Pattern Recognition 80 (2018) 109–117

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Adaptive Batch Normalization for practical domain adaptation

Yanghao Li[a], Naiyan Wang[b], Jianping Shi[c], Xiaodi Hou[b], Jiaying Liu[a,*]

[a] Institute of Computer Science and Technology, Peking University, Beijing 100871, PR China
[b] Tusimple, Beijing 100020, PR China
[c] SenseTime, Beijing 100084, PR China

ARTICLE INFO

Article history:
Received 13 August 2017
Revised 26 February 2018
Accepted 4 March 2018
Available online 6 March 2018

Keywords:
Domain adaptation
Batch normalization
Neural networks

ABSTRACT

Deep neural networks (DNN) have shown unprecedented success in various computer vision applications such as image classification and object detection. However, it is still a common annoyance during the training phase, that one has to prepare at least thousands of labeled images to fine-tune a network to a specific domain. Recent study (Tommasi et al., 2015) shows that a DNN has strong dependency towards the training dataset, and the learned features cannot be easily transferred to a different but relevant task without fine-tuning. In this paper, we propose a simple yet powerful remedy, called *Adaptive Batch Normalization* (AdaBN) to increase the generalization ability of a DNN. By modulating the statistics from the source domain to the target domain in all Batch Normalization layers across the network, our approach achieves deep adaptation effect for domain adaptation tasks. In contrary to other deep learning domain adaptation methods, our method does not require additional components, and is parameter-free. It archives state-of-the-art performance despite its surprising simplicity. Furthermore, we demonstrate that our method is complementary with other existing methods. Combining AdaBN with existing domain adaptation treatments may further improve model performance.

1. Introduction

Training a DNN for a new image recognition task is expensive. It requires a large amount of labeled training images that are not easy to obtain. One common practice is to use labeled data from other related source such as a different public dataset, or harvesting images by keywords from a search engine. Because (1) the distributions of the source domains (third party datasets or Internet images) are often different from the target domain (testing images); and (2) DNN is particularly good at capturing dataset bias in its internal representation [2], which eventually leads to overfitting, imperfectly paired training and testing sets usually leads to inferior performance.

Known as domain adaptation, the effort to bridge the gap between training and testing data distributions has been discussed several times under the context of deep learning [3–6]. To make the connection between the domain of training and the domain of testing, most of these methods require additional optimization steps and extra parameters. Such additional computational burden could greatly complicate the training of a DNN which is already intimidating enough for most people.

In this paper, we propose a simple yet effective approach called *AdaBN* for batch normalized DNN domain adaptation. We hypothesize that the label related knowledge is stored in the weight matrix of each layer, whereas domain related knowledge is represented by the statistics of the Batch Normalization (BN) [7] layer. Therefore, we can easily transfer the trained model to a new domain by modulating the statistics in the BN layer. This approach is straightforward to implement, has zero parameter to tune, and requires minimal computational resources. Moreover, our AdaBN is ready to be extended to more sophisticated scenarios such as multi-source domain adaptation and semi-supervised settings. To summarize, our contributions are as follows:

- We propose a novel domain adaptation technique called Adaptive Batch Normalization (AdaBN). We show that AdaBN can naturally dissociate bias and variance of a dataset, which is ideal for domain adaptation tasks.
- We validate the effectiveness of our approach on standard benchmarks for both single source and multi-source domain adaptation. Our method achieves state-of-the-art results.
- We conduct experiments on the cloud detection for remote sensing images to further demonstrate the effectiveness of our approach in practical use.

* Corresponding author.
E-mail addresses: lyttonhao@pku.edu.cn (Y. Li), winsty@gmail.com (N. Wang), shijianping5000@gmail.com (J. Shi), xiaodi.hou@gmail.com (X. Hou), liujiaying@pku.edu.cn (J. Liu).

https://doi.org/10.1016/j.patcog.2018.03.005
0031-3203/

The rest of the paper is organized as follows. Related works are briefly reviewed in Section 2. In Section 3, we introduce a pilot experiment to analyze the domain shift in deep neural networks, and then present the details of the proposed AdaBN algorithm. Section 4 shows the experimental results of our proposed method and we also evaluate a practical application with remote sensing images. Finally, concluding remarks are given in Section 5.

2. Related works

Domain transfer in visual recognition tasks has gained increasing attention in recent literature [8–14]. Often referred to as *covariate shift* [15] or *dataset bias* [2], this problem poses a great challenge to the generalization ability of a learned model. One key component of domain transfer is to model the difference between source and target distributions. In [16], the authors assign each dataset with an explicit bias vector, and train one discriminative model to handle multiple classification problems with different bias terms. A more explicit way to compute dataset difference is based on Maximum Mean Discrepancy (MMD) [17]. This approach projects each data sample into a Reproducing Kernel Hilbert Space, and then computes the difference of sample means. To reduce dataset discrepancies, many methods are proposed, including sample selections [18–20], explicit projection learning [21–23], principal axes alignment [24–27] and non-negative embedding [28].

All of these methods face the same challenge of constructing the domain transfer function – a high-dimensional non-linear function. Due to computational constraints, most of the proposed transfer functions are in the category of simple shallow projections, which are typically composed of kernel transformations and linear mapping functions. In our method, we take advantage of the deep structure of the neural networks to achieve highly non-linear transformations with deep adaptation.

In the field of deep learning, feature transferability across different domains is a tantalizing yet generally unsolved topic [1,29]. To transfer the learned representations to a new dataset, pre-training plus fine-tuning [30] have become *de facto* procedures. However, adaptation by fine-tuning is far from perfect. It requires a considerable amount of labeled data from the target domain, and non-negligible computational resources to re-train the whole network. Instead, the proposed method is designed for the unsupervised domain adaptation which needs no label information from the target domain.

