Unsupervised Domain Adaptation
Based on the Predictive Uncertainty of Models

JoonHo Lee\textsuperscript{a,b}, Gyemin Lee\textsuperscript{a}

\textsuperscript{a}Department of Electronic and IT Media Engineering, Seoul National University of Science and Technology, Republic of Korea
\textsuperscript{b}Machine Learning Research Center, Samsung SDS Technology Research, Republic of Korea

Abstract

Unsupervised domain adaptation (UDA) aims to improve the prediction performance in the target domain under distribution shifts from the source domain. The key principle of UDA is to minimize the divergence between the source and the target domains. To follow this principle, many methods employ a domain discriminator to match the feature distributions. Some recent methods evaluate the discrepancy between two predictions on target samples to detect those that deviate from the source distribution. However, their performance is limited because they either match the marginal distributions or measure the divergence conservatively. In this paper, we present a novel UDA method that learns domain-invariant features that minimize the domain divergence. We propose model uncertainty as a measure of the domain divergence. Our UDA method based on model uncertainty (MUDA) adopts a Bayesian framework and provides an efficient way to evaluate model uncertainty by means of Monte Carlo dropout sampling. Empirical results on image recognition tasks show that our method is superior to existing state-of-the-art methods. We also extend MUDA to multi-source domain adaptation problems.

Keywords: unsupervised domain adaptation, model uncertainty, predictive variance, Monte Carlo dropout, image classification

1. Introduction

Deep neural networks (DNNs) have shown great success in numerous image processing and computer vision tasks thanks to large amounts of well-annotated training data. However, DNN models often fail to work well in real world applications due to the distribution shift of the target data from the data used to train the models. Collecting new large-scale labeled data for the target task is prohibitively expensive and time-consuming in many cases.

Domain adaptation (DA) addresses this domain shift problem by adapting a model trained on a source domain to a target domain. In particular, unsupervised DA (UDA) assumes that only unlabeled samples are available from the source domain.
target domain, whereas labels are available for samples from the source domain. We present a new UDA method in this paper.

While a number of UDA approaches have been proposed, many recent advances conform to the theory proposed by Ben-David et al. [1]. Ben-David et al. formulated the domain shift in terms of domain divergence. More specifically, they demonstrated that the target error is bounded by the source error and the divergence between the two domains. Hence, to realize a successful adaptation of a source-trained model to the target domain, small source-target divergence is essential.

In this regard, the majority of modern DNN-based UDA methods [2, 3, 4, 5] work by learning the features that minimize the domain divergence. To achieve this goal, they typically employ a domain discriminator that separates the source against the target samples. This domain discriminator empirically measures the domain divergence using unlabeled data from both domains. In these methods, a feature extractor is trained in an adversarial manner to maximize the confusion of the domain discriminator and to align the distributions of features between the source and the target domains. The underlying assumption is that once domain-invariant features are learned, a model trained on source data will correctly classify the target data. However, these approaches show limited performance in reducing the target error because they inherently focus on matching marginal distributions from both domains. Though the domain discriminator is useful in learning domain-invariant features, it lacks class-aware information [1, 6]. A recent study [7] also suggests that the target error can increase when the label distributions of both domains differ significantly.

To resolve these issues, we take class-aware information into consideration for domain adaptation. In this work, we propose predictive uncertainty as a domain divergence measure. Our intuition is that less discriminative target samples that cause domain discrepancy are likely to show higher predictive uncertainty when inferred with source-only models. Thus, if we can train a feature extractor that minimizes model uncertainty on target samples, it will generate features that are more consistent across different domains. This process is illustrated in Fig. 1. We justify our approach by associating the model uncertainty with the classifier-induced domain divergence. According to [1], this divergence provides a theoretically tighter bound than domain discriminator-based divergence.

A related study [6] proposed the disagreement between two task-specific classifiers as a measure of domain divergence. This method trains two classifiers on labeled source samples and computes their discrepancy with the $L_1$-distance.
between their softmax outputs for target samples. A feature generator is trained to minimize the maximum classifier discrepancy. A similar approach appears in [8], where a Gaussian process model was proposed, with the maximum classifier discrepancy translated into the maximum posterior separation. Variational inference is used to train a feature embedding network that maximizes the margin between top-1 and top-2 predictions.

However, these methods are overly conservative because domain divergence is measured with only two classifiers. A recent work [9] reports that using more classifiers improves UDA performance. Whereas two classifiers may fail to identify misaligned features, this failure is less likely with more classifiers. Therefore, we relax the restriction by considering the expectation over an approximately infinite number of classifiers. To this end, we adopt a Bayesian framework and define a distribution of classifiers given the source data. We show that this modified domain divergence is related to our model uncertainty. We also present Monte Carlo (MC) dropout sampling as an efficient way to assess this model uncertainty. Our approach is motivated by the recent interpretation of dropout as approximate Bayesian inference [10, 11]. Thus, we can evaluate the model uncertainty from multiple feedforward passes on the target samples.

We evaluate our method on various image recognition tasks to demonstrate its effectiveness. Our contributions are summarized below.

