ActMAD: Activation Matching to Align Distributions for Test-Time-Training

M. Jehanzeb Mirza[‡1,2] Pol Jané Soneira[3] Wei Lin[1,4] Mateusz Kozinski[1]
Horst Possegger[1] Horst Bischof[1,2]

[1]Institute for Computer Graphics and Vision, TU Graz, Austria.
[2]Christian Doppler Laboratory for Embedded Machine Learning.
[3]Institute of Control Systems, KIT, Germany.
[4]Christian Doppler Laboratory for Semantic 3D Computer Vision.

Abstract

Test-Time-Training (TTT) is an approach to cope with out-of-distribution (OOD) data by adapting a trained model to distribution shifts occurring at test-time. We propose to perform this adaptation via Activation Matching *(ActMAD): We analyze activations of the model and align activation statistics of the OOD test data to those of the training data. In contrast to existing methods, which model the distribution of entire channels in the ultimate layer of the feature extractor, we model the distribution of each feature in multiple layers across the network. This results in a more fine-grained supervision and makes ActMAD attain state of the art performance on CIFAR-100C and Imagenet-C. ActMAD is also architecture- and task-agnostic, which lets us go beyond image classification, and score 15.4% improvement over previous approaches when evaluating a KITTI-trained object detector on KITTI-Fog. Our experiments highlight that ActMAD can be applied to online adaptation in realistic scenarios, requiring little data to attain its full performance.*

1. Introduction

When evaluated in laboratory conditions, deep networks outperform humans in numerous visual recognition tasks [6, 8, 9]. But this impressive performance is predicated on the assumption that the test and training data come from the same distribution. In many practical scenarios, satisfying this requirement is either impossible or impractical. For example, a perception system of an autonomous vehicle can be exposed not only to fog, snow, and rain, but also to rare conditions including smoke, sand storms, or substances distributed on the road in consequence of a traffic accident. Unfortunately, distribution shifts between the training and test data can incur a significant performance penalty [1, 20, 25, 27, 34, 36]. To address this shortcoming, test-time-training (TTT) methods attempt to adapt a deep network to the actual distribution of the test data, at test-time.

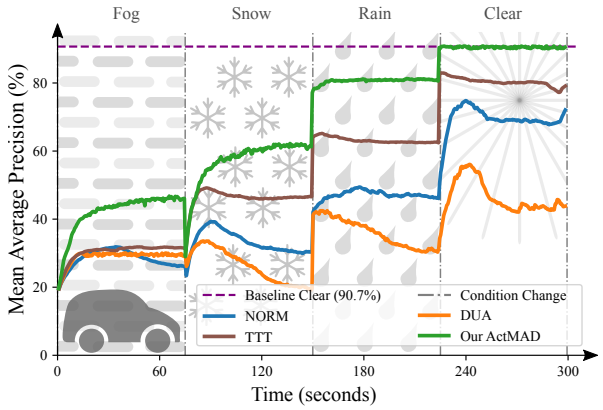

Figure 1. ActMAD is well suited for online test-time adaptation irrespective of network architecture and task. Our intended application is object detection on board of a vehicle driving in dynamically changing and unpredictable weather conditions. Here, we report the class-averaged Mean Average Precision (mAP@50) over adaptation time[1], attained by continuously adapting a KITTI [12]-trained YOLOv3 [29] detector. *Clear* refers to the weather condition of the original dataset, mostly acquired in sunny weather, while *Fog*, *Rain*, and *Snow* are generated by perturbing the original images [15, 18]. Gray vertical lines mark transitions between weather conditions.

Existing TTT techniques limit the performance drop provoked by the shift in test distribution, but the goal of online, data-efficient, source-free, and task-agnostic adaptation remains elusive: the methods that update network parameters by self-supervision on test data [10, 25, 32] can only be used if the network was first trained in a multi-

‡Correspondence: muhammad.mirza@icg.tugraz.at

[1]Time (seconds) for adaptation is hardware specific. For reference, we run these experiments on an RTX-3090 NVIDIA graphics card and it takes ∼75s to adapt to 3741 images (one weather condition) in the KITTI test set.

task setup, the approaches that minimise the entropy of test predictions [24, 34], or use pseudo-labels, or pseudo-prototypes [19, 20], are not easily adapted to object detection or regression, and the ones that update statistics of the batch normalization layers [27, 30] are limited to architectures that contain them. As a consequence of these limitations, while existing methods fare well in image classification, none of them is well suited for the scenario where the need for online adaptation is the most acute, that is, object detection on board a car driving in changing weather conditions.

Our goal is to lift the limitations of the current methods and propose a truly versatile, task-, and architecture-agnostic technique, that would extend TTT beyond image classification and enable its deployment in object detection for automotive applications. To that end, we revisit feature alignment, the classical domain adaptation technique [7, 31, 38], also adopted by several TTT algorithms [20, 25, 27, 30]. It consists in aligning the distribution of test set features to that of the training set. By contrast to previous methods, that align the distributions of entire channels in the ultimate layer of the feature extractor, ActMAD brings the feature alignment approach to another level by individually aligning the distribution of each feature in multiple feature maps across the network. On the one hand, this makes the alignment location-aware. For example, the features from the bottom of the image, most often representing road and other vehicles, are not mixed with features from the top part, more likely to describe trees, buildings, or the sky. On the other hand, it results in a more fine-grained supervision of the adapted network. Our ablation studies show that ActMAD owes most of its performance to these two contributions. Additionally, while several authors suggested aligning higher-order moments [20, 38], we demonstrate that aligning means and variances should be preferred when working with small batches. Unlike methods that only update mean and variance [27, 30], or affine parameters of batch normalization layers [20, 27, 30], ActMAD updates all network parameters. Our approach is architecture- and task-agnostic, and does not require access to the training data or training labels, which is important in privacy-sensitive applications. It does require the knowledge of activation statistics of the training set, but this requirement can be easily satisfied by collecting the data during training, or by computing them on unlabelled data without distribution shift.

