Test-time Adaptation for Regression by Subspace Alignment

Kazuki Adachi* Shin'ya Yamaguchi*† Atsutoshi Kumagai*
*NTT Corporation †Kyoto University
{kazuki.adachi,shinya.yamaguchi,atsutoshi.kumagai}@ntt.com

Abstract

This paper investigates test-time adaptation for regression*, where a regression model pre-trained in a source domain is adapted to an unknown target distribution with unlabeled target data. Although regression is a fundamental task in machine learning, most of the existing TTA methods have classification-specific designs, which assume that models output class-categorical predictions, whereas regression models typically output only single scalar values. To enable TTA for regression, we adopt a feature alignment approach, which aligns the feature distributions between the source and target domains to mitigate the domain gap. However, we found that a naïve feature alignment is ineffective or even worse for regression because the features are distributed in a small subspace and many of the raw feature dimensions have little significance to the output. Here, for an effective feature alignment in TTA for regression, we propose* Significant-subspace Alignment (SSA)*. SSA consists of two components: a subspace detection and dimension weighting. First, the subspace detection finds the feature subspace that is representative and significant to the output. Then, the feature alignment in the subspace is performed during TTA. Second, the dimension weighting raises the importance of the dimensions of the feature subspace that have greater significance to the output. We experimentally show that SSA outperforms various baselines on real-world image datasets.*

1. Introduction

Deep neural networks have achieved remarkable success in various tasks [9, 18, 27, 29]. In particular, regression, which is one of the fundamental tasks in machine learning, is widely used in practical tasks such as human pose estimation or age prediction [28]. The successes of deep learning have usually relied on the assumption that the training and test datasets are sampled from an i.i.d. distribution. In the real world, however, such an assumption is often invalid since the test data are sampled from distributions different from the training one due to distribution shifts caused by changes in environments. The performance of these models thus deteriorates when a distribution shift occurs [20, 42]. To address this problem, *test-time adaptation (TTA)* [33] has been studied. TTA aims at adapting a model pre-trained in a source domain to the target domain with only unlabeled target data.

However, most of the existing TTA methods are designed for classification; that is, TTA for regression has not been explored much [33]. Regarding classification, a representative TTA method is Tent [46], which minimizes the entropy of the predictive distribution during testing. The subsequent methods of Tent [38, 52] also adopt the entropy minimization approach. However, the entropy minimization is classification-specific because it assumes that a model directly outputs predictive distributions, *i.e.*, a probability for each class. On the other hand, typical regression models output only single scalar values, not distributions. Thus, we cannot use the entropy minimization approach for regression models.

In this paper, we address TTA for regression. Since entropy cannot be computed in ordinary regression models, we adopt another TTA approach, feature alignment, that does not rely on entropy. Feature alignment methods preliminarily compute the statistics of intermediate features of the source dataset after pre-training in the source domain [1, 11, 22, 24, 26]. Then, upon moving to the target domain, the feature distribution of the target data is aligned to the source distribution by matching the target feature statistics with the pre-computed source ones without accessing the source dataset. This approach is applicable to regression because it allows arbitrary forms of the model output.

However, we found that naïvely applying feature alignment to regression does not work well. This is because regression models trained with standard mean squared error (MSE) loss tend to make features less diverse than classification models do [49]. In particular, we experimentally observed that the features of a trained regression model are distributed in only a small subspace of the entire feature space (Tab. 1). Due to this property, naïvely aligning the entire feature space results in poor alignment in the subspace

since many of the entire feature dimensions have small effects on the subspace, which leads to ineffective and unstable performance in TTA.

To resolve this problem in TTA for regression, we propose *Significant-subspace Alignment (SSA)*. SSA consists of two components based on the aforementioned observation: *subspace detection* and *dimension weighting*. Subspace detection uses principal component analysis (PCA) to find a subspace of the feature space in which the features are concentrated. This subspace is representative and significant to the model output. Then, we perform feature alignment within this subspace, which improves the effectiveness and stability of TTA. Further, in regression, a feature vector is finally projected onto a one-dimensional line so as to output a scalar value. Thus, the subspace dimensions that have an effect on the line need a precise feature alignment. To do so, dimension weighting raises the importance of the subspace dimensions with respect to their effect on the output.

We conducted experiments on various regression tasks, such as SVHN-MNIST [30, 36], UTKFace [50], Biwi Kinect [13], and California Housing [39]. The results showed that SSA outperforms existing TTA baselines that were originally designed for classification.

Our main contributions are summarized as follows:

- We observed that the features of trained regression models tend to be distributed in small subspaces within the entire feature spaces, which makes naïve feature alignment approaches unstable and ineffective in regression.
- We propose SSA, the first TTA method specialized for regression. SSA detects the subspace of the feature space in which the features are concentrated and performs feature alignment in the subspace. In addition, dimension weighting prioritizes the subspace dimensions on the basis of the significance to the output.
- We experimentally show that SSA outperforms the existing TTA baselines on image and tabular regression tasks. We observed that SSA keeps the target features fit within the source subspace during TTA, which is the key to TTA in regression.

Dataset	#Valid dims.	#Subspace dims.
SVHN	353	14
UTKFace	2041	76
Biwi Kinect (Male, Pitch)	677	33
Biwi Kinect (Male, Yaw)	735	12
Biwi Kinect (Male, Roll)	640	39
Biwi Kinect (Female, Pitch)	699	40
Biwi Kinect (Female, Yaw)	823	34
Biwi Kinect (Female, Roll)	704	49

Table 1. Number of valid (having non-zero variance) feature dimensions and feature subspace dimensions (*i.e.*, the rank of the feature covariance matrix). Although the original feature space has 2048 dimensions in the experiment, the features are distributed within the subspaces that have less than a hundred dimensions. See Sec. 5.3.1 for more details.

2. Related Work

2.1. Unsupervised Domain Adaptation

Unsupervised domain adaptation (UDA) has been actively studied as a way to transfer knowledge in the source domain to the target domain [8]. Theoretically, it is known that the upper bound of the error on the target domain includes a distribution gap between the source and target domains [3, 15, 37]. For regression, Cortes & Mohri [7] theoretically explored regression UDA. RSD [5] and DARE-GRAM [35] take into account that the feature scale matters in regression and explicitly align the feature scale during the feature alignment. However, UDA requires the source and target datasets to be accessed simultaneously during training, which can be restrictive when datasets cannot be accessed due to privacy or security concerns, or storage limitations.

