobustBench, and ResNet-50 from TTT++. During warm-up training, data augmentation (e.g., color jittering, padding, random affine, cropping, inversion, and flipping) is applied to the source data. The model is then fine-tuned for one epoch using cross-entropy loss and the Adam optimizer (learning rate: 0.00025) to identify crucial adaptation layers. At test time, we use the Adam optimizer with a learning rate of 0.001 for CIFAR datasets and 0.00025 for ImageNet. The entropy threshold H_0 is set to $0.4 \times \ln C$, where C is the number of classes, and $\lambda = 1$ and $\alpha = 0.1$ are empirically set.

We assess various methods through two configurations: 1) **Under memory constraints**, we test the error rates; 2) **Under the same batch size**, we compute the error rates and memory consumption. We compare our method with Source, Continual TENT [2], CoTTA [3], ECoTTA [10], EATA [8], SAR [14], SWA [13], and LAW [4].

B. Performance Evaluation

Performance with memory constraints. We evaluate accuracy under memory constraints by adjusting batch sizes to ensure comparable memory consumption across methods. Table II shows per-domain error rates in CTTA, along with average memory consumption and error rate. Analysis of Table II reveals that FoCTTA significantly improves accuracy across all datasets and models in memory-constrained environments. Methods like TENT, EATA, and SAR, which adapt all BN layer affine parameters, or CoTTA, SWA, and LAW, which update all model parameters, require storing large activations for backpropagation. This forces the use of smaller batch sizes, leading to performance degradation and collapse. For instance, with a 50MB memory constraint,

TABLE I
COMPARISON OF ERROR RATE (%) AND MEMORY CONSUMPTION (MB) ON THE HIGHEST CORRUPTION SEVERITY UNDER THE SAME BATCH SIZE

