Hierarchical Adaptive networks with Task vectors for Test-Time Adaptation – Supplementary material –

Sameer Ambekar[1,2,4], Marta Hasny[1,2], Laura Daza[1,2], Daniel M. Lang[1,2*], Julia A. Schnabel[1,2,3,4*]
[1]School of Computation, Information and Technology, Technical University of Munich, Germany
[2]Institute of Machine Learning in Biomedical Imaging, Helmholtz Munich, Germany
[3]School of Biomedical Engineering and Imaging Sciences, King's College London, UK
[4]Munich Center for Machine Learning (MCML)

Contents

*Shared last-authorship

A. Additional discussions

A.1. Rationale for using Hierarchical layers in FC Layers but not in the Encoder

Common convolutional encoders such as ResNets [16] learn multiple representations, with initial layers capturing low-level features, and deeper layers learning high-level semantics [48]. However, standard fully connected (FC) layers attached to the encoder flatten these structured hierarchies into a single vector through the transformation $\phi(\mathbf{x}) \mapsto \mathbf{W}\phi(\mathbf{x})+\mathbf{b}$, leading to 'information diffusion' [22]. This conflicts with the encoder's multi-scale inductive bias, as shown by the rank inequality $\text{Rank}(\phi_{\text{conv}}) \gg \text{Rank}(\phi_{\text{FC}})$, where ϕ_{conv} represents convolutional features and ϕ_{FC} the FC-processed output [11]. To address this, matryoshka representation learning [22] introduced nested linear projections $\{f_i\}_{i=1}^{k}$, where each $f_i : \mathbb{R}^d \to \mathbb{R}^{d_i}$ satisfies dimensional constraints $d_1 < \cdots < d_k = d$, enforced by truncation conditions $\forall i < j,\ f_i(\phi(\mathbf{x})) = \text{Trunc}_{d_i}(f_j(\phi(\mathbf{x})))$ [22]. By using hierarchical linear layers [22] exclusively as the classification layers after the encoder, we enable it to learn coarse-to-fine features while preserving the encoder's hierarchical structure. This avoids redundancy and adaptation of already robust encoder features and addresses the limitations of fixed-dimensional FC layers, which are likely to fail to adapt to diverse distribution shifts for test-time adaptation.

A.2. Modification of Source Training

The test-time adaptation landscape comprises two primary approaches: (i) Fully test-time adaptation methods and (ii) Methods that modify source training (often termed training preparation [42]). Fully test-time adaptation methods [23, 40] leave source training unaltered, and in line with this, our three proposed innovations are applied exclusively at test time. Common methods, however, often rely on single-dimensional architectures that fail to capture multi-scale learning across the model. Consequently, we enhance common TTA methods to tackle diverse distribution shifts by incorporating hierarchical linear layers during source training. Similarly, other TTA approaches [10, 27, 33, 43, 44] also modify source training to develop strategic priors for adaptation.

A.3. Training Hierarchical linear layers

Training the hierarchical linear layers during source training is straightforward. All layers are optimized using a standard cross-entropy loss with equal weight scaling across the hierarchical linear layers [22], i.e., requiring no additional optimizations. At test time, our gradient selection mechanism adapts only the layer with the lowest gradient norm per batch, while the others are updated via a sharing mechanism.

A.4. Rationale for the usage of gradient norm to select the optimal linear layer for inference and adaptation

Gradient norm quantifies the magnitude of alteration that needs to be applied to align the model output with the target distribution. In our case, each hierarchical linear layer learns coarse-to-fine representations. Therefore, the smallest gradient norm corresponds to the layer whose parameters are already well-aligned to model the target data and require only minor adjustments. Hence, by choosing the layer with the lowest gradient norm for inference and adaptation, our method relies on the most stable features while minimizing the risk of over-adaptation and instability [19, 40]. This criterion promotes efficient test-time adaptation by using the layer's inherent common target features with minimal updates.

A.5. Motivation for using task vectors to share target information between the linear layers

The hierarchical linear layers are designed to learn coarse-to-fine representations of increasing granularity. If only the most suitable linear layer is adapted during test time, while the remaining layers stay unchanged, the non-adapted layers will not incorporate any new information or statistics about the target data. This undermines the utility of learning coarse-to-fine representations, as the non-adapted layers do not reflect the updated target distribution's statistics. To address this, task vectors are employed to propagate target information from the adapted layer to the remaining layers. By leveraging lightweight transformations, task vectors ensure that all layers align with the target distribution without requiring additional backpropagation. This maintains the cohesiveness of the hierarchical structure and allows all layers to contribute meaningfully to robust adaptation under diverse test-time conditions.

A.6. Rationale behind using cosine similarity to find similar layers for target information sharing

The weights of hierarchical linear layers have varying dimensionalities, as each layer is designed to learn coarse-to-fine representations. Cosine similarity is used to identify similar layers, eligible for target information sharing. The metric measures the alignment between weight vectors, focusing on their directional similarity rather than their magnitude. This property makes cosine similarity particularly effective for comparing weights across layers with different scales or norms. Computationally, to handle the dimensional mismatch between weights, smaller weight matrices are padded with zeros to match the largest dimensionality among the layers. This zero-padding aligns all weight vectors in a common vector space without altering their intrinsic directional properties, as the added zeros do not affect the cosine similarity computation [8]. By leveraging cosine similarity in this manner, we can robustly and efficiently identify layers with similar weights, enabling effective target information sharing across hierarchical linear layers during test-time adaptation.

A.7. Motivation for using mutual information in Hierarchical linear layers Agreement

Mutual information quantifies the dependency between two sets of variables, capturing both linear and nonlinear relationships [15, 52]. In Hi-Vec, we use mutual information to evaluate the agreement between the logits of the optimal hierarchical layer, $\mathbf{p}_{\phi^*}$, and those of the remaining layers, $\mathbf{p}_{\boldsymbol{\phi}}$ for $\boldsymbol{\phi} \in \Phi \setminus \{\boldsymbol{\phi}^*\}$. The hierarchical linear layers in Hi-Vec capture coarse-to-fine representations, where each layer learns features at different levels of granularity. These layers are connected sequentially in a nested architecture, transforming the encoder's output vector 'z' to match the dimensionality required by the linear layer. Agreement among their corresponding logits reflects the presence of shared features across these levels. ID samples exhibit high agreement across hierarchical layers as their features align with the source domain's representations. In contrast, OOD samples disrupt this agreement, leading to lower shared information between the logits. By quantifying this inter-layer agreement, mutual information enables Hi-Vec to detect OOD samples and avoid unnecessary adaptation to noisy batches. Moreover, mutual information has proven effective in domain adaptation by aligning features and reducing discrepancies [6, 52]. For instance, Zhao et al. [52] use mutual information to align features in unsupervised domain adaptation, while De Bernardi et al. [6] leverage it for OOD detection. Hi-Vec uses Mutual information to ensure selective adaptation on ID samples, preserving model stability by avoiding model misspecification and improving computational efficiency during test-time adaptation.