More recently, source-free domain adaptation (SFDA), which does not access the source dataset during adaptation, has been studied [6, 19, 31, 32, 43, 47]. The SFDA setting is similar to TTA in that SFDA adapts models with only unlabeled target data. However, SFDA requires to store the whole target dataset and access the dataset for multiple epochs to train additional models [6, 19, 31, 43, 47] or perform clustering [32].

On the other hand, TTA does not train additional models nor access the target dataset for multiple epochs, which enables instant adaptation with low computational resource and storage.

2.2. Test-time Training

Test-time training (TTT) is also similar to TTA as it adapts models with unlabeled target data. The main difference is that TTT requires to modify the model architecture and training procedure in the source domain. The main approach of TTT is additionally training an self-supervised branch simultaneously with the main supervised task in the source domain. Then, during adaptation, the model is updated via minimizing the self-supervised loss on the target data. On the basis of this approach, TTT methods with various self-supervised tasks have been proposed such as rotation prediction [45], contrastive learning [34], clustering [17], or distribution modeling with normalizing flow [40]. However, training with additional loss prohibits the use of off-the-shelf pre-trained models or may potentially affect the performance on the main task.

In contrast, TTA accepts arbitrary training methods in the source domain and thus off-the-shelf-models can be

adapted.

2.3. Test-time Adaptation

Test-time adaptation (TTA) aims to adapt a model trained on the source domain to the target domain without accessing the source data [33]. The difference between UDA and TTA is that TTA requires only unlabeled target data while UDA requires both labeled source data and unlabeled target data. TTA for classification has attracted attention for its practicality. Various types of TTA methods have been studied.

Entropy-based. Wang *et al*. [46] found that the entropy of prediction strongly correlates with accuracy on the target domain and proposed test-time entropy minimization (Tent), which is the most representative of the TTA methods. BACS [52], MEMO [48], and EATA [38] follow the idea of Tent and improve adaptation performance. T3A [23] adjusts the prototype in the feature space during testing. However, these TTA methods are designed for classification and cannot be applied to regression. For instance, computing entropy, which is widely adopted in TTA [23, 38, 46, 51, 52], requires a predictive probability for each class, whereas ordinary regression models only output a single predicted value. Thus, we investigate an approach that does not rely on entropy.

Feature alignment. Another approach of TTA is feature alignment [1, 4, 11, 51]. It is based on the insight about UDA and makes the target feature distribution close to the source one. Since accessing the source data is restricted in the TTA setting, methods based on feature alignment match the statistics of the target features to those of the precomputed source. BN-adapt [4] updates the feature mean and variance stored in batch normalization (BN) layers [21]. DELTA [51] modifies BN and introduces class-wise loss re-weighting. BUFR [11] and CAFe [1] incorporates precomputed source statistics. Although some of these methods are directly applicable to regression, we have observed that they are not effective or even degrade regression performance.

Input adaptation. This approach aims at not updating models but modifying inputs. They use image transformation models [12, 16] or trainable visual prompt [14]. They are also oriented to classification and regression has not been explored.

3. Problem Setting

We consider a setting with a neural network regression model $f_\theta : \mathcal{X} \to \mathbb{R}$ pre-trained on a labeled source dataset $\mathcal{S} = \{(\mathbf{x}_i^{\mathrm{s}}, y_i^{\mathrm{s}}) \in \mathcal{X} \times \mathbb{R}\}_{i=1}^{N_{\mathrm{s}}}$, where $\mathbf{x}_i^{\mathrm{s}}$ and y_i^{s} are an input and its label, and $\mathcal{X}$ is the input space. Our goal is to adapt f_θ to the target domain by using an unlabeled target dataset $\mathcal{T} = \{\mathbf{x}_i^{\mathrm{t}} \in \mathcal{X}\}_{i=1}^{N_{\mathrm{t}}}$ without accessing $\mathcal{S}$. Note that the target labels $y_i^{\mathrm{t}} \in \mathbb{R}$ are not available. In the source dataset $\mathcal{S}$, the data $\{(\mathbf{x}_i^{\mathrm{s}}, y_i^{\mathrm{s}})\}$ are sampled from the source distribution p_{s} over $\mathcal{X} \times \mathbb{R}$. In the target dataset $\mathcal{T}$, we assume covariate shift [44], which is a distribution shift that often occurs in the real world. In other words, the target data $\mathbf{x}_i^{\mathrm{t}}$ are sampled from the target distribution p_{t} over $\mathcal{X}$ that is different from p_{s}, but the predictive distribution is the same, *i.e.*, $p_{\mathrm{s}}(\mathbf{x}) \neq p_{\mathrm{t}}(\mathbf{x})$ and $p_{\mathrm{s}}(y|\mathbf{x}) = p_{\mathrm{t}}(y|\mathbf{x})$.

We split the regression model f_θ into a feature extractor $g_\phi : \mathcal{X} \to \mathbb{R}^D$ and linear regressor $h_\psi(\mathbf{z}) = \mathbf{w}^\top \mathbf{z} + b$, where ϕ and $\psi = (\mathbf{w}, b)$ are the parameters of the models ($\mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}$), and D is the number of feature dimensions. The whole regression model using the feature extractor and linear regressor is denoted by $f_\theta = h_\psi \circ g_\phi$, where $\theta = (\phi, \psi)$.

4. Test-time Adaptation for Regression

In this section, we describe the basic idea behind *Significant-subspace Alignment (SSA)* in Sec. 4.1 and describe it in detail in Sec. 4.2.

4.1. Basic Idea: Feature Alignment

The basic idea of our TTA method for regression is to align the feature distributions of the source and target domains instead of using entropy minimization, as usually done in TTA for classification. As we assume a covariate shift where only the input distribution changes, we update the feature extractor g_ϕ to pull back the target feature distribution to the source one. Here, we describe a naïve implementation of the idea and its problem.

