Visual Domain Adaptation by Consensus-Based Transfer to Intermediate Domain

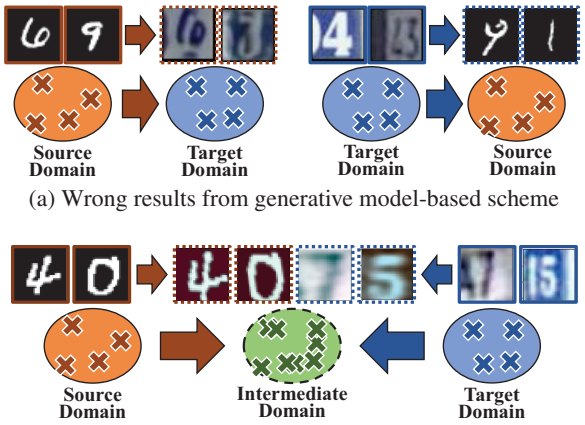
Jongwon Choi, Youngjoon Choi,* Jihoon Kim, Jinyeop Chang,
Ilhwan Kwon, Youngjune Gwon, Seungjai Min
Samsung SDS

Abstract

We describe an unsupervised domain adaptation framework for images by a transform to an abstract intermediate domain and ensemble classifiers seeking a consensus. The intermediate domain can be thought as a latent domain where both the source and target domains can be transferred easily. The proposed framework aligns both domains to the intermediate domain, which greatly improves the adaptation performance when the source and target domains are notably dissimilar. In addition, we propose an ensemble model trained by confusing multiple classifiers and letting them make a consensus alternately to enhance the adaptation performance for ambiguous samples. To estimate the hidden intermediate domain and the unknown labels of the target domain simultaneously, we develop a training algorithm using a double-structured architecture. We validate the proposed framework in hard adaptation scenarios with real-world datasets from simple synthetic domains to complex real-world domains. The proposed algorithm outperforms the previous state-of-the-art algorithms on various environments.

Introduction

With the advent of deep convolutional neural networks (CNN), supervised learning has substantially improved the performance of image classification algorithms. However, supervised learning requires laborious and expensive labeling efforts to prepare the train data to cover all possible test environments. To alleviate this effort and cost of labeling the training data in various environments, Unsupervised Domain Adaptation (UDA) has been introduced to automatically generate the labels of the unlabeled data of new domain by exploiting the knowledge of the labeled data (Saito et al. 2018; Ganin and Lempitsky 2015; Shu et al. 2018; Bousmalis et al. 2017; Ghifary, Kleijn, and Zhang 2014). Conventionally, the labeled data is called by *source domain*, while the unlabeled data is denoted by *target domain*. However, UDA does not necessarily produce satisfactory results when the characteristics of the source and target domains are very distinct from each other. The different characteristics include diverse backgrounds, viewpoints, and image capturing systems.



(a) Wrong results from generative model-based scheme

(b) Aligned samples in intermediate domain

Figure 1: **intermediate domain transfer approach.** (a) The important attributes to classify the digits were distorted by the generative model-based algorithm (Russo et al. 2018) when the source and target domains are far different. (b) To tackle the problem, the proposed framework aligns the two domains simultaneously into a hidden intermediate domain.

The recent UDA approaches for CNN can be categorized by two schemes: domain classifier-based scheme (Ganin and Lempitsky 2015; Shu et al. 2018; Saito et al. 2018) and generative model-based scheme (Bousmalis et al. 2017; Russo et al. 2018; Volpi et al. 2018). The domain classifier-based scheme utilizes the additional classifier to guess the domain of the input samples. Then, by training the feature generator to confuse the domain classifier, the features of the target domain become similar to the features of the source domain. The generative model-based scheme utilizes a generative model to change the characteristics of the source images to be similar with the target images. Then, a CNN is trained by the target-like images changed by the generative model. The previous works are described in the following related work section in detail.

However, both schemes have shown limited performance

*Y. Choi was with Samsung SDS while working on this paper.

when the source and target domains are significantly different. When the generative model-based scheme learns the style transfer between the two distinct domains, the essential information to classify the target domain can be lost, as shown in Fig. 1 (a). Likewise, the domain classifier-based scheme cannot adapt the distinct domains because its assumption of the overlapped feature distribution is broken when the two domains are too different. However, to solve the hard scenarios adapting simple synthetic domains to complex real-world domains, the adaptation between the two distinct domains is significantly important.

To tackle the limitation, we propose a framework using a new scheme of intermediate domain alignment and consensus-based ensemble classifier. In the intermediate domain alignment, the source and target domains are aligned into the intermediate domain without losing critical information inherent in these two domains. The intermediate domain is obtained by cross-domain image adaptors such that the differences between the source and target domains are semantically reduced as shown in Fig. 1 (b). In addition, to classify the ambiguous target samples correctly, we determine their labels according to the consensus of multiple classifiers. To robustly find the intermediate domain even with the missing target labels, we also propose a iterative training method based on a double-structured architecture. The proposed framework is validated by numerous experiments including the real-world datasets, which show the state-of-the-art performance and the large improvement for hard adaptation scenarios.

