Ramen: Robust Test-Time Adaptation of Vision-Language Models with Active Sample Selection

Wenxuan Bao* Yanjun Zhao* Xiyuan Yang Jingrui He†
University of Illinois Urbana-Champaign
{wbao4,yanjunzh,xiyuany4,jingrui}@illinois.edu

Abstract

Pretrained vision-language models such as CLIP exhibit strong zero-shot generalization but remain sensitive to distribution shifts. Test-time adaptation adapts models during inference without access to source data or target labels, offering a practical way to handle such shifts. However, existing methods typically assume that test samples come from a single, consistent domain, while in practice, test data often include samples from mixed domains with distinct characteristics. Consequently, their performance degrades under mixed-domain settings. To address this, we present Ramen, a framework for robust test-time adaptation through active sample selection. For each incoming test sample, Ramen retrieves a customized batch of relevant samples from previously seen data based on two criteria: domain consistency, which ensures that adaptation focuses on data from similar domains, and prediction balance, which mitigates adaptation bias caused by skewed predictions. To improve efficiency, Ramen employs an embedding-gradient cache that stores the embeddings and sample-level gradients of past test images. The stored embeddings are used to retrieve relevant samples, and the corresponding gradients are aggregated for model updates, eliminating the need for any additional forward or backward passes. Our theoretical analysis provides insight into why the proposed adaptation mechanism is effective under mixed-domain shifts. Experiments on multiple image corruption and domain-shift benchmarks demonstrate that Ramen achieves strong and consistent performance, offering robust and efficient adaptation in complex mixed-domain scenarios. Our code is available at https://github.com/baowenxuan/Ramen.

1. Introduction

Pretrained vision-language models (VLMs) such as CLIP [33] have demonstrated strong zero-shot generalization across a wide range of vision tasks [31, 34, 52]. However, their performance can degrade significantly under distribution shifts, including image corruptions [17] and domain shifts [14]. Test-time adaptation (TTA) has emerged as a promising strategy for improving model robustness under distribution shifts, by adapting the model during test-time without accessing source data or target labels [24, 42, 44]. This property makes TTA particularly suitable for the adaptation of pretrained VLMs, where the source training data is often large-scale, proprietary, or unavailable at deployment.

Although existing TTA methods have demonstrated impressive performance on standard benchmarks, most of them rely on a key assumption that testing samples are drawn from a single, consistent domain. In practice, however, this assumption is often violated, as test data may contain samples from multiple domains with distinct characteristics [28, 29, 45]. For instance, a user's photo gallery on a smartphone may include images captured under different weather conditions and lighting environments, or even collected from different platforms. In such **mixed-domain scenarios**, conventional TTA algorithms often exhibit degraded performance, as they struggle to generalize across diverse and inconsistent test distributions [9, 25, 27–29, 38].

We argue that this degradation arises because a single model is forced to adapt simultaneously to data from multiple, distinct domains. As a result, the model cannot specialize for each domain but instead adapts to an averaged domain representation, leading to suboptimal adaptation. To address this issue, we propose *Ramen*, which enables **R**obust test-time **a**daptation with sa**m**ple s**e**lectio**n**. Instead of passively adapting on the entire mixed test stream, which dilutes domain-specific signals, *Ramen* actively constructs a **customized support set**, a small batch of previously seen test samples most relevant to the current one, for model adaptation. The selected samples are determined by two criteria: (1) **domain consistency**, which ensures that adaptation focuses on data from similar domains measured by their distance in image embeddings, and (2) **prediction balance**, which mitigates adaptation bias caused

*Equal contribution.
†Corresponding author.

by skewed model predictions. To further reduce the computational cost of per-sample adaptation, *Ramen* employs an **embedding-gradient cache** that stores both the embeddings and sample-level gradients of past test images. The stored embeddings are used for sample retrieval, and the corresponding gradients are aggregated for model updates, eliminating the need for additional forward or backward passes. Our theoretical analysis provides insight into why *Ramen* is effective under mixed-domain shifts, and extensive experiments across image corruption and domain-shift benchmarks demonstrate that *Ramen* achieves robust and efficient adaptation in complex mixed-domain scenarios.

We summarize our main contributions as follows:

- We introduce an *active sample selection* framework to enable effective adaptation under mixed-domain shifts, guided by two empirical selection criteria: *domain consistency* and *prediction balance*.
- We propose an efficient algorithm, *Ramen*, which implements active sample selection through an *embedding-gradient cache* that reuses stored embeddings and gradients for lightweight adaptation.
- We provide theoretical analysis that reveals the underlying principles of a broad class of TTA methods and explains why *Ramen* improves adaptation performance under mixed-domain settings.
- We conduct extensive experiments on image corruption and domain-shift benchmarks, demonstrating both the *effectiveness* and *efficiency* of *Ramen*.

2. Related Works

In this section, we summarize related works of TTA in both single-domain and mixed-domain scenarios. We provide a broader discussion of related works in Appendix A.1.

Single-domain TTA TTA adapts a pre-trained model to an unlabeled target domain without access to source data. Single-domain TTA assumes that all target samples are drawn from one consistent domain. Among existing methods, one common approach optimizes a self-supervised objective, such as entropy minimization [12, 23, 27, 39], image–text alignment [15, 30], or inter-class variance [3], to adjust the model's normalization layers in response to distribution shifts. Another category is memory-based, which stores embeddings of high-confidence samples and uses embedding similarity to refine model predictions [21, 50, 51]. Augmentation-based methods generate multiple augmented views for each image and either aggregate predictions across views [9, 11] or minimize marginal entropy to enforce consistency [36, 37], but these approaches often incur substantially higher computational costs.

Mixed-domain TTA and more Mixed-domain TTA refers to the setting where a model must adapt to a target data stream containing samples from multiple domains. Prior works such as SAR [28] and ROID [25] revealed the performance drop of single-domain TTA methods in this setting and proposed techniques like sharpness-aware minimization and weight ensembling to mitigate model collapse. However, these approaches still rely on a single model adapting simultaneously to diverse domains, fundamentally limiting their effectiveness. UnMix-TNS [38] addressed this issue by modifying BatchNorm [20] to maintain multiple sets of running statistics, but this solution applied only to BatchNorm layers, which are generally discouraged under mixed-domain settings [28]. A related yet distinct scenario is *continual TTA* [41, 45], where the model is sequentially adapted to a series of domains, one at a time. The key difference lies in the data continuity: in continual TTA, consecutive samples typically come from the same domain, whereas in mixed-domain TTA, even samples within a single batch may originate from different domains.

3. Proposed Method: *Ramen*

In this section, we introduce our proposed method, *Ramen*. Subsection 3.1 formally defines the research problem and the key challenges. Subsection 3.2 presents the active sample selection mechanism and its selection criteria. Subsection 3.3 explains how we leverage an embedding–gradient cache to improve the efficiency of active sample selection, and summarizes the overall algorithm. Figure 1 gives an overview of our proposed method.

3.1. Preliminary

CLIP [33] is a vision–language model composed of an image encoder and a text encoder, jointly trained to align visual and textual representations. Leveraging large-scale image–caption pairs, CLIP learns generalizable representations and demonstrates strong zero-shot recognition capability. For a classification task with C categories, the text encoder converts each class description (e.g., "`a photo of a {class}`") into normalized text embeddings $\boldsymbol{T} = [\boldsymbol{t}_1, \cdots, \boldsymbol{t}_C]^\top \in \mathbb{R}^{C\times d}$, where d denotes the embedding dimension. Given a test image $\boldsymbol{x}_i$, the image encoder outputs a normalized embedding $\boldsymbol{z}_i \in \mathbb{R}^d$. The predicted probability vector is $\boldsymbol{p}_i = \mathrm{softmax}(\exp(t) \cdot \boldsymbol{z}_i \boldsymbol{T}^\top)$, where $\exp(t)$ is a temperature parameter. The predicted label corresponds to the class with highest probability, i.e., $\hat{y}_i = \arg\max_y p_{iy}$. Despite its strong generalization ability, CLIP's performance drops substantially when facing distribution shifts [3, 15, 30], such as image corruptions [17] or domain shifts [14].

Entropy minimization Many existing TTA methods minimize entropy as a surrogate loss [23, 27, 28, 39]. Given CLIP's predicted probability vector $\boldsymbol{p}_i = [p_{i1}, \cdots, p_{iC}]^\top$, the entropy for sample i is computed as $H(\boldsymbol{x}_i) = -\sum_{c=1}^{C} p_{ic} \log p_{ic}$, which measures the model's uncer-

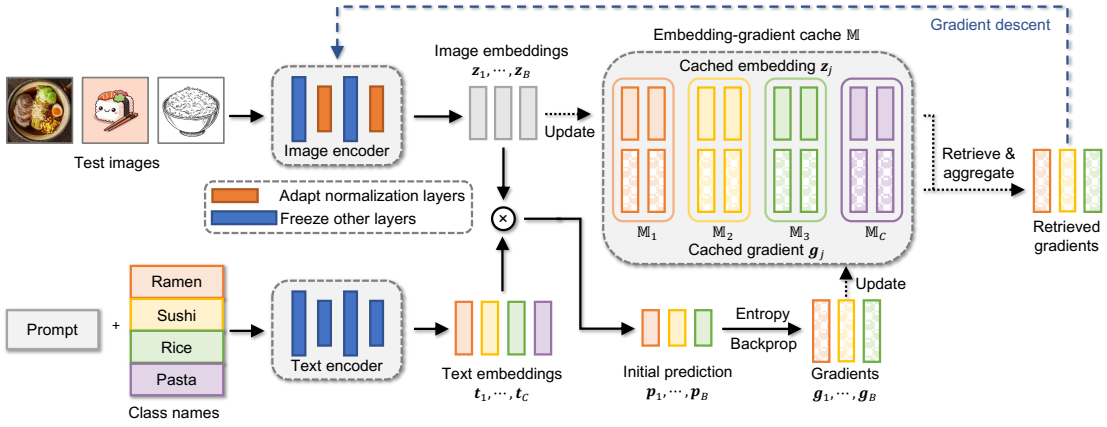


Figure 1. Overview of *Ramen*. For each test sample, (1) compute its image embedding, pseudo-label, and sample-level gradient; (2) update the class-specific memory with these entries; (3) retrieve a support set from the memory; (4) aggregate the cached gradients for model update; and (5) perform inference and reset the model parameters.

tainty on the current sample. The model parameters are then updated by minimizing the average entropy over a batch or in an online (streaming) manner. In practice, most methods only update the normalization layers rather than the entire model. Such an update is highly parameter-efficient (e.g., for the ViT-B/16 visual encoder, fewer than 0.05% of parameters are involved) and helps mitigate catastrophic forgetting during continuous adaptation. Our proposed method follows this general paradigm, as it also minimizes the entropy loss while adapting only the normalization layers.

Challenge of mixed-domain shift During the testing stage, the model performs on-the-fly adaptation and evaluation over a stream of data batches. Conventional TTA methods typically assume that the test data are drawn from a single, consistent domain. However, in real-world scenarios, the test data may come from a mixture of domains. It has been observed that under such mixed-domain shifts, the effectiveness of TTA drops significantly [9, 25, 27–29, 38], which is also confirmed by our experiments (see Figure 2). We argue that this degradation arises because a single model is forced to adapt simultaneously to data from multiple, distinct domains. For example, in a single-domain setting, data from different domains are encountered separately, allowing sufficient adaptation to each domain. In contrast, under mixed-domain settings, even samples within the same batch may originate from different domains, yet they are forced to share one common model with identical parameters. As a result, the model cannot specialize for each domain. Instead, it adapts to an averaged representation across domains, leading to suboptimal adaptation.

3.2. Active Sample Selection

To address this issue, we assign each test sample $\boldsymbol{x}_i$ a distinct model weight $\boldsymbol{w}_i$. Each $\boldsymbol{w}_i$ is obtained by the adapting the model on a sample-specific *support set* $\mathbb{S}_i$, i.e., a subset of previously seen data for adaptation. Formally, this process can be expressed as

$$\hat{y}_i = \text{CLIP}(\boldsymbol{x}_i; \boldsymbol{w}_i), \quad \text{where } \boldsymbol{w}_i = \text{TTA}(\boldsymbol{w}; \mathbb{S}_i), \quad (1)$$

where $\boldsymbol{w}$ denotes the weights of the pre-trained model, and $\text{TTA}()$ represents the test-time adaptation process, which takes the model weights and a support set as input and outputs the adapted weights. $\text{CLIP}()$ refers to the CLIP prediction process as described in Subsection 3.1.

In this subsection, we introduce how to select the support set $\mathbb{S}_i$ for each test sample $\boldsymbol{x}_i$. We propose two selection criteria: domain consistency and prediction balance.

- **Domain consistency** requires that the samples in $\mathbb{S}_i$ should come from the same or similar domain as $\boldsymbol{x}_i$. Although the domain label of each sample and the total number of domains are unknown, pretrained VLMs are typically trained for general-purpose understanding, and their image embeddings implicitly contain domain-related information. Therefore, the more similar two image embeddings are, the more likely they originate from the same domain. We provide an empirical validation in Appendix A.2.
- **Prediction balance** ensures that the samples in $\mathbb{S}_i$ have balanced predictions across classes. If $\mathbb{S}_i$ is dominated by samples predicted as a single class, the adaptation process may introduce a prediction bias, making the model more likely to assign future samples to that class [26, 28]. Maintaining prediction balance helps prevent such bias and improves adaptation stability.

To realize active sample selection that satisfies both domain consistency and prediction balance, we maintain a class-split memory $\mathbb{M}$ to store information of previously seen test samples. These stored samples are later used for

model adaptation (the detailed contents of the memory will be discussed in Subsection 3.3). The memory $\mathbb{M}$ consists of C first-in-first-out (FIFO) queues $\mathbb{M}_1, \cdots, \mathbb{M}_C$, one for each class. Each queue $\mathbb{M}_c$ stores up to K most recently observed samples that are predicted as class c by the zero-shot classifier, where K is the maximum capacity per queue.

When a new test sample $\boldsymbol{x}_i$ arrives, we use its image embedding $\boldsymbol{z}_i$ to retrieve the top-k most similar samples $\mathbb{S}_{ic}$ from each class queue. This design *ensures domain consistency*, as retrieved samples are close to $\boldsymbol{x}_i$ in the embedding space. Then, by concatenating an equal number of samples from each class queue, we form the final support set $\mathbb{S}_i$, which *inherently satisfies prediction balance*. Formally, this process can be expressed as

$$\mathbb{S}_i = \bigcup_{c=1}^{C} \mathbb{S}_{ic}, \quad \text{where } \mathbb{S}_{ic} = \underset{j \in \mathbb{M}_c}{\text{Top}-k} \left(\boldsymbol{z}_i^\top \boldsymbol{z}_j \right). \tag{2}$$

3.3. Embedding-Gradient Cache

While the proposed active sample selection enables customized adaptation for each sample, a naive implementation can be computationally prohibitive. In standard TTA, each test sample requires one backward pass for adaptation.[1] However, if we directly recompute gradients on the support set $\mathbb{S}_i$, each test sample $\boldsymbol{x}_i$ requires $C \cdot k$ backward passes, inflating the overall computational cost by a factor of $C \cdot k$. Such a multiplicative increase makes a straightforward implementation impractical for online or real-time inference.