First, in the source domain, we compute the source feature statistics (mean and variance of each dimension) on $\mathcal{S}$ after the source training:

$$\boldsymbol{\mu}^{\mathrm{s}} = \frac{1}{N_{\mathrm{s}}} \sum_{i=1}^{N_{\mathrm{s}}} \mathbf{z}_i^{\mathrm{s}}, \tag{1}$$

$$\boldsymbol{\sigma}^{\mathrm{s}\,2} = \frac{1}{N_{\mathrm{s}}} \sum_{i=1}^{N_{\mathrm{s}}} (\mathbf{z}_i^{\mathrm{s}} - \boldsymbol{\mu}^{\mathrm{s}}) \odot (\mathbf{z}_i^{\mathrm{s}} - \boldsymbol{\mu}^{\mathrm{s}}), \tag{2}$$

where $\mathbf{z}_i^{\mathrm{s}} = g_\phi(\mathbf{x}_i^{\mathrm{s}}) \in \mathbb{R}^D$ is a source feature and $\odot$ is the element-wise product.

Then, we move to the target domain, where we cannot access the source dataset $\mathcal{S}$. Given a target mini-batch $\mathcal{B} = \{\mathbf{x}_i^{\mathrm{t}}\}_{i=1}^{B}$ sampled from $\mathcal{T}$, we compute the mini-batch mean and variance $\hat{\boldsymbol{\mu}}^{\mathrm{t}}$ and $\hat{\boldsymbol{\sigma}}^{\mathrm{t}\,2}$ analogously to Eqs. (1) and (2).

For feature alignment, we seek to make the target statistics similar to the source ones. For this purpose, we use the KL-divergence as Nguyen *et al*. [37] proved that it is included in an upper bound of the target error in unsupervised domain adaptation. Concretely, we minimize the KL-divergence between two diagonal Gaussian distri-

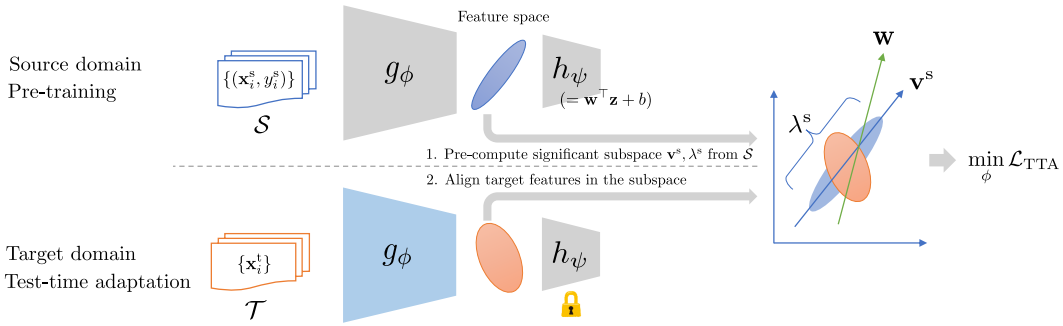


Figure 1. Overview of significant-subspace alignment (SSA).

butions $\mathcal{N}(\boldsymbol{\mu}^{\mathrm{s}}, \boldsymbol{\sigma}^{\mathrm{s}\,2})$ and $\mathcal{N}(\hat{\boldsymbol{\mu}}^{\mathrm{t}}, \hat{\boldsymbol{\sigma}}^{\mathrm{t}\,2})$:

$$\mathcal{L}_{\mathrm{TTA}}(\phi) = \sum_{d=1}^{D} D_{\mathrm{KL}}\left(\mathcal{N}(\mu_d^{\mathrm{s}}, \sigma_d^{\mathrm{s}\,2}) \| \mathcal{N}(\hat{\mu}_d^{\mathrm{t}}, \hat{\sigma}_d^{\mathrm{t}\,2})\right) + D_{\mathrm{KL}}\left(\mathcal{N}(\hat{\mu}_d^{\mathrm{t}}, \hat{\sigma}_d^{\mathrm{t}\,2}) \| \mathcal{N}(\mu_d^{\mathrm{s}}, \sigma_d^{\mathrm{s}\,2})\right), \quad (3)$$

where the subscripts d represent the d-th elements of the mean and variance vectors. Here, we used both directions of the KL-divergence because it empirically had good results as recommended by Nguyen *et al*. [37]. The KL-divergence between two univariate Gaussians can be written in a closed form as [10]

$$D_{\mathrm{KL}}\left(\mathcal{N}(\mu_1, \sigma_1^2) \| \mathcal{N}(\mu_2, \sigma_2^2)\right) = \frac{1}{2}\left(\log \frac{\sigma_2^2}{\sigma_1^2} + \frac{(\mu_1 - \mu_2)^2 + \sigma_1^2}{\sigma_2^2} - 1\right). \quad (4)$$

However, in regression models, the features tend to be less diverse than in classification [49]. In fact, we observed that the features of regression models trained on $\mathcal{S}$ are distributed in only small subspaces of the feature spaces and many dimensions of the feature spaces had zero variances. Tab. 1 shows the numbers of valid (having non-zero variance) feature dimensions and feature subspace dimensions (see Sec. 5.3.1). This property makes the naïve feature alignment described above unstable since the KL-divergence in Eq. (4) includes the variance in the denominator. Also, this naïve feature alignment is ineffective because many of the feature dimensions have a small effect on the subspace.

4.2. Significant-subspace Alignment

In this section, we describe our method, Significant-subspace Alignment (SSA), to tackle the aforementioned problem of naïve feature alignment.

Fig. 1 shows an overview of SSA. As described in Sec. 4.1, the features of a regression model tend to be distributed in a small subspace of the feature space. Thus, we introduce *subspace detection* to detect a subspace that is representative and significant to the output and then perform feature alignment in the subspace. Subspace detection is similar to principal component analysis (PCA). Further, in the regression model we consider, a feature $\mathbf{z} = g_\phi(\mathbf{x})$ is projected onto a one-dimensional line determined by $\mathbf{w}$ of h_ψ in order to output a scalar value $\mathbf{w}^\top \mathbf{z} + b$. Within the subspace, there are dimensions that have smaller contribution (λ^{s}) but have an effect on the line. We use *dimension weighting* to prioritize such dimensions.

Subspace Detection. After the training on the source dataset $\mathcal{S}$, we detect the subspace in which the source features are distributed. Instead of computing the variance of each dimension as Eq. (2), we compute the covariance matrix:

$$\boldsymbol{\Sigma}^{\mathrm{s}} = \frac{1}{N_{\mathrm{s}}} \sum_{i=1}^{N_{\mathrm{s}}} (\mathbf{z}_i^{\mathrm{s}} - \boldsymbol{\mu}^{\mathrm{s}})(\mathbf{z}_i^{\mathrm{s}} - \boldsymbol{\mu}^{\mathrm{s}})^\top \quad (5)$$

where the source mean vector $\boldsymbol{\mu}^{\mathrm{s}}$ is the same as Eq. (1).

