TRUST: Test-Time Refinement using Uncertainty-Guided SSM Traverses

Sahar Dastani[1,2*] Ali Bahri[1*] Gustavo Adolf Vargas Hakim[1]
Moslem Yazdanpanah[1] Mehrdad Noori[1] David Osowiechi[1] Samuel Barbeau[1]
Ismail Ben Ayed[1] Herve Lombaert[2,3]
Christian Desrosiers[1]
[1]LIVIA, ILLS, ÉTS Montréal, Canada,
[2]Mila - Quebec AI Institute, [3]Polytechnique Montreal

Abstract

State Space Models (SSMs) have emerged as efficient alternatives to Vision Transformers (ViTs), with VMamba standing out as a pioneering architecture designed for vision tasks. However, their generalization performance degrades significantly under distribution shifts. To address this limitation, we propose TRUST (Test-Time Refinement using Uncertainty-Guided SSM Traverses), a novel test-time adaptation (TTA) method that leverages diverse traversal permutations to generate multiple causal perspectives of the input image. Model predictions serve as pseudo-labels to guide updates of the Mamba-specific parameters, and the adapted weights are averaged to integrate the learned information across traversal scans. Altogether, TRUST is the first approach that explicitly leverages the unique architectural properties of SSMs for adaptation. Experiments on seven benchmarks show that TRUST consistently improves robustness and outperforms existing TTA methods. The code is available at: `https://github.com/Sahardastani/trust`.

1 Introduction

The field of visual representation learning has advanced rapidly due to the ability of deep neural networks to extract rich and generalizable features. Convolutional Neural Networks (CNNs) [1, 2, 3, 4, 5] excel in modeling local patterns through strong inductive biases but struggle with global context. Vision Transformers (ViTs) [6, 7, 8, 9] address this challenge using a self-attention mechanism, although at higher computational cost. More recently, State Space Models (SSMs) [10, 11, 12] emerged as a scalable alternative, providing global receptive fields with linear complexity. Despite their efficiency, the performance of SSMs degrades under distribution shift. This is mainly due to violations of the Independently and Identically Distributed (i.i.d) assumption, which is often disrupted in real-world settings. While generalization strategies are well developed for CNNs and ViTs, vision-specific strategies for SSMs are still lacking.

This work focuses on *VMamba* [13], a vision-adapted variant of the Mamba architecture [14], designed for sequential visual processing. VMamba introduces 2D Selective Scan (SS2D), a four-way traversal mechanism that scans image patches along predefined spatial directions. However, this directional processing introduces a strong inductive bias by aligning internal representations with fixed traversal paths [15], which may hinder generalization under distribution shifts. Additionally, the hidden states of VMamba store historical context over the traversal sequence. When exposed to unseen domains, this context may accumulate domain-specific artifacts, leading to amplified bias during propagation and ultimately degrading generalization [16].

*Equal contribution

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

To address these challenges, we propose **T**est-Time **R**efinement using **U**ncertainty-Guided **S**SM **T**raverses (TRUST), a novel Test Time Adaptation (TTA) strategy specifically designed for SSMs that leverages VMamba's internal traversal mechanism. As illustrated in the offline phase of Figure 1, different traversal permutations result in varying prediction entropy levels, revealing the sensitivity of the model to causal ordering. TRUST systematically generates multiple traversal permutations by reordering the four directional scans, exposing the model to diverse causal perspectives of the same input. At test time, we compute the entropy of predictions from each permutation and select the ones with the lowest entropy, which are associated with more stable and domain-robust hidden states.

Inspired by previous work on flat minima and weight averaging [17, 18], our method aggregates the outputs of different traversal permutations to implicitly explore a broader and flatter region of the loss landscape. Each permutation activates a distinct computational pathway through the model, effectively sampling different local minima. Selecting the top-k traversal permutations with the lowest predictive entropy and computing a weighted average of their model parameters mitigates the impact of noisy or uncertain predictions. This enhances generalization without requiring source data or model retraining.

We outline the main contributions of our work as follows:

- We propose TRUST, the first TTA approach specifically designed for Mamba-based vision models. Unlike prior methods that depend on data augmentation or auxiliary models, our approach takes advantage of the internal traversal dynamics of VMamba to improve generalization under distribution shift.
- TRUST introduces a novel weighted averaging strategy to promote robustness, which permutes traversal directions and averages model weights from the most confident paths.
- We validate our method on **seven standard benchmarks**, where it consistently outperforms existing TTA methods, establishing new state-of-the-art results.

2 Related Work

Test Time Adaptation (TTA) is a paradigm of Domain Adaptation (DA) [19] that intends to enhance model generalization during deployment. Specifically, a pre-trained model is adapted to incoming data batches without requiring label supervision or having access to source data. As a pioneering approach, Tent [20] minimizes prediction entropy as an adaptation objective, based on the principle that distribution shifts in the target domain reduce the confidence of a model in its predictions. Examples of Tent-based approaches include using confidence thresholds for sample selection before adaptation [21, 22], minimizing the sharpness of the entropy loss [23], using class-balanced memory queues for better adaptation sampling [24], and meta-learning the entropy loss [25], among other alternatives [26, 27, 28]. The literature also includes several other adaptation strategies that do not minimize entropy, such as contrastive learning [29, 30], Laplacian optimization [31, 32], and prototype-based pseudo-labeling [33]. While flexible, these techniques do not capitalize on the unique properties of specific architectures such as VMamba.

Architecture-specific TTA. Recent advances in TTA have introduced a variety of model-specific strategies. Among those, FOA [34] loosely adheres to ViTs by incorporating gradient-free learning of additional prompts in the form of token embeddings. For Vision-language Models (VLMs) [35, 36, 37, 38], recent strategies incorporate text information as an auxiliary tool for adaptation. Examples include prompt learning via entropy minimization [35], positive and negative caching with text-based pseudo-labels [36], transductive adaptation using conformal pseudo-captions [38], and weight averaging across text templates [37]. SSM-oriented TTA has also been explored to a smaller extent. In a recent work, STAD [39] uses Markov processes to learn time-varying prototypes for classifying examples during inference and adapts the classification head of the model.

Weight Averaging for Robust Adaptation. Initially introduced in the context of Domain Generalization (DG) [40, 41, 42], weight averaging has gained popularity as an effective technique for enhancing model robustness to domain shifts [43, 18]. Notably, methods like SWAD [18] aim to identify flat minima in the loss landscape by minimizing loss fluctuations across diverse inputs. This is often accomplished by averaging the weights of models trained on different image augmentations. While similar ideas have been adopted in the TTA literature [23, 37, 44, 45, 46], they typically rely on explicit input diversification, which introduces additional computational overhead. In contrast, our

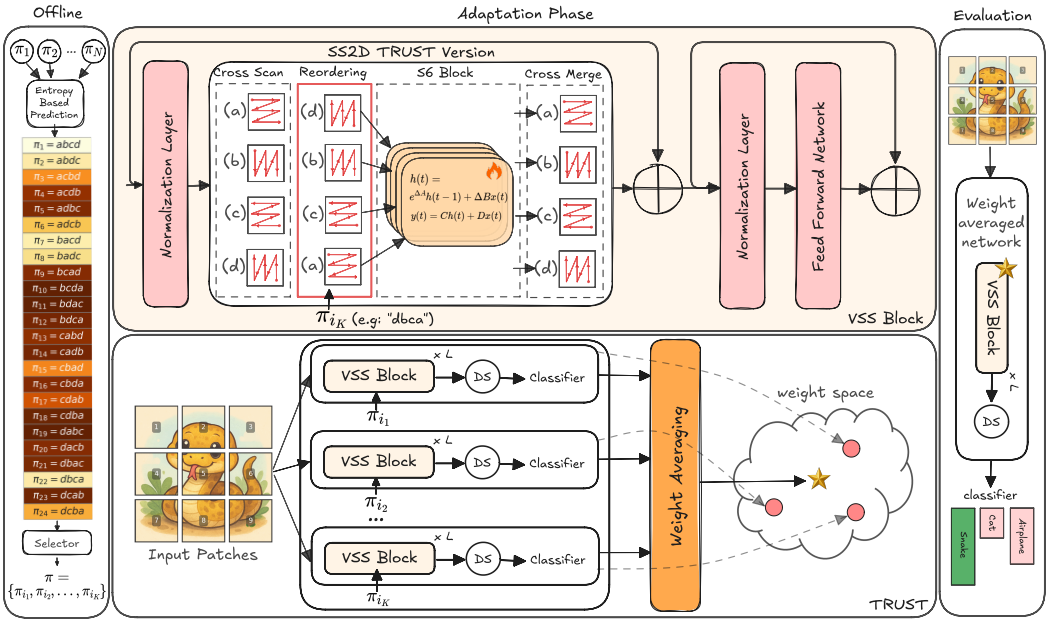

Figure 1: An overview of the proposed method. Our network consists of three stages: offline, adaptation, and evaluation. In the Offline stage, multiple traversal permutations are generated and ranked by entropy. The top-K most confident permutations are selected. During adaptation, each permutation reorders the cross-scan traversals, and the Mamba-specific state-space parameters are updated accordingly. The resulting models are then merged via parameter-space averaging. The final averaged model is used for inference in evaluation stage.

approach leverages the intrinsic diversity of SSM-based traversals to induce variation directly in the weights of our model, without requiring external data augmentations.