To address this issue, we propose an efficient mechanism called **embedding-gradient cache**, which leverages the point-wise nature of common TTA objectives such as the entropy loss. For the entropy loss, the loss over a batch $\mathbb{S}_i$ can be written as a weighted average of the sample-level losses:

$$H(\mathbb{S}_i) = \sum_{j \in \mathbb{S}_i} \alpha_{ij} H(\boldsymbol{x}_j), \tag{3}$$

where α_{ij} denotes the weight of sample j. Consequently, the gradient over the batch also satisfies

$$\nabla_{\boldsymbol{w}} H(\mathbb{S}_i) = \sum_{j \in \mathbb{S}_i} \alpha_{ij} \boldsymbol{g}_j, \quad \text{where } \boldsymbol{g}_j = \nabla_{\boldsymbol{w}} H(\boldsymbol{x}_j). \tag{4}$$

Therefore, we can cache the sample-level gradients $\boldsymbol{g}_i = \nabla_{\boldsymbol{w}} H(\boldsymbol{x}_j)$ in memory. In this way, we no longer need to recompute gradients over the support set $\mathbb{S}_j$; instead, we can simply aggregate the cached gradients to obtain the batch-level gradient efficiently.

[1] Following [27], when computing the gradient over a batch of B samples, we count it as B backward passes, since the overall computation cost scales linearly with B.

Therefore, we adopt an embedding–gradient cache mechanism. In the memory, we store the image embeddings $\boldsymbol{z}_j$ and the corresponding sample-level gradients $\boldsymbol{g}_j$ of previously seen samples. For each incoming sample i, we perform the following steps:

1. Compute its image embedding $\boldsymbol{z}_i$, pseudo-label $\hat{y}_i$, and sample-level gradient $\boldsymbol{g}_i$;
2. Update memory $\mathbb{M}_{\hat{y}_i}$ with the new entries $(\boldsymbol{z}_i, \boldsymbol{g}_i)$;
3. Retrieve the corresponding support set $\mathbb{S}_i$ using Eq. (2);
4. Aggregate the cached gradients according to Eq. (4) and update the model;
5. Perform inference with the updated model, and then reset the model parameters.

For the aggregation weights α_{ij}, we employ two common strategies in TTA: entropy weighting [23, 27, 28] and similarity weighting [4, 21, 51]:

$$\alpha_{ij} = \exp(-H(\boldsymbol{x}_j)) \cdot \exp(-\beta \cdot \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2). \tag{5}$$

where $H(\boldsymbol{x}_j)$ denotes the entropy of sample j, and β is a hyper-parameter controlling the influence of similarity. Entropy weighting assigns larger weights to samples with lower entropy (i.e., higher prediction confidence), as they are considered more reliable. Similarity weighting assigns larger weights to samples that are closer to the target sample in the embedding space, since they are more likely to belong to the same domain.

Although the above discussion focuses on a single test sample, in practice we use a simple reparameterization trick to parallelize sample-level gradient computation within a batch, significantly improving efficiency. Implementation details are provided in Appendix A.3.

4. Theoretical Analysis

In this section, we analyze the underlying principles of TTA methods that update normalization layers by minimizing prediction entropy, and explain why active sample selection can further improve their performance.

Setup and assumptions Most TTA methods [3, 12, 15, 28, 30, 39] adapt models by updating the affine parameters in normalization layers. Therefore, we focus our analysis on this component. Common normalization layers [1, 20, 43, 48] share a same structure: a normalization step followed by an element-wise affine/linear transformation. Although the normalization step differs across layer types, the parameterization of the affine/linear transformation remains similar. Following [3], we consider a single normalization layer in a binary classification setting. Let $\boldsymbol{v}_i \in \mathbb{R}^d$ denote the intermediate normalized feature, obtained after the normalization step but before applying its affine transformation. The final image embedding is given by an element-wise linear transformation:

$$\boldsymbol{z}_i = \boldsymbol{v}_i \odot \boldsymbol{w} + \boldsymbol{b}, \tag{6}$$

where $\odot$ denotes element-wise multiplication, and $\boldsymbol{w}, \boldsymbol{b} \in \mathbb{R}^d$ is the trainable parameter, initialized as $\boldsymbol{w} = \mathbf{1}, \boldsymbol{b} = \mathbf{0}$. The text embeddings are given by $\boldsymbol{t}_0, \boldsymbol{t}_1 \in \mathbb{R}^d$. We omit the temperature parameter $\exp(t)$, as it can be absorbed into the text embeddings.

We examine how the parameter $\boldsymbol{w} = [w_1, \cdots, w_d]^\top$ changes after adaptation. Since scaling $\boldsymbol{w}$ by a constant does not affect the direction of the image embedding $\boldsymbol{z}_i$ or the prediction, we analyze the following normalized quantity, referred to as the *feature importance*:

$$r_h = \frac{|w_h|}{\sum_{l=1}^{d} |w_l|}. \tag{7}$$

A larger r_h indicates that the h-th feature contributes more to the prediction. Ideally, class-relevant features should have larger ratios, while domain-specific ones should have smaller ratios to ensure robustness to distribution shifts. Before adaptation, $\boldsymbol{w} = \mathbf{1}$, so all features have equal importance, i.e., $r_h = 1/d, \forall h$. The following Theorem 4.1 analyzes how adaptation changes these feature importance.

Theorem 4.1. *Perform one step of gradient descent on the support set $\mathbb{S}$ to minimize the prediction entropy. Assume the learning rate η is sufficiently small such that it does not change the sign of any element in $\boldsymbol{w}$, we have*

$$r_h = \frac{1 + \eta \cdot (\boldsymbol{e}_h \odot (\boldsymbol{t}_1 - \boldsymbol{t}_0))^\top \boldsymbol{M} (\boldsymbol{t}_1 - \boldsymbol{t}_0))}{d + \eta \cdot (\boldsymbol{t}_1 - \boldsymbol{t}_0)^\top \boldsymbol{M} (\boldsymbol{t}_1 - \boldsymbol{t}_0))}, \tag{8}$$

where $\boldsymbol{e}_h$ is the h-th standard basis (i.e., one-hot vector with 1 at the h-th position), and $\boldsymbol{M} = \mathbb{E}_{j \in \mathbb{S}}[(p_{j0} p_{j1}) \cdot \boldsymbol{v}_j \boldsymbol{v}_j^\top]$ denotes the probability-weighted uncentered covariance matrix (i.e., the second moment matrix) under the weighting $p_{j0} p_{j1}$. For better clarity, when $\boldsymbol{M}$ is a diagonal matrix, i.e., features are uncorrelated, the equation above can be simplified as

$$r_h = \frac{1 + \eta \cdot (t_{1h} - t_{0h})^2 \cdot M_{hh}}{d + \eta \cdot \sum_{l=1}^{d} (t_{1l} - t_{0l})^2 \cdot M_{ll}}, \tag{9}$$

where $M_{hh} = \mathbb{E}_{j \in \mathbb{S}}[(p_{j0} p_{j1}) \cdot v_{jh}^2]$ denotes the probability-weighted second moment of the h-th feature.

Remark 4.2 (Principle of TTA). Theorem 4.1 implies that entropy minimization amplifies the contribution of features whose $(t_{1h} - t_{0h})^2 \cdot M_{hh}$ is above average, while suppressing those with smaller values. In other words, it performs *feature reweighting* based on two factors:

- Larger $(t_{1h} - t_{0h})^2$, which indicates that the **text embeddings** of the two classes differ more along dimension h, meaning that *this feature is discriminative in the text description*;
- Larger $M_{hh} = \mathbb{E}[(p_{i0} p_{i1}) \cdot v_{ih}^2]$, which captures greater variance within the **image embeddings** in target domain, reflecting *distributional information of target domain*.

To build an intuition for this reweighting effect, we qualitatively discuss how features of different types may behave under this adaptation:

- **Class-relevant features.** These features vary significantly across samples from different classes. This leads to a large variance M_{hh}, which the model interprets as containing discriminative signals; such features are therefore amplified during adaptation.
- **Domain-relevant features.** Under mixed-domain shift, the dataset contains samples from multiple domains. As a result, a domain-relevant feature can also exhibit large variance M_{hh}. Therefore, domain-specific features may *not* be sufficiently suppressed and can even be mistakenly enhanced, leading to degraded performance.

Remark 4.3 (Effect of active sample selection). The two criteria in active sample selection can enhance the effectiveness of entropy minimization under mixed-domain shifts:

- **Domain consistency.** Retrieving samples that are close to the query within each pseudo-class ensures that class-irrelevant features, including domain-relevant ones, are more consistent within the constructed support set. As a result, their variance M_{hh} becomes smaller, allowing them to be more effectively suppressed.
- **Prediction balance.** Maintaining a balanced number of samples across pseudo-classes keeps the class-relevant features exhibiting large variance M_{hh}, so that these discriminative features are still amplified during adaptation.

5. Experiments

In this section, we conduct experiments to answer the following research questions:

- **RQ1**: Compared with existing TTA methods, is *Ramen* more robust under mixture-of-domain settings and able to achieve better performance?
- **RQ2**: What types of samples does *Ramen* retrieve as the support set for adaptation?
- **RQ3**: Does the design of *Ramen* ensure computational efficiency in practice?

Datasets Following the evaluation protocol of SAR [28], we conduct experiments under mixed distribution shifts on standard corruption benchmarks [17]: ViT-B/32 [10] on CIFAR-10-C [22], ViT-B/16 on CIFAR-100-C, and ViT-L/14 on ImageNet-C [7]. We further evaluate ViT-B/32 on DomainNet [32] to assess the performance of *Ramen* under mixture of domain shifts. Note that prior VLM studies [21, 36, 51] typically evaluate models under natural distribution shifts (e.g., ImageNet-V2, -A, -R, -Sketch) [18, 19, 35, 40] or cross-domain benchmarks (e.g., Flower102, Food101) [6]. However, these datasets each represent a single domain and are therefore not suitable for evaluating mixed-domain shifts.

Baselines We compare our method with a wide range of

Table 1. Mean accuracy (%) on corruption benchmarks under mixture of 15 corruptions. Best and second-best results are shown in **bold** and underlined, respectively. Error bars are reported in Appendix C.2.

Method	Venue	ViT-B/32 on CIFAR-10-C															
		Noise			Blur				Weather				Digital				Avg.
		Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	
CLIP [33]	ICML'21	35.5	40.0	43.2	70.0	41.4	64.5	70.2	70.8	72.3	66.7	81.4	64.5	59.6	48.2	56.7	59.0
Ensemble	-	38.6	42.6	42.7	72.4	43.9	66.6	71.6	73.8	75.7	69.0	83.6	67.0	61.9	51.8	58.6	61.3
Tent [39]	ICLR'21	39.5	43.1	46.4	77.8	56.6	72.7	77.9	79.7	79.9	74.0	86.9	75.5	69.7	60.3	63.9	66.9
NOTE [12]	NeurIPS'22	44.9	48.3	48.9	77.1	57.0	74.0	77.6	78.0	79.2	73.0	86.5	73.5	68.7	57.9	62.2	67.1
SAR [28]	ICLR'23	48.8	51.8	51.0	77.5	60.5	74.8	**79.2**	79.6	**80.7**	76.6	87.4	77.0	71.0	57.5	62.8	69.1
RoTTA [45]	CVPR'23	53.1	56.4	51.3	**78.1**	61.0	75.2	79.0	79.0	80.1	**80.0**	85.6	**82.9**	**73.7**	65.3	**69.9**	71.4
TDA [21]	CVPR'24	39.8	42.8	43.4	73.5	45.4	67.6	73.2	74.3	76.4	69.8	84.0	66.6	62.5	52.1	57.7	61.9
DMN-ZS [51]	CVPR'24	38.4	41.7	42.4	73.1	44.6	67.2	72.8	74.6	76.3	69.5	84.0	66.5	62.3	52.4	58.3	61.6
WATT-S [30]	NeurIPS'24	47.4	49.7	49.5	77.0	54.3	72.9	77.3	77.8	79.5	74.8	86.4	74.1	67.8	57.2	62.9	67.2
CLIPArTT [15]	WACV'25	37.2	41.5	44.8	70.5	48.9	65.5	70.7	72.9	75.3	67.4	82.8	66.9	61.9	58.5	59.5	61.6
Mint [3]	NeurIPS'25	47.9	50.9	50.3	74.1	51.3	75.8	76.9	76.4	77.0	73.8	85.8	73.8	65.8	47.3	56.4	65.6
Ramen	-	**61.0**	**63.3**	**56.0**	77.2	**64.1**	**77.2**	78.9	**80.9**	80.3	78.0	**88.1**	79.8	72.0	**65.5**	67.5	**72.7**

Method	Venue	ViT-B/16 on CIFAR-100-C															
		Noise			Blur				Weather				Digital				Avg.
		Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	
CLIP [33]	ICML'21	19.7	21.4	25.3	42.5	20.2	43.1	48.0	48.4	49.7	41.7	57.0	34.5	29.2	23.9	32.4	35.8
Ensemble	-	22.9	24.4	29.6	43.6	20.1	43.7	48.7	48.9	50.4	41.8	58.1	35.3	29.2	26.3	33.6	37.1
Tent [39]	ICLR'21	24.9	26.7	34.4	49.7	23.9	47.5	53.9	52.9	51.4	45.3	62.9	43.6	32.0	31.7	37.1	41.2
NOTE [12]	NeurIPS'22	25.7	27.3	35.4	49.9	23.9	47.6	54.2	52.7	51.8	45.9	63.7	44.0	32.4	30.2	36.0	41.4
SAR [28]	ICLR'23	28.4	30.5	38.5	50.4	24.6	49.2	55.1	54.1	53.4	47.2	64.0	45.0	33.6	29.7	37.4	42.7
RoTTA [45]	CVPR'23	22.9	24.5	32.4	47.9	25.4	46.5	50.1	50.5	50.7	50.8	58.3	**53.1**	**37.7**	29.9	36.2	41.1
TDA [21]	CVPR'24	23.4	25.2	30.3	44.1	20.3	43.8	49.1	49.3	51.3	42.2	58.7	36.1	29.3	26.4	33.6	37.5
DMN-ZS [51]	CVPR'24	22.9	24.4	29.6	44.3	20.2	44.1	49.4	49.7	51.1	42.2	58.8	35.5	29.4	26.2	34.0	37.5
WATT-S [30]	NeurIPS'24	26.4	28.4	35.6	50.2	24.2	48.5	54.4	54.2	54.0	47.7	63.5	43.3	34.0	31.4	37.7	42.2
CLIPArTT [15]	WACV'25	21.1	22.8	29.4	46.3	24.5	45.0	51.1	51.5	51.8	44.0	61.1	39.0	33.2	28.3	36.7	39.0
Mint [3]	NeurIPS'25	22.3	24.5	33.0	49.2	22.4	47.8	54.2	52.4	51.6	47.7	63.7	42.2	31.8	24.0	34.3	40.1
Ramen	-	**30.3**	**32.9**	**42.8**	**52.5**	**29.1**	**52.2**	**55.5**	**55.4**	**55.1**	**52.6**	**65.9**	51.0	36.8	**40.4**	**39.6**	**46.1**

