Asymmetric Tri-training for Unsupervised Domain Adaptation

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

Abstract

Deep-layered models trained on a large number of labeled samples boost the accuracy of many tasks. It is important to apply such models to different domains because collecting many labeled samples in various domains is expensive. In unsupervised domain adaptation, one needs to train a classifier that works well on a target domain when provided with labeled source samples and unlabeled target samples. Although many methods aim to match the distributions of source and target samples, simply matching the distribution cannot ensure accuracy on the target domain. To learn discriminative representations for the target domain, we assume that artificially labeling target samples can result in a good representation. Tri-training leverages three classifiers equally to give pseudo-labels to unlabeled samples, but the method does not assume labeling samples generated from a different domain. In this paper, we propose an *asymmetric* tri-training method for unsupervised domain adaptation, where we assign pseudo-labels to unlabeled samples and train neural networks as if they are true labels. In our work, we use three networks *asymmetrically*. By *asymmetric*, we mean that two networks are used to label unlabeled target samples and one network is trained by the samples to obtain target-discriminative representations. We evaluate our method on digit recognition and sentiment analysis datasets. Our proposed method achieves state-of-the-art performance on the benchmark digit recognition datasets of domain adaptation.

1. Introduction

With the development of deep neural networks including deep convolutional neural networks (CNN) (Krizhevsky et al., 2012), the recognition abilities of images and languages have improved dramatically. Training deep-layered networks with a large number of labeled samples enables us to correctly categorize samples in diverse domains. In addition, the transfer learning of CNN is utilized in many studies. For object detection or segmentation, we can transfer the knowledge of a CNN trained with a large-scale dataset by fine-tuning it on a relatively small dataset (Girshick et al., 2014; Long et al., 2015a). Moreover, features from a CNN trained on ImageNet (Deng et al., 2009) are useful for multimodal learning tasks including image captioning (Vinyals et al., 2015) and visual question answering (Antol et al., 2015).

One of the problems of neural networks is that although they perform well on the 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. One example is images collected from the Internet, which may come in abundance and be fully labeled. They have a distribution different from the images taken from a camera. Thus, a classifier that performs well on various domains is important for practical use. To realize this, 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 on the condition that we are provided labeled source samples and unlabeled target samples during training. Most of the previous deep domain adaptation methods have been proposed mainly under the assumption that the adaptation can be realized by matching the distribution of features from different domains. These methods aimed to obtain 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), theoretically, if a classifier that works well on both the source and the target domains does not exist, we cannot expect a discriminative classifier for the target domain. That is, even if the distributions are matched on the non-discriminative representations, the classifier may not work

[1]The University of Tokyo, Tokyo, 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>.

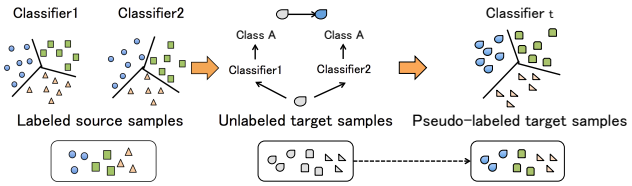


Figure 1. Outline of our model. We assign pseudo-labels to unlabeled target samples based on the predictions from two classifiers trained on source samples.

well on the target domain. Since directly learning discriminative representations for the target domain, in the absence of target labels, is considered very difficult, we propose to assign pseudo-labels to target samples and train 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, the methods do not assume labeling samples from different domains. Since our goal is to classify unlabeled target samples that have different characteristics from labeled source samples, we propose *asymmetric* tri-training for unsupervised domain adaptation. By *asymmetric*, we mean that we assign different roles to three 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 neural networks utilizing the samples. As described in Fig. 1, two networks are used to label unlabeled target samples and the remaining network is trained by the pseudo-labeled target samples. Our method does not need any special implementations. We evaluate our method on the digit classification task, traffic sign classification task and sentiment analysis task using the Amazon Review dataset, and demonstrate state-of-the-art performance in nearly all experiments. In particular, in the adaptation scenario, MNIST→SVHN, our method outperformed other methods by more than 10%.

2. Related Work

As many methods have been proposed to tackle various tasks in domain adaptation, we present details of the research most closely related to our paper.

A number of previous methods attempted to realize adaptation by utilizing the measurement of divergence between different domains (Ganin & Lempitsky, 2014; Long et al., 2015b; Li et al., 2016). The 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) expected loss for the source domain; (ii) domain divergence between source and target; and (iii) the minimum value of a shared expected loss. The shared expected loss means the sum of the loss on the source and target domain. As the third term, which is usually considered to be very low, cannot be evaluated when labeled target samples are absent, most methods try to minimize the first term and the second term. With regards to training deep architectures, the maximum mean discrepancy (MMD) or a loss of 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 CNN, which simultaneously extract representations and recognize them. The third term can easily be large when the representations are not discriminative for the target domain. Therefore, we focus on how to learn target-discriminative representations considering the third term. In (Long et al., 2016) the focus was on the point we have stated and a target-specific classifier was constructed using a residual network structure. Different from their method, we constructed a target-specific network by providing artificially labeled target samples.

