JUMPSTYLE: A FRAMEWORK FOR DATA-EFFICIENT ONLINE ADAPTATION

Aakash Singh, Manogna Sreenivas, Soma Biswas
Image Analysis and Computer Vision Lab
Indian Institute of Science
Bangalore, India
{aakashsingh, manognas, somabiswas}@iisc.ac.in

ABSTRACT

Research in deep learning is restrictive in developing countries due to a lack of computational resources, quality training data, and expert knowledge, which negatively impacts the performance of deep networks. Moreover, these models are prone to suffer from distribution shift during testing. To address these challenges, this paper presents a novel approach for fine-tuning deep networks in a Domain Generalization setting. The proposed framework, JumpStyle, comprises two key components: (1) an innovative initialization technique that jumpstarts the adaptation process, and (2) the use of style-aware augmentation with pseudo-labeling, in conjunction with a simple and effective test-time adaptation baseline named Tent. Importantly, JumpStyle only requires access to a pre-trained model and is not limited by the training method. The effectiveness of this approach is extensively evaluated through experiments.

1 INTRODUCTION

Research in deep learning has achieved remarkable progress in solving several computer vision tasks like image classification, object detection, segmentation Deng et al. (2009); Lin et al. (2014); Everingham et al. (2010); Chen et al. (2017); He et al. (2017); Ren et al. (2015). However, their performance usually drops significantly when data from previously unseen domains are encountered during testing, which is quite common in real scenarios. To overcome this, there has been a considerable amount of research interest in areas like Unsupervised Domain Adaptation (UDA), Domain Generalization (DG), etc. UDA setting assumes access to labeled source and unlabeled target domain data during training. On the other hand, the objective of DG is to use multiple source domains to learn domain invariant representations, thus preparing the model for future deployment. But, none of these approaches leverage the rich information inherently present in the test data.

This recently opened up a new avenue of research, namely Test-Time Adaptation (TTA) designed to leverage the test data to adapt any off-the-shelf model, hence reducing the adverse effect of distribution shift. TTA can be used to improve the performance of deep learning models in healthcare, autonomous driving, agriculture etc. One such application is to use an object detection model trained using European and American road data to adapt in Indian roads. The ability of the this framework to improve the performance of deep networks in situations where there is limited access to quality training data or computational resources makes it a valuable tool for various industries and applications in developing countries.

Here, we specifically focus on the approaches which can adapt any off-the-shelf model using the unlabeled test data in an online fashion. Very recently, a classifier adjustment framework Iwasawa & Matsuo (2021) was proposed for test-time adaptation in DG setup, which reports impressive performance for several backbones. Inspired by this work, here, we present a complementary analysis, where we analyze if different backbones trained using simple Empirical Risk Minimization (ERM) or even state-of-the-art DG approaches Zhou et al. (2021) specialized for generalization can further benefit from TTA using unlabelled test data.

Towards this goal, we propose **Jump-Style** framework that builds upon the state-of-the-art TTA method Tent Wang et al. (2021) and suitably adapt it for the DG application.

2 Problem Statement

Domain Generalization (DG): The objective here is to use the given multiple labeled source domains, $\mathcal{D}_{train} = D_1 \cup D_2 \ldots \cup D_{d_{tr}}$, to learn a model $F_{\boldsymbol{\theta}}$ using $\mathcal{D}_{train}$, such that it can generalize well to an unseen test domain $D_{test} \notin \mathcal{D}_{train}$.

Testing phase: In general, the trained model $F_{\boldsymbol{\theta}}$ is directly used for testing. However, it is possible to further improve its performance by using online data available in batches to perform TTA of $F_{\boldsymbol{\theta}}$.

3 Method

The JumpStyle framework, we propose for TTA in DG setting has two primary components: (1) Effective Initialization (Jump Start), and (2) Consistent style-aware augmentations for pseudo-labeling. We first briefly describe the TTA method Tent Wang et al. (2021) which we use as a baseline here.