3 Method

Section 3.1 introduces key concepts in TTA and SSM, laying the groundwork for our approach. In Section 3.2, we propose traversal permutations to generate complementary causal perspectives of the input, followed by a weight averaging strategy in Section 3.3 for robust and efficient inference. Figure 1 shows an overview of our method.

3.1 Preliminaries

Test Time Adaptation (TTA) in Image Classification. Consider a classification model composed of a feature processor f_θ, parameterized by θ, and a classifier h_ϕ parameterized by ϕ. The model is pre-trained on a source dataset $\mathcal{D}_s = \{(\mathbf{X}_j, y_j)\}_{j=1}^{N_s}$ consisting of images $\mathbf{X}_j \in \mathbb{R}^{W\times H\times 3}$ and corresponding labels $y_j \in \{1, \ldots, C\}$, where C is the number of classes. Formally, the model learns a map $F_s : \mathcal{X}_s \to \mathcal{Y}_s$ between the input distribution $\mathcal{X}_s$ and output distribution $\mathcal{Y}_s$ from the source. Training on $\mathcal{D}_s$ is performed using a supervised loss $\mathcal{L}_{\text{train}}$ (typically cross-entropy). At test-time, the model is exposed to a target dataset $\mathcal{D}_t = \{\mathbf{X}_j\}_{j=N_s+1}^{N_s+N_t}$, which lacks labels and follows a different distribution, giving rise to the map $F_t : \mathcal{X}_t \to \mathcal{Y}_t$. Although the source and target datasets share the same label space ($\mathcal{Y}_s = \mathcal{Y}_t$), their joint distribution differ, *i.e.,* $P(\mathcal{X}_s, \mathcal{Y}_s) \neq P(\mathcal{X}_t, \mathcal{Y}_t)$. Such difference is referred to as *covariate domain shift* [31].

State Space Model (SSM) Formulation. SSM-based models [10, 14] map a 1D input sequence $\mathbf{x}(t) \in \mathbb{R}$ to an output $\mathbf{y}(t) \in \mathbb{R}$ through a learnable hidden state $\mathbf{h}(t) \in \mathbb{R}^N$, governed by a linear dynamical system:

$$\mathbf{h}'(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{x}(t), \quad \mathbf{y}(t) = \mathbf{C}\mathbf{h}(t), \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$, $\mathbf{B} \in \mathbb{R}^{N\times 1}$, and $\mathbf{C} \in \mathbb{R}^{1\times N}$ are learnable parameters. To integrate this formulation into deep learning, it is discretized using the zero-order hold (ZOH) method with step

size $\Delta \in \mathbb{R}$, resulting in the discrete recurrence:

$$\mathbf{h}(t) = \overline{\mathbf{A}}\mathbf{h}(t-1) + \overline{\mathbf{B}}\mathbf{x}(t), \quad \mathbf{y}(t) = \mathbf{C}\mathbf{h}(t), \quad \texttt{where} \quad \overline{\mathbf{A}} = \exp(\Delta\mathbf{A}), \quad \overline{\mathbf{B}} \approx \Delta\mathbf{B}. \tag{2}$$

Unrolling this recurrence over time yields a 1D convolution with kernel $\overline{\mathbf{K}} \in \mathbb{R}^L$, where $L \in \mathbb{N}$ denotes the kernel length. This form enables efficient parallel computation during training.

$$\mathbf{y} = \mathbf{x} \odot \overline{\mathbf{K}}, \quad \overline{\mathbf{K}} = \left(\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, \ldots, \mathbf{C}\overline{\mathbf{A}}^{L-1}\overline{\mathbf{B}}\right). \tag{3}$$

SS2D for 2D Visual Processing. To extend 1D state-space models to visual data, VMamba introduces the SS2D module, which reshapes an input image $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$ into a sequence of T non-overlapping patches $\{\mathbf{x}_t\}_{t=1}^T$. As shown in Fig. 1, these patches are traversed in a predefined causal order determined by one of four canonical directions using the Cross-Scan module: left-to-right (a), top-to-bottom (b), right-to-left (c), and bottom-to-top (d). For each direction, the patch sequence is processed recurrently by a shared, discretized 1D SSM, ensuring causality and directional consistency. The outputs from all four scans are then aggregated using the Cross-Merge module.

3.2 Traversal Permutation

To address the limitations of VMamba under distribution shifts, we propose a TTA strategy that leverages VMamba's intrinsic directional traversal mechanism. We define a set of traversal permutations

$$\mathcal{P} = \{\pi_1, \pi_2 \ldots, \pi_N\}, \tag{4}$$

where each π_i represents a unique ordering of the four canonical scan directions, with $\pi_1 = \{a, b, c, d\}$ corresponding to the original ordering used in VMamba (see Fig. 1 for the definition of directions a, b, c and d). We assess the predictive confidence of the model under each traversal permutation $\pi_i \in \mathcal{P}$ by computing the *Shannon entropy* of its output distribution. Permutations are then ranked in ascending order of entropy, and the top-K permutations with the lowest entropy are selected (the offline mode of Figure 1):

$$\mathcal{P}_{\texttt{selected}} = \{\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_K}\}. \tag{5}$$

During the adaptation phase, for each selected permutation π_{i_k}, $k = 1, \ldots, K$, the input is processed according to the corresponding traversal order. Denoting $p(\mathbf{X};\, \pi_{i_k}) \in [0,1]^C$ as the output of the model for an image $\mathbf{X}$ when using traversal permutation π_{i_k}, we compute a pseudo-label $\hat{y}_k$ by taking the class with maximum predicted probability:

$$\hat{y}_k = \underset{c\in\{1,\ldots,C\}}{\arg\max} \left[p(\mathbf{X};\, \pi_{i_k})\right]_c. \tag{6}$$

The Mamba-specific parameters are then updated to minimize the average cross-entropy (*log loss*) between the model prediction and the corresponding pseudo-label over each target sample in batch $\mathcal{B}$:

$$\boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}} -\frac{1}{|\mathcal{B}|}\sum_{\mathbf{X}\in\mathcal{B}} \log\left[p(\mathbf{X};\, \pi_{i_k})\right]_{\hat{y}_k}. \tag{7}$$

This is achieved using back-propagation as in standard methods. After optimization, the adapted parameter set $\boldsymbol{\theta}_k$ is cached, allowing a subsequent ensemble-based aggregation at inference.

Our method processes the same image through multiple directional permutations, enabling VMamba to exploit complementary causal views of the input. These distinct trajectories expose the model to both global consistency (identical token set) and local variation (different hidden-state evolutions), which helps find a flatter minima offering a stronger out-of-distribution (OOD) generalization [18]. Crucially, re-ordering prevents domain-specific artifacts from always entering the recurrence at the same time step. If an artifact $\boldsymbol{\varepsilon}$ is embedded in a corrupted patch $\mathbf{x}_{t_{\boldsymbol{\varepsilon}}}$, then under the default traversal π_1, this patch always appears at time step $t = t_{\boldsymbol{\varepsilon}}$ in the processing sequence. The hidden state update at that step becomes:

$$\mathbf{h}^{(1)}(t_{\boldsymbol{\varepsilon}}) = f\left(\mathbf{h}^{(1)}(t_{\boldsymbol{\varepsilon}} - 1),\, \mathbf{x}_{t_{\boldsymbol{\varepsilon}}} + \boldsymbol{\varepsilon}\right). \tag{8}$$

Since the hidden state $\mathbf{h}(t)$ is recurrent and accumulates over time, injecting $\boldsymbol{\varepsilon}$ early in the sequence allows its influence to propagate through many subsequent updates. Therefore, the impact of the artifact depends heavily on the position of $t_{\boldsymbol{\varepsilon}}$ within the sequence. In our method, we apply a traversal

permutation π_{i_k}, which changes the order in which patches are processed. Under π_{i_k}, the corrupted patch $\mathbf{x}_{t_\varepsilon}$ appears at time step $t = \pi_{i_k}(t_\varepsilon)$, leading to the updated hidden state:

$$\mathbf{h}^{(i)}\left(\pi_{i_k}(t_\varepsilon)\right) = f\left(\mathbf{h}^{(i)}\left(\pi_{i_k}(t_\varepsilon) - 1\right), \mathbf{x}_{t_\varepsilon} + \boldsymbol{\varepsilon}\right). \tag{9}$$

Across different permutations, the corrupted patch enters the sequence at varying time steps. This shifts the effect of artifact to different hidden states, breaking its consistent influence pattern. During test-time adaptation, these variations allow the model to learn diverse responses, and the subsequent weight averaging of adapted models helps suppress artifact-induced bias.

