Asymmetric Tri-training for Unsupervised Domain Adaptation

Kuniaki Saito [1] **Yoshitaka Ushiku** [1] **Tatsuya Harada** [1 2]

Abstract

It is important to apply models trained on a large number of labeled samples to different domains because collecting many labeled samples in various domains is expensive. To learn discriminative representations for the target domain, we assume that artificially labeling the target samples can result in a good representation. Tri-training leverages three classifiers equally to provide pseudo-labels to unlabeled samples; however, the method does not assume labeling samples generated from a different domain. In this paper, we propose the use of an *asymmetric* tri-training method for unsupervised domain adaptation, where we assign pseudo-labels to unlabeled samples and train the neural networks as if they are true labels. In our work, we use three networks *asymmetrically*, and by *asymmetric*, we mean that two networks are used to label unlabeled target samples, and one network is trained by the pseudo-labeled samples to obtain target-discriminative representations. Our proposed method was shown to achieve a state-of-the-art performance on the benchmark digit recognition datasets for domain adaptation.

1. Inroduction

With the development of deep neural networks, including deep convolutional neural networks (CNN) (Krizhevsky et al., 2012), the ability to recognize images and languages has improved dramatically. Training deep-layered networks using a large number of labeled samples enables us to correctly categorize samples in diverse domains. In addition, the transfer learning of a CNN has been utilized in many studies. For object detection or segmentation, we can transfer the knowledge of a CNN trained using a large-scale dataset by fine-tuning it on a relatively small dataset (Girshick et al., 2014; Long et al., 2015a).

One of the problems inherent to neural networks is that, although such networks perform well on samples generated from the same distribution as the training samples, they may find it difficult to correctly recognize samples from different distributions at the test time. An example of this is images collected from the Internet, which may come in abundance and are fully labeled. Such images have a distribution that differs from images taken from a camera. Thus, a classifier that performs well on various domains is important for practical use. To realize such a classifier, it is necessary to learn domain-invariantly discriminative representations. However, acquiring such representations is not easy because it is often difficult to collect a large number of labeled samples, and because samples from different domains have domain-specific characteristics.

In unsupervised domain adaptation, we try to train a classifier that works well on a target domain under the condition that we are provided labeled source samples and unlabeled target samples during training. Most of the previously developed deep domain adaptation methods operate mainly under the assumption that the adaptation can be realized by matching the distribution of features from different domains. These methods have been aimed at obtaining domain-invariant features by minimizing the divergence between domains, as well as a category loss on the source domain (Ganin & Lempitsky, 2014; Long et al., 2015b; 2016). However, as shown in (Ben-David et al., 2010), if a classifier that works well on both the source and the target domains does not exist, we theoretically cannot expect a discriminative classifier to be applicable to the target domain. That is, even if the distributions are matched with the non-discriminative representations, the classifier may not work well on the target domain. Because the direct learning discriminative representations for the target domain, in the absence of target labels, is considered very difficult, we propose assigning pseudo-labels to the target samples and training the target-specific networks as if they were true labels.

Co-training and tri-training (Zhou & Li, 2005) leverage multiple classifiers to artificially label unlabeled samples and retrain the classifiers. However, such methods do not assume labeling samples from different domains. Because

[1]The University of Tokyo, Tokyo, Japan [2]RIKEN, Japan. Correspondence to: Kuniaki Saito <k-saito@mi.t.u-tokyo.ac.jp>, Yoshitaka Ushiku <ushiku@mi.t.u-tokyo.ac.jp>, Tatsuya Harada <harada@mi.t.u-tokyo.ac.jp>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

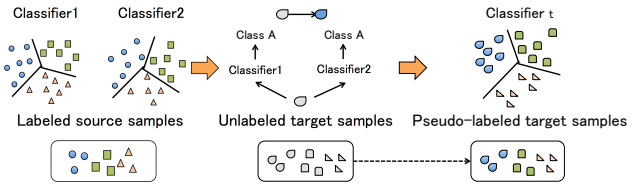


Figure 1. Outline of our model. We assign pseudo-labels to unlabeled target samples based on the predictions from two classifiers trained on the source samples.

our goal is to classify unlabeled target samples that have different characteristics from labeled source samples, we propose the use of *asymmetric* tri-training for unsupervised domain adaptation. By *asymmetric*, we mean that we assign different roles to three different classifiers.

In this paper, we propose a novel tri-training method for unsupervised domain adaptation, where we assign pseudo-labels to unlabeled samples, and train the neural networks utilizing these samples. As described in Fig. 1, two networks are used to label unlabeled target samples, and the remaining network is trained using the pseudo-labeled target samples. We evaluated our method using digit classification tasks, traffic sign classification tasks, and sentiment analysis tasks using the Amazon Review dataset, and demonstrated its state-of-the-art performance for nearly all of the conducted experiments. In particular, for the adaptation scenario, MNIST→SVHN, our method outperformed other methods by more than 10%.

2. Related Work

A number of previous methods have attempted to realize adaptation by measuring the divergence between different domains (Ganin & Lempitsky, 2014; Long et al., 2015b; Li et al., 2016). Such methods are based on the theory proposed in (Ben-David et al., 2010), which states that the expected loss for a target domain is bounded by three terms: (i) the expected loss for the source domain, (ii) the domain divergence between the source and target, and (iii) the minimum value of a shared expected loss. A shared expected loss indicates the sum of the loss on the source and target domains. Because the third term, which is usually considered to be very low, cannot be evaluated when labeled target samples are absent, most methods attempt to minimize the first and second terms. With regard to the training of deep architectures, the maximum mean discrepancy (MMD), or the loss of a domain classifier network, is utilized to measure the divergence corresponding to the second term (Gretton et al., 2012; Ganin & Lempitsky, 2014; Long et al., 2015b; 2016; Bousmalis et al., 2016). However, the third term is very important in training a CNN, which simultaneously extracts and recognizes the representations. The third term can easily become large when the representations are not discriminative for the target domain. Therefore, we focus on how to learn the target-discriminative representations to consider the third term. In (Long et al., 2016), the focus was on this point, and a target-specific classifier was constructed using a residual network structure. Differing from their method, we constructed a target-specific network by providing artificially labeled target samples.

