Partially Zero-shot Domain Adaptation from Incomplete Target Data with Missing Classes

Masato Ishii[1,2,3] masato0713@gmail.com
Takashi Takenouchi[2,4] ttakashi@fun.ac.jp
Masashi Sugiyama[2,1] sugi@k.u-tokyo.ac.jp

[1]The University of Tokyo
Tokyo 113-8654, Japan

[2]RIKEN
Tokyo 103-0027, Japan

[3]NEC
Kanagawa 211-8666, Japan

[4]Future University of Hakodate
Hokkaido 041-8655, Japan

Abstract

We tackle a domain adaptation problem under partially zero-shot setting. In this setting, a certain subset of classes is missing in the unlabeled target data, while all classes appear in the labeled source data, and the goal is to discriminate all classes at the target domain. To solve this problem, we utilize an adversarial training scheme and adopt instance weighting to estimate the loss related to unavailable target data in the missing classes. The instance weight is computed on the basis of the prediction of deep neural networks, implying which instance would be similar to unseen data and having useful information for the loss estimation. This estimation makes it possible to explicitly consider all classes during the domain adaptation training even in the partially zero-shot setting, which leads to accurate adaptation between domains. Experimental results with several benchmark datasets validate the advantage of our method.

1. Introduction

The latest deep neural networks (DNNs) exhibit promising performance in various applications [10], such as image classification, audio recognition, and robotics. One of the key factors of their success is the availability of large-scale training datasets. Since a DNN has extremely high flexibility, a tremendous number of training data are required to utilize its full potential capability. However, obtaining such large-scale datasets may be typically hard in practical cases, therefore it has become a large obstacle to develop practical products or solutions using DNNs. To solve this problem, many researchers have worked on transfer learning [17] that enables us to reduce the number of training data by transferring knowledge or data from other related domains. In this paper, we focus on domain adaptation [6] that is one of the most popular methods for transfer learning.

Domain adaptation basically aims to match data distributions between the source and target domains so that a classifier trained on the matched source data performs well on the target data [6]. Due to the motivation of the domain adaptation, the amount of the target data is often small, which results in insufficient variation of the target data to be matched with the source data. A possible and important situation we want to focus on in this paper is that, in classification tasks, unlabeled target data do not cover a certain subset of class candidates (we call them missing classes), while labeled source data include enough data from all classes. We call this problem setting "partially zero-shot domain adaptation."

Suppose we want to detect anomalous behavior from surveillance videos by action recognition. Here, the action recognition is required to discriminate anomalous actions (e.g., fighting or falling down) as well as normal actions (e.g., walking or standing). When we have already constructed a training dataset for this recognition task at a certain surveillance camera, we can transfer it to a new camera at a different location by conducting the domain adaptation. In this case, unlabeled target data obtained by the new camera may not contain any anomalous actions, because such actions are rare. Although we have no annotations for the target data, we can guarantee that the target data do not contain the anomaly, if we know that any incident did not occur at the surveillance area while obtaining the target data. In this setting, standard domain adaptation methods would fail, because the source data that correspond to the missing classes do not have appropriate target data to be matched, and we cannot essentially match the source data with the target data as a whole. Generally, such a situation can appear if the class candidates have higher level groups (e.g., normal vs. anomalous actions) or a hierarchical structure. In that case, the target data may be only obtained at a subset of classes, and several classes are missing.

In this paper, we propose a novel method for the partially zero-shot domain adaptation. We adopt an adversarial training scheme that is widely used in the existing domain adaptation methods and extend it so that it can explicitly match the source data of the missing classes with the corresponding unseen target data. To this end, we consider the weighted loss of the given data instead of the loss of the unseen target data. By appropriately setting the instance weight based on the posterior estimated by DNNs, we can conduct accurate domain adaptation even though we do not have target data of the missing classes. Experimental results with several benchmark datasets validate the advantage of our method over both a standard domain adaptation method and a state-of-the-art partial domain adaptation method.

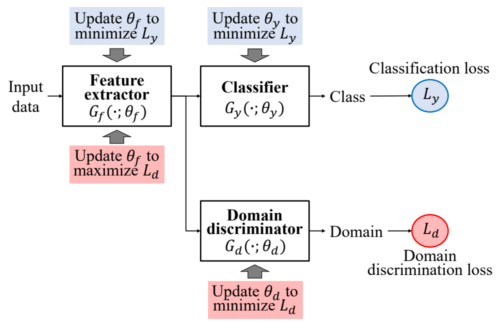
2. Related works

Domain adaptation is one of the most important problems in the field of machine learning as well as pattern recognition, and there is a large body of literature on this topic [6]. Recently, the feature-transform based approach [16, 9, 7, 8, 2, 23, 14, 20] has been typically adopted. In this approach, domain adaptation is achieved by embedding both domain data to a common feature space in which the distributions of both data are matched. Pioneering works [16, 9, 7] used a linear or kernel model for the feature embedding, while deep neural networks [8, 2, 23, 14, 20] have become quite popular in these days.

In this paper, we focus on the partially zero-shot setting where a certain subset of classes is missing in the unlabeled target data. A similar situation to ours was considered in several recent studies. The most related study is partial domain adaptation [24, 3, 4], and it considered the missing class in the source domain as an outlier class that we can ignore in classification. Our setting can be also considered as an extreme case of the domain adaptation with a class prior shift [25, 5] where the priors corresponding to the missing classes are set to zero at the target domain. In open-set domain adaptation [18, 21, 1], several unknown classes appear only at the target domain, while some classes are missing at the target domain in our setting. All the above existing works cannot solve our problem, because they cannot explicitly match the source data of missing classes with corresponding unseen target data, which results in poor performance of the classifier with the missing class data. Since we want to discriminate all classes included in the source data at the target domain, we need to consider how to match the missing class data between both domains.