A.8. Motivation for not using efficient and fixed-feature versions from Matryoshka representation learning

The Matryoshka model [22] also includes an efficient version that nests the linear layers within one shared single linear layer. Additionally, the fixed feature version is provided that maps encoder output to one single dimension between the 8 to the highest dimension of the encoder and treats it as one single fc layer. However, in our work, we focus on diverse shifts at test time, which requires flexible models or layers capable of capturing varied features across multiple dimensions. The efficient version of [22], due to the nesting of representations within a single fully connected layer, is not useful for adaptation since the adaptation of one layer affects all of the other layers in the nest. Similarly, the fixed-feature version depends on static representations tied to a single dimension, which will also diffuse information [22] as in the common setup. Training multiple encoders with such fixed-feature models to handle diverse shifts is computationally inefficient and lacks the flexibility required for test-time adaptation.

A.9. Additional Notations for Merging Layer Weights of Small and Large Linear Layers

Let W_ϕ denote the weight matrix of layer ϕ with dimensions $(m \times n)$, and let W_{ϕ^*} denote the weight matrix of the dynamically selected layer ϕ^* with dimensions $(k \times l)$. These matrices can be written explicitly as,

$$W_\phi = (w_{\phi,ij})_{i=1,\ldots,m;\ j=1,\ldots,n}\,,$$

$$W_{\phi^*} = (w_{\phi^*,ij})_{i=1,\ldots,k;\ j=1,\ldots,l}\,.$$

For dimension-aware merging, we define the shared dimensions

$$g = \max(m,k), \quad h = \min(n,l),$$

where $\max$ selects the larger dimension and min selects the smallest dimension in respective directions. Then, the weights of W_ϕ are projected into the common shape $(g \times h)$, aligned with the indexing of W_{ϕ^*}:

$$W_\phi[\phi^*] := (w_{\phi,ij})_{i=1,\ldots,g;\ j=1,\ldots,h}\,.$$

A.10. Extending to Parameter-free baselines

We also extend Hi-Vec to parameter-free baselines such as Laplacian Adjusted Maximum-likelihood Estimation (LAME) [3], which adjusts the model's output for target batches. Specifically, we integrate Hi-Vec's dynamic layer selection to perform targeted inference for each batch instead of representations from a static linear layer. Next, leveraging hierarchical layer agreement to decide whether the model's output adjustment with LAME [3] is necessary. More specifically, we implement LAME with a RestNet-18 encoder with and without Hi-Vec for the CIFAR-10-C dataset with 20% outliers, which depicts the standard setting of our ablation studies. We find that with the implementation of Hi-Vec, the standard implementation of LAME can be improved from 68.4% to 79.7% accuracy, which is significantly higher than the source baseline of 57.3%. This highlights the ability of Hi-Vec to also integrate with parameter-free methods apart from backpropagation-based methods and motivates us to further investigate this in future work.

A.11. Algorithms

We provide the details about source model training in Algorithm 1. The detailed test-time algorithm is in the main paper.

Algorithm 1 Source Model Training

Input: $\mathcal{D}_s$: source domain with labeled samples $\{(\mathbf{x}_s, \mathbf{y}_s)\}$; $\boldsymbol{\theta}$: encoder parameters; $\Phi = \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_k\}$: hierarchical linear models; $\{W_m\}_{m\in\mathcal{M}}$: hierarchical linear layer weights where $\mathcal{M} = \{m_1, m_2, \ldots, m_k\}$ spans the dimensions.

Output: learned θ with Φ hierarchical linear layers.

```
1: for epoch in $1, \ldots, N_{\text{epochs}}$ do
2:     for batch $\{\mathbf{x}_s^{(i)}, \mathbf{y}_s^{(i)}\}_{i=1}^{\mathcal{B}_{tr}}$ in $\mathcal{D}_s$ do
3:         Extract representations: $\mathbf{z} = f_{\boldsymbol{\theta}}(\mathbf{x}_s)$.
4:         Compute predictions for all layers $\boldsymbol{\phi} \in \Phi$ and corresponding dimensions $m \in \mathcal{M}$:
           $W_{\boldsymbol{\phi}}^{\top}\mathbf{z}_m$, where $\mathbf{z}_m$ are the first $m$ dimensions of $\mathbf{z}_s$.
5:         Compute cross-entropy loss:
           $\mathcal{L} = \sum_{\phi\in\Phi}\sum_{m\in\mathcal{M}} \mathcal{L}_{\text{CE}}(W_{\boldsymbol{\phi}}^{\top}\mathbf{z}_{1:m}, \mathbf{y}_s)$.
6:         Update $\boldsymbol{\theta}$ and $\{W_{\boldsymbol{\phi}}\}_{\boldsymbol{\phi}\in\Phi}$ as in Eq.2 from main paper.
7:     end for
8: end for
```

	CIFAR10-C (8:2)			CIFAR100-C (80:20)		
Methods	Acc	Auc	H-score	Acc	Auc	H-score
Source	60.6	61.5	60.0	37.1	60.6	44.2
Source	60.6	61.5	60.0	37.1	60.6	44.2
MRL	71.3	63.8	66.4	43.4	62.2	50.3
BN Stats [29]	79.4	66.9	72.6	55.5	66.2	60.2
EATA [30]	81.5	67.9	74.1	61.3	67.4	64.2
CoTTA [41]	81.8	65.4	72.5	57.1	66.3	61.3
RoTTA [47]	79.4	62.4	69.8	50.6	63.7	56.2
SoTTA [12]	82.7	62.7	71.3	61.4	68.2	64.6
OWTTT [25]	65.9	63.5	64.4	56.2	66.0	60.6
Tent [40]	80.1	65.9	72.3	60.4	65.6	62.8
SAR [31]	82.3	66.0	73.2	63.4	69.1	66.1
STAMP	85.0	69.4	76.4	66.0	71.2	68.5
Tent + ***Hi-Vec***	83.9 ▲	75.0 ▲	79.2 ▲	60.6 ▲	68.2 ▲	64.1 ▲
SAR + ***Hi-Vec***	84.1 ▲	74.9 ▲	79.2 ▲	63.8 ▲	69.7 ▲	66.6 ▲
STAMP + ***Hi-Vec***	**86.5** ▲	**69.9** ▲	**77.3** ▲	**66.8** ▲	**72.1** ▲	**69.3** ▲

Table 1. **Comparisons on adaptation with open-set datasets** using CIFAR10-C-20 and CIFAR100-C-80 as target datasets. We report the baselines provided by Yu et al. [46] with ResNet-18. Our results are averaged over fine runs. As in the main results, Hi-Vec improves the performance (▲) and is the top-performer (bold).