3.3 Leveraging Traversal Permutations Through Weight Averaging

To improve both stability and generalization under distribution shifts, we aggregate the adapted models obtained from the top-K traversal permutations. During the evaluation phase, we utilize the averaged weights

$$\bar{\theta} = \frac{1}{K}\sum_{k=1}^{K}\theta_k \tag{10}$$

and evaluate the model using the default traversal path π_1. In this setting, each θ_k represents an adapted model under a distinct traversal permutation. This results in a single, consolidated model that captures the benefits of multi-directional adaptation.

This simple yet effective averaging strategy draws inspiration from the notion of *flat minima* in the loss landscape [43, 18], where parameter configurations that lie in flat regions tend to exhibit greater generalization and robustness to perturbations. Weight averaging encourages convergence to a region in parameter space with low curvature, thereby reducing sensitivity to input variations and improving performance under distribution shifts.

To further illustrate this point, Figure 2 represents the test loss surface over model parameters under Gaussian noise corruption from the ImageNet-C dataset. Each triangle marks a model adapted via a different traversal permutation, while the central cross denotes the weight-averaged model. The axes depict linear interpolations between parameter sets, offering a 2D view of the high-dimensional landscape. This visualization demonstrates how different traversal permutations lead to diverse optima, and how weight averaging converges toward a smoother, lower-loss region, enhancing robustness and generalization.

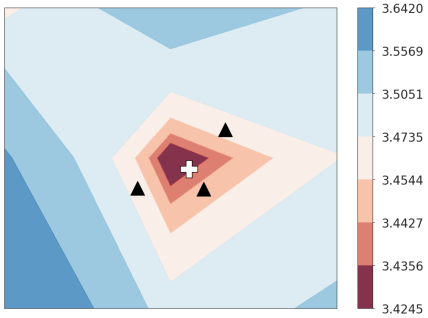

Figure 2: Loss surface of model parameters.

Unlike traditional TTA approaches that rely on data augmentation, external source data, or auxiliary objectives to generate diverse model variants for weight averaging, our approach leverages the internal architecture of VMamba, specifically, its canonical traversal mechanism. By permuting traversal scans, we induce structural variability that yields diverse perspectives of the same input, enabling effective adaptation without external augmentation.

4 Experiments

In this section, we present a comprehensive evaluation of our proposed method across seven benchmark datasets. We begin by describing the datasets used for evaluation, followed by implementation details and the baselines employed. We then report our main results and conclude with a set of ablation studies to analyze key components of our approach.

4.1 Datasets

We evaluate the performance and generalization of our method across a wide range of TTA benchmarks, covering corruption robustness and domain generalization. For corruption-based robustness, we use CIFAR10-C [47], CIFAR100-C [47], and ImageNet-C [47], which apply 15 corruption types (e.g., noise, blur, weather, digital artifacts) at five severity levels. These datasets scale from 10

(CIFAR10-C) to 100 (CIFAR100-C) to 1000 (ImageNet-C) classes, enabling systematic analysis of model degradation.

For domain generalization, we assess on PACS [48], ImageNet-S [49], ImageNet-V2 [50], and ImageNet-R [51]. PACS includes four visual styles with seven shared classes, using leave-one-domain-out evaluation. ImageNet-S features sketch-style abstractions; ImageNet-V2 provides a cleaner, independently collected test set; and ImageNet-R introduces style shifts across 200 classes with diverse artistic renditions. Together, these benchmarks test adaptability to unseen distributions, styles, and abstractions.

4.2 Implementation Details

We perform adaptation exclusively on the Mamba-specific state space parameters within the SS2D module of each VMamba block, while keeping the rest of the model frozen. The base VMamba model is trained with batch normalization layers. The adaptation is guided using a pseudo-labeling strategy, where confident model predictions are treated as supervision targets in a cross-entropy loss. This enables self-supervised adaptation without requiring access to ground-truth labels. The adaptation proceeds in an online manner, with model updates applied sequentially (i.e., without resetting the weights to their pre-adaptation values). Optimization is performed using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 128, ensuring consistent dynamics and fair comparison across benchmarks. All experiments were conducted using a single NVIDIA $A6000$ GPU.

4.3 Baselines

We compare our method against several state-of-the-art TTA approaches. Source Only refers to the performance of VMamba without any adaptation. ETA [21] minimizes an entropy loss modulated by a sample-adaptive weighting mechanism, updating only the affine parameters of normalization layers. LAME [31] enforces smooth decision boundaries by introducing Laplacian regularization over the model predictions. SAR [23] promotes robustness by explicitly optimizing for flat minima using sharpness-aware minimization. SHOT [19] aligns the classifier by minimizing entropy while encouraging confident and diverse predictions, using a fixed feature extractor. Tent [20] adapts the model by updating only the affine parameters of normalization layers via entropy minimization.

We evaluate two variants of our approach: TRUST naive, which adapts only the Mamba-specific SS2D parameters, and TRUST, which further enhances robustness by averaging over multiple traversal permutations during adaptation.

4.4 Main Results

Table 1 presents the Top-1 accuracy under the highest corruption severity (level 5) for CIFAR10-C, CIFAR100-C, and ImageNet-C datasets. TRUST consistently outperforms all baselines across the three datasets, demonstrating strong generalization under severe distribution shifts. It also surpasses its naive variant, underscoring the effectiveness of permutation-based strategy in enhancing robustness.

CIFAR10-C. TRUST achieves a mean accuracy of 77.5%, outperforming both Tent and SHOT by margins of 11.0% and 10.7%, respectively. On challenging corruptions, TRUST yields the largest improvements over SHOT, such as elastic (7.5%), glass blur (10.8%), and motion blur (7.4%). Compared to its naive variant, TRUST provides a further 3.3% boost. These results highlight the benefits of traversal diversity in enhancing corruption robustness.

CIFAR100-C. On a more fine-grained benchmark, TRUST attains a mean accuracy of 54.3%, exceeding SHOT and SAR by 12.3% and 12.4%, respectively. The largest absolute improvements over SHOT are observed under challenging corruptions such as impulse noise (23.2%), contrast (18.9%), and shot noise (16.1%), demonstrating the robustness of our method under severe distribution shifts. It also improves upon TRUST naive by 4.5%.

ImageNet-C. TRUST achieves a strong mean accuracy of 56.1% on the large-scale ImageNet-C benchmark, outperforming both Tent and SHOT by 14.4%, and TRUST naive by 2.7%. On challenging corruptions, TRUST yields the largest improvements over SHOT, such as glass blur (14.8%), elastic (19.3%), and jpeg compression (14.4%), reflecting the advantage of our method in mitigating both spatial distortions and digital artifacts.

