Domain-Specific Batch Normalization for Unsupervised Domain Adaptation

Woong-Gi Chang*[1,2] Tackgeun You*[1,2] Seonguk Seo*[1] Suha Kwak[2] Bohyung Han[1]
[1]Computer Vision Lab., ECE & ASRI, Seoul National University, Korea
[2]Computer Vision Lab., CSE, POSTECH, Korea

Abstract

We propose a novel unsupervised domain adaptation framework based on domain-specific batch normalization in deep neural networks. We aim to adapt to both domains by specializing batch normalization layers in convolutional neural networks while allowing them to share all other model parameters, which is realized by a two-stage algorithm. In the first stage, we estimate pseudo-labels for the examples in the target domain using an external unsupervised domain adaptation algorithm—for example, MSTN [27] or CPUA [14]—integrating the proposed domain-specific batch normalization. The second stage learns the final models using a multi-task classification loss for the source and target domains. Note that the two domains have separate batch normalization layers in both stages. Our framework can be easily incorporated into the domain adaptation techniques based on deep neural networks with batch normalization layers. We also present that our approach can be extended to the problem with multiple source domains. The proposed algorithm is evaluated on multiple benchmark datasets and achieves the state-of-the-art accuracy in the standard setting and the multi-source domain adaption scenario.

1. Introduction

Unsupervised domain adaptation is a learning framework to transfer knowledge learned from source domains with a large number of annotated training examples to target domains with unlabeled data only. This task is challenging due to domain shift problem, which is the phenomenon that source and target datasets have different characteristics. The domain shift is common in real-world problems and should be handled carefully for broad applications of trained models. Unsupervised domain adaptation aims to learn robust models for handling the issue, and is getting popular these days since it can be a rescue for visual recognition tasks relying on the datasets with limited diversity and variety.

*indicates equal contribution.

Recent advances in unsupervised domain adaptation owe to its success of deep neural networks. Traditional domain adaptation techniques based on shallow learning are reformulated using deep neural networks with proper loss functions. Strong representation power of deep networks rediscovers effectiveness of the previous methods and facilitates development of brand-new algorithms. There is a large volume of research on unsupervised domain adaptation based on deep neural networks [3, 4, 10, 14, 23, 27, 30], and in recent years we have witnessed significant performance improvement.

One of the drawbacks in many existing unsupervised domain adaptation techniques [3, 4, 14, 23, 27] is the fact that source and target domains share the whole network for training and prediction. Shared components between both domains are inevitable because there is something common in the two domains; we often need to rely on information in the source domain to learn the networks adapting to unlabeled target domain data. However, we believe that better generalization performance can be achieved by separating domain-specific information from domain-invariant one since the two domains obviously have different characteristics that are not compatible within a single model.

To separate domain-specific information for unsupervised domain adaptation, we propose a novel building block for deep neural networks, referred to as *Domain-Specific Batch Normalization* (DSBN). A DSBN layer consists of two branches of Batch Normalization (BN), each of which is in charge of a single domain exclusively. DSBN captures domain-specific information using BN parameters and transforms domain-specific data into domain-invariant representations using the parameters. Since this idea is generic, DSBN is universally applicable to various deep neural networks for unsupervised domain adaptation that have BN layers. Furthermore, it can be easily extended for multi-source domain adaptation scenarios.

Based on the main idea, we introduce a two-stage framework based on DSBN for unsupervised domain adaptation, where our network first generates pseudo-labels of unlabeled data in the target domain and then learns a fully supervised model using the pseudo-labels. Specifically, the first stage estimates the initial pseudo-labels of target domain data through an existing unsupervised domain adaptation network incorporating DSBN. In the second stage, a multi-task classification network with DSBN layers is trained with full supervision using the data from both source and target domains, where the pseudo-labels generated at the first stage are assigned to the target domain data. To further improve accuracy, we iterate the second stage training and refine the labels of the examples in the target domain.

Our main contributions are summarized as follows:

- We propose a novel unsupervised domain adaptation framework based on DSBN, which is a generic method applicable to various deep neural network models for domain adaptation.
- We introduce a two-stage learning method with DSBN comprising pseudo-label estimation followed by multi-task classification, which is naturally integrated into existing unsupervised domain adaptation methods.
- Our framework provides a principled algorithm for unsupervised domain adaptation with multiple sources via its straightforward extension.
- We achieve the state-of-the-art performance on the standard benchmarks including Office-31 and VisDA-C datasets by integrating our framework with two recent domain adaptation techniques.

The rest of our paper has the following organization. We first review unsupervised domain adaptation approaches in Section 2, and then discuss two recent backbone models to integrate DSBN in Section 3. Section 4 presents the main idea of DSBN and Section 5 describes how to integrate our framework into the existing unsupervised domain adaptation techniques. We show experimental results in Section 6, and conclude this paper with a brief remark in Section 7.

2. Related Work

This section reviews mainstream approaches in unsupervised domain adaptation that are related to our approach.

2.1. Domain-invariant Learning

Learning domain-invariant representation is critical for success of unsupervised domain adaptation, and many existing approaches attempt to align data distributions between source and target domains either globally or locally.