Several transductive methods use 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 labeling metrics by using 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 target samples based on pseudo-labeled samples. Our approach does not require any special modules.

Many methods proposed to give pseudo-labels to unlabeled samples by utilizing the predictions of a classifier and retraining it including the pseudo-labeled samples, which is 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). Then, the unlabeled samples are added to training set if at least one classifier is confident about the predictions. The generalization ability of co-training is theoretically ensured (Balcan et al., 2004; Dasgupta et al., 2001) under some 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. Tri-training can be regarded as the extension of co-training (Zhou & Li, 2005). Similar to co-training, tri-training uses the output of three different classifiers to give pseudo-labels to unlabeled samples. Tri-training does not require partitioning features into different views; instead, tri-training initializes each classifier differently. However,

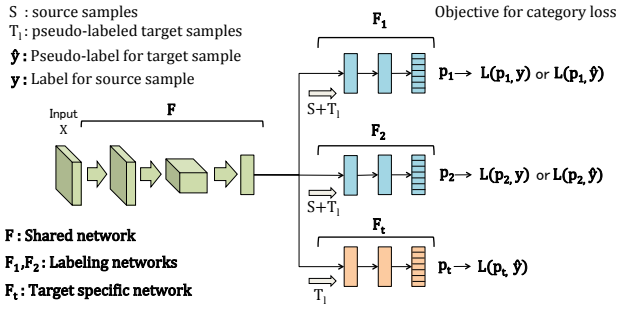


Figure 2. The proposed method includes a shared feature extractor (F), classifiers for labeled samples (F_1, F_2), which learn from labeled source samples, and newly labeled target samples. In addition, a target-specific classifier (F_t) learns from pseudo-labeled target samples. Our method first trains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

tri-training does not assume that the unlabeled samples follow the different distributions from the ones which labeled samples are generated from. Therefore, we develop a tri-training method suitable for domain adaptation by using three classifiers asymmetrically.

In (Lee, 2013), the effect of pseudo-labels in a neural network was investigated. They argued that the effect of training a classifier with pseudo-labels is equivalent to entropy regularization, thus leading to a low-density separation between classes. In addition, in our experiment, we observe that 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-specific network by utilizing pseudo-labeled target samples. Simultaneously, we expect two labeling networks to acquire target-discriminative representations and gradually increase accuracy on the target domain.

We show our proposed network structure in Fig. 2. Here F denotes the network which outputs shared features among three networks, F_1 and F_2 classify features generated from F. Their predictions are utilized to give pseudo-labels. The classifier F_t classifies features generated from F, which is a target-specific network. Here F_1, F_2 learn from source and pseudo-labeled target samples and F_t learns only from pseudo-labeled target samples. The shared network F learns from all gradients from F_1, F_2, F_t. Without such a shared network, another option for the network architecture we can think of is training three networks separately, but 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 by the feedback from F_t.

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

3.1. Loss for Multiview Features Network

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

As we aim to label target samples with high accuracy, we expect F_1, F_2 to classify samples based on different viewpoints. Therefore, we make a constraint for the weight of F_1, F_2 to make their inputs different to each other. We add the term $|{W_1}^T W_2|$ to the cost function, where W_1, W_2 denote fully connected layers' weights of F_1 and F_2 which are first applied to the feature $F(x_i)$. Each network will learn from different features with this constraint. The objective for learning F_1, 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 decided the trade-off parameter λ based on validation split.

3.2. Learning Procedure and Labeling Method

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

The entire procedure of training the network is shown in Algorithm 1. First, we train the entire network with source training set $\mathcal{S}$. Here F_1, F_2 are optimized by Eq. (1) and F_t is trained on standard category loss. After training on $\mathcal{S}$, to provide pseudo-labels, we use predictions of F_1 and F_2, namely y^1, y^2 obtained from x_k. When C_1, C_2 denote the class which has the maximum predicted probability for y^1, 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 give pseudo-labels, which means two different classifiers agree with the prediction. The second requirement is that the maximizing probability of y^1 or y^2 exceeds the threshold parameter, which we set as 0.9 or 0.95 in the experiment. We suppose that unless one of two classifiers is confident of the prediction, the prediction is not reliable. If the two requirements are satisfied, $\big(x_k, \hat{y}_k = C_1 = C_2\big)$ is added to $\mathcal{T}_l$. To prevent the overfitting to pseudo-labels, we resample the candidate for labeling samples in each step. We set the