Related work

Domain Classifier-based UDA: In contrast to the traditional approaches (Ben-David et al. 2010; Wang and Mahadevan 2011; Ghifary, Kleijn, and Zhang 2014) with handcrafted features and kernel-based mappings, the domain classifier-based UDA algorithms merge the feature vectors from a neural network by confusing the domain classifier that distinguishes the source and target domains. Ganin and Lempitsky (2015) proposed a new model containing a feature generator, a label predictor, and a domain classifier, which has a gradient reversal layer to prohibit the feature generator from distinguishing the domains. Following Ganin and Lempitsky (2015), many researchers have utilized the domain classifier for UDA problems (Tzeng et al. 2017; Shu et al. 2018; Saito et al. 2018; Long et al. 2015; Bousmalis et al. 2016; Saito, Ushiku, and Harada 2017). Tzeng *et al*. (2017) combined the feature generator providing similar features from the source and target domains and the classifier trained only by the source domain. Saito *et al*. (2018) proposed a framework which unifies the feature vectors by iteratively minimizing and maximizing the discrepancy of two class predictors.

Generative Model-based UDA: The generative model-based UDA approaches have been proposed recently following the increased interest to the generative adversarial networks (Goodfellow et al. 2014; Zhu et al. 2017). Bousmalis *et al*. (2017) proposed a fundamental architecture for the generative model-based UDA approaches, which showed high applicability into the real-world settings. Sankaranarayanan *et al*. (2018) trained an embedding network and used its embedded vector for the input noises of the generative adversarial network that transforms the source images to be similar to the target images. Hoffman *et al*. (2018) developed cyCADA that considers the pixel-level and feature-level consistency under the cyclic reconstruction scheme. Russo *et al*. (2018) suggested SBADA containing two networks each trained by considering the source and target domains reversely. The proposed algorithm transfers the two domains into the hidden intermediate domain while cyCADA and SBADA transfer the two domains into the other ones respectively. Gong *et al*. (2019) utilized the intermediate domain that interpolates the styles of source and target domains, which is different from the proposed intermediate domain that is an entirely new domain where the source and target domains are simplified simultaneously. By employing the intermediate domain, we can improve the performance when the two domains are notably distinct from each other.

Besides the generative adversarial network, some of the generative model-based UDA approaches have utilized auto-encoding networks (Murez et al. 2018; Volpi et al. 2018). Volpi *et al*. (2018) proposed a scheme to augment the embedded features by an adversarial learning scheme. Murez *et al*. (2018) suggested a framework to estimate a shared embedding domain by using numerous cyclic combinations among the three different domains and the class labels. However, the algorithm needs a lot of augmented data and hyperparameters to control the numerous cyclic combinations, which increases the cost to tune the algorithms.

Methodology

Overall architecture

Fig. 2 shows the overall architecture of the proposed framework. In front of a classification model, there are two *cross-domain image adaptors* (A_s, A_t) for the source and target domains, respectively. After A_s and A_t, there are two networks of a forward network (G, C_1,...,C_{N_c}) and an inverse network (G^-, C_1^-,...,$C_{N_c}^-$). When we use only the conventional *forward network*, we found that only A_t is updated and just the target domain is aligned into the source domain. As shown by the gray and yellow lines in Fig. 3, the transform parameters in A_s do not change at all without the inverse network. This happens due to the unbalanced training where A_s can be trained by the strong supervision loss using source labels, while A_t cannot. Hence, we resolve the problem by adding the inverse network, which can give the same types of losses to both of A_s and A_t. Through updating A_s and A_t respectively by the inverse and forward networks, the balanced cross-domain image adaptors can be obtained, which results in the intended intermediate domain.

The model including the forward and inverse networks is defined by a double-structured architecture. The forward network is composed of one feature generator (G) and N_c classifiers ($C_1, ..., C_{N_c}$). Similarly, the inverse network has one feature generator (G^-), and N_c classifiers ($C_1^-, ..., C_{N_c}^-$). The forward and inverse networks have same layer composition, but the inverse network is trained to adapt the target domain to the source domain by using the class la-

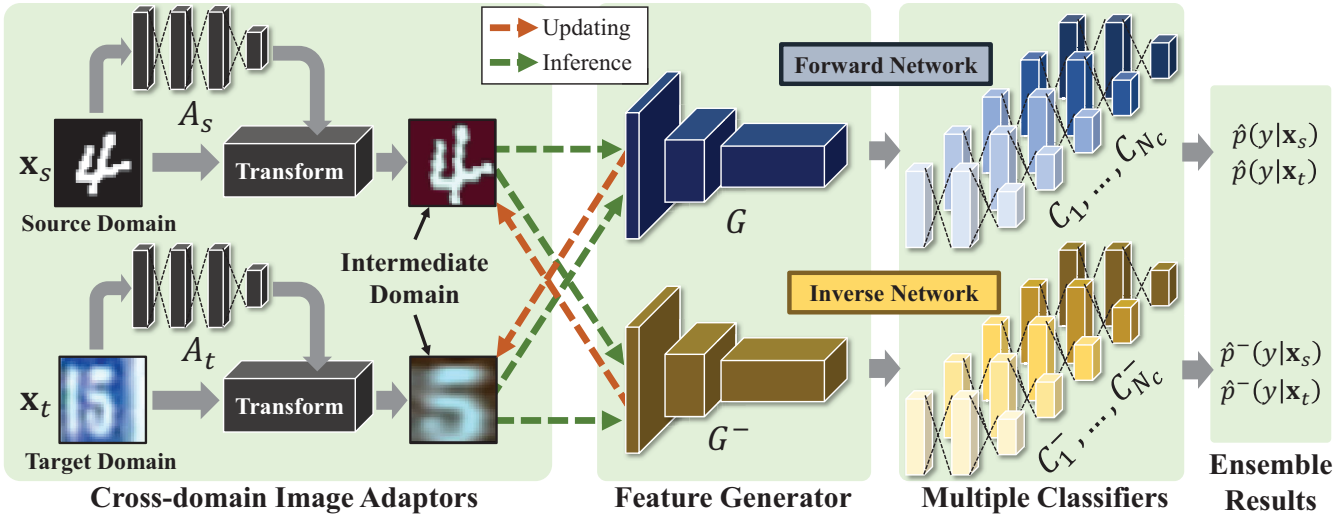


Figure 2: Overall network architecture

bels of the target domain predicted by the forward network. All the classifiers have same structures including last softmax layers but are initialized by different random weights.