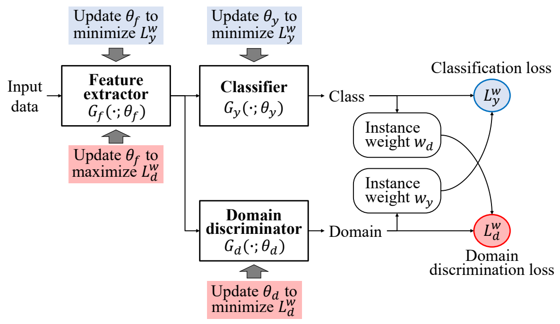
3. Partially zero-shot domain adaptation

In this section, we propose a novel method for partially zero-shot domain adaptation. First, we introduce a baseline method of domain adaptation shown in Fig. 1(a). By adopting instance weights estimated by the outputs of the classifier and domain discriminator as shown in Fig. 1(b), we formulate our partially zero-shot domain adaptation method. For this formulation, we assume a binary classification at first and will extend it to a multi-class classification later.



Update θ_f to minimize L_y
Update θ_y to minimize L_y
Input data
Feature extractor $G_f(\cdot;\theta_f)$
Classifier $G_y(\cdot;\theta_y)$
Class
Classification loss
L_y
Update θ_f to maximize L_d
Domain discriminator $G_d(\cdot;\theta_d)$
Domain
L_d
Domain discrimination loss
Update θ_d to minimize L_d

(a) Domain adversarial neural networks [8]



Update θ_f to minimize L_y^w
Update θ_y to minimize L_y^w
Input data
Feature extractor $G_f(\cdot;\theta_f)$
Classifier $G_y(\cdot;\theta_y)$
Class
Classification loss
L_y^w
Instance weight w_d
Instance weight w_y
Update θ_f to maximize L_d^w
Domain discriminator $G_d(\cdot;\theta_d)$
Domain
L_d^w
Domain discrimination loss
Update θ_d to minimize L_d^w

(b) Proposed method

Figure 1. Comparison of the architecture for domain adaptation between (a) the baseline method and (b) the proposed method.

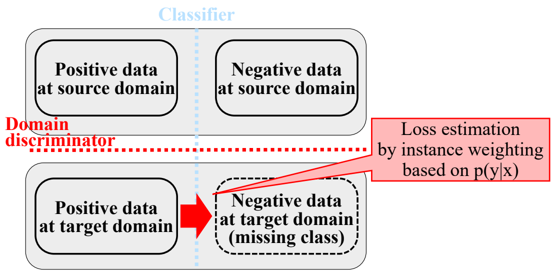
3.1. Baseline method

As a baseline method, we introduce Domain Adversarial Neural Networks (DANN) [8] that is one of the most popular domain adaptation methods based on adversarial training. Let $x \in \mathbb{R}^m, y \in C$ and $d \in \{\mathrm{S}, \mathrm{T}\}$ denote input data, labels, and domains, respectively, where m is the dimensionality of the input data, C is the set of the class candidates, S represents the source domain, and T represents the target domain. In DANN, we jointly train feature extractor $G_f : \mathbb{R}^m \to \mathbb{R}^{m'}$, classifier $G_y : \mathbb{R}^{m'} \to C$, and domain discriminator $G_d : \mathbb{R}^{m'} \to \{\mathrm{S}, \mathrm{T}\}$ based on two kinds of losses. The first loss is a classification loss that penalizes misclassification of the classifier:

$$L_y(\theta) = \sum_y \int \ell_y(x, y|\theta) p(x, y) \mathrm{d}x, \quad (1)$$

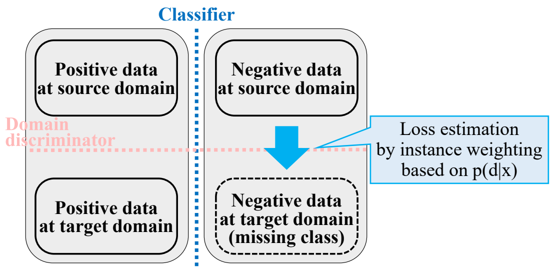
$$\ell_y(x, y|\theta) = \ell_{cp}(G_y(G_f(x; \theta_f); \theta_y), y), \quad (2)$$

where θ is a set of trainable parameters of DNNs and ℓ_{cp} is a loss function for the class prediction. In DANN, the standard cross-entropy loss is used for ℓ_{cp}. Since we train three networks G_f, G_y, and G_d, we have three sets of pa-



Classifier
Positive data at source domain
Negative data at source domain
Domain discriminator
Loss estimation by instance weighting based on p(y|x)
Positive data at target domain
Negative data at target domain (missing class)

(a) Calculation of the domain discrimination loss.



Classifier
Positive data at source domain
Negative data at source domain
Domain discriminator
Loss estimation by instance weighting based on p(d|x)
Positive data at target domain
Negative data at target domain (missing class)

(b) Calculation of the classification loss.