Then, we detect the source subspace. On the basis of PCA, the subspace is spanned by the eigenvectors of the covariance matrix $\boldsymbol{\Sigma}^{\mathrm{s}}$, denoted by $\mathbf{v}_k^{\mathrm{s}}$ ($\|\mathbf{v}_k^{\mathrm{s}}\|_2 = 1$). The corresponding eigenvalues λ_k^{s} represent the variance of the source features along the direction $\mathbf{v}_k^{\mathrm{s}}$. We use the top-$K$ largest eigenvalues $\lambda_1^{\mathrm{s}}, \ldots, \lambda_K^{\mathrm{s}}$ ($\lambda_1^{\mathrm{s}} > \cdots > \lambda_K^{\mathrm{s}}$), the corresponding source bases $\mathbf{v}_1^{\mathrm{s}}, \ldots, \mathbf{v}_K^{\mathrm{s}}$, and the source mean $\boldsymbol{\mu}^{\mathrm{s}}$ as the source statistics.

Dimension weighting. We weight the subspace dimensions for prioritizing the feature alignment of each dimension on the basis of its significance to the model output. Since we assume that the output is computed with a linear regressor $h_\psi(\mathbf{z}) = \mathbf{w}^\top \mathbf{z} + b$, only the direction along $\mathbf{w}$ affects the output in the feature subspace. Thus, precise feature alignment is necessary along the direction of $\mathbf{w}$.

We determine the weight of each subspace dimension as follows:

$$\alpha_d = 1 + |\mathbf{w}^\top \mathbf{v}_d^{\mathrm{s}}|. \quad (6)$$

Feature Alignment. This step is done in the target domain. Given a target mini-batch $\mathcal{B}$ sampled from the target dataset $\mathcal{T}$, we project the target features $\mathbf{z}_i^{\mathrm{t}} = g_\phi(\mathbf{x}_i^{\mathrm{t}})$ into the source subspace and then compute the feature alignment

Algorithm 1 Significant-subspace alignment (SSA).

Input: Pre-trained source model f_θ, source bases $\mathbf{V}^{\mathrm{s}}$, source mean $\boldsymbol{\mu}^{\mathrm{s}}$, source variances $\boldsymbol{\lambda}^{\mathrm{s}}$, target dataset $\mathcal{T}$
Output: Adapted model $f_{\theta'}$
Compute weights for each dimension of the source subspace α_d according to Eq. (6)
for all mini-batch $\{\mathbf{x}_i^{\mathrm{t}}\}_i$ in $\mathcal{T}$ **do**
  Extract target features $\{\mathbf{z}_i^{\mathrm{t}} = g_\phi(\mathbf{x}_i^{\mathrm{t}})\}_i$
  Project target features $\{\mathbf{z}_i^{\mathrm{t}}\}_i$ into $\{\tilde{\mathbf{z}}_i^{\mathrm{t}}\}_i$ according to Eq. (7)
  Compute projected target mean $\tilde{\boldsymbol{\mu}}^{\mathrm{t}}$ and variances $\tilde{\boldsymbol{\sigma}}^{\mathrm{t}\,2}$ analogously to Eqs. (1) and (2)
  Update feature extractor g_ϕ to minimize $\mathcal{L}_{\mathrm{TTA}}$ according to Eq. (8)
end for

loss. The projection of the target feature is computed as follows:

$$\tilde{\mathbf{z}}_i^{\mathrm{t}} = \mathbf{V}^{\mathrm{s}}(\mathbf{z}_i^{\mathrm{t}} - \boldsymbol{\mu}^{\mathrm{s}}), \tag{7}$$

where $\mathbf{V}^{\mathrm{s}} = [\mathbf{v}_1^{\mathrm{s}}, \dots, \mathbf{v}_K^{\mathrm{s}}]^\top \in \mathbb{R}^{K \times D}$. With $\tilde{\mathbf{z}}_i^{\mathrm{t}} \in \mathbb{R}^K$, we compute the projected target mean and variance over the mini-batch analogously to Eqs. (1) and (2); this is denoted by $\tilde{\boldsymbol{\mu}}^{\mathrm{t}}$ and $\tilde{\boldsymbol{\sigma}}^{\mathrm{t}\,2}$. On the other hand, the projected source mean and variance are $\mathbf{0}$ and the eigenvalues $\boldsymbol{\lambda}^{\mathrm{s}} = [\lambda_1^{\mathrm{s}}, \dots, \lambda_K^{\mathrm{s}}]$ since $\boldsymbol{\Sigma}^{\mathrm{s}}\mathbf{v}_k^{\mathrm{s}} = \lambda_k^{\mathrm{s}}\mathbf{v}_k^{\mathrm{s}}$. Thus, the KL-divergence in the detected subspace is computed between two K-dimensional diagonal Gaussians $\mathcal{N}(\mathbf{0}, \boldsymbol{\lambda}^{\mathrm{s}})$ and $\mathcal{N}(\tilde{\boldsymbol{\mu}}^{\mathrm{t}}, \tilde{\boldsymbol{\sigma}}^{\mathrm{t}\,2})$.

Using the subspace detection and dimension weighting, the loss of SSA is:

$$\begin{aligned}\mathcal{L}_{\mathrm{TTA}}(\phi) &= \sum_{d=1}^{K} \alpha_d \left\{ D_{\mathrm{KL}}\left(\mathcal{N}(0, \lambda_d^{\mathrm{s}}) \| \mathcal{N}(\tilde{\mu}_d^{\mathrm{t}}, \tilde{\sigma}_d^{\mathrm{t}\,2})\right) \right. \\ &\qquad \left. + D_{\mathrm{KL}}\left(\mathcal{N}(\tilde{\mu}_d^{\mathrm{t}}, \tilde{\sigma}_d^{\mathrm{t}\,2}) \| \mathcal{N}(0, \lambda_d^{\mathrm{s}})\right) \right\} \\ &= \frac{1}{2} \sum_{d=1}^{K} \alpha_d \left(\frac{(\tilde{\mu}_d^{\mathrm{t}})^2 + \lambda_d^{\mathrm{s}}}{\tilde{\sigma}_d^{\mathrm{t}\,2}} + \frac{(\tilde{\mu}_d^{\mathrm{t}})^2 + \tilde{\sigma}_d^{\mathrm{t}\,2}}{\lambda_d^{\mathrm{s}}} - 2 \right).\end{aligned} \tag{8}$$

During TTA, we optimize the feature extractor g_ϕ to minimize $\mathcal{L}_{\mathrm{TTA}}$, *i.e.*, we seek $\phi^* = \min_\phi \mathcal{L}_{\mathrm{TTA}}(\phi)$. We update only the affine parameters γ and β of the normalization layers such as batch normalization [21] or layer normalization [2] to avoid forgetting the source knowledge inspired by Tent [46].