Method	Venue	ViT-L/14 on ImageNet-C															
		Noise			Blur				Weather				Digital				Avg.
		Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	
CLIP [33]	ICML'21	27.4	29.4	28.7	34.6	25.3	41.0	36.7	49.8	44.1	49.7	65.4	35.1	30.3	53.5	42.2	39.6
Ensemble	-	29.1	30.4	30.1	37.5	27.3	44.2	39.2	52.4	46.4	52.6	67.8	34.4	32.4	56.2	44.2	41.6
Tent [39]	ICLR'21	36.3	38.0	38.5	40.3	37.4	47.7	43.5	54.2	49.5	56.4	67.8	45.4	40.6	57.6	46.6	46.7
NOTE [12]	NeurIPS'22	36.3	38.0	38.6	40.0	37.3	47.6	43.5	54.4	49.7	56.4	67.8	44.5	40.8	57.5	46.2	46.6
SAR [28]	ICLR'23	36.3	38.3	38.5	39.2	38.5	46.8	43.4	53.9	50.4	56.6	66.8	45.5	41.8	56.2	46.6	46.6
RoTTA [45]	CVPR'23	37.4	38.7	39.6	36.9	34.5	44.0	39.3	53.5	48.6	54.2	67.2	43.0	39.7	56.3	48.4	45.4
TDA [21]	CVPR'24	29.6	31.0	31.5	38.1	28.9	44.6	40.0	53.4	47.5	53.3	68.5	39.1	33.4	57.1	45.1	42.7
DMN-ZS [51]	CVPR'24	29.2	30.5	30.2	37.6	27.6	44.4	39.2	52.5	46.6	52.7	67.9	33.8	32.5	56.5	44.5	41.7
WATT-S [30]	NeurIPS'24	31.8	33.1	33.6	39.2	30.7	45.7	41.3	53.5	47.3	53.7	67.9	40.1	34.7	57.3	45.6	43.7
CLIPArTT [15]	WACV'25	28.2	30.2	29.8	35.0	26.5	41.4	37.3	50.3	43.7	49.7	65.0	38.5	31.0	53.5	42.3	40.2
Mint [3]	NeurIPS'25	33.8	35.2	35.8	39.3	33.3	46.2	41.5	53.6	47.4	53.3	67.7	36.3	38.2	58.1	47.4	44.5
Ramen	-	**37.5**	**38.8**	**41.1**	**42.2**	**39.6**	**49.9**	**46.2**	**57.3**	**51.6**	**59.4**	**68.7**	**46.4**	**45.2**	**59.4**	**54.7**	**49.2**

baselines. For generic TTA methods, we include Tent [39], NOTE [12], SAR [28], and RoTTA [45]. Among them, SAR is specifically designed for wild test scenarios, including mixed-domain settings, while NOTE and RoTTA also employ prediction-balanced memory banks but do not explicitly account for domain mixtures. For VLM-based TTA methods, we consider two memory-based approaches, TDA [21] and DMN-ZS [51], as well as three recent and highly competitive baselines: WATT-S [30], CLIPArTT [15], and Mint [3]. CLIP and Ensemble represent the zero-shot performance of CLIP using a single template ("`a photo of a {class}`") and seven templates from [49], respectively. For all TTA methods, except from CLIPArTT which modifies the prompts, also use the seven templates. More details of experiments setup, including hyperparameter, are provided in Appendix C.1.

Table 2. Mean accuracy (%) on DomainNet under mixture of six domains. Best and second-best results are shown in **bold** and underlined, respectively. Error bars are reported in Appendix C.2.

Method	Venue	ViT-B/32 on DomainNet						
		clip	info	paint	quick	real	sketch	Avg.
CLIP [33]	ICML'21	67.6	41.5	62.7	12.8	81.0	58.1	54.0
Ensemble	-	68.9	44.2	64.7	13.3	82.3	60.3	55.6
Tent [39]	ICLR'21	69.1	44.3	64.9	13.0	82.3	60.3	55.7
NOTE [12]	NeurIPS'22	69.6	45.1	65.0	16.4	82.1	60.8	56.5
SAR [28]	ICLR'23	69.4	44.7	65.2	14.5	82.5	60.8	56.2
RoTTA [45]	CVPR'23	69.4	44.8	65.3	14.3	82.4	60.8	56.2
TDA [21]	CVPR'24	68.9	44.7	65.1	14.6	82.3	60.5	56.0
DMN-ZS [51]	CVPR'24	68.9	44.6	64.8	12.7	**82.6**	60.2	55.6
WATT-S [30]	NeurIPS'24	68.3	42.8	63.5	17.0	81.4	59.9	55.5
CLIPArTT [15]	WACV'25	67.2	41.9	62.9	12.6	80.8	57.8	53.9
Mint [3]	NeurIPS'25	69.3	44.4	65.0	15.0	82.3	60.5	56.1
Ramen	-	**70.1**	**45.2**	**65.8**	**17.6**	82.4	**61.5**	**57.1**

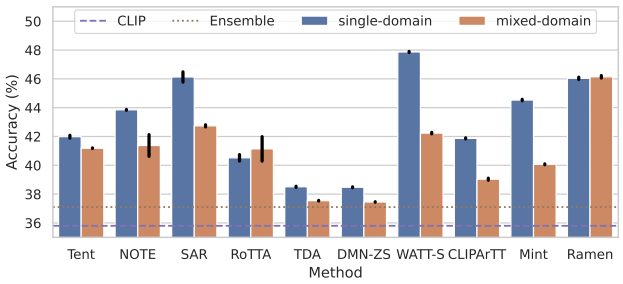


Figure 2. Performance comparison of TTA methods under single-domain and mixed-domain shifts on CIFAR-100-C. Results on other datasets are provided in Appendix C.3.

Main results (RQ1) We evaluate all methods under mixed-domain settings, where test samples from multiple domains are fully interleaved during adaptation rather than evaluated separately. Following prior works [3, 30], we report the accuracy for each domain individually and then average them across all domains for a fair comparison. The results are summarized in Table 1 and Table 2. Overall, *Ramen* consistently achieves the best performance across all datasets and architectures. Specifically, it improves the average accuracy by +1.3% on CIFAR-10-C, +3.4% on CIFAR-100-C, +2.5% on ImageNet-C, and +0.6% on DomainNet compared with the strongest baseline.

Robustness to domain mixture (RQ1) We further compare the performance of different methods under both single-domain and mixed-domain settings. In the single-domain evaluation, the model is tested on each domain separately without mixing test samples, and we report the average accuracy across all domains. The results are presented in Figure 2. We observe that existing baseline methods, especially recent VLM-based TTA approaches, perform well under the single-domain setting. However, most of them suffer from significant performance drops when domains are mixed, while earlier methods are also affected to a lesser extent. In contrast, *Ramen* maintains consistently strong performance across both settings.

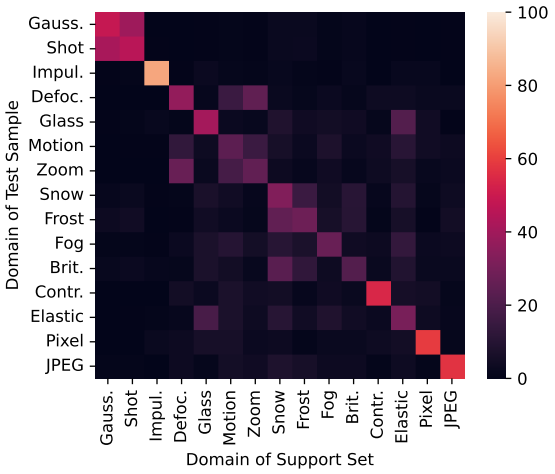


Figure 3. Visualization of active sample selection on CIFAR-100-C. Each entry (i, j) indicates the average proportion (%) of support samples from domain j when the test sample comes from domain i. On average, 40.9% of the support samples come from the same domain as the test sample, which is substantially higher than random selection (6.7%). Results on other datasets are provided in Appendix C.4.

Visualization of active sample selection (RQ2) To further understand the mechanism of *Ramen*'s active sample selection, we analyze the domain composition of its retrieved samples. In Figure 3, each cell in row i and column j represents the proportion of samples from domain j within the customized support set when the new test sample (query) comes from domain i. Hence, the diagonal elements indicate the frequency of retrieving samples from the same domain. We observe that on average, 40.9% of the retrieved support samples come from the same domain as the query test sample, which is substantially higher than random selection (6.7%). This indicates that *Ramen*'s active sample selection effectively captures domain consistency, allowing the model to preferentially retrieve samples from similar domains and thereby perform more targeted adaptation.

Efficiency (RQ3) We measure the testing time of *Ramen* and baseline methods on the CIFAR-100-C dataset (150,000 images) to evaluate their efficiency. As shown in Table 3, the embedding-gradient cache in *Ramen* substantially reduces the computational cost, achieving a 490× speedup compared to the naive implementation. As a result, the overall computation time of *Ramen* is comparable to existing model-updating TTA methods, while delivering superior performance. We further analyze the *GPU memory usage* of *Ramen* in Appendix C.5.

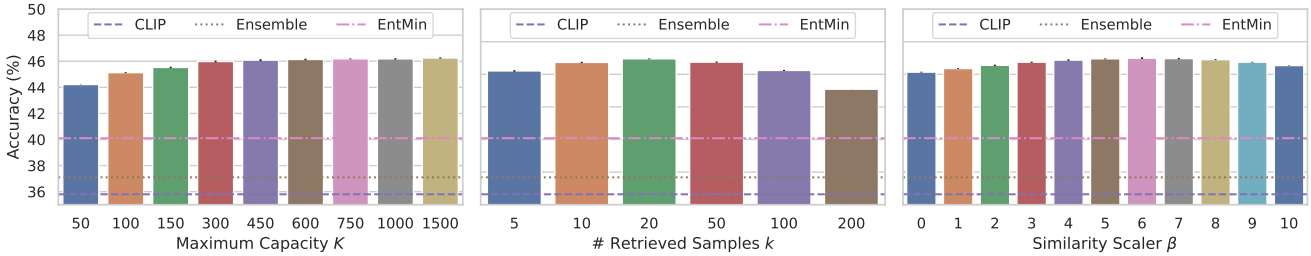


Figure 4. Hyperparameter sensitivity on CIFAR-100-C. (EntMin refers to entropy minimization without *Ramen*.)

Table 3. Comparison of testing time on CIFAR-100-C.

Method	Testing Time	Acc. (%)	Gain (%)
CLIP	5m27s	35.8	-
Tent	9m27s	41.2	+5.4
NOTE	11m42s	41.4	+5.6
SAR	15m50s	42.7	+6.9
RoTTA	19m29s	41.1	+5.3
TDA	7m59s	37.5	+1.7
DMN-ZS	7m36s	37.5	+1.7
WATT-S	18h54m50s	42.2	+6.4
CLIPArTT	3h05m36s	39.0	+3.2
Mint	13m36s	40.1	+4.3
Ramen	14m08s	46.1	+10.3
(w/o embed-grad cache)	115h42m18s	46.1	+10.3

Hyperparameter sensitivity In Figure 4, we examine the sensitivity of *Ramen* to its hyperparameters. The maximum capacity K controls the range of active sample selection: a larger K provides a richer candidate pool and generally improves performance, though with diminishing returns once K becomes sufficiently large. The retrieval size k determines how many samples are selected for adaptation. As k increases, the proportion of samples from different domains in the support set tends to grow, which weakens domain consistency; when k is too small, the gradients are computed from too few samples and thus can be noisy. The similarity scaler β plays a similar role to k. However, across a wide range of hyperparameter settings, *Ramen* consistently improves TTA performance compared to vanilla entropy minimization (MinEnt), demonstrating its robustness to hyperparameter choices. Beyond the hyperparameters introduced by *Ramen*, we also evaluate *Ramen* under different learning rates and batch sizes, and observe consistent performance gains across all configurations. Appendix C.6 provides full results of hyperpararameter sensitivity.

Ablation study To further understand the effects of the two criteria in active sample selection, we conduct an ablation study with the following variants:

- *Without prediction balance (w/o PB)*: Instead of maintaining C FIFO queues (one for each class) with capacity K and retrieving the top-k samples from each, we maintain a single FIFO queue without distinguishing classes, with capacity $C \cdot K$, and retrieve the top $C \cdot k$ samples.
- *Without domain consistency (w/o DC)*: We no longer select the top-k most similar samples. Instead, k samples are randomly chosen from the queue, and we set $\beta = 0$.
- *Without both prediction balancing and domain consistency (w/o PB & DC)*: This variant approximately corresponds to randomly selecting a subset of previously seen samples for model updates.

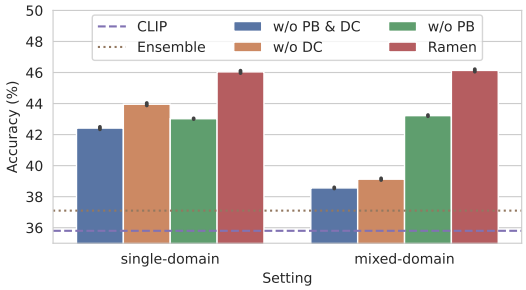


Figure 5. Ablation study. *DC*: domain consistency, *PB*: prediction balance. Both rules are beneficial to the performance of *Ramen*.

As shown in Figure 5, both the domain consistency and prediction balance criteria contribute to the overall performance of *Ramen*. Specifically, domain consistency makes *Ramen* perform more consistently across both single-domain and mixed-domain settings, while prediction balance provides steady performance gains in both settings. In Appendix C.7, we further conduct an ablation study on the two components of the gradient aggregation weights: entropy and similarity. Results show that both are beneficial.

6. Conclusion

We present *Ramen*, a framework for robust test-time adaptation under mixed-domain shifts. By actively selecting domain-consistent and prediction-balanced samples, it achieves stable and unbiased adaptation. With an efficient embedding-gradient cache, it enables fast updates without extra computation. Theoretical and empirical results demonstrate its robustness and efficiency.