Figure 2. Loss calculation in the proposed method under a binary classification setting.

rameters: $\theta = \{\theta_f, \theta_y, \theta_d\}$. The second loss is a domain discrimination loss that penalizes mis-discrimination of the domain discriminator:

$$L_d(\theta) = \sum_d \int \ell_d(x, d|\theta) p(x, d) \mathrm{d}x, \quad (3)$$

$$\ell_d(x, d|\theta) = \ell_{dp}(G_d(G_f(x; \theta_f); \theta_d), d), \quad (4)$$

where ℓ_{dp} is a loss function for the domain prediction. In DANN, a logistic loss is used for ℓ_{dp}. Finally, training DANN is formulated as the following optimization:

$$E(\theta) = L_y(\theta) - \lambda L_d(\theta), \quad (5)$$

$$\theta_d^* = \arg\max_{\theta_d} E(\theta), \quad (6)$$

$$(\theta_f^*, \theta_y^*) = \arg\min_{\theta_f, \theta_y} E(\theta). \quad (7)$$

The optimization in Eq. (6) aims to discriminate domains of the input data by G_d, while that in Eq. (7) aims to fool G_d by G_f as well as accurately classifying the data by G_y. As a result, $G_f(x; \theta_f^*)$ becomes good feature embedding for domain adaptation, because it would be easy to classify as well as hard to discriminate domains.

3.2. Instance weighting for partially zero-shot domain adaptation

In this subsection, we formulate a novel method for partially zero-shot domain adaptation. In our scenario, a certain class (or classes) does not appear in the target data. For the moment, to make our formulation simple, we assume binary classification ($y \in \{\mathrm{P}, \mathrm{N}\}$: positive class vs. negative class). Note that we will extend our method to multi-class classification in Section 3.3 and that experiments will also be carried out for multi-class problems. Without loss of generality, we can consider that the negative class does not appear in the target data. In this subsection, first, we deform the domain discrimination loss L_d in Eq. (3) and clarify what problem will happen in our scenario when we use the standard domain adaptation method (Section 3.2.1). Then, we present a key trick of how to avoid the problem in our novel domain adaptation method (Section 3.2.2), which is instance weighting based on the class posterior $p(y|x)$ as shown in Fig. 2(a). The same trick will be also adopted for classification loss (Section 3.2.3), which results in instance weighting based on the domain posterior $p(d|x)$ as shown in Fig. 2(b). Finally, we formulate the overall optimization for our domain adaptation (Section 3.2.4).

3.2.1 The problem of the baseline method under the partially zero-shot setting

To clarify the problem in our scenario, we rewrite L_d in Eq. (3) as follows:

$$\begin{aligned} L_d(\theta) &= \sum_d \int \ell_d(x, d; \theta) \left(\sum_y p(x, y, d) \right) \mathrm{d}x \\ &= \pi_{\mathrm{PS}} \int \ell_d(x, \mathrm{S}; \theta) p(x|y = \mathrm{P}, d = \mathrm{S}) \mathrm{d}x \\ &+ \pi_{\mathrm{NS}} \int \ell_d(x, \mathrm{S}; \theta) p(x|y = \mathrm{N}, d = \mathrm{S}) \mathrm{d}x \\ &+ \pi_{\mathrm{PT}} \int \ell_d(x, \mathrm{T}; \theta) p(x|y = \mathrm{P}, d = \mathrm{T}) \mathrm{d}x \\ &+ \pi_{\mathrm{NT}} \int \ell_d(x, \mathrm{T}; \theta) p(x|y = \mathrm{N}, d = \mathrm{T}) \mathrm{d}x, \end{aligned} \quad (8)$$

where $\pi_{\cdot *} = p(y = \cdot, d = *)$. These four terms correspond to the domain discrimination losses for the source positive data, source negative data, target positive data, and target negative data, respectively. Since we do not have any target negative data, we cannot calculate the fourth term. If we ignore the fourth term by setting π_{NT} to be zero, the negative data only appear in the second term. It means that, if $G_f(x)$ is easy to classify, G_d can easily discriminate the domain of the negative data by just predicting as the source domain. Consequently, to fool G_d, we need to make $G_f(x)$ hard to classify, which results in poor classification performance.

3.2.2 Instance weighting for domain discrimination loss

To avoid the above-mentioned problem, we need to calculate the fourth term in Eq. (8) without the target negative data. Inspired by positive confidence learning [11], we estimate the value of that term based on the target positive data by adopting instance weights as

$$\begin{aligned}&\pi_{\mathrm{NT}}\int \ell_d(x,\mathrm{T};\theta)p(x|y{=}\mathrm{N},d{=}\mathrm{T})\mathrm{d}x\\ &=\pi_{\mathrm{PT}}\!\int\!\frac{\pi_{\mathrm{NT}}}{\pi_{\mathrm{PT}}}w_d(x)\ell_d(x,\mathrm{T};\theta)p(x|y{=}\mathrm{P},d{=}\mathrm{T})\mathrm{d}x,\end{aligned} \tag{9}$$

where $w_d(x)$ is the instance weight that can be defined as

$$w_d(x) \;=\; \frac{p(x|y=\mathrm{N},d=\mathrm{T})}{p(x|y=\mathrm{P},d=\mathrm{T})}. \tag{10}$$