Algorithm 1 $iter$ denotes the iteration of training. The function *Labeling* means the method of labeling. 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
$\mathcal{S} = \{(x_i, t_i)\}_{i=1}^{m}$, $\mathcal{T} = \{(x_j)\}_{j=1}^{n}$
$\mathcal{T}_l = \emptyset$
for $j = 1$ **to** $iter$ **do**
  Train F, F_1, F_2, F_t with mini-batch from training set $\mathcal{S}$
end for
$N_t = N_{init}$
$\mathcal{T}_l$ = Labeling($F, F_1, F_2, \mathcal{T}, N_t$)
$\mathcal{L} = \mathcal{S} \cup \mathcal{T}_l$
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 $\mathcal{T}_l$
  end for
  $\mathcal{T}_l = \emptyset$, $N_t = k/20 * n$
  $\mathcal{T}_l$ = Labeling($F, F_1, F_2, \mathcal{T}, N_t$)
  $\mathcal{L} = \mathcal{S} \cup \mathcal{T}_l$
end for

number of the initial candidates N_{init} as 5,000. We gradually increase the number of the candidates $N_t = k/20 * n$, where n denotes the number of all target samples and k denotes the number of steps, and we set the maximum number of pseudo-labeled candidates as 40,000. After the pseudo-labeled training set $\mathcal{T}_l$ is composed, F, F_1, F_2 are updated by the objective Eq. (1) on the labeled training set $L = \mathcal{S} \cup \mathcal{T}_l$. Then, F, F_t are simply optimized by the category loss for $\mathcal{T}_l$.

Discriminative representations will be learned by constructing a target-specific network trained only on target samples. However, if only noisy pseudo-labeled samples are used for training, the network may not learn useful representations. Then, we use both source samples and pseudo-labeled samples for training F, F_1, F_2 to ensure the accuracy. Also, as the learning proceeds, F will learn target-discriminative representations, resulting in an improvement in accuracy in F_1, 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 to accelerate training speed and enhance the accuracy of the model. In addition, in domain adaptation, whitening the hidden layer's output is effective for improving the performance, which make the distribution in different domains similar (Sun et al., 2016; Li et al., 2016).

Input samples of F_1, F_2 include both pseudo-labeled target samples and source samples. Introducing BN will be useful for matching the distribution and improves the performance. We add the BN layer in the last layer in F.

4. Analysis

In this section, we provide a theoretical analysis to our approach. First, we provide an insight into existing theory, then we introduce a simple expansion of the theory related to our method.

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 source and target domain, $\mathcal{H}\Delta\mathcal{H}$-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 on each hypothesis. The theorem is as follows.

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

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

This theorem means 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 the existing work (Ganin & Lempitsky, 2014; Long et al., 2015b), C was disregarded because it was considered to be negligibly small. If we are provided with fixed features, we do not need to consider the term because the term is also fixed. However, if we assume that $x_s \sim \mathcal{S}$, $x_t \sim \mathcal{T}$ are obtained from the last fully connected layer of deep models, we note that C is determined by the output of the layer, and further note the necessity of considering this term.

We consider the pseudo-labeled target samples set $T_l = \{(x_i, \hat{y}_i)\}_{i=1}^{m_t}$ given false labels at the ratio of ρ. The shared error of h^* on $\mathcal{S}, \mathcal{T}_l$ is denoted as C'. Then, the following inequality holds:

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

$\leq R_{\mathcal{S}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) + C' + \rho$

We show a simple derivation of the inequality in the Supplementary material. In Theorem 1, we cannot measure C in the absence of labeled target samples. We can approximately evaluate and minimize it by 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 certain h and h' as F_1 and F_2, respectively, $\mathop{\mathbf{E}}_{\mathbf{x}\sim\mathcal{S}_{\mathbf{x}}}[h(\mathbf{x}) \neq h'(\mathbf{x})]$ should be very low because training is based on the same labeled samples. Moreover, for the same reason, $\mathop{\mathbf{E}}_{\mathbf{x}\sim\mathcal{T}_{\mathbf{x}}}[h(\mathbf{x}) \neq h'(\mathbf{x})]$ is expected to be low, although we use the training set $\mathcal{T}_l$ instead of genuine labeled target samples. Thus, our method will consider both the second and the third term in Theorem 1.

5. Experiment and Evaluation

We perform extensive evaluations of our method on image datasets and a sentiment analysis dataset. We evaluate the accuracy of target-specific networks in all experiments.

Visual Domain Adaptation For visual domain adaptation, we perform our evaluation on the digits datasets and traffic signs datasets. Digits 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 evaluate our method on traffic sign datasets including Synthetic Traffic Signs (SYN SIGNS) (Moiseev et al., 2013) and German Traffic Signs Recognition Benchmark (Stallkamp et al., 2011) (GTSRB). In total, five adaptation scenarios are evaluated in this experiment. As the datasets used for evaluation are varied in previous works, we extensively evaluate our method on the five scenarios.