	Method	gaussian noise	shot noise	impulse noise	defocus blur	glass blur	motion blur	zoom blur	frost	snow	fog	brightness	contrast	elastic	pixelate	jpeg compression	Mean
CIFAR10-C	Source only	46.8	48.4	45.0	73.5	52.6	73.0	78.7	71.8	75.8	77.3	85.7	69.6	63.7	67.9	59.0	65.9
	ETA [21]	46.7	48.3	44.8	73.5	52.6	73.0	78.6	75.7	71.4	77.2	85.7	69.6	63.7	67.9	59.0	65.8 (↓0.1)
	LAME [31]	46.7	48.3	44.8	73.5	52.6	73.0	78.6	71.8	75.8	77.2	85.7	69.6	63.7	67.9	59.0	65.9
	SAR [23]	47.7	49.5	46.2	74.3	53.4	73.8	79.1	72.5	76.5	78.0	86.1	70.8	64.5	68.9	60.0	66.8 (↑0.9)
	SHOT [19]	47.8	49.7	46.3	74.3	53.7	74.0	79.3	72.6	76.6	78.1	86.3	70.7	64.5	68.9	59.9	66.8 (↑0.9)
	Tent [20]	47.3	49.2	45.8	74.2	53.1	73.7	79.1	72.2	76.3	77.9	86.1	70.4	64.3	68.7	59.6	66.5 (↑0.6)
	TRUST naive	58.9	61.8	62.0	79.8	60.9	79.1	82.6	80.5	81.8	83.6	88.8	81.8	70.3	75.1	66.0	74.2 (↑8.3)
	TRUST	**63.1**	**67.8**	**70.3**	**81.0**	**64.5**	**81.4**	**85.0**	**83.2**	**85.4**	**85.8**	**90.1**	**85.7**	**72.1**	**79.1**	**68.6**	**77.5 (↑11.6)**
CIFAR100-C	Source only	21.0	22.1	18.3	50.6	27.7	51.0	56.2	45.3	50.6	52.4	65.3	43.2	39.0	41.7	33.4	41.2
	ETA [21]	21.2	22.3	18.7	50.8	27.8	51.2	56.3	45.4	50.8	52.7	65.5	43.5	39.2	42.0	33.7	41.4 (↑0.2)
	LAME [31]	21.0	22.1	18.3	50.6	27.7	51.0	56.2	45.3	50.6	52.5	65.4	43.2	39.0	41.7	33.4	41.2
	SAR [23]	21.9	22.8	19.3	51.1	28.2	51.5	56.7	46.4	51.4	53.1	65.8	44.2	39.9	42.8	34.2	41.9 (↑0.7)
	SHOT [19]	21.9	22.9	19.1	51.4	28.3	51.8	56.8	46.3	51.4	53.3	66.0	44.2	39.8	42.7	34.3	42.0 (↑0.8)
	Tent [20]	21.6	22.6	18.9	51.1	28.2	51.5	56.7	46.2	51.1	53.1	65.8	44.0	39.7	42.5	34.0	41.8 (↑0.6)
	TRUST naive	32.1	32.8	34.1	56.9	35.4	57.2	61.6	54.6	57.8	60.1	69.6	55.6	46.6	50.8	41.0	49.8 (↑8.6)
	TRUST	**37.8**	**38.9**	**42.3**	**60.9**	**36.6**	**60.8**	**65.4**	**59.0**	**62.2**	**64.5**	**71.7**	**63.1**	**50.3**	**56.6**	**44.9**	**54.3 (↑13.1)**
ImageNet-C	Source only	24.3	26.1	25.1	22.2	23.2	35.4	43.2	49.3	48.4	56.9	70.0	26.8	45.1	43.7	41.4	38.7
	ETA [21]	26.4	28.4	27.2	23.5	24.6	37.2	45.1	50.8	51.0	58.8	70.6	29.1	47.7	46.9	45.0	40.8 (↑2.1)
	LAME [31]	24.3	26.1	25.1	22.2	23.2	35.4	43.2	49.3	48.4	56.9	70.0	26.8	45.1	43.7	41.4	38.8 (↑0.1)
	SAR [23]	26.5	29.2	28.0	24.5	25.3	37.4	45.1	51.0	51.7	59.1	70.5	31.5	48.2	48.6	46.3	41.5 (↑2.8)
	SHOT [19]	28.0	30.1	28.8	25.0	26.0	38.0	45.7	51.0	51.5	59.1	70.6	30.2	48.4	47.8	45.8	41.7 (↑3.0)
	Tent [20]	27.8	30.0	28.8	24.9	25.9	38.0	45.5	51.0	51.3	59.1	70.6	30.0	48.2	47.8	45.7	41.7 (↑3.0)
	TRUST naive	43.4	45.6	44.9	38.3	36.6	53.0	54.9	57.1	60.2	66.0	72.2	50.2	59.0	61.1	58.5	53.4 (↑14.7)
	TRUST	**46.8**	**49.4**	**48.5**	**42.8**	**40.8**	**57.1**	**57.9**	**57.3**	**61.7**	**66.8**	**71.9**	**54.9**	**61.4**	**63.6**	**60.2**	**56.1 (↑17.4)**

Table 1: Top-1 classification accuracy (%) under various corruption types on CIFAR10-C, CIFAR100-C, and ImageNet-C datasets. Increases/decreases in mean accuracy compared to performing no adaptation (Source only) is highlighted in green/red color.

In addition to corruption-specific robustness, our method demonstrates strong generalization to broader distribution shifts, as summarized in Table 2. On **ImageNet-S**, our method achieves 41.5% accuracy, outperforming SHOT by a substantial margin of 8.9%, and providing a modest 0.4% improvement over the naive variant. For **ImageNet-V2**, TRUST attains 64.0%, exceeding SHOT by 1.6% and providing a 0.6% gain over its naive variant. The most significant boost is observed on the **ImageNet-R** benchmark, which features real-world renditions of ImageNet categories. Here, our method achieves 44.3%, surpassing both Tent and SHOT by 12.4%, and outperforming TRUST naive by 4.6%. On the **PACS** dataset, our model reaches 69.9% accuracy, offering consistent improvements over SHOT and Tent by 2.5%, and over the naive variant by 2.8%. These consistent gains across varied benchmarks highlight the adaptability of our permutation-based strategy, which not only improves corruption robustness but also scales effectively to domain generalization tasks.

Method	CIFAR10-C	CIFAR100-C	ImageNet-C	ImageNet-S	ImageNet-V2	ImageNet-R	PACS
Source only	65.9	41.2	38.7	31.4	62.2	31.3	66.7
ETA [21]	65.8 (↓0.1)	41.4 (↑0.2)	40.8 (↑2.1)	31.4	62.2	31.3	66.7
LAME [31]	65.9	41.2	38.8 (↑0.1)	31.4	62.2	31.3	66.7
SAR [23]	66.8 (↑0.9)	41.9 (↑0.7)	41.5 (↑2.8)	32.6 (↑1.2)	62.4 (↑0.2)	32.0 (↑0.7)	67.3 (↑0.6)
SHOT [19]	66.8 (↑0.9)	42.0 (↑0.8)	41.7 (↑3.0)	32.6 (↑1.2)	62.4 (↑0.2)	31.9 (↑0.6)	67.4 (↑0.7)
Tent [20]	66.5 (↑0.6)	41.8 (↑0.6)	41.7 (↑3.0)	32.5 (↑1.1)	62.3 (↑0.1)	31.9 (↑0.6)	67.4 (↑0.7)
TRUST naive	74.2 (↑8.3)	49.8 (↑8.6)	53.4 (↑14.7)	41.1 (↑9.7)	63.4 (↑1.2)	39.7 (↑8.4)	67.1 (↑0.4)
TRUST	**77.5 (↑11.6)**	**54.3 (↑13.1)**	**56.1 (↑17.4)**	**41.5 (↑10.1)**	**64.0 (↑1.8)**	**44.3 (↑13.0)**	**69.9 (↑3.2)**

Table 2: Top-1 classification accuracy (%) across datasets. For CIFAR10-C, CIFAR100-C, and ImageNet-C, values are averaged over all corruptions; for ImageNet-S, V2, R, and PACS, they reflect test set accuracy. Increases/decreases in mean accuracy compared to performing no adaptation (Source only) is highlighted in green/red color.

4.5 Ablation Study

In this section, we present a comprehensive ablation study on the CIFAR10-C dataset to assess the impact of key factors on the performance of our model. Specifically, we analyze the effects of batch size, augmentation types, number of traversal permutations, number of adaptation iterations, aggregation strategies, effect of traversing type in evaluation and computational overhead.

Batch Size. Figure 3 demonstrates that TRUST maintains a consistent advantage over Tent across different batch sizes. Even at smaller sizes such as 16 and 32, it delivers an accuracy improvement of around 8%. This indicates that our approach remains reliable and effective, even when operating with limited data per batch.

Augmentation Impact. To evaluate whether standard data augmentations can serve as effective alternatives to our traversal permutation strategy, we compare TRUST against common augmentations including rotation, random cropping, and color jitter. As shown in Figure 4, these traditional augmentations yield only modest improvements over the baseline source-only model on CIFAR10-C, with accuracies ranging from 65.9% to 68.3%. In contrast, our method achieves a substantial performance boost, reaching 77.5%, which is significantly higher than all augmentation baselines. This indicates that simple augmentations fail to sufficiently address distribution shifts.

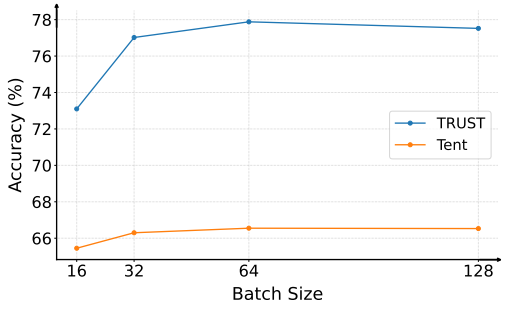

Figure 3: Accuracy comparison between TRUST and Tent across varying batch sizes on CIFAR10-C dataset.

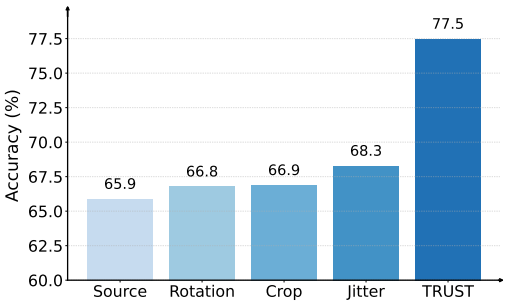

Figure 4: Performance comparison between standard augmentations and TRUST on CIFAR10-C dataset.