Our contribution consists in a new method to use Activation Matching to Align Distributions for TTT, which we abbreviate as ActMAD. Its main technical novelty is that we model the distribution of each point in the feature map, across multiple feature maps in the network. While most previous works focus on combining different approaches to test-time adaptation, ActMAD is solely based on feature alignment. This is necessary for our method to be applicable across different architectures and tasks, but we show that discarding the bells and whistles, while aligning activation distributions in a location-aware manner results in matching or surpassing state-of-the-art performance on a number of TTT benchmarks. Figure 1 presents the performance of ActMAD in a simulated online adaptation scenario, in which an object detector runs onboard a vehicle driven in changing weather conditions. Most existing methods cannot be used in this setup, because they cannot run online, or because they cannot be used for object detection. Note, that our method recovers, and even exceeds, the performance of the initial network once the weather cycle goes back to the conditions in which the detector was trained.

2. Related Work

Unsupervised domain adaptation. The idea of aligning neural feature distributions to bridge the domain gap between training and test data was first used for unsupervised domain adaptation (UDA). In CORAL [31], the alignment is performed by first whitening the target domain features and then re-coloring them with covariances computed on the source domain. Alternative methods train the network to minimize a difference of features statistics, like the higher-order central moments [38], or the higher-order cumulant tensor [2]. These UDA algorithms cannot be used for test-time-training (TTT), because they work offline and require access to the source training data and entire test data. Our ActMAD, and some existing TTT approaches *e.g.* [25, 27], also explicitly compute and align feature statistics of the training and test data, but perform online adaptation without access to the training data.

Test-Time-Training. TTT consists in adapting a network to a distribution shift between the test and training data at test-time without access to the training data. The technique that gave the name to the whole group of methods [32] augments the training routine with a self-supervised task, which enables re-training the network on the test set even though no test labels are available. The original work employed rotation prediction [13] as the auxiliary task, but subsequent works [10, 25] replaced it with the Masked Autoencoder reconstruction task [16] or contrastive learning [3]. These methods improve performance, but cannot be used if the network was not trained with the auxiliary task. Moreover, contrastive learning requires a large portion of unlabelled data to be effective. By contrast, our ActMAD can be used to adapt a network trained with an arbitrary protocol, and requires very little data to attain its full performance.

Foregoing a multi-task training setup, CoTTA [35] employs a student-teacher setup for test-time adaptation and relies on entropy of the predicted class distribution to transfer the knowledge from teacher to the student. ActMAD performs on par with CoTTA on its own benchmark. However, reliance on entropy makes their setup classification-specific.

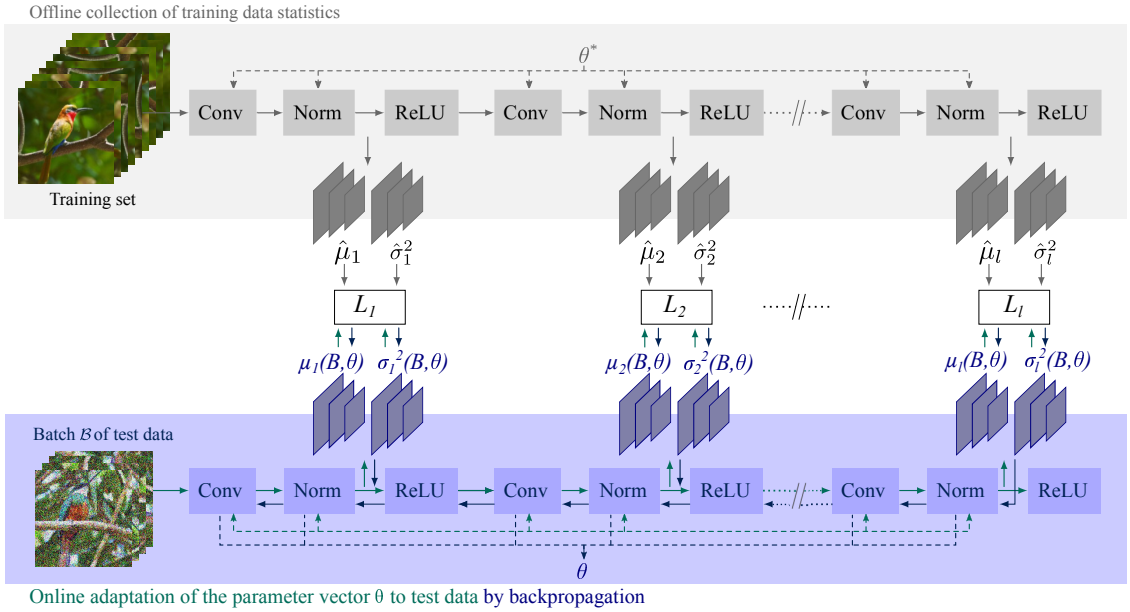

Figure 2. Schematic of ActMAD. Given a pre-trained model and statistics of the clean activations from the training data, it aligns the activation responses from the shifted test data to the clean activations at test-time. We model the activation distributions in terms of the means and variances of each activation, such that the statistics have the same shape as the feature maps. The statistics of the training activations are pre-computed on the training set, or computed on unlabelled data without distribution shift.

While other methods adapt network parameters, T3A [19] casts test-time adaptation as prototype learning. It relies on pseudo-labels to create test-set-specific pseudo-prototypes, which then substitute the classifier learned on the training set. ActMAD outperforms T3A in classifying ImageNet-C and CIFAR-100C images. Similar to the authors of the T3A, Boudiaf *et al*. [1] propose LAME, to forego updating the network parameters and develop an alternative prediction strategy by generating class likelihoods and minimizing an objective function which promotes consistency of predictions coupled with Laplacian regularization. LAME is effective for alleviating the shifts in the prior distribution, but brings little improvement when the shift affects appearance of test images, which is the focus of our work.