Acknowledgement

This work is supported by National Science Foundation under Award No. IIS-2416070, IIS-2117902. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

References

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. 4, 14, 15

[2] Wenxuan Bao, Tianxin Wei, Haohan Wang, and Jingrui He. Adaptive test-time personalization for federated learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. 13

[3] Wenxuan Bao, Ruxi Deng, and Jingrui He. Mint: A simple test-time adaptation of vision-language models against common corruptions. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2 - 7, 2025*, 2025. 2, 4, 6, 7, 15, 17, 18, 24

[4] Wenxuan Bao, Ruxi Deng, Ruizhong Qiu, Tianxin Wei, Hanghang Tong, and Jingrui He. Latte: Collaborative test-time adaptation of vision-language models in federated learning. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, Hawaii, USA, October 19-23, 2025*. IEEE, 2025. 4, 13

[5] Wenxuan Bao, Zhichen Zeng, Zhining Liu, Hanghang Tong, and Jingrui He. Matcha: Mitigating graph structure shifts with test-time adaptation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. 13

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - mining discriminative components with random forests. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI*, pages 446–461. Springer, 2014. 5

[7] 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 Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009. 5

[8] Ruxi Deng, Wenxuan Bao, Tianxin Wei, and Jingrui He. Panda: Test-time adaptation with negative data augmentation. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 3551–3559. AAAI Press, 2026. 15

[9] Mario Döbler, Robert A. Marsden, Tobias Raichle, and Bin Yang. A lost opportunity for vision-language models: A comparative study of online test-time adaptation for vision-language models. In *Computer Vision - ECCV 2024 Workshops - Milan, Italy, September 29-October 4, 2024, Proceedings, Part XVIII*, pages 117–133. Springer, 2024. 1, 2, 3, 17

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 5

[11] Matteo Farina, Gianni Franchi, Giovanni Iacca, Massimiliano Mancini, and Elisa Ricci. Frustratingly easy test-time adaptation of vision-language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. 2

[12] Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. NOTE: robust continual test-time adaptation against temporal correlation. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. 2, 4, 6, 7, 15, 17, 18, 24

[13] Taesik Gong, Yewon Kim, Taeckyung Lee, Sorn Chottananurak, and Sung-Ju Lee. Sotta: Robust test-time adaptation on noisy data streams. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. 13

[14] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1, 2, 17

[15] Gustavo Adolfo Vargas Hakim, David Osowiechi, Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Moslem Yazdanpanah, Ismail Ben Ayed, and Christian Desrosiers. Clipartt: Adaptation of CLIP to new domains at test time. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2025, Tucson, AZ, USA, February 26 - March 6, 2025*, pages 7092–7101. IEEE, 2025. 2, 4, 6, 7, 15, 17, 18, 24

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. 24

[17] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 1, 2, 5, 17

[18] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt,

and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 8320–8329. IEEE, 2021. 5

[19] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 15262–15271. Computer Vision Foundation / IEEE, 2021. 5

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456. JMLR.org, 2015. 2, 4, 15

[21] Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El-Saddik, and Eric P. Xing. Efficient test-time adaptation of vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 14162–14171. IEEE, 2024. 2, 4, 5, 6, 7, 13, 17, 18, 24

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[23] Jonghyun Lee, Dahuin Jung, Saehyung Lee, Junsung Park, Juhyeon Shin, Uiwon Hwang, and Sungroh Yoon. Entropy is not enough for test-time adaptation: From the perspective of disentangled factors. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. 2, 4

[24] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *Int. J. Comput. Vis.*, 133(1):31–64, 2025. 1

[25] Robert A. Marsden, Mario Döbler, and Bin Yang. Universal test-time adaptation through weight ensembling, diversity weighting, and prior correction. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2024, Waikoloa, HI, USA, January 3-8, 2024*, pages 2543–2553. IEEE, 2024. 1, 2, 3

[26] Saypraseuth Mounsaveng, Florent Chiaroni, Malik Boudiaf, Marco Pedersoli, and Ismail Ben Ayed. Bag of tricks for fully test-time adaptation. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2024, Waikoloa, HI, USA, January 3-8, 2024*, pages 1925–1934. IEEE, 2024. 3

[27] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, pages 16888–16905. PMLR, 2022. 1, 2, 3, 4

[28] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. 1, 2, 3, 4, 5, 6, 7, 13, 15, 17, 18, 24

[29] Shuaicheng Niu, Guohao Chen, Deyu Chen, Yifan Zhang, Jiaxiang Wu, Zhiquan Wen, Yaofo Chen, Peilin Zhao, Chunyan Miao, and Mingkui Tan. Adapt in the wild: Test-time entropy minimization with sharpness and feature regularization. *CoRR*, abs/2509.04977, 2025. 1, 3, 13

[30] David Osowiechi, Mehrdad Noori, Gustavo Adolfo Vargas Hakim, Moslem Yazdanpanah, Ali Bahri, Milad Cheraghalikhani, Sahar Dastani, Farzad Beizaee, Ismail Ben Ayed, and Christian Desrosiers. WATT: weight average test time adaptation of CLIP. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. 2, 4, 6, 7, 15, 17, 18, 24

[31] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 2065–2074. IEEE, 2021. 1

[32] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1406–1415. IEEE, 2019. 5, 17

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 8748–8763. PMLR, 2021. 1, 2, 6, 7, 17, 18, 24

[34] Yongming Rao, Wenliang Zhao, Guangyi Chen, Yansong Tang, Zheng Zhu, Guan Huang, Jie Zhou, and Jiwen Lu. Denseclip: Language-guided dense prediction with context-aware prompting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 18061–18070. IEEE, 2022. 1

[35] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5389–5400. PMLR, 2019. 5

[36] Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. 2, 5

[37] Elaine Sui, Xiaohan Wang, and Serena Yeung-Levy. Just shift it: Test-time prototype shifting for zero-shot generalization with vision-language models. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2025, Tucson, AZ, USA, February 26 - March 6, 2025*, pages 825–835. IEEE, 2025. 2

[38] Devavrat Tomar, Guillaume Vray, Jean-Philippe Thiran, and Behzad Bozorgtabar. Un-mixing test-time normalization

statistics: Combatting label temporal correlation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. 1, 2, 3

[39] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 2, 4, 6, 7, 13, 15, 17, 18, 24

[40] Haohan Wang, Songwei Ge, Zachary C. Lipton, and Eric P. Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10506–10518, 2019. 5

[41] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 7191–7201. IEEE, 2022. 2, 13

[42] Zixin Wang, Yadan Luo, Liang Zheng, Zhuoxiao Chen, Sen Wang, and Zi Huang. In search of lost online test-time adaptation: A survey. *Int. J. Comput. Vis.*, 133(3):1106–1139, 2025. 1

[43] Yuxin Wu and Kaiming He. Group normalization. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, pages 3–19. Springer, 2018. 4, 15

[44] Zehao Xiao and Cees G. M. Snoek. Beyond model adaptation at test time: A survey. *CoRR*, abs/2411.03687, 2024. 1

[45] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 15922–15932. IEEE, 2023. 1, 2, 6, 7, 13, 17, 18, 24

[46] Maxime Zanella, Clément Fuchs, Christophe De Vleeschouwer, and Ismail Ben Ayed. Realistic test-time adaptation of vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 25103–25112. Computer Vision Foundation / IEEE, 2025. 13

[47] Zhichen Zeng, Wenxuan Bao, Xiao Lin, Ruizhong Qiu, Tianxin Wei, Xuying Ning, Yuchen Yan, Chen Luo, Monica Xiao Cheng, Jingrui He, and Hanghang Tong. Subspace alignment for vision-language model test-time adaptation. *CoRR*, abs/2601.08139, 2026. 15

[48] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12360–12371, 2019. 4, 15

[49] Renrui Zhang, Wei Zhang, Rongyao Fang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free adaption of CLIP for few-shot classification. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXV*, pages 493–510. Springer, 2022. 6, 17

[50] Yifan Zhang, Xue Wang, Kexin Jin, Kun Yuan, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. Adanpc: Exploring non-parametric classifier for test-time adaptation. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, pages 41647–41676. PMLR, 2023. 2

[51] Yabin Zhang, Wenjie Zhu, Hui Tang, Zhiyuan Ma, Kaiyang Zhou, and Lei Zhang. Dual memory networks: A versatile adaptation approach for vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 28718–28728. IEEE, 2024. 2, 4, 5, 6, 7, 13, 17, 18, 24

[52] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, and Jianfeng Gao. Regionclip: Region-based language-image pretraining. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 16772–16782. IEEE, 2022. 1

[53] Lihua Zhou, Mao Ye, Shuaifeng Li, Nianxin Li, Xiatian Zhu, Lei Deng, Hongbin Liu, and Zhen Lei. Bayesian test-time adaptation for vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 29999–30009. Computer Vision Foundation / IEEE, 2025. 13

Ramen: Robust Test-Time Adaptation of Vision-Language Models with Active Sample Selection

Supplementary Material

Contents

A Discussion and Details **13**
- A.1. More Related Works . . . 13
- A.2. Empirical Validation of Domain Consistency . . . 13
- A.3. Reparameterization Trick . . . 14

B Missing Proofs **15**

C Additional Experiments **17**
- C.1. Experimental Setup . . . 17
- C.2. Error Bars (RQ1) . . . 18
- C.3. Comparison of Single-Domain and Mixed-Domain Shifts (RQ1) . . . 19
- C.4. Visualization of Active Sample Selection (RQ2) . . . 20
- C.5. Memory Usage (RQ3) . . . 21
- C.6. Hyperparameter Sensitivity . . . 22
- C.7. Ablation Study . . . 24
- C.8. More Architectures . . . 24

A. Discussion and Details

A.1. More Related Works

Differet variants of TTA settings

- *Single-domain TTA* [5, 39]: the most common TTA setting, where a pre-trained model is adapted to a single, consistent domain with a stable distribution.
- *Mixed-domain TTA* [28, 29]: the main setting studied in this paper, where test samples from multiple domains are fully mixed, and the domain identity of each sample is unknown during testing.
- *Multi-target TTA* [2, 4]: assumes that a pre-trained model is adapted to a series of related but distinct domains, where domain relationships can be leveraged to improve performance. Unlike mixed-domain TTA, the domain label of each sample is known.
- *Continual TTA* [41, 45]: assumes that the model is adapted to a sequence of domains, simulating gradual distribution shifts over time. Although domain IDs are unavailable, domains exhibit temporal continuity—consecutive samples typically come from the same domain.
- *Noisy TTA* [13]: assumes adaptation to a single domain, but the test data may contain noisy or out-of-distribution samples that are not drawn from the target distribution. This setting resembles mixed-domain TTA, but the noisy samples may not belong to any of the task classes, and their accuracy is not of interest.

Comparison to existing memory-based methods Existing memory-based approaches can be broadly categorized into two types: methods such as TDA [21] and DMN-ZS [51] that directly store embeddings of previously seen test samples, and methods such as StatA [46] and BCA [53] that maintain statistics computed from these embeddings. *Ramen* differs fundamentally from both categories in the following aspects.

- *What is cached.* Prior methods either store forward-pass embeddings or update summary statistics based on these embeddings. However, such designs do not support gradient-based model updates, and therefore cannot effectively address the mismatch between the visual encoder and the target domain, limiting their adaptation potential. In contrast, *Ramen* caches both embeddings and gradients, enabling model updates and leading to greater adaptation gains.
- *How the cache is used.* Existing methods typically apply a shared model to all test samples. For example, TDA uses the same cached embeddings regardless of the current input. Notably, while the cache itself may evolve over time, it is not conditioned on the specific test sample. In contrast, *Ramen* performs active, per-sample retrieval to construct a support set tailored to each input, resulting in more robust and effective adaptation.

A.2. Empirical Validation of Domain Consistency

In Subsection 3.2, we claimed that *the more similar two image embeddings are, the more likely they originate from the same domain*. Here, we provide empirical validation.

We conduct a study on CIFAR-100-C by computing pairwise embedding similarities, grouping sample pairs into 10 bins from high to low similarity (top 10%, 10%–20%, . . ., bottom 10%), and measuring the fraction of same-domain pairs in each bin. Results are summarized in Table 4.

We observe that higher similarity consistently corresponds to a higher same-domain ratio, supporting our claim.

Table 4. Empirical validation of embedding similarity for domain consistency.

Similarity bin (high → low)	1	2	3	4	5	6	7	8	9	10
Same-domain ratio (%)	40.1	16.9	12.3	10.0	8.6	7.6	6.8	6.2	5.6	4.5

A.3. Reparameterization Trick

In the main text, we describe our algorithm based on a single test sample, where computing the sample-level gradient is straightforward. However, when the batch size $B > 1$, performing forward and backward propagation for each sample individually has low parallelism and is thus inefficient. To address this, we adopt a reparameterization trick that enables more efficient computation of sample-level gradients.

Take LayerNorm [1] as an example, where the input has a shape of $L \times B \times D$, with L being the sequence length, B the batch size, and D the hidden dimension. The affine parameters (weight and bias) normally have a dimension of D. In standard backpropagation, the gradient is averaged over the B samples. To obtain per-sample gradients, we replicate the affine parameters B times, resulting in a parameter dimension of $B \times D$, and change the reduction function in the entropy loss from mean to sum. During backpropagation, the resulting gradient also has a dimension of $B \times D$, corresponding directly to the gradient of each individual sample, since each sample's forward pass is computed independently with its own replicated affine parameters.

This approach is significantly more efficient than computing gradients one by one, because it leverages parallel computation in existing deep learning frameworks. Specifically, all samples share the same forward and backward computation graph, so their gradients can be computed in a single backward pass without serializing B independent backward calls.

In addition, when the batch size is not excessively large, this reparameterization does not lead to a noticeable increase in memory usage. The majority of the computation graph remains unchanged, since only the LayerNorm affine parameters are duplicated, and these parameters account for a negligible fraction of the total model parameters (e.g., $< 0.05\%$ on ViT-B/16).

B. Missing Proofs

Setup and assumptions Most TTA methods [3, 8, 12, 15, 28, 30, 39, 47] adapt models by updating the affine parameters in normalization layers. Therefore, we focus our analysis on this component. Common normalization layers [1, 20, 43, 48] share a same structure: a normalization step followed by an element-wise affine/linear transformation. Although the normalization step differs across layer types, the parameterization of the affine/linear transformation remains similar. Following [3], we consider a single normalization layer in a binary classification setting. Let $\boldsymbol{v}_i \in \mathbb{R}^d$ denote the intermediate normalized feature, obtained after the normalization step but before applying its affine transformation. The final image embedding is given by an element-wise linear transformation:

$$\boldsymbol{z}_i = \boldsymbol{v}_i \odot \boldsymbol{w} + \boldsymbol{b}, \tag{10}$$

where $\odot$ denotes element-wise multiplication, and $\boldsymbol{w}, \boldsymbol{b} \in \mathbb{R}^d$ is the trainable parameter, initialized as $\boldsymbol{w} = \mathbf{1}, \boldsymbol{b} = \mathbf{0}$. The text embeddings are given by $\boldsymbol{t}_0, \boldsymbol{t}_1 \in \mathbb{R}^d$. We omit the temperature parameter $\exp(t)$, as it can be absorbed into the text embeddings.