B. Additional Experiments

B.1. Results across the open-set adaptation

Hi-Vec leverages hierarchical representations and task vectors propagation to adapt to unseen domains, making it effective for open-set test-time adaptation. In this open-set setting, a subset of classes is unseen during source training but present in the test set. Following [46], we evaluate Hi-Vec on Cifar-10-c and Cifar-100-c. For Cifar-10-c, the source model is trained on 8 classes and evaluated on 2 unseen classes in the corrupted version. Similarly, for Cifar-100-c, the source model is trained on 80 classes and evaluated on 20 unseen classes. Results are given in Table 1. Open-set [34] adaptation lacks prior class and corresponding sample information during training, posing challenges for common methods. Hi-Vec improves performance across both scenarios by learning coarse-to-fine representations, avoiding overfitting to learned fixed features and corresponding limited classes during training, thus enabling robust adaptation to unseen classes.

B.2. Results across common test-time adaptation setting

Following common test-time adaptation methods [13, 32, 40, 46], we also provide results without any outlier datasets at test time. In Table 2, we compare the performance of all baselines across the domains of the Cifar-10-c and Cifar-100-c datasets. Hi-Vec achieves the highest accuracy compared to all recent test-time adaptation methods on both datasets.

B.3. Inference time

During source training, Hi-Vec introduces a small parameter overhead of approximately 5% compared to standard implementations. For example, for ResNet-18, the encoder has approximately 11M parameters while the added layers contribute approximately 0.5M parameters. This overhead scales modestly to larger models, even when implemented for intermediate layers such as recent Matformer [7]. At test-time, Hi-Vec needs to compute multiple gradient norms for layer selection. However, to adapt to target data, it doesn't introduce any additional backward passes or parameters needed for backpropagation. Specifically, Hi-Vec uses only one of the hierarchical linear layers for backpropagation and during inference, ensuring no additional parameters are introduced. In Table 3, we compare the inference times for common test-time adaptation methods with and without Hi-Vec integration based on a single Nvidia A100 GPU. Hi-Vec introduces a modest increase in compute time for Tent and SAR, with Tent + Hi-Vec taking 1m 25s compared to 31s for Tent and SAR + Hi-Vec taking 1m 42s compared to 50s for SAR. For STAMP, Hi-Vec results in a negligible change. Overall, Hi-Vec achieves robust adaptation with minimal computational overhead, making it useful for test-time adaptation.

B.4. Mitigates Catastrophic Forgetting

Retaining source domain knowledge during test-time adaptation is essential for leveraging the shared features between source and target domains [50]. Catastrophic forgetting, where the model loses previously acquired target information when adapting to new data, can severely degrade performance on source data samples. We also test Hi-Vec's capability for such cases and provide the results for catastrophic forgetting of source knowledge in Fig 1. In this setup, we adapt the model to a target domain and evaluate it on the source domain for the Cifar-10-c dataset with noise as an outlier with ResNet-18. Integrating Hi-Vec with methods

	Method	Gauss	Shot	Imp	Defoc	Glass	Rot	Zoom	Snow	Frost	Fog	Bright	Contr	Elast	Pixel	JPEG	**Avg.**
CIFAR10-C	Source	28.7	35.2	24.2	57.3	49.0	66.4	64.8	76.7	62.9	73.4	90.3	31.4	78.8	46.5	74.7	57.3
	BN Stats[29]	70.2	72.0	63.6	87.6	66.5	85.9	86.9	82.1	80.5	84.1	90.8	85.4	77.3	78.6	74.3	79.0
	CoTTA[41]	76.1	77.6	73.9	87.9	71.9	86.9	87.5	82.8	82.0	85.2	90.8	85.7	79.3	80.4	78.3	81.8
	EATA[30]	76.5	77.9	71.7	89.1	70.4	87.4	89.0	85.2	84.2	86.0	91.5	88.4	79.9	83.9	78.5	82.6
	RoTTA[47]	69.6	71.0	62.9	87.5	67.0	85.9	86.9	82.4	79.6	84.7	91.2	73.5	78.3	77.9	74.9	78.2
	SoTTA[12]	75.8	79.2	71.5	89.4	70.6	87.8	89.0	85.6	84.0	87.2	92.4	87.4	79.9	84.5	79.0	82.9
	OWTTT[25]	70.1	72.0	63.0	86.9	66.5	85.7	86.7	82.7	80.9	84.5	91.2	83.0	77.9	76.8	74.9	78.9
	Tent[40]	76.1	78.4	70.5	87.8	70.1	86.8	87.5	84.8	82.0	85.1	91.1	86.6	79.4	82.7	78.5	81.8
	SAR[31]	70.7	72.1	66.8	87.6	68.2	85.9	86.9	82.1	80.5	84.1	90.8	85.4	77.3	78.6	74.3	79.4
	STAMP	80.9	82.9	77.2	87.6	74.9	86.6	87.7	85.9	85.9	88.1	90.4	87.2	80.3	86.7	82.6	84.3
	Tent + ***Hi-Vec***	77.7	79.6	70.6	88.3	72.3	87.6	87.9	85.7	82.8	85.7	91.9	86.5	80.2	82.9	78.9	82.5 ▲
	SAR + ***Hi-Vec***	78.5	80.2	69.1	91.4	73.8	88.7	90.7	88.0	88.3	89.0	93.2	91.6	80.5	86.4	79.2	84.6 ▲
	STAMP + ***Hi-Vec***	**83.6**	**84.7**	**77.3**	**90.7**	**77.3**	**88.8**	**90.7**	**88.5**	**89.6**	**89.1**	**92.9**	**90.3**	**83.4**	**88.3**	**83.6**	**86.6** ▲
CIFAR100-C	Source	12.4	14.5	7.2	36.0	44.7	45.1	45.2	49.5	41.6	36.9	63.3	13.2	57.5	23.8	46.6	35.8
	BN Stats[29]	41.1	41.3	38.9	63.1	51.5	60.8	63.8	51.2	53.5	53.2	64.4	59.0	58.4	58.0	47.6	53.7
	CoTTA[41]	47.0	47.8	45.1	59.6	54.0	59.5	61.3	53.3	55.0	52.9	62.6	50.3	58.1	61.8	53.1	54.8
	EATA[30]	50.7	53.5	48.1	67.0	55.6	64.8	67.0	59.1	59.2	60.4	67.7	63.9	61.8	63.3	54.6	59.8
	RoTTA[47]	35.9	36.6	33.8	60.6	47.1	57.7	60.8	48.0	42.2	50.8	59.2	32.1	53.8	52.3	44.4	47.7
	SoTTA[12]	51.6	53.8	47.4	66.9	56.9	65.3	68.1	58.9	60.1	60.1	69.4	63.1	62.3	62.8	54.8	60.1
	OWTTT[25]	41.6	42.8	38.8	63.4	52.6	61.6	64.8	53.3	54.8	54.5	65.8	58.4	60.1	57.6	49.3	54.6
	Tent[40]	52.3	52.1	47.7	66.9	56.1	64.3	65.3	58.3	58.7	60.0	67.8	62.1	61.8	63.0	54.5	59.4
	SAR[31]	55.1	55.0	51.2	68.4	58.2	66.0	67.4	60.3	60.8	61.9	69.8	65.5	63.6	66.2	56.8	61.7
	STAMP[46]	57.2	58.5	52.8	69.9	61.4	68.1	70.1	63.3	63.9	64.8	72.2	69.9	66.5	69.2	59.0	64.4
	Tent + ***Hi-Vec***	52.9	53.4	47.9	67.8	56.7	64.9	65.7	58.4	58.9	61.9	68.3	62.7	62.4	63.5	54.9	60.2 ▲
	SAR + ***Hi-Vec***	55.7	55.3	51.8	69.2	58.5	66.7	67.9	61.5	61.7	62.5	70.5	66.2	63.9	66.7	57.4	62.3 ▲
	STAMP + ***Hi-Vec***	**57.9**	**60.2**	**53.5**	**70.2**	**61.9**	**68.9**	**71.4**	**63.9**	**64.3**	**65.4**	**72.6**	**70.4**	**66.9**	**69.7**	**59.3**	**65.1** ▲