Global alignment techniques include Maximum Mean Discrepancy (MMD) [1, 28], Central Moment Discrepancy (CMD) [29], Wassterstein distance [21], and CORrelation ALignment (CORAL) [22], some of which are reformulated using deep neural networks [12, 23, 28]. Also, adversarial learning has been widely used for global domain alignment. Domain Adversarial Neural Networks (DANN) [3] learn domain-invariant representations using domain adversarial loss while Adversarial Discriminative Domain Adaptation (ADDA) [24] combines adversarial learning with discriminative feature learning. Joint Adaptation Network (JAN) [12] combines adversarial learning with MMD [1] by aligning the joint distributions of multiple domain-specific layers across domains.

Local alignment approaches learn domain-invariant models by aligning examples of the same classes across source and target domains. Cycle-Consistent Adversarial Domain Adaptation (CyCADA) [4] introduces a cycle-consistency loss, which enforces the model to ensure a correct semantic mapping. Conditional Domain Adaptation Network (CDAN) [10] conditions an adversarial adaptation model on discriminative information conveyed in the classifier predictions.

2.2. Domain-specific Learning

Approaches in this category learn domain-specific information together with domain-invariant one for domain adaptation. Domain Separation Network (DSN) [2] learns feature representations by separating domain-specific network components from shared ones. Collaborative and Adversarial Network (CAN) [30] proposes a new loss function that enforces lower-level layers of the network to have high domain-specificity and low domain-invariance while making higher-level layers to have the opposite properties.

Batch normalization parameters have been used to model domain-specific information in unsupervised domain adaptation scenarios. AdaBN [7] proposes a post-processing method to re-estimate batch normalization statistics using target samples. Domain alignment layers used in [15] deal with the domain shift problem by aligning source and target distributions to a reference Gaussian distribution. Also, [13] learns to automatically discover multiple latent source domains, which are used to align target and source distributions. Note that our algorithm is more flexible and efficient as it learns domain-specific properties only with affine parameters of batch normalization and all network parameters except them are used for domain invariant representation.

2.3. Learning with Pseudo Labels

Recent algorithms often estimate pseudo labels in target domain and directly learn a domain-specific model for target domain. To obtain the pseudo labels, Moving Semantic Transfer Network (MSTN) [27] exploits semantic matching and domain adversarial losses while Class Prediction Uncertainty Alignment (CPUA) [14] employs class scores as feature for adversarial learning. Asymmetric Tri-training Network (ATN) [18] and Collaborative and Adversarial Network (CAN) [30] estimate pseudo labels based on the consensus of multiple models.

Our framework also exploits pseudo labels to learn domain-specific models effectively. Compared to the existing method, our framework estimates more reliable pseudo labels since domain-specific information is captured effectively by domain-specific batch normalization.

3. Preliminaries

In unsupervised domain adaptation, we are given two datasets; $\mathcal{X}_S$ is for the labeled source domain and $\mathcal{X}_T$ is the unlabeled dataset for target domain, where n_S and n_T denote the cardinalities of $\mathcal{X}_S$ and $\mathcal{X}_T$, respectively. Our goal is to classify examples in the target domain by transferring knowledge of classification learned from the source domain based on full supervision. This section discusses the details of two state-of-the-art approaches for the integration of our domain-specific batch normalization technique.

3.1. Moving Semantic Transfer Network

MSTN [27] proposes a semantic matching loss function to align the centroids of the same classes across domains, based on pseudo-labels of unlabeled target domain samples. The formal definition of the total loss function $\mathcal{L}$ is given by

$$\mathcal{L} = \mathcal{L}_{\text{cls}}(\mathcal{X}_S) + \lambda \mathcal{L}_{\text{da}}(\mathcal{X}_S, \mathcal{X}_T) + \lambda \mathcal{L}_{\text{sm}}(\mathcal{X}_S, \mathcal{X}_T). \quad (1)$$

The classification loss $\mathcal{L}_{\text{cls}}(\mathcal{X}_S)$ is the cross-entropy loss for source dataset, and the domain adversarial loss $\mathcal{L}_{\text{da}}$ makes a network confused about the domain membership of an example as discussed in [3]. The semantic matching loss aligns the centroids of the same classes across domains. Note that pseudo-labels should be estimated to compute the semantic matching losses. Intuitively, the loss function in Eq. (1) encourages two domains to have the same distribution, especially by adding adversarial and semantic matching loss terms. Hence, the learned network based on the loss function can be applied to examples in target domain.

3.2. Class Prediction Uncertainty Alignment

CPUA [14] is a strikingly simple approach that only aligns the class probabilities across domains. CPUA addresses class imbalance issue in both domain and introduces a class weighted loss function to exploit the class priors.

Let $p_S(c) = n_S^c / n_S$ be the fraction of source samples that have class label c and $\widetilde{p}_T(c) = n_T^c / n_T$ be the fraction of targets samples that have pseudo-label c. n_T^c denotes the cardinality of $\{x \in \mathcal{X}_T \mid \widetilde{y}(x) = c\}$, where $\widetilde{y}(x) = \operatorname{argmax}_{i \in C} F(x)[i]$. Then the class weights for each domain is respectively given by

$$w_S(x, y) = \frac{\max_{y'} p_S(y')}{p_S(y)} \quad (2)$$

and

$$w_T(x) = \frac{\max_{y'} \widetilde{p}_T(y')}{\widetilde{p}_T(\widetilde{y}(x))}. \quad (3)$$