| Datasets | Method | Batch Size | | | | | | | | | | | | Avg. err | Avg. mem |
|------------|----------------|-------------|---------|-------------|--------|-------------|--------|-------------|--------|-------------|--------|-------------|-------|-------------|----------|
| | | 128 | | 64 | | 32 | | 16 | | 8 | | 4 | | | |
| | | Err. | Mem. | Err. | Mem. | Err. | Mem. | Err. | Mem. | Err. | Mem. | Err. | Mem. | | |
| CIFAR10-C | Source | 43.5 | 204.3 | 43.5 | 120.4 | 43.5 | 78.4 | 43.5 | 57.5 | 43.5 | 47.0 | 43.5 | 41.7 | 43.5 | 91.6 |
| | EATA | 18.7 | 1240.3 | 20.3 | 638.4 | 23.9 | 337.5 | 28.9 | 187.0 | 47.2 | 111.8 | 47.2 | 74.2 | 31.0 | 431.5 |
| | Continual TENT | 25.6 | 1240.3 | 38.1 | 638.4 | 46.5 | 337.5 | 62.2 | 187.0 | 78.7 | 111.8 | 86.4 | 74.2 | 56.3 | 431.5 |
| | CoTTA | 17.9 | 2939.1 | 18.7 | 1688.4 | 22.2 | 1063.1 | 34.5 | 750.4 | 59.3 | 594.1 | 79.0 | 515.9 | 38.6 | 1133.4 |
| | SWA | 17.7 | 2939.1 | 18.6 | 1688.4 | 21.7 | 1063.1 | 31.8 | 750.4 | 56.9 | 594.1 | 76.3 | 515.9 | 37.2 | 1133.4 |
| | ECoTTA | 19.6 | 747.2 | 21.8 | 397.0 | 22.3 | 221.8 | 39.8 | 134.3 | 46.7 | 90.5 | 54.2 | 68.6 | 34.1 | 276.6 |
| | SAR | 20.4 | 1240.3 | 20.7 | 638.4 | 21.4 | 337.5 | 22.9 | 187.0 | 32.6 | 111.8 | 75.7 | 74.2 | 32.3 | 431.5 |
| | LAW | 16.3 | 2647.3 | 17.5 | 1396.6 | 18.6 | 771.3 | 22.8 | 458.6 | 51.2 | 302.3 | 78.2 | 224.1 | 34.1 | 966.7 |
| | FoCTTA (Ours) | 16.7 | 356.0 | 17.3 | 197.9 | 18.9 | 118.9 | 21.6 | 79.4 | 27.1 | 59.7 | 35.7 | 49.8 | 22.9 | 143.6 |
| CIFAR100-C | Source | 46.8 | 35.8 | 46.8 | 19.0 | 46.8 | 10.6 | 46.8 | 6.4 | 46.8 | 4.4 | 46.8 | 3.3 | 46.8 | 13.3 |
| | EATA | 36.1 | 367.2 | 37.0 | 184.7 | 39.7 | 93.5 | 44.1 | 47.9 | 51.7 | 25.1 | 74.7 | 13.7 | 47.2 | 122.0 |
| | Continual TENT | 41.3 | 367.2 | 49.0 | 184.7 | 79.2 | 93.5 | 87.0 | 47.9 | 95.4 | 25.1 | 98.3 | 13.7 | 75.0 | 122.0 |
| | CoTTA | 38.1 | 783.6 | 39.6 | 405.3 | 43.4 | 216.2 | 51.6 | 121.6 | 71.4 | 74.3 | 91.3 | 50.7 | 55.9 | 275.3 |
| | SWA | 37.8 | 783.6 | 39.1 | 405.3 | 42.8 | 216.2 | 50.3 | 121.6 | 69.6 | 74.3 | 90.1 | 50.7 | 55.0 | 275.3 |
| | ECoTTA | 37.2 | 174.8 | 37.9 | 88.8 | 39.6 | 45.8 | 46.2 | 24.3 | 85.7 | 13.6 | 96.4 | 8.2 | 57.2 | 59.3 |
| | SAR | 35.5 | 367.2 | 36.2 | 184.7 | 40.2 | 93.5 | 68.7 | 47.9 | 94.1 | 25.1 | 98.3 | 13.7 | 62.2 | 122.0 |
| | LAW | 35.6 | 779.0 | 36.1 | 400.7 | 38.1 | 211.6 | 42.4 | 117.0 | 56.9 | 69.7 | 92.0 | 46.1 | 50.2 | 270.7 |
| | FoCTTA (Ours) | 34.3 | 88.4 | 34.7 | 45.9 | 35.8 | 24.7 | 38.7 | 14.1 | 44.5 | 8.8 | 73.6 | 6.1 | 43.6 | 31.3 |
| ImageNet-C | Source | 82.4 | 1053.2 | 82.4 | 539.4 | 82.4 | 282.5 | 82.4 | 154.0 | 82.4 | 89.8 | 82.4 | 57.7 | 82.4 | 362.8 |
| | EATA | 59.1 | 5780.3 | 60.8 | 2903.0 | 63.7 | 1464.4 | 68.9 | 745.0 | 79.9 | 385.4 | 86.4 | 205.5 | 69.8 | 1913.9 |
| | Continual TENT | 67.0 | 5780.3 | 67.5 | 2903.0 | 71.7 | 1464.4 | 91.7 | 745.0 | 97.4 | 385.4 | 99.3 | 205.5 | 82.4 | 1913.9 |
| | CoTTA | 65.9 | 11520.4 | 66.1 | 5913.5 | 67.3 | 3110.1 | 82.2 | 1708.3 | 96.5 | 1007.5 | 99.7 | 657.1 | 79.6 | 3986.2 |
| | SWA | 65.3 | 11520.4 | 65.8 | 5913.5 | 66.7 | 3110.1 | 81.4 | 1708.3 | 93.6 | 1007.5 | 99.4 | 657.1 | 78.7 | 3986.2 |
| | ECoTTA | 80.8 | 2540.2 | 89.7 | 1332.7 | 97.8 | 729.0 | 99.5 | 427.2 | 99.8 | 276.2 | 99.8 | 200.8 | 94.6 | 917.7 |
| | SAR | 61.9 | 5780.3 | 62.4 | 2903.0 | 63.8 | 1464.4 | 76.7 | 745.0 | 80.4 | 385.4 | 85.2 | 205.5 | 71.7 | 1913.9 |
| | LAW | 60.7 | 11316.0 | 61.4 | 5709.1 | 62.9 | 2905.9 | 74.4 | 1503.9 | 94.2 | 803.1 | 98.9 | 452.7 | 75.4 | 3531.1 |
| | FoCTTA (Ours) | 61.4 | 1598.4 | 62.8 | 814.8 | 65.1 | 423.0 | 68.1 | 227.2 | 77.0 | 129.2 | 83.0 | 80.3 | 69.6 | 545.5 |

TABLE II
COMPARISON OF ERROR RATE (%) ON THE HIGHEST CORRUPTION SEVERITY UNDER MEMORY CONSTRAINTS.