A series of progress has been made in DNN to facilitate domain transfer. Early works of domain adaptation either focus on reordering fine-tuning samples [31], or regularizing MMD [17] in a shallow network [32]. It is only until recently that the problem is directly attacked under the setting of classification of unlabeled target domain using modern convolutional neural network (CNN) architecture. DDC [3] used the classical MMD loss to regularize the representation in the last layer of CNN. DAN [4] further extended the method to multiple kernel MMD and multiple layer adaptation. Besides adapting features using MMD, RTN [33] also added a gated residual layer for classifier adaptation. RevGrad [6] devised a gradient reversal layer to compensate the back-propagated gradients that are domain specific. Recently, by explicitly modeling both private and shared components of the domain representations in the network, Bousmalis et al. [34] proposed a Domain Separation Network to extract better domain-invariant features. All the above methods design specific layers or structure for the neural network to generate better representations for data from target domain. Differently, our method only modulates the statistic of BN layers with very light-weight computation and no additional components or parameters, which is desirable for practical domain adaptation.

Another related work is CORAL [35]. This model focuses on the last layer of CNN. CORAL whitens the data in source domain, and then re-correlates the source domain features to target domain. This operation aligns the second order statistics of source domain and target domain distributions. Surprisingly, such simple approach yields state-of-the-arts results in various text classification and visual recognition tasks. Recently, Deep CORAL [36] also extends the method into DNN by incorporating a CORAL loss. Different from CORAL methods which only transform features in the last layer of CNN, our method can achieve deep adaptation by applying in all BN layers inside neural networks.

Conditional Batch Normalization (CBN) [37–40] shares similar ideas with our method to modify the batch normalization layer for different tasks. It has been proven effective for image stylization [37] and visual question answering [38]. Besides the different target tasks, another main difference between CBN and our method is that CBN modulates the trainable scaling parameters in BN layers as a function of specific input, while our method only updates the mean and variance statistics in BN layers without additional components or parameters.

2.1. Batch normalization

In this section, we briefly review Batch Normalization (BN) [7] which is closely related to our AdaBN. The BN layer is originally designed to alleviate the issue of internal covariate shifting – a common problem while training a very deep neural network. It first standardizes each feature in a mini-batch, and then learns a common slope and bias for each mini-batch. Formally, given the input to a BN layer $\mathbf{X} \in \mathbb{R}^{n \times p}$, where n denotes the batch size, and p is the feature dimension, BN layer transforms a feature $j \in \{1 \dots p\}$ into:

$$\hat{x}_j = \frac{x_j - \mathbb{E}[\mathbf{X}_{\cdot j}]}{\sqrt{\mathrm{Var}[\mathbf{X}_{\cdot j}]}}, \qquad y_j = \gamma_j \hat{x}_j + \beta_j, \tag{1}$$

where x_j and y_j are the input/output scalars of one neuron response in one data sample; $\mathbf{X}_{\cdot j}$ denotes the jth column of the input data; and γ_j and β_j are parameters to be learned. This transformation guarantees that the input distribution of each layer remains unchanged across different mini-batches. For Stochastic Gradient Descent optimization, a stable input distribution could greatly facilitate model convergence, leading to much faster training speed for CNN. Moreover, if training data are shuffled at each epoch, the same training sample will be applied with different transformations, or in other words, more comprehensively augmented throughout the training. During the testing phase, the global statistics of all training samples is used to normalize every mini-batch of test data.

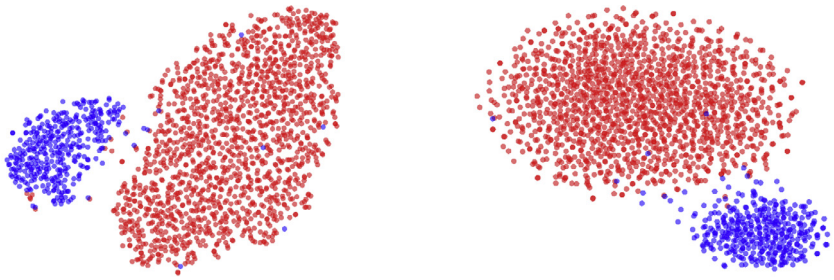
Extensive experiments have shown that BN significantly reduces the number of iteration to converge, and improves the final performance at the same time. BN layer has become a standard component in recent top-performing CNN architectures, such as deep residual network [41], and Inception V3 [42].

3. Adaptive Batch Normalization for domain adaptation

In Section 3.1, we first analyze the domain shift in deep neural network, and reveal two key observations. Then in Section 3.2, we introduce our Adaptive Batch Normalization (AdaBN) method based on these observations.

3.1. A pilot experiment

The Batch Normalization (BN) technique is originally proposed to help SGD optimization by aligning the distribution of training data. From this perspective, it is interesting to examine the BN parameters (batch-wise mean and variance) over different dataset at different layers of the network.



(a) Shallow layer distributions (b) Deep layer distributions

Fig. 1. t-SNE [47] visualization of the mini-batch BN feature vector distributions in both shallow and deep layers, across different datasets. Each point represents the BN statistics in one mini-batch. Red dots come from Bing domain, while the blue ones are from Caltech-256 domain. The size of each mini-batch is 64. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In this pilot experiment, we use MXNet implementation [43] of the Inception-BN model [7] pre-trained on ImageNet classification task [44] as our baseline DNN model. Our image data are drawn from [45], which contains the same classes of images from both Caltech-256 dataset [46] and Bing image search results. For each mini-batch sampled from one dataset, we concatenate the mean and variance of all neurons from one layer to form a feature vector. Using linear SVM, we can almost perfectly classify whether the mini-batch feature vector is from Caltech-256 or Bing dataset. Fig. 1 visualizes the distributions of mini-batch feature vectors from two datasets in 2D. It is clear that BN statistics from different domains are separated into clusters.

This pilot experiment suggests:

1. Both shallow layers and deep layers of the DNN are influenced by domain shift. Domain adaptation by manipulating the output layer alone is not enough.
2. The statistics of BN layer contain the traits of the data domain.

Both observations motivate us to adapt the representation across different domains by BN layer.