Number of Traversal Permutations. Figure 5 shows how accuracy varies with the number of traversal permutations across CIFAR10-C, CIFAR100-C, and ImageNet-C. Overall, increasing the number of permutations consistently improves performance on all datasets. On CIFAR10-C, accuracy rises from 75.6% (2 permutations) to 77.6% (8 permutations). For CIFAR100-C, the gain is from 51.7% to 54.7%, and for ImageNet-C, from 54.4% to a peak of 56.1% at six permutations before slightly dropping to 55.5% at eight. These results confirm that incorporating multiple traversal permutations enhances robustness under distribution shifts. Although accuracy improves with more permutations, gains diminish after six, suggesting that six permutations strike a good balance between performance and computational efficiency. For a breakdown of memory overhead, see Figure 9.

Number of Adaptation Iterations. Figure 6 illustrates that accuracy improves progressively with an increasing number of iterations. Despite the potential for further improvement with more iterations, we chose to use only a single iteration in subsequent experiments to enable faster adaptation without compromising on competitive performance.

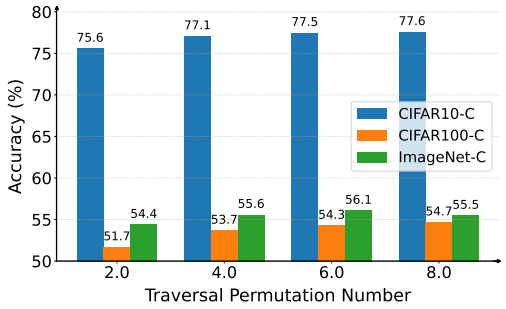

Figure 5: Effect of traversal permutation count on accuracy across three datasets.

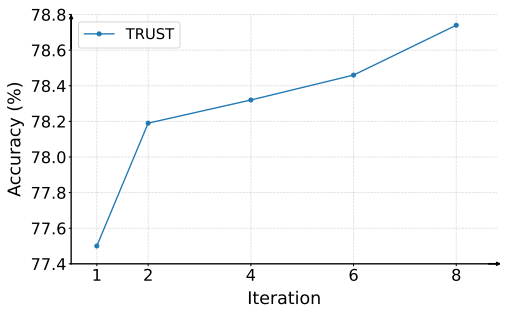

Figure 6: Model performance across adaptation iterations on CIFAR10-C dataset.

Aggregation Strategy. We evaluate two baseline strategies for aggregation across traversal permutations: (1) an ensemble approach that averages model predictions from independently processed traversal orders, and (2) a repetition baseline, where the same traversal permutation is applied k times,

followed by weight averaging. As shown in Figure 7, both baselines yield lower performance on CIFAR10-C, with ensemble achieving 68.1% and repetition 69.6%, compared to 75.6% with our method.

The ensemble variant offers traversal diversity, but because its individual models are never weight-averaged, it lacks parameter alignment. The repetition baseline, in contrast, repeats the same traversal and updates the model sequentially; the weights, therefore, remain aligned, yet no traversal diversity is introduced. Both baselines perform worse than our method, demonstrating that neither unaligned traversal diversity (ensemble) nor aligned repetitions without diversity (repetition) is sufficient on its own. By combining these two ingredients, traversal diversity and parameter alignment, our approach achieves better generalization under distribution shift.

Effect of Traversal Permutation in Evaluation. Figure 8 illustrates how accuracy varies with different traversal permutations during evaluation. We observe that the choice of permutation impacts the performance. The default permutation used in the original VMamba architecture, "$abcd$", achieves the highest accuracy at 77.5%, while others, such as "$badc$", yield lower performance (e.g., 71.6%). Given this sensitivity, we adopt the "$abcd$" traversal permutation, on which the model was trained, for all evaluations to ensure consistency and leverage the strongest baseline.

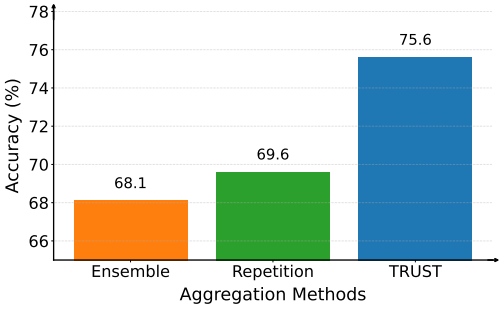

Figure 7: Accuracy comparison of different aggregation strategies on CIFAR10-C dataset.

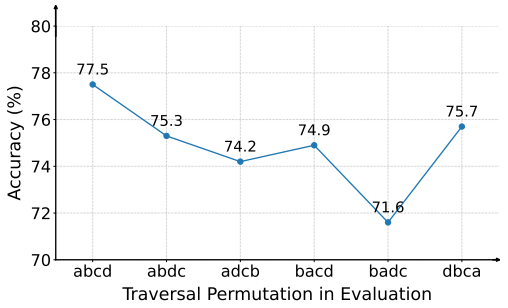

Figure 8: Impact of traversal permutation during evaluation on CIFAR10-C dataset.

Computational Overhead. We evaluate GPU memory usage as a function of the number of traversal permutations used during parallel adaptation. Since only the SS2D blocks are updated, we instantiate one SS2D block per traversal while sharing the rest of the network. Traversals are batched and routed to their corresponding SS2D blocks, and their outputs are then concatenated. This design minimizes computational overhead. As shown in Figure 9, memory usage increases moderately, 1.1×, 1.3×, 1.5×, and 1.7× for 2, 4, 6, and 8 traversals, respectively, relative to the 2.1 GB baseline (measured with batch size 1). In sequential mode, forward and backward passes are performed separately for each traversal. We measure a per-traversal time of approximately 135 ms (with the batch size of 1), leading to a total cost of $K \times 135$ ms for K traversals. As expected, memory usage remains nearly constant, while adaptation time grows linearly. Conversely, in parallel mode, adaptation time remains nearly constant, with moderate memory overhead, offering a practical trade-off given the performance gains. For more details, refer to the Supplementary Material.

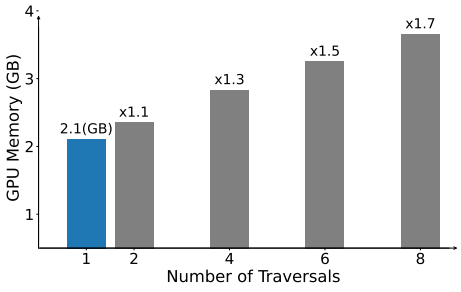

Figure 9: GPU memory usage across traversals.

5 Conclusion

In this paper, we introduced TRUST, a test-time adaptation strategy specifically designed for vision-oriented state-space models. Our method addresses two key challenges in VMamba under distribution shift: (1) the strong inductive bias introduced by fixed traversal scans, and (2) the accumulation of domain-specific artifacts in hidden states during sequential processing. To overcome these issues, TRUST exploits directional traversal mechanism of VMamba to generate complementary causal views of the input, performing adaptation based on each. The resulting models are aggregated via

weighted parameter averaging, promoting convergence toward flatter, more robust regions in the loss landscape. Experiments on seven standard benchmarks show that TRUST consistently outperform popular TTA models including Tent, SHOT, and SAR, in image classification tasks. Our method does not yet generalize to the medical domain. Future work will adapt TRUST to address data scarcity, modality differences, and privacy constraints in medical settings.

Acknowledgements

We appreciate the computational resources and support provided by Compute Canada and the Digital Research Alliance of Canada.

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International conference on learning representations*, 2014.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[4] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[5] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, G Heigold, S Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[7] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[8] Xiaosong Zhang, Yunjie Tian, Lingxi Xie, Wei Huang, Qi Dai, Qixiang Ye, and Qi Tian. Hivit: A simpler and more efficient design of hierarchical vision transformer. In *The Eleventh International Conference on Learning Representations*, 2023.

[9] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[10] Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2024.

[11] Tri Dao, Daniel Y Fu, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry Hungry Hippos: Towards language modeling with state space models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023.

[12] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. In *ICLR*, 2023.

[13] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. VMamba: Visual state space model. *Advances in neural information processing systems*, 37:103031–103063, 2024.

[14] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

[15] Sahar Dastani, Ali Bahri, Moslem Yazdanpanah, Mehrdad Noori, David Osowiechi, Gustavo Adolfo Vargas Hakim, Farzad Beizaee, Milad Cheraghalikhani, Arnab Kumar Mondal, Herve Lombaert, et al. Spectral state space model for rotation-invariant visual representation learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23881–23890, 2025.

[16] Shaocong Long, Qianyu Zhou, Xiangtai Li, Xuequan Lu, Chenhao Ying, Yuan Luo, Lizhuang Ma, and Shuicheng Yan. DGMamba: Domain generalization via generalized state space model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3607–3616, 2024.

[17] Nitish Shirish Keskar, Jorge Nocedal, Ping Tak Peter Tang, Dheevatsa Mudigere, and Mikhail Smelyanskiy. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[18] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. SWAD: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418, 2021.

[19] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, pages 6028–6039. PMLR, 2020.

[20] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[21] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16888–16905. PMLR, 17–23 Jul 2022.