We do not evaluate our method on Office (Saenko et al., 2010), which is the most commonly used dataset for visual domain adaptation. As pointed out by (Bousmalis et al., 2016), some labels in that dataset are noisy and some images contain other classes' objects. Furthermore, 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 the situation where we have access to only one source domain and one target domain.

Adaptation in Amazon Reviews To investigate the behavior on language datasets, we also evaluated our method on the Amazon Reviews dataset (Blitzer et al., 2006) with the same preprocessing as 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 evaluate our method on 12 domain adaptation scenarios. The results are shown in Table 1.

Baseline Methods We compare 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 Networks (DSN) (Bousmalis et al., 2016), and k-Nearest Neighbor based adaptation (kNN-Ad) (Sener et al., 2016). We cite the results of MMD from (Bousmalis et al., 2016). In addition, we compare our method with CNN trained only on source samples. We compare our method with Variational Fair AutoEncoder (VFAE) (Louizos et al., 2015) and DANN (Ganin et al., 2016) in the Amazon Reviews experiment.

5.1. Implementation Detail

In experiments on image datasets, we employ the architecture of CNN used in (Ganin & Lempitsky, 2014). For a fair comparison, we separate the network at the hidden layer from which (Ganin & Lempitsky, 2014) constructed discriminator networks. Therefore, when considering one classifier, for example, $F_1 \circ F$, the architecture is identical to previous work. We also follow (Ganin & Lempitsky, 2014) in the other protocols. 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 use MomentumSGD for optimization and set the momentum as 0.9, while the learning rate is determined on validation splits and uses either $[0.01, 0.05]$. λ is set 0.01 in all scenarios. In our Supplementary material, we provide details of the network architecture and hyper-parameters.

For experiments on the Amazon Reviews dataset, we use a similar architecture to that used in (Ganin et al., 2016): with sigmoid activated, one dense hidden layer with 50 hidden units, and softmax output. We extend the architecture to our method similarly in the architecture of CNN. λ is set as 0.001 based on the validation. Since the input is sparse, we use Adagrad (Duchi et al., 2011) for optimization. We repeat this evaluation 10 times and report mean accuracy.

5.2. Experimental Result

In Tables 1 and 3, we show the main results of the experiments. When training only on source samples, the effect of the BN is not clear as in Tables 1. However, in all image recognition experiments, the effect of BN in our method is clear; at the same time, the effect of our method is also clear when we do not use BN in the network architecture. The effect of the weight constraint is obvious in MNIST→SVHN.

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**	**86.2**	49.7	92.4	94.0
Ours	94.0	85.0	**52.8**	92.9	**96.2**

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

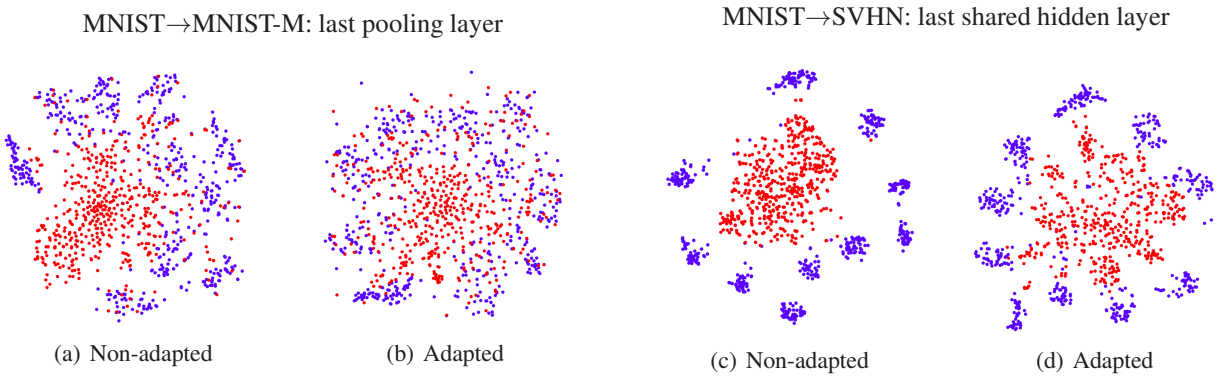
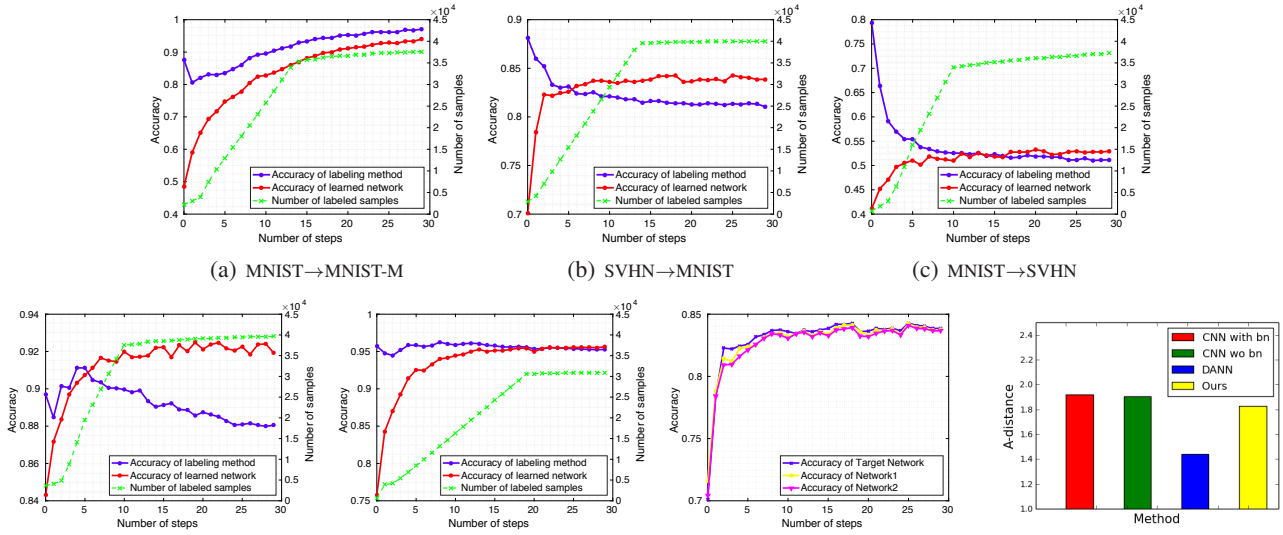