Table 2. **Comparisons on common test-time adaptation setting.** for CIFAR10-C and CIFAR100-C. Integrating Hi-Vec improves the common methods and achives the the best results (bold)

Methods	**Time**
Tent [40]	31s
Tent + **Hi-Vec**	1m 25s
SAR [31]	50s
SAR + **Hi-Vec**	1m 42s
STAMP [46]	11m 48s
STAMP + **Hi-Vec**	11m 49s

Table 3. **The total inference time across all the fifteen domains** on Cifar-10-C with ResNet-18. The usage of Hi-Vec leads to a moderate increase in commute time for SAR and Tent. For STAMP, Hi-Vec introduces a negligible change in the inference time.

such as SAR and Tent improves source domain accuracy during adaptation. Hi-Vec achieves this by employing its gradient-based layer selection mechanism to adapt only the most relevant hierarchical layer and its hierarchical layer agreement mechanism to prevent unnecessary updates for out-of-distribution samples. These Hi-Vec mechanisms preserve source knowledge while enabling robust adaptation to target shifts.

B.5. Analysis of Hierarchical Layers Agreement and Benefits

Hierarchical layers agreement, as detailed in the methodology and algorithm, enables Hi-Vec to skip adaptation for batches containing outliers, thereby preventing noisy model updates that could lead to model misspecification. This is particularly important because adapting to noisy OOD samples can destabilize the model by error accumulation and reinforcing irrelevant spurious features. Hierarchical layers in Hi-Vec capture coarse-to-fine representations, and their agreement reflects shared, meaningful features across these levels. When agreement is low, it indicates the absence of common features, signaling extreme OOD samples. To provide insights into this mechanism, Figure 3 shows the number of times adaptation was skipped during test-time using ResNet-18 on Cifar-10-c with noise as outlier data. The x-axis represents the domain count for the Cifar-10-c dataset with outliers during test-time adaptation, while the y-axis indicates the count of skipped adaptations. For Tent + Hi-Vec and STAMP + Hi-Vec, adaptation was skipped more frequently, focusing on inference instead, while SAR + Hi-Vec skipped adaptation for nearly half the batches. In comparison, common test-time adaptation [23, 26, 40, 46] do not skip adaptation (corresponding to 0 skips in the Figure 3) and instead perform backpropagation for every batch. By avoiding adaptation on noisy batches that lead to error accumulation for the model and its predictions, Hi-Vec achieves stability on noisy target samples.

B.6. Grad-CAM Visualizations and Layer selection insights

At test-time, to analyze the behavior of hierarchical linear layers and the selection of ϕ^* at test-time, we provide qualitative visualizations in Figure 2. Specifically, we illustrate the Grad-CAM [37] outputs with STAMP + Hi-Vec for the selected hierarchical linear layer (ϕ^i) alongside a histogram showing the frequency of selected layers across test batches of Cifar-10-c dataset with noise. Grad-CAM [37] visualizations highlight how different dimensions focus on distinct regions of the input image, enabling accurate classification by leveraging features relevant to the target distribution, even when it deviates significantly from the source. The histogram further demonstrates that multiple dimensions are utilized dynamically across test batches, reflecting Hi-Vec's ability to adaptively select layers suited to varying shifts. This adaptive mechanism ensures that hierarchical representations effectively address diverse target shifts by focusing on features most relevant for each batch. Additionally, In Figure 4, we also provide insights into layer selection for the Waterbirds dataset using DeYo + Hi-Vec for test-time adaptation. Unlike the shifts observed between CIFAR-10 and Cifar-10-c, the Waterbirds dataset introduces distinct distribution shifts that require different hierarchical layers to handle each batch effectively. The figure illustrates Hi-Vec's ability to dynamically select the optimal layer for each batch, enhancing the adaptability of test-time adaptation methods and ensuring robust performance across diverse shifts.

C. Additional dataset and implementation details

C.1. Additional datasets information

We train source models respectively on the source datasets as per the training and evaluation procedure as in [13, 23, 40, 46]. Akin to these common methods, the source datasets to train the respective source models include CIFAR-10, CIFAR-100, Waterbirds, and ImageNet as datasets. We utilize ResNet-18 and ResNet-50 with batch norm as the encoders. Our experimental setup involves multiple datasets at test time to evaluate the proposed method under two distinct types of diverse distribution shifts at test time. First, we consider outlier-aware scenarios where each test batch contains samples from Cifar-10-c [17], Cifar-100-c [17], or ImageNet-C [17] datasets combined with a significant proportion of outlier datasets, including LSUN-C [46], SVHN-C [46], TinyImageNet-C [46], Places-365-C [46], and Textures-C [46]. For the in-distribution data, Cifar-10-c [17] and Cifar-100-c [17] include 15 corruption types (e.g., Gaussian noise, motion blur, fog) applied at five severity levels, with Cifar-10-c [17] containing 10,000 images per corruption and Cifar-100-c [17] containing 10,000 images per class. ImageNet-C [17] serves as a large-scale benchmark with similar corruption types across 1,000 categories. We follow [46] to generate the outlier datasets. The outlier datasets are derived by applying the same 15 corruption types at the highest severity level of 5 to datasets such as LSUN, SVHN, TinyImageNet, Places-365, and Textures. In this setup, a single test batch contains both in-distribution samples from the corrupted CIFAR or ImageNet datasets and out-of-distribution samples from these corrupted outlier datasets. Next, we evaluate robustness to spurious correlations using datasets specifically designed to introduce misleading or non-causal correlations. The Waterbirds [36] dataset consists of 11,788 images where spurious correlations arise due to the background (e.g., water or land) being associated with specific bird species. Similarly, ColoredMNIST [36] modifies the MNIST dataset by introducing color as a spurious feature correlated with digit labels, consisting of 70,000 images across 10-digit classes. We train source models on Cifar-10, Cifar-100, Waterbirds, and ImageNet, respectively, using ResNet-18 and ResNet-50 as encoders. For ablations and additional experiments, we integrate Hi-Vec with re-implemented baselines from their official GitHub repositories.