3.2. Adaptive Batch Normalization

The core of AdaBN is to adopt domain specific normalization for different domains. As shown in Fig. 2, during training, we train a standard DNN model which is equipped with BN layers with the available labeled images. For inference, we adopt an online algorithm [48] to efficiently and accurately estimate the mean and variance, instead of the common moving average scheme in the training. Specifically, when given a batch of k simples for the neuron j in a BN layer, the mean μ_j and variance σ_j^2 can be updated as follows:

$$\begin{aligned} d =& \mu - \mu_j, \\ \mu_j \leftarrow& \mu_j + \frac{dk}{n_j}, \\ \sigma_j^2 \leftarrow& \frac{\sigma_j^2 n_j}{n_j + k} + \frac{\sigma^2 k}{n_j + k} + \frac{d^2 n_j k}{(n_j + k)^2}, \\ n_j \leftarrow& n_j + k, \end{aligned} \tag{2}$$

where μ and σ^2 are the mean and variance of the current input batch for neuron j, and n_j is the stored statistic of the number of samples for neuron j in the past iterations. Note that the mean μ_j and variance σ_j^2 are initialized as zero and one, respectively. Consequently, we can update the mean and variance of the whole target domain after we iterate over all the samples.

Given the pre-trained DNN model and a target domain, our Adaptive Batch Normalization algorithm is summarized as follows:

The intuition behind our method is straightforward: the standardization of each layer by domain ensures that each layer receives data from a similar distribution, no matter it comes from the source domain or the target domain.

For K domain adaptation where $K > 2$, we standardize each sample by the statistics in its own domain. During training, the statistics are calculated for every mini-batch, the only thing that we need to make sure is that the samples in every mini-batch are from the same domain. For (semi-)supervised domain adaptation, we may use the labeled data to fine-tune the weights as well. As a result, our method could fit in all different settings of domain adaptation with minimal effort.

Algorithm 1 Adaptive Batch Normalization (AdaBN).

for neuron j in DNN **do**
  Collect the neuron responses $\{x_j(m)\}$ on all images of target domain t, where $x_j(m)$ is the response for image m.
  Compute the mean and variance of the target domain: μ_j^t and σ_j^t by Eq. (2).
end for
for neuron j in DNN, testing image m in target domain **do**
  Compute BN output $y_j(m) := \gamma_j \frac{\left(x_j(m) - \mu_j^t\right)}{\sigma_j^t} + \beta_j$
end for

3.3. Discussion about AdaBN

The goal of the domain standardization of each layer in AdaBN is to make each layer receive data from a similar distribution to mitigate the domain shift problem. Our AdaBN shares similar distribution alignment idea with other common domain discrepancy metrics, such as MMD, which is also widely adopted in domain adaptation. Actually, MMD with Gaussian kernel can be viewed as minimizing distance between weighted sums of all moments. Since AdaBN normalizes samples from both two domains with *zero* mean and *one* variance, it can also be viewed as the matching of the first moment and the second moments. In addition, AdaBN matches these two order moment explicitly and does not require time-consuming kernel computations in MMD. This advantage also makes AdaBN possible to apply adaptation scheme inside the whole network.

The simplicity of AdaBN is in sharp contrast to the complication of the domain shift problem. One natural question to ask is

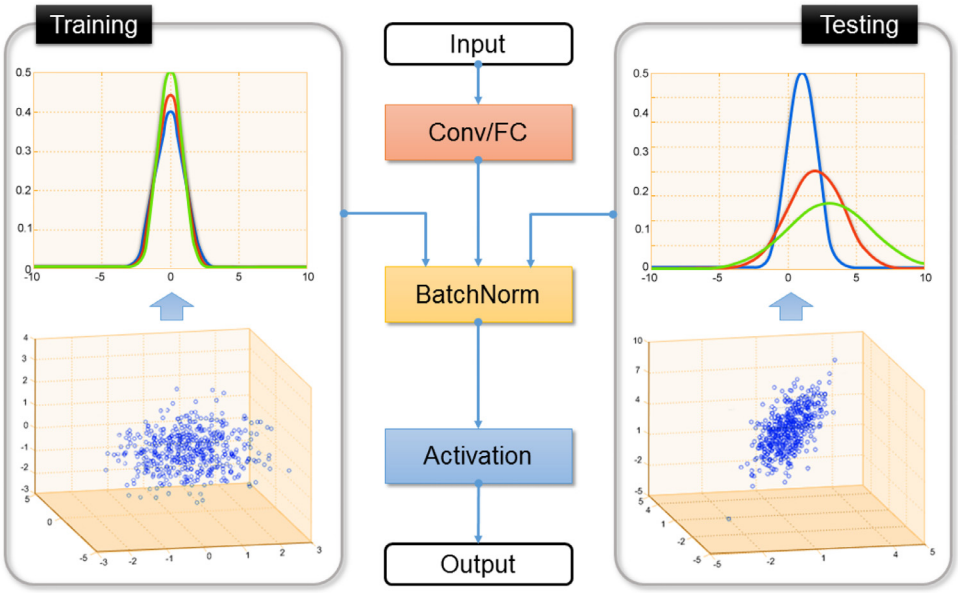


Fig. 2. Illustration of the proposed method. The scatter points correspond to the samples from the two domains, respectively. The samples of the two domains are assumed to be composed of different underlying distributions which are represented by the distribution lines with different colors. For each convolutional or fully connected layer, we use different bias/variance terms to perform batch normalization for the training domain and the test domain. The domain specific normalization mitigates the domain shift issue.

whether such simple translation and scaling operations could approximate the intrinsically non-linear domain transfer function.