When image samples are given, A_s and A_t get the images from the source and target domains, respectively, and the images are aligned to the intermediate domain. The aligned images are fed into G or G^-, and then the generated features are classified by the following $\{C_1, ..., C_{N_c}\}$ or $\{C_1^-, ..., C_{N_c}^-\}$. The notations $(A_s, A_t, G, G^-, C_1, C_1^-, \cdots)$ for the network architecture also represent the weight parameters in the corresponding network components. For simplicity, we define $\theta_k = \{A_s, A_t, G, C_k\}$ for forward network and $\theta_k^- = \{A_s, A_t, G^-, C_k^-\}$ for inverse network $(k \in \{1, \cdots, N_c\})$. Then, for an input sample $\mathbf{x}$, we can define the probability vectors predicted from C_k and C_k^- by $p(\mathbf{y}|\mathbf{x}, \theta_k)$ and $p(\mathbf{y}|\mathbf{x}, \theta_k^-)$, respectively.

Then, the proposed framework should be trained to find transformation parameters of A_s and A_t and classify the target samples well by G and $C_1, ..., C_{N_c}$. When the features are overlapped well, the results from $C_1, ..., C_{N_c}$ would make a consensus on the class label predicted for a specific target image. Thus, we train the network by considering the labels of the source domain and the consensus of multiple classifiers simultaneously.

Cross-domain image adaptors

When the two domains are augmented properly, the performance of the domain adaptation becomes much improved (Saito, Ushiku, and Harada 2017), and the design of the cross-domain image adaptors is motivated from the conventional augmentation methods. We utilize two transformers for each of A_s and A_t: color transformer and spatial transformer. The color transformer works for the augmentations of color styles such as contrast adjusting and intensity inverting, and the spatial transformer is used for the augmentations of camera viewpoints such as cropping and rotating.

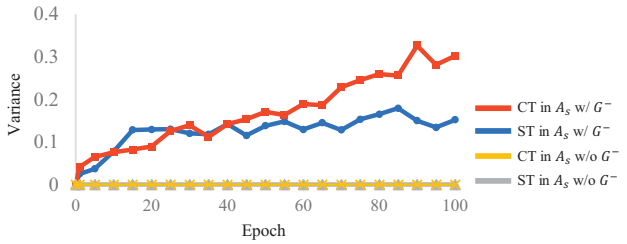


Figure 3: Necessity of inverse network

The color transformer is a small CNN fed by the input images to give a color transform matrix $\mathbf{T}_c \in \mathtt{R}^{3\times 9}$ as its output. Based on the resulted $\mathbf{T}_c \in \mathtt{R}^{3\times 9}$, we transform the pixel-wise colors of input images in a quadratic form by:

$$[r_i', g_i', b_i']^T = \mathbf{T}_c[r_i, g_i, b_i, r_i^2, g_i^2, b_i^2, r_i g_i, r_i b_i, g_i b_i]^T, \quad (1)$$

where $[r_i, g_i, b_i]$ is the original *rgb* color vector for i-th pixel and $[r_i', g_i', b_i']$ means the color vector transformed by the color transformer for i-th pixel. Using $\mathbf{T}_c$ from the color transformer, the colors of the entire pixels contained in the input image are transformed. Since the quadratic form of a color vector $([r_i, g_i, b_i, r_i^2, g_i^2, b_i^2, r_i g_i, r_i b_i, g_i b_i])$ is a constant variable and the color transformation is a kind of linear calculations, the color transformer can be trained by an end-to-end scheme.

Similarly, the spatial transformer contains a small CNN that gets the color-transformed image as an input and results in an affine transform matrix $\mathbf{T}_s \in \mathtt{R}^{2\times 3}$. According to the estimated $\mathbf{T}_s \in \mathtt{R}^{2\times 3}$, the intensity of the color-transformed images at the position (x, y) is mapped to the different location (x', y') as:

$$[x', y']^T = \mathbf{T}_s[x, y, 1]^T. \quad (2)$$

Then, we can estimate the pixel intensity of the spatial transformed image at (x', y') by a bilinear interpolation at a sub-pixel position $\mathbf{T}_s^{-1}[x', y']$ of the color transformed image.

Like the color transformer, the spatial transformer can be trained in an end-to-end scheme because the affine transform mapping and the bilinear interpolation are a combination of linear calculations. The detail description of the spatial transformer can be referred by the spatial transform network (Jaderberg et al. 2015).

Training loss terms

The proposed network should be trained by utilizing two clues: the labels of the source samples and the consensus of the predicted labels for the target samples. Based on the two feasible information, we build three loss terms: *supervised loss*, *consensus loss*, and *fooling loss*.

