Few-Shot Adaptation of Pre-Trained Networks for Domain Shift

Wenyu Zhang[1] , **Li Shen**[1] , **Wanyue Zhang**[2*] and **Chuan-Sheng Foo**[1,3]
[1]Institute for Infocomm Research, A*STAR
[2]Max Planck Institute for Informatics
[3]Centre for Frontier AI Research, A*STAR
{zhang_wenyu, lshen}@i2r.a-star.edu.sg, wzhang@mpi-inf.mpg.de, foo_chuan_sheng@i2r.a-star.edu.sg

Abstract

Deep networks are prone to performance degradation when there is a domain shift between the source (training) data and target (test) data. Recent test-time adaptation methods update batch normalization layers of pre-trained source models deployed in new target environments with streaming data to mitigate such performance degradation. Although such methods can adapt on-the-fly without first collecting a large target domain dataset, their performance is dependent on streaming conditions such as mini-batch size and class-distribution, which can be unpredictable in practice. In this work, we propose a framework for few-shot domain adaptation to address the practical challenges of data-efficient adaptation. Specifically, we propose a constrained optimization of feature normalization statistics in pre-trained source models supervised by a small support set from the target domain. Our method is easy to implement and improves source model performance with as few as one sample per class for classification tasks. Extensive experiments on 5 cross-domain classification and 4 semantic segmentation datasets show that our method achieves more accurate and reliable performance than test-time adaptation, while not being constrained by streaming conditions.

1 Introduction

While deep neural networks have demonstrated remarkable ability in representation learning, their performance relies heavily on the assumption that training *(source domain)* and test *(target domain)* data distributions are the same. However, as real-world data collection can be difficult, time-consuming or expensive, it may not be feasible to adequately capture all potential variation in the training set, such that test samples may be subject to *domain shift* (also known as *covariate shift*) due to factors such as illumination, pose and style [Gulrajani and Lopez-Paz, 2021; Koh *et al.*, 2020]. To prevent severe performance degradation when models are deployed, timely adaptation to the target test distribution is needed.

To carry out this adaptation, a range of methods with varying requirements on the availability of source and target domain data have been developed. In the classic *domain adaptation* (DA) setting, methods assume source and target data are jointly available for training [Wilson and Cook, 2020], which compromises the privacy of source domain data. To address this, methods for the *source-free DA* setting [Qiu *et al.*, 2021; Yang *et al.*, 2021; Liang *et al.*, 2020] instead adapt a pre-trained source model using only unlabeled target data, but they still require access to the entire unlabeled target dataset like traditional DA methods. This can delay adaptation, or even make it impractical, when collecting the unlabeled target data is costly in terms of time or other resources.

Very recently, *test-time adaptation* methods [Nado *et al.*, 2020; Schneider *et al.*, 2020; Wang *et al.*, 2021] have been proposed to adapt pre-trained source models on-the-fly without first collecting a large target domain dataset, by updating batch normalization (BN) layers in pre-trained source models with every mini-batch of target domain data. While these methods reduce the delay to initiate adaptation, they face three main challenges: 1) there is no guarantee that the self-supervised or unsupervised objectives used in these methods can correct domain shift without using target domain labels, 2) performance is dependent on having large mini-batches to obtain good estimates of BN parameters and statistics, 3) test samples need to be class-balanced, which is not always the case in real-world deployments. We show in Figure 1 on the VisDA benchmark that while in ideal conditions (large batch sizes, balanced classes) test-time adaptation methods can improve performance of source models, their performance is severely affected outside of these ideal conditions; in fact, for one of the methods, we observe catastrophic failure where all outputs "collapse" to one or a few classes.

In this work, we propose a different solution to the higher-level objective of source-free domain adaptation with limited target domain data. We suggest that adaptation with a few

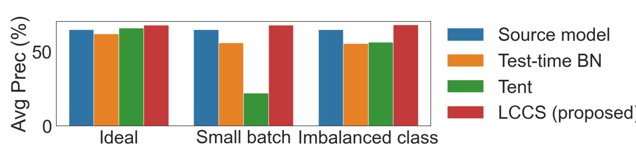


Figure 1: Comparison of cross-domain classification performance on VisDA under various test mini-batch (streaming) conditions.

[*]Work done while at Institute for Infocomm Research.

Setups	**Source-free**	**Training inputs**			**Test inputs**
		Source domain(s)	Target domain	Size of available target data	
Domain adaptation	✗	$L^{s_1}, \dots, L^{s_N}$	entire U^t	$\lvert U^t \rvert$	U^t
k-shot domain adaptation	✗	$L^{s_1}, \dots, L^{s_N}$	k-shot support set $L^{spt} \subset L^t$	$k \times$ # classes	U^t
Domain generalization	✗	$L^{s_1}, \dots, L^{s_N}$	-	0	U^t
Source-free domain adaptation	✓	Pre-trained model on $L^{s_1}, \dots, L^{s_N}$	entire U^t	$\lvert U^t \rvert$	U^t
Test-time adaptation	✓	Pre-trained model on $L^{s_1}, \dots, L^{s_N}$	mini-batch U^t	$\lvert$mini-batch$\rvert^*$, typically 128	U^t
Source-free k-shot adaptation	✓	Pre-trained model on $L^{s_1}, \dots, L^{s_N}$	k-shot support set $L^{spt} \subset L^t$	$k \times$ # classes	U^t

Table 1: Setups where s and t denote source and target domains. L^d and U^d denote labeled and unlabeled datasets from domain d. $*$ denotes sample size to start adaptation, and model completes adaptation after passing through entire U^t. $|U^t|$ can be as large as 55,000 for VisDA.