Consider a simple neural network with input $\mathbf{x} \in \mathbb{R}^{p_1 \times 1}$. It has one BN layer with mean and variance of each feature being μ_i and σ_i^2 $(i \in \{1 \dots p_2\})$, one fully connected layer with weight matrix $\mathbf{W} \in \mathbb{R}^{p_1 \times p_2}$ and bias $\mathbf{b} \in \mathbb{R}^{p_2 \times 1}$, and a non-linear transformation layer $f(\cdot)$, where p_1 and p_2 correspond to the input and output feature size. The output of this network is $f(\mathbf{W}_a\mathbf{x} + \mathbf{b}_a)$, where

$$\begin{aligned} \mathbf{W}_a &= \mathbf{W}^T \boldsymbol{\Sigma}^{-1}, \\ \mathbf{b}_a &= -\mathbf{W}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \mathbf{b}, \\ \boldsymbol{\Sigma} &= \mathrm{diag}(\sigma_1, \dots, \sigma_{p_1}), \\ \boldsymbol{\mu} &= (\mu_1, \dots, \mu_{p_1}). \end{aligned} \tag{3}$$

The output without BN is simply $f(\mathbf{W}^T\mathbf{x} + \mathbf{b})$. We can see that the transformation is not simple even for one computation layer. As CNN architecture goes deeper, it will gain increasing power to represent more complicated highly non-linear transformations.

Another question is why we transform the neuron responses independently, not decorrelate and then re-correlate the responses as suggested in [35]. Under certain conditions, decorrelation could improve the performance. However, in CNN, the mini-batch size is usually smaller than the feature dimension, leading to singular covariance matrices that is hard to be inversed. As a result, the covariance matrix is always singular. In addition, decorrelation requires to compute the inverse of the covariance matrix which is computationally intensive, especially if we plan to apply AdaBN to all layers of the network.

4. Experiments

In this section, we demonstrate the effectiveness of AdaBN on standard domain adaptation datasets, and empirically analyze our AdaBN model. We also evaluate our method on a practical application with remote sensing images.

4.1. Experimental settings

We first introduce our experiments on two standard datasets: Office [49] and Caltech-Bing [45].

Office [49] is a standard benchmark for domain adaptation, which is a collection of 4652 images in 31 classes from three different domains: *Amazon*(**A**), *DSRL*(**D**) and *Webcam*(**W**). Similar to [3,4,35], we evaluate the pairwise domain adaption performance of AdaBN on all six pairs of domains. For the multi-source setting, we evaluate our method on three transfer tasks {**A, W**} → **D**, {**A, D**} → **W**, {**D, W**} → **A**.

Caltech-Bing [45] is a much larger domain adaptation dataset, which contains 30,607 and 121,730 images in 256 categories from two domains Caltech-256(**C**) and Bing(**B**). The images in the Bing set are collected from Bing image search engine by keyword search. Apparently Bing data contains noise, and its data distribution is dramatically different from that of Caltech-256.

We compare our approach with a variety of methods, including four shallow methods: mSDA [13], SA [24], LSSA [26], GFK [25], CORAL [35], and four deep methods: DDC [3], DAN [4], RevGrad [6], Deep CORAL [36]. Specifically, mSDA introduces marginalized Stacked Denoising Autoencoder to learn better representation for different domains. GFK models domain shift by integrating an infinite number of subspaces that characterize changes in statistical properties from the source to the target domain. SA, LSSA and CORAL align the source and target subspaces by explicit feature space transformations that would map source distribution into the target one. DDC and DAN are deep learning based methods which maximize domain invariance by adding to AlexNet one or several adaptation layers using MMD. RevGrad incorporates a gradient reversal layer in the deep model to encourage learning domain-invariant features. Deep CORAL extends CORAL to perform end-to-end adaptation in DNN. It should be noted that these deep learning methods have the adaptation layers on top of the output layers of DNNs, which is a sharp contrast to our method that delves into early convolution layers as well with the help of BN layers.

We follow the full protocol [30] for the single source setting; while for multiple sources setting, we use all the samples in the source domains as training data, and use all the samples in the target domain as testing data. We fine-tune the Inception-BN [7] model on source domain in each task for 100 epochs. The learning rate is set to 0.01 initially, and then is dropped by a factor 0.1 every 40 epochs. Since the office dataset is quite small, fol-

Table 1
Single source domain adaptation results on Office-31 dataset with standard unsupervised adaptation protocol.

Method	A → W	D → W	W → D	A → D	D → A	W → A	Avg
AlexNet [50]	61.6	95.4	99.0	63.8	51.1	49.8	70.1
DDC [3]	61.8	95.0	98.5	64.4	52.1	52.2	70.6
DAN [4]	68.5	96.0	99.0	67.0	54.0	53.1	72.9
Deep CORAL [36]	66.4	95.7	99.2	66.8	52.8	51.5	72.1
RevGrad [6]	73.0	96.4	99.2	–	–	–	–
Inception BN [7]	70.3	94.3	**100**	70.5	**60.1**	57.9	75.5
mSDA [13]	66.1	96.2	99.4	69.1	57.3	56.7	74.3
SA [24]	69.8	95.5	99.0	71.3	59.4	56.9	75.3
GFK [25]	66.7	**97.0**	99.4	70.1	58.0	56.9	74.7
LSSA [26]	67.7	96.1	98.4	71.3	57.8	57.8	74.9
CORAL [35]	70.9	95.7	99.8	71.9	59.0	60.2	76.3
AdaBN	74.2	95.7	99.8	**73.1**	59.8	57.4	76.7
AdaBN + CORAL	**75.4**	96.2	99.6	72.7	59.0	**60.5**	**77.2**

Table 2
Multi-source domain adaptation results on Office-31 dataset with standard unsupervised adaptation protocol.

Method	A, D → W	A, W → D	D, W → A	Avg
Inception BN [7]	90.8	95.4	60.2	82.1
CORAL [35]	92.1	96.4	**61.4**	83.3
AdaBN	94.2	97.2	59.3	83.6
AdaBN + CORAL	**95.0**	**97.8**	60.5	**84.4**

Table 3
Single source domain adaptation results on Caltech-Bing [45] dataset.