By using Bayes' theorem and assuming the covariate shift condition [22], $p(y|x,d=\mathrm{S})=p(y|x,d=\mathrm{T})=p(y|x)$, we can rewrite the weight as follows:

$$\begin{aligned}w_d(x) &= \frac{p(y=\mathrm{N}|x,d=\mathrm{T})p(d=\mathrm{T}|x)p(x)}{p(y=\mathrm{N},d=\mathrm{T})}\\ &\quad\cdot\frac{p(y=\mathrm{P},d=\mathrm{T})}{p(y=\mathrm{P}|x,d=\mathrm{T})p(d=\mathrm{T}|x)p(x)}\\ &= \frac{p(y=\mathrm{P},d=\mathrm{T})}{p(y=\mathrm{N},d=\mathrm{T})}\cdot\frac{p(y=\mathrm{N}|x,d=\mathrm{T})}{p(y=\mathrm{P}|x,d=\mathrm{T})}\\ &= \frac{\pi_{\mathrm{PT}}}{\pi_{\mathrm{NT}}}\cdot\frac{p(y=\mathrm{N}|x)}{p(y=\mathrm{P}|x)}.\end{aligned} \tag{11}$$

If we empirically estimate π_{NT}, it should be zero and the weight diverges. Therefore, we regard π_{NT} and π_{PT} as hyper-parameters in our method. In this paper, we set them as $\pi_{\mathrm{NT}}=\pi_{\mathrm{NS}}$ and $\pi_{\mathrm{PT}}=\pi_{\mathrm{PS}}$. This setting corresponds to the assumption where $p(d=\mathrm{S})$ is equal to $p(d=\mathrm{T})$ and the class prior does not change between the source and target domains. Based on this assumption, Eq. (11) can be rewritten as the following equation:

$$w_d(x)=\frac{p(y=\mathrm{P}|d=\mathrm{S})}{p(y=\mathrm{N}|d=\mathrm{S})}\cdot\frac{p(y=\mathrm{N}|x)}{p(y=\mathrm{P}|x)}. \tag{12}$$

In the above equation, $p(y|d=\mathrm{S})$ is the class prior of the source data, therefore, it can be easily estimated by using the source data. On the other hand, $p(y|x)$ is the class posterior, and how to obtain it is not a trivial problem. In [11], the posterior is assumed to be given, but it is not a reasonable assumption in domain adaptation problems, so we need to estimate it. Fortunately, in DANN, we jointly train the classifier as well as the domain discriminator, and the output of the classifier is a probability distribution over the class candidates, which can be used as the class posterior like in [24]. We use the output of the classifier instead of the class posterior to calculate the instance weight in Eq. (12).

By substituting Eq. (12) into Eq. (9) and using it instead of the fourth term in Eq. (8), we can obtain a new formulation of the domain discrimination loss L_d^w that can be calculated without the target negative data as

$$\begin{aligned}L_d^w(\theta)=&\pi_{\mathrm{PS}}\int \ell_d(x,\mathrm{S};\theta)p_{\mathrm{PS}}(x)\mathrm{d}x\\ &+\pi_{\mathrm{NS}}\int \ell_d(x,\mathrm{S};\theta)p_{\mathrm{NS}}(x)\mathrm{d}x\\ &+\pi_{\mathrm{PS}}\!\int\!\left(1+\frac{\pi_{\mathrm{NS}}}{\pi_{\mathrm{PS}}}\hat{w}_d(x)\right)\ell_d(x,\mathrm{T};\theta)p_{\mathrm{PT}}(x)\mathrm{d}x,\end{aligned} \tag{13}$$

where $\hat{w}_d(x)$ is the instance weight in Eq. (12) calculated by the output of the classifier, and $p_{\cdot *}(x)$ represents $p(x|y=\cdot,d=*)$.

3.2.3 Instance weighting for classification loss

Since we assume binary classification, we can know that the target data are all positive data, which should be useful information for training the classifier. However, if we use the target data to calculate the classification loss L_y, the similar problem will happen as we have previously shown in the case of the domain discrimination loss. As in Eq. (8), we can decompose L_y in Eq. (1) as follows:

$$\begin{aligned}L_y(\theta) \;=\; &\pi_{\mathrm{PS}}\int \ell_y(x,\mathrm{P};\theta)p_{\mathrm{PS}}(x)\mathrm{d}x\\ &+\pi_{\mathrm{NS}}\int \ell_y(x,\mathrm{N};\theta)p_{\mathrm{NS}}(x)\mathrm{d}x\\ &+\pi_{\mathrm{PT}}\int \ell_y(x,\mathrm{P};\theta)p_{\mathrm{PT}}(x)\mathrm{d}x\\ &+\pi_{\mathrm{NT}}\int \ell_y(x,\mathrm{N};\theta)p_{\mathrm{NT}}(x)\mathrm{d}x.\end{aligned} \tag{14}$$

Again, we cannot calculate the fourth term in Eq. (14). Compared with the previous case, it seems not so problematic, because we do not fool the classifier. Even though the target data can be easily classified into the negative class in this case, it does not directly affect the training of DANN. However, since it leads to over-estimation of the performance of the classifier, it would disturb the training of the classifier. Therefore, we also adopt the instance weight to estimate the loss for the target negative data. In this case, we estimate the loss value based on the source negative data as

$$\int \ell_y(x,\mathrm{N};\theta)p_{\mathrm{NT}}(x)\mathrm{d}x=\int w_y(x)\ell_y(x,\mathrm{N};\theta)p_{\mathrm{NS}}(x)\mathrm{d}x, \tag{15}$$

where $w_y(x)$ is the instance weight that can be defined in a similar manner to Eq. (11), which results in

$$w_y(x)=\frac{\pi_{\mathrm{NS}}}{\pi_{\mathrm{NT}}}\cdot\frac{p(d=\mathrm{T}|x)}{p(d=\mathrm{S}|x)}=\frac{p(d=\mathrm{T}|x)}{p(d=\mathrm{S}|x)}. \tag{16}$$