Their total loss function can be written as

$$\mathcal{L} = \mathcal{L}_{\text{cls}}(\mathcal{X}_S) + \lambda \mathcal{L}_{\text{da}}(\mathcal{X}_S, \mathcal{X}_T), \quad (4)$$

where

$$\mathcal{L}_{\text{cls}}(\mathcal{X}_S) = \frac{1}{n_S} \sum_{(x,y) \in \mathcal{X}_S} w_S(x, y) \ell(F(x), y), \quad (5)$$

$$\mathcal{L}_{\text{da}}(\mathcal{X}_S, \mathcal{X}_T) = \frac{1}{n_S} \sum_{(x,y) \in \mathcal{X}_S} w_S(x, y) \ell(D(F(x)), 1) + \frac{1}{n_T} \sum_{x \in \mathcal{X}_T} w_T(x) \ell(D(F(x)), 0). \quad (6)$$

Note that $F(\cdot)$ is a classification network, $\ell(\cdot, \cdot)$ denotes the cross-entropy loss and $D(\cdot)$ is a domain discriminator.

4. Domain-Specific Batch Normalization

This section briefly reviews Batch Normalization (BN) for a comparison to our DSBN, and then presents DSBN and its extension for multi-source domain adaptation.

4.1. Batch Normalization

BN [5] is a widely used training technique in deep networks. A BN layer whitens activations within a mini-batch of N examples for each channel dimension and transforms the whitened activations using affine parameters γ and β. Denoting by $\mathbf{x} \in \mathbb{R}^{H \times W \times N}$ activations in each channel, BN is expressed as

$$\text{BN}(\mathbf{x}[i,j,n]; \gamma, \beta) = \gamma \cdot \hat{\mathbf{x}}[i,j,n] + \beta, \quad (7)$$

where

$$\hat{\mathbf{x}}[i,j,n] = \frac{\mathbf{x}[i,j,n] - \mu}{\sqrt{\sigma^2 + \epsilon}}. \quad (8)$$

The mean and variance of activations within a mini-batch, μ and σ, are computed by

$$\mu = \frac{\sum_n \sum_{i,j} \mathbf{x}[i,j,n]}{N \cdot H \cdot W}, \quad (9)$$

$$\sigma^2 = \frac{\sum_n \sum_{i,j} \left(\mathbf{x}[i,j,n] - \mu\right)^2}{N \cdot H \cdot W}. \quad (10)$$

and ϵ is a small constant to avoid divide-by-zero.

During training, BN estimates the mean and variance of the entire activations, denoted by $\bar{\mu}$ and $\bar{\sigma}$, through exponential moving average with update factor α. Formally, given the t^{th} mini-batch, the mean and variance are given by

$$\bar{\mu}^{t+1} = (1 - \alpha)\bar{\mu}^t + \alpha \mu^t, \quad (11)$$

$$\left(\bar{\sigma}^{t+1}\right)^2 = (1 - \alpha)\left(\bar{\sigma}^t\right)^2 + \alpha \left(\sigma^t\right)^2. \quad (12)$$

In the testing phase, BN uses the estimated mean and variance for whitening input activations. Note that sharing the mean and variance for both source and target domain are inappropriate if domain shift is significant.

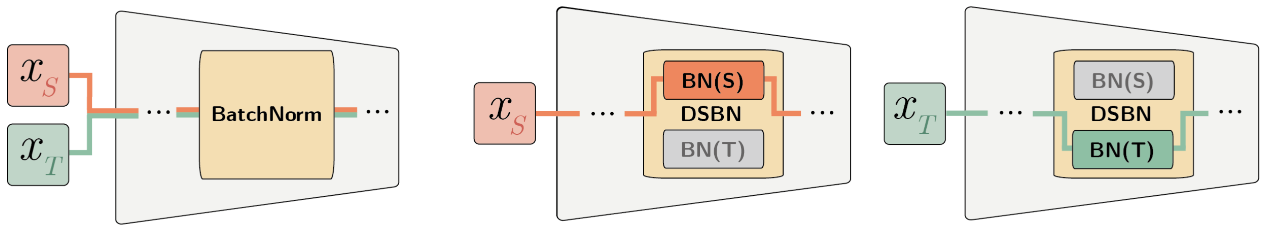


Figure 1. Illustration of difference between BN and DSBN. A DSBN layer consists of two branches in the batch normalization layer—one is for source domain (S), and the other is for target domain (T). Each input example chooses one of the branches according to its domain. In a domain adaptation network with DSBN layers, all parameters except those of DSBN are shared across the two domains and learn the information common in both domains effectively, while domain-specific properties are captured efficiently through the domain specific BN parameters of DSBN layers. Note that DSBN layers can be plugged in any unsupervised domain adapatation networks with BN layers.