Method	C → B	B → C	Avg
Inception BN [7]	35.1	64.6	49.9
CORAL [35]	**35.3**	67.2	51.3
AdaBN	35.2	**68.1**	**51.7**
AdaBN + CORAL	35.0	67.5	51.2

lowing the best practice in [4], we freeze the first three groups of Inception modules, and set the learning rate of fourth and fifth group one tenth of the base learning rate to avoid overfitting. For Caltech-Bing dataset, we fine-tune the whole model with the same base learning rate.

4.2. Results

4.2.1. Office dataset

Our results on Office dataset is reported in Tables 1 and 2 for single/multi source(s), respectively. Note that the first 5 models of the Table 1 are pre-trained on AlexNet [50] instead of the Inception-BN [7] model, due to the specific design based on AlexNet structure or the lack of publicly available implementation of some methods. Thus, the relative improvements over the baseline (AlexNet/Inception BN) make more sense than the absolute numbers of each algorithm.

From Table 1, we first notice that the Inception-BN indeed improves over the AlexNet on average, which means that the CNN pre-trained on ImageNet has learned general features, the improvements on ImageNet can be transferred to new tasks. Among the methods based on Inception-BN features, our method improves the most over the baseline. Moreover, since our method is complementary to other methods, we can simply apply CORAL on the top of AdaBN. Not surprisingly, this simple combination exhibits 0.5% increase in performance. This preliminary test reveals further potential of AdaBN if combined with other advanced domain adaptation methods. Finally, we could improve 1.7% over the baseline, and advance the state-of-the-art results for this dataset.

None of the compared methods has reported their performance on multi-source domain adaptation. To demonstrate the capacity of AdaBN under multi-domain settings, we compare it against CORAL, which is the best performing algorithm in the single source setting. The result is reported in Table 2. We find that simply combining two domains does not lead to better performance. The result is generally worse compared to the best performing single domain between the two. This phenomenon suggests that if we cannot properly cope with domain bias, the increase of training samples may be reversely affect to the testing performance. This result confirms the necessity of domain adaptation. In this more challenging setting, AdaBN still outperforms the baseline and CORAL on average. Again, when combined with CORAL, our method demonstrates further improvements. At last, our method archives 2.3% gain over the baseline. This improvement should owe to the flexibility of AdaBN to extend to multi-source domain adaptation. As explained in Section 3.2, AdaBN could standardize each sample by the statistics in its own domain. The training data could be treated specifically for different domain during training, while for other methods, the samples from different source domains are mixed when training neural networks.

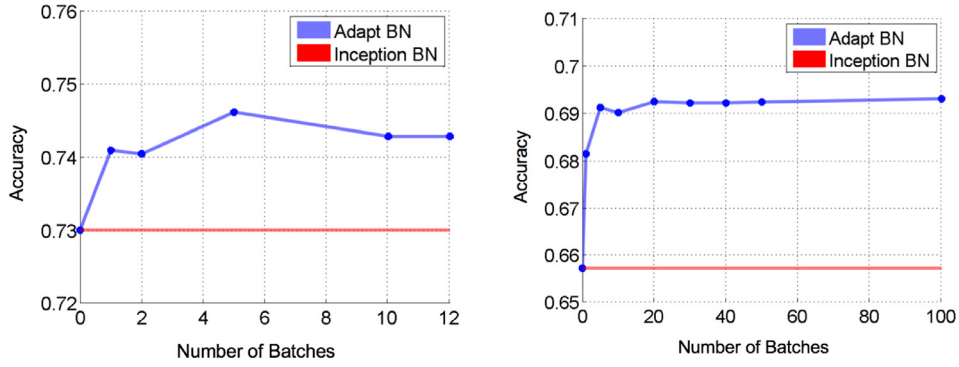
4.2.2. Caltech-Bing Dataset

To further evaluate our method on the large-scale dataset, we show our results on Caltech-Bing Dataset in Table 3. Compared with CORAL, AdaBN achieves better performance, which improves 1.8% over the baseline. Note that all the domain adaptation methods show minor improvements over the baseline in the task **C** → **B**. One of the hypotheses to this relatively small improvement is that the images in Bing dataset are collected from Internet, which are more diverse and noisier [45]. Thus, it is not easy to adapt on the Bing dataset from the relatively clean dataset Caltech-256. Combining CORAL with our method does not offer further improvements. This might be explained by the noise of the Bing dataset and the imbalance of the number of images in the two domains.

4.3. Empirical analysis

In this section, we empirically investigate the influence of the number of samples in target domain to the performance and analyze the adaptation effect of different BN layers.

Sensitivity to target domain size. Since the key of our method is to calculate the mean and variance of the target domain on different BN layers, it is very natural to ask how many target images is necessary to obtain stable statistics. In this experiment, we randomly select a subset of images in target domain to calculate the statistics and then evaluate the performance on the whole target set. Fig. 3 illustrates the effect of using different number of



(a) A → W (b) B → C

Fig. 3. Accuracy when varying the number of mini-batches used for calculating the statistics of BN layers in **A** → **W** and **B** → **C**, respectively. For **B** → **C**, we only show the results of using less than 100 batches, since the results are very stable when adding more examples. The batch size is 64 in this experiment. For even smaller number of examples, the performance may be not consistent and drop behind the baseline (e.g. 0.652 with 16 samples, 0.661 with 32 samples).