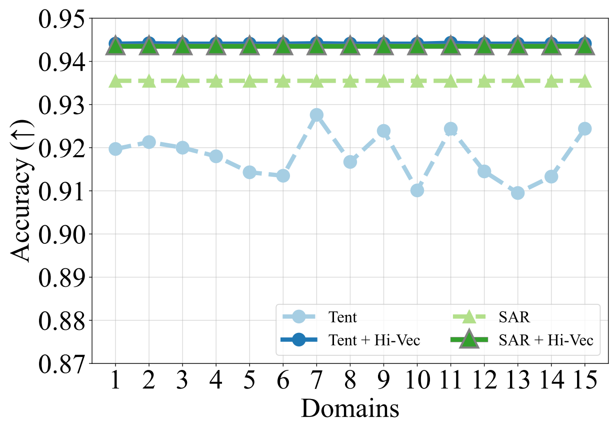

Figure 1. **Mitigates Catastrophic Forgetting.** Resluts reported for Cifar-10-c using ResNet-18. We evaluate the model on the source domain after adapting it to every target domain. Hi-Vec preserves the source domain knowledge and prevents forgetting on the dataset at test-time.

C.2. Hyperparameters for the baselines

We utilize the hyperparameters as provided in STAMP [46] and DeYo [23] repositories that include the implementation

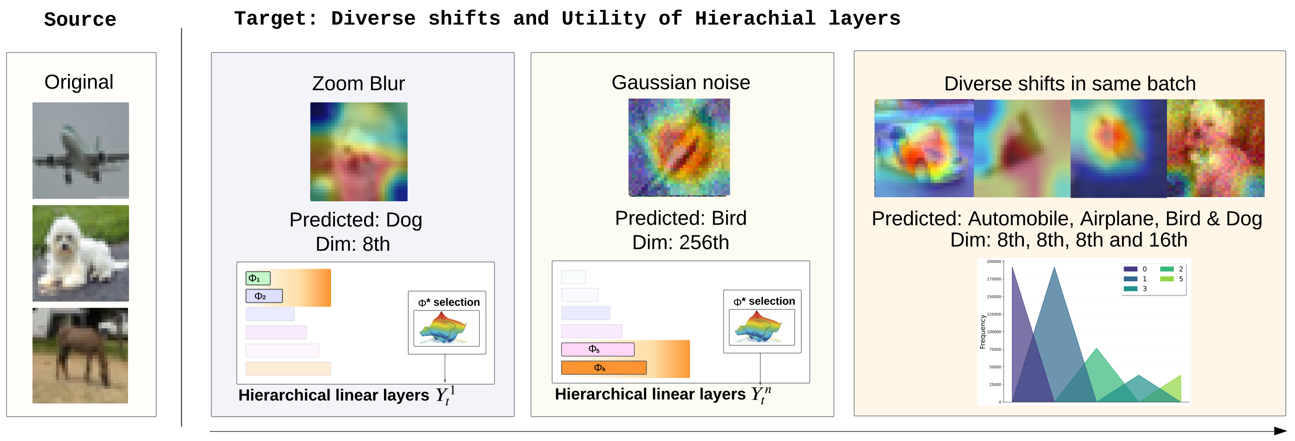
Source
Target: Diverse shifts and Utility of Hierachial layers
Original
Zoom Blur
Predicted: Dog
Dim: 8th
Gaussian noise
Predicted: Bird
Dim: 256th
Diverse shifts in same batch
Predicted: Automobile, Airplane, Bird & Dog
Dim: 8th, 8th, 8th and 16th
Φ* selection
Hierarchical linear layers Y_t^1
Hierarchical linear layers Y_t^n
Frequency

Figure 2. **Grad-CAM Visualizations and Layer selection insights** on Cifar-10-c with ResNet-18 by Stamp + Hi-Vec. We provide the histogram figures for the outputs of the hierarchical linear layers. Together with the dimension of the model that is being used for the prediction and histogram of dimensions (where layer 0 has 8 dimensions, layer 1 has 16, and layer n has 2^{n+1} dimensions) for a random batch of the Cifar-10-c dataset.

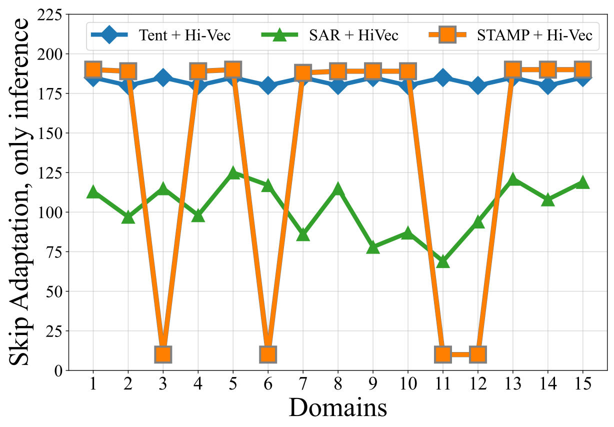

Figure 3. **Analysis of Hierarchical Layers Agreement and Benefits.** Reported for the Cifar-10C dataset with a ResNet-18 encoder. We provide insights into how the hierarchical layers agreement works and how it counts. The common methods perform back-propagation on every batch (0 skips in the figure). Hi-Vec opts to skip adaptation due to the hierarchical layer agreement mechanism, improving the process by avoiding adaptation on noisy samples and noisy predictions.

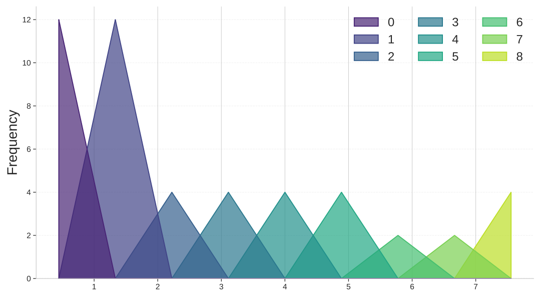

Figure 4. **Layer selection insights.** We also provide the Layer selection insights with a histogram of counts for Waterbirds dataset by using DeYo + Hi-Vec with ResNet-18 at test-time. Layer 0 has 8 dimensions, layer 1 has 16, and layer n has 2^{n+1} dimensions. Hi-Vec dynamically selects layers across varying dimensions to address diverse distribution shifts effectively.

of other baseline methods such as Tent [40], SAR [32]. As in these methods, we use ResNet-18 with batch norm for all our results. We list all the hyperparameters for the experiments that align with the original GitHub implementation repositories and implementations provided by STAMP[1] [46] and DeYo[2] [23].