Figure 3. We confirm the effect our method by visualization of the learned representations by using t-distributed stochastic neighbor embedding (t-SNE) (Maaten & Hinton, 2008). Red points are target samples and blue points are source samples. The samples are all from testing samples. **(a), (c)** The case where we only use source samples for training. **(b), (d)** The case of adaptation by our method. In both scenarios, MNIST→SVHN and MNIST→MNIST-M, we can see that the target samples are more dispersed through adaptation.

MNIST→MNIST-M First, we evaluate the adaptation scenario between the hand-written digits dataset MNIST and its transformed dataset MNIST-M. MNIST-M is composed by merging the clip of the background from BSDS500 datasets (Arbelaez et al., 2011). A patch is randomly taken from the images in BSDS500, merged to MNIST digits. Even with this simple domain shift, the adaptation performance of CNN is much worse than the case where it was trained on target samples. From 59,001 target training samples, we randomly select 1,000 labeled target samples as a validation split and tuned hyper-parameters.

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

SVHN↔MNIST We increase the gap between distributions in this experiment. We evaluate adaptation between SVHN (Netzer et al., 2011) and MNIST in a ten-class classification problem. SVHN and MNIST have distinct appearance, thus this adaptation is a challenging scenario especially in MNIST→SVHN. SVHN is colored and some images 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 samples are



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

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

Figure 4. **(a)** $\sim$ **(e)**: Comparison of the actual accuracy of pseudo-labels and learned network accuracy during training. The blue curve is 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 is the number of labeled target samples in each step. **(f)**: Comparison of the accuracy of three networks in our model. Three networks almost simultaneously improve accuracy. **(g)**: Comparison of the $\mathcal{A}$-distance of different methods. Our model slightly reduced the divergence of the domain compared with source-only trained CNN.

centered in images, 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 evaluate our method on both adaptation scenarios and achieved state-of-the-art performance on both datasets. In particular, for the adaptation MNIST→SVHN, we outperformed other methods by more than 10%. In Fig. 3(c)(d), we visualize the representations in MNIST→SVHN. Although the distributions seem to be separated between domains, the red SVHN samples become more discriminative using our method compared with non-adapted embedding. We also show the comparison between actual labeling method accuracy and testing accuracy in Fig. 4(b)(c). In this figure, we can see that the labeling accuracy rapidly drops in the initial adaptation stage. On the other hand, testing accuracy continues to improve, and finally exceeds the labeling accuracy. There are two questions about this interesting phenomenon. The first question is why does the labeling method continue to decrease despite the increase in the test accuracy? Target samples given pseudo-labels always include mistakenly labeled samples whereas those given no labels are ignored in our method. Therefore, the error will be reinforced in the target samples that are included in training set. The second question is why does the test accuracy continue to increase despite the lower labeling accuracy? The assumed reasons are that the network already acquires target discriminative representations in this phase and they can improve the accuracy using source samples and correctly labeled target samples.

In Fig. 4(f), we also show the comparison of accuracy of the three networks F_1, F_2, F_t in SVHN→MNIST. The accuracy of three networks is nearly the same in every step. The same thing is observed in other scenarios. From this result, we can state that the target-discriminative representations are shared in all three networks.

SYN DIGITS→SVHN In 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 use 479,400 source samples and 73,257 target samples for training, and 26,032 target samples for testing. We use 1,000 SVHN samples as a validation set.