The procedure of SSA is listed in Algorithm 1.

5. Experiment

We evaluated SSA and the baselines on various regression tasks. First, we checked whether the learned features are distributed in a small subspace (Sec. 5.3.1) and then evaluated the regression performance (Secs. 5.3.2 and 5.3.3). We also analyzed the feature subspace and distributions (Secs. 5.3.4 and 5.3.5).

5.1. Dataset

We selected regression datasets with two types of covariate shift, *i.e.*, domain shift and image corruption.

SVHN-MNIST. SVHN [36] and MNIST [30] are famous digit-recognition datasets. Although they are mainly used for classification, we used them for regression by training models to directly output a scalar value of the labels. We used SVHN and MNIST as the source and target domains, respectively.

UTKFace [50]. UTKFace is a dataset consisting of face images. The task is to predict the age of the person in an input image. For the source model, we trained models on the original UTKFace images. For the target domain, we added corruptions such as noise or blur to the images. The types of corruption were the same as those of ImageNet-C [20]. We applied 13 types of corruption at the highest severity level of the five levels.

Biwi Kinect [13]. Biwi Kinect is a dataset consisting of person images. The task is to predict the head pose of the person in an input image in terms of pitch, yaw, and roll angles. We separately trained models to predict each angle. The source and target domains are the gender of the person in the image. We conducted experiments on six combinations of the source/target gender and task, *i.e.*, $\{\text{male} \to \text{female}, \text{female} \to \text{male}\} \times \{\text{pitch}, \text{yaw}, \text{roll}\}$.

5.2. Setting

Source model. We used ResNet-26 [18] for SVHN and ResNet-50 for UTKFace and Biwi Kinect. We modified the last fully-connected layer to output single scalar values and trained the models with the standard MSE loss on each dataset and task.

Test-time adaptation with SSA. We minimized $\mathcal{L}_{\mathrm{TTA}}$ on the target datasets. We used the outputs of the penultimate layer of the model as features, which had 2048 dimensions. We set the number of dimensions of the feature subspace to $K = 100$ as the default throughout the experiments. For optimization, we used Adam [25] with a learning rate $= 0.001$, $(\beta_1, \beta_2) = (0.9, 0.999)$, and weight decay $= 0$, which is the default setting in PyTorch [41]. We set the batch size to 64 following other TTA baselines.

Baseline. Since there are no TTA baselines designed for regression, we compared SSA with TTA methods designed for classification that can be naïvely applied to regression. *Source* simply makes predictions without any model updates. *BN-adapt* [4] updates the feature mean and variance stored in the BN layers during testing. *Feature restoration (FR)* [11] uses the source statistics of the features and outputs (instead of logits in classification) as a form of dimension-wise histogram and aligns the target feature histogram to the source one. *Prototype* is a tweaked version of

Method	R^2	RMSE
Source	$0.406_{\pm 0.00}$	$2.232_{\pm 0.00}$
DANN	$0.307_{\pm 0.09}$	$2.406_{\pm 0.16}$
TTT	$0.288_{\pm 0.02}$	$2.443_{\pm 0.03}$
BN-adapt	$0.396_{\pm 0.00}$	$2.251_{\pm 0.01}$
Prototype	$0.491_{\pm 0.00}$	$2.065_{\pm 0.01}$
FR	$0.369_{\pm 0.01}$	$2.300_{\pm 0.02}$
SSA (ours)	$\mathbf{0.511}_{\pm \mathbf{0.03}}$	$\mathbf{2.024}_{\pm \mathbf{0.06}}$

Table 2. Test R^2 score (higher is better) and RMSE (lower is better) on SVHN-MNIST. The best scores are **bolded**.

T3A [23]; it regards the weight vector $\mathbf{w}$ of the last fully-connected layer as the prototype and replaces it with the mean of the arriving target feature vectors during testing. In addition, we used the following methods other than TTA as baselines: test-time training (*TTT*) [45] incorporates a self-supervised rotation prediction task during pre-training in the source domain; then it updates the model by minimizing the self-supervised loss during testing. *DANN* [15] is an unsupervised domain adaptation method which adversarially trains a feature extractor and domain discriminator to learn domain-invariant features.

5.3. Results

5.3.1 Number of Dimensions of the Feature Subspace

After the pre-training on the source dataset, we counted the numbers of valid feature dimensions (*i.e.*, having non-zero variances) and dimensions of the feature subspace in which the source features are distributed. The latter value corresponds to the number of non-zero eigenvalues of the covariance matrix of the source features in Eq. (5). Tab. 1 shows the numbers of valid feature dimensions and subspace dimensions computed for each source dataset. Although the number of feature dimensions is 2048 in ResNet, many feature dimensions have zero variance because of ReLU activation. This is the cause of the failure of the naïve feature alignment, as described in Sec. 4.1. Moreover, the source features are distributed in only a small subspace with fewer than a hundred dimensions, which is smaller than that of valid feature dimensions. This property makes the naïve feature alignment harder since the alignment of the original feature space is ineffective in the subspace in which the features are actually distributed.

5.3.2 Regression Performance

We evaluated the performance of the regression models in terms of the R^2 score (coefficient of determination), which is widely used in regression tasks. Tab. 2 shows the scores for the SVHN-pretrained model tested on MNIST. SSA outperformed the baselines; some of them even underperformed the Source. This is because the baselines were designed for classification tasks and they affected the feature subspaces learned by the source model (see Sec. 5.3.4). Tab. 3 shows the R^2 scores on the UTKFace data with image corruption. We can see that SSA had the highest R^2 scores for most of the corruption types. In particular, SSA outperformed the baselines by a large margin on noise-type corruption which significantly degraded the performance of Source. Tab. 4 shows the R^2 scores on Biwi Kinect with genders different from the source domains. SSA constantly had higher R^2 scores than the baselines; the baselines' scores sometimes significantly dropped or even diverged (Prototype).