labeled target domain samples instead of mini-batches of unlabeled data can address the identified challenges with test-time adaptation methods. We propose a new method under a new *source-free k-shot adaptation* setting, that adapts a pre-trained source model using k-shot labeled (or *support*) samples from the target domain; a comparison with existing DA settings is shown in Table 1. Our proposed method adapts batch normalization (BN) layers of deep source models with the support samples using a reparameterization of BN-layers and a low-dimensional approximation of the optimal target domain BN statistics. Although BN layer modulation has been explored for DA [Chang *et al.*, 2019; Nado *et al.*, 2020; Schneider *et al.*, 2020; Wang *et al.*, 2021], reliably optimizing BN layers with extremely few support samples (as few as one per class) is a new and challenging problem. Naively optimizing high-dimensional network parameters directly risks a severely *ill-posed* problem caused by data scarcity. Our method approximates optimal target domain BN statistics with a small set of spanning vectors representing both source and support datasets and finetunes only the combination coefficients, significantly reducing the number of parameters that need to be learned during adaptation. Our method is inspired by the success of controlling sample stylization through BN layers [Huang and Belongie, 2017; Zhou *et al.*, 2021; Nam and Kim, 2018], and we aim to approximate the optimal style to stylize the target samples to address domain shift.

More broadly, our proposed source-free k-shot adaptation setting addresses salient challenges associated with data availability and inference efficiency in real-world deployments. Specifically, the setting helps to protect the privacy of source domains, and has low requirements for target data availability of only k labeled samples per class during adaptation. During inference, test batches can be of *any* size and class composition with no restrictions. As model parameters are not further updated at test-time after k-shot adaptation, our proposed setting enables more efficient inference in comparison to the test-time adaptation setting.

We extensively evaluate our proposed method on 5 image classification and 4 semantic segmentation benchmark datasets and show that it achieves competitive adaptation performance through comprehensive experimental validations and comparisons with state-of-the-art methods.

2 Related Work

Domain adaptation (DA). In the DA setting, a network is trained jointly on labeled source dataset and unlabeled target dataset to optimize task performance on the target domain, as surveyed in [Wilson and Cook, 2020]. Existing source-free DA approaches adapt a pre-trained source model with unlabeled target dataset by generating source-like representations [Qiu *et al.*, 2021], making use of clustering structures in target features for classification [Yang *et al.*, 2021; Liang *et al.*, 2020], or aligning target samples to source hypothesis through entropy minimization and information maximization [Liang *et al.*, 2020; Kundu *et al.*, 2020; Li *et al.*, 2020]. While these methods are similar to our proposed method in not accessing source data, they use a large number of unlabeled target samples for adaptation.

Domain generalization (DG). DG methods aim to learn a model robust to unseen domain shifts by training only on labeled source datasets [Gulrajani and Lopez-Paz, 2021; Koh *et al.*, 2020]. Our proposed method can work directly on source models learned by DG strategies, and hence complements existing DG methods to further close the generalization gap on specific domains of interest.

Test-time adaptation. Test-time adaptation methods update a pre-trained source model continuously during test time with unlabeled target samples to improve performance on the target distribution. [Sun *et al.*, 2020] update network parameters guided by image rotation estimation. However, the method requires training with the self-supervised task on the source dataset, and hence does not work directly with any pre-trained source model. Recent works propose adapting BN layers of the source model instead [Nado *et al.*, 2020; Schneider *et al.*, 2020; Li *et al.*, 2016]. These test-time BN methods re-estimate BN statistics on each mini-batch with mini-batch evaluation to correct domain shifts due to shifts in first and second moments of the data distribution [Nado *et al.*, 2020; Schneider *et al.*, 2020]. Tent [Wang *et al.*, 2021] further updates batch-norm weights and biases with entropy minimization to obtain more confident predictions. However, the reliability of test-time adaptation depends on the number and class-distribution of samples in each mini-batch.

Few-shot transfer learning. There is a wide range of works that learn a metric space specifically for k-shot tasks, and use meta-learning to learn adaptation strategies [Pan and Yang, 2010]. These approaches typically need specific network architectures, loss functions, training strategies requiring multiple source domains or joint training with source and support data together. Since they do not directly work with a pre-trained source model, we do not focus on them here.

Another popular strategy is model finetuning or weight transfer from a pre-trained source model, and our proposed method also falls in this category. Directly finetuning all source model weights on a limited support set with small k is known to severely overfit, so a support set of at least $k = 100$

is often required [Yosinski *et al.*, 2014; Scott *et al.*, 2018]. Recent works constrain the dimensionality of learnable parameter space. FLUTE [Triantafillou *et al.*, 2021] finetunes BN parameters with a nearest-centroid classifier. [Yoo *et al.*, 2018] clusters network parameters and constrains all parameters in a cluster to share the same update, but this method requires activations on the source dataset for clustering.

3 Proposed Method

We first revisit BN layer optimization for DA in Section 3.1 and establish that optimizing the BN layer can be achieved by optimizing BN statistics $\{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$. Then, we introduce our proposed method that optimizes $\{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ on a k-shot support set $\boldsymbol{L^{spt}}$ with a supervised loss in Section 3.2 in the context of classification. For classification tasks, the support set contains k labeled target samples $(\boldsymbol{x}, \boldsymbol{y})$ per class, with image $\boldsymbol{x}$ and one-hot class vector $\boldsymbol{y}$. For semantic segmentation, the support set contains k labeled target samples in total, and $\boldsymbol{y}$ is the segmentation map corresponding to image $\boldsymbol{x}$. Implementation details are described in Section 3.3.