Another group of TTT methods adjusts network parameters using entropy-based objectives on the test data. SHOT [24] minimizes the expected entropy of individual predictions, while maximizing the entropy of the predicted classes over the entire test set. TENT [34] adjusts the scale and shift parameters of batch normalization layers to minimize the entropy of test predictions. EATA [28] proposes to selectively minimize entropy of output predictions. ActMAD outperforms SHOT and TENT by a fair margin, but lags behind EATA for image classification, however, the entropy-based loss functions prevent direct application of these methods to object detection and regression.

For online object detection, MemCLR [33] proposes to use a contrastive learning setup. ActMAD out-performs MemCLR in their setup, while being computationally cheaper. Interactron [21] is trained to choose actions that let it record optimal new observations. However, Interactron is not directly comparable with our setup, because our system cannot take autonomous actions.

ActMAD is most closely related to approaches that align statistics of training and test features. Along these lines, NORM [30] and DUA [27] update the batch normalization statistics computed on the training set to match the distribution of the test set. BUFR [7] proposes a bottom-up feature learning strategy to align the features from the source data with those obtained from the distribution shifted test data. CFA [20] updates the affine layers in the network to match the class-conditioned higher-order central moments from the output of the encoder. The reliance on class-conditioning prevents application of CFA to object detection. By contrast, ActMAD is free from this constraint. It aligns means and variances of individual features in multiple layers across the network. ActMAD outperforms DUA, NORM and BUFR, and matches the performance of CFA in image classification.

3. ActMAD

We are given a deep network $f(\mathbf{x};\theta)$, where $\mathbf{x}$ is an input image and θ is a vector of parameters. We are also given a parameter vector θ^*, obtained by training the network on a dataset $\mathcal{S}$ of images and their annotations. Our goal is to use f to process a test set $\mathcal{T}$ of images that may differ in appearance from the ones in $\mathcal{S}$, but are consistent with

them in terms of their content. In many applications, the test images are delivered in a stream, and while we can compose them into batches of moderate size, we are not capable of iterating over the test set. In our flagship application, f is an object detector deployed in a car, $\mathcal{S}$ is a sequence of images acquired on a sunny day, and $\mathcal{T}$ is a video stream acquired during driving the car in changing weather conditions. To address both the detection and classification scenarios, we impose no constraints on the form of $f(\mathbf{x};\theta)$. To ensure maximum versatility of ActMAD, we make no assumptions about the architecture of the network, or the training process.

The requirements of task- and architecture-independence rule out the use of techniques that rely on the output of the network to be a probability distribution [19, 24, 34, 35], or employ auxiliary tasks during training [25, 32]. We thus follow the only viable design choice: we treat selected layers of the network as random variables and align the distribution of these variables in the test set to that of the training set. An overview of our method is presented in Figure 2.

Location-aware activation alignment. We denote the activation responses of the l-th layer of the network f computed for image $\mathbf{x}$ by $a_l(\mathbf{x};\theta)$. This is a feature map of shape $C \times W \times H$, denoting the number of channels and its spatial dimensions. Typically, a_l does not depend on all model parameters θ, but we abuse the notation in the interest of simplicity. Existing TTT methods based on feature alignment [20, 25, 27, 30] treat all the features in a channel as instantiations of the same random variable, irrespective of their location in the image. By contrast, we assume that different locations in the feature map may be distributed differently. This is particularly the case in photographs, because humans frame objects of interest when taking pictures, and in driving scenarios, where different objects occur in different regions of the image. We therefore separately align each of the $C \times W \times H$ activations in the l-th layer. This results in a stronger supervisory signal than the classical approach of integrating over the spatial dimensions and aligning the distributions of C channels at the output of the encoder. Our experiments show that this stronger supervision leads to faster and more effective adaptation.

Aligning means and variances. To align activations, we could use one of the existing feature alignment methods, for example adversarial alignment [11], but this would require access to the training data, which we are not granted. We are therefore limited to methods that characterize the distribution in terms of statistics, which could be pre-computed on the training data, and enforce distribution alignment by minimizing the discrepancy of the test statistics with those of the training set. The classical way to do this is to minimize the difference between higher-order sample moments of the distributions [2, 38]. However, unlike in domain adaptation, where moment-based methods gained popularity, we need to adapt the distributions based on small samples, since we model each location in each channel separately, and run the adaptation online, on small batches. We thus forego higher-order statistics and use feature means

$$\mu_l(\mathcal{D};\theta) = \frac{1}{N_\mathcal{D}} \sum_{\mathbf{x}\in\mathcal{D}} a_l(\mathbf{x};\theta) \tag{1}$$

and variances

$$\sigma_l^2(\mathcal{D};\theta) = \frac{1}{N_\mathcal{D}} \sum_{\mathbf{x}\in\mathcal{D}} \Big(a_l(\mathbf{x};\theta) - \mu_l(\mathcal{D};\theta)\Big)^2, \tag{2}$$

where $N_\mathcal{D}$ is the number of images in $\mathcal{D}$, *i.e.* the dataset over which the estimates are computed. For each selected layer l, we pre-compute the activation statistics of the training set $\mathcal{S}$,

$$\hat{\mu}_l = \mu_l(\mathcal{S};\theta^*) \qquad \text{and} \qquad \hat{\sigma}_l^2 = \sigma_l^2(\mathcal{S};\theta^*). \tag{3}$$

During adaptation, we compute the same statistics for each batch $\mathcal{B}$ of the test set, and minimize

$$L_l(\mathcal{B};\theta) = |\mu_l(\mathcal{B};\theta) - \hat{\mu}_l| + \left|\sigma_l^2(\mathcal{B};\theta) - \hat{\sigma}_l^2\right|, \tag{4}$$

where $|\cdot|$ denotes the element-wise L1 norm.

