Light-weight calibrator: a separable component for unsupervised domain adaptation

Shaokai Ye[1] Kailu Wu[2] Mu Zhou[3] Yunfei Yang[2] Sia huat Tan[2] Kaidi Xu[4] Jiebo Song[1]
Chenglong Bao[2*] Kaisheng Ma[2*]
[1]Institute for interdisciplinary information core technology(IIISCT), China;
[2]Tsinghua University, China; [3]University of Tsukuba, Japan; [4]Northeastern University, USA

Abstract

Existing domain adaptation methods aim at learning features that can be generalized among domains. These methods commonly require to update source classifier to adapt to the target domain and do not properly handle the trade-off between the source domain and the target domain. In this work, instead of training a classifier to adapt to the target domain, we use a separable component called data calibrator to help the fixed source classifier recover discrimination power in the target domain, while preserving the source domain's performance. When the difference between two domains is small, the source classifier's representation is sufficient to perform well in the target domain and outperforms GAN-based methods in digits. Otherwise, the proposed method can leverage synthetic images generated by GANs to boost performance and achieve state-of-the-art performance in digits datasets and driving scene semantic segmentation. Our method also empirically suggests the potential connection between domain adaptation and adversarial attacks. Code release is available at `https://github.com/yeshaokai/Calibrator-Domain-Adaptation`

1. Introduction

Deep neural networks have achieved great performance in solving diverse machine learning problems. However, solving the so-called domain shift problem is challenging when neural networks are trying to generalize across domains [29, 35, 25]. Extensive efforts have been made on unsupervised domain adaptation [29, 6, 38, 32, 13, 37, 12, 20, 30]. Early domain adaptation methods use different distance metrics or statistics data to align neural networks' feature distribution of source domain with their feature distribution of target domain. Adversarial domain adaptation methods [6, 37] leverage a two players adversarial game to achieve domain adaptation: A domain classifier is encouraged to learn the difference between the feature distribution of two domains while the classification model is encouraged to maximize the classification loss of the domain classifier by learning domain invariant representation that is indistinguishable to the domain classifier. In addition to feature-level adversarial game, there is another line of works that use Generative Adversarial Networks(GANs) [8] to generate source domain images with target domain styles, playing a pixel-level adversarial game.

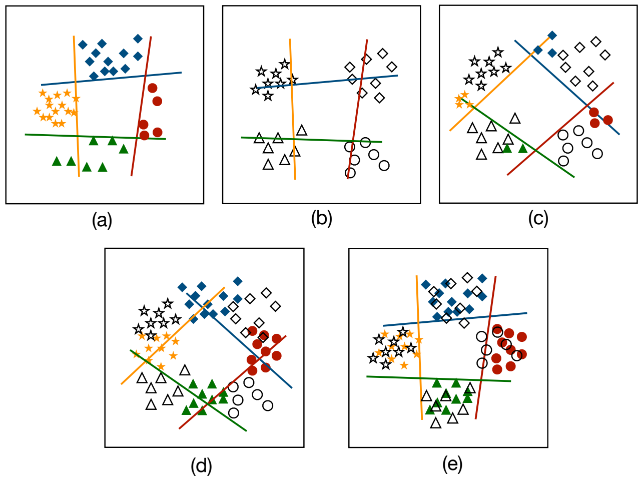

Figure 1. **Concept Illustration**. (a) The source classifier in labeled source domain. (b) The source classifier in unlabeled target domain. (c) Existing methods that are developed to learn domain-invariant features. (d) In real world, the testing set consists of both source domain images and target domain images. (e) The proposed method keeps the representation of source classifier and calibrates target images to fit the source classifier's representation.

However, there are issues that have been rarely discussed. Consider a neural network that is deployed in a device and the device needs to move between different domains. It moves from a domain that is close to its trained source domain to another domain that has no labeled data.

*Corresponding Authors

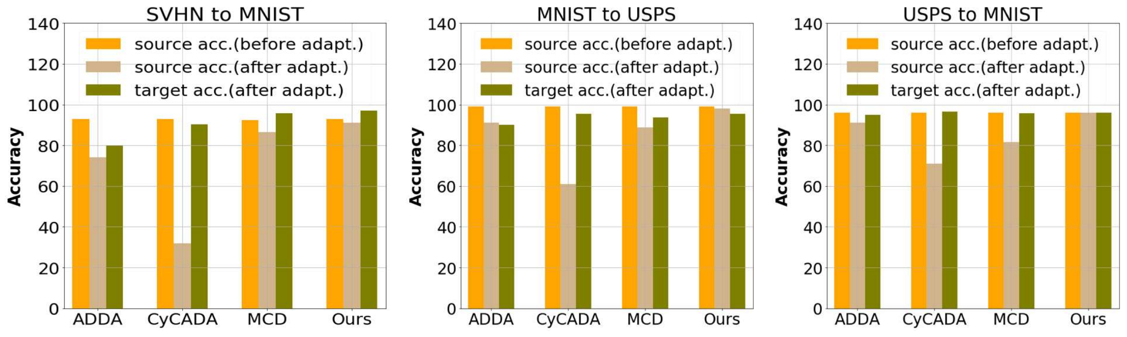

Figure 2. **Performance trade-off between source and target domain.** Some existing methods improve target performance at the expense of source domain performance. In contrast, the proposed method keeps good source domain performance and outperforms these methods in target domain performance.