3.1 Revisiting BN Layer Optimization for DA

The batch-norm layer operation f, shown below in Equation 1, is a function acting on input features $\mathbf{Z}$ parameterized by four variables: BN statistics $\{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ and BN parameters $\{\boldsymbol{\gamma}, \boldsymbol{\beta}\}$. Here $\boldsymbol{\mu} = \mathbb{E}[\mathbf{Z}] \in \mathbb{R}^C$ and $\boldsymbol{\sigma} = \sqrt{\mathbb{V}[\mathbf{Z}] + \epsilon} \in \mathbb{R}^C$ are channel-wise feature statistics for C channels, and $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^C$ are layer weight and bias parameters.

$$\mathbf{Z_{BN}} = f(\mathbf{Z}; \boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\gamma}, \boldsymbol{\beta}) = \left(\frac{\mathbf{Z} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}\right)\boldsymbol{\gamma} + \boldsymbol{\beta} \tag{1}$$

We denote $f(\mathbf{Z}; \boldsymbol{\mu_t}, \boldsymbol{\sigma_t}, \boldsymbol{\gamma_t}, \boldsymbol{\beta_t})$ as the BN operation with optimal target domain BN statistics and parameters $\{\boldsymbol{\mu_t}, \boldsymbol{\sigma_t}, \boldsymbol{\gamma_t}, \boldsymbol{\beta_t}\}$. By rewriting this operation in terms of source model BN parameters $\boldsymbol{\beta_s}$ and $\boldsymbol{\gamma_s}$

$$\begin{aligned} f(\mathbf{Z}; \boldsymbol{\mu_t}, \boldsymbol{\sigma_t}, \boldsymbol{\gamma_t}, \boldsymbol{\beta_t}) &= \left(\frac{\mathbf{Z} - \boldsymbol{\mu_t}}{\boldsymbol{\sigma_t}}\right)\boldsymbol{\gamma_t}\frac{\boldsymbol{\gamma_s}}{\boldsymbol{\gamma_s}} + \boldsymbol{\beta_t} + \boldsymbol{\beta_s} - \boldsymbol{\beta_s} \\ &= \left(\frac{\mathbf{Z} - \tilde{\boldsymbol{\mu}}}{\tilde{\boldsymbol{\sigma}}}\right)\boldsymbol{\gamma_s} + \boldsymbol{\beta_s}, \end{aligned} \tag{2}$$

we see that it is equivalent to changing BN statistics in the source model BN operation $f(\mathbf{Z}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}, \boldsymbol{\gamma_s}, \boldsymbol{\beta_s})$ from $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma_s}$ and $\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu_s}$ to $\tilde{\boldsymbol{\sigma}} = \frac{\boldsymbol{\sigma_t}\boldsymbol{\gamma_s}}{\boldsymbol{\gamma_t}}$ and $\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu_t} - \frac{(\boldsymbol{\beta_t} - \boldsymbol{\beta_s})\boldsymbol{\sigma_t}}{\boldsymbol{\gamma_t}}$, where $\{\boldsymbol{\mu_s}, \boldsymbol{\sigma_s}, \boldsymbol{\gamma_s}, \boldsymbol{\beta_s}\}$ are BN statistics and parameters in the pre-trained source model. This observation implies that we can obtain the optimal target domain BN layer $f(\mathbf{Z}; \boldsymbol{\mu_t}, \boldsymbol{\sigma_t}, \boldsymbol{\gamma_t}, \boldsymbol{\beta_t})$ by optimizing only the BN *adaptation statistics* $\{\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}\}$. Edge cases of zero-valued elements in $\boldsymbol{\gamma_s}$ can be avoided by adding a small number to such elements. Setting elements as zero in $\boldsymbol{\gamma_t}$ implies that the corresponding features are completely ignored, which we assume unlikely given a well-trained source feature extractor. A detailed explanation is provided in Appendix B.

3.2 Low Dimensional Approximation for BN Layer Adaptation

Overall formulation. In our setting with extremely few support samples, optimizing BN layers by estimating the high-dimensional BN parameters risks solving an ill-posed problem caused by data scarcity. We hypothesize that source and support set statistics are related to those of the target domain due to shared classes and the relatedness of domains, and propose to approximate target BN statistics in the linear span of these available statistics. We assume a linear relationship as it can be interpreted as style mixing and for computational efficiency. Specifically, for a BN layer with C channels, we approximate the optimal target BN statistics by

Network	# parameters	# BN parameters	# LCCS
ResNet-18	12 million	9,600	80
ResNet-50	26 million	53,120	212
ResNet-101	45 million	105,344	416
DenseNet-121	29 million	83,648	484

Table 2: Comparison of parameter counts for different networks.

$$\boldsymbol{\mu_{LCCS}} = \boldsymbol{M\eta} = [\boldsymbol{\mu_s} \;\; \boldsymbol{M_{spt}}][\eta_s \;\; \boldsymbol{\eta_{spt}^T}]^T \tag{3}$$

$$\boldsymbol{\sigma_{LCCS}} = \boldsymbol{\Sigma\rho} = [\boldsymbol{\sigma_s} \;\; \boldsymbol{\Sigma_{spt}}][\rho_s \;\; \boldsymbol{\rho_{spt}^T}]^T \tag{4}$$

where $\boldsymbol{M}, \boldsymbol{\Sigma} \in \mathbb{R}^{C\times(n+1)}$ contain $n+1$ spanning vectors for first and second moment statistics, and $\boldsymbol{\eta}, \boldsymbol{\rho} \in \mathbb{R}^{n+1}$ are the learnable Linear Combination Coefficients for batch normalization Statistics (LCCS). Our proposed formulation generalizes the original BN operation where $\boldsymbol{\eta_{spt}} = \boldsymbol{\rho_{spt}} = \mathbf{0}$ and $\eta_s = \rho_s = 1$. It also generalizes channel-wise finetuning of BN statistics and parameters: by setting support set spanning vectors as basis vectors and $n = C$, our method is equivalent to unconstrained optimization of BN layers. The key advantage of our approach is that when $n << C$, we can optimize BN layers using significantly fewer learnable parameters.