We examine how the parameter $\boldsymbol{w} = [w_1, \cdots, w_d]^\top$ changes after adaptation. Since scaling $\boldsymbol{w}$ by a constant does not affect the direction of the image embedding $\boldsymbol{z}_i$ or the prediction, we analyze the following normalized quantity, referred to as the *feature importance*:

$$r_h = \frac{|w_h|}{\sum_{l=1}^{d} |w_l|}. \tag{11}$$

A larger r_h indicates that the h-th feature contributes more to the prediction. Ideally, class-relevant features should have larger ratios, while domain-specific ones should have smaller ratios to ensure robustness to distribution shifts. Before adaptation, $\boldsymbol{w} = \mathbf{1}$, so all features have equal importance, i.e., $r_h = 1/d, \forall h$. The following Theorem 4.1 analyzes how adaptation changes these feature importance.

Theorem 4.1. *Perform one step of gradient descent on the support set $\mathbb{S}$ to minimize the prediction entropy. Assume the learning rate η is sufficiently small such that it does not change the sign of any element in $\boldsymbol{w}$, we have*

$$r_h = \frac{1 + \eta \cdot (\boldsymbol{e}_h \odot (\boldsymbol{t}_1 - \boldsymbol{t}_0))^\top \boldsymbol{M} (\boldsymbol{t}_1 - \boldsymbol{t}_0))}{d + \eta \cdot (\boldsymbol{t}_1 - \boldsymbol{t}_0)^\top \boldsymbol{M} (\boldsymbol{t}_1 - \boldsymbol{t}_0))}, \tag{8}$$

where $\boldsymbol{e}_h$ is the h-th standard basis (i.e., one-hot vector with 1 at the h-th position), and $\boldsymbol{M} = \mathbb{E}_{j \in \mathbb{S}}[(p_{j0} p_{j1}) \cdot \boldsymbol{v}_j \boldsymbol{v}_j^\top]$ denotes the probability-weighted uncentered covariance matrix (i.e., the second moment matrix) under the weighting $p_{j0} p_{j1}$. For better clarity, when $\boldsymbol{M}$ is a diagonal matrix, i.e., features are uncorrelated, the equation above can be simplified as

$$r_h = \frac{1 + \eta \cdot (t_{1h} - t_{0h})^2 \cdot M_{hh}}{d + \eta \cdot \sum_{l=1}^{d} (t_{1l} - t_{0l})^2 \cdot M_{ll}}, \tag{9}$$

where $M_{hh} = \mathbb{E}_{j \in \mathbb{S}}[(p_{j0} p_{j1}) \cdot v_{jh}^2]$ denotes the probability-weighted second moment of the h-th feature.

Proof. In the forward pass, for each image $j \in \mathbb{S}$, its logits $\boldsymbol{l}_j = [l_{j0}, l_{j1}]^\top$, predicted probabilities $\boldsymbol{p}_j = [p_{j0}, p_{j1}]^\top$, and entropy H_j are given by

$$\begin{aligned} l_{jc} &= \boldsymbol{z}_j^\top \boldsymbol{t}_c, \quad c \in \{0, 1\}, \\ p_{jc} &= \frac{\exp(l_{jc})}{\exp(l_{j0}) + \exp(l_{j1})}, \quad c \in \{0, 1\}, \\ H_j &= -\sum_{c \in \{0,1\}} p_{jc} \log p_{jc}. \end{aligned}$$

And the total entropy on the support set $\mathbb{S}$ is

$$H = \mathbb{E}_{j \in \mathbb{S}} H_i.$$

We then compute the back-propagation. For each sample j,

$$\begin{aligned}\frac{\partial H_j}{\partial l_{j0}} &= \frac{\partial H_j}{\partial p_{j0}} \cdot \frac{\partial p_{j0}}{\partial l_{j0}} + \frac{\partial H_j}{\partial p_{j1}} \cdot \frac{\partial p_{j1}}{\partial l_{j0}} \\ &= -(\log p_{j0} + 1) \cdot (p_{j0}p_{j1}) - (\log p_{j1} + 1) \cdot (-p_{j0}p_{j1}) \\ &= (\log p_{j1} - \log p_{j0}) \cdot (p_{j0}p_{j1}) \\ &= (l_{j1} - l_{j0}) \cdot (p_{j0}p_{j1}).\end{aligned}$$

By symmetry,

$$\frac{\partial H_j}{\partial l_{j1}} = (l_{j0} - l_{j1}) \cdot (p_{j0}p_{j1}).$$

The gradient w.r.t. $\boldsymbol{z}_j$ is

$$\begin{aligned}\frac{\partial H_j}{\partial \boldsymbol{z}_j} &= \frac{\partial H_j}{\partial l_{j0}} \cdot \frac{\partial l_{j0}}{\partial \boldsymbol{z}_j} + \frac{\partial H_j}{\partial l_{j1}} \cdot \frac{\partial l_{j1}}{\partial \boldsymbol{z}_j} \\ &= (l_{j1} - l_{j0}) \cdot (p_{j0}p_{j1}) \cdot \boldsymbol{t}_0 + (l_{j0} - l_{j1}) \cdot (p_{j0}p_{j1}) \cdot \boldsymbol{t}_1 \\ &= -(l_{j1} - l_{j0}) \cdot (p_{j0}p_{j1}) \cdot (\boldsymbol{t}_1 - \boldsymbol{t}_0)\end{aligned}$$

Therefore, the gradient w.r.t. $\boldsymbol{w}$ is

$$\begin{aligned}\frac{\partial H_j}{\partial \boldsymbol{w}} &= \frac{\partial \boldsymbol{z}_j}{\partial \boldsymbol{w}} \frac{\partial H_j}{\partial \boldsymbol{z}_j} \\ &= \frac{\partial H_j}{\partial \boldsymbol{z}_j} \odot \boldsymbol{v}_j \\ &= -(l_{j1} - l_{j0}) \cdot (p_{j0}p_{j1}) \cdot (\boldsymbol{t}_1 - \boldsymbol{t}_0) \odot \boldsymbol{v}_j \\ &= -(p_{j0}p_{j1}) \cdot (\boldsymbol{t}_1 - \boldsymbol{t}_0) \odot \boldsymbol{v}_j (\boldsymbol{z}_j^\top \boldsymbol{t}_1 - \boldsymbol{z}_j^\top \boldsymbol{t}_0) \\ &= -(p_{j0}p_{j1}) \cdot (\boldsymbol{t}_1 - \boldsymbol{t}_0) \odot \boldsymbol{v}_j (\boldsymbol{v}_j^\top \boldsymbol{t}_1 - \boldsymbol{v}_j^\top \boldsymbol{t}_0) && (\boldsymbol{w} = \mathbf{1}, \boldsymbol{b} = \mathbf{0}) \\ &= -(p_{j0}p_{j1}) \cdot (\boldsymbol{t}_1 - \boldsymbol{t}_0) \odot \boldsymbol{v}_j \boldsymbol{v}_j^\top (\boldsymbol{t}_1 - \boldsymbol{t}_0) \\ &= -(p_{j0}p_{j1}) \cdot \operatorname{diag}(\boldsymbol{t}_1 - \boldsymbol{t}_0) \boldsymbol{v}_j \boldsymbol{v}_j^\top (\boldsymbol{t}_1 - \boldsymbol{t}_0) \\ &= -\operatorname{diag}(\boldsymbol{t}_1 - \boldsymbol{t}_0) \left[(p_{j0}p_{j1}) \boldsymbol{v}_j \boldsymbol{v}_j^\top\right] (\boldsymbol{t}_1 - \boldsymbol{t}_0) \\ \frac{\partial H}{\partial \boldsymbol{w}} &= \mathbb{E}_{j \in \mathbb{S}} \frac{\partial H_j}{\partial \boldsymbol{w}} \\ &= -\operatorname{diag}(\boldsymbol{t}_1 - \boldsymbol{t}_0) \mathbb{E}_{j \in \mathbb{S}} \left[(p_{j0}p_{j1}) \boldsymbol{v}_j \boldsymbol{v}_j^\top\right] (\boldsymbol{t}_1 - \boldsymbol{t}_0)\end{aligned}$$

Therefore, when we conduct gradient descent with learning rate η,

$$\boldsymbol{w} \leftarrow \mathbf{1} + \eta \operatorname{diag}(\boldsymbol{t}_1 - \boldsymbol{t}_0) \mathbb{E}_{j \in \mathbb{S}} \left[(p_{j0}p_{j1}) \boldsymbol{v}_j \boldsymbol{v}_j^\top\right] (\boldsymbol{t}_1 - \boldsymbol{t}_0).$$

Besides, the gradient w.r.t. $\boldsymbol{b}$ is expected to be small:

$$\frac{\partial H}{\partial \boldsymbol{b}} = \mathbb{E}_{j \in \mathbb{S}} \frac{\partial H_j}{\partial \boldsymbol{z}_j} = -\left(\mathbb{E}_{j \in \mathbb{S}} (l_{j1} - l_{j0}) \cdot (p_{j0}p_{j1})\right) \cdot (\boldsymbol{t}_1 - \boldsymbol{t}_0).$$

When the retrieved samples are class-balanced, we have

$$\mathbb{E}_{j \in \mathbb{S}} (l_{j1} - l_{j0}) \cdot (p_{j0}p_{j1}) \approx 0,$$

and thus the gradient w.r.t. bias becomes negligible. □

C. Additional Experiments

C.1. Experimental Setup

Compute resources All of our experiments are conducted on single NVIDIA Tesla V100 with 32GB memory, except for experiments on large batch size are conducted on single NVIDIA Tesla A100 with 80GB memory.

Datasets For image corruption benchmarks (CIFAR-10-C, CIFAR-100-C, ImageNet-C), we use the data provided in [17], following the implementation in [9]. For DomainNet [32], we use the implementation provided by DomainBed [14].

Pre-trained models We use the pre-trained models provided in the original CLIP repository [33]. We use batch size $B = 100$ for ViT-B/32 and ViT-B/16, and $B = 50$ for ViT-L/14 to save GPU memory usage.

Text embeddings For all methods, except from CLIPArTT [15] which modifies the prompts, we use the 7 template in [49]:
- "`itap of a {class}`"
- "`a bad photo of the {class}`"
- "`a origami {class}`"
- "`a photo of the large {class}`"
- "`a {class} in a video game`"
- "`art of the {class}`"
- "`a photo of the small {class}`"

The text embedding for each class y is computed by

$$\boldsymbol{t}_y = \text{normalize}\left(\sum_{\kappa=1}^{k} \boldsymbol{t}_{y,\kappa}\right), \text{where } \boldsymbol{t}_{y,\kappa} = \text{normalize}(\text{text\_encoder}(\{\text{template}_\kappa, \text{classname}_y\})) \tag{12}$$

As a reference, we report the zero-shot performance of CLIP using both a single template "`a photo of a {class}.`" and an ensemble of seven templates, denoted as CLIP and Ensemble, respectively.

Hyperparameters We rerun all baseline methods and perform separate hyperparameter searches for both the single-domain and mixed-domain settings, reporting the best configurations for each. This is particularly important for online TTA methods (Tent [39], NOTE [12], SAR [28]), which suffer from batch dependency: their optimal learning rate is inversely correlated with the number of batches (i.e., update steps). Prior work [28] uses the same hyperparameters for both settings, which may underestimate the baselines' performance under mixed-domain shifts. Our separate tuning ensures that all results are fairly compared and free of bias.The final hyperparameter choices are listed below.

- **Tent** [39]. On CIFAR-10-C/CIFAR-100-C/ImageNet-C/DomainNet (same order below), we use SignSGD optimizer, and lr = 2e-5/2e-5/1e-4/1e-6 (mixed-domain shift) or lr = 1e-4/2e-4/2e-3/1e-6 (single-domain shift).
- **NOTE** [12]. Memory size is set to be the same as batch size. We use SignSGD optimizer, and lr = 2e-5/2e-5/1e-4/2e-5 (mixed-domain shift) or lr = 1e-4/2e-4/2e-3/1e-4 (single-domain shift).
- **SAR** [28]. We use SignSGD as the base optimizer of SAM, and lr = 2e-5/2e-5/1e-4/2e-5 (mixed-domain shift) or lr = 1e-4/2e-4/2e-3/1e-4 (single-domain shift). For all setting we use the same $E_0 = 0.4 \times \ln C$, where C is the number of classes, $\rho = 0.05$, and $e_0 = 0.2$, following their original hyperparameters.
- **RoTTA** [45]. We use SignSGD optimizer, and lr = 1e-2/1e-2/1e-2/1e-5 (mixed-domain shift) or lr = 2e-2/2e-2/2e-2/1e-4 (single-domain shift).
- **TDA** [21]. We use α = 1.0 for mixed-domain shift and α = 1.0/1.0/2.0/2.0 for single-domain shift. We use the default value for other hyperparameters.
- **DMN** [51]. We use α = 0.1 for mixed-domain shift and α = 0.2/1.0/1.0/0.2 for single-domain shift.
- **WATT-S** [30]. We use Adam optimizer, lr = 1e-3, $L = 2$ and $M = 5$, provided by the original paper in the same datasets.
- **CLIPArTT** [15]. We use Adam optimizer, lr = 1e-3, $K = 3$ and Iter=10, provided by the original paper in the same datasets.
- **Mint** [3]. We use Adam optimizer, lr = 5e-3/5e-3/1e-2/5e-3 (mixed-domain shift) or lr = 7e-3/7e-3/1.5e-2/5e-3 (single-domain shift), $K_{\text{prior}} = +\infty$ for mixed-domain shift and 10,000/10,000/10,000/100,000 for single-domain shifts. Single-domain hyperparameters follow those in the original paper.
- ***Ramen***. We use the same set of hyperparameters for both mixed-domain and single-domain shifts to demonstrate the robustness of *Ramen*. We set the maximum capacity K =7,500/750/75/300 on CIFAR-10-C, CIFAR-100-C, ImageNet-C, and DomainNet, respectively. As a result, the total capacity $C \cdot K$ remains roughly consistent across datasets. We use k =50/5/1/10, and β = 5.0/5.0/0.0/5.0. The optimizer is SignSGD with learning rate 1e-2 of for all datasets.

C.2. Error Bars (RQ1)

Tables 5 and 6 report the results in Tables 1 and 2 with error bars, respectively.

Table 5. Accuracy (mean (s.d.) %) on corruption benchmarks under mixture of 15 corruptions. Best and second-best results are shown in **bold** and underlined, respectively.