Traditional unsupervised domain adaptation suffices to handle this simple case. However, the devices can freely move to other domains, which include the source domain. This simple but more realistic scenario brings issues to existing methods. The issues are two folds: (1) Existing methods commonly require to finetune or train a new classifier during domain adaptation. It is not flexible if models are compressed and deployed[11, 45]. (2) Previous methods omit to show the trade-off between source domain performance and target domain performance. Some of them have poor performance trade-off as indicated in Figure 2. Therefore, when the environments are constantly changing, existing methods are likely to have performance degradation and are not able to adapt to new environments in a flexible way.

Some prior works try to work on changing domains [39, 1]. Bobu *et al*. [1] propose to adapt to continuously changing target domains and Wulfmeier *et al*. [39] propose to incrementally adapt to changing domains. However their methods require to finetune the model, and after the model is deployed, the method cannot work properly for unanticipated new domains. We thereby propose two properties a domain adaptation method should have for changing target domains with deployed models.

(1) Good trade-off between source and target domain. Given the complexity of the real world, it is unrealistic to assume that the one chosen target domain is the ultimate application domain. Existing methods assume that the source domain only consists of synthetic images and omit to show the source domain performance after domain adaptation, mostly because that it is assumed the source domain will not be encountered again. A counter example is that both source domain and target domain consist of real world images and source domain will also be encountered. In this case, sacrificing source domain performance is not acceptable.

(2) Flexibility to adapt to arbitrary new domains after being deployed. Deep neural networks are widely deployed in specialized accelerator [10, 34] or mobile phones [21] and are compressed via model compression methods [45, 44, 41, 26] before being deployed and they are not expected to be updated after being deployed. As far as we know, all existing domain adaptation methods require finetune the deployed models to counter new environments.

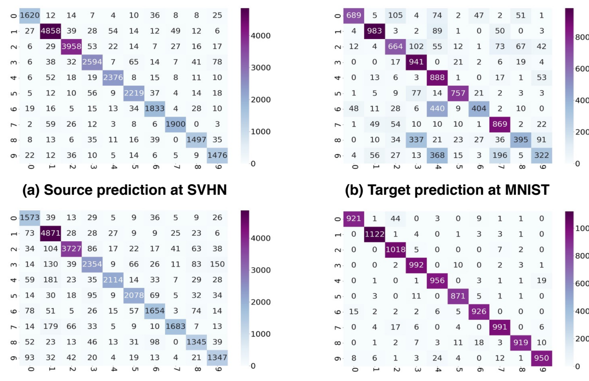

Figure 3. SVHN to MNIST task. Source classifier LeNet is trained in SVHN. (a) The source classifier's prediction on SVHN. (b) The source classifier's prediction on MNIST. (c) The source classifier's prediction on SVHN, with data calibrator. (d) The source classifier's prediction on MNIST, with data calibrator.

It is natural to expect that collecting more data will make a neural network learn universal representation and tremendous investment is made for collecting bigger datasets [5, 16]. However, datasets are found to contain database bias [35, 25]. Training against large datasets does not guarantee the performance of models under changing environments. Therefore, adapting to unanticipated new environments will be necessary and lacking of the flexibility will be an issue.

In this work, we take the first step to mitigate both limitations and formulate unsupervised domain adaptation in a novel way. Figure 1 illustrates the difference between previous methods and our method in the conceptual level. Previous methods commonly update the source classifier's weights when domain adaptation is needed while ours modifies inputs to achieve domain adaptation.

We refer existing methods that attempt to learn cross-domain models as monolithic domain adaptation approach. In contrast, we propose a separable component called data calibrator to achieve domain adaptation, which can be seen as a distributed domain adaptation approach. In our framework, the source classifier is responsible for learning representation under supervised training and the data calibrator is responsible for achieving domain adaptation via unsupervised training.

Our core observation is that the learnt representation from the source domain is not as bad as we thought as shown in Figure 3. The performance degradation brought by domain shift can be mitigated by slightly modifying the target domain images by adding perturbation , which we refer as calibration, to the images. By applying calibration to target domain images, these images fit the source classifier's learnt representation significantly better. We show that we can train a light-weight data calibrator whose number of parameters is only 0.25% to 5.8% of the deployed model and we can use it to adapt the deployed model to arbitrary target domains.

We also want to emphasize that our study focuses on the setting that the source domain and the target domain share the common label space otherwise the source classifier will not work properly in the target domain.

To summarize our contributions:

- We propose a data calibrator to calibrate target domain images to better fit source classifier's representation while maintaining the source domain performance. We improve previous state-of-the-art average accuracy from 95.1% to 97.6% in digits experiments and frequency weighted IoU from 72.4% to 75.1% in GTA5 to CityScapes adaptation.
- The proposed data calibrator is light weight and can be less than 1% in terms of number of parameters compared to the deployed model in GTA5 to CityScapes adaptation and it is a separable domain adaptation approach for it does not need to update the source classifier's weights, thus very convenient for deployment.
- We give new insights on what causes the performance degradation under domain shift and show how to counter it correspondingly.