Entropy Based Tent Framework: Tent is a fully test-time adaptation method designed to adapt any given off-the-shelf model using only the available test data. In general, during training, the BN layers estimate the channel-wise statistics μ, σ of the feature maps using the training data. While these statistics are relevant when the test samples are drawn from the same distribution as the training data, they are not optimal when there is a distribution shift during testing. In Tent, instead of the source data statistics $\{\mu_s, \sigma_s\}$, the test data statistics $\{\mu_t, \sigma_t\}$ are used. Further, the BN affine parameters $\{\gamma, \beta\}$ are finetuned to minimize the test prediction entropy (defined later).

We now describe the two proposed modules that we integrate with the Entropy-based Tent framework for this application. Specifically, given a trained model $F_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$, our objective is to utilize the target samples x_t of batch size n, to adapt the model.

1) Jump start initialization of the BN parameters: Tent accounts for covariate shift by replacing the training batch normalization (BN) statistics with the statistics of the test data. However, the number of target samples available in online testing is usually limited, which may not be a good representation of the entire target distribution. To address this, we propose a simple, yet effective way to correct the test batch statistics by using the training domain statistics as a prior (Schneider et al., 2020). Also, as the quality of the estimated test domain statistics depends on the test batch size n, we combine the source and target statistics as follows

$$\begin{aligned}\bar{\mu} &= \alpha(n)\mu_s + (1-\alpha(n))\mu_t \\ \bar{\sigma^2} &= \alpha(n)\sigma_s^2 + (1-\alpha(n))\sigma_t^2\end{aligned} \tag{1}$$

where μ_t and σ_t^2 are the online estimated test batch statistics, μ_s and σ_s^2 are the source data statistics available as part of the given trained model.

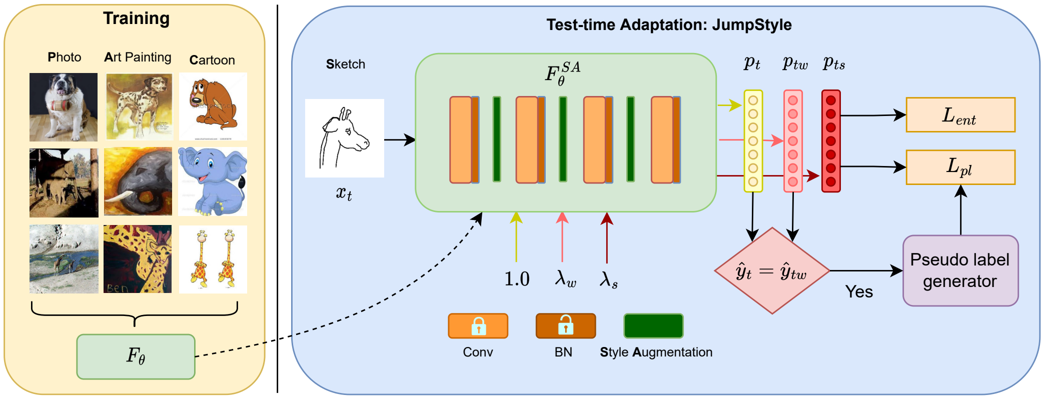
The weight $\alpha(n)$ is a function of batch size n and has a significant effect on the final performance. In Schneider et al. (2020), a method was proposed to compute this weight based on the batch size n and an additional hyper-parameter. Since the weight should ideally be a function of only the batch-size, in this work, we design $\alpha(n)$ to be:

$$\alpha(n) = 0.5(1 + e^{-\kappa n}); \quad \text{where } \kappa = 0.05 \tag{2}$$

The weight is designed such that it satisfies the following criteria: As the number of samples in the batch n decreases, the weight for the source statistics $\alpha(n)$ should increase. In the extreme case, when $n = 0$, $\alpha(n) = 1$. But when $n > 0$, since the number of test samples available is still limited, the smallest value of $\alpha(n)$ is constrained to not fall below 0.5. The value of κ is obtained empirically, but the proposed weighting rule has the advantage that it only depends on the batch-size as desired. This weighting is used for all the experiments reported in this work. In addition to this initialization, after the data from a test-batch is passed through the model, its style-aware weak and strong augmentations are used to further update the BN affine parameters.