Aligning multiple layers. We found that aligning just the statistics of the final layer, following the common practice [20, 25], is not enough for fast and effective distribution alignment. We show this in an ablation study in Section 4.5. We therefore define a set $\mathcal{L}$ of multiple layers for alignment and the corresponding alignment objective as

$$L(\mathcal{B};\theta) = \sum_{l\in\mathcal{L}} L_l(\mathcal{B};\theta). \tag{5}$$

Intuitively, the aligned layers should be distributed evenly across the network. In the case of convolutional architectures with the standard convolution–normalization–ReLU sequences, we found that it is best to align the features obtained after normalization and before the nonlinearities.

Updating the complete parameter set. Finally, while numerous existing methods only update affine parameters or statistics of the batch normalization layers [20, 27, 30, 34], we update all the parameters of the adapted network. It might seem *a priori* that this increases the risk of making the parameter vector drift away from its initial value, but continuously aligning a very rich set of statistics actually allows the network to recover its initial performance after the distribution of the test data becomes consistent with that of the training set, as shown in Figure 1. The ablation studies in Section 4.5 further confirm that there is no significant performance difference between the two strategies, and thus, we opt for the more deployment-friendly of the two alternatives and make a gradient step on

$$\min_\theta L(\mathcal{B};\theta) \tag{6}$$

for each batch $\mathcal{B}$ of the test data $\mathcal{T}$.

Corruptions:	Gauss	Shot	Impul	Defcs	Gls	Mtn	Zm	Snw	Frst	Fg	Brt	Cnt	Els	Px	Jpg	Mean
Source	28.8	22.9	26.2	9.5	20.6	10.6	9.3	14.2	15.3	17.5	7.6	20.9	14.7	41.3	14.7	18.3
SHOT† (Offline)	13.4	11.6	16.3	7.3	15.9	8.2	7.1	9.4	9.4	10.2	6.3	8.3	12.8	9.8	13.6	10.6
TTT++† (Offline)	12.8	11.1	11.2	7.3	17.1	8.2	6.5	9.4	9.9	7.9	5.0	5.1	13.7	8.8	10.6	9.6
DUA*	15.4	13.4	17.3	8.0	18.0	9.1	7.7	10.8	10.8	12.1	6.6	10.9	13.6	13.0	14.3	12.1 ±0.1
NORM*	15.9	13.7	18.0	7.8	18.3	8.9	8.0	10.8	9.6	12.7	6.1	9.4	13.5	14.3	14.5	12.1 ±0.01
T3A*	15.7	13.9	17.8	7.9	18.2	9.0	8.2	10.9	9.7	12.6	6.1	9.2	13.4	14.2	14.4	12.1 ±0.4
P-L*	15.8	14.1	17.8	7.8	18.1	8.9	8.0	10.8	9.7	12.4	6.1	9.3	13.4	14.1	14.5	12.0 ±0.2
CFA†	15.8	13.8	17.9	7.8	18.2	9.0	8.1	10.7	9.6	12.4	6.1	9.3	13.5	13.7	14.5	12.0 ±0.01
BUFR†	18.5	16.3	22.6	9.0	21.8	10.4	9.7	12.7	13.4	15.2	7.5	12.0	16.3	15.1	17.5	14.5 ±0.01
TTT++† (Online)	15.5	14.1	23.6	9.1	25.1	11.4	8.1	13.2	13.1	13.4	6.6	**6.9**	17.6	12.5	**13.6**	13.6 ±0.03
SHOT† (Online)	14.5	12.3	17.7	7.8	17.8	8.7	7.9	10.4	9.6	12.1	6.1	9.0	13.4	11.4	14.4	11.5 ±0.02
TENT*	14.5	12.4	17.7	7.7	17.7	8.8	7.9	10.3	9.6	12.0	6.1	9.0	13.4	11.3	14.5	11.5 ±0.02
ActMAD†	**13.0**	**11.2**	**15.1**	**7.4**	**15.9**	**8.3**	**7.1**	**9.5**	**9.3**	**10.6**	**5.9**	8.4	**12.3**	**9.3**	**13.6**	**10.4** ±0.06

Table 1. Top-1 Classification Error (%) for all corruptions in CIFAR-10C (level 5). Lower is better. All results are for a WRN-40-2 backbone. *Source* denotes the performance on the corrupted test data without any adaptation. For ease of readability, we highlight the lowest error in bold and the second best as underlined.

Corruptions:	Gauss	Shot	Impul	Defcs	Gls	Mtn	Zm	Snw	Frst	Fg	Brt	Cnt	Els	Px	Jpg	Mean
Source	65.7	60.1	59.1	32.0	51.0	33.6	32.4	41.4	45.2	51.4	31.6	55.5	40.3	59.7	42.4	46.7
SHOT† (Offline)	37.2	36.2	36.7	27.5	38.2	28.5	27.8	31.8	32.0	33.4	25.8	29.6	34.5	29.8	37.2	32.4
TTT++† (Offline)	40.7	36.4	41.5	27.5	47.8	31.1	25.1	36.5	34.7	33.7	23.3	24.7	40.2	30.5	33.3	33.8
DUA*	42.2	40.9	41.0	30.5	44.8	32.2	29.9	38.9	37.2	43.6	29.5	39.2	39.0	35.3	41.2	37.7 ±0.3
NORM*	42.5	41.8	42.6	29.7	43.9	30.6	29.7	35.7	34.6	42.2	26.9	32.8	38.1	35.5	40.9	36.5 ±0.03
T3A*	42.4	41.8	42.5	29.7	44.3	30.5	29.5	35.9	34.5	42.1	26.8	32.8	38.0	35.9	40.7	36.5 ±0.02
P-L*	41.3	40.5	42.5	29.6	43.1	30.3	29.4	35.8	34.3	41.7	26.7	32.4	37.8	33.5	40.8	36.0 ±0.05
CFA†	40.4	39.3	42.1	29.4	42.3	30.2	29.2	35.1	34.1	39.8	26.7	32.1	37.6	32.8	40.6	35.5 ±0.01
BUFR†	44.5	44.3	47.5	32.4	45.9	33.0	33.1	38.7	38.2	45.7	30.1	36.1	40.7	37.1	44.3	39.4 ±0.03
TTT++† (Online)	43.9	40.0	56.3	32.5	54.2	35.9	29.9	42.2	39.4	**39.7**	27.5	**29.6**	44.2	37.0	**37.4**	39.3 ±0.04
SHOT† (Online)	39.7	38.9	42.1	**29.0**	41.9	30.2	29.3	34.8	34.2	**39.7**	26.7	32.2	37.2	32.5	40.4	35.3 ±0.1
TENT*	39.9	39.1	42.2	**29.0**	42.0	30.2	29.3	34.9	34.2	**39.7**	26.7	32.3	37.4	32.4	40.4	35.3 ±0.03
ActMAD†	**39.6**	**38.4**	**39.5**	29.1	**41.5**	**30.0**	**29.1**	**34.0**	**33.2**	40.2	**26.4**	31.5	**36.4**	**31.4**	38.9	**34.6** ±0.01