2. Related Work

Unsupervised Domain Adaptation Visual domain adaptation can trace back to [29]. Early domain adaptation methods focus on aligning deep representation between two domains by using Maximum Mean Discrepancy(MMD) [24, 38, 19] whereas deep Correlation Alignment (CORAL) [32] used statistics such as mean and covariance to achieve feature alignment.

Another line of works leverages the idea of domain classifiers. Torralba *et al*. [35] used ”name the database” to demonstrate that databases are commonly biased and it is even possible to train a domain classifier to correctly classify images to databases they come from. Intuitively, if a domain classifier can learn the difference between source domain and target domain from pixels, then it is also possible for a domain classifier to learn the difference between deep representation of source domain images and target domain images. A line of works explores the idea of training a classifier that confuses the domain classifier by maximizing the domain confusion loss [36, 6, 37, 7, 31, 38]. In addition to the attempt of confusing a domain classifier in the feature level, pixel level adaptation is also explored. Hoffman *et al*. [13] achieve pixel level adaptation for segmentation task, but it uses neural networks' hidden layer output for pixel level adaptation. Our method incorporates both pixel level domain classifier and feature level domain classifier. The pixel level classifier we use directly takes the pixels as inputs, closer to the spirit of ”name the dataset” [35].

Generative Adversarial Networks Another line of works leverages the power of Generative Adversarial Networks (GANs) [8] to generate source images with target images' style. The first of this kind is CoGANs [18] that jointly learns the source domain representation and the target domain representation by forcing the weight sharing between two GANs. Bousmalis *et al*. [2] used GANs to produce images that have similar styles to target domain and make the target task classifier to train images of both. Hoffman *et al*. [12] propose to use semantic consistency loss and cycle consistency loss and achieve significantly better domain adaptation performance. As a comparison, our method can outperform those methods without requiring high-resources to train GANs.

3. A Separable Calibrator For Unsupervised Domain Adaptation

3.1. The overview of the method

In unsupervised domain adaptation, we have access to source domain images X_s and labels Y_s drawn from the source domain distribution $p_s(x, y)$, and target domain images X_t drawn from a target domain distribution $p_t(x, y)$, where there are no labels. Let F_s be the learned classifier for source domain images. The goal of our work is to design a data calibrator G_c such that $F_s \circ G_c$ achieves high accuracy on both source and target domain data. As the classifier F_s is only trained on source domain and there is no information related to the target, the data calibrator G_c has to satisfy:

$$F_s(G_c(X_t)) \sim F_s(X_s), \quad F_s(G_c(X_s)) \sim F_s(X_s) \qquad (1)$$

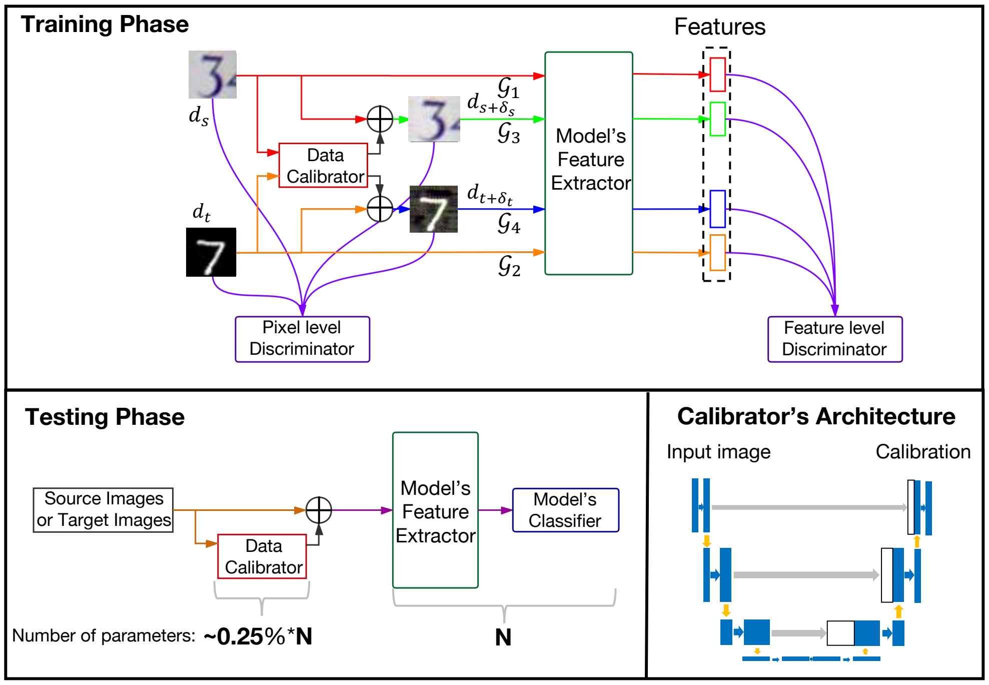

Figure 4. **Training, testing phase and data calibrator architecture.** In the training phase, the pixel level discriminators and the feature space discriminator try to discriminate images to 4 groups while the data calibrator tries to fool both discriminators to treat calibrated images as the source images. In the testing phase, the deployed model takes calibrated images as inputs. The architecture for the data calibrator consists of down sampling layers, up sampling layers and skip connections.

where X_t and X_s are from target and source domain respectively.