Several transductive methods use a similarity of features to provide labels for unlabeled samples (Rohrbach et al., 2013; Khamis & Lampert, 2014). For unsupervised domain adaptation, in (Sener et al., 2016), a method was proposed to learn the labeling metrics by utilizing the k-nearest neighbors between unlabeled target samples and labeled source samples. In contrast to this method, our method explicitly and simply backpropagates the category loss for the target samples based on pseudo-labeled samples.

Many methods have proposed giving pseudo-labels to unlabeled samples by utilizing the predictions of a classifier and retraining it, including pseudo-labeled samples, a process called self-training. The underlying assumption of self-training is that one's own high-confidence predictions are correct (Zhu, 2005). As the predictions are mostly correct, utilizing samples with high confidence will further improve the performance of the classifier. Co-training utilizes two classifiers, which have different views on one sample, to provide pseudo-labels (Blum & Mitchell, 1998; Tanha et al., 2011). The unlabeled samples are then added to the training set if at least one classifier is confident regarding the predictions. The generalization capability of co-training is theoretically ensured (Balcan et al., 2004; Dasgupta et al., 2001) under certain assumptions, and applied to various tasks (Wan, 2009; Levin et al., 2003). In (Chen et al., 2011), the idea of co-training was incorporated into domain adaptation. Similar to co-training, tri-training uses the output of three different classifiers to provide pseudo-labels to unlabeled samples (Zhou & Li, 2005). Tri-training does not require partitioning features into different views; instead, tri-training initializes each classifier in a different manner. However, tri-training does not assume that the unlabeled samples follow different distributions from those the labeled ones are generated from. Hence, we developed a tri-training method for domain adaptation that utilizes three classifiers asymmetrically.

In (Lee, 2013), the effects of pseudo-labels on a neural network were investigated. The authors argued that the effect of training a classifier using pseudo-labels is equivalent to entropy regularization, thus leading to a low-density separation between classes. In our experiments, we observed that the target samples are separated in hidden features.

3. Method

In this section, we provide details of the proposed model for domain adaptation. We aim to construct a target-

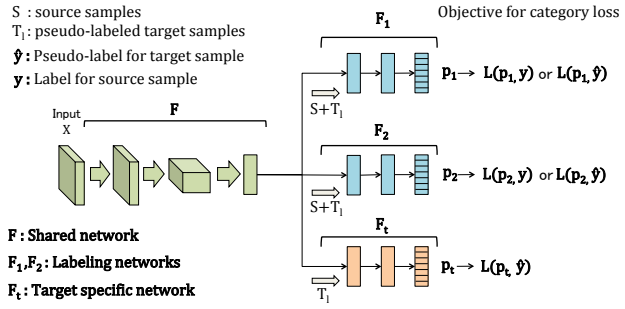


Figure 2. The proposed method includes a shared feature extractor (F), classifiers for labeled samples (F_1 and F_2) that learn from labeled source samples, and newly labeled target samples. In addition, a target-specific classifier (F_t) learns from pseudo-labeled target samples. Our method first trains networks from only labeled source samples, and then labels the target samples based on the output of F_1 and F_2. We train all architectures using these samples under the assumption that they are correctly labeled.

specific network by utilizing pseudo-labeled target samples. Simultaneously, we expect two labeling networks to acquire target-discriminative representations and gradually increase the accuracy on the target domain.

Our proposed network structure is shown in Fig. 2. Here, F denotes a network that outputs shared features from among three different networks, and F_1 and F_2 classify the features generated from F. Their predictions are utilized to provide pseudo-labels. The classifier F_t classifies features generated from F, which is a target-specific network. Here, F_1 and F_2 learn from the source and pseudo-labeled target samples, and F_t learns only from the pseudo-labeled target samples. The shared network F learns from all gradients from F_1, F_2, and F_t. Without such a shared network, another option for the network architecture is training the three networks separately, although this is inefficient in terms of training and implementation. Furthermore, by building a shared network, F, F_1, and F_2 can also harness the target-discriminative representations learned through the feedback from F_t.

The set of source samples is defined as $\{(x_i, y_i)\}_{i=1}^{m_s} \sim \mathbf{X^s}$, the unlabeled target set is $\{(x_i)\}_{i=1}^{m_t} \sim \mathbf{X^t}$, and the pseudo-labeled target set is $\{(x_i, \hat{y}_i)\}_{i=1}^{n_t} \sim \mathbf{X^t}_l$.

3.1. Loss for Multiview Features Network

In existing studies (Chen et al., 2011) on co-training for domain adaptation, the given features are divided into separate parts, and considered to be different views.