4.2. Domain-Specific Batch Normalization

DSBN is implemented by using multiple sets of BN [5] reserved for each domain. Figure 1 illustrates the difference between BN and DSBN. Formally, DSBN allocates domain-specific affine parameters γ_d and β_d for each domain label $d \in \{S, T\}$. Let $\mathbf{x}_d \in \mathbb{R}^{H\times W\times N}$ denote activations at each channel belong to a domain label d, then DSBN layer can be written as

$$\text{DSBN}_d(\mathbf{x}_d[i,j,n]; \gamma_d, \beta_d) = \gamma_d \cdot \hat{\mathbf{x}}_d[i,j,n] + \beta_d, \quad (13)$$

where

$$\hat{\mathbf{x}}_d[i,j,n] = \frac{\mathbf{x}_d[i,j,n] - \mu_d}{\sqrt{\sigma_d^2 + \epsilon}}, \quad (14)$$

and

$$\mu_d = \frac{\sum_n \sum_{i,j} \mathbf{x}_d[i,j,n]}{N \cdot H \cdot W}, \quad (15)$$

$$\sigma_d^2 = \frac{\sum_n \sum_{i,j} \left(\mathbf{x}_d[i,j,n] - \mu_d\right)^2}{N \cdot H \cdot W}. \quad (16)$$

During training, DSBN estimates the mean and variance of activations for each domain separately by exponential moving average with update factor α, which is given by

$$\bar{\mu}_d^{t+1} = (1-\alpha)\bar{\mu}_d^t + \alpha\mu_d^t, \quad (17)$$

$$\left(\bar{\sigma}_d^{t+1}\right)^2 = (1-\alpha)\left(\bar{\sigma}_d^t\right)^2 + \alpha\left(\sigma_d^t\right)^2. \quad (18)$$

The estimated mean and variance for each domain are used for the examples in the corresponding domain at the testing phase of DSBN.

We expect DSBN to capture the domain-specific information by estimating batch statistics and learning affine parameters for each domain separately. We believe that DSBN allows the network to learn domain-invariant features better because domain-specific information within the network can be removed effectively by exploiting the captured statistics and learned parameters from the given domain.

DSBN is easy to be plugged in existing deep neural networks for unsupervised domain adaptation. An existing classification network $F(\cdot)$ can be converted to a domain-specific network by replacing all BN layers with DSBN layers and training the entire network using data with domain labels. The domain-specific network is denoted by $F_d(\cdot)$, which is specialized to either the source or the target domain depending on the domain variable $d \in \{S, T\}$.

4.3. Extension to Multi-Source Domain Adaptation

DSBN is easily extended for multi-source unsupervised domain adaptation by adding more domain branches in it. In addition, a new loss function for the multi-source domain adaptation is simply defined by the sum of losses from all source domains as follows:

$$\mathcal{L} = \frac{1}{|\mathcal{D}_S|}\sum_{i}^{|\mathcal{D}_S|}\Big(\mathcal{L}_{\text{cls}}(\mathcal{X}_{S_i}) + \mathcal{L}_{\text{align}}(\mathcal{X}_{S_i}, \mathcal{X}_T)\Big), \quad (19)$$

where $\mathcal{D}_S = \{\mathcal{X}_{S_1}, \mathcal{X}_{S_2}, ...\}$ is a set of source domains and $\mathcal{L}_{\text{align}}$ can be any kind of loss for aligning source and target domains. The remaining procedure for training is identical to the single-source domain adaptation case.

5. Domain Adaptation with DSBN

DSBN is a generic technique for unsupervised domain adaptation, and can be integrated into various algorithms based on deep neural networks with batch normalization. Our framework trains deep networks for unsupervised domain adaptation in two stages. In the first stage, we train an existing unsupervised domain adaptation network to generate initial pseudo-labels of target domain data. The second stage learns the final models of both domains using the ground-truth in the source domain the pseudo-labels in the target domain as supervision, where the pseudo-labels in the

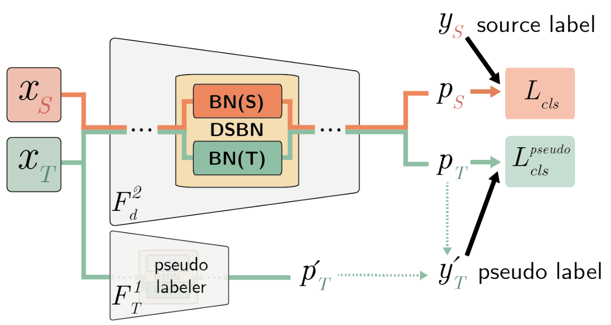


Figure 2. Overview of the second stage training. To use intermediate pseudo labels for target domain samples, we employ the network trained in the first stage, $F_T^1(x)$, as a pseudo labeler in the second stage. In this stage, the network is trained with classification losses only in both domains.

target domain are refined progressively during the training procedure. The networks in both stages incorporate DSBN layers to learn domain-invariant representations more effectively and better adapt to the target domain consequently. To further improve accuracy, we can perform additional iterations of the second stage training, where the pseudo-labels are updated using the results in the preceding iteration. The remaining parts of this section present details of our two stage training method with DSBN.

5.1. Stage 1: Training Initial Pseudo Labeler

Since our framework is generic and flexible, any unsupervised domain adaptation network can be employed to estimate initial pseudo-labels of target domain data as long as it has BN layers. In this paper, we choose two state-of-the-art models as the initial pseudo-label generator: MSTN [27] and CPUA [14]. As described in Section 4.2, we replace their BN layers with DSBNs so that they learn domain-invariant representations more effectively. The networks are then trained by their original losses and learning strategies. A trained initial pseudo-label generator is denoted by F_T^1.