| Method | Metrics | Datasets | | |
|----------------|----------|-----------|------------|------------|
| | | CIFAR10-C | CIFAR100-C | ImageNet-C |
| EATA | Avg. err | 22.3 | 44.1 | 79.9 |
| | Mem. | 394.0 | 47.9 | 385.4 |
| Continual TENT | Avg. err | 34.0 | 87.0 | 97.4 |
| | Mem. | 394.0 | 47.9 | 385.4 |
| CoTTA | Avg. err | 99.9 | 91.3 | 99.7 |
| | Mem. | 394.2 | 50.7 | 393.8 |
| SWA | Avg. err | 99.9 | 90.1 | 99.6 |
| | Mem. | 394.2 | 50.7 | 393.8 |
| ECoTTA | Avg. err | 21.8 | 39.6 | 99.5 |
| | Mem. | 397.0 | 45.8 | 427.2 |
| SAR | Avg. err | 21.2 | 68.7 | 80.4 |
| | Mem. | 394.0 | 47.9 | 385.4 |
| LAW | Avg. err | 24.1 | 92.0 | 98.9 |
| | Mem. | 378.8 | 46.1 | 452.7 |
| FoCTTA (Ours) | Avg. err | 16.7 | 34.7 | 65.1 |
| | Mem. | 356.0 | 45.9 | 423.0 |

TABLE III
ABLATION EXPERIMENTS ON CIFAR100-C WITH A BATCH SIZE OF 32.

| Method | FoCTTA | w/o Reg. | w/o LS. | w/o Reg. and LS. |
|-------------|--------|----------|---------|------------------|
| Avg. err(%) | 35.8 | 36.1 | 98.7 | 98.7 |

TENT achieves 87% accuracy on CIFAR100-C, while the original model performs at 46.8%. In contrast to ECOTTA, which updates additional side-way meta networks, FoCTTA only updates 1.0% of the representation layers, resulting in an average accuracy improvement of 14.8% across three datasets. FoCTTA outperforms state-of-the-art methods like SAR, ECOTTA, and EATA, achieving accuracy improvements

TABLE IV
ABLATION STUDY ON DATA AUGMENTATION TYPE.

| type | jitter | +flip | +blur | +invert |
|-------------|--------|-------|-------|---------|
| Avg err.(%) | 36.4 | 36.2 | 35.9 | 35.8 |

of 4.5%, 4.9%, and 14.8% on CIFAR10-C, CIFAR100-C, and ImageNet-C, respectively.

Our findings highlight FoCTTA’s superiority in resource-constrained environments. Unlike methods that update all BN layers, FoCTTA selectively adapts only the drift-sensitive representation layers, avoiding the need for large batch sizes. This targeted update reduces memory usage by eliminating unnecessary activations, optimizing both batch size and activation memory efficiency for CTTA adaptation. Furthermore, we reveal the challenges of full-parameter update methods under memory constraints, particularly the performance collapse observed with smaller batch sizes. This research deepens our understanding of the adaptability and performance of various methods in memory-limited settings.

Performance with the same batch size. Table I presents the average error rate and memory consumption on the corrupted dataset under CTTA, considering different batch sizes, models, and datasets. FoCTTA reduces memory consumption by threefold while improving accuracy by 8.1%, 3.6%, and 0.2% on CIFAR-10C, CIFAR-100C, and ImageNet-C, respectively. This demonstrates FoCTTA’s robustness across batch sizes and memory efficiency. Its superior performance stems from updating only adaptation-critical representation layers, reducing reliance on batch size. Unlike methods that optimize all BN layers (TENT, EATA, SAR) or update all parameters

TABLE V
AVG ERR.(%) WITH DIFFERENT VALUES OF λ .

| λ | 0.1 | 0.5 | 0.9 | 1.0 | 1.2 | 1.5 |
|-------------|------|------|------|------------|------|------|
| Avg err.(%) | 36.0 | 35.9 | 35.9 | 35.8 | 36.0 | 36.1 |

TABLE VI
AVG ERR.(%) WITH DIFFERENT VALUES OF α .

| α | 0.03 | 0.05 | 0.08 | 0.1 | 0.15 | 0.20 |
|-------------|------|------|------|------------|------|------|
| Avg err.(%) | 37.9 | 36.8 | 36.0 | 35.8 | 35.8 | 36.1 |

(CoTTA, SWA, LAW), which require larger batch sizes or store excessive activations, FoCTTA optimizes both batch size and activation, providing significant memory savings without sacrificing accuracy.

C. Ablation Study

In the following experiments, if not specified, we use CIFAR100-C and robustly pre-trained WideResNet-40.

Necessity of Each Design. Table III demonstrates the influence of removing individual designs in FoCTTA on its performance. The results show a significant performance decline when any FoCTTA design is removed, highlighting the crucial role of each design in achieving exceptional performance. Particularly, the choice of the adaptation-critical layer is vital for FoCTTA, and its absence leads to performance collapse.

Influence of λ and α : Table V and Table VI show the effect of λ and α on the error rate of FoCTTA, respectively. Here, λ represents the weight of the regularization term. α represents the number of adaptation-critical layers selected for optimization. We choose the setting that achieves the best performance, with $\lambda = 1.0$ and $\alpha = 0.1$ as default values.