Because we aim to label the target samples with high accuracy, we expect F_1 and F_2 to classify the samples based on different viewpoints. Therefore, we make a constraint for the weights of F_1 and F_2 to make their inputs different from each other. We add the term $|{W_1}^T W_2|$ to the cost function, where W_1 and W_2 denote fully connected layer weights of F_1 and F_2, which are first applied to the feature $F(x_i)$. With this constraint, each network will learn from different features. The objective for the learning of F_1 and F_2 is defined as

$$E(\theta_F, \theta_{F_1}, \theta_{F_2}) = \frac{1}{n}\sum_{i=1}^{n}\big[L_y(F_1 \circ F(x_i)), y_i) + L_y(F_2 \circ (F(x_i)), y_i)\big] + \lambda|{W_1}^T W_2| \tag{1}$$

where L_y denotes the standard softmax cross-entropy loss function. We determined the trade-off parameter λ based on a validation split.

3.2. Learning Procedure and Labeling Method

Pseudo-labeled target samples will provide target-discriminative information to the network. However, because they certainly contain false labels, we have to pick up reliable pseudo-labels, which our labeling and learning method is aimed at realizing.

The entire training procedure of the network is shown in Algorithm 1. First, we train the entire network using the source training set $\mathbf{X^s}$. Here, F_1 and F_2 are optimized through Eq. (1), and F_t is trained based on a standard category loss. After training on $\mathbf{X^s}$, to provide pseudo-labels, we use the predictions of F_1 and F_2, namely, $\hat{y}^1, \hat{y}^2$ obtained from x_k. When C_1 and C_2 denote the class that has the maximum predicted probability for $\hat{y}^1, \hat{y}^2$, we assign a pseudo-label to x_k if the following two conditions are satisfied. First, we require $C_1 = C_2$ to provide pseudo-labels, which means the two different classifiers agree with the prediction. The second requirement is that the maximizing probability of $\hat{y}^1$ or $\hat{y}^2$ exceed the threshold parameter, which we set as 0.9 or 0.95 in the experiment. We suppose that unless one of the two classifiers is confident of the prediction, the prediction is not reliable. If the two requirements are satisfied, $\big(x_k, y_k = C_1 = C_2\big)$ is added to $\mathbf{X^t}_l$. To prevent an overfitting to the pseudo-labels, we resample the candidate for labeling the samples in each step. We set the number of initial candidates N_{init} to 5,000. We gradually increase the number of candidates $N_t = K/20 * n$, where n denotes the number of all target samples, and K denotes the number of steps; in addition, we set the maximum number of pseudo-labeled candidates to 40,000. We set K to 30 in the experiments. After the pseudo-labeled training set $\mathbf{X^t}_l$ is composed, F, F_1, and F_2 are updated based on the objective in Eq. (1) for the labeled training set $L = \mathbf{X^s} \cup \mathbf{X^t}_l$. Then, F and F_t are simply optimized based on the category loss for $\mathbf{X^t}_l$.

Discriminative representations will be learned by constructing a target-specific network trained only on the target samples. However, if only noisy pseudo-labeled samples are used for the training, the network may not learn any

Algorithm 1 *iter* denotes the iteration of the training. The function *Labeling* indicates the labeling method. We assign pseudo-labels to samples when the predictions of F_1 and F_2 agree, and at least one of them is confident of their predictions.

Input: data
$\mathbf{X^s} = \{(x_i, t_i)\}_{i=1}^{m}$, $\mathbf{X^t} = \{(x_j)\}_{j=1}^{n}$
$\mathbf{X^t}_l = \emptyset$
for $j = 1$ **to** *iter* **do**
  Train F, F_1, F_2, F_t with a mini-batch from the training set $\mathcal{S}$
end for
$N_t = N_{init}$
$\mathbf{X^t}_l$ = Labeling(F, F_1, F_2,$\mathbf{X^t}$, N_t)
$\mathcal{L} = \mathbf{X^s} \cup \mathbf{X^t}_l$
for K steps **do**
  for $j = 1$ **to** *iter* **do**
    Train F, F_1, F_2 with mini-batch from training set $\mathcal{L}$
    Train F, F_t with mini-batch from training set $\mathbf{X^t}_l$
  end for
  $\mathbf{X^t}_l = \emptyset$, $N_t = K/20 * n$
  $\mathbf{X^t}_l$ = Labeling(F, F_1, F_2,$\mathbf{X^t}$, N_t)
  $\mathcal{L} = \mathbf{X^s} \cup \mathbf{X^t}_l$
end for

useful representations. We then use both the source samples and pseudo-labeled samples for the training of F, F_1, and F_2 to ensure the accuracy. In addition, as the learning proceeds, F will learn target-discriminative representations, resulting in an improvement in accuracy for F_1 and F_2. This cycle will gradually enhance the accuracy in the target domain.

3.3. Batch Normalization for Domain Adaptation

Batch normalization (BN) (Ioffe & Szegedy, 2015), which whitens the output of the hidden layer in a CNN, is an effective technique for accelerating the training speed and enhancing the accuracy of the model. In addition, in domain adaptation, whitening the output of the hidden layer is effective in improving the performance, and makes the distribution in different domains similar (Sun et al., 2016; Li et al., 2016).

The input samples of F_1 and F_2 include both pseudo-labeled target samples and source samples. Introducing BN will be useful for matching the distribution and improving the performance. We add BN layers to F, F_1 and F_2, which we detail in our supplementary material.

4. Analysis

In this section, we provide a theoretical analysis to our approach. First, we provide insight into existing theory, and then introduce a simple expansion of the theory related to our method. The distribution of the source samples is denoted as $\mathcal{S}$; that of the target samples, as $\mathcal{T}$; and that of the pseudo-labeled target samples, as $\mathcal{T}_l$.