2) Pseudo-labeling based on consistency of style-augmented targets: Performing pseudo supervision using pseudo labels for samples with consistent predictions across augmented versions of unlabelled data has shown remarkable success in semi-supervised learning (SSL) Sohn et al. (2020); Berthelot et al. (2019). Here, we explore whether such techniques aid TTA in DG scenario, which to the best of our knowledge, has not been explored before. More specifically, we propose to check consistency of style-augmented target samples, which is more suited for the DG task.

Figure 1: DG training (left) using Photo, Art-painting, Cartoon as source domains. TTA using JumpStyle (right) on test sample x_t from test domain *Sketch*. Consistency across predictions of true sample p_t and weak style augmentation p_{tw} are used to pseudo label x_t. BN affine parameters are updated to minimize the pseudo label and entropy loss.



During testing, given two target samples x_i and x_j, we create an augmented feature of x_i using the style of x_j. Let f_i and f_j denote their respective feature maps at a certain layer. The channel-wise means $\mu(f_i), \mu(f_j)$ and standard deviations $\sigma(f_i), \sigma(f_j)$ are representative of image styles. In this layer, these feature statistics are mixed, thereby generating a pseudo-style, which is then applied to the style normalized feature of f_i to obtain style augmented feature f_i^{SA} Zhou et al. (2021)

$$\begin{aligned} \mu_{mix}(f_i;\lambda) &= \lambda\mu(f_i) + (1-\lambda)\mu(f_j) \\ \sigma_{mix}(f_i;\lambda) &= \lambda\sigma(f_i) + (1-\lambda)\sigma(f_j) \\ f_i^{SA} &= \sigma_{mix}(f_i;\lambda) * \frac{f_i - \mu(f_i)}{\sigma(f_i)} + \mu_{mix}(f_i;\lambda) \end{aligned} \tag{3}$$

where $\lambda \in [0,1]$ is the mixing coefficient. Features thus obtained preserve the semantic content of the input x_i, while only the style is perturbed using that of the other image.

Inspired by Sohn et al. (2020), we compute two types of style augmentations for each target sample, namely weak style augmentation and strong style augmentation as described next. Let $F_{\boldsymbol{\theta}}^{SA}(;\lambda)$ denote the entire model including the feature extractor, classifier and the softmax layer with the style augmentations. Setting the mixing coefficient $\lambda = 1$ reduces the model $F_{\boldsymbol{\theta}}^{SA}(;\lambda)$ to the original backbone $F_{\boldsymbol{\theta}}$. Given a test batch x_t, the samples are randomly permuted within the batch to obtain $\tilde{x}$. The features of x_t are perturbed by instance-wise mixing of styles from features of $\tilde{x}$ as described in eqn. (3). For a sample x_t, we denote its prediction as p_t, and those of its weak and strong augmentations as p_{tw} and p_{ts} respectively. These are obtained as follows

$$p_t = F_{\boldsymbol{\theta}}^{SA}(x_t;1); \quad p_{tw} = F_{\boldsymbol{\theta}}^{SA}(x_t;\lambda_w); \quad p_{ts} = F_{\boldsymbol{\theta}}^{SA}(x_t;\lambda_s) \tag{4}$$

To better utilise the target samples during test-time, we generate pseudo labels for the samples whose predictions are confident and robust against weak domain shifts. The pseudo labels for the test sample and its weak augmentation are obtained as $\hat{y}_t = argmax(p_t)$ and $\hat{y}_{tw} = argmax(p_{tw})$ respectively. The pseudo label loss is then computed as

$$\mathcal{L}_{pl} = \mathbb{E}_{x_t \in \mathcal{S}}[-\log p_{ts}(\hat{y}_t)]; \qquad \mathcal{S} = \{x_t | \hat{y}_t = \hat{y}_{tw}; max(p_t) > \tau\} \tag{5}$$

Inspired from Tent (Wang et al., 2021), we also use entropy loss to enforce confident predictions. In this work, we define this only for the strong style augmentations as follows:

$$\mathcal{L}_{ent} = -\frac{1}{n}\sum_{t=1}^{n}\sum_{c} p_{ts}(c)\log p_{ts}(c) \tag{6}$$

where c denotes the class index and n is the test batch size.