5.2. Stage 2: Self-training with Pseudo Labels

In the second stage, we utilize data and their labels from both domains to take advantage of rich domain-invariant representations and train the final models for both domains with full supervision. The network is trained with two classification losses—one for the source domain with ground-truth labels and the other for the target domain with pseudo-labels—and the resulting networks are denoted by F_d^2 where $d \in \{S, T\}$. The total loss function is given by a simple summation of two loss terms from two domains as follows:

$$\mathcal{L} = \mathcal{L}_{\text{cls}}(\mathcal{X}_S) + \mathcal{L}_{\text{cls}}^{\text{pseudo}}(\mathcal{X}_T) \tag{20}$$

where

$$\mathcal{L}_{\text{cls}}(\mathcal{X}_S) = \sum_{(x,y)\in\mathcal{X}_S} \ell(F_S^2(x), y), \tag{21}$$

$$\mathcal{L}_{\text{cls}}^{\text{pseudo}}(\mathcal{X}_T) = \sum_{x\in\mathcal{X}_T} \ell(F_T^2(x), y'). \tag{22}$$

In Eq. (21) and (22), $\ell(\cdot,\cdot)$ is the cross-entropy loss and y' denotes the pseudo-label assigned to a target domain example $x \in \mathcal{X}_T$.

The pseudo-label y' is initialized by F_T^1 and progressively refined by F_T^2 as follows:

$$y' = \operatorname*{argmax}_{c\in C} \left\{(1-\lambda)F_T^1(x)[c] + \lambda F_T^2(x)[c]\right\}, \tag{23}$$

where $F_T^i(x)[c]$ indicates the prediction score of the class c given by F_T^i and λ is a weight factor that changes gradually from 0 to 1 during training. This approach can be considered as a kind of self-training since F_T^2 takes part in the pseudo-label generation while trained. At an early phase of training, we put more weights on the initial pseudo-labels given by F_T^1 since the prediction by F_T^2 may be unreliable. The weight λ then increases gradually, and at the last phase of training, the pseudo labels rely totally on F_T^2. We follow [3] to suppress potentially noisy pseudo-labels; the adaptation factor λ is gradually increased by $\lambda = \frac{2}{1+\exp(-\gamma\cdot p)} - 1$ with $\gamma = 10$.

Target domain images are recognized more accurately by F_T^2 than F_T^1 in general since F_T^2 exploits reasonable initial pseudo-labels given by F_T^1 for training while F_T^1 relies only on weak information for domain alignment. It is natural to employ F_T^2 to estimate more accurate initial pseudo-labels for the purpose of further accuracy improvement. Thus, we conduct the second stage procedure iteratively, where the initial pseudo-labels are updated using the prediction results of the model in the preceding iteration. We emperically observe that this iterative approach is effective to improve classification accuracy in the target domain.

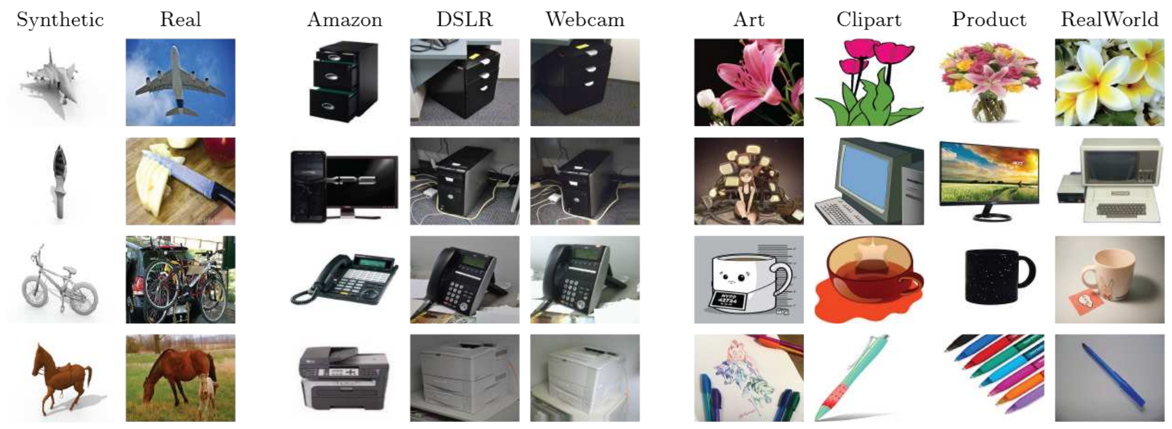
6. Experiments

We present the empirical results to validate the proposed framework and compare our method with the state-of-the-art domain adaptation methods.

6.1. Experimental Settings

We discuss datasets used for training and evaluation and present implementation details including hyperparameter setting.

Datasets We employ three datasets for our experiment: VisDA-C [16], Office-31 [17] and Office-Home [26]. VisDA-C is a large-scale benchmark dataset used for Visual Domain Adaptation Challenge 2017. It consists of two



(a) VisDA-C (b) Office-31 (c) Office-Home

Figure 3. Sample images of each dataset. (a) VisDA-C dataset images of two domains, (b) Office-31 dataset images of three domains, (c) Office-Home dataset images of four domains.

Table 1. Classification performance (%) of multiple algorithms on VisDA-C validation dataset using a ResNet-101 backbone network. The results clearly show that our two-stage learning framework with DSBN is effective to improve accuracy.