In summary, SSA consistently improved the scores whereas the baselines sometimes even underperformed Source.

5.3.3 Ablation Study

We investigated the effect of the subspace detection and dimension weighting by running SSA with and without them. For the SSA variant without subspace detection (*i.e.*, naïvely aligning the original feature space), we simply selected the top-K feature dimensions that had the largest variances. In this case, we directly used the weight of the linear regressor h_ψ to compute the dimension weight α_d as $\alpha_d = 1 + |w_d|$ instead of Eq. (6). Tab. 5 shows the test R^2 scores with and without subspace detection and dimension weighting on each dataset. Without subspace detection, the scores were worse than Source on MNIST and Biwi Kinect, and of the same level as simple baselines like BN-adapt [4] on UTKFace (Tab. 3). In contrast, the subspace detection significantly improved the scores on all three datasets. Dimension weighting also improved the scores, although the gain was smaller than in the case of subspace detection. This is because the variance of the feature subspace dimension correlates with the weight; *i.e.*, the top-K selected dimensions with respect to variance tended to have high importance to the output. However, the dimension weighting raised the importance of the feature dimensions that have low variance but affect the output, which further improved regression performance. Tab. 7 lists the correlation coefficients between α_d and variance λ^s_d.

Next, we investigated the effect of the number of feature subspace dimensions K. We varied K within $\{10, 25, 50, 75, 100, 200, 400, 1000, 2048\}$. Tab. 6 shows the test R^2 scores. Although the best K differs among the datasets, $K = 100$ consistently produced competitive results. With increasing K, the best or competitive scores were when K was close to the number of the subspace dimensions in Tab. 1. This indicates the importance of the subspace feature alignment. When $K \geq 400$ in MNIST and $K \geq 1000$ in Biwi Kinect, the loss became unstable or diverged because SSA attempted to align too many degenerated feature

Method	Defocus blur	Motion blur	Zoom blur	Contrast	Elastic transform	Jpeg comp.	Pixelate	Gaussian noise	Impulse noise	Shot noise	Brightness	Fog	Snow	Mean
Source	0.410	0.159	0.658	−3.906	0.711	0.069	0.595	−2.536	−2.539	−2.522	0.661	−0.029	−0.544	−0.678
DANN	0.512	0.586	0.637	−0.720	0.729	0.698	0.807	−4.341	−3.114	−3.744	0.590	−0.131	−0.425	−0.609
TTT	0.748	0.761	0.773	0.778	0.826	0.772	0.861	0.525	0.532	0.477	0.775	0.397	0.493	0.671
BN-Adapt	0.727	0.759	0.763	0.702	0.826	0.778	0.850	0.510	0.510	0.446	0.790	0.392	0.452	0.654
Prototype	−1.003	−1.020	−1.016	−0.719	−0.967	−0.908	−0.974	−0.514	−0.512	−0.512	−1.004	−0.823	−0.822	−0.830
FR	0.794	**0.839**	0.849	0.756	**0.899**	0.825	**0.946**	0.509	0.522	0.458	0.861	0.408	0.428	0.700
SSA (ours)	**0.803**	**0.839**	**0.851**	**0.792**	**0.899**	**0.829**	0.943	**0.580**	**0.592**	**0.560**	**0.863**	**0.440**	**0.517**	**0.731**

Table 3. Test R^2 scores on UTKFace with image corruption. The best scores are **bolded**.

	Female → Male			Male → Female			
Method	Pitch	Roll	Yaw	Pitch	Yaw	Roll	Mean
Source	0.759	0.956	0.481	0.763	0.791	0.485	0.706
DANN	$0.698_{\pm 0.03}$	$0.826_{\pm 0.03}$	$-0.039_{\pm 0.08}$	$0.711_{\pm 0.01}$	$0.850_{\pm 0.01}$	$0.076_{\pm 0.05}$	$0.520_{\pm 0.02}$
TTT	$-0.062_{\pm 0.20}$	$0.606_{\pm 0.00}$	$0.031_{\pm 0.02}$	$0.750_{\pm 0.00}$	$0.725_{\pm 0.00}$	$-0.321_{\pm 0.00}$	$0.288_{\pm 0.03}$
BN-adapt	$0.771_{\pm 0.00}$	$0.953_{\pm 0.00}$	$0.493_{\pm 0.01}$	$0.832_{\pm 0.00}$	$0.842_{\pm 0.00}$	$\mathbf{0.585_{\pm 0.00}}$	$0.746_{\pm 0.00}$
Prototype	$-318_{\pm 0.00}$	-	-	-	-	-	$-318_{\pm 0.00}$
FR	$-1.27_{\pm 0.70}$	$0.742_{\pm 0.05}$	$-2.69_{\pm 0.79}$	$0.622_{\pm 0.06}$	$0.855_{\pm 0.01}$	$-0.406_{\pm 0.30}$	$-0.357_{\pm 0.23}$
SSA (ours)	$\mathbf{0.860_{\pm 0.00}}$	$\mathbf{0.962_{\pm 0.00}}$	$\mathbf{0.513_{\pm 0.01}}$	$\mathbf{0.869_{\pm 0.00}}$	$\mathbf{0.886_{\pm 0.00}}$	$0.575_{\pm 0.00}$	$\mathbf{0.778_{\pm 0.00}}$

Table 4. Test R^2 scores on Biwi Kinect. The best scores are **bolded**.

Subspace	Weight	SVHN	UTKFace	Biwi Kinect
		0.333	0.642	0.672
	✓	0.338	0.641	0.672
✓		0.508	0.728	**0.778**
✓	✓	**0.511**	**0.731**	**0.778**
Source		0.406	0.020	0.706

Table 5. Test R^2 scores of SSA with and without subspace detection and dimension weighting. Scores averaged over corruption types and gender-task combinations are reported for UTKFace and Biwi Kinect, respectively. The best scores are **bolded**.

K	MNIST	UTKFace	Biwi Kinect
10	0.494	0.693	0.688
25	**0.538**	0.717	0.761
50	0.524	0.728	0.767
75	0.516	**0.732**	0.774
100	0.511	0.731	**0.778**
200	0.496	0.731	0.771
400	-	0.731	0.755
1000	-	**0.732**	-
2048	-	0.725	-

Table 6. Test R^2 scores of SSA for different numbers of feature subspace dimensions K. The best scores are **bolded**.

dimensions. In contrast, although setting $K \geq 1000$ gave the good scores on UTKFace, $K = 100$ produced a competitive score.