Spanning vectors. We first describe how source and support set spanning vectors are obtained.

Source domain spanning vectors ($\boldsymbol{\mu_s}, \boldsymbol{\sigma_s}$)*:* We directly utilize BN statistics $\{\boldsymbol{\mu_s}, \boldsymbol{\sigma_s}\}$ from the pre-trained source model. We incorporate source representations to benefit from knowledge shared between source and target domains.

Support set spanning vectors ($\boldsymbol{M_{spt}}, \boldsymbol{\Sigma_{spt}}$)*:* The support set is represented by n spanning vectors extracted by aggregation functions or dimensionality reduction methods such as singular value decomposition (SVD). We choose vectors from the orthogonal basis vectors of per-sample feature statistics at the BN layer. Specifically, for $\boldsymbol{M_{spt}}$ (and similarly for $\boldsymbol{\Sigma_{spt}}$), let $\tilde{\mathbf{Z}}_\mu \in \mathbb{R}^{C\times|\boldsymbol{L^{spt}}|}$ denote the matrix of channel-wise feature mean for each sample. We set the first spanning vector as the overall channel-wise feature mean $\boldsymbol{\mu_{spt}}$ computed across all support samples. We then subtract this first vector as $\tilde{\mathbf{Z}}_\mu^{(\mathbf{1})} = \tilde{\mathbf{Z}}_\mu - \tilde{\mathbf{Z}}_\mu\boldsymbol{\mu_{spt}}\boldsymbol{\mu_{spt}^T}/\|\boldsymbol{\mu_{spt}}\|^2$, factorize $\tilde{\mathbf{Z}}_\mu^{(\mathbf{1})} = \boldsymbol{USV^T}$ using SVD, and extract the top $n-1$ vectors in $\boldsymbol{US}$ as the remaining spanning vectors. The resulting features post-BN are then $\mathbf{Z_{BN}} = \left(\frac{\mathbf{Z} - \boldsymbol{\mu_{LCCS}}}{\boldsymbol{\sigma_{LCCS}}}\right)\boldsymbol{\gamma_s} + \boldsymbol{\beta_s}$.

Learning LCCS parameters. We finetune LCCS parameters $\{\boldsymbol{\eta}, \boldsymbol{\rho}\}$ from all BN layers by minimizing cross-entropy on the support set. After finetuning the LCCS parameters, we use the estimated BN adaptation statistics $\boldsymbol{\mu_{LCCS}}$ and $\boldsymbol{\sigma_{LCCS}}$ on the target domain for inference. Note that we do not adjust the BN statistics further during test time. Hence, unlike test-time BN, our adapted model is not affected by test-time mini-batch size and class distribution.

Comparison of parameter counts. In Table 2, we summarize the number of network parameters versus LCCS parameters when $n = 1$ for common network architectures. Model finetuning involves updating network parameters according to a new objective. Without sufficient labeled samples, updating a large number of parameters is an ill-posed problem. While adapting only the BN parameters allows a smaller number of parameters to be tuned, this is still a large number of learnable parameters in deep networks. Our proposed method dramatically reduces this number to only 4 LCCS parameters per BN layer when $n = 1$, a greater than 100-fold reduction compared to the number of BN parameters. As a result, even when provided with only an extremely small support set, our proposed method is less prone to overfitting.

3.3 Implementation Details

The overall adaptation objective is to finetune LCCS parameters to minimize cross-entropy loss $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\rho}) = -\sum_{(\boldsymbol{x},\boldsymbol{y})\in \boldsymbol{L}^{spt}} \boldsymbol{y} \log h(\boldsymbol{x}; \boldsymbol{\eta}, \boldsymbol{\rho})$ on the support set, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are input and one-hot class encoding of support samples $\boldsymbol{L^{spt}}$, and h is the source model with learnable LCCS parameters $\{\boldsymbol{\eta}, \boldsymbol{\rho}\}$. The proposed method comprises an initialization stage and a gradient update stage.

Initialization stage. We search for initialization values for LCCS to warm start the optimization process. We first compute the support BN statistics $\boldsymbol{\mu_{spt}}$ and $\boldsymbol{\sigma_{spt}}$ by exponential moving average (EMA) for m epochs to allow $\boldsymbol{\mu_{spt}}$ and $\boldsymbol{\sigma_{spt}}$ to update smoothly. Then, we conduct a one-dimensional grid search on the LCCS parameters by setting $\boldsymbol{\eta_{spt}} = \boldsymbol{\rho_{spt}} = [v\, 0 \cdots 0]^T$ where $v \in \{0, 0.1, \ldots, 1.0\}$ and $\eta_s = \rho_s = 1 - v$ with values tied across all BN layers. The initialization value that minimizes cross-entropy loss on the support samples is selected.

Gradient update stage. We compute support set spanning vectors $\boldsymbol{M_{spt}}$ and $\boldsymbol{\Sigma_{spt}}$ with initialized LCCS parameters, and further optimize the parameters with gradient descent for m epochs. In this stage, parameter values are not tied across BN layers and we do not impose sum-to-one constraints on the coefficients to allow more diverse combinations.

We set $m = 10$ epochs in all our experiments. Support samples are augmented with the same data augmentations for source model training. We retain the pre-trained source classifier with extremely small support sets, and update it when sufficient support samples are available to represent the target domain and learn a new target classifier. In our experiments, we use the nearest-centroid (NCC) classifier following [Triantafillou *et al.*, 2021] as the default classifier for $k \geq 5$.