Method	aero	bicycle	bus	car	horse	knife	motor	person	plant	skate	train	truck	Avg
Source only	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
DAN [9]	87.1	63.0	76.5	42.0	90.3	42.9	85.9	53.1	49.7	36.3	85.8	20.7	61.1
DANN [3]	81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
MCD [20]	87.0	60.9	83.7	64.0	88.9	**79.6**	84.7	76.9	88.6	40.3	83.0	25.8	71.9
ADR [19]	87.8	79.5	83.7	65.3	92.3	61.8	88.9	73.2	87.8	60	85.5	32.3	74.8
MSTN (reproduced)	89.3	49.5	74.3	67.6	90.1	16.6	93.6	70.1	86.5	40.4	83.2	18.5	65.0
with DSBN (Stage 1)	90.3	78.4	75.0	53.5	90.1	42.8	85.5	76.5	86.7	64.1	81.5	43.6	72.3
with DSBN (Stage 1 and 2)	**94.7**	**86.7**	76.0	**72.0**	**95.2**	75.1	87.9	**81.3**	**91.1**	68.9	**88.3**	45.5	**80.2**
CPUA (reproduced)	90.7	53.4	79.6	59.9	87.9	18.7	**94.5**	61.6	90.4	51.3	81.8	29.3	66.6
with DSBN (Stage 1)	91.1	77.2	**84.0**	49.5	90.7	37.9	87.8	76.6	85.6	60.9	78.1	43.9	71.9
with DSBN (Stage 1 and 2)	93.0	83.6	79.9	55.8	94.7	20.6	89.7	80.2	**91.1**	**80.8**	84.8	**59.7**	76.2

domains—*synthetic* and *real*—and have 152,409 synthetic images and 55,400 real images of 12 common object classes from MS-COCO [8] dataset. Office-31 is a standard benchmark for domain adaptation, which consists of three different domains in 31 categories: Amazon (A) with 2,817 images, Webcam (W) with 795 images and DSLR (D) with 498 images. Office-Home [26] has four domains: Art (Ar) with 2,427 images, Clipart (Cl) with 4,365 images, Product (Pr) with 4,439 images, and Real-World (Rw) with 4,357 images. Each domain contains 65 categories of common daily objects. We adopt the fully transductive protocol introduced in [3] to evaluate our framework on the datasets.

Implementation details Following [3, 20], as the backbone networks of our framework, we adopt ResNet-101 for the VisDA-C dataset and ResNet-50 for Office-31 and Office-Home datasets. All the networks have BN layers and are pre-trained on the ImageNet. To compare the sheer difference between BN and DSBN layers, we construct mini-batches for each domain and forward them separately. The batch size is set to 40, which is identical for all experiments. We use Adam optimizer [6] with $\beta_1 = 0.9, \beta_2 = 0.999$. We set initial learning rate $\eta_0 = 1.0 \times 10^{-4}$ and 5.0×10^{-5} for stage 1 and 2, respectively. As suggested in [3], the learning rate is adjusted by a formula, $\eta_p = \frac{\eta_0}{(1+\alpha p)^\beta}$, where $\alpha = 10$, $\beta = 0.75$, and p denotes training progress linearly changing from 0 to 1. The maximum number of iterations of the optimizer is set to 50,000.

6.2. Results

We present the experimental results on the standard benchmark datasets based on single- and multi-source domain adaptation.

VisDA-C Table 1 quantifies performance of our approach employing MSTN and CPUA as its initial pseudo-label generators, and compares them with state-of-the-art records on the VisDA-C dataset. In the table, "DSBN (Stage 1)" means that we replace BN layers with DSBNs and perform the first stage training, and "DSBN (Stage 1 and 2)" indicates that we conduct both the first and second stage training. Our proposed method improves accuracy substantially and con-

Table 2. Classification accuracies (%) on Office-31 dataset (ResNet-50). *The original paper reported average accuracy of 79.1% with AlexNet.[†]The original paper reported average accuracy of 87.9% with ResNet-50.

Method	$A \rightarrow W$	$W \rightarrow A$	$A \rightarrow D$	$D \rightarrow A$	$W \rightarrow D$	$D \rightarrow W$	Avg
Source Only	73.5	59.8	76.5	56.7	99.0	93.6	76.5
DDC [25]	76.0	63.7	77.5	67.0	98.2	94.8	79.5
DAN [9]	80.5	62.8	78.6	63.6	99.6	97.1	80.4
RTN [11]	84.5	64.8	77.5	66.2	99.4	96.8	81.6
DANN [3]	79.3	63.2	80.7	65.3	99.6	97.3	80.9
JAN [12]	86.0	70.7	85.1	69.2	99.7	96.7	84.6
iCAN [30]	92.5	69.9	90.1	72.1	**100.0**	98.8	87.2
CDAN-M [10]	93.1	70.3	**93.4**	71.0	**100.0**	98.6	87.7
MSTN (reproduced) [27]*	91.3	65.6	90.4	**72.7**	**100.0**	98.9	86.5
with DSBN (Stage 1)	92.5	73.1	90.6	72.2	**100.0**	98.5	87.8
with DSBN (Stage 1 and 2)	92.7	**74.4**	92.2	71.7	**100.0**	99.0	**88.3**
CPUA (reproduced) [14][†]	90.1	71.6	86.8	71.3	**100.0**	98.6	86.4
with DSBN (Stage 1)	92.3	72.7	88.8	72.0	**100.0**	**99.1**	87.5
with DSBN (Stage 1 and 2)	**93.3**	73.9	90.8	**72.7**	**100.0**	**99.1**	**88.3**

Table 3. Classification accuracies (%) in the multi-source scenario on Office-31 dataset using MSTN as a baseline. (ResNet-50)