Method	Venue	Noise			Blur				Weather				Digital				Avg.
		Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	
ViT-B/32 on CIFAR-10-C																	
CLIP [33]	ICML'21	35.5	40.0	43.2	70.0	41.4	64.5	70.2	70.8	72.3	66.7	81.4	64.5	59.6	48.2	56.7	59.0
Ensemble	-	38.6	42.6	42.7	72.4	43.9	66.6	71.6	73.8	75.7	69.0	83.6	67.0	61.9	51.8	58.6	61.3
Tent [39]	ICLR'21	39.5 (0.6)	43.1 (0.4)	46.4 (0.3)	77.8 (0.3)	56.6 (0.3)	72.7 (0.4)	77.9 (0.4)	79.7 (0.2)	79.9 (0.2)	74.0 (0.3)	86.9 (0.1)	75.5 (0.1)	69.7 (0.3)	60.3 (0.1)	63.9 (0.4)	66.9 (0.2)
NOTE [12]	NeurIPS'22	44.9 (4.4)	48.3 (3.9)	48.9 (1.7)	77.1 (0.5)	57.0 (0.7)	74.0 (1.4)	77.6 (0.7)	78.0 (0.8)	79.2 (0.9)	73.0 (1.4)	86.5 (0.5)	73.5 (1.8)	68.7 (1.1)	57.9 (1.5)	62.2 (0.7)	67.1 (0.8)
SAR [28]	ICLR'23	48.8 (0.7)	51.8 (0.6)	51.0 (0.5)	77.5 (0.2)	60.5 (0.2)	74.8 (0.5)	**79.2** (0.3)	79.6 (0.2)	**80.7** (0.4)	76.6 (0.4)	87.4 (0.1)	77.0 (0.2)	71.0 (0.3)	57.5 (0.4)	62.8 (0.3)	69.1 (0.2)
RoTTA [45]	CVPR'23	53.1 (1.3)	56.4 (1.4)	51.3 (0.9)	**78.1** (0.5)	61.0 (1.0)	75.2 (0.2)	79.0 (0.4)	79.0 (0.6)	80.1 (1.0)	**80.0** (0.4)	85.6 (0.5)	**82.9** (0.4)	**73.7** (0.9)	65.3 (0.9)	**69.9** (0.7)	71.4 (0.4)
TDA [21]	CVPR'24	39.8 (0.4)	42.8 (0.4)	43.4 (0.3)	73.5 (0.3)	45.4 (0.2)	67.6 (0.2)	73.2 (0.2)	74.3 (0.2)	76.4 (0.3)	69.8 (0.2)	84.0 (0.2)	66.6 (0.2)	62.5 (0.1)	52.1 (0.3)	57.7 (0.4)	61.9 (0.1)
DMN-ZS [51]	CVPR'24	38.4 (0.1)	41.7 (0.1)	42.4 (0.1)	73.1 (0.0)	44.6 (0.0)	67.2 (0.1)	72.8 (0.1)	74.6 (0.1)	76.3 (0.0)	69.5 (0.0)	84.0 (0.0)	66.5 (0.0)	62.3 (0.0)	52.4 (0.1)	58.3 (0.1)	61.6 (0.0)
WATT-S [30]	NeurIPS'24	47.4 (0.3)	49.7 (0.1)	49.5 (0.3)	77.0 (0.2)	54.3 (0.2)	72.9 (0.2)	77.3 (0.2)	77.8 (0.2)	79.5 (0.1)	74.8 (0.2)	86.4 (0.1)	74.1 (0.2)	67.8 (0.2)	57.2 (0.4)	62.9 (0.2)	67.2 (0.0)
CLIPArTT [15]	WACV'25	37.2 (0.2)	41.5 (0.2)	44.8 (0.2)	70.5 (0.1)	48.9 (0.2)	65.5 (0.2)	70.7 (0.3)	72.9 (0.3)	75.3 (0.1)	67.4 (0.2)	82.8 (0.3)	66.9 (0.1)	61.9 (0.3)	58.5 (0.2)	59.5 (0.3)	61.6 (0.1)
Mint [3]	NeurIPS'25	47.9 (0.1)	50.9 (0.1)	50.3 (0.1)	74.1 (0.0)	51.3 (0.1)	75.8 (0.0)	76.9 (0.0)	76.4 (0.1)	77.0 (0.1)	73.8 (0.1)	85.8 (0.1)	73.8 (0.0)	65.8 (0.1)	47.3 (0.0)	56.4 (0.1)	65.6 (0.0)
Ramen	-	**61.0** (0.4)	**63.3** (0.3)	**56.0** (0.2)	77.2 (0.1)	**64.1** (0.2)	**77.2** (0.1)	78.9 (0.3)	**80.9** (0.1)	80.3 (0.2)	78.0 (0.2)	**88.1** (0.2)	79.8 (0.2)	72.0 (0.1)	**65.5** (0.1)	67.5 (0.2)	**72.7** (0.1)
ViT-B/16 on CIFAR-100-C																	
CLIP [33]	ICML'21	19.7	21.4	25.3	42.5	20.2	43.1	48.0	48.4	49.7	41.7	57.0	34.5	29.2	23.9	32.4	35.8
Ensemble	-	22.9	24.4	29.6	43.6	20.1	43.7	48.7	48.9	50.4	41.8	58.1	35.3	29.2	26.3	33.6	37.1
Tent [39]	ICLR'21	24.9 (0.2)	26.7 (0.2)	34.4 (0.1)	49.7 (0.1)	23.9 (0.2)	47.5 (0.1)	53.9 (0.1)	52.9 (0.1)	51.4 (0.2)	45.3 (0.2)	62.9 (0.2)	43.6 (0.1)	32.0 (0.1)	31.7 (0.1)	37.1 (0.1)	41.2 (0.0)
NOTE [12]	NeurIPS'22	25.7 (1.4)	27.3 (1.8)	35.4 (1.2)	49.9 (0.7)	23.9 (1.2)	47.6 (0.7)	54.2 (1.1)	52.7 (0.7)	51.8 (1.2)	45.9 (0.5)	63.7 (0.9)	44.0 (1.1)	32.4 (1.4)	30.2 (2.1)	36.0 (0.8)	41.4 (0.8)
SAR [28]	ICLR'23	28.4 (0.2)	30.5 (0.3)	38.5 (0.2)	50.4 (0.2)	24.6 (0.4)	49.2 (0.2)	55.1 (0.2)	54.1 (0.2)	53.4 (0.2)	47.2 (0.2)	64.0 (0.3)	45.0 (0.2)	33.6 (0.1)	29.7 (0.2)	37.4 (0.2)	42.7 (0.1)
RoTTA [45]	CVPR'23	22.9 (0.6)	24.5 (0.8)	32.4 (1.0)	47.9 (1.2)	25.4 (1.2)	46.5 (1.2)	50.1 (0.9)	50.5 (1.4)	50.7 (1.1)	50.8 (0.7)	58.3 (1.0)	**53.1** (0.7)	**37.7** (0.8)	29.9 (1.0)	36.2 (0.8)	41.1 (0.8)
TDA [21]	CVPR'24	23.4 (0.1)	25.2 (0.2)	30.3 (0.2)	44.1 (0.2)	20.3 (0.2)	43.8 (0.1)	49.1 (0.2)	49.3 (0.2)	51.3 (0.1)	42.2 (0.1)	58.7 (0.1)	36.1 (0.1)	29.3 (0.1)	26.4 (0.2)	33.6 (0.1)	37.5 (0.0)
DMN-ZS [51]	CVPR'24	22.9 (0.1)	24.4 (0.2)	29.6 (0.1)	44.3 (0.1)	20.2 (0.1)	44.1 (0.0)	49.4 (0.1)	49.7 (0.1)	51.1 (0.1)	42.2 (0.1)	58.8 (0.1)	35.5 (0.1)	29.4 (0.0)	26.2 (0.1)	34.0 (0.1)	37.5 (0.0)
WATT-S [30]	NeurIPS'24	26.4 (0.2)	28.4 (0.3)	35.6 (0.3)	50.2 (0.3)	24.2 (0.3)	48.5 (0.2)	54.4 (0.2)	54.2 (0.3)	54.0 (0.2)	47.7 (0.3)	63.5 (0.4)	43.3 (0.2)	34.0 (0.2)	31.4 (0.2)	37.7 (0.4)	42.2 (0.1)
CLIPArTT [15]	WACV'25	21.1 (0.3)	22.8 (0.2)	29.4 (0.2)	46.3 (0.1)	24.5 (0.3)	45.0 (0.1)	51.1 (0.1)	51.5 (0.1)	51.8 (0.2)	44.0 (0.3)	61.1 (0.2)	39.0 (0.2)	33.2 (0.2)	28.3 (0.3)	36.7 (0.1)	39.0 (0.1)
Mint [3]	NeurIPS'25	22.3 (0.0)	24.5 (0.1)	33.0 (0.1)	49.2 (0.1)	22.4 (0.1)	47.8 (0.1)	54.2 (0.1)	52.4 (0.0)	51.6 (0.1)	47.7 (0.1)	63.7 (0.1)	42.2 (0.1)	31.8 (0.0)	24.0 (0.1)	34.3 (0.0)	40.1 (0.0)
Ramen	-	**30.3** (0.3)	**32.9** (0.4)	**42.8** (0.3)	**52.5** (0.3)	**29.1** (0.4)	**52.2** (0.2)	**55.5** (0.1)	**55.4** (0.1)	**55.1** (0.2)	**52.6** (0.2)	**65.9** (0.2)	51.0 (0.2)	36.8 (0.2)	**40.4** (0.4)	**39.6** (0.2)	**46.1** (0.1)
ViT-L/14 on ImageNet-C																	
CLIP [33]	ICML'21	27.4	29.4	28.7	34.6	25.3	41.0	36.7	49.8	44.1	49.7	65.4	35.1	30.3	53.5	42.2	39.6
Ensemble	-	29.1	30.4	30.1	37.5	27.3	44.2	39.2	52.4	46.4	52.6	67.8	34.4	32.4	56.2	44.2	41.6
Tent [39]	ICLR'21	36.3 (0.4)	38.0 (0.1)	38.5 (0.1)	40.3 (0.3)	37.4 (0.5)	47.7 (0.3)	43.5 (0.4)	54.2 (0.4)	49.5 (0.2)	56.4 (0.3)	67.8 (0.2)	45.4 (0.7)	40.6 (0.5)	57.6 (0.3)	46.6 (0.3)	46.7 (0.1)
NOTE [12]	NeurIPS'22	36.3 (0.4)	38.0 (0.2)	38.6 (0.3)	40.0 (0.4)	37.3 (0.4)	47.6 (0.3)	43.5 (0.3)	54.4 (0.4)	49.7 (0.5)	56.4 (0.5)	67.8 (0.3)	44.5 (0.4)	40.8 (0.4)	57.5 (0.3)	46.2 (0.5)	46.6 (0.1)
SAR [28]	ICLR'23	36.3 (0.8)	38.3 (0.4)	38.5 (0.4)	39.2 (0.7)	38.5 (0.8)	46.8 (0.5)	43.4 (0.6)	53.9 (0.3)	50.4 (0.5)	56.6 (0.4)	66.8 (0.2)	45.5 (0.1)	41.8 (0.8)	56.2 (0.6)	46.6 (0.5)	46.6 (0.3)
RoTTA [45]	CVPR'23	37.4 (0.5)	38.7 (0.4)	39.6 (0.7)	36.9 (0.5)	34.5 (0.5)	44.0 (0.5)	39.3 (0.7)	53.5 (0.6)	48.6 (0.6)	54.2 (0.7)	67.2 (0.3)	43.0 (0.8)	39.7 (0.6)	56.3 (0.5)	48.4 (0.6)	45.4 (0.4)
TDA [21]	CVPR'24	29.6 (0.1)	31.0 (0.1)	31.5 (0.2)	38.1 (0.1)	28.9 (0.1)	44.6 (0.1)	40.0 (0.1)	53.4 (0.1)	47.5 (0.1)	53.3 (0.2)	68.5 (0.1)	39.1 (0.2)	33.4 (0.2)	57.1 (0.2)	45.1 (0.2)	42.7 (0.0)
DMN-ZS [51]	CVPR'24	29.2 (0.1)	30.5 (0.1)	30.2 (0.0)	37.6 (0.1)	27.6 (0.1)	44.4 (0.1)	39.2 (0.1)	52.5 (0.0)	46.6 (0.1)	52.7 (0.0)	67.9 (0.0)	33.8 (0.1)	32.5 (0.0)	56.5 (0.1)	44.5 (0.1)	41.7 (0.0)
WATT-S [30]	NeurIPS'24	31.8 (0.1)	33.1 (0.1)	33.6 (0.2)	39.2 (0.2)	30.7 (0.3)	45.7 (0.1)	41.3 (0.1)	53.5 (0.2)	47.3 (0.3)	53.7 (0.2)	67.9 (0.3)	40.1 (0.1)	34.7 (0.3)	57.3 (0.1)	45.6 (0.3)	43.7 (0.0)
CLIPArTT [15]	WACV'25	28.2 (0.3)	30.2 (0.3)	29.8 (0.2)	35.0 (0.2)	26.5 (0.3)	41.4 (0.2)	37.3 (0.1)	50.3 (0.3)	43.7 (0.3)	49.7 (0.2)	65.0 (0.2)	38.5 (0.2)	31.0 (0.3)	53.5 (0.2)	42.3 (0.3)	40.2 (0.1)
Mint [3]	NeurIPS'25	33.8 (0.2)	35.2 (0.2)	35.8 (0.2)	39.3 (0.2)	33.3 (0.3)	46.2 (0.2)	41.5 (0.2)	53.6 (0.3)	47.4 (0.2)	53.3 (0.1)	67.7 (0.1)	36.3 (0.5)	38.2 (0.3)	58.1 (0.1)	47.4 (0.7)	44.5 (0.1)
Ramen	-	**37.5** (0.2)	**38.8** (0.1)	**41.1** (0.2)	**42.2** (0.3)	**39.6** (0.4)	**49.9** (0.4)	**46.2** (0.4)	**57.3** (0.3)	**51.6** (0.1)	**59.4** (0.1)	**68.7** (0.2)	**46.4** (0.4)	**45.2** (0.2)	**59.4** (0.2)	**54.7** (0.2)	**49.2** (0.1)

Table 6. Mean accuracy (%) on DomainNet under mixture of six domains. Best and second-best results are shown in **bold** and underlined, respectively.