Our method also outperforms other methods in this experiment. In this experiment, the effect of BN is not clear compared with other scenarios. The domain gap is considered small in this scenario as the performance of the source-only classifier shows. In Fig. 4(d), although the labeling accuracy is dropping, the accuracy of the learned network's prediction is improving as in MNIST↔SVHN.

SYN SIGNS→GTSRB This setting is similar to the pre-

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→GTRB	96.5	93.1	96.2

Table 2. Results of Gradient stop experiment. When stopping gradients from F_t, we do not use backward gradients from F_t to F, and F learns only from F_1, F_2. When stopping gradients from F_1, F_2, we do not use backward gradients from F_1, F_2 to F, and F learns from F_t. *None* denotes our proposed method, we backward all gradients from all branches to F. In these three adaptation scenarios, our method shows stable performance.

vious setting, adaptation from synthetic images to real images, but we have a larger number of classes, namely 43 classes instead of 10. We use the SYN SIGNS dataset (Ganin & Lempitsky, 2014) for the source dataset and the GTSRB dataset (Stallkamp et al., 2011) for the target dataset, which consist of real traffic sign images. We select randomly 31,367 samples for target training samples and evaluate accuracy on the rest of the samples. A total of 3,000 labeled target samples are used for validation.

In this scenario, our method outperforms other methods. This result shows that our method is effective for the adaptation from synthesized images to real images, which have diverse classes. In Fig. 4(e), the same tendency as in MNIST↔SVHN is observed in this adaptation scenario.

Gradient Stop Experiment We evaluate the effect of the target-specific network in our method. We stop the gradient from upper layer networks F_1, F_2, and F_t to examine the effect of F_t. Table 2 shows three scenarios including the case where we stop the gradient from F_1, F_2, and F_t. In all scenarios, when we backward all gradients from F_1, F_2, F_t, we obtain clear performance improvements.

In the experiment MNIST→MNIST-M, we can assume that only the backpropagation from F_1, F_2 cannot construct discriminative representations for target samples and confirm the effect of F_t. For the adaptation MNIST→SVHN, the best performance is realized when F receives all gradients from upper networks. Backwarding all gradients will ensure both target-specific discriminative representations in difficult adaptations. In SYN SIGNS→GTSRB, backwarding only from F_t produces the worst performance because these domains are similar and noisy pseudo-labeled target samples worsen the performance.

$\mathcal{A}$-distance From the theoretical results in (Ben-David et al., 2010), $\mathcal{A}$-distance is usually used as a measure of domain discrepancy. The way of estimating empirical $\mathcal{A}$-distance is simple, in which we train a classifier to classify a domain from each domains' feature. Then, the approximate distance is calculated as $\hat{d}_{\mathcal{A}} = 2(1 - 2\epsilon)$, where ϵ is the generalization error of the classifier. In Fig. 4(g), we show the $\mathcal{A}$-distance calculated from each CNN features. We used linear SVM to calculate the distance. From this graph, we can see that our method certainly reduces the $\mathcal{A}$-distance compared with the CNN trained on only source samples. In addition, when comparing DANN and our method, although DANN reduces $\mathcal{A}$-distance much more than our method, our method shows superior performance. This indicates that minimizing the domain discrepancy is not necessarily an appropriate way to achieve better performance.

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

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

6. Conclusion

In this paper, we have proposed a novel asymmetric tri-training method for unsupervised domain adaptation, which is simply implemented. We aimed to learn discriminative representations by utilizing pseudo-labels assigned to unlabeled target samples. We utilized three classifiers, two networks assign pseudo-labels to unlabeled target samples and the remaining network learns from them.

We evaluated our method both on domain adaptation on a visual recognition task and a sentiment analysis task, outperforming other methods. In particular, our method outperformed the other methods by more than 10% in the MNIST→SVHN adaptation task.

7. Acknowledgement

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

References

Antol, Stanislaw, Agrawal, Aishwarya, Lu, Jiasen, Mitchell, Margaret, Batra, Dhruv, Lawrence Zitnick, C., and Parikh, Devi. Vqa: Visual question answering. In *ICCV*, 2015.

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.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

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.

Saenko, Kate, Kulis, Brian, Fritz, Mario, and Darrell, Trevor. Adapting visual category models to new domains. In *ECCV*, 2010.

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.

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. In *CVPR*, 2015.

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.

Proof of Theorem

We introduce the derivation of theorem of the main paper. 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 on each hypothesis.