Single	A→W	D→W	A→D	W→D	W→A	D→A
BN	91.3	98.9	90.4	100.0	65.6	72.7
DSBN	92.5	98.5	90.6	100.0	73.1	72.2
Merged	(A+D)→W		(A+W) →D		(W+D)→A	
BN	99.6		99.6		71.3	
DSBN	99.1		99.6		73.2	
Separate	A, D→W		A, W → D		W, D→A	
BN	**99.9**		99.8		69.9	
DSBN	**99.9**		**100.0**		**75.6**	

Table 4. Classification accuracies (%) in the multi-source scenario on Office-Home dataset using MSTN as a baseline. (ResNet-50)

Single	Ar→Rw	Cl→Rw	Pr→Rw
BN	76.4	69.3	76.9
DSBN	75.4	70.4	77.7
Merged	(Ar+Cl+Pr)→Rw		
BN	81.2		
DSBN	82.3		
Separate	Ar, Cl, Pr→Rw		
BN	81.4		
DSBN	**83.0**		

sistently by applying DSBNs to the baseline models, and achieves the state-of-the-art performance when combined with MSTN [27]. Note also that our model reliably recognizes hard classes such as knife, person, skate, and truck.

Office-31 Table 2 presents overall scores of our approach using MSTN and CPUA on the Office-31 dataset. Models with DSBN trained in both stages achieve state-of-the-art performance and consistently outperform two baseline models. Table 2 also shows that our framework can be applied to current domain adaptation algorithm successfully and greatly improves performance.

Multiple Source Domains Table 3 and Table 4 presents the results of multiple source domain adaptation on the Office-31 and Office-Home datasets, respectively. To compare multi-source to single-source domain adaptation, we report single-source results on the top of the table as "*Single*" and append two different multi-source scenarios: "*Merged*" and "*Separate*". *Merged* means that data from multiple source domains are combined and constructed a new larger source domain dataset while *separate* indicates that each source domain is considered separately. In the *separate* case, we have a total of $|\mathcal{D}_S|+1$ domains and the same number of DSBN branches in the network. While there is a marginal performance gain between BN and DSBN when target tasks are easy, our models consistently outperform the BN models for all settings. In particular, for the hard domain adaptation to task "A" on Table 3, DSBN with source domain separation is considerably better than the *merged* case. This results imply that DSBN has an advantage in multi-source domain adaptation tasks, too. Note that the *seperate* case is not always better than the *merged* case without DSBN.

6.3. Analysis

Ablation Study We perform ablative experiments on our framework to analyze the effect of DSBN in comparison to BN. Table 5 summarizes the ablation results on VisDA-C dataset using MSTN and CPUA as baseline architectures, where the last column in the table presents the accuracy gain by the second stage training with respect to the results from the first stage training only. We test several combinations of different training procedures for two stage training. The results directly show that DSBN plays a crucial role on both the training procedures. Another important point is that the second stage training with DSBN boosts perfor-

Table 5. Ablation results for the combination of batch normalization variations on VisDA-C validation dataset. (ResNet-101), where Δ means the accuracy gain by the second stage training with respect to the results from the first stage training only.

Baseline	Stage 1	Stage 2	aero	bicycle	bus	car	horse	knife	motor	person	plant	skate	train	truck	Avg	Δ
MSTN	BN	BN	81.2	48.7	77.8	75.3	90.2	4.6	**94.4**	73.5	91.1	32.5	85.3	5.9	63.4	-1.6
	DSBN	BN	87.4	67.7	77.2	62.3	92.7	17.8	**94.4**	74.5	**92.5**	63.7	84.4	38.3	71.1	-1.2
	BN	DSBN	92.8	70.3	**78.4**	**76.9**	93.3	14.9	92.7	**81.7**	91.9	**80.2**	85.0	20.3	73.2	+7.2
	DSBN	DSBN	**94.7**	**86.7**	76.0	72.0	**95.2**	**75.1**	87.9	81.3	91.1	68.9	**88.3**	**45.5**	**80.2**	+7.9
CPUA	BN	BN	88.9	43.9	69.8	**68.8**	90.8	9.5	**94.0**	46.0	**92.1**	49.2	82.8	24.8	63.4	-2.8
	DSBN	BN	89.3	53.6	77.9	54.2	91.7	16.1	93.7	74.3	91.6	58.4	79.6	52.4	69.4	-2.6
	BN	DSBN	91.1	66.0	79.3	**68.8**	92.6	18.5	91.2	73.1	91.3	65.7	80.2	37.9	71.3	+4.7
	DSBN	DSBN	**93.0**	**83.6**	**79.9**	55.8	**94.7**	**20.6**	89.7	**80.2**	91.1	**80.8**	**84.8**	**59.7**	**76.2**	+4.3

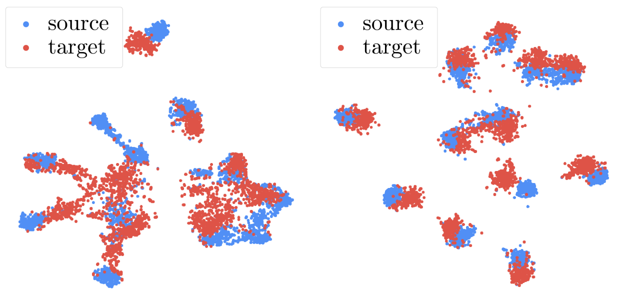


Figure 4. t-SNE plots of the sample representations from ResNet-101 models trained with BN (left) and DSBN (right) using MSTN as a baseline algorithm on VisDA-C validation dataset. They illustrate that DSBN improves the consistency of representations across domains,

mance additionally by large margins while the ordinary BN in the second stage is not helpful. It implies that separating domain-specific information during training phase helps to get reliable pseudo-labels. Note that, especially for hard classes, such a tendency is more pronounced.