The *supervised loss* is a conventional cross-entropy loss to train the given labels for the sample images in a supervised learning manner. Thus, the supervised loss is defined as:

$$L_s(\mathbf{X}, \mathbf{Y}, \theta) = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_o^{(i)^T} \log p(\mathbf{y}|\mathbf{x}^{(i)}, \theta), \tag{3}$$

where $\mathbf{x}^{(i)}$ and $\mathbf{y}_o^{(i)}$ mean the i-th sample image and its corresponding label which are contained in $\{\mathbf{X}, \mathbf{Y}\}$, respectively, N is the number of sample images in $\mathbf{X}$, and θ is one of θ_k or θ_k^- for $k \in \{1, \cdots, N_c\}$. By applying the supervised loss to train the network, the result from a specific classifier is driven to the given labels for the samples.

The second term is the *consensus loss*. The results from the multiple classifiers should make a consensus by predicting the same class label for the specific sample if we find the ideal feature generator. Thus, we design the consensus loss term to drive the classifiers to give the same probability estimated by their ensemble model. The ensemble results of the forward network ($\widehat{p}(\mathbf{y}|\mathbf{x}^{(i)})$) and the inverse network ($\widehat{p^-}(\mathbf{y}|\mathbf{x}^{(i)})$) are designed by averaging the results of all the classifiers as:

$$\begin{aligned} \widehat{p}(\mathbf{y}|\mathbf{x}^{(i)}) &= \frac{1}{N_c}\sum_{k=1}^{N_c} p(\mathbf{y}|\mathbf{x}^{(i)}, \theta_k) \\ \widehat{p^-}(\mathbf{y}|\mathbf{x}^{(i)}) &= \frac{1}{N_c}\sum_{k=1}^{N_c} p(\mathbf{y}|\mathbf{x}^{(i)}, \theta_k^-). \end{aligned} \tag{4}$$

Based on the ensemble model, the consensus loss terms for the forward network (L_c) and the inverse network (L_c^-) are respectively designed as following:

$$\begin{aligned} L_c(\mathbf{X}) &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N_c} \widehat{p}(\mathbf{y}|\mathbf{x}^{(i)}) \log p(\mathbf{y}|\mathbf{x}^{(i)}, \theta_k) \\ L_c^-(\mathbf{X}) &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N_c} \widehat{p^-}(\mathbf{y}|\mathbf{x}^{(i)}) \log p(\mathbf{y}|\mathbf{x}^{(i)}, \theta_k^-). \end{aligned} \tag{5}$$

Thus, the consensus loss drives the multiple classifiers to have the same results as their ensemble results.

The final term is the *fooling loss*. When we update the networks only by the supervised loss and the consensus loss, the multiple classifiers converge early to make a consensus before the feature generator and the cross-domain image adaptors generalize the two domains. To prevent the problem, the fooling loss is invented to make the convergence slow.

Algorithm 1: Sequential training method

Input: $\mathbf{X}_s$, $\mathbf{Y}_s$, $\mathbf{X}_t$, $\{G, C_1,..., C_{N_c}\}$, $\{G^-, C_1^-,..., C_{N_c}^-\}$
while *not at the maximum iteration* **do**
  Update G, C_1,...,C_{N_c} using Eq. 7
  Update C_1,...,C_{N_c} using Eq. 8
  Update A_t, G using Eq. 9
  Estimate $\mathbf{Y}_t^*$ using Eq. 10
  Update G^-, C_1^{inv},...,$C_{N_c}^-$ using Eq. 11
  Update C_1^-,...,$C_{N_c}^-$ using Eq. 12
  Update G^- using Eq. 13
  Update A_s using Eq. 14
end

Thus, the fooling loss should let a specific classifier predict ambiguously, which is derived by an inverse of a marginal entropy as follows:

$$L_f(\mathbf{X}, \theta) = \frac{1}{N}\sum_{i=1}^{N} p(\mathbf{y}|\mathbf{x}^{(i)}, \theta) \log p(\mathbf{y}|\mathbf{x}^{(i)}, \theta). \tag{6}$$

Training method

As summarized in Algorithm **??**, the proposed loss terms are applied sequentially by seven steps at every iteration. The first three steps are performed to update A_t and the forward network, while A_s and the inverse network are trained in the last four steps. By adopting the multiple losses sequentially, we can ignore the scale factors among the loss terms. Furthermore, the amount of memory usage can be reduced because only a part of the entire model is updated in each step. All the updating steps are performed by Adam optimizer (Kingma and Ba 2015) and iterated until the predefined value of maximum iterations.

We have a source image $\mathbf{x}_s^{(i)}$ and its corresponding scalar label $y_s^{(i)}$, which are i-th labeled sample in a source domain $\{\mathbf{X}_s, \mathbf{Y}_s\}$. $\mathbf{x}_t^{(i)}$ represents i-th target image composing a target domain $\{\mathbf{X}_t\}$ without any corresponding label. We can represent the input flow for every step as following; step 1: $C_k(G(A_s(\mathbf{X}_s)))$, Step 2, 3: $C_k(G(A_t(\mathbf{X}_t)))$, Step 4: $C_k^-(G^-(A_t(\mathbf{X}_t)))$, Step 5, 6, 7: $C_k^-(G^-(A_s(\mathbf{X}_s)))$.

Step 1: In the first step, G and $\{C_1,..., C_{N_c}\}$ are updated by the supervised loss L_s, so the first updating step can be represented as:

$$\arg\min_{G, C_1, \ldots, C_{N_c}} \sum_{k=1}^{N_c} L_s(\mathbf{X}_s, \mathbf{Y}_s, \theta_k). \tag{7}$$

Through the first step, the forward network are driven to predict correct labels for the source images. Thus, the semantic information to classify the data sample preserves in the network even with the fooling and the consensus losses.