[22] Jungsoo Lee, Debasmit Das, Jaegul Choo, and Sungha Choi. Towards open-set test-time adaptation utilizing the wisdom of crowds in entropy minimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16380–16389, October 2023.

[23] 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 *Internetional Conference on Learning Representations*, 2023.

[24] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15922–15932, 2023.

[25] Sachin Goyal, Mingjie Sun, Aditi Raghunanthan, and Zico Kolter. Test-time adaptation via conjugate pseudo-labels. *Advances in Neural Information Processing Systems*, 2022.

[26] Taesik Gong, Yewon Kim, Taeckyung Lee, Sorn Chottananurak, and Sung-Ju Lee. SoTTA: Robust test-time adaptation on noisy data streams. *Advances in Neural Information Processing Systems*, 36, 2024.

[27] Yongcan Yu, Lijun Sheng, Ran He, and Jian Liang. STAMP: Outlier-aware test-time adaptation with stable memory replay. In *European Conference on Computer Vision*, pages 375–392, 2024.

[28] Zhengqing Gao, Xu-Yao Zhang, and Cheng-Lin Liu. Unified entropy optimization for open-set test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23975–23984, June 2024.

[29] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022.

[30] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 295–305, 2022.

[31] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8344–8353, 2022.

[32] Haopeng Sun, Lumin Xu, Sheng Jin, Ping Luo, Chen Qian, and Wentao Liu. Program: Prototype graph model based pseudo-label learning for test-time adaptation. In *The Twelfth International Conference on Learning Representations*, 2024.

[33] Minguk Jang, Sae-Young Chung, and Hye Won Chung. Test-time adaptation via self-training with nearest neighbor information. In *The Eleventh International Conference on Learning Representations*, 2024.

[34] Shuaicheng Niu, Chunyan Miao, Guohao Chen, Pengcheng Wu, and Peilin Zhao. Test-time model adaptation with only forward passes. In *International Conference on Machine Learning*, pages 38298–38315. PMLR, 2024.

[35] 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. *Advances in Neural Information Processing Systems*, 35:14274–14289, 2022.

[36] Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El Saddik, and Eric Xing. Efficient test-time adaptation of vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14162–14171, 2024.

[37] 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 *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[38] Gustavo A 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 *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 7092–7101, February 2025.

[39] Mona Schirmer, Dan Zhang, and Eric Nalisnick. Temporal test-time adaptation with state-space models. *arXiv preprint arXiv:2407.12492*, 2024.

[40] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[41] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with MixStyle. In *International Conference on Learning Representations*, 2021.

[42] Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Gustavo A Vargas Hakim, David Osowiechi, Moslem Yazdanpanah, Ismail Ben Ayed, and Christian Desrosiers. FDS: Feedback-guided domain synthesis with multi-source conditional diffusion models for domain generalization. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 8493–8503, February 2025.

[43] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

[44] Ali Bahri, Moslem Yazdanpanah, Mehrdad Noori, Sahar Dastani, Milad Cheraghalikhani, David Osowiechi, Farzad Beizaee, Gustavo A Vargas Hakim, Ismail Ben Ayed, and Christian Desrosiers. Test-time adaptation in point clouds: Leveraging sampling variation with weight averaging. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 266–275. IEEE, 2025.

[45] Moslem Yazdanpanah, Ali Bahri, Mehrdad Noori, Sahar Dastani, Gustavo Adolfo Vargas Hakim, David Osowiechi, Ismail Ben Ayed, and Christian Desrosiers. Purge-Gate: Backpropagation-free test-time adaptation for point clouds classification via token purging. *arXiv preprint arXiv:2509.09785*, 2025.

[46] Ali Bahri, Moslem Yazdanpanah, Sahar Dastani, Mehrdad Noori, Gustavo Adolfo Vargas Hakim, David Osowiechi, Farzad Beizaee, Ismail Ben Ayed, and Christian Desrosiers. SMART-PC: Skeletal model adaptation for robust test-time training in point clouds. *arXiv preprint arXiv:2505.19546*, 2025.

[47] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[48] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

[49] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. *Advances in neural information processing systems*, 32, 2019.

[50] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

[51] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8340–8349, 2021.

[52] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[53] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 891–898, 2014.

[54] Mehrdad Noori, David Osowiechi, Gustavo Adolfo Vargas Hakim, Ali Bahri, Moslem Yazdanpanah, Sahar Dastani, Farzad Beizaee, Ismail Ben Ayed, and Christian Desrosiers. Test-time adaptation of vision-language models for open-vocabulary semantic segmentation. *arXiv preprint arXiv:2505.21844*, 2025.

[55] Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. In *European Conference on Computer Vision*, pages 207–223. Springer, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract is the summary of the content of the paper and the introduction is clearly state the main contributions of the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In the last part of the conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide details of implementation and experiments in section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The data is available online and we will release the code with the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Provided in section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Experiments were run with a single seed. We plan to include multi-seed results in future revisions.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides details on the hardware used, and reports a computational overhead analysis in the ablation section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, we do.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: No relevant.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No relevant.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We followed the protocol.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The paper does not introduce any new datasets or benchmarks. All experiments are conducted on existing public datasets using open-source baselines.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Not relevant.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [No]

Justification: This work does not use LLMs in the core methodology, scientific rigorousness, or originality of the research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

TRUST: Test-Time Refinement using Uncertainty-Guided SSM Traverses - Appendix

A Pseudo-code

In this section, we give the pseudo-code for our proposed test-time adaptation method, TRUST. This pseudo-code provides a concise summary of the key steps involved in our approach, offering a high-level abstraction of the implementation. Algorithm 1 outlines the overall TRUST procedure for test-time adaptation. For each corruption in the evaluation set, the model is first reset to its original (pre-adaptation) weights. The input x is then processed using the FORWARD_AND_ADAPT function (Algorithm 2), which outputs the adapted prediction and a list of model parameter sets θ, each corresponding to a different traversal permutation used during adaptation. These parameters are averaged to obtain the final adapted parameters, θ_{final}, which are loaded back into the model. The final prediction is then computed by re-evaluating the model on the same input x.

Algorithm 1 TRUST

for each corruption **do**
    MODEL.RESET
    $(out, \theta) \leftarrow$ MODEL.FORWARD_AND_ADAPT(x)
    $\theta_{\text{final}} \leftarrow$ MEAN(θ)
    MODEL.LOAD_STATE_DICT(θ_{final})
    $out \leftarrow$ MODEL.EVALUATE(x)
end for

Algorithm 2 defines the FORWARD_AND_ADAPT function. For each traversal permutation $\pi_{i_k} \in \mathcal{P}$, the model performs a forward pass with input x and permutation π_{i_k}. The cross-entropy loss between the model's prediction and the pseudo-labels is computed, and a gradient descent step is performed to update the model parameters. Each updated parameter state is stored in the list θ. After all permutations have been processed, the method returns the final output and the list of different model parameters.

Algorithm 2 FORWARD_AND_ADAPT(x)

$\theta \leftarrow$ empty list
for each $\pi_{i_k} \in \mathcal{P}$ **do**
    $out \leftarrow$ model(x, π_{i_k})
    $loss \leftarrow$ CE($out, pseudo\_labels$)
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
    θ.append(model.parameters)
end for
return (out, θ)

B Finetuning Process

The backbone model, VMamba, was originally trained on ImageNet-C, which contains 1,000 classes. To enable test-time use on corrupted datasets with different label sets, we fine-tune only the classifier layer on the clean version of each target dataset, keeping the rest of the model frozen. This procedure is applied to CIFAR10-C, CIFAR100-C, and PACS, using a learning rate of 5e-4 for 300 epochs. For PACS, which includes four domains (photo, art painting, cartoon, and sketch), we follow the standard protocol: one domain is held out for evaluation while training on the remaining three. Specifically, we use the photo domain as the held-out test set. For datasets such as ImageNet-S, ImageNet-V2, and ImageNet-R, which share the same label space as ImageNet, no fine-tuning is required.

C Efficient Parallel Implementation

We mentioned in the main paper (Computational Overhead section) that our method supports an efficient parallel adaptation strategy. Figure 10 provides a detailed diagram of this process. In parallel mode, we handle K traversal permutations simultaneously. A batch $\mathcal{B}$ is first split into K subsets, each corresponding to a different permutation π_{i_k}. Each subset is then passed to an independent SS2D TRUST Version block, where the SS2D parameters are adapted in parallel while the rest of the model remains shared across all paths, this design significantly reduces memory usage.

After adaptation, the outputs are concatenated back into a single batch, which allows for efficient GPU utilization. For evaluation, we perform a weight averaging step across all adapted SS2D modules, producing a single unified SS2D TRUST Version block. This averaged block is then used for inference on the full batch. By leveraging parallelism in both data and traversal space, our method achieves scalable and low-overhead test-time adaptation while preserving performance across permutations.