Method	**VLCS**	**PACS**	**OfficeHome**	**Terra**	**Average**
ResNet-50	74.3±0.5	84.1±0.1	66.9±0.2	45.8±1.8	67.8
SHOT-IM	61.5±1.7	84.6±0.3	68.0±0.0	33.8±0.3	62.0
SHOT	61.6±1.8	84.8±0.5	68.0±0.0	34.6±0.3	62.3
PL	63.4±1.8	80.1±3.5	61.3±1.5	36.8±4.4	60.4
PL-C	73.3±0.8	84.7±0.3	66.4±0.3	**47.0±1.7**	67.9
Tent-Full	75.4±0.6	87.0±0.2	66.9±0.2	42.6±0.8	68.0
BN-Norm	71.3±0.4	85.8±0.1	66.4±0.1	42.3±0.4	66.5
Tent-C	72.4±1.5	84.4±0.1	66.2±0.2	42.4±3.1	66.4
Tent-BN	65.6±1.4	84.9±0.0	67.7±0.2	42.7±0.5	65.2
T3A	76.0±0.3	85.1±0.2	68.2±0.1	44.6±0.9	68.5
JumpStyle	**76.9±0.7**	**87.5±0.6**	**69.1±0.5**	44.7±0.7	**69.5**

Table 1: Results with ERM approach using ResNet-50 backbone.

Although inspired from the SSL approach Sohn et al. (2020), there are significant differences between the two approaches as: (i) The weak and strong style augmentations proposed in this work are better suited for the Domain Generalization objective as compared to the standard image augmentations as in Sohn et al. (2020), which we demonstrate in ablation study (Table(4)). (ii) Unlike the semi-supervised approaches, where the whole network is trained/fine-tuned using the pseudo-labelling loss, here only the BN layers are updated.

Final Test-time adaptation loss: The total loss for adaptation during test time is computed as a weighted combination of the pseudo-label loss and the entropy loss. The BN affine parameters, denoted by $\{\boldsymbol{\gamma}, \boldsymbol{\beta}\}$ are updated in an online fashion each time a new batch is available, to minimize the following test time loss:

$$\mathcal{L}_{test} = \eta * \mathcal{L}_{pl} + (1 - \eta) * \mathcal{L}_{ent} \tag{7}$$

The parameter η balances the two losses, and is empirically set to 0.8 for all the experiments.

4 Experimental Evaluation

Here, we describe the experiments done to evaluate the effectiveness of the proposed framework.

Datasets used: We perform experiments on four benchmark DG datasets following the same protocol as in T3A Iwasawa & Matsuo (2021), namely PACS (Li et al., 2017), VLCS (Fang et al., 2013), Office-Home (Venkateswara et al., 2017), Terra-Incognita (Beery et al., 2018). We describe these comprehensively in section A.1

TTA-Baselines: We compare the proposed JumpStyle with the following test-time adaptation baselines: 1) **SHOT-IM** (Liang et al., 2020): updates the feature extractor to minimize entropy and the diversity regularizer; 2) **SHOT** (Liang et al., 2020): uses pseudo-label loss along with information maximization as in (1); 3) **PL (Pseudo labelling)** (Lee, 2013): updates the entire network by minimizing the cross-entropy between the prediction and pseudo labels; 4) **PL-C** (Lee, 2013): minimizes the pseudo-label loss as above and updates only the linear classifier; 5) **Tent-Full** (Wang et al., 2021): is the original method, where the BN statistics and transformations are updated; 6) **BN-Norm** (Schneider et al., 2020): only the BN statistics are updated while keeping the affine parameters fixed; 7) **Tent-C** (Wang et al., 2021): updates only the classifier to reduce the prediction entropy; 8) **Tent-BN** (Wang et al., 2021): adds one BN layer just before the linear classifier and then modulates its affine parameters.