4 Experiments

We evaluate on image classification and segmentation tasks using publicly available benchmark datasets and compare our proposed method to existing source-free methods. For each dataset, our source models are the base models trained on source domain(s) using empirical risk minimization (ERM) or DG methods that have state-of-the-art performance on that dataset. Support samples are randomly selected and experiment results are averaged over at least 3 seeds. We use a mini-batch size of 32 for classification and 1 for segmentation, and use the Adam optimizer with 0.001 learning rate for finetuning LCCS. We provide brief descriptions of task-specific experimental setups in the respective sections; detailed descriptions of the datasets, their corresponding source models and implementation are provided in Appendix C.1.

4.1 Image Classification

We evaluate on 5 classification datasets covering various types of shifts: PACS (7 classes in 4 domains) for style shift, VisDA (12 classes in 2 domains) for synthetic-to-real shift, Camelyon17 (2 classes in 2 domains) and iWildCam (182 classes in 2 domains) for naturally-occurring shifts, and Office (31 classes in 3 domains) for objects photographed in different environments. We adopt evaluation metrics used by recent works: accuracy for PACS and Camelyon17, macro-F1 for iWildCam, average precision for VisDA and average per-class accuracy for Office. For LCCS we set $n = 1$ on iWildCam due to memory constraints, and $n = k \times$ # classes otherwise.

Comparison to test-time adaptation. We compare our method with the baseline source models and test-time adaptation methods that also augment only batch normalization layers. These are Test-time BN [Nado *et al.*, 2020] and the state-of-the-art Tent [Wang *et al.*, 2021] that adapts BN parameters by entropy minimization with default hyperparameters: Adam or SGD with momentum with mini-batch size 128 (larger than support set size with $k = 10$ on datasets evaluated) and learning rate 0.001.

We see from Table 3 that our proposed method improves over source models even with a single example per class for all datasets evaluated, and outperforms Test-time BN and Tent on most cases when $k \geq 5$. We observe that Tent performance is dependent on optimizer choice, with Adam outperforming SGD by approximately 2% on PACS and SGD outperforming Adam by as much as 39.2% on VisDA. Since the better performing optimizer is dataset-dependent, this makes the practical usage of Tent challenging. On PACS, our proposed method outperforms the best case of Tent when the support set has 5 or 10 samples per class. On Camelyon17, all BN adaptation methods improve over source model accuracy by at least 18% in the best case, implying that replacing source BN statistics with target BN statistics is effective in addressing domain shift in this dataset. On the 182-class iWildCam dataset, the test dataset is imbalanced: 102 classes have at least 1 sample, and only 87 classes have at least 5 samples. Hence, to prevent introducing more class imbalance in the adaptation and evaluation processes, we only evaluate the setting where $k = 1$. Our proposed method improves over the source model, while all test-time adaptation methods degrade performance. Similarly, on VisDA, our proposed method obtains the best performance on the target domain.

Effect of test batches: Although test-time adaptation methods can improve the source model without the supervision of labeled samples, their performance relies on streaming conditions and severely degrades with smaller mini-batch size and class-imbalanced mini-batches as shown in Table 4, and

Method	PACS; ERM				PACS; MixStyle				Camelyon17	iWildCam	VisDA
	Art	**Cartoon**	**Photo**	**Sketch**	**Art**	**Cartoon**	**Photo**	**Sketch**			
Source model	76.4	75.8	96.0	67.0	83.9	79.1	95.8	73.8	70.3	<u>31.0</u>	64.7
+ Test-time BN	81.0	79.8	96.2	67.5	83.3	82.1	96.7	74.9	89.9	30.5	60.7
+ Tent (Adam)	83.5	81.8	<u>96.8</u>	71.3	<u>86.0</u>	83.6	96.8	79.2	64.1	18.3	26.5
+ Tent (SGD)	81.1	79.6	96.5	68.2	83.7	82.0	96.4	75.6	**91.4**	29.9	65.7
+ LCCS ($k = 1$)	77.9	80.0	95.9	72.5	82.2	80.4	95.9	78.9	76.6	**31.8**	67.8
+ LCCS ($k = 5$)	<u>85.0</u>	<u>83.3</u>	96.5	**81.5**	85.7	<u>85.5</u>	<u>97.2</u>	<u>80.0</u>	88.3	-	<u>76.0</u>
+ LCCS ($k = 10$)	**86.8**	**86.4**	**97.7**	<u>79.4</u>	**87.7**	**86.9**	**97.5**	**83.0**	<u>90.2</u>	-	**79.2**

Table 3: Comparison with test-time adaptation on classification tasks: 7-class classification on PACS, binary classification on Camelyon17, 182-class classification on iWildCam, and 12-class classification on VisDA.

Method		PACS			VisDA		
	Test batch	Bal.	$\alpha = 10$	$\alpha = 100$	Bal.	$\alpha = 10$	$\alpha = 100$
Source model	any	83.1	79.9	76.9	64.7	55.6	54.7
+ Test-time BN	8	78.6	74.6	63.9	55.7	52.5	51.5
	32	83.3	78.1	66.8	60.7	57.1	55.2
	128	84.3	78.6	66.6	61.9	58.1	55.3
+ Tent	8	81.1	77.2	67.8	22.0	38.7	46.7
	32	86.1	80.6	69.4	59.0	60.8	57.5
	128	86.4	80.0	67.7	65.7	59.8	56.2
+ LCCS ($k = 1$)	any	84.4	81.3	78.4	67.8	67.7	68.0
+ LCCS ($k = 5$)	any	<u>87.1</u>	<u>84.9</u>	<u>81.9</u>	<u>76.0</u>	<u>77.2</u>	<u>77.8</u>
+ LCCS ($k = 10$)	any	**88.8**	**86.8**	**84.3**	**79.2**	**78.8**	**79.1**

Table 4: Classification performance in streaming settings where test-time adaptation can degrade performance: small batch-sizes and imbalanced class distribution. α is the ratio of samples in the largest to smallest class, and Bal. denotes a balanced class-distribution.

in detail in Appendix C.2. We constructed long-tailed imbalanced PACS and VisDA following the procedure in [Cao *et al.*, 2019], where sample sizes decay exponentially from the first to last class with α being the ratio of samples in the largest to smallest class. The good performance of test-time methods is dependent on having large, class-balanced mini-batches whereas our method is independent of streaming conditions and can more reliably adapt to the target domain.