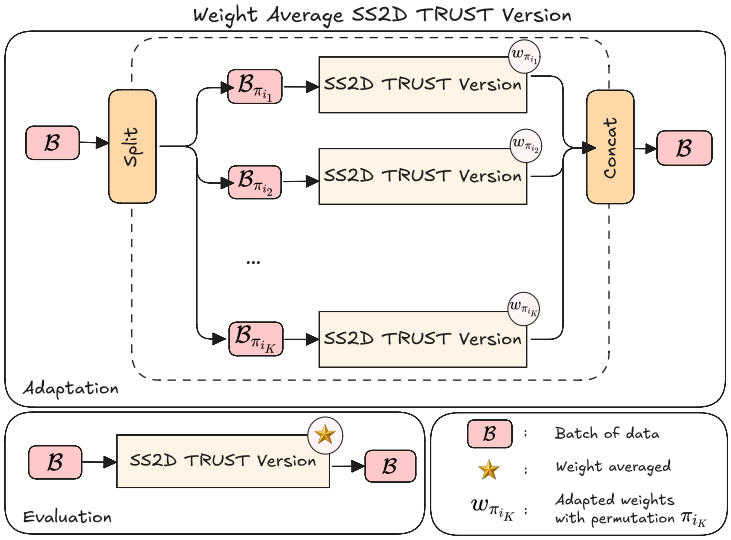

Figure 10: Detailed diagram of TRUST in Parallel mode.

D Standard Mode

Table 3 reports Top-1 accuracy under various corruption types at severity level 5 for CIFAR10-C, CIFAR100-C, and ImageNet-C in the standard setting, where the model is reset for each test batch. On CIFAR10-C, TRUST achieves the highest mean accuracy of 66.2%, surpassing Tent and SHOT by 0.2%. On CIFAR100-C, it reaches 41.7%, outperforming all baselines by at least 0.4%, and improving over its naive variant by 0.5%, underscoring the benefits of permutation-based refinement. On the large-scale ImageNet-C, TRUST attains 39.9%, exceeding Tent by 1.1%, and outperforming its naive counterpart by 1.2%. These results highlight the scalability and robustness of permutation-aware adaptation across diverse datasets. Table 3 evaluates performance under domain-level distribution shifts. TRUST achieves 31.6% on **ImageNet-S**, 62.3% on **ImageNet-V2**, and 31.5% on **ImageNet-R**. On the **PACS** benchmark, it obtains 71.3%, outperforming its naive variant by 4.6%. While gains are narrower in these less corrupted settings, TRUST consistently improves generalization, particularly under significant domain shifts.

E TRUST with Batch Norm Adaptation

In this experiment, we compare two adaptation strategies within our framework: updating only the BN layers versus updating the SS2D parameters. As shown in Table 4, our method outperforms Tent in both settings, by 3.1% when using BatchNorm adaptation and by 11.8% with SS2D adaptation. We opt to proceed with SS2D adaptation, as traversal permutations directly influence Mamba-specific parameters encoded in the SS2D blocks. Updating these parameters is therefore essential to fully capture the effects of traversal-based modifications.

Method	CIFAR10-C	CIFAR100-C	ImageNet-C	ImageNet-S	ImageNet-V2	ImageNet-R	PACS
Source only	65.9	41.2	38.7	31.4	62.2	31.3	66.7
ETA	65.8 (↓0.1)	41.2	38.8 (↑0.1)	31.4	62.2	31.4 (↑0.1)	66.7
LAME	65.9	41.2	38.8 (↑0.1)	31.4	62.2	31.4	66.7
SAR	66.0 (↑0.1)	41.3 (↑0.1)	38.8 (↑0.1)	31.4	62.2	31.4 (↑0.1)	67.1 (↑0.4)
SHOT	66.0 (↑0.1)	41.3 (↑0.1)	38.9 (↑0.2)	31.4	62.2	31.4 (↑0.1)	67.3 (↑0.6)
Tent	66.0 (↑0.1)	41.3 (↑0.1)	38.8 (↑0.1)	31.4	62.2	31.4 (↑0.1)	67.2 (↑0.5)
TRUST naive	65.9	41.2	38.9 (↑0.2)	31.5 (↑0.1)	62.3 (↑0.1)	31.5 (↑0.2)	66.8 (↑0.1)
TRUST	**66.2** (↑0.3)	**41.7** (↑0.5)	**39.9** (↑1.2)	**31.6** (↑0.2)	**62.3** (↑0.1)	**31.5** (↑0.2)	**71.3** (↑4.6)

Table 3: Top-1 classification accuracy (%) across datasets in standard setting. For CIFAR10-C, CIFAR100-C, and ImageNet-C, values are averaged over all corruptions; for ImageNet-S, V2, R, and PACS, they reflect test set accuracy. Increases/decreases in mean accuracy compared to performing no adaptation (Source only) is highlighted in green/red color.

	Method	gaussian	shot	impulse	defocus	glass	motion	zoom	frost	snow	fog	brightness	contrast	elastic	pixelate	jpeg	Mean
	Source only	24.3	26.1	25.1	22.2	23.2	35.4	43.2	49.3	48.4	56.9	70.0	26.8	45.1	43.7	41.4	38.7
BN	Tent	27.8	30.0	28.8	24.9	25.9	38.0	45.5	51.0	51.3	59.1	70.6	30.0	48.2	47.8	45.7	41.7
	TRUST	32.8	35.1	34.0	26.8	28.5	40.8	47.7	52.9	53.8	61.1	71.3	34.1	50.7	51.8	50.2	44.8 (↑3.1)
SS2D	Tent	29.6	31.7	34.7	25.1	22.0	45.7	44.8	46.0	55.8	62.2	69.8	32.5	52.8	56.8	54.3	44.3
	TRUST	46.8	49.4	48.5	42.8	40.8	57.1	57.9	57.3	61.7	66.8	71.9	54.9	61.4	63.6	60.2	56.1 (↑11.8)

Table 4: Comparison of adaptation strategies using BN and SS2D parameters on ImageNet-C dataset. Increases/decreases in mean accuracy compared to Tent is highlighted in green/red color.

F Mean Entropy of Different Traversal Permutations

As illustrated in Figure 11, varying the order of spatial traversals in VMamba leads to substantial differences in mean entropy across datasets, revealing the sensitivity of the model's internal representations to traversal patterns. Higher entropy values indicate less confident predictions, often aligning with reduced robustness under distributional shifts. This highlights the critical role of traversal selection in ensuring reliable adaptation. To offer a comprehensive analysis, we report entropy values for all permutations across CIFAR10-C, CIFAR100-C, ImageNet-C, ImageNet-S, ImageNet-V2, ImageNet-R, and PACS. Notably, the top-2, top-4, and top-6 traversal permutations with the lowest entropy, highlighted with green cross-hatching (i.e., diagonal lines overlaid on the heatmap cells), consistently reappear across datasets. This pattern suggests that certain traversal orders inherently yield more stable and confident model outputs, providing a principled foundation for selecting effective traversal subsets in our approach.

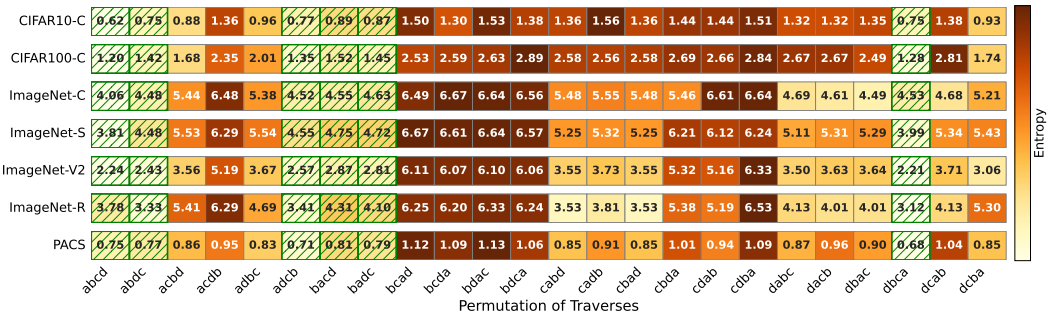

Figure 11: Mean entropy of different traversal permutation across seven benchmarks.

G Combination of TRUST with Augmentation

According to Figure 4 in the main paper, Jitter augmentation yields the highest performance among the augmentation strategies evaluated. Building on this, we examine the effectiveness of combining Jitter with our proposed method, TRUST, under two augmentation settings: (1) applying different Jitter augmentations for each traversal permutation, and (2) applying different Jitter augmentations per batch. The former setting achieves a modest improvement of 0.3% over the source-only baseline, while the latter results in a slightly higher gain of 1.8%. However, both improvements are substantially

lower than the performance boost achieved by the original TRUST, which yields an 11.6% increase. These results highlights that although augmentation contributes to performance, the core strength of TRUST lies in its ability to adapt representations based on traversal-specific dynamics rather than augmentation diversity alone.