Implementation Details: Following Iwasawa & Matsuo (2021), we split the data in each domain into a training (80%) and validation (20%) split. We follow the leave one out protocol for training and evaluation. In each experiment, three domains act as the source whose training splits are used to train the model, while the validation splits are used to select the learning rate. Further, we perform test-time adaptation on the target domain and report the average accuracy over all the domains in the dataset. The parameters for a TTA framework has to be selected prior to deployment,

Method		**PACS**			**VLCS**	
	Batch size=8	Batch size=32	Batch size=64	Batch size=8	Batch size=32	Batch size=64
MixStyle*	82.4	82.4	82.4	75.7	75.7	75.7
Tent-Full	81.2 ±0.9	85.9±0.7	86.7±0.8	69.2±0.6	71.4±0.4	71.9±0.4
T3A	80.5±0.7	84.9±0.5	85.6±0.5	72.6±0.7	75±0.5	75.5±0.6
JumpStyle	**85.9±0.7**	**86.4±0.6**	**86.8±0.6**	**76.3±0.4**	**76.5±0.3**	**76.1±0.3**

Table 2: Results on PACS and VLCS datasets using MixStyle trained DG model. * denotes results obtained using the official MixStyle Zhou et al. (2021) implementation.

Method	**VOC**	**LabelMe**	**Caltech**	**SUN09**	**Average**
Tent	69.2±0.3	60.6±0.5	91.0±0.8	66.0±0.6	71.7
+Jump	69.6±0.5	66.0±0.4	96.3±0.6	66.6±0.5	74.6
+Jump+FixMatch	69.8±0.5	64.8±0.3	95.8±0.3	67.7±0.4	74.5
+JumpStyle	**71.3±0.4**	**66.5±0.5**	**96.5±0.7**	**68.5±0.7**	**75.7**

Table 3: Ablation study on VLCS dataset using ResNet-18 backbone.

before one has access to test data. Following T3A (Iwasawa & Matsuo, 2021), we set the batch size to 32 and use training domain validation set to tune the hyperparameters for fair comparison. The learning rates used were $1e^{-4}$ for PACS, VLCS, OfficeHome and $1e^{-5}$ for Terra Incognita. We set $\alpha(n)$ to 0.6 which is computed using eqn.(2) for n=32 and set η to 0.8. We set λ_w and λ_s in eqn. (4) to 0.9 and 0.75 respectively. The parameter κ in eqn. (2) is fixed to 0.05 for all the experiments. We describe the selection of hyperparameters in the Appendix A.3

4.1 Results with DG baselines:

(1) Empirical Risk Minimization: First, we test the proposed TTA framework with the ERM approach for DG, where labelled samples from multiple source domains are collectively used to train the network using CE loss. The results of the proposed framework and comparisons with the other TTA approaches using ResNet-50 backbone in Table 1 for the four datasets. We also perform experiments with a light-weight ResNet-18 backbone, which we report in the Appendix A.2 We observe that the proposed JumpStyle outperforms the other approaches for three of the four datasets, and also on an average. This explains the generalization ability of the proposed approach across different datasets and backbones.

(2) Mixstyle: Here, we analyze whether TTA can also benefit from the state-of-the-art DG approaches, which have been designed specifically to obtain domain invariant representations. Since online TTA depends upon the test batch size, here, we also experiment with different batch sizes to analyze its effect on the final performance. We report the results obtained using MixStyle with ResNet-18 backbone and its performance on doing TTA using Tent-Full, T3A and JumpStyle in Table 2. From these results on PACS and VLCS datasets, we observe the following: (1) The performance of Tent-Full and T3A improves significantly for higher batch sizes. However, their performance is not satisfactory for smaller batch sizes.