Comparison to few-shot transfer learning. As far as we know, there are no existing methods designed and evaluated specifically for our source-free few-shot DA setting. We thus apply existing methods in DA and few-shot transfer learning to provide benchmark performance. We compare with AdaBN [Li *et al.*, 2016] by replacing source BN statistics with those calculated on target support set, finetuning the source model on BN parameters, classifier, or feature extractor, finetuning the entire model with L^2 or L^2-SP [Li *et al.*, 2018] or DELTA [Li *et al.*, 2019] regularization, Late Fusion [Hoffman *et al.*, 2013] which averages scores from source and target classifier, and FLUTE [Triantafillou *et al.*, 2021] which optimizes BN parameters with nearest-centroid classifier. FLUTE assumes the availability of multiple source datasets to train multiple sets of BN parameters for further blending to initialize the finetuning process. Since we only have access to the pre-trained source model in our setting, we reduce FLUTE to the single source dataset case and initialize FLUTE with single source BN parameters. We follow learning settings in [Li *et al.*, 2019] for regularized finetuning and use the SGD optimizer with momentum and weight decay 0.0004, and L^2 regularization is also added to finetuning on classifier or feature extractor to prevent overfitting. For all other methods that finetune on BN layers, we use the Adam optimizer following [Wang *et al.*, 2021]. We set learning rate 0.001, mini-batch size 32 and epochs 10 for all methods and datasets evaluated.

From Table 5, we observe that regularized finetuning tends to adapt well at $k = 1$, but performance can lag behind with larger support sets. AdaBN does not consistently improve adaptation; we show in Appendix D that completely replacing source BN statistics degrades performance for VisDA. Overall, our proposed method has the best performance in most cases. We also compare our method to L^2 and FLUTE on Office in Table 6, and our method consistently outperforms on all 6 source-target combinations. Though the transfer learning methods produce comparable performance on the relatively easy dataset Camelyon17 (two classes, two domains), LCCS outperforms on more difficult datasets, which demonstrates the effectiveness of the proposed low-dimensional finetuning strategy.

Comparison to source-free unsupervised domain adaptation. We additionally compare our proposed method with source-free unsupervised DA methods which adapt with the entire unlabeled target dataset, including AdaBN [Li *et al.*, 2016], SHOT [Liang *et al.*, 2020], SFDA [Kim *et al.*, 2020] and SDDA [Kurmi *et al.*, 2021] in Table 6 on all 6 source-target pairs in Office. We observe that self-supervision over the entire unlabeled target can produce good adaptation performance. However, despite using only 5 samples per class, our proposed method with finetuned linear classifier (source classifier finetuned for 200 epochs) has best adaptation performance in 5 out of 6 domain pairs, with an average accuracy of 88.9% compared to 88.5% obtained by state-of-the-art source-free unsupervised DA method SHOT.

SHOT outperforms on a more challenging OfficeHome dataset (71.8% vs 67.8%), reflecting the difficulty of adaptation with limited data. Nonetheless, our proposed method outperforms other few-shot methods evaluated by at least 4.5% on average, which demonstrates its effectiveness in the few-shot setting. Full results are provided in Appendix C.4.

Method	PACS			Camelyon17			VisDA		
$k =$	1	5	10	1	5	10	1	5	10
AdaBN	82.9	85.5	85.8	72.9	87.8	<u>90.2</u>	56.5	60.9	61.8
finetune BN	79.0	84.3	85.4	72.6	87.7	90.1	59.1	<u>70.9</u>	74.9
finetune classifier	82.5	83.7	83.8	70.5	70.4	70.5	<u>67.6</u>	69.7	<u>77.4</u>
finetune feat. extractor	<u>83.6</u>	<u>86.0</u>	86.1	<u>79.3</u>	86.5	88.3	67.3	68.4	74.7
L^2	**84.4**	85.8	85.6	**79.6**	<u>88.2</u>	89.5	66.0	66.4	69.6
L^2-SP	**84.4**	85.8	85.6	**79.6**	<u>88.2</u>	89.5	66.0	66.4	69.6
DELTA	**84.4**	85.8	85.6	**79.6**	<u>88.2</u>	89.5	65.9	66.5	70.1
Late Fusion	83.2	83.6	83.6	70.4	70.4	70.5	67.2	69.8	74.5
FLUTE	73.4	85.8	<u>88.1</u>	73.1	86.5	**90.9**	48.3	67.1	65.7
LCCS	**84.4**	**87.1**	**88.8**	76.6	**88.3**	<u>90.2</u>	**67.8**	**76.0**	**79.2**

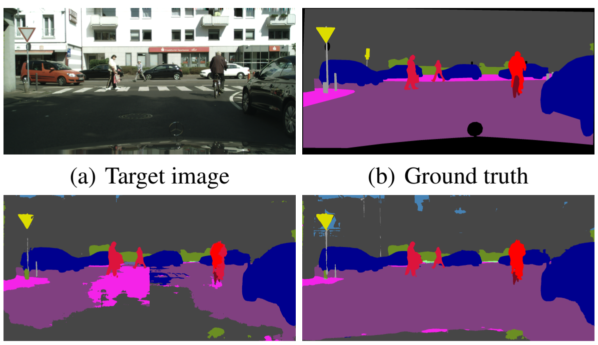
Table 5: Comparison with few-shot transfer learning.