Since we have already assumed $\pi_{\mathrm{NS}}=\pi_{\mathrm{NT}}$, $\frac{\pi_{\mathrm{NS}}}{\pi_{\mathrm{NT}}}$ was eliminated. Unlike the case of the domain discrimination loss, we need the domain posterior $p(d|x)$ instead of the class posterior to calculate the instance weights. Similarly to the class posterior, we use the output of the domain discriminator as an estimate of the domain posterior. By substituting Eq. (16) into Eq. (15) and using it instead of the fourth term in Eq. (14), we obtain a new formulation of the classification loss L_y^w that can be calculated without the target negative data:

$$\begin{aligned} L_y^w(\theta) &= \pi_{\mathrm{PS}} \int \ell_y(x, \mathrm{P}; \theta) p_{\mathrm{PS}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{NS}} \int (1 + \hat{w}_y(x)) \ell_y(x, \mathrm{N}; \theta) p_{\mathrm{NS}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{PS}} \int \ell_y(x, \mathrm{P}; \theta) p_{\mathrm{PT}}(x) \mathrm{d}x, \end{aligned} \tag{17}$$

where $\hat{w}_y(x)$ represents the instance weight in Eq. (16) calculated by the output of the domain discriminator.

3.2.4 Adversarial training for partially zero-shot domain adaptation

Since we have already derived a new domain discrimination loss and a classification loss that can be calculated without the target negative data, we can train DNNs based on these losses for the partially zero-shot domain adaptation. In a similar manner with DANN, we formulate our domain adaptation as follows:

$$E^w(\theta) = L_y^w(\theta) - \lambda L_d^w(\theta), \tag{18}$$

$$\theta_d^* = \arg\max_{\theta_d} E^w(\theta), \tag{19}$$

$$\left(\theta_f^*, \theta_y^*\right) = \arg\min_{\theta_f, \theta_y} E^w(\theta). \tag{20}$$

The above optimization seems to be the same as that of the baseline method, but, by adopting appropriate instance weights, we can accurately conduct domain adaptation, even though we do not have any target negative data.

3.3. Extension to multi-class classification

In the case of binary classification, only one of the classes does not appear in the target data. On the other hand, in the case of multi-class classification, multiple classes can disappear in the target data. Let A and M denote a set of classes that appear in the target data and a set of classes that do not appear, respectively. By using $y \in A$ instead of $y = \mathrm{P}$ and $y \in M$ instead of $y = \mathrm{N}$, we can derive our method under multi-class classification setting in a similar manner to what we have shown in the previous subsection. For the domain discrimination loss, the instance weight in Eq. (12) is formulated as

$$w_d(x) = \frac{p(y \in A|d = \mathrm{S})}{p(y \in M|d = \mathrm{S})} \cdot \frac{p(y \in M|x)}{p(y \in A|x)}. \tag{21}$$

Using this weight, we can derive the domain discrimination loss for multi-class classification setting as

$$\begin{aligned} L_d^w(\theta) = &\pi_{\mathrm{AS}} \int \ell_d(x, \mathrm{S}; \theta) p_{\mathrm{AS}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{MS}} \int \ell_d(x, \mathrm{S}; \theta) p_{\mathrm{MS}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{AS}} \int \left(1 + \frac{\pi_{\mathrm{MS}}}{\pi_{\mathrm{AS}}} \hat{w}_d(x)\right) \ell_d(x, \mathrm{T}; \theta) p_{\mathrm{AT}}(x) \mathrm{d}x, \end{aligned} \tag{22}$$

where $\pi_{\cdot *} = p(y \in \cdot, d = *)$, $p_{\cdot *}(x) = p(x|y \in \cdot, d = *)$, and $\hat{w}_d(x)$ is the instance weight in Eq. (21) calculated by the output of the classifier. Since ℓ_d does not depend on y, how to calculate the domain discrimination loss is almost the same as that for binary classification setting. On the other hand, in the case of the classification loss, it becomes somewhat different. The instance weight for the classification loss in Eq. (16) stays same, because it does not depend on y. However, the classification loss is changed to

$$\begin{aligned} L_y^w(\theta) = &\pi_{\mathrm{AS}} \sum_{i \in A} \int \ell_y(x, i; \theta) p_{i\mathrm{S}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{MS}} \sum_{j \in M} \int (1 + \hat{w}_y(x)) \ell_y(x, j; \theta) p_{j\mathrm{S}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{AS}} \int \ell_y(x, A; \theta) p_{\mathrm{AT}}(x) \mathrm{d}x. \end{aligned} \tag{23}$$

For the first and second terms in the right-hand side of Eq. (23), we can easily calculate ℓ_y, because the labels are provided in the source data. However, for the third term, since there are no labels in the target data, we cannot calculate ℓ_y in the same way as that in the first and second terms. Considering that ℓ_y is the cross entropy loss, we adopt simple extension of the loss function as

$$\ell_y(x, A; \theta) = -\log \sum_{y \in A} \hat{p}(y|x; \theta), \tag{24}$$

where $\hat{p}(y|x; \theta)$ is the output of the classifier when the input is x and the parameters of DNNs are set to θ.