In (Ben-David et al., 2010), an equation was introduced showing that the upper bound of the expected error in the target domain depends on three terms, which include the divergence between different domains and the error of an ideal joint hypothesis. The divergence between the source and target domains, $\mathcal{H}\Delta\mathcal{H}$-distance, is defined as follows:

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) = 2 \sup_{(h,h')\in\mathcal{H}^2} \left| \mathop{\mathbf{E}}_{\mathbf{x}\sim\mathcal{S}}[h(\mathbf{x}) \neq h'(\mathbf{x})] - \mathop{\mathbf{E}}_{\mathbf{x}\sim\mathcal{T}}[h(\mathbf{x}) \neq h'(\mathbf{x})] \right|$$

This distance is frequently used to measure the adaptability between different domains.

The ideal joint hypothesis is defined as $h^* = \arg\min_{h\in H}\big(R_{\mathcal{S}}(h) + R_{\mathcal{T}}(h)\big)$, and its corresponding error is $C = R_{\mathcal{S}}(h^*) + R_{\mathcal{T}}(h^*)$, where R denotes the expected error for each hypothesis. The theorem is as follows.

Theorem 1. (Ben-David et al., 2010)
Let H be the hypothesis class. Given two different domains, $\mathcal{S}$ and $\mathcal{T}$, we have

$$\forall h \in H, R_{\mathcal{T}}(h) \leq R_{\mathcal{S}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) + C \qquad (2)$$

This theorem indicates that the expected error on the target domain is upper bounded by three terms: the expected error on the source domain, the domain divergence measured by the disagreement of the hypothesis, and the error of the ideal joint hypothesis. In an existing work (Ganin & Lempitsky, 2014; Long et al., 2015b), C was disregarded because it was considered to be negligible. If we are provided with fixed features, we do not need to consider this term because it is also fixed. However, if we assume that $x_s \sim \mathcal{S}$ and $x_t \sim \mathcal{T}$ are obtained from the last fully connected layer of the deep models, we should note that C is determined based on the output of the layer, as well as the necessity of considering this term.

We consider the pseudo-labeled target sample distributions $\mathcal{T}_l$ given false labels at a ratio of ρ. The shared error of h^* on $\mathcal{S}, \mathcal{T}_l$ is denoted as C'. The following inequality then holds:

$$\begin{aligned} \forall h \in H, R_{\mathcal{T}}(h) &\leq R_{\mathcal{S}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) + C \\ &\leq R_{\mathcal{S}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) + C' + \rho \end{aligned} \qquad (3)$$

We show a simple derivation of the inequality in the Supplementary materials section. In Theorem 1, we cannot measure C in the absence of labeled target samples. We can evaluate and minimize it approximately using pseudo-labels. Furthermore, when we consider the second term on the right-hand side, our method is expected to reduce

this term. This term intuitively denotes the discrepancy between different domains in the disagreement of two classifiers. If we regard h and h' as F_1 and F_2, respectively, $\mathop{\mathbf{E}}_{\mathbf{x}\sim\mathcal{S}}[h(\mathbf{x}) \neq h'(\mathbf{x})]$ should be very low because the training is based on the same labeled samples. Moreover, for the same reason, $\mathop{\mathbf{E}}_{\mathbf{x}\sim\mathcal{T}}[h(\mathbf{x}) \neq h'(\mathbf{x})]$ is expected to be low, although we use the training set $\mathbf{X^t}_l$ instead of the genuine labeled target samples. Thus, our method considers both the second and third terms in Theorem 1.

5. Experiment and Evaluation

We conducted extensive evaluations of our method on image datasets and a sentiment analysis dataset. We evaluated the accuracy of the target-specific networks.

Visual Domain Adaptation For visual domain adaptation, we conducted our evaluation on the digit and traffic sign datasets. The digit datasets include MNIST (LeCun et al., 1998), MNIST-M (Ganin & Lempitsky, 2014), Street View House Numbers (SVHN) (Netzer et al., 2011), and Synthetic Digits (SYN DIGITS) (Ganin & Lempitsky, 2014). We further evaluated our method on traffic sign datasets including Synthetic Traffic Signs (SYN SIGNS) (Moiseev et al., 2013) and the German Traffic Sign Recognition Benchmark (Stallkamp et al., 2011) (GTSRB). In total, five adaptation scenarios were evaluated during this experiment. Because the datasets used for evaluation are varied in previous studies, we extensively evaluated our method using these five scenarios.

Many previous studies have evaluated the fine-tuning of pretrained networks using ImageNet. This protocol assumes the existence of another source domain. In our work, we want to evaluate a situation in which we have access to only a single source domain and a single target domain.

Adaptation in Amazon Reviews To investigate its behavior on the language datasets, we evaluated our method on the Amazon Review dataset (Blitzer et al., 2006) through the same preprocessing used by (Chen et al., 2011; Ganin et al., 2016). The dataset contains reviews on four types of products: books, DVDs, electronics, and kitchen appliances. We evaluated our method under 12 domain adaptation scenarios. The results are shown in Table 1.

Baseline Methods We compared our method with five methods for unsupervised domain adaptation, including state-of-the art methods in visual domain adaptation: Maximum Mean Discrepancy (MMD) (Long et al., 2015b), Domain Adversarial Neural Network (DANN) (Ganin & Lempitsky, 2014), Deep Reconstruction Classification Network (DRCN) (Ghifary et al., 2016), Domain Separation Network (DSN) (Bousmalis et al., 2016), and k-Nearest Neighbor based adaptation (kNN-Ad) (Sener et al., 2016). We cited the results of MMD from (Bousmalis et al., 2016). In addition, we compared our method with CNN trained only on the source samples. We compared our method with Variational Fair AutoEncoder (VFAE) (Louizos et al., 2015) and DANN (Ganin et al., 2016) in our experiment on the Amazon Review dataset.