Feature Visualization Figure 4 visualizes the instance embedding of BN (left) and DSBN (right) using MSTN as a baseline on VisDA-C dataset. We observe that the examples of two domains in the same class are aligned better by integrating DSBN, which implies that DSBN is effective to learn the domain-invariant representations.

Iterative Learning Our framework adopts the network obtained in the first stage as a pseudo labeler in the second stage, and the network learned in the second stage is stronger than the pseudo labeler. Thus we can expect further performance improvement by applying the second stage learning procedure iteratively, where the pseudo-labels in the current iteration are given by the results in the preceding one. To validate this idea, we evaluate the classification accuracy in each iteration on VisDA-C dataset using MSTN as a baseline algorithm. As shown in Table 6, the iterative learning of the second stage gradually improves accuracy over iterations.

Table 6. Classification accuracies (%) of Iterative learning on VisDA-C dataset using MSTN as a baseline. (ResNet-101)

Stage 1	Stage 2			
	Iter 1	Iter 2	Iter 3	Iter 4
72.3	80.2	81.4	82.2	**82.7**

7. Conclusion

We presented the domain-specific batch normalization for unsupervised domain adaptation. The proposed framework has separate branches of batch normalization layers, one for each domain while sharing all other parameters across domains. This idea is generically applicable to deep neural networks with batch normalization layers. This framework with two stage training strategy is applied to two recent unsupervised domain adaptation algorithms, MSTN and CPUA, and demonstrates outstanding performance in both cases on the standard benchmark datasets. We also present the capability of our framework to be extended to multi-source domain adaptation problems, and report significantly improved results compared to other methods.

Acknowledgments This work was partly supported by Kakao and Kakao Brain Corporation and Korean ICT R&D program of the MSIP/IITP grant [2016-0-00563, 2017-0-01778].

References

[1] Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola. Integrating Structured Biological Data by Kernel Maximum Mean Discrepancy. *Bioinformatics*, 22(14):e49–e57, July 2006. 2

[2] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain Separation Networks. In *NIPS*, 2016. 2

[3] 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(1):2096–2030, 2016. 1, 2, 3, 5, 6, 7

[4] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. CyCADA: Cycle Consistent Adversarial Domain Adaptation. In *ICML*, 2018. 1, 2

[5] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015. 3, 4

[6] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 6

[7] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive Batch Normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018. 2

[8] 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 *ECCV*, 2014. 6

[9] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning Transferable Features with Deep Adaptation Networks. In *ICML*, 2015. 6, 7

[10] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional Adversarial Domain Adaptation. In *NIPS*, 2018. 1, 2, 7

[11] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised Domain Adaptation with Residual Transfer Networks. In *NIPS*, 2016. 7

[12] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep Transfer Learning with Joint Adaptation Networks. In *ICML*, 2017. 2, 7

[13] Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bul, Barbara Caputo, and Elisa Ricci. Boosting Domain Adaptation by Discovering Latent Domains. In *CVPR*, 2018. 2

[14] Jeroen Manders, Elena Marchiori, and Twan van Laarhoven. Simple Domain Adaptation with Class Prediction Uncertainty Alignment. *arXiv preprint arXiv:1804.04448*, 2018. 1, 2, 3, 5, 7

[15] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. AutoDIAL: Automatic DomaIn Alignment Layers. In *ICCV*, 2017. 2

[16] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. VisDA: The Visual Domain Adaptation Challenge, 2017. 5

[17] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting Visual Category Models to New Domains. In *ECCV*, 2010. 5

[18] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric Tri-training for Unsupervised Domain Adaptation. In *ICML*, 2017. 2

[19] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial Dropout Regularization. In *Proc. International Conference on Learning Representations (ICLR)*, 2018. 6

[20] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum Classifier Discrepancy for Unsupervised Domain Adaptation. In *CVPR*, 2018. 6

[21] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein Distance Guided Representation Learning for Domain Adaptation. In *AAAI*, 2018. 2

[22] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of Frustratingly Easy Domain Adaptation. In *AAAI*, 2016. 2

[23] Baochen Sun and Kate Saenko. Deep CORAL: Correlation Alignment for Deep Domain Adaptation. In *ECCV Workshops*, 2016. 1, 2

[24] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial Discriminative Domain Adaptation. In *CVPR*, 2017. 2

[25] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep Domain Confusion: Maximizing for Domain Invariance. *CoRR*, abs/1412.3474, 2014. 7

[26] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep Hashing Network for Unsupervised Domain Adaptation. In *CVPR*, 2017. 5, 6

[27] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning Semantic Representations for Unsupervised Domain Adaptation. In *ICML*, 2018. 1, 2, 3, 5, 7

[28] Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the Class Weight Bias: Weighted Maximum Mean Discrepancy for Unsupervised Domain Adaptation. In *CVPR*, 2017. 2

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

[30] Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu. Collaborative and Adversarial Network for Unsupervised domain adaptation. In *CVPR*, 2018. 1, 2, 7