Step 2: The second step leads $\{C_1,...,C_{N_c}\}$ to give ambiguous prediction for the target images by using L_f. We just consider the predictions for the target images in Step 2,

which can be represented as:

$$\arg \min_{C_1,...,C_{N_c}} \sum_{k=1}^{N_c} L_f(\mathbf{X}_t, \theta_k). \quad (8)$$

In this step, we only update the classifiers because the fooling loss targets on confusing the predictions of classifiers.

Step 3: The third step is applied to unify the latent features of two domains based on the consensus of the multiple classifiers. Meanwhile, A_t is trained in this step. Thus, the update at Step 3 can be derived as:

$$\arg \min_{A_t,G} L_c(\mathbf{X}_t). \quad (9)$$

Then, by driving the classifiers to make a consensus, A_t and G are trained for the hidden intermediate domain and the unified latent feature, respectively.

After the third step, to train the inverse network, the predicted labels of the target images are obtained by the updated forward network as:

$$\widehat{\mathbf{y}}_t^{(i)} = \widehat{p}(\mathbf{y}|\mathbf{x}_t^{(i)}), \quad (10)$$

and a set containing all the predicted target labels $\widehat{\mathbf{y}}_t^{(i)}$ is denoted by $\mathbf{Y}_t^*$. In the remaining steps, the inverse network is updated similarly to the first three training steps for the forward network, while considering the source and target images reversely by using $\mathbf{Y}_t^*$ as the source labels. Thus, the inverse network considers $\{\mathbf{X}_t, \mathbf{Y}_t^*\}$ and $\{\mathbf{X}_s\}$ as its source and target domains, respectively.

Step 4: This step updates G^- and $\{C_1^-,...,C_{N_c}^-\}$ in a supervised manner as:

$$\arg \min_{G^-,C_1^-,...,C_{N_c}^-} \sum_{k=1}^{N_c} L_s(\mathbf{X}_t, \mathbf{Y}_t^*, \theta_k^-). \quad (11)$$

Step 5: Then, the fifth step confuses $\{C_1^-,...,C_{N_c}^-\}$ by the fooling loss as following:

$$\arg \min_{C_1^-,...,C_{N_c}^-} \sum_{k=1}^{N_c} L_f(\mathbf{X}_s, \theta_k^-). \quad (12)$$

Step 6: The sixth step is a little bit different from Step 3 where the image adaptor and the feature generator of the forward network are updated simultaneously. However, in Step 6, only the feature generator is updated by the consensus loss as represented by:

$$\arg \min_{G^-} L_c^-(\mathbf{X}_s). \quad (13)$$

Step 7: At the final step, we train A_s to let $\{C_1^-,...,C_{N_c}^-\}$ make a consensus to $\mathbf{Y}_s$. Thus, the final step is given as:

$$\arg \min_{A_s} \sum_{k=1}^{N_c} L_s(\mathbf{X}_s, \mathbf{Y}_s, \theta_k^-). \quad (14)$$

After the training, the inverse network is removed, and then we can predict the labels of the target images by Eq. 10 through the cross-domain image adaptor for the target domain and the forward network. Even though the target pseudo-labels are noisy, the noisy labels are used in Step 4-6 training the inverse network (G^-, C_1^-,...,$C_{N_c}^-$) that is removed after the training. Thus, the noisy target pseudo-labels do not affect the adaptation performance.

Table 1: Comparison on algorithm complexity

	Acc.	Sec/Epoch	#(Epoch)	#(Loss Param.)
SBADA	61.1%	76.73 s	500	6
Proposed	78.5%	1.12 s	100	0

Table 2: Quantitative results for variant algorithms

Source→Target	**Distinct Domains** MNIST → SVHN	**Similar Domains** USPS → MNIST
Source Only	47.8%	63.4%
DANN	35.7%	73.0%
DRCN	40.1%	73.7%
SBADA	61.1%	95.0%
IEDA-noIAN	63.5% (– 15.0%)	87.4% (– 9.9%)
IEDA-noSTN	63.0% (– 15.5%)	85.1% (– 12.2%)
IEDA-noCTN	72.9% (– 5.6%)	97.2% (– 0.3%)
IEDA-single	69.7% (– 8.8%)	89.1% (– 8.2%)
IEDA-3Step	73.6% (– 4.9%)	97.3% (– 0.2%)
IEDA-1C	72.4% (– 6.1%)	75.1% (– 22.1%)
IEDA-2C	71.8% (– 6.7%)	83.0% (– 14.3%)
IEDA-noFooling	64.9% (– 13.6%)	94.1% (– 3.2%)
IEDA	**78.5%**	**97.5%**

Experimental results

Implementation

The number of multiple classifiers was set to $N_c = 5$. The learning rates for the feature generator and the classifiers were set to 0.0002, while the learning rates of the relatively small image adaptors are set to $2e^{-6}$. The decay weight loss was added to all the optimization steps with a loss scale of 0.0005. The detail network architectures are given in Appendix A of the supplementary document[1].

We used PyTorch (Paszke et al. 2017) to implement the proposed framework. The computational environment had an Intel i7-8700 CPU @ 3.20GHz, 32GB RAM, and an NVIDIA GTX 1080 Ti GPU. For our target task where a simple domain is adapted to a complex domain, the previous state-of-the-art algorithms are complicated and require much training time. In Table 1, we present the complexity of the proposed framework for MNIST→SVHN that is one of the challenging scenarios. IEDA is simpler and much faster than SBADA (Russo et al. 2018) that is the state-of-the-art algorithm on the scenario, even with better performance.