Tent. Wang et al. [40] provided by [46] uses the Adam optimizer with beta set to 0.9 and with a momentum of 0.9. For Cifar-10-c, the learning rate (`lr`) was set to 0.001. For Cifar-100-c, the learning rate was reduced to 0.0001. For ImageNet, the learning rate was further adjusted to 0.00025.

SAR. Niu et al. [32] provided by [46] uses the Adam optimizer with beta set to 0.9 and with a momentum of 0.9. For Cifar-10-c, the learning rate (`lr`) was set to 0.005 and the reset constant (`rst`) was 0.3. For Cifar-100-c, the learning rate was also 0.005, but `rst` was reduced to 0.2. For ImageNet, the learning rate was 0.0005 while `rst` was adjusted to 0.1.

STAMP. Yu et al. [46] uses the Adam optimizer with beta set to 0.9 with a momentum of 0.9.For Cifar-10-c, the learning rate (`lr`) was set to 0.1 and the stamp parameter (`alpha`) was 0.25. For Cifar-100-c, `lr` was set to 0.05 and `alpha` was 0.9. For ImageNet, `lr` was set to 0.01, and `alpha` was 0.8. The MI threshold τ_{OOD}
value of 1.6, Cosine threshold τ of 0.6 was used for the datasets. We experimented with the hyperparameter values

[1] https://github.com/yuyongcan/STAMP
[2] https://github.com/Jhyun17/DeYO/

and have reported the values for which the performance is highest.

DeYo. Lee et al. [23] uses SGD as an optimizer with a momentum of 0.9. For the Waterbirds dataset, a pseudo-label threshold (`plpd_threshold`) of 0.5, a deyo margin (`deyo_margin`) of 0.5, an early deyo margin (`deyo_margin_e0`) of 0.4, a learning rate multiplier (`lr_mul`) of 5, and an evaluation interval of 10 was applied. For ColoredMNIST, after pretraining a pseudo-label threshold of 0.5 was used, both deyo margins were set to 1.0, and a learning rate multiplier of 5 and an evaluation interval of 30 has been utilized.

Hi-Vec. Our approach employs the hyperparameters listed above for the corresponding baselines for integration. Settings specific to our method involve the selection of a scaling parameter and a mutual information threshold classifying hierarchical layer agreement. For Cifar10, Imagenet, and Cifar100 experiments, we utilize a scaling (α) parameter of 0.7 and a mutual information threshold τ_{OOD} of 1.2 for the experiments. We have detailed the implementation of our method in Section 3, the detailed algorithm, and Figures 1 and 2 from the main paper. We will release the full code in the final version.

C.3. Results for the main paper with standard deviations.

We also provide the standard deviations for the main results from the main paper.

D. Additional Related work

D.1. Test-time adaptation.

In addition to the aforementioned applications, test-time adaptation has also been used in vision language models [1, 2, 38, 51], continual learning [18, 28, 45], classification that consider practical scenarios [4, 5, 9, 14, 21, 24, 39]. Moreover, recently, [20] proposed a framework that uses linear mode connectivity for adapted models during inference, with standard ResNet models.

		Noise			**SVHN-C**			**LSUN-C**			**TinyImageNet-C**		
	Methods	Acc	AUC	H-score	Acc	AUC	H-score	Acc	AUC	H-score	ACC	AUC	H-score
Cifar-10-C	Source	57.3	70.4	62.3	57.3	67.4	61.1	57.3	62.8	59.6	57.3	64.5	59.4
	MRL [22]	59.7	65.7	62.6	59.7	64.4	62.0	59.7	61.9	60.2	59.7	64.1	61.8
	BN Stats [29]	72.9	68.6	70.6	78.7	75.3	76.9	79.4	79.4	79.4	79.0	72.9	75.8
	EATA [30]	72.9	68.5	70.6	78.8	75.3	76.9	79.4	79.4	79.4	78.9	73.1	75.9
	CoTTA [41]	77.3	62.4	67.3	81.6	78.6	80.1	82.2	84.2	83.2	81.9	**75.3**	78.4
	RoTTA [47]	77.6	74.3	75.6	78.4	76.0	77.2	78.8	79.5	79.1	78.6	73.3	75.8
	SoTTA [12]	77.8	51.7	61.6	79.3	72.8	75.9	79.8	77.9	78.8	79.6	72.6	75.9
	OWTTT [25]	62.3	64.4	58.5	66.1	75.3	69.6	63.1	78.9	68.5	56.3	58.8	56.2
	Tent [40]	77.4	48.7	59.7	80.8	54.9	65.1	81.2	62.3	70.4	81.1	65.6	72.4
	SAR [31]	72.9	68.5	70.6	78.7	75.3	76.9	79.4	79.4	79.4	79.0	72.9	75.8
	STAMP [46]	77.9	83.2	80.1	82.3	79.2	80.6	83.5	86.3	84.8	82.6	74.9	78.5
	Tent + ***Hi-Vec***	80.7 ±0.2 ▲	63.0 ±0.2 ▲	70.5 ±0.2 ▲	81.7 ±0.1 ▲	55.4 ±0.1 ▲	66.0 ±0.1 ▲	81.7 ±0.2 ▲	62.8 ±0.2 ▲	71.0 ±0.2 ▲	82.5 ±0.1 ▲	65.8 ±0.1 ▲	73.2 ±0.1 ▲
	SAR + ***Hi-Vec***	77.7 ±0.1 ▲	63.8 ±0.1 ▲	70.7 ±0.1 ▲	82.5 ±0.1 ▲	73.3 ±0.1 ▲	77.7 ±0.1 ▲	80.2 ±0.2 ▲	79.9 ±0.2 ▲	80.0 ±0.2 ▲	83.0 ±0.1 ▲	69.7 ±0.1 ▲	75.7 ±0.1 ▲
	STAMP + ***Hi-Vec***	**83.6** ±0.1 ▲	**91.4** ±0.1 ▲	**87.3** ±0.1 ▲	**85.7** ±0.1 ▲	**82.7** ±0.1 ▲	**84.2** ±0.1 ▲	**84.3** ±0.2 ▲	**86.9** ±0.2 ▲	**85.5** ±0.2 ▲	**86.5** ±0.1 ▲	**81.1** ±0.1 ▲	**83.7** ±0.1 ▲
Cifar-100-C	Source	35.8	43.1	38.0	35.8	49.4	40.1	35.8	58.2	43.2	35.8	57.1	42.7
	MRL [22]	41.4	60.3	47.8	41.4	61.0	47.3	41.4	58.0	48.3	41.4	63.3	48.4
	BN Stats [29]	45.8	80.9	58.4	52.7	72.5	60.9	53.7	73.8	62.0	53.2	68.6	59.7
	EATA [30]	55.2	86.1	67.1	58.1	75.6	65.6	58.8	77.2	66.7	58.6	70.7	64.0
	CoTTA [41]	47.0	83.4	59.9	53.7	73.2	61.8	54.3	76.9	63.6	54.5	68.1	60.4
	RoTTA [47]	47.9	54.0	49.4	47.3	67.0	55.3	48.3	69.5	56.7	47.8	65.5	55.0
	SoTTA [12]	54.4	53.3	52.8	53.6	70.3	60.7	54.4	70.8	61.4	53.9	68.4	60.1
	OWTTT [25]	47.1	70.3	56.2	53.9	74.3	62.3	54.5	73.5	62.5	54.2	68.5	60.4
	Tent [40]	47.9	55.8	51.2	54.4	70.4	61.2	55.4	72.4	62.7	55.0	68.6	60.9
	SAR [31]	57.5	88.6	68.9	59.2	65.2	61.9	60.5	73.5	66.3	60.8	72.1	65.9
	STAMP [46]	57.9	98.4	72.8	63.7	82.1	71.7	63.7	**82.6**	71.9	63.9	75.5	69.2
	Tent + ***Hi-Vec***	54.9 ±0.2 ▲	68.2 ±0.2 ▲	60.1 ±0.2 ▲	54.7 ±0.2 ▲	73.9 ±0.2 ▲	62.3 ±0.2 ▲	57.1 ±0.2 ▲	72.7 ±0.2 ▲	63.9 ±0.2 ▲	55.3 ±0.1 ▲	69.9 ±0.1 ▲	61.1 ±0.1 ▲
	SAR + ***Hi-Vec***	57.9 ±0.1 ▲	89.2 ±0.1 ▲	69.4 ±0.1 ▲	54.9 ±0.1 ▲	73.4 ±0.1 ▲	62.8 ±0.1 ▲	60.9 ±0.2 ▲	73.9 ±0.2 ▲	66.7 ±0.2 ▲	62.1 ±0.1 ▲	74.4 ±0.1 ▲	68.4 ±0.1 ▲
	STAMP + ***Hi-Vec***	**58.2** ±0.1 ▲	**89.6** ±0.1 ▲	**73.5** ±0.1 ▲	**64.4** ±0.1 ▲	**82.5** ±0.1 ▲	**72.1** ±0.1 ▲	**63.8** ±0.2 ▲	82.5 ±0.2 ▲	**72.0** ±0.2 ▲	**64.6** ±0.1 ▲	**75.8** ±0.1 ▲	**70.4** ±0.1 ▲