Method	gaussian	shot	impulse	defocus	glass	motion	zoom	frost	snow	fog	brightness	contrast	elastic	pixelate	jpeg	Mean
Source only	46.8	48.4	45.0	73.5	52.6	73.0	78.7	71.8	75.8	77.3	85.7	69.6	63.7	67.9	59.0	65.9
TRUST + Jitter per permutation	53.6	54.2	54.0	63.8	48.0	69.3	77.1	75.4	75.2	77.2	85.8	76.6	59.1	66.5	57.8	66.2 (↑0.3)
TRUST + Jitter per batch	57.0	53.4	57.2	70.0	48.9	65.6	79.6	77.5	77.9	77.3	88.2	79.3	57.9	65.4	60.6	67.7 (↑1.8)
TRUST	63.1	67.8	70.3	81.0	64.5	81.4	85.0	83.2	85.4	85.8	90.1	85.7	72.1	79.1	68.6	77.5 (↑11.6)

Table 5: Comparison of augmentation impact on CIFAR10-C dataset. Increases/decreases in mean accuracy compared to performing no adaptation (Source only) is highlighted in green/red color.

H High Entropy Traversal Permutations

We have also tested the performance of TRUST using the top-k high-entropy (i.e., less confident) traversal permutations. As shown in Table 6, this setting leads to a significant performance drop of 12.5% compared to the source-only baseline. This highlights the critical role of traversal selection in our method. High-entropy permutations, which correspond to uncertain and unstable model predictions, introduce noise into the adaptation process and hinder effective generalization. These findings further support our strategy of entropy-based traversal filtering, where low-entropy permutations are prioritized to ensure reliable and robust adaptation.

Method	gaussian	shot	impulse	defocus	glass	motion	zoom	frost	snow	fog	brightness	contrast	elastic	pixelate	jpeg	Mean
Source only	46.8	48.4	45.0	73.5	52.6	73.0	78.7	71.8	75.8	77.3	85.7	69.6	63.7	67.9	59.0	65.9
TRUST (top-k high entropy)	38.5	32.2	36.4	57.9	37.7	61.5	64.9	52.5	64.0	70.3	74.1	49.3	57.9	56.2	48.2	53.4 (↓12.5)
TRUST (top-k low entropy)	63.1	67.8	70.3	81.0	64.5	81.4	85.0	83.2	85.4	85.8	90.1	85.7	72.1	79.1	68.6	77.5 (↑11.6)

Table 6: Comparison of TRUST performance with low and high entropy traversal permutations on CIFAR10-C dataset.

I TRUST Application beyond Classification Settings

We conducted additional experiments on segmentation tasks using various datasets, including Pascal VOC21 [52] and Pascal Context 59 [53], applying corruptions following the protocol of [54]. The results demonstrate that TRUST performs well in segmentation, outperforming methods like Tent. This further supports the generalizability and effectiveness of our approach beyond classification settings.

Dataset	Method	gaussian noise	shot noise	impulse noise	defocus blur	glass blur	motion blur	zoom blur	frost	snow	fog	brightness	contrast	elastic	pixelate	jpeg compression	Mean
V21	Source only	29.1	33.1	28.3	21.0	8.2	33.1	25.4	50.9	50.3	70.7	76.5	63.9	25.5	22.2	59.2	39.8
	Tent	33.0	35.7	32.0	22.3	14.7	38.2	25.3	46.5	49.0	60.2	63.9	66.2	38.5	28.8	43.9	39.9
	TRUST	**38.8**	**42.0**	**38.7**	**29.8**	**22.6**	**45.1**	**29.8**	**50.5**	**53.5**	**63.4**	**66.4**	**68.5**	**45.1**	**37.7**	**48.6**	**45.4** (↑5.6)
P59	Source only	17.1	19.6	17.4	27.4	14.9	29.2	19.5	30.2	28.5	42.1	50.8	41.0	23.9	30.4	38.4	28.7
	Tent	17.6	18.9	17.8	22.2	15.9	27.5	17.9	26.9	30.0	36.7	41.9	42.9	25.8	28.2	28.3	26.6
	TRUST	**24.4**	**27.4**	**25.4**	**24.6**	**21.2**	**30.1**	**19.8**	**29.8**	**32.8**	**39.2**	**42.4**	**43.2**	**31.5**	**36.1**	**31.6**	**30.6** (↑1.9)

Table 7: Segmentation performance under different corruptions of V21 and P59 datasets.

J Performance Comparison with ViT-based Method

To provide a fair evaluation, we adapted the ViT-based method from [34] by replacing its backbone with VMamba and modifying it to be compatible with VMamba backbone. Specifically, we replaced

the CLS token (absent in VMamba) with the mean of all tokens and adjusted the number of learnable tokens to match VMamba's dimensionality. This ensures that both methods operate under the same architectural constraints. Our experimental results demonstrate that TRUST, which is specifically designed to exploit VMamba's traversal mechanism, significantly outperforms the adapted baseline, highlighting its robustness under distribution shift.

Method	gaussian	shot	impulse	defocus	glass	motion	zoom	frost	snow	fog	brightness	contrast	elastic	pixelate	jpeg	Mean
Source only	24.3	26.1	25.1	22.2	23.2	35.4	43.2	49.3	48.4	56.9	70.0	26.8	45.1	43.7	41.4	38.7
Tent	27.8	30.0	28.8	24.9	25.9	38.0	45.5	51.0	51.3	59.1	70.6	30.0	48.2	47.8	45.7	41.7
FOA	17.8	19.4	18.5	15.1	18.2	22.7	28.6	38.7	33.9	44.3	57.4	22.7	38.2	40.9	41.7	30.5 (↓8.2)
TRUST	46.8	49.4	48.5	42.8	40.8	57.1	57.9	57.3	61.7	66.8	71.9	54.9	61.4	63.6	60.2	56.1 (↑17.4)

Table 8: Performance comparison across corruption types with the ViT-based Method.

K Weight Diversity Across Traversals

To analyze the consistency of the SS2D block parameters under different traversal permutations, we compute statistics based on the L2 norms of each parameter tensor across six traversal-adapted models. We extract each weight or bias tensor from each model, flatten it, and compute its L2 norm. We then calculate the mean of these norms to reflect the overall magnitude across traversals. To quantify variability, we also compute the standard deviation across the corresponding elements of these tensors and report the average of these deviations.

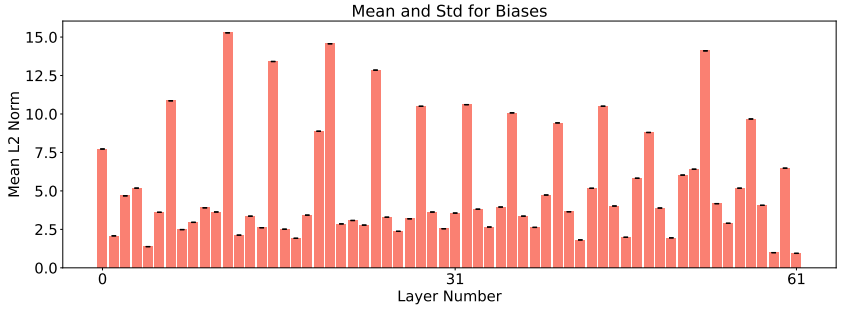

Figure 12: Mean and standard deviation of the L2 norm for the bias parameters

Figure 12 shows the mean and standard deviation of the L2 norm for the bias parameters, while Figure 13 shows the same for the weight parameters. These visualizations indicate that although the magnitude of parameters varies across layers, the variation across traversals remains relatively low. This suggests that SS2D block parameters adapted with different traversal orders remain geometrically close in parameter space. Such consistency is conceptually aligned with the findings in the Model Stock method [55], which emphasizes the benefits of maintaining proximity in the weight space across fine-tuned or adapted models, such as enabling effective weight averaging and improving generalization. This experiment is conducted using the CIFAR-10C dataset.

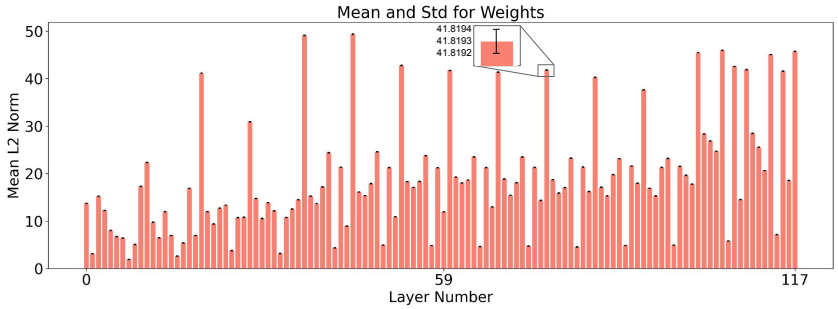

Figure 13: Mean and standard deviation of the L2 norm for the weight parameters