For validating the performance of the proposed algorithm, we compare it with the state-of-the-art algorithms including MMD (Long et al. 2015), DANN (Ganin and Lempitsky 2015), DRCN (Ghifary et al. 2016), DSN (Bousmalis et al. 2016), CoGAN (Liu and Tuzel 2016), ATT (Saito, Ushiku, and Harada 2017), MCDDA (Saito et al. 2018), SBADA (Russo et al. 2018), SAFN (Xu et al. 2019), and DANSVD (Shkodrani, Hofmann, and Gavves 2018). All their results are obtained using the public results or their original programs.

Ablation tests

To analyze the proposed framework of Intermediate domain transfer and Ensemble classifier for Domain Adap-

[1]https://sites.google.com/site/jwchoivision/

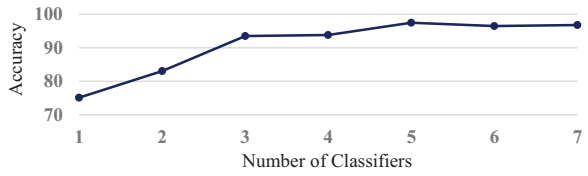


Figure 4: Performance and the number of classifiers

Table 3: Quantitative comparison on classification datasets

Source Target	MNIST SVHN	SVHN MNIST	MNIST USPS	USPS MNIST	STL CIFAR	CIFAR STL
Source Only	47.8%	67.1%	76.7%	63.4%	55.8%	72.1%
MMD	-	71.1%	-	-	-	-
DANN	35.7%	71.1%	77.1%	73.0%	-	-
DRCN	-	82.0%	91.8%	73.7%	58.7%	66.4%
DSN	-	82.7%	91.3%	-	-	-
CoGAN	-	-	91.2%	89.1%	-	-
ATT	52.8%	86.0%	-	-	-	-
MCDDA	28.7%	96.2%	94.2%	94.1%	-	-
SBADA	61.1%	76.1%	**97.6%**	95.0%	-	-
DANSVD	-	97.8%	-	-	-	61.5%
IEDA	**78.5%**	**98.9%**	95.0%	**97.5%**	**62.3%**	**78.3%**

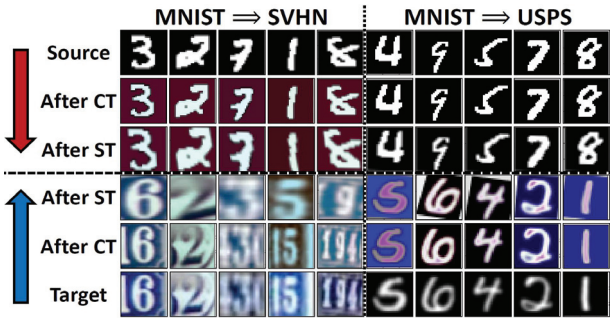


Figure 5: Transformed images by the image adaptors

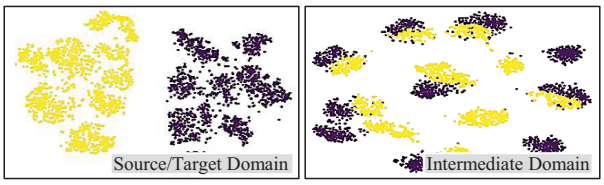


Figure 6: Feature distribution by t-SNE

tation (IEDA), we additionally implement several variants of IEDA: *IEDA-noIAN*, *IEDA-noSTN*, *IEDA-noCTN*, *IEDA-noINV*, *IEDA-3Step*, *IEDA-1C*, *IEDA-2C*, and *IEDA-noFooling*. With digit datasets of MNIST (LeCun et al. 1998), SVHN (Netzer et al. 2011), and USPS (LeCun et al. 1998), we built two scenarios of distinct domains (MNIST→SVHN) and similar domains (USPS→MNIST) to compare the effectiveness of the framework components for the two different cases. Especially, MNIST→SVHN is a challenging task because SVHN has a much higher diversity of color change, clutter, and affine transformation than MNIST. Table 2 compares the variants quantitatively.

The first three variants (*IEDA-noIAN*, *IEDA-noSTN*, *IEDA-noCTN*) are implemented to validate the effectiveness of the intermediate domain. In *IEDA-noIAN*, the two cross-domain image adaptors were entirely removed from IEDA. In *IEDA-noSTN*, the spatial transformers in the image adaptors were excluded, while the color transformer remains. On the contrary, the image adaptors in *IEDA-noCTN* contain the spatial transformers only. The intermediate domain scheme was powerful when the two domains are distinct to each other, so the improved performance for the distinct domains (MNIST→SVHN) was much larger than the similar domains (USPS→MNIST) from the results of *IEDA-noIAN*. In *IEDA-noCTN*, the CTN shows little effectiveness for USPS→MNIST because both datasets are gray-level.

The next two variants (*IEDA-single*, *IEDA-3Step*) validate the strength of the double-structured architecture. *IEDA-single* only utilizes the forward network. Then, A_s cannot be trained, which results in no image transfer of the source images. Likewise, *IEDA-3Step* ignores the inverse network, but A_s is trained in Step 1 by the supervised loss (Eq. 7). From the result of *IEDA-single*, when the image transfer of the source images is neglected, the performance drops for both of the two scenarios. With *IEDA-3Step*, the intermediate domain is biased to the source domain, which results in the performance reduction for the distinct domains.