Let $F_s = C_s \circ M_s$ where M_s is the feature extractor and C_s is the final classifier. A relaxed condition for achieving (1) is to impose the Lipschitz condition on $F_s \circ G_c$, i.e.

$$\|F_s \circ G_c(x) - F_s \circ G_c(y)\| \leq L\|x - y\|,$$

for some constant $L > 0$ which is a stability condition. Therefore, the following two constraints are imposed on the data calibrator:

$$\begin{aligned} G_c(X_t) \sim X_s, \quad & G_c(X_s) \sim X_s \\ M_s(G_c(X_t)) \sim M_s(X_s), \quad & M_s(G_c(X_s)) \sim M_s(X_s) \end{aligned} \tag{2}$$

It is noted that $G_c(x)$ denotes the input of F_s and M_s denotes the feature map which implies the alignment on both pixel and feature level for source and target domain data. This motivates the following loss function:

$$\begin{aligned} \min_{G_c} \; & H(X_s||G_c(X_t)) + H(M_s(X_s)||M_s(G_c(X_t))) \\ & H(X_s||G_c(X_s)) + H(M_s(X_s)||M_s(G_c(X_s))), \end{aligned} \tag{3}$$

where H denotes the Cross entropy. The loss function in (3) encourages the data calibrator for domain adaption while keeping the performance in source domain. In this work, the data calibrator is set as $G_c = I + G_c^{'}$, i.e. only the perturbation is learned by the calibrator. However, as the target information is blind, minimizing (3) is difficult and another method is needed for training the calibrator G_c.

3.2. Adversarial Domain Adaptation with Proposed Calibrator

In this work, we extend the traditional adversarial domain adaption methods [6, 7, 37] and train the proposed calibrator via adversarial learning instead of minimizing (3).

Traditional adversarial domain adaptation methods play a adversarial game between the target classifier F_t and feature discriminator D_{feat}. Because they update weight parameters of F_t to maximize the confusion loss of domain discriminators, the resulted adapted models lack the flexibility of adjusting to new domains after being deployed and are under the risk of sacrificing source domain performance.

In contrast, the basic idea of our extended adversarial domain adaption method is that let there be pixel level domain discriminator D_{pixel} and feature level domain discriminator D_{feat}. And let a data calibrator modify images such that domain discriminators D_{pixel} can no longer distinguish between $G_c(X_t)$ and X_s nor between $G_c(X_s)$ and X_s. Meanwhile, the corresponding features of calibrated images are also confusing D_{feat} such that the feature level discriminator can no longer distinguish between $M_s(G_c(X_s))$ and $M_s(X_s)$ nor between $M_s(G_c(X_t))$ and $M_s(X_s)$. After the calibrator is trained, inputs are fed to the calibrator before fed to the model, as shown in the testing phase at Figure 4.

As shown in Figure 4, the training of the proposed method needs a trained source classifier F_s. Let the source classifier F_s be trained by the following loss function:

$$\mathcal{L}_{\text{source}}\left(f_S, X_S, Y_S\right) = -\mathbb{E}_{(x_s,y_s)\sim(X_S,Y_S)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log\left(\sigma\left(f_S^{(k)}\left(x_s\right)\right)\right). \quad (4)$$

Based on the learned classifier F_s, the pixel level domain discriminator D_{pixel} and feature level domain discriminator D_{feat} are proposed for training the calibrator such that the pixel and feature level alignment conditions (2) is satisfied. Furthermore, in order to have a finer discrimination power among images and features from source domain and target domain, we divide the inputs of the domain discriminators into 4 groups inspired by the few-shot domain-adaptation [22].

These four groups ($\mathcal{G}_i$,i=1,2,3,4) are defined as follows: $\mathcal{G}_1$ represents source domain images X_s, $\mathcal{G}_2$ represents target domain images X_t. Therefore, learning to distinguish images and features from $\mathcal{G}_1$ and $\mathcal{G}_2$ encourages the domain discriminators to learn the distributions of source domain and target domain. Additionally, calibrated source images $G_c(X_s)$ are defined to belong to $\mathcal{G}_3$ and calibrated target images $G_c(X_t)$ are defined to belong to $\mathcal{G}_4$ as to provide learning signal for the adversarial game. Let $y_{\mathcal{G}_i}, i = 1, 2, 3, 4$ be the group labels for each group.

Feature Level Discriminator. The feature level discriminator aims to discriminate feature level distribution $M_s(\mathcal{G}_i)$. Its objective is to minimize categorical cross entropy loss as following:

$$\mathcal{L}_{feat-D} = -E\left[\sum_{i=1}^{4} y_{\mathcal{G}_i} \log(D_{feat}(M(\mathcal{G}_i)))\right], \quad (5)$$

In our work, the feature level discriminator D_{feat} is a simple neural network with only two fully connected layers. During training, the feature level discriminator learns to discriminate features distribution of $M_s(\mathcal{G}_i)$.