Dataset	SVHN	UTKFace	Biwi Kinect (Female, Pitch)
Correlation	0.787	0.917	0.782

Table 7. Correlation coefficients between the top $K = 100$ variances of source features along the source bases λ_d^{s} and weight α_d in Eq. (6).

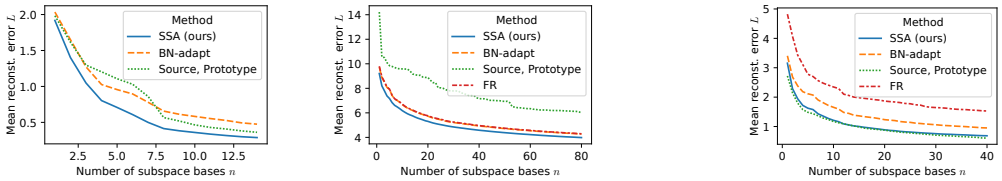
5.3.4 Feature Subspace Analysis

To verify that the reason why the baseline methods degrade the regression performance is that they affect the feature subspace learned by the source model as mentioned in Sec. 5.3.2, we examined the reproducibility of the target features with the source bases $\mathbf{V}^{\mathrm{s}}$ after TTA. That is, the target features can be represented by a linear combination of the source bases if the model retains the source subspace throughout TTA and the target features fit within the subspace. To measure this quantitatively, we computed the reconstruction error L as the Euclidean distance between a target feature vector $\mathbf{z}^{\mathrm{t}}$ and $\mathbf{z}_{\mathrm{r}}^{\mathrm{t}}$, the one reconstructed with n source bases:

$$L = \|\mathbf{z}_{\mathrm{r}}^{\mathrm{t}} - \mathbf{z}^{\mathrm{t}}\|_2,\ \mathbf{z}_{\mathrm{r}}^{\mathrm{t}} = \boldsymbol{\mu}^{\mathrm{s}} + \sum_{d=1}^{n} ((\mathbf{z}^{\mathrm{t}} - \boldsymbol{\mu}^{\mathrm{s}})^{\top} \mathbf{v}_d^{\mathrm{s}}) \mathbf{v}_d^{\mathrm{s}}, \quad (9)$$

where $\mathbf{z}^{\mathrm{t}}$ is a target feature vector extracted with the model after TTA, and n is the number of dimensions of the source subspace listed in Tab. 1.

Fig. 2 plots the reconstruction error L versus n on the three datasets. The error decreased as n increased for all



(a) SVHN (b) UTKFace (Gaussian noise) (c) Biwi Kinect (Female → Male, Pitch)

Figure 2. Reconstruction error of features reconstructed with the source bases relative to the original target features. Note that Source and Prototype are the same, since Prototype does not update the feature extractor of the model. FR [11] is not plotted in (a) because it had huge errors.

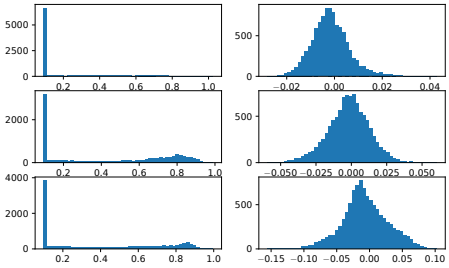


Figure 3. Histograms of three randomly selected target feature dimensions on SVHN-MNIST. **Left**: Original features. **Right**: Projected features.

methods, but SSA reduced the error with a smaller n than in those of the baselines, indicating that it could make the target features fit within the source subspace. Especially in the case of Biwi Kinect (c), the baseline methods produced larger errors than Source; *i.e.*, they broke the learned subspace.

5.3.5 Another Effect of Subspace Detection

For the feature alignment, we used the KL-divergence between two diagonal Gaussian distributions (Eq. (4)) to measure the distribution gap between the source and target features, under the assumption that the features follow a Gaussian distribution. Here, while it is not clear that the assumption actually holds especially when features are output through activation functions like ReLU as in ResNet, we argue that the subspace detection of SSA has, in addition to an effective feature alignment, the effect of making such features follow a distribution close to a Gaussian.

We visualized the histograms of the target features $\mathbf{z}^{\mathrm{t}}_i$ extracted with the source model and ones projected to the source subspace with Eq. (7). Fig. 3 shows the histograms of three randomly selected dimensions of the original and projected features of SVHN-MNIST. In the left column of the figure, the histograms of the original features concentrate on zero because of the ReLU activation and do not follow a Gaussian distribution, which makes the KL-divergence computation with Eq. (4) inaccurate. On the other hand, the histograms of the projected features in the right column are close to Gaussians. Thus, subspace detection makes it easier to align the features with the Gaussian KL-divergence also from this perspective.

The reason why the projected features follow a Gaussian distribution can be interpreted as follows. The k-th element of a projected feature vector $\tilde{z}^{\mathrm{t}}_{i,k}$ is

$$\tilde{z}^{\mathrm{t}}_{i,k} = \sum_{d=1}^{D} (z^{\mathrm{t}}_{i,d} - \mu^{\mathrm{s}}_d) v^{\mathrm{s}}_{k,d}. \tag{10}$$

Here, we regard each term $a_{i,d} := (z^{\mathrm{t}}_{i,d} - \mu^{\mathrm{s}}_d) v^{\mathrm{s}}_{k,d}$ as a random variable. Assuming that $a_{i,d}$ is independent of the feature dimension d, the central limit theorem guarantees that the distribution of the projected features, *i.e.*, the sum of $a_{i,d}$, becomes closer to a Gaussian as the total number of dimensions D increases.

6. Conclusion

We proposed significant-subspace alignment (SSA), a novel test-time adaptation method for regression models. Since we have found that the naïve feature alignment fails in regression TTA because the learned features are distributed in a small subspace, we incorporated subspace detection and dimension weighting procedures into SSA. The subspace detection procedure detects the feature subspace in which source features are distributed and the dimension weighting computes the importance of each dimension of the subspace in order to improve the effectiveness of the feature alignment and retain the source subspace during TTA. Experimental results show that SSA achieved higher R^2 scores on various regression tasks than did baselines that were originally designed for classification tasks. Further, we observed that SSA retains the source subspace and makes the feature distribution closer to a Gaussian, which makes it easier to align the features within the subspace.