3.4. Implementation details

Our partially zero-shot domain adaptation adopts the instance weight to calculate the losses, and the weight is calculated by taking the ratio between the outputs of the classifier or those of the domain discriminator as shown in Eqs. (12) and (16), respectively. In the early stage of training DNNs, these outputs are often inaccurate, and this error would be magnified by taking the ratio, which results in quite inaccurate estimation of the weight that can severely damage the training process. Since evaluating how accurate the weights are is hard, we suppress the influence of the weight to alleviate this problem by two heuristic techniques. For simplicity, we explain these techniques for the binary classification case, but the same ones can be also applied in the multi-class classification case. First, to avoid extremely large weights, we clip the weight to be less than a certain upper bound:

$$\hat{w}_d(x) = \min\left[\alpha, \frac{p(y=\mathrm{P}|d=\mathrm{S})}{p(y=\mathrm{N}|d=\mathrm{S})} \cdot \frac{\hat{p}(y=\mathrm{N}|x;\theta)}{\hat{p}(y=\mathrm{P}|x;\theta)+\epsilon}\right], \quad (25)$$

$$\hat{w}_y(x) = \min\left[\alpha, \frac{\hat{p}(d=\mathrm{T}|x;\theta)}{\hat{p}(d=\mathrm{S}|x;\theta)+\epsilon}\right], \quad (26)$$

where α represents the upper bound of the weight. Here, a small positive constant ϵ is also introduced to avoid computational instability. In this paper, we set $\alpha = 5$ and $\epsilon = 0.01$, and they are fixed in all experiments. Second, to suppress the influence of the weighted losses, we reduce the contribution of the loss that stems from the target negative data by using $\beta\hat{w}_d(x)$ and $\beta\hat{w}_y(x)$ instead of $\hat{w}_d(x)$ and $\hat{w}_y(x)$ to calculate L_d^w in Eq. (13) and L_y^w in Eq. (17), respectively. The coefficient β ($0 < \beta \leq 1$) represents the confidence of the loss from the target negative data, and we set it to $1/\alpha$ in all experiments, which means that we allow the influence of the loss from the unseen target-negative data at most the same as that from the available target-positive data.

4. Experiments

We conducted experiments with several datasets that are commonly used to benchmark domain adaptation methods in existing works. Specifically, we used digit classification datasets (MNIST [13], MNIST-M [8], and SVHN [15]) and an object recognition dataset (Office-31 dataset [19]). To make partially zero-shot setting, we chose a certain number of missing classes from the dataset with alphabetical order of class name and did not use corresponding target data while training. Note that the missing-class data do not appear in the training data of the target domain but are included in the test data. After training, we applied the trained classifier to the test data of the target domain and evaluated its accuracy to compare the performance of the domain adaptation methods.

For comparison, we used DNN trained with only source data (without domain adaptation) and DANN [8] as baseline methods and Partial Adversarial Domain Adaptation (PADA) [4] as a state-of-the-art partial domain adaptation method that are most related to our work. In PADA, the class-wise instance weight is set to be proportional to the average of the classifier predictions over the target data and is used when calculating the losses. This instance weighting enables one to ignore the missing-class data, because the missing class would be rarely predicted. Since PADA ignores the missing-class data in the classification loss as well as in the domain discrimination loss, it would result in poor classification performance in our scenario in which the missing class will appear in test data. Therefore, we also

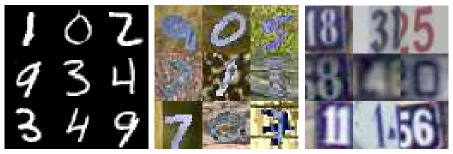

(a) MNIST (b) MNIST-M (c) SVHN

Figure 3. Example images of MNIST, MNIST-M, and SVHN.

Table 1. The network architecture used for the domain adaptation from MNIST to MNIST-M. MP2, BN, and FC denote 2×2 max-pooling, batch normalization, and a fully-connected layer, respectively.

Feature extractor	
Layer type	Filter size / # Filters
conv. + ReLU	5×5 / 32
MP2 + BN	2×2 / 32
conv. + ReLU	5×5 / 48
MP2 + BN	2×2 / 48

Classifier	
FC + ReLU	1 / 100
FC + ReLU	1 / 100
FC + softmax	1 / 10

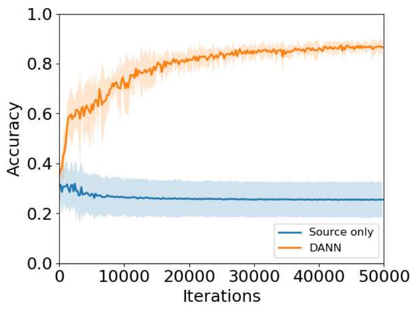
Domain discriminator	
FC + ReLU	1 / 100
FC + sigmoid	1 / 1

compared a variant of PADA that ignores the missing class only in the domain discrimination loss but does not ignore it in the classification loss. This method is referred to as the PADA-classifier in [4]. For each method, we conducted experiments five times with random initialization of DNNs and will report their averaged accuracy.

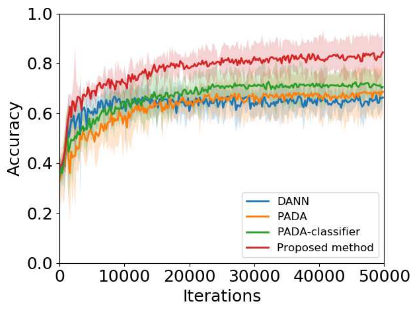
Another possible choice of partial domain adaptation methods is Importance Weighted Adversarial Nets (IWAN) [24]. However, IWAN is not essentially suitable for partially zero-shot setting. Since IWAN adopts two separate feature extractors for each domain, the missing class data are never learned while training of the target-side feature extractor. It results in poor classification performance in our scenario, and we cannot easily avoid this problem. Therefore, we did not include IWAN in our experiment.