Table 2. Top-1 Classification Error (%) for all corruptions in CIFAR-100C (level 5). Lower is better. The results were obtained by adapting a WRN-40-2 backbone, trained on CIFAR100, to CIFAR-100C.

Methods:	Source	TENT*	SHOT†	CFA†	ActMAD†
CIFAR-10C	14.6±0.2	10.9±0.2	8.9±0.1	8.4±0.01	**7.7**±0.1
CIFAR-100C	35.1±0.3	27.4±0.2	25.6±0.1	24.6±0.03	**22.4**±0.1

Table 3. Mean Top-1 Classification Error (%) over all corruptions in CIFAR-10/100C (level 5). Lower is better. All results are for a ViT-B/16 backbone. ActMAD results are averaged over 10 runs.

Methods:	Source	NORM*	TENT*	CoTTA*	ActMAD†
CIFAR-10C	18.3	13.8±0.4	12.1±0.7	11.5±0.1	**11.4**±0.1
CIFAR-100C	46.7	42.6±0.8	38.2±0.7	**35.7**±0.4	35.9±0.5

Table 4. Continuous Adaptation (following CoTTA [35]) to distribution shifts at test-time. We report the mean error of 10 runs, while shuffling the order of the corruptions for each run.

4. Experimental Evaluation

We first summarize the datasets and baselines used for the evaluation, and then discuss our results.

4.1. Datasets

We ran experiments on the following data sets:

- *CIFAR-10C and CIFAR-100C* [18] were created by corrupting test images of the CIFAR-10 and CIFAR-100 datasets [22]. We used them to benchmark the performance of ActMAD in image classification, and for ablation studies. Following the common protocol [25, 27, 30, 32, 34], we used 15 corruption types and 5 corruption levels.

- *ImageNet-C* [18] contains ImageNet [4] test images, corrupted in the same way as those of CIFAR-10/100C. We

*Fully Test-Time Adaptation approaches

†Approaches requiring some supervision from the training data

Corruptions:	Gauss	Shot	Impul	Defcs	Gls	Mtn	Zm	Snw	Frst	Fg	Brt	Cnt	Els	Px	Jpg	Mean
Source	98.4	97.7	98.4	90.6	92.5	89.8	81.8	89.5	85.0	86.3	51.1	97.2	85.3	76.9	71.7	86.2
SHOT† (Offline)	73.8	70.5	72.2	79.2	80.6	58.5	54.0	53.6	63.0	47.3	39.2	97.7	48.7	46.1	53.0	62.5
TTT†	96.9	95.5	96.5	89.9	93.2	86.5	81.5	82.9	82.1	80.0	53.0	85.6	79.1	77.2	74.7	83.6
DUA*	89.4	87.6	88.1	88.0	88.6	84.7	74.3	77.8	78.4	68.6	45.6	95.9	72.2	66.5	67.4	78.2 ±0.7
NORM*	87.1	90.6	89.5	87.6	93.4	80.0	71.9	70.6	81.5	65.9	46.8	89.8	73.5	63.2	67.5	77.3 ±0.3
T3A*	85.5	84.0	85.0	86.6	85.9	76.1	65.4	70.3	71.0	58.7	41.3	86.8	60.5	54.4	61.0	71.5 ±0.5
SHOT† (Online)	83.9	82.3	83.7	83.9	83.8	72.6	61.9	65.7	68.6	54.8	39.4	85.9	58.1	53.1	62.3	69.3 ±0.03
P-L*	82.0	79.7	81.5	84.2	83.0	71.0	60.7	65.4	68.6	52.9	41.7	82.6	55.5	51.1	55.7	67.7 ±0.3
TENT*	80.8	78.6	80.4	82.5	82.5	72.1	60.5	63.7	66.7	52.1	**39.2**	84.2	55.5	50.8	58.2	67.2 ±0.02
CFA†	78.2	**76.4**	78.2	81.9	80.4	**69.6**	**60.1**	**63.4**	67.6	**52.0**	41.5	79.5	**54.3**	**50.2**	**55.1**	**65.9** ±0.01
ActMAD†	**76.3**	77.4	**77.4**	**76.1**	**75.4**	72.0	62.8	66.6	**65.8**	55.8	40.9	**78.8**	55.7	51.4	57.6	66.0 ±0.02

Table 5. Top-1 Classification Error (%) for all corruptions in ImageNet-C (level 5). Lower is better. All results are for a ResNet-18 network pre-trained on the clean train set. *Source* denotes its performance on the corrupted test data without any adaptation.

used them for large-scale classification experiments. Again, we tested for 15 corruption types and 5 severity levels.

- We used the *KITTI* dataset [12] to evaluate the performance of ActMAD on object detection for automotive applications. KITTI contains 8 classes of traffic participants, and was recorded mostly in clear weather. To simulate degrading weather conditions on the KITTI dataset, we used physics-based rendering of fog and rain [15], and followed Mirza et al. [26] to simulate snow.