Pixel Level Discriminator. The limitation of using only feature level discriminator is that feature level discriminator cannot fully capture the information in the pixel level after images are transformed via pooling layers and strided convolutional layers of the model. Thus, following the original idea of [35], a pixel level discriminator D_{pixel} is added to learn pixel level distribution of $\mathcal{G}_i$ by following objective function:

$$\mathcal{L}_{pixel-D} = -E\left[\sum_{i=1}^{4} y_{\mathcal{G}_i} \log(D_{pixel}(\mathcal{G}_i))\right]. \quad (6)$$

The pixel level discriminator D_{pixel} shares the same architecture as the feature level discriminator D_{feat}, i.e. a two layer fully connected network. The biggest challenge for the pixel level discriminator D_{pixel} is its tendency of over-fitting to the training set. From our observations in experiments, the validation accuracy starts going down when the training loss for the pixel discriminator gets very low, Indeed, if the calibrator is optimized towards to a pixel level discriminator that overfits, it looses the generalization power. Therefore, we apply following tricks to the inputs of pixel level discriminator to prevent it from overfitting: (1) A image patch is randomly taken from the image. (2) The pixels of the patch is randomly shuffled in the spatial axis. By applying the above two tricks, the overfitting is mitigated.

Data Calibrator. The data calibrator's goal is to fool the pixel level discriminator D_{pixel} and feature level discriminator D_{pixel} by the following loss function:

$$\begin{aligned}\mathcal{L}_{Calibrator} = -E[&y_{\mathcal{G}_1} \log(D_{feat}(M_s(\mathcal{G}_3))) \\ &+ y_{\mathcal{G}_1} \log(D_{feat}(M_s(\mathcal{G}_4))) \\ &+ y_{\mathcal{G}_1} \log(D_{pixel}(\mathcal{G}_3)) \\ &+ y_{\mathcal{G}_1} \log(D_{pixel}(\mathcal{G}_4))], \end{aligned} \quad (7)$$

from which the learned calibrator is expected to learn knowledge in source and target domain and satisfies (2). The total training loss of our data calibrator can be divided into two parts. When the calibrator tries to fool domain discriminators to treat $\mathcal{G}_3$ as $\mathcal{G}_1$, the calibrator tends to approximate the identity mapping. In contrast, when the calibrator tries to fool domain discriminators to treat $\mathcal{G}_4$ as $\mathcal{G}_1$, the calibrator is to calibrate target domain images to mitigate the domain shift.

The ResNet generator [15] is used as the architecture of the calibrator for digits and GTA5 to CityScapes experiments. It consists of downsampling layers, upsampling layers and skip connections, as shown in Figure 4. It is noted that the performance does not simply get better when the calibrator network gets larger. Also, reducing the width can improve training as it is believed that it prevents the data calibrator from overfitting when the training data is not sufficient. Additionally, the output of calibrator is constrained by pre-defined L_∞ norm, which is shown to play an important role in GTA5 to CityScapes adaptation. We will give a more detailed discussion about this constrain in Section 5.

4. Evaluation and Results

In this section, we evaluate our method under unsupervised domain adaptation setting on digits and driving scene semantic segmentation tasks.

Digits We evaluate our method on three commonly used digits datasets: MNIST [17], USPS, and SVHN [23]. We use the same data processing and LeNet architecture as Hoffman *et al*. [12] and perform three unsupervised domain

Method	MNIST to USPS	USPS to MNIST	SVHN to MNIST	Average Acc.
ADDN [37]	90.1	95.2	80.1	88.5
CoGAN [18]	91.2	89.1	-	-
SBADA [28]	**97.6**	95.0	76.1	89.6
CYCADA [12]	95.6	96.5	90.4	94.2
CDAN [20]	95.6	98.0	89.2	94.3
PFA [3]	95.0	-	93.9	-
MSTN [40]	92.9	97.6	93.3	94.6
MCD [30]	93.8	95.7	95.8	95.1
Ours	95.6	97.1	97.1	96.6
CyCleGAN+Ours	97.1	**98.3**	**97.5**	**97.6**

Table 1. **Results on digits datasets for unsupervised domain adaptation**. Our method achieves state-of-the-art performance without using stylized source images. Our method can be further improved by using stylized source images.