4.1. Digit classification

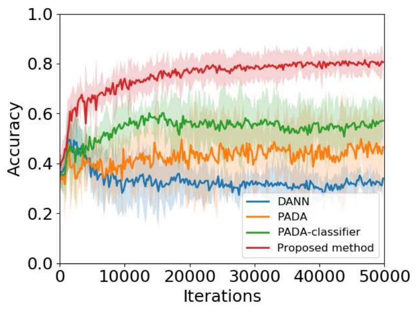
MNIST, MNIST-M, and SVHN are digit image datasets, and the task is to classify these images into ten classes that correspond to digit numbers. Example images of each dataset are shown in Fig. 3. In the experiments, we tried two popular domain adaptation scenarios: from MNIST to MNIST-M and from SVHN to MNIST. Since MNIST-M images are made by artificially synthesizing MNIST images with background extracted from natural images, domain adaptation between MNIST and MNIST-M is relatively easy. On the other hand, SVHN and MNIST images are collected at totally different environment. Therefore, these images have largely different appearance to each other, which makes the domain adaptation harder. We almost followed



(a) No missing classes.



(b) Three classes are missing.



(c) Seven classes are missing.

Figure 4. Accuracy on the test data of the target domain while training adaptation from MNIST to MNIST-M. Solid line represents the averaged accuracy, and shaded area represents $\pm 2\sigma$.

Table 2. The network architecture used for the domain adaptation from SVHN to MNIST. MP3s2, BN, and FC denote 3×3 max-pooling with stride 2, batch normalization, and a fully-connected layer, respectively.

Feature extractor	
Layer type	Filter size / # Filters
BN + conv. + ReLU	5×5 / 64
MP3s2 + BN	3×3 / 64
conv. + ReLU	5×5 / 64
MP3s2 + BN	3×3 / 64
conv. + ReLU + BN	5×5 / 128

Classifier	
FC + ReLU	1 / 500
FC + ReLU	1 / 500
FC + softmax	1 / 10

Domain discriminator	
FC + ReLU	1 / 100
FC + sigmoid	1 / 1

Table 3. Experimental results of domain adaptation from MNIST to MNIST-M.

	The number of missing classes				
	0	3	5	7	9
Source only	26.7%	-	-	-	-
DANN	85.2%	64.9%	51.7%	32.6%	24.7%
PADA	-	65.5%	58.8%	50.0%	27.1%
PADA-classifier	-	70.1%	57.9%	56.4%	36.8%
Proposed method	-	**85.5%**	**85.4%**	**81.0%**	**52.2%**

Table 4. Experimental results of domain adaptation from SVHN to MNIST.

	The number of missing classes				
	0	3	5	7	9
Source only	63.4%	-	-	-	-
DANN	67.8%	66.5%	67.6%	**66.7%**	62.5%
PADA	-	54.7%	58.3%	59.9%	49.0%
PADA-classifier	-	66.0%	66.6%	64.8%	**64.1%**
Proposed method	-	**67.2%**	**69.0%**	58.1%	62.2%

the same setup for the experiments in [8]. The network architectures are shown in Table 1 and 2, respectively. These networks are trained by the stochastic gradient descent with momentum. The size of mini-batch was set to 128. Its half is for the source data, and the rest is for the target data. The momentum was set to 0.9, and the learning rate was decayed while training as $\mu = \mu_0 / (1 + \alpha p)^{\beta}$, where p is the training progress linearly changing from 0 to 1, $\mu_0 = 0.01$, $\alpha = 10$, and $\beta = 0.75$. The hyper-parameter λ in Eq. (18) was set to 1, but, for training feature extractor, we adopt gradually changing λ_p as $\lambda_p = 2/(1 + \exp(-\gamma p)) - 1$, where γ was set to 10. For fair comparison, we used doubled value of the above equation as λ in our method to match the contribution of the original discrimination loss of the target data between our method and the other methods. For the partially zero-shot setting, we evaluated the accuracy with varying the number of missing classes.

Table 3 shows the experimental results of the adaptation from MNIST to MNIST-M. When the number of the missing classes is zero, which corresponds to the standard setting of existing domain adaptation methods, DANN works well and substantially increases the accuracy from 26.7% to 85.2%. As increasing the number of the missing classes, the performance of DANN becomes deteriorated and even worse than that of source only case when nine classes disappear in the target data. This is because DANN hopelessly tries to match the distribution of the source data with that of the target data as a whole even though the source data of the missing class do not have appropriate data to be matched due to lack of the missing class in the target data. On the other hand, PADA and PADA-classifier actively ignore the missing class while domain adaptation, which results in better performance than DANN under the partially zero-shot setting. PADA-classifier performs better than PADA, because it does not ignore the missing-class data when calculating the classification loss. Compared with these methods, our method achieves much better performance, because it explicitly considers the loss of unseen target data of the missing classes for domain adaptation, while the other

Figure 5. Example images of Office-31 dataset.

methods do not. Amazingly, the proposed method keeps almost same accuracy with the standard adaptation setting even when a half of classes do not appear in the target data.

The test accuracy while training is shown in Fig. 4. When there is no missing class, the test accuracy of DANN gradually increases while training and converges to a certain level in the end as shown in Fig. 4(a). On the other hand, in partially zero-shot setting, the accuracy of DANN does not substantially increase and even decrease when seven classes are missing. PADA works slightly better than DANN, but its accuracy fluctuates when the number of missing classes is large as shown in Fig. 4(c). This is because PADA tries to ignore the missing-class data while training, which results in instable accuracy on the missing classes in the test data. Compared with PADA, the proposed method show better and more stable performance due to considering the missing class in the calculation of the losses. Specifically, the proposed method behaves almost same as DANN with no missing classes, even when seven classes are missing.