Method	k	Office						
		A → W	A → D	W → A	W → D	D → A	D → W	Avg
SHOT	all†	90.1	**94.0**	74.3	**99.9**	74.7	98.4	88.5
SFDA	all†	91.1	92.2	71.2	99.5	71.0	98.2	87.2
SDDA	all†	82.5	85.3	67.7	99.8	71.0	98.2	84.1
AdaBN	all†	78.2	81.3	59.0	**99.9**	60.3	97.9	79.4
L^2	5	78.9	79.4	64.3	**99.9**	64.8	97.8	80.9
FLUTE	5	84.6	88.2	66.4	99.1	66.4	95.3	83.3
LCCS*	5	**92.8**	91.8	**75.1**	**99.9**	**75.4**	**98.5**	**88.9**

Table 6: Classification accuracy for 31-class classification on Office. Our proposed method can outperform unsupervised DA at improved data efficiency. † denotes target samples are unlabeled. $*$ denotes linear classifier is finetuned on support set.

Method	Cityscapes	BDD-100K	Mapillary	SYNTHIA
ERM	29.0	25.1	28.2	26.2
SW	29.9	27.5	29.7	27.6
IBN-Net	33.9	32.3	37.8	27.9
IterNorm	31.8	32.7	33.9	27.1
ISW	36.6	35.2	40.3	28.3
ISW + L^2	39.5	35.1	40.9	28.1
ISW + LCCS	**43.6**	**37.4**	**42.7**	**29.1**

Table 7: Semantic segmentation mIoU, with GTAV source domain.



(a) Target image (b) Ground truth

(c) ISW (SOTA) (d) ISW + LCCS

Figure 2: Semantic segmentation results for GTAV→ Cityscapes.

4.2 Semantic Segmentation

For semantic segmentation, we transfer from source dataset GTAV (synthetic) to 4 target datasets: SYNTHIA (photo-realistic), Cityscapes (real, urban), BDD-100K (real, urban-driving), and Mapillary (real, street-view). We use the mean-Intersection-over-Union (mIoU) metric to evaluate performance, and set $n = 1$ for LCCS due to memory constraints. We use a total of 5 labeled images as the support set. We compare our method with SW [Pan *et al.*, 2019], IBN-Net [Pan *et al.*, 2018], IterNorm [Huang *et al.*, 2019] and ISW [Choi *et al.*, 2021] which adapt on each sample by instance whitening or standardization, as well as L^2-regularized finetuning on ISW pre-trained models. From Table 7, LCCS finetuning further improves over the state-of-the-art ISW and outperforms ISW + L^2 on all 4 target datasets. The improvement on Cityscapes is as much as 7%, and is visualized in Figure 2.

4.3 Further Analysis

We provide further analysis of our method in the classification setting with base configuration $n = 1$ and source classifier. Additional analyses are provided in Appendix D.

Initialization and gradient update stages. We conduct ablation studies on the initialization and gradient update stages of the proposed method on the VisDA dataset. From Table 8, we see that each stage independently improves the base model's performance, showing that both stages help adaptation.

Stage		Avg Prec (%)		
Initialization	Gradient update	$k = 1$	$k = 5$	$k = 10$
✗	✗	64.7	64.7	64.7
✓	✗	65.9	66.6	66.5
✗	✓	66.0	66.6	68.6
✓	✓	**67.0**	**68.1**	**69.3**

Table 8: Ablation of initialization and gradient update on VisDA.

n	PACS			Camelyon17			VisDA		
$k =$	1	5	10	1	5	10	1	5	10
1	**85.0**	86.0	86.3	76.1	87.6	88.7	67.0	68.1	69.3
10	84.2	**86.2**	**86.7**	**76.9**	88.4	88.6	67.7	69.0	71.2
$k\times$ # classes	84.4	**86.2**	**86.7**	76.6	**88.6**	**88.9**	**67.8**	**69.5**	**72.5**
90% explained var	84.3	**86.2**	**86.7**	**76.9**	88.0	88.4	67.7	69.0	71.4

Table 9: Classification accuracy with n support set spanning vectors.

Design choices for optimizing LCCS. We further experiment with different algorithm design choices in optimizing LCCS parameters. Initializing LCCS with values tied across all BN layers (67.0% avg prec) is better than greedily initializing each BN layer sequentially starting from the shallowest layer (66.1% avg prec). In the gradient update stage, linear combination of statistics (67.0% avg prec) in Equations 3 and 4 performs better than restricting to a convex combination (66.1% avg prec).

Choice of n. From Table 9, we see that with larger support sets, more vectors can be used to represent the target domain and performance improves with a larger number of adaptable LCCS parameters. We observe in general that setting $n = k \times$ # classes for $k \geq 5$ obtains the best performance, while the choice at $k = 1$ is dataset-dependent.

Computational cost. Our method requires little extra training time compared to that of the original network, especially with small k. For simplicity, consider one epoch of gradient-based training. Computation time for our proposed BN layer is $\mathcal{O}(N(A+Cn))$ for constant A denoting the time for back-propagating from the loss to the BN layer's output, versus $\mathcal{O}(N(A+C))$ for the original BN layer. We expect $A >> C$ for large neural networks. Empirically, on VisDA using a Tesla V100-SXM2 GPU, the average training time per epoch on a vanilla ResNet-101 is 3.18s, 3.56s and 4.32s for $k = 1$, 5 and 10 respectively. With our default LCCS parameters of $n = k \times$ # classes, average times are 3.48s, 4.43s and 6.40s.