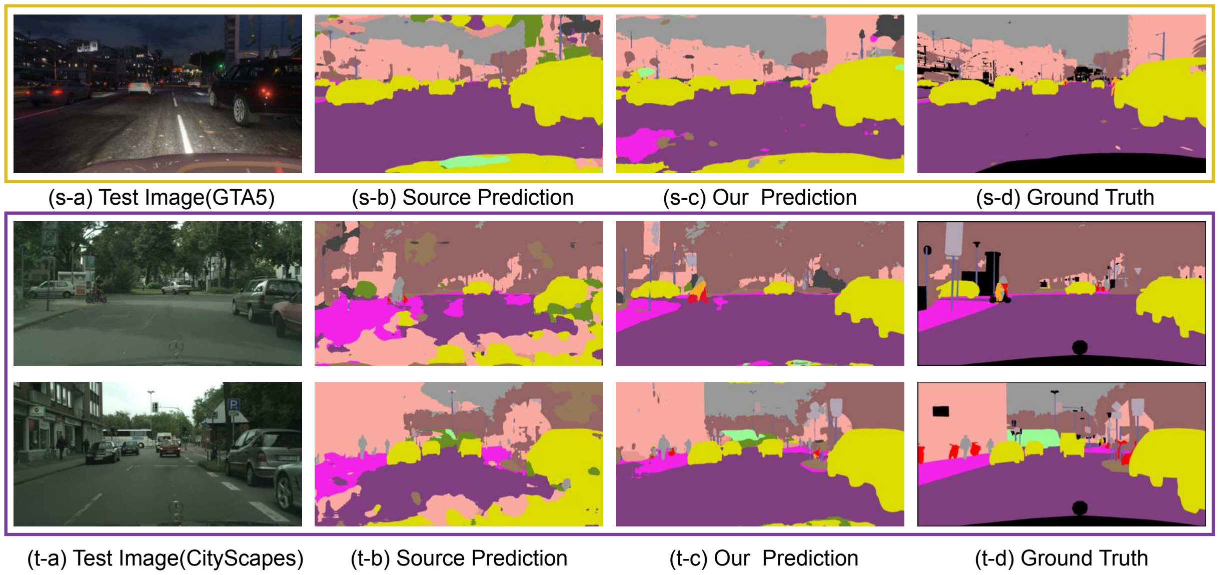

Figure 5. Semantic Segmentation results for GTA5 to CityScapes. (s-a) Test images from GTA5. (s-b) Predictions from the model trained in GTA5. (s-c) Our prediction. (s-d) Ground truth annotations for test images. (t-a) Test images from CityScapes. (t-b) Predictions from the model trained in GTA5. (t-c) Predictions from our method. (t-d) Ground truth annotations for test images.

adaptation tasks: USPS to MNIST, MNIST to USPS and SVHN to MNIST. We report our results of using unstylized source images and stylized source images produced by CycleGAN [46] respectively.

GTA5 to CityScapes GTA5 [27] is a synthetic driving scene dataset and CityScapes [4] is a real world driving scene dataset. The GTA5 dataset has 24966 densely labeled RGB images of size 1914×1052, which contains 19 common classes with CityScapes, as we included in Table 2. The CityScapes dataset contains 5000 densely labeled RGB images of size 2040×1016 from 27 cities. In this work, we use DRN-26 [43] as the source classifier. We use the released DRN-26 model from CyCADA [12] as our source classifier, which is trained in stylized GTA5 images.

All components are implemented using Pytorch. For digits experiments, source classifiers and other components are trained with the Adam optimizer with learning rate 1e-4. We use batches of 128 samples from each domain and the images are zero-centered and rescaled to $[-1, 1]$. For GTA5 to CityScapes experiments, we use Adam optimizer with learning rate 1e-4 with batch size 6. We use same LeNet architecture as CyCADA for all digits experiments and DRN-26 [43] for GTA5 to CityScapes task. Our best results are obtained within 50 epochs for digits and within 10 epochs for GTA5 to CityScapes.

Details about other components such as architecture of the data calibrator and domain discriminators can be found at Appendix.

4.1. Digits Experiments

As we show in Figure 3, the learnt representation of source classifier is not as bad as we thought. To prove that, we show that without training a new classifier or using stylized source images produced by GANs, we can just use the

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorbike	bicycle	**mIoU**	**fwIoU**	**Pixel acc.**
Source only	42.7	26.3	51.7	5.5	6.8	13.8	23.6	6.9	75.5	11.5	36.8	49.3	0.9	46.7	3.4	5.0	0.0	5.0	1.4	21.7	47.4	62.5
CyCADA	79.1	33.1	77.9	23.4	17.3	32.1	33.3	31.8	81.5	26.7	69.0	62.8	14.7	74.5	20.9	25.6	6.9	18.8	20.4	39.5	72.4	82.3
Ours	**83.5**	**35.2**	**79.9**	**24.6**	16.2	**32.8**	33.1	31.8	**81.7**	**29.2**	66.3	**63.0**	14.3	**81.8**	**21.0**	**26.5**	**8.5**	16.7	**24.0**	**40.5**	**75.1**	**84.0**
Target	97.3	79.8	88.6	32.5	48.2	56.3	63.6	73.3	89.0	58.9	93.0	78.2	55.2	92.2	45.0	67.3	39.6	49.9	73.6	67.4	89.6	94.3

Table 2. **Adaptation between GTA5 and CityScapes**. Source only shows results of DRN-26 [43] trained in GTA5 and tested in CityScapes. Target only shows results of DRN-26 trained in CityScapes and tested in CityScapes. Our method outperforms CyCADA in mean IoU, freqency weighted IoU and pixel accuracy. In particular, our frequency weighted IoU is 2.7% better than CyCADA.

source classifier trained in the source domain and train a data calibrator to modify the images to fit the source classifier's representation. As we show in Table 1, using data calibrator alone can outperform previous methods in average accuracy. For difficult task such as SVHN to MNIST, we can further boost our performance by using stylized source images [46] as source domain, resulting in 7% performance improvement compared to CyCADA, another method that leverages stylized source images for unsupervised domain adaptation.