5.1. Implementation Detail

In our experiments on the image datasets, we employed the architecture of CNN used in (Ganin & Lempitsky, 2014). For a fair comparison, we separated the network at the hidden layer from which (Ganin & Lempitsky, 2014) constructed discriminator networks. Therefore, when considering a single classifier, for example, $F_1 \circ F$, the architecture is identical to a previous work. We also followed (Ganin & Lempitsky, 2014) with the other protocols. Based on a validation, we set the threshold value for the labeling method as 0.95 in MNIST↔SVHN. In other scenarios, we set it as 0.9. We used MomentumSGD for optimization, and set the momentum as 0.9, whereas the learning rate was set 0.01. λ was set to 0.01 for all scenarios based on our validation. In the Supplementary materials section, we provide details of the network architecture and the hyper-parameters.

For our experiments on the Amazon Review dataset, we used a similar architecture to that used in (Ganin et al., 2016): with the sigmoid activated, one dense hidden layer with 50 hidden units, and a softmax output. We extended its architecture to our method similarly to that of the CNN. λ was set to 0.001 based on a validation. Because the input is sparse, we used Adagrad (Duchi et al., 2011) for optimization. We repeated this evaluation ten times, and reported the mean accuracy.

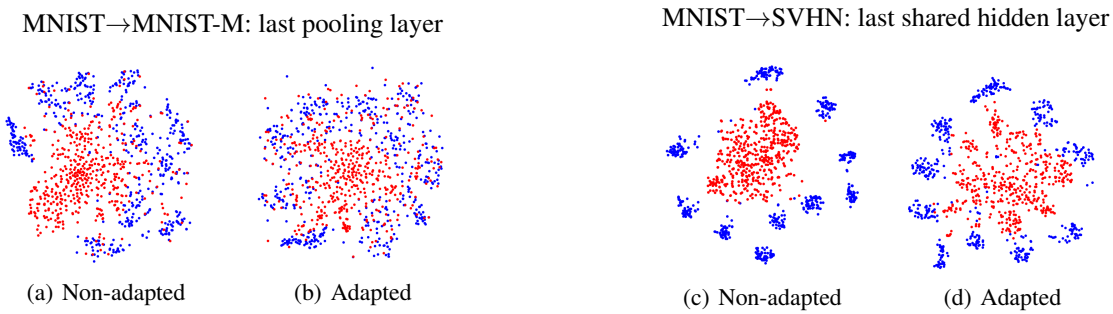
5.2. Experimental Result

In Tables 1 and 3, we show the main results of our experiments. When training only using source samples, the effect of the BN is not clear, as shown in the Tables 1. However, for most of the image recognition experiments, the effect of the BN with our method is clear; at the same time, the effect of our method is also clear when we do not use a BN in the network architecture compared to the *Source Only* method. The effect of the weight constraint is not obvious in other than MNIST→SVHN. This result indicates that we can obtain sufficiently different classifiers when initializing the layer parameters differently.

MNIST→MNIST-M First, we evaluated the adaptation between the hand-written digit dataset, MNIST, and its transformed dataset, MNIST-M. MNIST-M was composed by merging clips of a background from the BSDS500 datasets (Arbelaez et al., 2011). A patch was randomly taken from the images in BSDS500, and merged with the MNIST digits. From 59,001 target training samples, we randomly selected 1,000 labeled target samples as a validation split and tuned the hyper-parameters.

METHOD / SOURCE	MNIST	SVHN	MNIST	SYN DIGITS	SYN SIGNS
TARGET	MNIST-M	MNIST	SVHN	SVHN	GTSRB
Source Only w/o BN	59.1(56.6)	68.1(59.2)	37.2(30.5)	84.1(86.7)	79.2(79.0)
Source Only with BN	57.1	70.1	34.9	85.5	75.7
MMD (Long et al., 2015b)	76.9	71.1	-	88.0	91.1
DANN (Ganin & Lempitsky, 2014)	81.5	71.1	35.7	90.3	88.7
DRCN (Ghifary et al., 2016)	-	82.0	40.1	-	-
DSN (Bousmalis et al., 2016)	83.2	82.7	-	91.2	93.1
kNN-Ad (Sener et al., 2016)	86.7	78.8	40.3	-	-
Ours w/o BN	85.3	79.8	39.8	**93.1**	**96.2**
Ours w/o weight constraint ($\lambda = 0$)	**94.2**	**86.0**	49.7	92.4	94.0
Ours	94.0	85.8	**52.8**	92.9	**96.2**

Table 1. Results of the visual domain adaptation experiment on digit and traffic sign datasets. In every setting, our method outperforms other methods by a large margin. In the source-only results, we show the results reported in (Bousmalis et al., 2016) and (Ghifary et al., 2016) in parentheses.