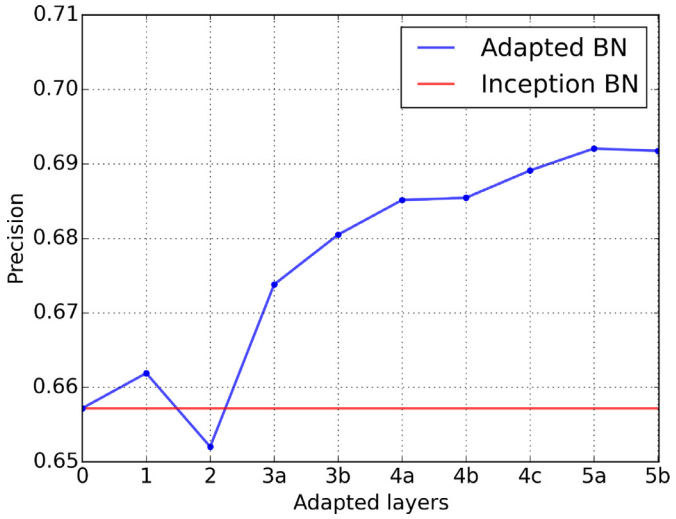


Fig. 4. Accuracy when adapting with different BN blocks in **B** → **C**. $x = 0$ corresponds to the result with non-adapt method, and *1, 2, 3a, 3b, 4a, 4b, 4c, 5a, 5b* correspond to the nine different blocks in Inception-BN network.

batches. The results demonstrate that our method can obtain good results when using only a small part of the target examples. It should also be noted that in the extremal case of one batch of target images, our method still achieves better results than the baseline. This is valuable in practical use since a large number of target images are often not available.

Adaptation effect for different BN layers. In this experiment, we analyze the effect of adapting on different BN layers with our AdaBN method. According to the structure of Inception-BN network, we categorize the BN layers into 9 blocks: *1, 2, 3a, 3b, 4a, 4b, 4c, 5a, 5b*. Since the back BN layers are influenced by the outputs of previous BN layers, when adapting a specific block we adapted all the blocks before it. Fig. 4 illustrates the adaptation effect for different BN layers. It shows that adapting BN layers consistently improves the results over the baseline method in most cases. Specifically, when incorporating more BN layers in the adaptation, we could achieve better transfer results.

4.4. Practical application for cloud detection in remote sensing images

In this section, we further demonstrate the effectiveness of AdaBN on a practical problem: Cloud Detection in Remote Sensing Images. Since remote sensing images are taken by different satellites with different sensors and resolutions, the captured images are visually different in texture, color, and value range distributions, as shown in Fig. 5. How to adapt a model trained on one satellite to another satellite images is naturally a domain adaptation problem.

Table 4
Domain adaptation results (mIOU) on GF1 and Tianhui datasets training on GF2 datasets.

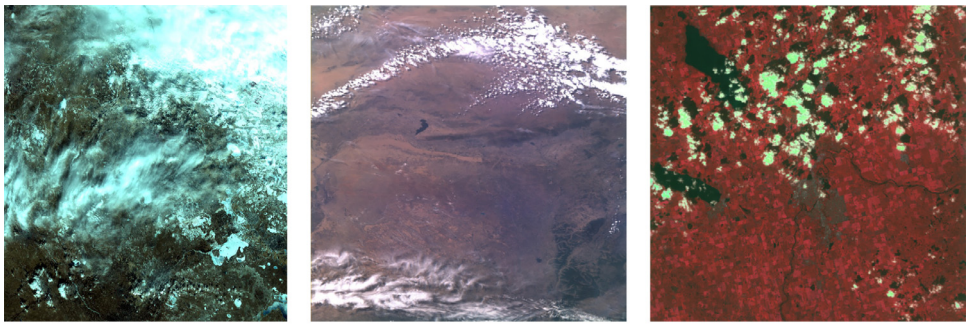
Method	GF1 (%)	Tianhui (%)
Baseline	38.95	14.54
AdaBN	**64.50**	**29.66**

Our task here is to identify cloud from the remote sensing images, which can be regarded as a semantic segmentation task. The experiment is taken under a self-collected dataset, which includes three image sets, from GF2, GF1 and Tianhui satellites. Each image set contains 635, 324 and 113 images with resolution over 6000×6000 pixels, respectively. We name the three different datasets following the satellite names. GF2 dataset is used as the training dataset while GF1 and Tianhui datasets are for testing. We use a state-of-art semantic segmentation method [51] as our baseline model.

The results on GF1 and Tianhui datasets are shown in Table 4.The relatively low results of the baseline method indicate that there exists large distribution disparity among images from different satellites. Thus, the significant improvement after applying AdaBN reveals the effectiveness of our method. Some of the visual results are shown in Fig. 6. Since other domain adaptation methods require either additional optimization steps and extra components (*e.g.* MMD) or post-processing distribution alignment (like CORAL), it is very hard to apply these methods from image classification to this large-size (6000×6000) segmentation problem. Comparatively, besides the effective performance, our method needs no extra parameters and very few computations over the whole adaptation process.

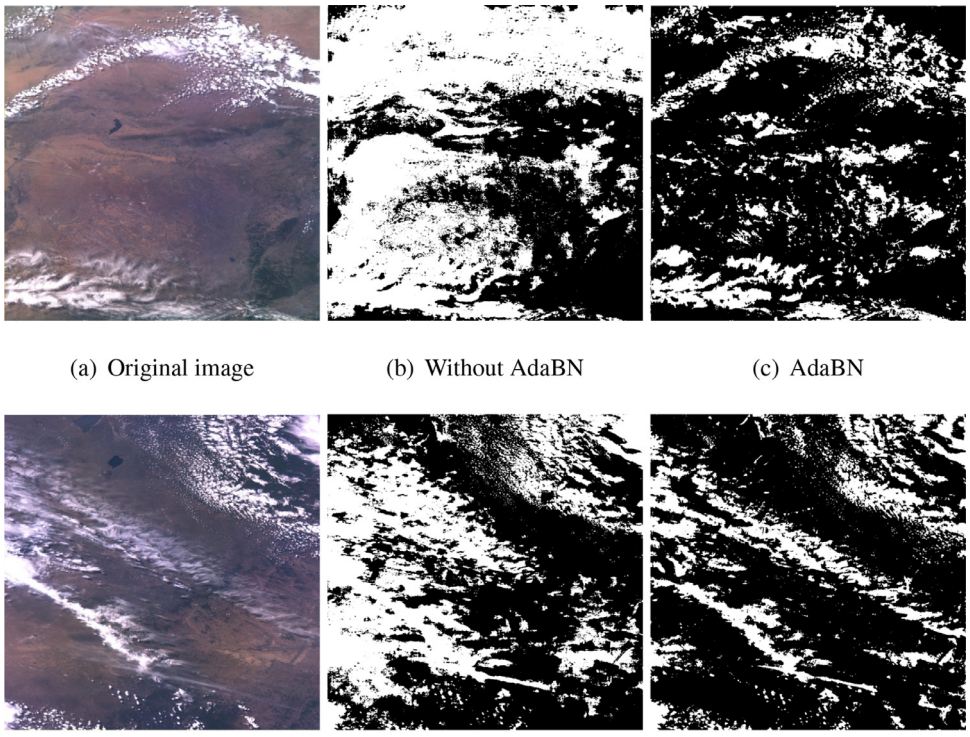
5. Conclusion and future works

In this paper, we have introduced a simple yet effective approach for domain adaptation on batch normalized neural networks. Besides its original uses, we have exploited another functionality of Batch Normalization (BN) layer: domain adaptation.