We consider the pseudo-labeled target samples set $T_l = \big\{(x_i, \hat{y}_i)\big\}_{i=1}^{m_t}$ given false labels at the ratio of ρ. The minimum shared error on $\mathcal{S}, \mathcal{T}_l$ is denoted as C'. Then, the following inequality holds:

$$\forall h \in H, R_{\mathcal{T}}(h) \leq R_{\mathcal{S}}(h) + \tfrac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}_{\mathbf{X}}, \mathcal{T}_{\mathbf{X}}) + C$$

$$\leq R_{\mathcal{S}}(h) + \tfrac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}_{\mathbf{X}}, \mathcal{T}_{\mathbf{X}}) + C' + \rho$$

Proof. The probabiliy of false labels in the pseudo-labeled set T_l is ρ. When we consider 0-1 loss function for l, the difference between the error based on the true labeled set and pseudo-labeled set is

$$|l(h(x_i), y_i) - l(h(x_i), \hat{y}_i)| = \begin{cases} 1 & y_i \neq \hat{y}_i \\ 0 & y_i = \hat{y}_i \end{cases}$$

Then, the difference in the expected error is,

$$\mathbf{E}[|l(h(x_i), y_i) - l(h(x_i), \hat{y}_i)|] \leq |R_{\mathcal{T}_l}(h) - R_{\mathcal{T}}(h)| \leq \rho$$

From the characteritic of the loss function, the triangle inequality will hold, then

$$\begin{aligned} R_{\mathcal{S}}(h) + R_{\mathcal{T}}(h) &= R_{\mathcal{S}}(h) + R_{\mathcal{T}}(h) - R_{\mathcal{T}_l}(h) + R_{\mathcal{T}_l}(h) \\ &\leq R_{\mathcal{S}}(h) + R_{\mathcal{T}_l}(h) + |R_{\mathcal{T}_l}(h) - R_{\mathcal{T}}(h)| \\ &\leq R_{\mathcal{S}}(h) + R_{\mathcal{T}_l}(h) + \rho \end{aligned}$$

From this result, the main inequality holds. □

CNN Architectures and training detail

Four types of architectures are used for our method, which is based on (Ganin & Lempitsky, 2014). The network topology is shown in Figs 6, 7 and 8. The other hyperparameters are decided on the validation splits. The learning rate is set to 0.05 in SVHN↔MNIST. In the other scenarios, it is set to 0.01. The batchsize for training F_t, F is set as 128, the batchsize for training F_1, F_2, F is set as 64 in all scenarios.

In MNIST→MNIST-M, the dropout rate used in the experiment is 0.2 for training F_t, 0.5 for training F_1, F_2. In MNIST→SVHN, we did not use dropout. We decreased learning rate to 0.001 after step 10. In SVHN→MNIST, the dropout rate used in the experiment is 0.5. In SYNDIGITS→SVHN, the dropout rate used in the experiment is 0.5. In SYNSIGNS→GTSRB, the dropout rate used in the experiment is 0.5.

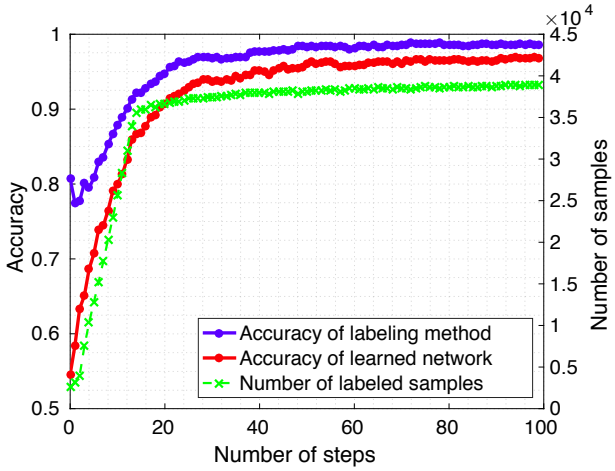


Figure 5. The behavior of our model when increasing the number of steps up to 100. Our model achieves accuracy of about 97%.

Supplementary experiments on MNIST→MNIST-M

We observe the behavior of our model when increasing the number of steps up to one hundred. We show the result in Fig. 5. Our model's accuracy gets about 97%. In our main experiments, we set the number of steps thirty, but from this experiment, further improvements can be expected.