Table 4 shows the results of the adaptation from SVHN to MNIST. All methods except for our method behaved similar to those in the previous experiment, but our method gets relatively worse when the number of the missing classes becomes large. Considering that PADA-classifier that adopts the class posterior to calculate the loss works well, this is probably because the estimation error of the domain posterior becomes quite large due to the large discrepancy between SVHN and MNIST as well as a large amount of missing-class data. In addition, our assumption on the class prior in Eq. (12) is violated in this experiment. The number of training data is almost balanced between classes in MNIST, but it largely varies in SVHN, which results in the change of the class prior.

4.2. Object recognition

The Office-31 dataset is one of the most popular datasets specialized for benchmarking domain adaptation methods. This dataset contains object images with 31 categories, and three domains are defined: Amazon, DSLR, and webcam. Example images in each domain are shown in Fig. 5. We can consider six scenarios of domain adaptation in this dataset, but we focus on the most difficult two scenarios that are from Amazon to webcam and from Amazon to DSLR. This is because DSLR and webcam are relatively similar to each other and the effect of the original DANN is marginal (e.g. $96.1\% \rightarrow 96.4\%$ in the case of DSLR-to-webcam reported in [8]). We also follow the same experimental setup with [8]. We used AlexNet pretained with ImageNet [12] to construct the initial feature extractor by removing the output layer and adding the 256-dimensional bottleneck. For the classifier and the domain discriminator, we used a single fully-connected layer ($256 \rightarrow 31$) and a fully-connected network with two hidden layers ($256 \rightarrow 1024 \rightarrow 1024 \rightarrow 1$). The learning rate for the pretrained layers is set to be ten times smaller than the other layers. The other setting is same as that in the previous experiment.

Table 5. Experimental results of domain adaptation from Amazon to webcam in Office-31 dataset.

	The number of missing classes		
	0	10	20
Source only	61.5%	-	-
DANN	70.1%	61.3%	52.4%
PADA	-	62.9%	31.6%
PADA-classifier	-	65.6%	55.3%
Proposed method	-	**68.2%**	**63.5%**

Table 6. Experimental results of domain adaptation from Amazon to DSLR in Office-31 dataset.

	The number of missing classes		
	0	10	20
Source only	66.3%	-	-
DANN	71.1%	62.2%	54.2%
PADA	-	64.2%	23.4%
PADA-classifier	-	67.9%	59.9%
Proposed method	-	**69.2%**	**70.0%**

Table 5 and 6 show the results with Office-31 dataset. As in the previous experiment, the proposed method achieves substantially better performance than the other methods. Even when a large proportion of the classes, such as twenty out of thirty one, does not appear in the target data, our method achieves better performance than source only case and does not fall into negative transfer.

5. Conclusion

We proposed a novel domain adaptation method designed for partial zero-shot scenario. In this scenario, a certain subset of classes only appear in the source data but do not appear in the target data, while we want to discriminate all classes at the target data after the domain adaptation. We derived how to estimate losses of unseen missing-class target data by adopting the instance weights that are estimated based on the outputs of DNNs. In the experiments, our method has shown excellent performance under partially zero-shot setting compared with existing domain adaptation methods.

Acknowledgements MS was supported by KAKENHI 17H01760. TT was supported by KAKENHI 16K00051.

References

[1] M. Baktashmotlagh, M. Faraki, T. Drummond, and M. Salzmann. Learning factorized representations for open-set domain adaptation. In *International Conference on Learning Representations*, 2019.

[2] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351. 2016.

[3] Z. Cao, M. Long, J. Wang, and M. I. Jordan. Partial transfer learning with selective adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2724–2732, 2018.

[4] Z. Cao, L. Ma, M. Long, and J. Wang. Partial adversarial domain adaptation. In *European Conference on Computer Vision*, pages 139–155, 2018.

[5] Q. Chen, Y. Liu, Z. Wang, I. Wassell, and K. Chetty. Re-weighted adversarial adaptation network for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7976–7985, 2018.

[6] G. Csurka, editor. *Domain Adaptation in Computer Vision Applications*. Advances in Computer Vision and Pattern Recognition. Springer, 2017.

[7] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *IEEE International Conference on Computer Vision*, pages 2960–2967, 2013.

[8] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[9] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, 2012.

[10] W. G. Hatcher and W. Yu. A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access*, 6:24411–24432, 2018.

[11] T. Ishida, G. Niu, and M. Sugiyama. Binary classification from positive-confidence data. In *Advances in Neural Information Processing Systems*, pages 5921–5932. 2018.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105. 2012.

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

[14] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *International Conference on Machine Learning*, pages 2208–2217, 2017.

[15] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[16] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[17] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

[18] P. Panareda Busto and J. Gall. Open set domain adaptation. In *IEEE International Conference on Computer Vision*, pages 754–763, 2017.

[19] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision*, pages 213–226, 2010.

[20] K. Saito, Y. Ushiku, and T. Harada. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997, 2017.

[21] K. Saito, S. Yamamoto, Y. Ushiku, and T. Harada. Open set domain adaptation by backpropagation. In *European Conference on Computer Vision*, pages 153–168, 2018.

[22] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[23] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2962–2971, 2017.

[24] J. Zhang, Z. Ding, W. Li, and P. Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8156–8164, 2018.

[25] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827, 2013.