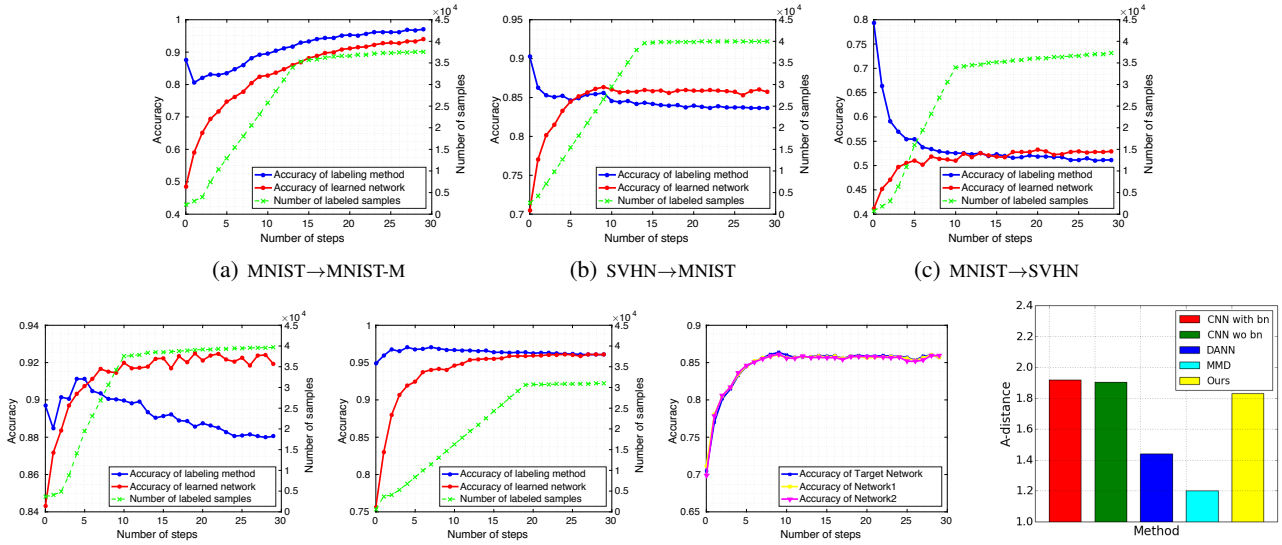
(a) Non-adapted (b) Adapted (c) Non-adapted (d) Adapted

Figure 3. We confirmed the effects our method through a visualization of the learned representations using t-distributed stochastic neighbor embedding (t-SNE) (Maaten & Hinton, 2008). The red points are the target samples, and the blue points are the source samples. **(a), (c)** The case in which only source samples are used for training. **(b), (d)** Adaptation using our proposed method. In both scenarios, MNIST→SVHN and MNIST→MNIST-M, we can see that the target samples are more dispersed through adaptation.

Our method outperformed the other existing method by about 7%. Visualization of the features in the last pooling layer is shown in Fig. 3(a)(b). We observed that the red target samples are more dispersed when adaptation is achieved. A comparison of the accuracy between the actual labeling accuracy on the target samples during the training and the test accuracy is shown in Fig. 4. The test accuracy is very low initially, but as the steps increase, the accuracy becomes closer to that of the labeling accuracy. With this adaptation, we can clearly see that the actual labeling accuracy gradually improves with the accuracy of the network.

SVHN↔MNIST We increased the gap between distributions during this experiment. We evaluated the adaptation between SVHN (Netzer et al., 2011) and MNIST in a ten-class classification problem. SVHN and MNIST have distinct appearances, and thus this adaptation is a challenging scenario, particularly in MNIST→SVHN. The images in SVHN are colored, and some contain multiple digits. Therefore, a classifier trained on SVHN is expected to perform well on MNIST, but the reverse is not true. MNIST does not include any samples containing multiple digits, and most of the samples are centered in the images, and thus adaptation from MNIST to SVHN is rather difficult. In both settings, we use 1,000 labeled target samples to find the optimal hyperparameters.

We evaluated our method under both adaptation scenarios and achieved a state-of-the-art performance for both datasets. In particular, for the adaptation MNIST→SVHN, our method outperformed the other methods by more than 10%. In Fig. 3(c)(d), the representations in MNIST→SVHN are visualized. Although the distributions seem to be separated between domains, the red SVHN samples become more discriminative when using our method compared with non-adapted embedding. A comparison between the actual labeling method accuracy and the testing accuracy is also shown in Fig. 4(b)(c). In this figure, it can be seen that the labeling accuracy rapidly decreases during the initial adaptation stage. On the other hand, the testing accuracy continues to improve, and finally exceeds the labeling accuracy. There are two questions regarding this interesting phenomenon. The first is why does the labeling method continue to decrease despite the increase in the



(a) MNIST→MNIST-M (b) SVHN→MNIST (c) MNIST→SVHN

(d) SYNDIGITS→SVHN (e) SYNSIGNS→GTSRB (f) Comparision of accuracy of three networks on SVHN→MNIST (g) $\mathcal{A}$-distance in MNIST→MNIST-M

Figure 4. **(a)** $\sim$ **(e)**: Comparison of the actual accuracy of the pseudo-labels and the learned network accuracy during training. The blue curve indicates the pseudo-label accuracy, and the red curve is the learned network accuracy. Note that the labeling accuracy is computed using (*the number of correctly labeled samples*)/(*the number of labeled samples*). The green curve shows the number of labeled target samples in each step. **(f)**: Comparison of the accuracy of the three networks in our model. The accuracy of the three networks improved almost simultaneously. **(g)**: Comparison of the $\mathcal{A}$-distance of the different methods. Our model slightly reduced the divergence of the domain compared with the source-only trained CNN.

testing accuracy? Target samples given pseudo-labels always include mistakenly labeled samples, whereas those given no labels are ignored in our method. Therefore, an error will be reinforced in the target samples included in the training set. The second question is why does the test accuracy continue to increase despite the lower labeling accuracy? The assumed reason is that the network already acquires target discriminative representations during this phase, which can improve the accuracy when using source samples and correctly labeled target samples.

In Fig. 4(f), we show a comparison of the accuracy of the three networks F_1, F_2, and F_t in SVHN→MNIST. The accuracy of these networks is nearly the same during every step. The same situation was observed for the other scenarios. Based on this result, we can state that target-discriminative representations are shared in three networks.