Table 4. **Results on adaptation with outlier datasets** using Cifar-10-C and Cifar-100-C as target datasets with four outlier datasets. We report the baselines and use evaluation metrics as provided by Yu et al. [46] with ResNet-18. Our results are averaged over fine runs. Hi-Vec consistently improves performance (indicated by ▲) over the common methods and is the top-performer (**bold**)

Dataset	**Methods**	**Acc (%)**	**Worst-Group Acc (%)**
ColoredMNIST	Source	63.40	20.05
	MRL [22]	85.24	60.31
	Tent [40]	57.06	9.80
	MEMO [49]	63.77	6.23
	SENTRY [35]	63.23	15.78
	EATA [30]	60.81	17.98
	SAR [32]	58.37	12.36
	DeYO [23]	78.24	67.39
	SAR + ***Hi-Vec***	62.71 ±0.3 ▲	15.68 ±0.3 ▲
	DeYO + ***Hi-Vec***	**79.53 ±0.2** ▲	**68.62 ±0.2** ▲
WaterBirds	Source	83.16	64.90
	MRL [22]	85.24	60.31
	Tent [40]	82.95	54.14
	MEMO [49]	82.34	50.47
	SENTRY [35]	85.77	60.90
	EATA [30]	82.38	52.38
	SAR [32]	82.60	53.41
	DeYO [23]	87.42	73.92
	SAR + ***Hi-Vec***	83.25 ±0.3 ▲	55.61 ±0.3 ▲
	DeYO + ***Hi-Vec***	**89.53 ±0.2** ▲	**77.23 ±0.2** ▲

Table 5. **Results for spurious correlation datasets** using ColoredMNIST and WaterBirds with ResNet-18. We report baselines provided by [23]. Our results are averaged over five runs. Hi-Vec improves (▲) the common methods and performs the best (bold).

	Places365-C			**Textures-C**		
Methods	ACC	AUC	H-score	ACC	AUC	H-score
Source	18.2	61.6	26.1	18.2	54.6	25.8
MRL [22]	18.4	61.9	28.3	18.4	54.6	27.4
BN Stats [29]	31.1	67.7	41.1	31.6	61.2	40.7
EATA [30]	46.4	72.6	56.0	46.4	62.2	52.8
CoTTA [41]	33.8	66.9	43.5	34.2	60.7	42.8
RoTTA [47]	36.6	68.6	46.5	37.0	65.3	46.5
SoTTA [12]	41.7	67.8	50.7	41.8	60.3	48.8
OWTTT [25]	9.1	54.0	13.9	9.4	59.4	14.6
Tent [40]	34.9	51.8	39.5	39.0	48.6	42.0
SAR [31]	44.9	73.3	55.0	45.6	67.0	54.0
STAMP	46.4	77.7	57.6	46.5	71.9	56.2
Tent + ***Hi-Vec***	35.4 ±0.4 ▲	52.0 ±0.4 ▲	42.1 ±0.4 ▲	39.4 ±0.3 ▲	59.1 ±0.3 ▲	47.2 ±0.3 ▲
SAR + ***Hi-Vec***	45.3 ±0.2 ▲	73.8 ±0.2 ▲	56.1 ±0.2 ▲	46.2 ±0.3 ▲	67.4 ±0.3 ▲	54.8 ±0.3 ▲
STAMP + ***Hi-Vec***	**46.9 ±0.3** ▲	**77.9 ±0.3** ▲	**58.5 ±0.3** ▲	**46.8 ±0.2** ▲	**71.7 ±0.2** ▲	**56.6 ±0.2** ▲

Table 6. **Results on adaptation with outlier datasets** using Imagenet-C with Places365-C and Textures-C as outlier datasets and ResNet-50. We report baselines and use evaluation metrics as provided by [46]. Our results are averaged over five runs. The conclusion is similar, improvement (▲) over common methods and performs the best (bold).