For KITTI, the annotations for the test set are not publicly available, so we divided the public train set randomly into train and test split with equal proportion. These splits and our codebase is available at this repository: https://github.com/jmiemirza/ActMAD.

4.2. Baselines

We compared ActMAD against the following approaches:

- *Source* is the model trained on source data and evaluated on test data without any adaptation.
- *TTT* [32] adapts the network to test data using self-supervision from a rotation prediction task.
- *TTT++* [25] employs a combination of self-supervised contrastive learning and feature alignment.
- *P-L* is our adaptation of the pseudo-labeling approach [23] to test-time training.
- *CFA* [20] aligns class-conditioned feature statistics.
- *BUFR* [7] relies on aligning a lightweight approximation of train and test data statistics.
- *T3A* [19] uses pseudo-prototypes to classify test data.
- *SHOT* [24] minimizes prediction entropy while maximizing class entropy over the test data.
- *NORM* [30] adapts the statistics of batch normalization layers to test data.
- *DUA* [27] is another approach to adapt the batch normalization statistics online.
- *TENT* [34] learns channel-wise affine transformations by minimizing the entropy of the predictions.

ActMAD experiments are run for 10 random seeds, while the online baselines are run for 3 random seeds (except for TTT [32] for ImageNet [4], since it requires retraining). We report the mean and standard deviations for these runs.

4.3. Image Classification

To evaluate ActMAD on image classification, we adapted a Wide-ResNet-40-2 [37] (WRN-40-2) from the Robust Bench [18] and a Vision Transformer [5] (ViT-B/16) backbone on the CIFAR-C datasets. For Imagenet-C dataset, we used an ImageNet pre-trained ResNet-18 [17] from the PyTorch model zoo, following previous work [27]. Results on ImageNet-C with other backbones are provided in the supplemental. We used a batch size of 128 for all image classification experiments. ActMAD was run with the learning rate of $1e{-}2$ for CIFAR-10C, $1e{-}3$ for CIFAR-100C and $25e{-}5$ for ImageNet-C, while the baselines used hyperparameters reported by their authors. Following prior literature [19, 27, 34], we performed the experiments online, using each test image to adapt the network only once. Since SHOT and TTT++ are better suited for the offline scenario, we additionally run them for 500 epochs on the entire test sets of the CIFAR-C data sets, with an early stopping criterion, according to the official TTT++ implementation†.

Results for CIFAR-10/100C for WRN-40-2 are reported in Table 1 and 2, while the ViT results are listed in Table 3. ActMAD outperforms other online methods in classifying CIFAR-10C and CIFAR-100C images by a fair margin, establishing a new state-of-the-art for both Transformer-based and convolutional architectures. When adapting the wide ResNet to CIFAR-10C, there is almost no gap between ActMAD and the offline approaches, while on CIFAR-100C offline adaptation fares better. For the ViT, we also achieved a new state-of-the art on CIFAR-10C and CIFAR-100C, outperforming even CFA [20], designed specifically for robustifying vision transformers.

† https://github.com/vita-epfl/ttt-plus-plus, commit: e429a194

(a) KITTI-Clear → KITTI-Fog

	car	van	truck	ped	sit	cyc	tram	misc	Mean
Source	31.3	15.0	6.0	34.8	33.6	20.2	6.7	9.1	19.6
TTT†	42.6	19.5	10.5	49.7	51.4	31.0	10.5	20.2	29.4 ±0.5
DUA*	51.4	13.5	9.1	48.1	57.3	36.3	14.5	18.0	31.0 ±2.1
NORM*	50.1	27.6	12.6	47.6	50.0	30.9	17.7	21.7	32.3 ±0.1
ActMAD†	**67.0**	**41.2**	**25.5**	**62.2**	**68.7**	**50.9**	**30.5**	**35.7**	**47.7** ±1.01

(b) KITTI-Clear → KITTI-Rain

Source	86.4	69.6	58.6	68.6	63.7	60.2	64.5	60.4	66.5
TTT†	86.4	76.1	68.0	68.7	66.6	66.3	75.0	65.2	71.5 ±0.4
DUA*	88.3	70.4	70.4	70.8	**67.7**	66.8	73.5	67.5	71.9 ±1.3
NORM*	88.3	77.0	65.7	69.1	61.5	66.7	69.1	67.1	70.6 ±0.1
ActMAD†	**94.2**	**89.2**	**87.3**	**74.1**	65.6	**77.9**	**82.5**	**80.1**	**81.4** ±0.2

(c) KITTI-Clear → KITTI-Snow

Source	54.8	27.8	31.7	35.7	1.3	15.7	18.2	13.3	24.8
TTT†	77.2	53.2	60.6	48.4	29.7	37.1	43.2	31.1	47.5 ±0.7
DUA*	64.6	38.9	49.3	44.0	20.8	22.8	27.8	25.4	36.7 ±0.9
NORM*	75.5	51.0	51.7	46.8	21.7	34.9	43.4	34.3	44.9 ±0.2
ActMAD†	**89.5**	**78.0**	**82.6**	**57.8**	**38.0**	**53.4**	**58.8**	**58.1**	**64.5** ±0.03

Table 6. Mean Average Precision (mAP@50) for a KITTI pre-trained YOLOv3 tested on rain, fog and snow datasets. Higher is better. a) Results for the most severe fog level, *i.e.* only 30m visibility. b) Results for the most severe rain level, *i.e.* 200mm/hr rain intensity. c) Results for snow. Best mAP is shown in bold, while the second best is underlined.