4.2. Performance Trade-off Among Domains

As we discuss in Section 1, existing methods omit to show the trade-off between source domain performance and target domain performance. In this subsection, we show that many existing methods have poor source and target domain performance trade-off. We use the released code from CyCADA [12],ADDA [37] and MCD [30], follow their setting and train their adapted models to get similar reported target domain performance. We then test their adapted model on the source domain and target domain, report the performance before domain adaptation and after domain adaptation. We observe from Figure 2 that, while ADDA has close performance at USPS to MNIST as ours in the target domain, its source domain performance is 5% lower than ours. CyCADA has a lot higher target domain performance compared to ADDA, however, it sacrifices source domain performance significantly. MCD is better than the other two in performance trade-off, but it uses a baseline that has over-parameterized fully connected layers and does not converge well when we replace their backbone with the same LeNet architecture other approaches and ours use. While our method can be further improved by using GAN generated images as source domain, using the data calibrator alone without stylized images can already surpass these methods in both source domain performance and target domain performance as indicated by Figure 2.

4.3. GTA5 to Cityscapes

GTA5 to Cityscapes is a unsupervised domain adaptation task that is closer to real world setting. Compared to classification task, segmentation task is more challenging because that finer domain adaptation methods are required to mitigate domain shift in pixel levels.

As shown in Table 1, our method has better results in all three commonly used metrics such as mIoU, fwIoU, and pixel accuracy. In particular, our fwIoU is 2.7% better than CyCADA. In Figure 5, we visualize our semantic segmentation results. From (s-b) to two rows at (t-b), we observe the performance degradation brought by the domain shift. (s-c) and (t-c) shows the segmentation results produced by our method. Our method largely mitigates the performance degradation in target domain as well as maintaining source domain performance. Because we improve the accuracy of cars by a large margin, the visualization for cars are quite close to the ground truth annotations.

5. Discussion

5.1. Fourier Perspective

We use Fast Fourier Transform(FFT) to analyze images before and after adding calibration. It can be seen in Figure 6 that the high frequency information is decreased after images are added with the output of our data calibrator. High frequency information is often related to textures that varies significantly across domains. Yin *et al.* [42] demonstrates that naturally trained models are biased towards high frequency information, which makes models suffer from high frequency noise. Our method might help remove these high frequency information from images thus mitigating the domain shift problem.

5.2. Connection to Adversarial Attack

Compared to other methods that train classifiers to adapt to target domains, in our domain adaptation framework, once trained in the source domain, the source classifier is not updated and we fully rely on the representation learnt in the source domain to perform tasks in the target domain. Thus the additive calibration produced by our data calibrator needs to figure out how to transform target domain images to a form that better fits the source classifier's representation.

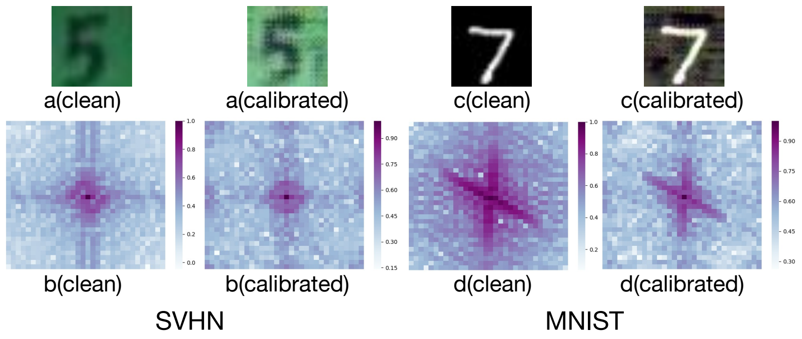

Figure 6. **Images from SVHN to MNIST adaptation** Images before and after being calibrated and their view in the frequency domain. The appearance of images are not changed much unlike what style transfer GANs do. In frequency domain, high frequency information is reduced.

But what does it mean by modifying target domain images to better fit the source classifier's representation? We first hypothesize that there are two candidate explanations of what the data calibrator does: (1) the data calibrator acts as a style-transfer GAN that converts the style of target domain images to source domain images's thus achieve domain adaptation. (2) the data calibrator learns to manipulate non-robust features that are useful to neural networks but are intriguing to human [14]. Our data calibrator might learn to suppress these non-robust features thus mitigate the issue brought by the domain shift.

As can be observed from Figure 6, the images modified by our calibrator do not change their appearance in the way the style transfer GAN usually does. We also follow the convention of adversarial attack [9] to limit L_∞ of the calibration and provide the plot in Appendix. Our best result in Table 2 is obtained by limiting the L_∞ of calibration to 0.01, so small that a human might not be able to tell. Essentially, our data calibrator is trained to produce a perturbation that fools the domain discriminators with human imperceivbale perturbation, which is very similar to the behavior of adversarial attacks [33, 9]. In summary, we believe the calibrator learns to either suppress non-robust features or learn to perform an unusual operation: style transferring non-robust features. Our result suggests that there is a potential connection between adversarial attack and domain adaptation and this should be interesting to both research communities.