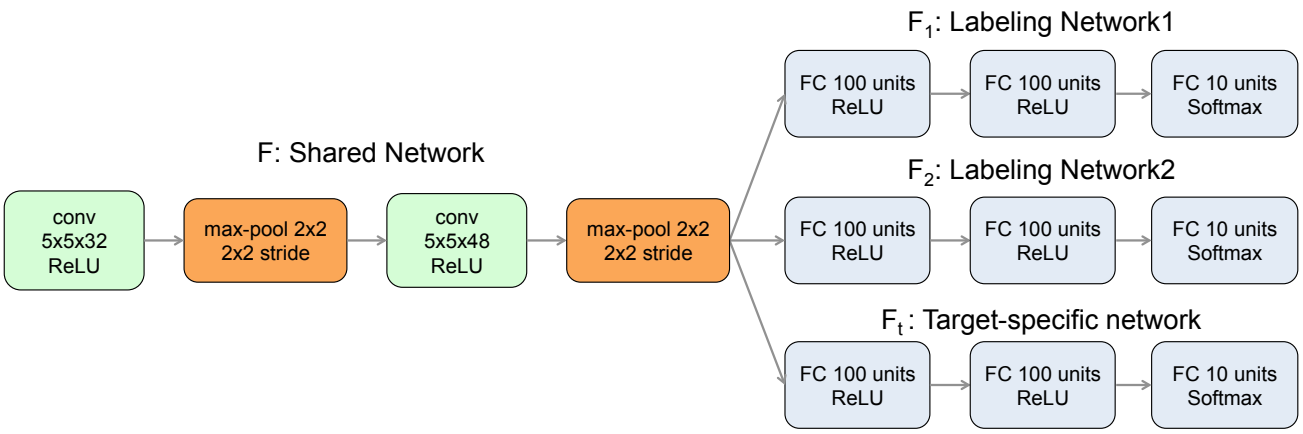


Figure 6. The architecture used for MNIST→MNIST-M. We added BN layer in the last convolution layer and FC layers in F_1, F_2. We also used dropout in our experiment.

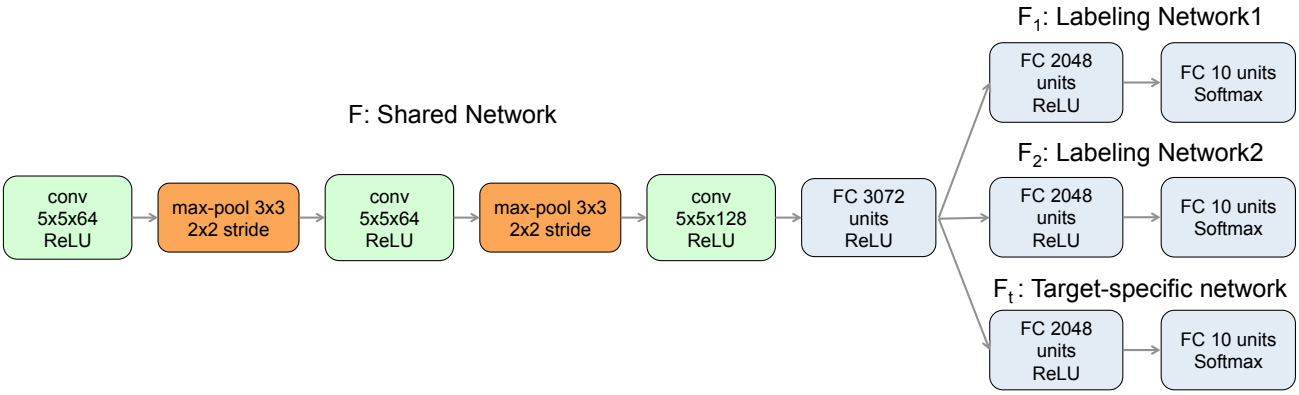


Figure 7. The architecture used for training SVHN. In MNIST→SVHN, we added a BN layer in the last FC layer in F. In SVHN→MNIST, SYN Digits↔SVHN, we added BN layer in the last convolution layer in F and FC layers in F_1,F_2 and also used dropout.

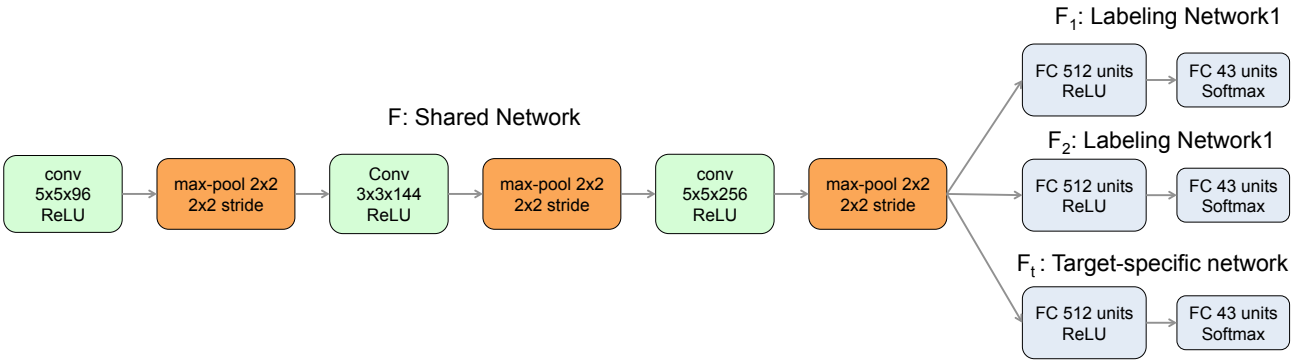


Figure 8. The architecture used in the adaptation Synthetic Signs→GTSRB. We added a BN layer after the last convolution layer in F and also used dropout.