5 Conclusion

In this paper, we present a novel framework termed *JumpStyle* for test-time adaptation in domain generalization setup. Firstly, we propose an effective scheme to correct the Batch-Normalization statistics based on the number of test samples available online. Further, we propose a test-time consistency regularization method to ensure consistent predictions across perturbed versions of test samples. In specific, we use MixStyle which is a label preserving feature perturbation module to obtain weak and strong augmentations, across which we enforce consistent predictions. Extensive experiments performed using backbones with different representation ability, training methods and augmentations demonstrate the effectiveness of the proposed framework.

REFERENCES

S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In *ECCV*, 2018.

D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *NeurIPS*, 2019.

L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017.

M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky. Exploiting hierarchical context on a large database of object categories. In *CVPR*, 2010.

J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

Chen Fang, Ye Xu, and Daniel N. Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *ICCV*, 2013.

Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 2006.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *ICLR*, 2021.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. *NeurIPS*, 2021.

Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML*, 2013.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *ECCV*, 2014.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.

Bryan Russell, Antonio Torralba, Kevin Murphy, and William Freeman. Labelme: A database and web-based tool for image annotation. *IJCV*, 2008.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *NeurIPS*, 2020.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *NeurIPS*, 2020.

H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017.

D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021.

K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. Domain generalization with mixstyle. In *ICLR*, 2021.

A Appendix

A.1 Dataset Details

PACS (Li et al., 2017) consists of four domains, Photo, Art painting, Cartoon and Sketch, where the domain shift is particularly due to image styles. It has $9,991$ images belonging to 7 classes. **VLCS** (Fang et al., 2013) is a collection of four datasets, Caltech101 (Fei-Fei et al., 2006), LabelMe (Russell et al., 2008), SUN09 (Choi et al., 2010), VOC2007 (Everingham et al., 2010) with $10,729$ samples from 5 classes. **Office-Home** (Venkateswara et al., 2017) consists of four domains, Art, Clipart, Product, Real-world, with $15,500$ images of 65 objects in office and home environments. **Terra-Incognita** (Beery et al., 2018) contains photos of wild animals. Following Gulrajani & Lopez-Paz (2021); Iwasawa & Matsuo (2021), we use the images captured at locations $L100, L46, L43, L38$ as the four domains. This contains 24788 examples of 10 different classes.

A.2 JumpStyle with different backbones

We perform experiments with ResNet-50 and ResNet-18 backbones and observe that Jumpstyle outperforms all the prior test-time adaptation methods.

Backbone	**Method**	**VLCS**	**PACS**	**OfficeHome**	**Terra**	**Average**
ResNet-50	ResNet-50	74.3±0.5	84.1±0.1	66.9±0.2	45.8±1.8	67.8
	SHOT-IM	61.5±1.7	84.6±0.3	68.0±0.0	33.8±0.3	62.0
	SHOT	61.6±1.8	84.8±0.5	68.0±0.0	34.6±0.3	62.3
	PL	63.4±1.8	80.1±3.5	61.3±1.5	36.8±4.4	60.4
	PL-C	73.3±0.8	84.7±0.3	66.4±0.3	**47.0±1.7**	67.9
	Tent-Full	75.4±0.6	87.0±0.2	66.9±0.2	42.6±0.8	68.0
	BN-Norm	71.3±0.4	85.8±0.1	66.4±0.1	42.3±0.4	66.5
	Tent-C	72.4±1.5	84.4±0.1	66.2±0.2	42.4±3.1	66.4
	Tent-BN	65.6±1.4	84.9±0.0	67.7±0.2	42.7±0.5	65.2
	T3A	76.0±0.3	85.1±0.2	68.2±0.1	44.6±0.9	68.5
	JumpStyle	**76.9±0.7**	**87.5±0.6**	**69.1±0.5**	44.7±0.7	**69.5**
ResNet-18	ResNet-18	73.0±0.6	79.5±0.4	61.8±0.3	41.7±0.9	64.0
	SHOT-IM	61.6±0.3	82.1±0.3	62.5±0.3	32.8±0.4	59.8
	SHOT	61.8±0.3	82.3±0.2	62.8±0.2	32.7±0.4	59.9
	PL	67.0±0.6	72.9±1.0	56.3±2.5	35.4±1.7	57.9
	PL-C	71.8±1.3	78.9±0.4	61.7±0.3	**43.1±0.9**	63.9
	Tent-Full	72.3±0.3	83.9±0.3	62.7±0.2	36.9±0.3	64.0
	BN-Norm	70.4±1.0	82.7±0.1	62.0±0.1	36.4±0.2	62.9
	Tent-C	71.3±1.5	74.6±1.9	60.5±0.4	40.9±0.5	61.8
	Tent-BN	64.7±0.7	81.1±0.2	62.5±0.3	36.4±0.9	61.2
	T3A	74.5±0.9	81.4±0.2	**63.2±0.4**	39.5±0.3	64.6
	JumpStyle	**75.7±0.4**	**86.1±0.6**	**63.3±0.3**	40.5±0.5	**66.4**