Method	Venue	ViT-B/32 on DomainNet						
		clip	info	paint	quick	real	sketch	Avg.
CLIP [33]	ICML'21	67.6	41.5	62.7	12.8	81.0	58.1	54.0
Ensemble	-	68.9	44.2	64.7	13.3	82.3	60.3	55.6
Tent [39]	ICLR'21	69.1 (0.0)	44.3 (0.0)	64.9 (0.0)	13.0 (0.0)	82.3 (0.0)	60.3 (0.0)	55.7 (0.0)
NOTE [12]	NeurIPS'22	69.6 (0.1)	45.1 (0.0)	65.0 (0.0)	16.4 (0.1)	82.1 (0.0)	60.8 (0.0)	56.5 (0.0)
SAR [28]	ICLR'23	69.4 (0.0)	44.7 (0.0)	65.2 (0.1)	14.5 (0.0)	82.5 (0.0)	60.8 (0.0)	56.2 (0.0)
RoTTA [45]	CVPR'23	69.4 (0.1)	44.8 (0.0)	65.3 (0.0)	14.3 (0.0)	82.4 (0.0)	60.8 (0.0)	56.2 (0.0)
TDA [21]	CVPR'24	68.9 (0.0)	44.7 (0.0)	65.1 (0.1)	14.6 (0.1)	82.3 (0.0)	60.5 (0.0)	56.0 (0.0)
DMN-ZS [51]	CVPR'24	68.9 (0.0)	44.6 (0.0)	64.8 (0.0)	12.7 (0.0)	**82.6** (0.0)	60.2 (0.0)	55.6 (0.0)
WATT-S [30]	NeurIPS'24	68.3 (0.0)	42.8 (0.1)	63.5 (0.1)	17.0 (0.0)	81.4 (0.0)	59.9 (0.1)	55.5 (0.0)
CLIPArTT [15]	WACV'25	67.2 (0.0)	41.9 (0.0)	62.9 (0.0)	12.6 (0.0)	80.8 (0.0)	57.8 (0.1)	53.9 (0.0)
Mint [3]	NeurIPS'25	69.3 (0.0)	44.4 (0.0)	65.0 (0.0)	15.0 (0.0)	82.3 (0.0)	60.5 (0.0)	56.1 (0.0)
Ramen	-	**70.1** (0.0)	**45.2** (0.0)	**65.8** (0.0)	**17.6** (0.1)	82.4 (0.0)	**61.5** (0.0)	**57.1** (0.0)

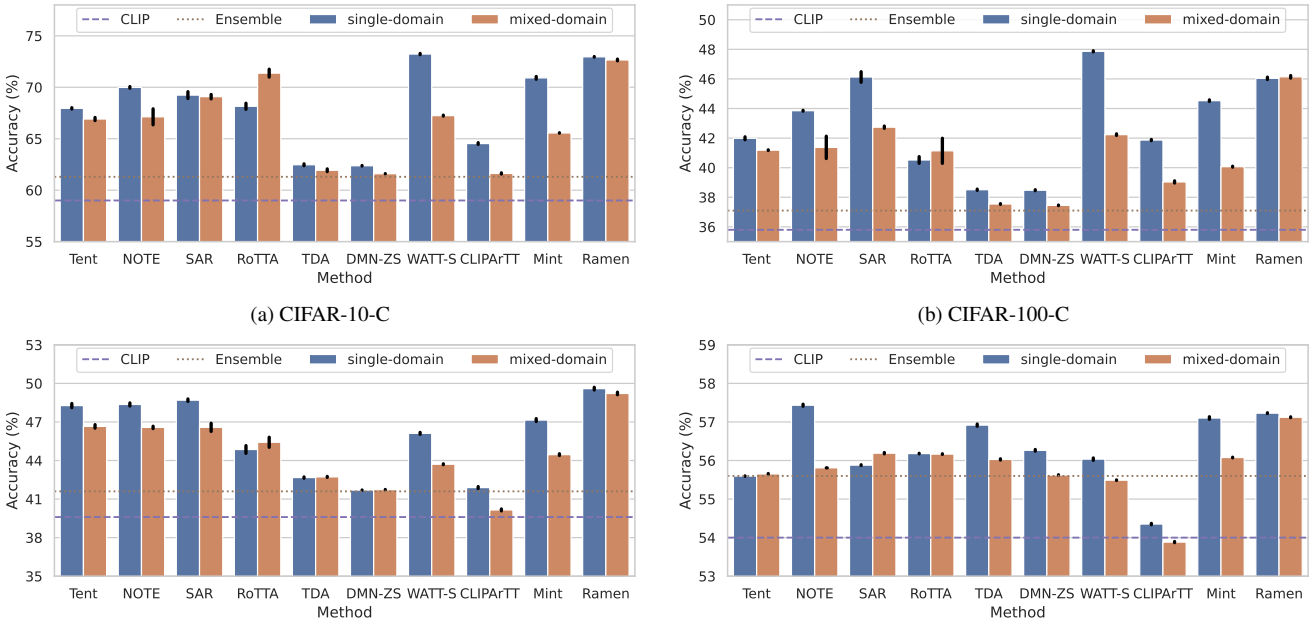
C.3. Comparison of Single-Domain and Mixed-Domain Shifts (RQ1)

We compare the performance of all methods across both single-domain and mixed-domain settings on all datasets. In the single-domain evaluation, models are tested on each domain separately without mixing test samples, and we report the average accuracy across all domains. For all baselines, especially online adaptation methods, we tune the learning rate separately for the two settings, since their performance is highly sensitive to the product of update steps and learning rate. In contrast, *Ramen* uses the same hyperparameters across both settings.

We observe that most existing methods experience a noticeable performance drop under mixed-domain shifts compared to the single-domain case, with a few exceptions.

- RoTTA, designed for continual TTA, shows limited degradation under domain mixing.
- On DomainNet, since different domains have distinct label distributions, mixing them unexpectedly reduces prediction bias for certain methods such as Tent and SAR. As a validation, NOTE, which already includes a prediction-balanced memory, still suffers under domain mixtures.

Across all four datasets, *Ramen* maintains consistently strong performance in both single-domain and mixed-domain settings.



(a) CIFAR-10-C

(b) CIFAR-100-C

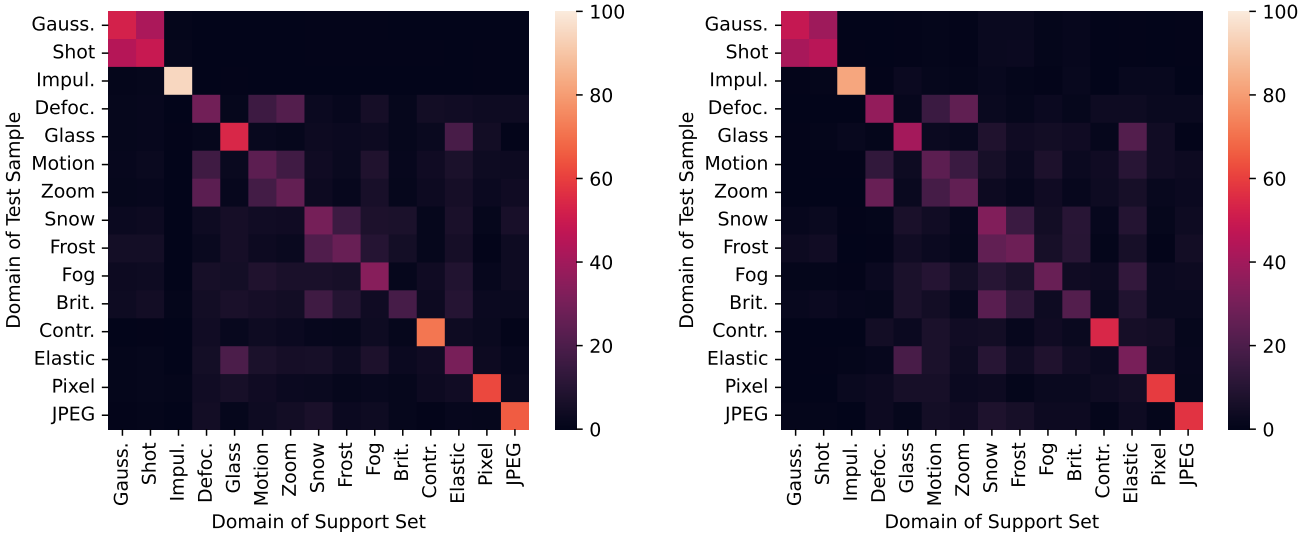
(c) ImageNet-C

(d) DomainNet

Figure 6. Performance comparison of TTA methods under single-domain and mixed-domain shifts.

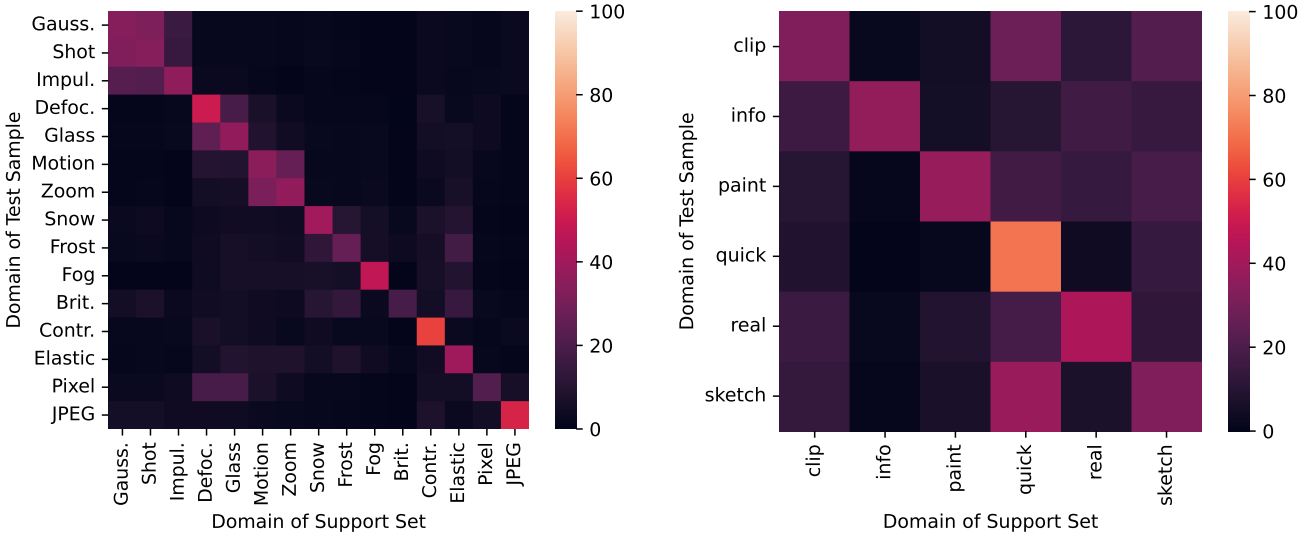
C.4. Visualization of Active Sample Selection (RQ2)

We visualize, for all datasets, the domain distribution of retrieved samples during active sample selection. Each entry (i, j) indicates the average proportion (%) of support samples from domain j when the test sample comes from domain i. Numbers in parentheses denote the proportion of support samples that come from the same domain as the test sample. On average, 44.5%/40.9%/38.0%/47.8% of the retrieved samples come from the same domain as the test sample on CIFAR-10-C/CIFAR-100-C/ImageNet-C/DomainNet, respectively, which is substantially higher than random selection (6.7%/6.7%/6.7%/16.7%). Moreover, even when the retrieved samples are from different domains, they typically belong to semantically or visually related domains (e.g., Gaussian noise and shot noise, motion blur and zoom blur, sketch and quickdraw).



(a) CIFAR-10-C (44.5%)

(b) CIFAR-100-C (40.9%)



(c) ImageNet-C (38.0%)

(d) DomainNet (47.8%)

Figure 7. Visualization of active sample selection. Each entry (i, j) indicates the average proportion (%) of support samples from domain j when the test sample comes from domain i. Numbers in the parentheses indicate the proportion of support samples that come from the same domain as the test sample.

C.5. Memory Usage (RQ3)

In the main text, we report the runtime of *Ramen* and baseline TTA methods. Here, we further analyze memory usage as reported in Table 7. Although *Ramen* is highly efficient in computation due to the embedding-gradient cache, this efficiency is achieved by trading memory for time: storing both embeddings and gradients inevitably consumes GPU memory. In our CIFAR-100-C experiments, the GPU memory usage of the embedding-gradient cache is

$$\underbrace{100}_{\text{Number of classes } C} \times \underbrace{750}_{\text{Maximum Capacity per class } K} \times \Big(\underbrace{512}_{\text{Embedding dim}} + \underbrace{39{,}936}_{\text{Gradient dim}} \Big) \times \underbrace{2\,\text{B}}_{\text{Half dtype}} \approx 5,785\ \text{MiB},$$

which constitutes the main source of GPU memory consumption. In addition, the reparameterization trick introduced in Appendix A.3 to improve parallelism also incurs about 1,300 MiB additional memory overhead. Overall, *Ramen* requires 14,526 MiB of GPU memory in total. While this represents an increase compared to the simplest baselines (e.g., EntMin), it remains within a reasonable range given the substantial performance and efficiency gains.

Ways to save memory When GPU memory is highly limited, several trade-offs are available. Reducing the maximum capacity K (e.g., from 750 to 300, which only slightly affects accuracy according to Figure 8) or lowering the batch size (which has negligible impact as shown in Figure 10) both effectively decrease memory usage. Additionally, we provide an alternative configuration that offloads gradients to the CPU: embeddings are stored on the GPU for the computation of distance and aggregation weights, while gradients are stored and aggregated on the CPU before the aggregated result is transferred back to the GPU. Although this design introduces extra CPU–GPU communication and increases computation time, the GPU memory used by the cache drops drastically to

$$\underbrace{100}_{\text{Number of classes } C} \times \underbrace{750}_{\text{Maximum Capacity per class } K} \times \underbrace{512}_{\text{Embedding dim}} \times \underbrace{2\,\text{B}}_{\text{Half dtype}} \approx 73\ \text{MiB},$$

which is nearly negligible compared to the model's computational graph.

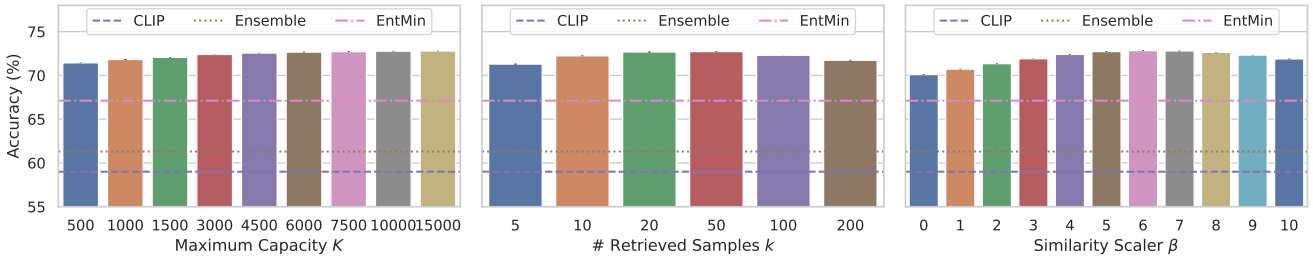
Table 7. Comparison of GPU Memory usage on CIFAR-100-C.