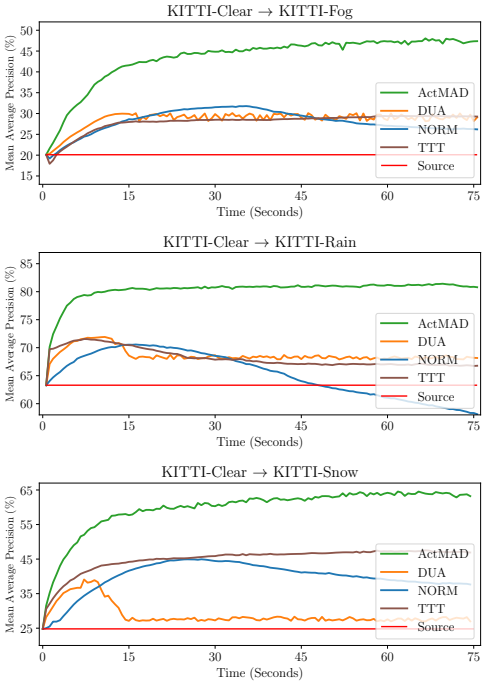

Figure 3. Online adaptation for each individual weather condition and comparison with baselines. We again report the Mean Average Precision (mAP@50) averaged over all the 8 classes in the KITTI dataset.

The results for the ImageNet-C dataset are shown in Table 5. ActMAD comes in second after CFA [20], but the performance gap in classification error is only 0.1 percentpoint. We suspect the advantage of CFA to stem from its alignment of class-conditioned statistics, which may become more important for the rich and diverse set of ImageNet classes than for the limited sets of classes of the CIFAR datasets. However, the use of class-conditional statistics limits CFA to the classification task. Additionally, ActMAD can also be combined with the entropy based methods to obtain further gains, these results and comparison with EATA [28] are provided in the supplementary.

Continuous adaptation to pertubations. Wang *et al*. evaluated their TTT method CoTTA [35] by continuously adapting to different corruptions, instead of adapting to only a single corruption at a time. We tested ActMAD and other baselines in such a scenario on CIFAR-10/100C. The results, listed in Table 4, show that ActMAD performs on par with CoTTA, and outperforms other baselines.

4.4. Object Detection

For object detection experiments, we adapted a KITTI-pretrained YOLOv3 [29] to degrading weather conditions, such as fog, rain, and snow. We used a learning rate of 1e−4, and a batch size of 30. We compared our object detection results with DUA, NORM, and TTT. DUA and NORM could be applied to detector adaptation without any modifications, thanks to the fact that they do not rely on the network to produce normalized probabilities and are not limited to adapting models trained in a multi-task setting. By contrast, to deploy TTT, we had to re-train the detector with the auxiliary, self-supervised rotation prediction task. We describe the implementation details in the supplemental. The results, reported in Table 6, show that ActMAD outperforms the baselines by a significant margin. Additionally, in Figure 3 we plot the performance of the network during adaptation, showing that ActMAD stabilizes its precision as adaptation progresses, while the baselines provoke performance fluctuations. Needless to say, consistent level of performance is of utmost importance in safety-critical applications, like autonomous driving. Further, for comparison with MemCLR [33] and results of ActMAD in other realistic scenarios, we refer to the supplementary material.

4.5. Ablation Studies

Effect of batch size. The number of samples used by ActMAD to compute the location-aware statistics is equal to the number of batch elements, and therefore much lower than the number of samples used to compute the channel-wise statistics in competing methods. Statistics computed on

				CIFAR-10C		KITTI-FOG	
				% Error ↓	Change	mAP@50 ↑	Change
Source (no adaptation)				18.3		19.6	
Full ActMAD				10.4	0	47.7	0
Replace	*multi-layer alignment*	by	*last layer alignment*	12.3	+1.9	36.0	-11.7
Replace	*pixel statistics*	by	*channel averaged statistics*	11.5	+1.1	38.6	- 9.1
Replace	*mean and variance alignment*	by	*central moment difference*	10.5	+0.1	41.2	- 6.5
Replace	*full parameter update*	by	*only affine parameter update*	10.6	+0.2	45.2	- 2.5

Table 7. Ablation study on design choices of ActMAD. We use a Wide-ResNet-40-2 for CIFAR-10C, and report the mean performance over 10 experiments for each of the 15 corruptions. We use YOLOv3 for the KITTI-FOG experiment.

BS	125	100	75	50	25	10	*Source Only*
Error ↓ (%)	66.0	66.4	68.9	70.5	71.7	72.8	86.2

Table 8. Top-1 Error (%) averaged over 15 corruptions in the ImageNet-C dataset while adapting an ImageNet pre-trained ResNet-18 with different batch sizes (BS).

small samples have higher variance, which could adversely affect adaptation performance. We investigate how decreasing the batch size affects the performance of ActMAD. Like in the image classification experiments, we use an ImageNet pre-trained ResNet-18 from the PyTorch model zoo. We decrease the learning rate in proportion to the decreasing batch size, following Goyal *et al*. [14]. As shown in Table 8, limiting the batch size decreases the performance gently, and even batches as small as 10 images still let us achieve 13.4 percent-point improvement over the un-adapted model.

Design alternatives. To verify how our design choices contribute to the performance of ActMAD, we removed each of its key features and evaluated the resulting algorithms on image classification and object detection. We adapted a wide ResNet from CIFAR-10 to CIFAR-10C, and YOLOv3 from KITTI to KITTI-Fog, in the same setup as in previous experiments. As can be seen in Table 7, a large performance drop was provoked by limiting the alignment to the last feature map, as opposed to multiple feature maps throughout the network, and by aligning channel, instead of pixel, statistics. This attests to the importance of fine-grained supervision, distributed over multiple layers and locations in the feature maps. Unsurprisingly, the use of location-aware statistics results in higher performance gain for KITTI ($\sim$ 10 percent-point) than for CIFAR ($\sim$ 1 percent-point). This is expected, since road scenes of KITTI are much more structured than the CIFAR images. We also hypothesize that ActMAD's localized feature alignment benefits object detection more, because detection relies on local features to a higher degree than the location-agnostic image classification. The type of statistics used for feature alignment seems to make no difference for CIFAR-10C, but aligning means and variances gives better results than aligning higher-order central moments for the KITTI object detector. This difference might stem from the fact that the KITTI experiments use almost ten times smaller batch size, and that low batch size affects higher-order moment estimates more than estimates of means and variances. Finally, limiting the parameter update to channel-wise affine transformations results in a small performance drop in object detection, while almost not affecting the image classification accuracy. However, since we observed no drift in the continuous adaptation experiment and limiting the update might require manually inserting affine layers to the network, we prefer to perform the full update.