The last three variants (*IEDA-1C*, *IEDA-2C*, *IEDA-noFooling*) explain the necessity of the consensus-based multiple classifiers. The number of multiple classifiers (N_c) were set to 1 and 2 in *IEDA-1C* and *IEDA-2C*, respectively. *IEDA-noFooling* ignores the training steps of Step 2 and Step 5 where the fooling loss is used. At first, from the results of *IEDA-1C* and *IEDA-2C*, the small number of classifiers works critical for the scenario of similar domains. Since the proposed framework cannot adapt the ambiguous samples yet for the distinct domains, the performance reduces by the limited amount with the small number of classifiers on MNIST→SVHN. In contrast, the performance decreased much for the similar domains where the ambiguous samples are classified well by the proposed framework. Since the number of classifiers is crucial for the similar domains, we perform the additional ablation study for the number of classifiers on USPS→MNIST. As shown in Fig. 4, the performance increases monotonically until being saturated at 5 classifiers. From the results of *IEDA-noFooling*, we can find that the fooling loss is essential to adapt two distinct domains, while the similar domains can be adapted reasonably without the fooling loss.

Experiments on digit datasets

To validate the proposed framework, we perform the experiments using various combinations of three conventional digit recognition datasets: MNIST (LeCun et al. 1998), SVHN (Netzer et al. 2011), and USPS (LeCun et al. 1998). All the quantitative results for the digit datasets are given at Table 3. In addition, the transformed images for the intermediate domain are represented in Fig. 5.

MNIST ↔ SVHN: MNIST dataset contains hand-written digit images, while SVHN is a dataset captured from real-world environments such as traffic signs and signboards.

Contrary to MNIST dataset where the digits are normalized to be located at the centers, the digits of SVHN dataset are positioned randomly. In contrast to the previous works, the proposed framework showed state-of-the-art adaptation accuracy (78.5%) when adapting MNIST to SVHN. Meanwhile, the proposed framework also showed the state-of-the-art performance even for the reversal adaptation from SVHN to MNIST. The qualitative results show that, when we adapt MNIST to SVHN for their intermediate domain, MNIST is painted by some color to cover the color variation of SVHN, while SVHN is translated and zoomed to place the digits at the center like MNIST.

USPS $\leftrightarrow$ MNIST: Like MNIST dataset, USPS dataset consists of numerous hand-written digit images, and the proposed algorithm also outperformed the state-of-the-art algorithms for the datasets. As shown in the qualitative results, the major difference between USPS and MNIST is the blurriness of the digits. However, the two domains become similar in the intermediate domain by blurring the clear MNIST, while the blurred USPS is clarified. Even though the color variation makes the two domains visually different, the color information does not affect the model adaptation because the two initial domains contain only the gray-level images. We estimate t-SNE of the sample features obtained by G as shown in Fig. 6. Even though the intermediate images look visually different in USPS$\rightarrow$MNIST, the intermediate images from USPS (Yellow points) and MNIST (Purple points) become much closer in the feature space.

Experiments on object datasets

We evaluate the proposed framework on object classification datasets: CIFAR10 (Krizhevsky and Hinton 2009), STL10 (Coates, Ng, and Lee 2011), and VisDA Dataset (Peng et al. 2017). CIFAR10 and STL10 datasets are the conventional object classification datasets, and they have 9 overlapped class labels as referred to (Ghifary et al. 2016). As shown in the last two columns of Table 3, even with the network resulting in a low accuracy when only the source samples are used for training (*Source Only*), the proposed framework showed largely improved performance. When we look at the intermediate domain images for CIFAR10 and STL10 as shown in Appendix B, the informative target objects are zoomed and colored by red, while the background is colored by green. This can be interpreted as a kind of attention mechanism, which can improve the domain adaptation performance. We also present the experimental results using VisDA dataset in Appendix C.

Experiments on X-ray medical images

To show the robustness of the proposed framework for challenging problems, we applied the proposed algorithm to adapt the recognition of Pneumonia from an aligned dataset (Kermany et al. 2018) into a misaligned dataset (Wang et al. 2017), which are shown in Fig. 7. Every experimental setting is equivalent to the VisDA experiment, and IEDA gives the best performance as shown in Table 4.

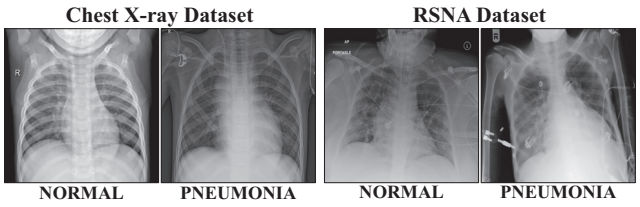


Figure 7: X-ray dataset examples

Table 4: Quantitative results on X-ray datasets

Algorithm	Source Only	MCDDA	IEDA
Accuracy	58.5%	67.21%	**72.37%**

Additional analysis

In contrast to the previous studies where the pseudo-labels from a single classifier are utilized (Cicek and Soatto 2019; Kumar et al. 2018), the consensus loss considers the ensemble predictive probability from the multiple classifiers. Thus, we can simultaneously consider the uncertainty among several labels and every correlation among the weak classifiers. To show the effectiveness of the consensus loss for the ensemble model, we perform additional experiments. When we replace the consensus loss by the cross-entropy loss based on the pseudo-labels from the ensemble classifiers, the performance reduces to 32.7% in MNIST$\rightarrow$SVHN. When the consensus loss is replaced by the marginal entropy for each classifier to ignore the correlations among the multiple classifiers, the performance decreases to 59.6% in MNIST$\rightarrow$SVHN.