References

[1] 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, 2025. 8

[2] Ali Bahri, Moslem Yazdanpanah, Mehrdad Noori, Sahar Dastani Oghani, 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 *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 266–275, 2025. 8

[3] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8344–8353, 2022. 3

[4] Marco Colussi, Sergio Mascetti, Jose Dolz, and Christian Desrosiers. Rec-ttt: Contrastive feature reconstruction for test-time training. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 6699–6708, 2025. 8

[5] Hamidreza Dastmalchi, Aijun An, Ali Cheraghian, Shafin Rahman, and Sameera Ramasinghe. Test-time adaptation of 3d point clouds via denoising diffusion models. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 1566–1576, 2025. 8

[6] Giacomo De Bernardi, Sara Narteni, Enrico Cambiaso, and Maurizio Mongelli. Rule-based out-of-distribution detection. *IEEE Transactions on Artificial Intelligence*, 5(6):2627–2637, 2023. 3

[7] Fnu Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit S Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham M. Kakade, Ali Farhadi, and Prateek Jain. Matformer: Nested transformer for elastic inference, 2024. 4

[8] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International conference on machine learning*, pages 1889–1898. PMLR, 2016. 3

[9] Chaoqun Du, Yulin Wang, Jiayi Guo, Yizeng Han, Jie Zhou, and Gao Huang. Unitta: Unified benchmark and versatile framework towards realistic test-time adaptation. *arXiv preprint arXiv:2407.20080*, 2024. 8

[10] Abhimanyu Dubey, Vignesh Ramanathan, Alex Pentland, and Dhruv Mahajan. Adaptive methods for real-world domain generalization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 14340–14349, 2021. 2

[11] Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael Jordan, and Zheng-Jun Zha. Rank diminishing in deep neural networks. *Advances in Neural Information Processing Systems*, 35:33054–33065, 2022. 2

[12] 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, 2023. 4, 5, 9

[13] Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and J Zico Kolter. Test time adaptation via conjugate pseudo-labels. In *Advances in Neural Information Processing Systems*, 2022. 4, 6

[14] Wenhao Gu, Li Gu, Ziqiang Wang, Ching Y Suen, and Yang Wang. Doctttt: Test-time training for handwritten document recognition using meta-auxiliary learning. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 1904–1913, 2025. 8

[15] Zirun Guo and Tao Jin. Smoothing the shift: Towards stable test-time adaptation under complex multimodal noises. In *The Thirteenth International Conference on Learning Representations*, 2021. 3

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2

[17] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. 6

[18] Raza Imam, Hanan Gani, Muhammad Huzaifa, and Karthik Nandakumar. Test-time low rank adaptation via confidence maximization for zero-shot generalization of vision-language models. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 5449–5459, 2025. 8

[19] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *Advances in Neural Information Processing Systems*, 2021. 2

[20] Byungjai Kim, Chanho Ahn, Wissam J. Baddar, Kikyung Kim, HUIJIN LEE, Saehyun Ahn, Seungju Han, Sungjoo Suh, and Eunho Yang. Test-time ensemble via linear mode connectivity: A path to better adaptation. In *The Thirteenth International Conference on Learning Representations*, 2025. 8

[21] Manuel Knott, Ignacio Serna, Ethan Mann, and Pietro Perona. A rapid test for accuracy and bias of face recognition technology. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 7731–7740, 2025. 8

[22] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022. 2, 3, 9

[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 *International Conference on Learning Representations*, 2024. 2, 6, 7, 8, 9

[24] Mingxi Lei, Chunwei Ma, Meng Ding, Yufan Zhou, Ziyun Huang, and Jinhui Xu. TTVD: Towards a geometric framework for test-time adaptation based on voronoi diagram. In *The Thirteenth International Conference on Learning Representations*, 2025. 8

[25] Yushu Li, Xun Xu, Yongyi Su, and Kui Jia. On the robustness of open-world test-time training: Self-training with dynamic prototype expansion. In *IEEE International Conference on Computer Vision*, 2023. 4, 5, 9

[26] 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*, 2020. 6

[27] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In *Advances in Neural Information Processing Systems*, 2021. 2

[28] Tianyi Ma and Maoying Qiao. Disentangle source and target knowledge for continual test-time adaptation. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 8013–8023, 2025. 8

[29] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020. 4, 5, 9

[30] 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*, 2022. 4, 5, 9

[31] 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 *Proc. ICLR*, 2022. 4, 5, 9

[32] 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 *International Conference on Learning Representations*, 2023. 4, 7, 9

[33] David Osowiechi, Gustavo A Vargas Hakim, Mehrdad Noori, Milad Cheraghalikhani, Ismail Ben Ayed, and Christian Desrosiers. Tttflow: Unsupervised test-time training with normalizing flow. In *Winter Conference on Applications of Computer Vision*, pages 2126–2134, 2023. 2

[34] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 754–763, 2017. 4

[35] Viraj Prabhu, Shivam Khare, Deeksha Kartik, and Judy Hoffman. Sentry: Selective entropy optimization via committee consistency for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8558–8567, 2021. 9

[36] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019. 6

[37] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: visual explanations from deep networks via gradient-based localization. *International journal of computer vision*, 128:336–359, 2020. 6

[38] Elaine Sui, Xiaohan Wang, and Serena Yeung-Levy. Just shift it: Test-time prototype shifting for zero-shot generalization with vision-language models. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 825–835, 2025. 8

[39] Guillaume Vray, Devavrat Tomar, Xufeng Gao, Jean-Philippe Thiran, Evan Shelhamer, and Behzad Bozorgtabar. Reservoirtta: Prolonged test-time adaptation for evolving and recurring domains. *arXiv preprint arXiv:2505.14511*, 2025. 8

[40] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 2, 4, 5, 6, 7, 9

[41] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022. 4, 5, 9

[42] Zehao Xiao and Cees GM Snoek. Beyond model adaptation at test time: A survey. *arXiv preprint arXiv:2411.03687*, 2024. 2

[43] Zehao Xiao, Xiantong Zhen, Ling Shao, and Cees G M Snoek. Learning to generalize across domains on single test samples. In *International Conference on Learning Representations*, 2022. 2

[44] Zehao Xiao, Xiantong Zhen, Shengcai Liao, and Cees G M Snoek. Energy-based test sample adaptation for domain generalization. In *International Conference on Learning Representations*, 2023. 2

[45] Xu Yang, Xuan Chen, Moqi Li, Kun Wei, and Cheng Deng. A versatile framework for continual test-time domain adaptation: Balancing discriminability and generalizability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23731–23740, 2024. 8

[46] 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*, 2024. 4, 5, 6, 7, 9

[47] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 15922–15932, 2023. 4, 5, 9

[48] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014. 2

[49] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. In *Advances in Neural Information Processing Systems*, pages 38629–38642, 2022. 9

[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*, 2023. 4

[51] Yunbei Zhang, Akshay Mehra, and Jihun Hamm. Ot-vp: Optimal transport-guided visual prompting for test-time adaptation. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 1122–1132, 2025. 8

[52] Haiteng Zhao, Chang Ma, Qinyu Chen, and Zhi-Hong Deng. Domain adaptation via maximizing surrogate mutual information. *arXiv preprint arXiv:2110.12184*, 2021. 3