(a) GF1 image (b) GF2 image (c) Tianhui image

Fig. 5. Remote sensing images in different domains.



(a) Original image (b) Without AdaBN (c) AdaBN

(a) Original image (b) Without AdaBN (c) AdaBN

Fig. 6. Visual cloud detection results on GF1. White pixels in (b) and (c) represent the detected cloud regions.

The main idea is to replace the statistics of each BN layer in source domain with those in target domain. The proposed method is easy to implement and parameter-free, and it takes almost no effort to extend to multiple source domains and semi-supervised settings. Our method established new state-of-the-art results on both single and multiple source(s) domain adaptation settings on standard benchmarks. At last, the experiments on cloud detection for large-size remote sensing images further demonstrate the effectiveness of our method in practical use. We believe our method opens up a new direction for domain adaptation.

In contrary to other methods that use Maximum Mean Discrepancy (MMD) or domain confusion loss to update the weights in CNN for domain adaptation, our method only modifies the statistics of BN layer. Therefore, our method is fully complementary to other existing deep learning based methods. It is interesting to see how these different methods can be unified under one framework.

Acknowledgment

This work was supported by National Natural Science Foundation of China under contract No. 61772043 and CCF-Tencent Open Research Fund. We also gratefully acknowledge the support of NVIDIA Corporation with the GPU for this research.

References

[1] T. Tommasi, N. Patricia, B. Caputo, T. Tuytelaars, A deeper look at dataset bias, in: Proceedings of the GCPR, 2015, pp. 504–516.
[2] A. Torralba, A.A. Efros, Unbiased look at dataset bias, in: Proceedings of the CVPR, 2011, pp. 1521–1528.
[3] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, T. Darrell, Deep domain confusion: Maximizing for domain invariance, 2014. arXiv:http://arxiv.org/abs/1412.3474.
[4] M. Long, Y. Cao, J. Wang, M. Jordan, Learning transferable features with deep adaptation networks, in: Proceedings of the ICML, 2015, pp. 97–105.
[5] E. Tzeng, J. Hoffman, T. Darrell, K. Saenko, Simultaneous deep transfer across domains and tasks, in: Proceedings of the ICCV, 2015, pp. 4068–4076.