Table 4: Results with ERM approach using ResNet-50 and ResNet-18 backbones.

A.3 Hyperparameter selection

As mentioned in Section 4, we use the training domains validation set to determine the hyperparameters η, α and the use of MixStyle layers.

1) We observed that $\eta = 0.8$ gave the best TTA performance on training domains validation set. For further insight, we vary η in JumpStyle and report the results in Table 5. Higher η implies higher weight for pseudo label loss when compared to entropy loss. Thus, consistency checked pseudo-labels provide stronger supervision and help to adapt to the target domain better, leading to improved performance.

η	**VOC**	**LabelMe**	**Caltech**	**SUN09**	**Average**
0.2	68.3±0.7	66.0±0.6	96.3±0.3	63.8±1.2	73.6
0.5	70.0±0.8	66.2±0.5	96.5±0.3	65.4±1.4	74.5
0.8	**71.3±0.4**	**66.5±0.5**	**96.5±0.4**	**68.5±0.6**	**75.7**

Table 5: Performance with varying η on VLCS using ResNet-18.

2) We study the choice of α to mix the source and target BN statistics. As the batch size can be varying during test-time and the quality of test statistics depends on its(higher batch size gives better estimates), we perform experiments setting α to constants 0.4, 0.5, 0.7 and compare the results with the proposed choice of $\alpha(n)$ using eqn.(2).

$\alpha(n)$	**VOC**	**LabelMe**	**Caltech**	**SUN09**	**Average**
0.4	70.7±0.4	64.8±0.7	96.5±0.3	67.3±0.4	74.8
0.5	71.0±0.5	65.9±0.5	95.9±0.4	67.7±0.4	74.8
0.7	71.0±0.4	66.3±0.5	96.5±0.4	68.0±0.6	75.4
Ours (0.6)	**71.3±0.4**	**66.5±0.5**	**96.5±0.4**	**68.5±0.6**	**75.7**

Table 6: Performance with varying α on VLCS using ResNet-18.

3)Based on the analysis presented in MixStyle (Zhou et al., 2021) and our experiments (Table 7), we insert the proposed Style Augmentation layers after the first three ResNet blocks as the early layers contain style information. Results in Table7 show that inserting these layers after each of the three ResNet blocks performs the best.

layers	**VOC**	**LabelMe**	**Caltech**	**SUN09**	**Average**
1	69.3±0.8	66.3±0.7	96.5±0.2	63.7±1.6	74.0
1, 2	70.33±0.7	66.4±0.5	96.3±0.3	65.8±1	74.7
1,2,3	**71.3±0.4**	**66.5±0.5**	**96.5±0.4**	**68.5±0.6**	**75.7**

Table 7: Performance with different layers for augmentation.