Method	GPU Memory	Testing Time
EntMin	7,290 MiB	11m42s
Ramen (cache on CPU)	8,642 MiB	2h13m03s
Ramen (embeddings on GPU, gradients on CPU)	8,724 MiB	46m47s
Ramen (cache on GPU)	14,526 MiB	14m08s

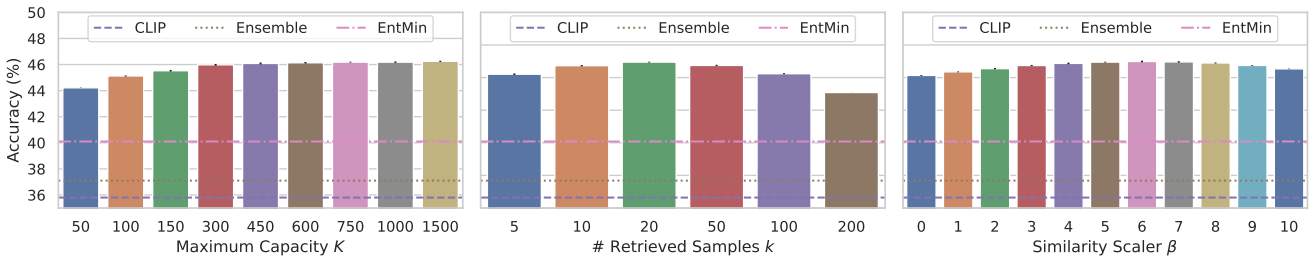
C.6. Hyperparameter Sensitivity

Effect of hyperparameters introduced by *Ramen* (K, k, β) *Ramen* introduces three key hyperparameters: the maximum capacity per class K, the number of retrieved samples per class k, and the similarity scaler β. We evaluate their effects on all datasets, as shown in Figure 8.

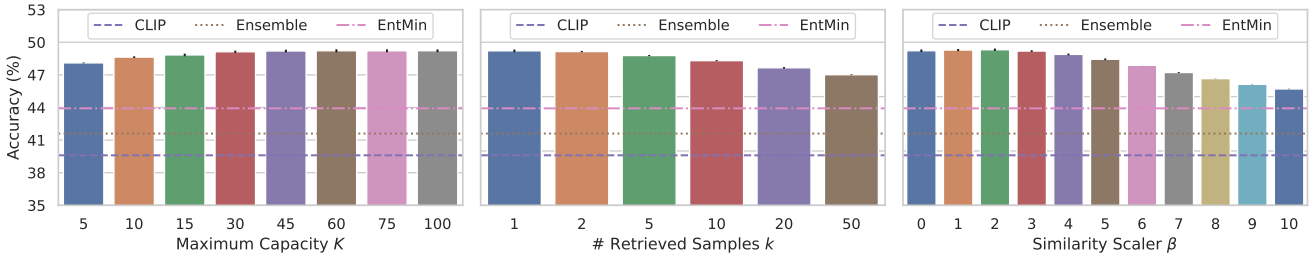
- Across all datasets, ***Ramen* consistently outperforms vanilla entropy minimization (EntMin) over a wide range of hyperparameter choices**, indicating that the method is generally robust and effective once the proposed components are incorporated.
- Moreover, the **influence patterns of these hyperparameters are remarkably consistent** across different datasets.
- On ImageNet-C, since the number of samples per class and per domain is relatively small, we set $k = 1$. In this case, the similarity scaler β becomes irrelevant, as its primary role is to assign finer-grained weights among multiple retrieved samples within each class, which is unnecessary when only one sample is retrieved.



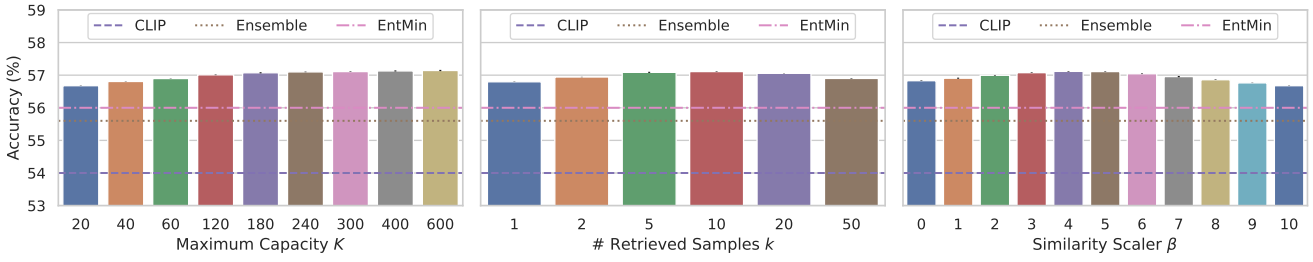
(a) CIFAR-10-C



(b) CIFAR-100-C



(c) ImageNet-C

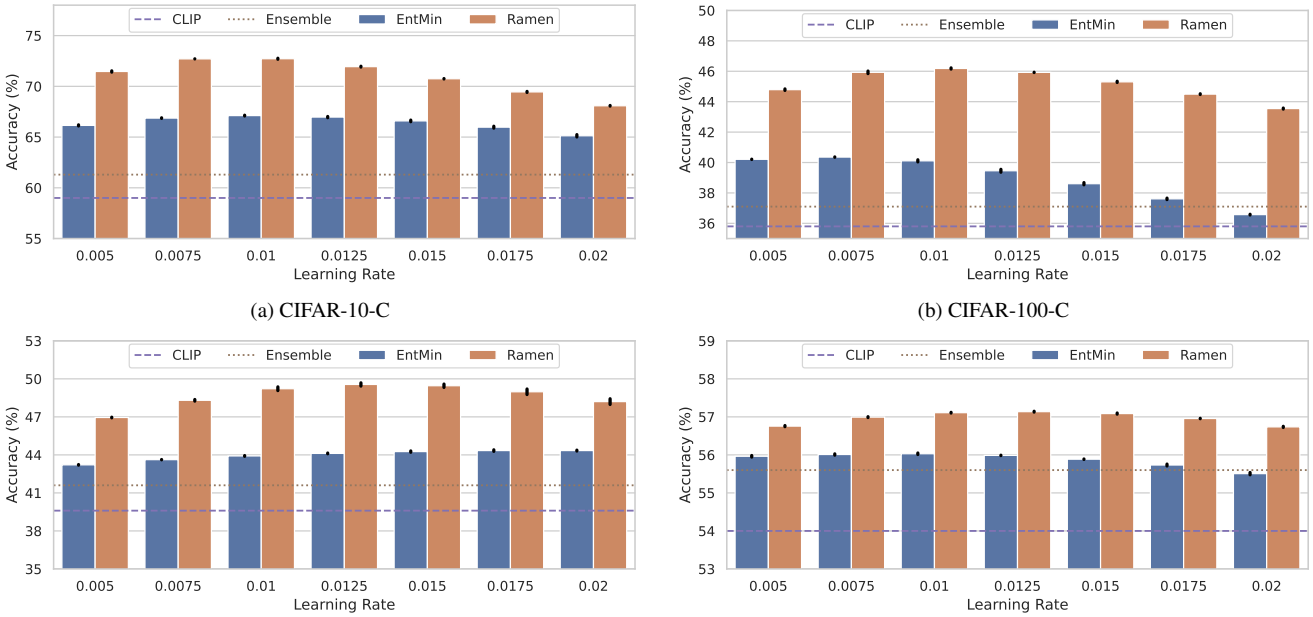


(d) DomainNet

Figure 8. Hyperparameter sensitivity. (EntMin refers to entropy minimization without *Ramen*.)

Effect of learning rate η The learning rate is one of the most critical hyperparameters in test-time adaptation. Figure 9 compares the performance of *Ramen* and vanilla entropy minimization (EntMin) under different learning rates. We observe that:

- Within a broad range of learning rates, *Ramen* yields noticeable performance gains compared to the non-adaptive ensemble baseline.
- *Ramen* consistently outperforms EntMin across all settings.
- The optimal learning rate remains highly consistent across different datasets and ViT model scales, demonstrating the robustness of the proposed method.



(a) CIFAR-10-C

(b) CIFAR-100-C

(c) ImageNet-C

(d) DomainNet

Figure 9. Effect of learning rate. (EntMin refers to entropy minimization without *Ramen*.)

Effect of batch size B A byproduct of *Ramen* is its insensitivity to the test-time batch size. Since the samples used for adaptation are retrieved from the memory (a total of $C \cdot k$ entries), the update process relies primarily on the retrieved support set rather than the incoming batch itself. As shown in Figure 10, the performance of *Ramen* remains largely stable across different batch sizes.

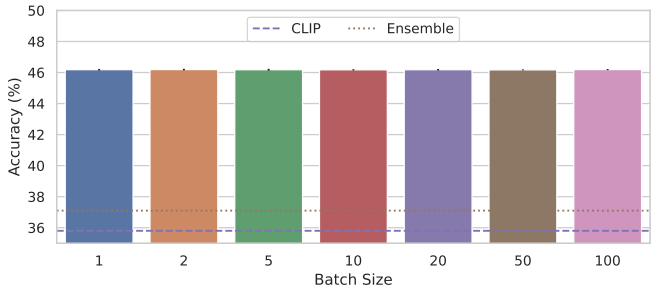


Figure 10. Effect of batch size on *Ramen*.

C.7. Ablation Study

We perform an ablation study on the two components of the gradient aggregation weights, entropy and similarity, on the CIFAR-100-C dataset. As shown in Figure 8, removing either component degrades performance, indicating that both contribute to the effectiveness of *Ramen*.

Table 8. Ablation study on sample weighting.

Method	Entropy Weighting	Similarity Weighting	Accuracy (%)
CLIP	-	-	35.8
Ensemble	-	-	37.1
EntMin	-	-	40.1 (0.1)
Ramen	✗	✗	43.9 (0.1)
	✗	✓	45.2 (0.0)
	✓	✗	45.2 (0.0)
	✓	✓	46.1 (0.1)

C.8. More Architectures

In the main text, we use ViT models of different size as the visual encoder, since they generally yield stronger performance. Here, we further evaluate *Ramen* using a ResNet (RN) [16] backbone to verify its generality. Table 9 summarizes the results on CIFAR-100-C with RN101 as the image encoder. We observe that *Ramen* remains effective, demonstrating its compatibility with different model architectures.

Table 9. Accuracy (mean (s.d.) %) of RN101 on CIFAR-100-C under mixture of 15 corruptions. Best and second-best results are shown in **bold** and underlined, respectively.

Method	Venue	RN101 on CIFAR-100-C															
		Noise			Blur				Weather				Digital				Avg.
		Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	
CLIP [33]	ICML'21	10.8	12.9	8.7	19.3	12.3	20.8	24.3	28.5	31.8	27.5	36.0	18.4	19.5	17.7	20.5	20.6
Ensemble	-	11.7	14.5	9.6	20.5	11.4	21.5	25.9	29.2	31.8	26.8	37.3	18.8	18.5	19.3	21.7	21.2
Tent [39]	ICLR'21	12.2 (0.1)	15.1 (0.1)	10.3 (0.1)	23.2 (0.2)	10.3 (0.1)	22.4 (0.1)	28.4 (0.1)	30.4 (0.2)	31.6 (0.1)	26.8 (0.1)	39.1 (0.2)	20.6 (0.1)	18.2 (0.1)	19.7 (0.2)	23.0 (0.2)	22.1 (0.0)
NOTE [12]	NeurIPS'22	13.9 (0.5)	17.2 (0.8)	13.2 (0.5)	29.9 (0.6)	11.8 (0.7)	25.8 (0.6)	34.2 (0.7)	34.0 (0.4)	35.1 (0.7)	30.9 (0.8)	44.4 (0.7)	27.2 (1.0)	20.8 (0.7)	20.2 (1.0)	25.1 (0.1)	25.6 (0.4)
SAR [28]	ICLR'23	13.0 (0.2)	16.2 (0.1)	11.3 (0.1)	26.0 (0.2)	10.2 (0.1)	24.1 (0.2)	31.0 (0.2)	31.8 (0.2)	32.8 (0.2)	28.2 (0.1)	41.3 (0.2)	22.8 (0.2)	18.8 (0.3)	19.7 (0.2)	23.9 (0.2)	23.4 (0.1)
RoTTA [45]	CVPR'23	13.3 (0.9)	16.3 (0.8)	12.7 (0.5)	28.5 (0.4)	9.6 (0.5)	24.4 (0.5)	30.5 (0.3)	22.4 (0.8)	28.2 (0.6)	31.5 (0.5)	32.5 (0.9)	29.9 (0.5)	19.5 (0.4)	18.4 (0.8)	22.4 (0.4)	22.7 (0.3)
TDA [21]	CVPR'24	12.8 (0.1)	15.6 (0.1)	11.0 (0.2)	20.5 (0.3)	11.7 (0.1)	22.5 (0.2)	26.4 (0.2)	30.9 (0.3)	33.1 (0.1)	27.7 (0.2)	38.7 (0.3)	19.5 (0.1)	19.6 (0.1)	20.2 (0.1)	22.4 (0.2)	22.2 (0.1)
DMN-ZS [51]	CVPR'24	12.0 (0.0)	14.7 (0.1)	7.8 (0.1)	21.0 (0.1)	11.7 (0.1)	22.1 (0.0)	26.4 (0.1)	30.0 (0.1)	32.4 (0.1)	27.1 (0.0)	37.9 (0.1)	19.3 (0.1)	19.0 (0.0)	19.6 (0.1)	22.4 (0.1)	21.6 (0.0)
WATT-S [30]	NeurIPS'24	15.8 (0.1)	18.9 (0.2)	15.4 (0.1)	33.9 (0.2)	**13.8** (0.1)	28.2 (0.1)	37.9 (0.1)	36.6 (0.3)	**38.2** (0.2)	34.6 (0.2)	47.9 (0.2)	31.6 (0.3)	**23.7** (0.2)	20.5 (0.2)	26.6 (0.1)	28.2 (0.1)
CLIPArTT [15]	WACV'25	11.0 (0.1)	13.4 (0.1)	10.4 (0.1)	26.7 (0.2)	12.4 (0.2)	23.1 (0.2)	30.5 (0.4)	31.3 (0.2)	32.6 (0.1)	28.6 (0.1)	40.8 (0.3)	24.2 (0.2)	19.1 (0.2)	21.8 (0.2)	23.4 (0.3)	23.3 (0.1)
Mint [3]	NeurIPS'25	15.3 (0.1)	18.6 (0.2)	13.4 (0.1)	31.2 (0.0)	11.1 (0.1)	26.5 (0.1)	35.7 (0.1)	34.9 (0.1)	35.1 (0.1)	31.5 (0.1)	44.9 (0.1)	27.9 (0.0)	21.7 (0.0)	18.5 (0.1)	24.8 (0.1)	26.1 (0.0)
Ramen	-	**19.0** (0.2)	**21.9** (0.2)	**16.3** (0.2)	**36.3** (0.4)	13.1 (0.1)	**30.4** (0.2)	**40.3** (0.2)	**36.7** (0.2)	36.8 (0.2)	**35.5** (0.2)	**49.3** (0.3)	**37.7** (0.3)	22.3 (0.4)	**25.9** (0.4)	**26.9** (0.2)	**29.9** (0.1)