[6] Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in: Proceedings of the ICML, 2015, pp. 1180–1189.
[7] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the ICML, 2015, pp. 448–456.
[8] O. Beijbom, Domain adaptations for computer vision applications, 2012. arXiv: https://arxiv.org/abs/1211.4860.
[9] V.M. Patel, R. Gopalan, R. Li, R. Chellappa, Visual domain adaptation: a survey of recent advances, IEEE Signal Process Mag. 32 (3) (2015) 53–69.
[10] G. Csurka, Domain adaptation for visual applications: a comprehensive survey, 2017. arXiv:https://arxiv.org/abs/1702.05374.
[11] L.A. Pereira, R. da Silva Torres, Semi-supervised transfer subspace for domain adaptation, Pattern Recognit. 75 (2018) 235–249.
[12] M. Chen, K.Q. Weinberger, J. Blitzer, Co-training for domain adaptation, in: Proceedings of the NIPS, 2011, pp. 2456–2464.
[13] M. Chen, W. EDU, Z.E. Xu, Marginalized denoising autoencoders for domain adaptation, in: Proceedings of the ICML, 2012.
[14] A.S. Mozafari, M. Jamzad, A svm-based model-transferring method for heterogeneous domain adaptation, Pattern Recognit. 56 (2016) 142–158.
[15] H. Shimodaira, Improving predictive inference under covariate shift by weighting the log-likelihood function, J. Sta.t Plan Inference 90 (2) (2000) 227–244.
[16] A. Khosla, T. Zhou, T. Malisiewicz, A.A. Efros, A. Torralba, Undoing the damage of dataset bias, in: Proceedings of the ECCV, 2012, pp. 158–171.
[17] A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, J. Mach. Learn. Res. 13 (1) (2012) 723–773.
[18] J. Huang, A. Gretton, K.M. Borgwardt, B. Schölkopf, A.J. Smola, Correcting sample selection bias by unlabeled data, in: Proceedings of the NIPS, 2006, pp. 601–608.
[19] B. Gong, K. Grauman, F. Sha, Connecting the dots with landmarks: discriminatively learning domain-invariant features for unsupervised domain adaptation, in: Proceedings of the ICML, 2013, pp. 222–230.
[20] X. Li, M. Fang, J.-J. Zhang, J. Wu, Sample selection for visual domain adaptation via sparse coding, Signal Process. Image Commun. 44 (2016) 92–100.
[21] S.J. Pan, I.W. Tsang, J.T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Trans. Neural Netw. 22 (2) (2011) 199–210.
[22] R. Gopalan, R. Li, R. Chellappa, Domain adaptation for object recognition: an unsupervised approach, in: Proceedings of the ICCV, 2011, pp. 999–1006.
[23] M. Baktashmotlagh, M. Harandi, B. Lovell, M. Salzmann, Unsupervised domain adaptation by domain invariant projection, in: Proceedings of the ICCV, 2013, pp. 769–776.
[24] B. Fernando, A. Habrard, M. Sebban, T. Tuytelaars, Unsupervised visual domain adaptation using subspace alignment, in: Proceedings of the Proceedings of the ICCV, 2013, pp. 2960–2967.
[25] B. Gong, Y. Shi, F. Sha, K. Grauman, Geodesic flow kernel for unsupervised domain adaptation, in: Proceedings of the CVPR, 2012, pp. 2066–2073.
[26] R. Aljundi, R. Emonet, D. Muselet, M. Sebban, Landmarks-based kernelized subspace alignment for unsupervised domain adaptation, in: Proceedings of the CVPR, 2015, pp. 56–63.
[27] B. Fernando, T. Tommasi, T. Tuytelaars, Joint cross-domain classification and subspace learning for unsupervised adaptation, Pattern Recognit. Lett. 65 (2015) 60–66.
[28] I. Redko, Y. Bennani, Non-negative embedding for fully unsupervised domain adaptation, Pattern Recognit. Lett. 77 (2016) 35–41.
[29] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? in: Proceedings of the NIPS, 2014, pp. 3320–3328.
[30] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, DeCAF: a deep convolutional activation feature for generic visual recognition, in: Proceedings of the ICML, 2014, pp. 647–655.
[31] S. Chopra, S. Balakrishnan, R. Gopalan, DLID: Deep learning for domain adaptation by interpolating between domains, in: Proceedings of the ICML Workshop on Challenges in Representation Learning, 2, 2013.
[32] M. Ghifary, W.B. Kleijn, M. Zhang, Domain adaptive neural networks for object recognition, in: Proceedings of the Pacific Rim International Conference on Artificial Intelligence, 2014, pp. 898–904.
[33] M. Long, J. Wang, M.I. Jordan, Unsupervised domain adaptation with residual transfer networks, in: Proceedings of the NIPS, 2016, pp. 136–144.
[34] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, D. Erhan, Domain separation networks, in: Proceedings of the NIPS, 2016, pp. 343–351.
[35] B. Sun, J. Feng, K. Saenko, Return of frustratingly easy domain adaptation., in: Proceedings of the AAAI, 6, 2016, p. 8.
[36] B. Sun, K. Saenko, Deep coral: correlation alignment for deep domain adaptation, in: Proceedings of the ECCV Workshop, 2016, pp. 443–450.
[37] V. Dumoulin, J. Shlens, M. Kudlur, A learned representation for artistic style, in: Proceedings of the ICLR, 2017.
[38] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, A.C. Courville, Modulating early visual processing by language, in: Proceedings of the NIPS, 2017, pp. 6576–6586.
[39] E. Perez, H. de Vries, F. Strub, V. Dumoulin, A. Courville, Learning visual reasoning without strong priors, 2017. arXiv:https://arxiv.org/abs/1707.03017.
[40] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, J. Shlens, Exploring the structure of a real-time, arbitrary neural artistic stylization network, in: Proceedings of the BMVC, 2017.
[41] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the CVPR, 2016, pp. 770–778.
[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the CVPR, 2016, pp. 2818–2826.
[43] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang, MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems, in: Proceedings of the NIPS Workshop on Machine Learning Systems, 2016.
[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[45] A. Bergamo, L. Torresani, Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach, in: Proceedings of the NIPS, 2010, pp. 181–189.
[46] G. Griffin, A. Holub, P. Perona, Caltech-256 Object Category Dataset, California Institute of Technology, 2007.
[47] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2579–2605) (2008) 85.
[48] E.K. Donald, The art of computer programming, Sorting Search. 3 (1999) 426–458.
[49] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: Proceedings of the ECCV, 2010, pp. 213–226.
[50] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the NIPS, 2012, pp. 1097–1105.
[51] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFS, 2016. arXiv:https://arxiv.org/abs/1606.00915.

Yanghao Li received the B.S. degree in computer science from Peking University, Beijing, China, in 2015, where he is currently pursuing the Master degree with the Institute of Computer Science and Technology. His current research interests include domain adaptation, action recognition and detection and computer vision.

Naiyan Wang received the B.S. degree in computer science from Zhejiang University, China, in 2011, and the Ph.D. degree in the Hongkong University of Science and Technology in 2015. He is currently a principal scientist in Tusimple. His research interests include applying statistical computational model to real problems in computer vision and data mining. Currently, he mainly works on the area of visual tracking, object detection, image classification and recommender system.

Jianping Shi received the B.S. degree in computer science and engineering from Zhejiang University, China, in 2011, and the Ph.D. degree in the Chinese University of Hong Kong, 2015. She received the Hong Kong Ph.D. Fellowship and Microsoft Research Asia Fellowship Award in 2011 and 2013, respectively. She is currently a Senior Research Scientist at SenseTime. Her research interests include in computer vision and machine learning. She currently works on developing algorithms for autonomous driving, scene understanding, mobile applications, remote sensing, etc.

Xiaodi Hou received the B.E. degree in computer science from Shanghai Jiao Tong University, China, in 2008, and the Ph.D. degree in the California Institute of Technology in 2014. He is currently the CTO of Tusimple. His current research interests include computer vision and deep learning.

Jiaying Liu received the B.E. degree in computer science from Northwestern Polytechnic University, Xi'an, China, and the Ph.D. degree with the Best Graduate Honor in computer science from Peking University, Beijing, China, in 2005 and 2010, respectively. She is currently an Associate Professor with the Institute of Computer Science and Technology, Peking University. She has authored over 90 technical articles in refereed journals and proceedings, and holds 19 granted patents. Her current research interests include image/video processing, compression, and computer vision. She was a Visiting Scholar with the University of Southern California, Los Angeles, from 2007 to 2008. She was a Visiting Researcher at Microsoft Research Asia (MSRA) in 2015 supported by "Star Track for Young Faculties". She has also served as TC member in IEEE CAS MSA and APSIPA IVM, and APSIPA distinguished lecture in 2016–2017. She is CCF/IEEE Senior Member.