Influence of the data augmentation type: We use data augmentation in the warm-up training phase to simulate domain shifts. Table IV demonstrates the influence of data augmentation types on FoCTTA, revealing its robustness. By using default settings, we choose color jittering and inverting, achieving the best performance.

VI. CONCLUSION

This paper addresses memory efficiency challenges during CTTA adaptation by proposing FoCTTA, a low-memory strategy. Instead of updating all BN layers, FoCTTA selectively adapts drift-sensitive representation layers, eliminating the need for large batch sizes. By updating only critical layers, it avoids storing excessive activations, enhancing memory efficiency while maintaining effective adaptation. This approach is validated through evaluations on various models and datasets.

VII. ACKNOWLEDGEMENT

This paper is supported by the National Key R&D Program of China under Grant No. 2023YFB4503100. Zimu Zhou’s research is funded by the Chow Sang Sang Group Research Fund No. 9229139 and CityU APRC Grant No. 9610633.

REFERENCES

- [1] Jian Liang, Dapeng Hu, and Jiashi Feng, “Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation,” in *International conference on machine learning*. PMLR, 2020, pp. 6028–6039.
- [2] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell, “Tent: Fully test-time adaptation by entropy minimization,” in *International Conference on Learning Representations*, 2020.
- [3] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai, “Continual test-time domain adaptation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7201–7211.
- [4] Junyoung Park, Jin Kim, Hyeongjun Kwon, Ilhoon Yoon, and Kwanghoon Sohn, “Layer-wise auto-weighting for non-stationary test-time adaptation,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 1414–1423.
- [5] Zhiqiang Yuan, Jiawei Zhang, Ying Deng, Yeshuang Zhu, Jie Zhou, and Jinchao Zhang, “Vsd2m: A large-scale vision-language sticker dataset for multi-frame animated sticker generation,” *arXiv preprint arXiv:2412.08259*, 2024.
- [6] Zhiqiang Yuan, Ting Zhang, Ying Deng, Jiawei Zhang, Yeshuang Zhu, Zexi Jia, Jie Zhou, and Jinchao Zhang, “Walkvln: Aid visually impaired people walking by vision language model,” *arXiv preprint arXiv:2412.20903*, 2024.
- [7] Zhiqiang Yuan, Ting Zhang, Ying Deng, Jiawei Zhang, Yeshuang Zhu, Zexi Jia, Jie Zhou, and Jinchao Zhang, “Rdtf: Resource-efficient dual-mask training framework for multi-frame animated sticker generation,” *arXiv preprint arXiv:2503.17735*, 2025.
- [8] Shuaicheng Niu, Jiayang Wu, Yifan Zhang, Yaofu Chen, Shijian Zheng, Peilin Zhao, and Minghui Tan, “Efficient test-time model adaptation without forgetting,” in *International conference on machine learning*. PMLR, 2022, pp. 16888–16905.
- [9] Junyuan Hong, Lingjuan Lyu, Jiayu Zhou, and Michael Spranger, “Mecta: Memory-economic continual test-time model adaptation,” in *2023 International Conference on Learning Representations*, 2023.
- [10] Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi, “Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 11920–11929.
- [11] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi, “Ttn: A domain-shift aware batch normalization in test-time adaptation,” in *The Eleventh International Conference on Learning Representations*, 2022.
- [12] Fuming You, Jingjing Li, and Zhou Zhao, “Test-time batch statistics calibration for covariate shift,” *arXiv preprint arXiv:2110.04065*, 2021.
- [13] Xu Yang, Yanan Gu, Kun Wei, and Cheng Deng, “Exploring safety supervision for continual test-time domain adaptation,” in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023, pp. 1649–1657.
- [14] Shuaicheng Niu, Jiayang Wu, Yifan Zhang, Zhiqian Wen, Yaofu Chen, Peilin Zhao, and Minghui Tan, “Towards stable test-time adaptation in dynamic wild world,” *arXiv preprint arXiv:2302.12400*, 2023.
- [15] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi, “A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations,” *arXiv preprint arXiv:2308.06767*, 2023.
- [16] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz, “Pruning convolutional neural networks for resource efficient inference,” *arXiv preprint arXiv:1611.06440*, 2016.
- [17] Song Han, Jeff Pool, John Tran, and William Dally, “Learning both weights and connections for efficient neural network,” *Advances in neural information processing systems*, vol. 28, 2015.
- [18] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, “Pruning filters for efficient convnets,” *arXiv preprint arXiv:1608.08710*, 2016.
- [19] Eric Arazo, Diego Ortego, Paul Albert, Noel E O’Connor, and Kevin McGuinness, “Pseudo-labeling and confirmation bias in deep semi-supervised learning,” in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.
- [20] A Krizhevsky, “Learning multiple layers of features from tiny images,” *Master’s thesis, University of Tront*, 2009.
- [21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.