5.3. Calibrator for Deployment

As we discuss in Section 1, one of the limitations of existing domain adaptation methods is the lack of flexibility. As far as we know, most existing domain adaptation methods will require to update the deployed model when there is a new target domain. However, the deployed model is usually compressed and stored in specialized hardwares thus adapting the deployed models to new domains requires a long, costly process and might not be fast enough for time-sensitive applications.

In contrast, our method does not require updating the deployed model and has greater flexibility when adapting to a new domain is desired. Additionally, the overhead brought by the calibrator is moderate. We tested the number of parameters of the classifier and data calibrator. For digits experiment, the number of parameter of LeNet is 3.1 millions while the data calibrator has 0.18 millions of parameters, only 5.8% compared to the model. For GTA5 to CityScapes experiments, the DRN-26 model has 20.6 millions of parameters while our data calibrator only has 0.05 millions of parameters, only 0.24% compared to the DRN-26 model.

We thereby conclude that the proposed data calibrator is light-weight compared to the deployed model and does not bring too much overhead during deployment.

6. Conclusion

In summary, the proposed method not only achieves state-of -the-art performance in unsupervised domain adaptation for digits classification task and driving scene semantic segmentation task, but also be suitable for deployed models to adapt to new domains without the need to update their weights. This approach provides a feasible solution for online unsupervised domain adaptation. While the community is trying to build a monolithic model that can work across as many domains as possible, the separable approach we propose is also worth investigating.

7. Acknowledgments

This work was partially supported by IIISCT (Institute for interdisciplinary Information Core Technology), National Natural Sciences Foundation of China (No. 31970972 and 11901338) , and Tsinghua University Initiative Scientific Research Program.

References

[1] Andreea Bobu, Eric Tzeng, Judy Hoffman, and Trevor Darrell. Adapting to continuously shifting domains. 2018. 2

[2] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017. 3

[3] Chaoqi Chen, Weiping Xie, Wenbing Huang, Yu Rong, Xinghao Ding, Yue Huang, Tingyang Xu, and Junzhou Huang. Progressive feature alignment for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 627–636, 2019. 6

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 6

[5] J Deng, W Dong, R Socher, LJ Li, K Li, and L ImageNet Fei-Fei. A large-scale hierarchical image database. 2009 ieee conf comput vis pattern recognit, 2009. 2

[6] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014. 1, 3, 4

[7] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 3, 4

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1, 3

[9] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 8

[10] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254. IEEE, 2016. 2

[11] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 2

[12] 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. *arXiv preprint arXiv:1711.03213*, 2017. 1, 3, 5, 6, 7

[13] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016. 1, 3

[14] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019. 8

[15] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 5

[16] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018. 2

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[18] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016. 3, 6

[19] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015. 3

[20] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650, 2018. 1, 6

[21] Xiaolong Ma, Fu-Ming Guo, Wei Niu, Xue Lin, Jian Tang, Kaisheng Ma, Bin Ren, and Yanzhi Wang. Pconv: The missing but desirable sparsity in dnn weight pruning for real-time execution on mobile devices. *arXiv preprint arXiv:1909.05073*, 2019. 2

[22] Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 6670–6680, 2017. 5

[23] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5

[24] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and N Lawrence. Covariate shift and local learning by distribution matching, 2008. 3

[25] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019. 1, 2

[26] Ao Ren, Tianyun Zhang, Shaokai Ye, Jiayu Li, Wenyao Xu, Xuehai Qian, Xue Lin, and Yanzhi Wang. Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 925–938. ACM, 2019. 2

[27] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016. 6

[28] Paolo Russo, Fabio M Carlucci, Tatiana Tommasi, and Barbara Caputo. From source to target and back: symmetric bi-directional adaptive gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8099–8108, 2018. 6

[29] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 1, 3

[30] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018. 1, 6, 7

[31] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 3

[32] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016. 1, 3

[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 8

[34] Zhanhong Tan, Jiebo Song, Xiaolong Ma, Sia-Huat Tan, Hongyang Chen, Yuanqing Miao, Yifu Wu, Shaokai Ye, Yanzhi Wang, Dehui Li, et al. Pcnn: Pattern-based fine-grained regular pruning towards optimizing cnn accelerators. *arXiv preprint arXiv:2002.04997*, 2020. 2

[35] Antonio Torralba, Alexei A Efros, et al. Unbiased look at dataset bias. In *CVPR*, volume 1, page 7. Citeseer, 2011. 1, 2, 3, 5

[36] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015. 3

[37] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017. 1, 3, 4, 6, 7

[38] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 1, 3

[39] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Incremental adversarial domain adaptation for continually changing environments. In *2018 IEEE International conference on robotics and automation (ICRA)*, pages 1–9. IEEE, 2018. 2

[40] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 5419–5428, 2018. 6

[41] Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both? In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2

[42] Dong Yin, Raphael Gontijo Lopes, Jonathon Shlens, Ekin D Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. *arXiv preprint arXiv:1906.08988*, 2019. 7

[43] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017. 6, 7

[44] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018. 2

[45] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017. 2

[46] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 6, 7