SYN DIGITS→SVHN With this experiment, we aimed to address a common adaptation scenario from synthetic images to real images. The datasets of synthetic numbers (Ganin & Lempitsky, 2014) consist of 500,000 images generated from Windows fonts by varying the text, positioning, orientation, background and stroke colors, and the amount of blur. We used 479,400 source samples and 73,257 target samples for training, and 26,032 target samples for testing. In addition, we used 1,000 SVHN samples as the validation set.

Our method also outperformed the other methods during this experiment. With this experiment, the effect of BN is not clear as compared with the other scenarios. The domain gap is considered small in this scenario, as the performance of the source-only classifier illustrates. In Fig. 4(d), although the labeling accuracy decreases, the accuracy of the learned network prediction improves, as in MNIST↔SVHN.

SYN SIGNS→GTSRB This setting is similar to the previous one, adaptation from synthetic images to real images, but we have a larger number of classes, namely, 43 classes instead of ten. We used the SYN SIGNS dataset (Ganin & Lempitsky, 2014) for the source, and the GTSRB dataset (Stallkamp et al., 2011) for the target, which consist of real images of traffic signs. We randomly selected 31,367 samples for the target training samples and evaluated the accuracy on the remaining samples. A total of 3,000 labeled target samples were used for validation.

Under this scenario, our method outperformed the other methods, which indicates that our method is effective for the adaptation from synthesized images to real images with diverse classes. As shown in Fig. 4(e), the same tendency as in MNIST↔SVHN was observed for this adaptation scenario.

Gradient Stop Experiment We evaluated the effects of a target-specific network using our method. We stopped the

Gradient stop branch	F_t	F_1, F_2	None
MNIST→MNIST-M	56.4	95.4	94.0
MNIST→SVHN	47.7	47.5	52.8
SYN SIGNS→GTSRB	96.5	93.1	96.2

Table 2. Results of gradient stop experiment. When stopping the gradients from F_t, we did not use backward gradients from F_t to F, and F only learned from F_1 and F_2. When stopping the gradients from F_1 and F_2, we did not use backward gradients from F_1 and F_2 on F, and F learned from F_t. *None* denotes our proposed method, in which we backwarded all gradients from all branches to F.

gradient from the upper layer networks F_1, F_2, and F_t to examine the effect on F_t. Table 2 shows three scenarios, including the case in which we stopped the gradients from F_1, F_2, and F_t.

In the experiment on MNIST→MNIST-M, we assumed that only the backpropagation from F_1 and F_2 cannot construct discriminative representations for the target samples, and confirmed the effect of F_t. For the adaptation on MNIST→SVHN, the best performance was realized when F received all gradients from the upper networks. Backwarding all gradients ensures both target-specific discriminative representations in difficult adaptations. In SYN SIGNS→GTSRB, backwarding only from F_t results in the worst performance because these domains are similar, and noisy pseudo-labeled samples worsen the performance.

$\mathcal{A}$-distance Based on the theoretical results in (Ben-David et al., 2010), the $\mathcal{A}$-distance is usually used as a measure of domain discrepancy. The method of estimating the empirical $\mathcal{A}$-distance is simple: We train a classifier to classify a domain from each domains' feature. The approximate distance is then calculated as $\hat{d}_{\mathcal{A}} = 2(1 - 2\epsilon)$, where ϵ is a generalization error of the classifier. We compared our method with the distribution matching methods, DANN and MMD. We calculated the distance using the last pooling layer features. We followed the implementation of DANN (Ganin et al., 2016) for the training. For MMD training, we followed the implementation in (Bousmalis et al., 2016). In Fig. 4(g), the $\mathcal{A}$-distance calculated from each CNN feature is shown. We used a linear SVM to calculate the distance. From this graph, we can see that our method clearly reduces the $\mathcal{A}$-distance compared with the CNN trained on only the source samples. In addition, when comparing the distribution matching methods against our own, although the former reduce the $\mathcal{A}$-distance much more, our method shows a superior performance as shown in Table 1.

Semi-supervised domain adaptation We evaluated our model in a semi-supervised domain adaptation setting on MNIST→SVHN. We randomly selected the labeled target samples for each class, and reported the mean accuracy for ten experiments. The resulting accuracy was 58% on average when using ten labeled target samples per class. We can see the effectiveness of our method in a semi-supervised setting. A detailed explanation of this is given in our Supplementary materials section.

Source→Target	VFAE	DANN	Our method
books→dvd	79.9	78.4	**80.7**
books→electronics	79.2	73.3	**79.8**
books→kitchen	81.6	77.9	**82.5**
dvd→books	**75.5**	72.3	73.2
dvd→electronics	**78.6**	75.4	77.0
dvd→kitchen	82.2	78.3	**82.5**
electronics→books	72.7	71.1	**73.2**
electronics→dvd	**76.5**	73.8	72.9
electronics→kitchen	85.0	85.4	**86.9**
kitchen→books	72.0	70.9	**72.5**
kitchen→dvd	73.3	74.0	**74.9**
kitchen→electronics	83.8	84.3	**84.6**

Table 3. Amazon Reviews experimental results. The accuracy (%) of the proposed method is shown with the result of VFAE (Louizos et al., 2015) and DANN (Ganin et al., 2016).

Amazon Reviews The reviews were encoded in 5,000 dimensional vectors of bag-of-word unigrams and bigrams with binary labels. Negative labels were attached to the samples if they were ranked with 1 to 3 stars. Positive labels were attached if they were ranked with 4 or 5 stars. We used 2,000 labeled source samples and 2,000 unlabeled target samples for the training, and between 3,000 and 6,000 samples for the testing. We used 200 labeled target samples for validation.