Limitation. One limitation of SSA is that it assumes a covariate shift, where $p(y|\mathbf{x})$ does not change. Addressing distribution shifts where $p(y|\mathbf{x})$ changes, *e.g.*, concept drift, will be tackled in future work.

References

[1] Kazuki Adachi, Shin'ya Yamaguchi, and Atsutoshi Kumagai. Covariance-aware feature alignment with pre-computed source statistics for test-time adaptation to multiple image corruptions. In *IEEE International Conference on Image Processing (ICIP)*, 2023. 1, 3

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[3] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010. 2

[4] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Revisiting Batch Normalization for Improving Corruption Robustness. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 494–503, 2021. 3, 5, 6

[5] Xinyang Chen, Sinan Wang, Jianmin Wang, and Mingsheng Long. Representation Subspace Distance for Domain Adaptation Regression. In *International Conference on Machine Learning (ICML)*, 2021. 2

[6] Tong Chu, Yahao Liu, Jinhong Deng, Wen Li, and Lixin Duan. Denoised maximum classifier discrepancy for source-free unsupervised domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 2

[7] Corinna Cortes and Mehryar Mohri. Domain adaptation in regression. In *International Conference on Algorithmic Learning Theory*, pages 308–323. Springer, 2011. 2

[8] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017. 2

[9] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 27:1071–1092, 2020. 1

[10] John Duchi. Derivations for Linear Algebra and Optimization, 2007. http://ai.stanford.edu/~jduchi/projects/general_notes.pdf. 4

[11] Cian Eastwood, Ian Mason, Chris Williams, and Bernhard Schölkopf. Source-Free Adaptation to Measurement Shift via Bottom-Up Feature Restoration. In *International Conference on Learning Representations*, 2022. 1, 3, 5, 8

[12] Shohei Enomoto, Naoya Hasegawa, Kazuki Adachi, Taku Sasaki, Shin'ya Yamaguchi, Satoshi Suzuki, and Takeharu Eda. Test-time Adaptation Meets Image Enhancement: Improving Accuracy via Uncertainty-aware Logit Switching. *arXiv preprint arXiv:2403.17423*, 2024. 3

[13] Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Van Gool. Random Forests for Real Time 3D Face Analysis. *Int. J. Comput. Vision*, 101(3):437–458, 2013. 2, 5

[14] Yulu Gan, Yan Bai, Yihang Lou, Xianzheng Ma, Renrui Zhang, Nian Shi, and Lin Luo. Decorate the newcomers: Visual domain prompt for continual test time adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. 3

[15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 2, 6

[16] Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang. Back to the source: Diffusion-driven adaptation to test-time corruption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[17] Gustavo A. Vargas Hakim, David Osowiechi, Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Ismail Ben Ayed, and Christian Desrosiers. ClusT3: Information Invariant Test-Time Training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2

[18] 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 (CVPR)*, pages 770–778, 2016. 1, 5

[19] Tianlang He, Zhiqiu Xia, Jierun Chen, Haoliang Li, and S-H Gary Chan. Target-agnostic source-free domain adaptation for regression tasks. In *IEEE International Conference on Data Engineering (ICDE)*, 2024. 2

[20] Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 1, 5

[21] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. 3, 5

[22] Masato Ishii and Masashi Sugiyama. Source-free Domain Adaptation via Distributional Alignment by Matching Batch Normalization Statistics. *arXiv preprint arXiv:2101.10842*, 2021. 1

[23] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 34, 2021. 3, 6

[24] Sanghun Jung, Jungsoo Lee, Nanhee Kim, Amirreza Shaban, Byron Boots, and Jaegul Choo. Cafa: Class-aware feature alignment for test-time adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19060–19071, 2023. 1

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[26] Takeshi Kojima, Yutaka Matsuo, and Yusuke Iwasawa. Robustifying vision transformer without retraining from scratch by test-time class-conditional feature alignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1009–1016. International Joint Conferences on Artificial Intelligence Organization, 2022. Main Track. 1

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[28] Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehensive analysis of deep regression. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2065–2081, 2019. 1

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[30] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. MNIST handwritten digit database, 1998. 2, 5

[31] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model Adaptation: Unsupervised Domain Adaptation Without Source Data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[32] 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 (ICML)*, 2020. 2

[33] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *arXiv preprint arXiv:2303.15361*, 2023. 1, 3

[34] 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

[35] Ismail Nejjar, Qin Wang, and Olga Fink. DARE-GRAM: Unsupervised Domain Adaptation Regression by Aligning Inverse Gram Matrices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[36] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning, 2011. 2, 5

[37] A Tuan Nguyen, Toan Tran, Yarin Gal, Philip HS Torr, and Atılım Güneş Baydin. KL Guided Domain Adaptation. *International Conference on Learning Representations*, 2022. 2, 3, 4

[38] 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*, pages 16888–16905. PMLR, 2022. 1, 3

[39] Cam Nugent. the California Housing Prices Dataset. Kaggle, 2017. https://www.kaggle.com/datasets/camnugent/california-housing-prices. 2

[40] 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 *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023. 2

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[42] 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. 1

[43] Sunandini Sanyal, Ashish Ramayee Asokan, Suvaansh Bhambri, Akshay Kulkarni, Jogendra Nath Kundu, and R Venkatesh Babu. Domain-specificity inducing transformers for source-free domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2

[44] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000. 3

[45] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, 2020. 2, 6

[46] 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 (ICLR)*, 2021. 1, 3, 5

[47] Haifeng Xia, Handong Zhao, and Zhengming Ding. Adaptive adversarial network for source-free domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[48] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *Advances in neural information processing systems*, 35:38629–38642, 2022. 3

[49] Shihao Zhang, Linlin Yang, Michael Bi Mi, Xiaoxu Zheng, and Angela Yao. Improving Deep Regression with Ordinal Entropy. In *International Conference on Learning Representations (ICLR)*, 2023. 1, 4

[50] Zhifei Zhang, Yang Song, and Hairong Qi. Age Progression/Regression by Conditional Adversarial Autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 2, 5

[51] Bowen Zhao, Chen Chen, and Shu-Tao Xia. DELTA: DEGRADATION-FREE FULLY TEST-TIME ADAPTATION. In *The Eleventh International Conference on Learning Representations*, 2023. 3

[52] Aurick Zhou and Sergey Levine. Bayesian Adaptation for Covariate Shift. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 3