5. Conclusion

We propose a new method – ActMAD – for online test-time-training based on location-aware feature alignment. In contrast to many previous TTT approaches, our ActMAD is task- and architecture-agnostic. It goes beyond image classification and can be readily applied to object detection. ActMAD outperforms existing approaches by a considerable margin, while being more stable and adapting faster to new distributions. The power of ActMAD stems from fine-grained supervision which results from individually aligning each pixel in multiple feature maps across the network. Experiments show that the fine-grained supervision provided by ActMAD from location-aware feature alignment is especially helpful for tasks which require a holistic understanding of the scene, for example object detection.

Acknowledgment

We gratefully acknowledge the financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association. This work was also partially funded by the FWF Austrain Science Fund Lise Meitner grant (M3374) and Austrian Research Promotion Agency (FFG) under the projects HighScene (884306) and SAFER (894164).

References

[1] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free Online Test-time Adaptation. In *Proc. CVPR*, 2022. 1, 3

[2] Chao Chen, Zhihang Fu, Zhihong Chen, Sheng Jin, Zhaowei Cheng, Xinyu Jin, and Xian-Sheng Hua. HoMM: Higher-order Moment Matching for Unsupervised Domain Adaptation. In *Proc. AAAI*, 2020. 2, 4

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proc. ICML*, 2020. 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-scale Hierarchical Image Database. In *Proc. CVPR*, 2009. 5, 6

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. ICLR*, 2020. 6

[6] Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Thomas L Griffiths, and Alexei A Efros. Investigating Human Priors for Playing Video Games. In *Proc. ICML*, 2018. 1

[7] Cian Eastwood, Ian Mason, Christopher KI Williams, and Bernhard Schölkopf. Source-free Adaptation to Measurement Shift via Bottom-up Feature Restoration. In *Proc. ICLR*, 2021. 2, 3, 6

[8] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021. 1

[9] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542:115–118, 2017. 1

[10] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. Test-time Training with Masked Autoencoders. *NeurIPS*, 2022. 1, 2

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *JMLR*, 17(59):1–35, 2016. 4

[12] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets Robotics: The KITTI Dataset. *IJR*, 32(11):1231–1237, 2013. 1, 6

[13] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. In *Proc. ICLR*, 2018. 2

[14] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv*, 2017. 8

[15] Shirsendu Sukanta Halder, Jean-François Lalonde, and Raoul de Charette. Physics-based Rendering for Improving Robustness to Rain. In *Proc. ICCV*, 2019. 1, 6

[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders are Scalable Vision Learners. In *Proc. CVPR*, 2022. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proc. CVPR*, 2016. 6

[18] Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *Proc. ICLR*, 2019. 1, 5, 6

[19] Yusuke Iwasawa and Yutaka Matsuo. Test-Time Classifier Adjustment Module for Model-Agnostic Domain Generalization. *NeurIPS*, 2021. 2, 3, 4, 6

[20] Takeshi Kojima, Yutaka Matsuo, and Yusuke Iwasawa. Robustifying Vision Transformer without Retraining from Scratch by Test-Time Class-Conditional Feature Alignment. *Proc. IJCAI*, 2022. 1, 2, 3, 4, 6, 7

[21] Klemen Kotar and Roozbeh Mottaghi. Interactron: Embodied adaptive object detection. In *Proc. CVPR*, 2022. 3

[22] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. Technical report, Department of Computer Science, University of Toronto, 2009. 5

[23] Dong-Hyun Lee et al. Pseudo-label: The Simple and Efficient Semi-supervised Learning Method for Deep Neural Networks. In *Proc. ICMLW*, 2013. 6

[24] J. Liang et al. Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation. In *ICML*, 2020. 2, 3, 4, 6

[25] 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 *NeurIPS*, 2021. 1, 2, 4, 5, 6

[26] M. Jehanzeb Mirza, Marc Masana, Horst Possegger, and Horst Bischof. An Efficient Domain-Incremental Learning Approach to Drive in All Weather Conditions. In *Proc. CVPRW*, 2022. 6

[27] M Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The Norm Must Go On: Dynamic Unsupervised Domain Adaptation by Normalization. In *Proc. CVPR*, 2022. 1, 2, 3, 4, 5, 6

[28] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient Test-Time Model Adaptation without Forgetting. In *Proc. ICML*, 2022. 3, 7

[29] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv*, 2018. 1, 7

[30] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving Robustness Against Common Corruptions by Covariate Shift Adaptation. In *NeurIPS*, 2020. 2, 3, 4, 5, 6

[31] Baochen Sun and Kate Saenko. Deep CORAL: Correlation Alignment for Deep Domain Adaptation. In *Proc. ECCVW*, 2016. 2

[32] 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 *Proc. ICML*, 2020. 1, 2, 4, 5, 6

[33] Vibashan VS, Poojan Oza, and Vishal M Patel. Towards Online Domain Adaptive Object Detection. In *Proc. WACV*, 2023. 3, 7

[34] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully Test-time Adaptation by Entropy Minimization. In *Proc. ICLR*, 2020. 1, 2, 3, 4, 5, 6

[35] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual Test-Time Domain Adaptation. In *Proc. CVPR*, 2022. 2, 4, 5, 7

[36] Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre Alvise-Rebuffi, Ira Ktena, Taylan Cemgil, et al. A Fine-Grained Analysis on Distribution Shift. In *Proc. ICLR*, 2022. 1

[37] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *Proc. BMVC*, 2016. 6

[38] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning. In *Proc. ICLR*, 2017. 2, 4