Based on the results in Table 3, our method performed better than VFAE (Louizos et al., 2015) and DANN (Ganin et al., 2016) in nine out of twelve settings. Our method was shown to be effective in learning a shallow network on different domains.

6. Conclusion

In this paper, we proposed a novel asymmetric tri-training method for unsupervised domain adaptation, which is implemented in a simple manner. We aimed at learning discriminative representations by utilizing pseudo-labels assigned to unlabeled target samples. We utilized three classifiers, two networks assigned pseudo-labels to unlabeled target samples, and the remaining network, which learned from them. We evaluated our method regarding both domain adaptation for a visual recognition and a sentiment analysis, and the results show that we outperformed all other methods. In particular, our method outperformed the other methods by more than 10% for MNIST→SVHN.

7. Acknowledgement

This work was partially funded by the ImPACT Program of the Council for Science, Technology, and Innovation (Cabinet Office, Government of Japan), and was partially supported by CREST, JST.

References

Arbelaez, Pablo, Maire, Michael, Fowlkes, Charless, and Malik, Jitendra. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011.

Balcan, Maria-Florina, Blum, Avrim, and Yang, Ke. Co-training and expansion: Towards bridging theory and practice. In *NIPS*, 2004.

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Blitzer, John, McDonald, Ryan, and Pereira, Fernando. Domain adaptation with structural correspondence learning. In *EMNLP*, 2006.

Blum, Avrim and Mitchell, Tom. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

Bousmalis, Konstantinos, Trigeorgis, George, Silberman, Nathan, Krishnan, Dilip, and Erhan, Dumitru. Domain separation networks. In *NIPS*, 2016.

Chen, Minmin, Weinberger, Kilian Q, and Blitzer, John. Co-training for domain adaptation. In *NIPS*, 2011.

Dasgupta, Sanjoy, Littman, Michael L, and McAllester, David. Pac generalization bounds for co-training. In *NIPS*, 2001.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(7):2121–2159, 2011.

Ganin, Yaroslav and Lempitsky, Victor. Unsupervised domain adaptation by backpropagation. In *ICML*, 2014.

Ganin, Yaroslav, Ustinova, Evgeniya, Ajakan, Hana, Germain, Pascal, Larochelle, Hugo, Laviolette, François, Marchand, Mario, and Lempitsky, Victor. Domain-adversarial training of neural networks. *JMLR*, 17(59): 1–35, 2016.

Ghifary, Muhammad, Kleijn, W Bastiaan, Zhang, Mengjie, Balduzzi, David, and Li, Wen. Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV*, 2016.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Gretton, Arthur, Borgwardt, Karsten M, Rasch, Malte J, Schölkopf, Bernhard, and Smola, Alexander. A kernel two-sample test. *JMLR*, 13(3):723–773, 2012.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

Khamis, Sameh and Lampert, Christoph H. Coconut: Co-classification with output space regularization. In *BMVC*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

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

Lee, Dong-Hyun. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML workshop on Challenges in Representation Learning*, 2013.

Levin, Anat, Viola, Paul A, and Freund, Yoav. Unsupervised improvement of visual detectors using co-training. In *ICCV*, 2003.

Li, Yanghao, Wang, Naiyan, Shi, Jianping, Liu, Jiaying, and Hou, Xiaodi. Revisiting batch normalization for practical domain adaptation. *arXiv:1603.04779*, 2016.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015a.

Long, Mingsheng, Cao, Yue, Wang, Jianmin, and Jordan, Michael I. Learning transferable features with deep adaptation networks. In *ICML*, 2015b.

Long, Mingsheng, Zhu, Han, Wang, Jianmin, and Jordan, Michael I. Unsupervised domain adaptation with residual transfer networks. In *NIPS*, 2016.

Louizos, Christos, Swersky, Kevin, Li, Yujia, Welling, Max, and Zemel, Richard. The variational fair autoencoder. *arXiv:1511.00830*, 2015.

Maaten, Laurens van der and Hinton, Geoffrey. Visualizing data using t-sne. *JMLR*, 9(11):2579–2605, 2008.

Moiseev, Boris, Konev, Artem, Chigorin, Alexander, and Konushin, Anton. Evaluation of traffic sign recognition methods trained on synthetically generated data. In *ACIVS*, 2013.

Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo, and Ng, Andrew Y. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

Rohrbach, Marcus, Ebert, Sandra, and Schiele, Bernt. Transfer learning in a transductive setting. In *NIPS*, 2013.

Sener, Ozan, Song, Hyun Oh, Saxena, Ashutosh, and Savarese, Silvio. Learning transferrable representations for unsupervised domain adaptation. In *NIPS*, 2016.

Stallkamp, Johannes, Schlipsing, Marc, Salmen, Jan, and Igel, Christian. The german traffic sign recognition benchmark: a multi-class classification competition. In *IJCNN*, 2011.

Sun, Baochen, Feng, Jiashi, and Saenko, Kate. Return of frustratingly easy domain adaptation. In *AAAI*, 2016.

Tanha, Jafar, van Someren, Maarten, and Afsarmanesh, Hamideh. Ensemble based co-training. In *BNAIC*, 2011.

Wan, Xiaojun. Co-training for cross-lingual sentiment classification. In *ACL*, 2009.

Zhou, Zhi-Hua and Li, Ming. Tri-training: Exploiting unlabeled data using three classifiers. *TKDE*, 17(11):1529–1541, 2005.

Zhu, Xiaojin. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, 2005.