5 Conclusion

In this work, we proposed the source-free k-shot domain adaptation setting and LCCS finetuning method. LCCS uses a low-dimensional approximation of BN statistics to significantly reduce the number of adaptation parameters, enabling robust adaptation using only a limited support set. Our method adapts source models with as few as one sample per class, is not affected by streaming conditions, and performs well across classification and semantic segmentation benchmarks. These characteristics make our proposed solution a favourable option for data-efficient adaptation, and provide a useful foundation for future work on this challenging topic.

Acknowledgments

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funds (Grant No. A20H6b0151).

References

[Cao *et al.*, 2019] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *NeurIPS*, 2019.

[Chang *et al.*, 2019] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han. Domain-specific batch normalization for unsupervised domain adaptation. *CVPR*, 2019.

[Choi *et al.*, 2021] S. Choi, S. Jung, H. Yun, J. Kim, S. Kim, and J. Choo. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. *CVPR*, 2021.

[Gulrajani and Lopez-Paz, 2021] I. Gulrajani and D. Lopez-Paz. In search of lost domain generalization. *ICLR*, 2021.

[Hoffman *et al.*, 2013] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, and Trevor D. One-shot adaptation of supervised deep convolutional models. *ArXiv*, 2013.

[Huang and Belongie, 2017] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *ICCV*, 2017.

[Huang *et al.*, 2019] L. Huang, Y. Zhou, F. Zhu, L. Liu, and L. Shao. Iterative normalization: Beyond standardization towards efficient whitening. *CVPR*, 2019.

[Kim *et al.*, 2020] Y. Kim, S. Hong, D. Cho, H. Park, and P. Panda. Domain adaptation without source data. *ArXiv*, 2020.

[Koh *et al.*, 2020] P. Koh, S. Sagawa, H. Marklund, S. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. Phillips, S. Beery, J. Leskovec, A. Kundaje, E. Pierson, S. Levine, C. Finn, and P. Liang. Wilds: A benchmark of in-the-wild distribution shifts. *ArXiv*, 2020.

[Kundu *et al.*, 2020] J. N. Kundu, N. Venkat, R. Ambareesh, R. M. V., and R. V. Babu. Towards inheritable models for open-set domain adaptation. *CVPR*, 2020.

[Kurmi *et al.*, 2021] V. Kurmi, Venkatesh K. Subramanian, and Vinay P. Namboodiri. Domain impression: A source data free domain adaptation method. *WACV*, 2021.

[Li *et al.*, 2016] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. *Pattern Recognition*, 80, 03 2016.

[Li *et al.*, 2018] X. Li, Y. Grandvalet, and F. Davoine. Explicit inductive bias for transfer learning with convolutional networks. *ICML*, 2018.

[Li *et al.*, 2019] X. Li, H. Xiong, H. Wang, Y. Rao, L. Liu, and J. Huan. DELTA: Deep learning transfer using feature map with attention for convolutional networks. *ICLR*, 2019.

[Li *et al.*, 2020] R. Li, Q. Jiao, W. Cao, H.-S. Wong, and S. Wu. Model adaptation: Unsupervised domain adaptation without source data. *CVPR*, 2020.

[Liang *et al.*, 2020] J. Liang, D. Hu, and J. Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. *ICML*, 2020.

[Nado *et al.*, 2020] Z. Nado, S. Padhy, D. Sculley, A. D'Amour, B. Lakshminarayanan, and J. Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *ArXiv*, 2020.

[Nam and Kim, 2018] H. Nam and H.-E. Kim. Batch-instance normalization for adaptively style-invariant neural networks. *NeurIPS*, 2018.

[Pan and Yang, 2010] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[Pan *et al.*, 2018] X. Pan, P. Luo, J. Shi, and X. Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. *ECCV*, 2018.

[Pan *et al.*, 2019] X. Pan, X. Zhan, J. Shi, X. Tang, and P. Luo. Switchable whitening for deep representation learning. *ICCV*, 2019.

[Qiu *et al.*, 2021] Z. Qiu, Y. Zhang, H. Lin, S. Niu, Y. Liu, Q. Du, and M. Tan. Source-free domain adaptation via avatar prototype generation and adaptation. *IJCAI*, 2021.

[Schneider *et al.*, 2020] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge. Improving robustness against common corruptions by covariate shift adaptation. *NeurIPS*, 2020.

[Scott *et al.*, 2018] T. R. Scott, K. Ridgeway, and M. Mozer. Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning. *NeurIPS*, 2018.

[Sun *et al.*, 2020] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt. Test-time training for out-of-distribution generalization. *ICML*, 2020.

[Triantafillou *et al.*, 2021] E. Triantafillou, H. Larochelle, R. S. Zemel, and V. Dumoulin. Learning a universal template for few-shot dataset generalization. *ICML*, 2021.

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

[Wilson and Cook, 2020] G. Wilson and D. J. Cook. A survey of unsupervised deep domain adaptation. *ACM Trans. Intell. Syst. Technol.*, 11(5), July 2020.

[Yang *et al.*, 2021] S. Yang, Y. Wang, J. Weijer, L. Herranz, and S. Jui. Generalized source-free domain adaptation. *ICCV*, 2021.

[Yoo *et al.*, 2018] D. Yoo, H. Fan, V. N. Boddeti, and K. M. Kitani. Efficient k-shot learning with regularized deep networks. *AAAI*, 2018.

[Yosinski *et al.*, 2014] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *NIPS*, 2014.

[Zhou *et al.*, 2021] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. Domain generalization with mixstyle. *ICLR*, 2021.