In contrast to Zou et al. (2019) regularizing the pseudo-labels, the proposed framework entirely ignores the predicted class by the fooling loss. When the fooling loss is replaced by the regularization loss proposed by Zou et al. (2019), the performance decreases to 58.62% for MNIST$\rightarrow$SVHN. Thus, we can confirm that the proposed fooling loss is effective for the ensemble classifier.

Conclusion

In this paper, a unsupervised domain adaptation framework based on the bi-directional domain transformation to the hidden intermediate domain was suggested. Our main contribution is to introduce the importance of the intermediate domain for the domain adaptation tasks, which be transformed easily from the two domains. Furthermore, the alternate updating of multiple classifiers makes the neural network classify the ambiguous samples robustly. The intermediate domain and the sample labels of target domain could be jointly estimated by the training scheme using the double-structured architecture. With numerous ablation tests, we validated the effectiveness of the intermediate domain for the various domains, and the proposed algorithm outperformed the previous state-of-the-art algorithms. In addition, while the previous studies need to tune the parameters for the respective dataset, the proposed algorithm shares all the training parameters for every dataset, which validates its simplicity and generality. Even though the proposed framework showed

significantly high performance just by two simple transformers in the image adaptors, the framework can be extended by adding new transformers for various environments.

References

Ben-David, S.; Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Vaughan, J. W. 2010. A theory of learning from different domains. *Machine Learning* 79(1-2):151–175.

Bousmalis, K.; Trigeorgis, G.; Silberman, N.; Krishnan, D.; and Erhan, D. 2016. Domain separation networks. In *NIPS*.

Bousmalis, K.; Silberman, N.; Dohan, D.; Erhan, D.; and Krishnan, D. 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*.

Cicek, S., and Soatto, S. 2019. Unsupervised domain adaptation via regularized conditional alignment. In *ICCV*.

Coates, A.; Ng, A.; and Lee, H. 2011. An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*.

Ganin, Y., and Lempitsky, V. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*.

Ghifary, M.; Kleijn, W. B.; Zhang, M.; Balduzzi, D.; and Li, W. 2016. Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV*.

Ghifary, M.; Kleijn, W. B.; and Zhang, M. 2014. Domain adaptive neural networks for object recognition. In *Pacific Rim International Conference on Artificial Intelligence*.

Gong, R.; Li, W.; Chen, Y.; and Gool, L. V. 2019. Dlow: Domain flow for adaptation and generalization. In *CVPR*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*.

Hoffman, J.; Tzeng, E.; Park, T.; Zhu, J.-Y.; Isola, P.; Saenko, K.; Efros, A. A.; and Darrell, T. 2018. Cycada: Cycle-consistent adversarial domain adaptation. *ICML*.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; et al. 2015. Spatial transformer networks. In *NIPS*.

Kermany, D. S.; Goldbaum, M.; Cai, W.; Valentim, C. C.; Liang, H.; Baxter, S. L.; McKeown, A.; Yang, G.; Wu, X.; Yan, F.; et al. 2018. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Kumar, A.; Sattigeri, P.; Wadhawan, K.; Karlinsky, L.; Feris, R.; Freeman, W. T.; and Wornell, G. 2018. Co-regularized alignment for unsupervised domain adaptation. In *NIPS*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Liu, M.-Y., and Tuzel, O. 2016. Coupled generative adversarial networks. In *NIPS*.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. I. 2015. Learning transferable features with deep adaptation networks. In *ICML*.

Murez, Z.; Kolouri, S.; Kriegman, D.; Ramamoorthi, R.; and Kim, K. 2018. Image to image translation for domain adaptation. In *CVPR*.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Peng, X.; Usman, B.; Kaushik, N.; Hoffman, J.; Wang, D.; and Saenko, K. 2017. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*.

Russo, P.; Carlucci, F. M.; Tommasi, T.; and Caputo, B. 2018. From source to target and back: symmetric bi-directional adaptive gan. In *CVPR*.

Saito, K.; Watanabe, K.; Ushiku, Y.; and Harada, T. 2018. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*.

Saito, K.; Ushiku, Y.; and Harada, T. 2017. Asymmetric tri-training for unsupervised domain adaptation. In *ICML*.

Sankaranarayanan, S.; Balaji, Y.; Castillo, C. D.; and Chellappa, R. 2018. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*.

Shkodrani, S.; Hofmann, M.; and Gavves, E. 2018. Dynamic adaptation on non-stationary visual domains. *ECCV*.

Shu, R.; Bui, H. H.; Narui, H.; and Ermon, S. 2018. A dirt-t approach to unsupervised domain adaptation. In *ICLR*.

Tzeng, E.; Hoffman, J.; Saenko, K.; and Darrell, T. 2017. Adversarial discriminative domain adaptation. In *CVPR*.

Volpi, R.; Morerio, P.; Savarese, S.; and Murino, V. 2018. Adversarial feature augmentation for unsupervised domain adaptation. In *CVPR*.

Wang, C., and Mahadevan, S. 2011. Heterogeneous domain adaptation using manifold alignment. In *IJCAI*.

Wang, X.; Peng, Y.; Lu, L.; Lu, Z.; Bagheri, M.; and Summers, R. M. 2017. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *CVPR*.

Xu, R.; Li, G.; Yang, J.; and Lin, L. 2019. Larger norm more transferable: an adaptive feature norm approach for unsupervised domain adaptation. In *ICCV*.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *CVPR*.

Zou, Y.; Yu, Z.; Liu, X.; Bhagavatula, V.; and Wang, J. 2019. Confidence regularized self-training. In *ICCV*.