- We develop a novel interpretation of the model uncertainty as a measure of the divergence between domains and propose a new UDA method based on the model uncertainty (MUDA).
- We propose to use dropout as a training method under the Bayesian approximation scheme. Hence, the proposed MUDA can be applied to any DNN model that supports dropout.
- Our extensive experimental results demonstrate that MUDA achieves state-of-the-art outcomes on popular benchmark datasets. We also show that MUDA performs competitively on multi-source DA problems despite the simple source-combine strategy.
- We present several qualitative analyses to verify that the proposed method successfully captures discriminative features for the target tasks.

A preliminary version of this work appeared in [12]. With respect to [12], this work includes substantially extended experiments and analyses.

2. Background

2.1. Problem Statement

We are given fully-labeled data \((X_s, Y_s)\) from the source domain distribution \(D_S\) and unlabeled data \(X_t\) from the target domain distribution \(D_T\). The goal of UDA is to build a classifier that correctly predicts the label \(y_t \in \{1, \ldots, K\}\) of a new target sample \(x_t\). Both domains are assumed to share the same set of labels.
2.2. Related Work

**Ben-David Theorem** Ben-David et al. [1] proposed the theory that provides the upper-bound of the expected target error of a hypothesis \( \epsilon_T(h) \), using three terms: (i) the expected source error \( \epsilon_S(h) \); (ii) the \( \mathcal{H} \triangle \mathcal{H} \) divergence between the two domains; and (iii) a hypothesis-independent constant \( \lambda \), as follows:

\[
\forall h \in \mathcal{H}, \quad \epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} d_{\mathcal{H} \triangle \mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda
\]

where \( \mathcal{H} \) is a hypothesis space of classifiers and \( d_{\mathcal{H} \triangle \mathcal{H}} \) denotes the divergence in the symmetric difference hypothesis space. The constant \( \lambda = \epsilon_S(h^*) + \epsilon_T(h^*) \) represents the combined error of the ideal joint hypothesis \( h^* \) and is assumed to be small in UDA. This inequality suggests that minimizing the source error \( \epsilon_S \) and divergence \( d_{\mathcal{H} \triangle \mathcal{H}} \) is essential.

**Domain Adaptation** Based on the theory in [1], a number of current UDA studies have focused on reducing the domain divergence. Earlier works [13, 14] aligned the feature statistics of the source and the target domains. More recent developments adopted the adversarial learning strategy to transfer knowledge between domains, inspired by GAN [15]. These methods typically employ a domain discriminator to measure the difference between the two domains. DANN [2] is the first to introduce this domain classifier. To learn domain-invariant features, DANN trains its model with gradient-reversed backpropagation. ADDA [5] makes two domain-specific feature extractors which are trained to generate embeddings that confuse the domain discriminator. A large family of works, including JAN [16], MADA [17], and CADA [18], also train their models in an adversarial fashion. Saito et al. employ two instances of task-specific classifiers to use as a domain discriminator [6, 19]. Their method, MCDA, measures the domain divergence by the maximum discrepancy between two classifiers on the target samples. MCDA trains a feature generator to push the target feature distribution away from the decision boundary. GPDA(2019) [8] introduces the Gaussian Process (GP) model to extend earlier results [6] and reformulates the maximum classifier discrepancy principle into the maximum posterior separation. STAR [9] extends MCDA by modeling classifier weights with a Gaussian distribution where its variance represents the inter-classifier discrepancy. In other lines of approach, SAFN [20] argues that smaller feature norms cause model degradation and proposes to adapt the norms to a larger range. SWD [21] uses the Wasserstein distance in domain alignment. Optimal transport strategy is used in DeepJDOT [22] and RWOT [23]. DMRL [24] introduces mixup regularization to adversarial domain adaptation. GPDA(2021) [25] performs graph dual regularization to preserve data statistics and geometric properties. DMAT [26] adversarially trains a dual-module network to learn domain invariant features and domain discriminative features separately.

**Variational Inference** In a Bayesian framework, a model \( h \) such as the GP has a set of random parameters \( \omega \) with a prior distribution \( p(\omega) \). Given a dataset
(X, Y), the predictive distribution for a new sample x is given by

\[ p(y|x, X, Y) = \int p(y|x, \omega) p(\omega|X, Y) \, d\omega \]  

(2)

where \( p(\omega|X, Y) \) is the posterior distribution. However, exact evaluation of this posterior distribution is usually intractable. In variational inference, we approximate the posterior distribution \( p(\omega|X, Y) \) with a simpler distribution \( q(\omega) \). To make \( q(\omega) \) similar to \( p(\omega|X, Y) \), we minimize the Kullback-Leibler (KL) divergence \( KL(q(\omega)||p(\omega|X, Y)) \). The KL divergence minimization is equivalently achieved by maximizing the evidence lower bound

\[ \int q(\omega) \log p(Y|X, \omega) \, d\omega - KL(q(\omega)||p(\omega)) \]  

(3)

with respect to \( q(\omega) \) [27]. By replacing the true posterior distribution \( p(\omega|X, Y) \) with its variational approximation \( q(\omega) \), we obtain the approximate predictive distribution

\[ q(y|x) = \int p(y|x, \omega) q(\omega) \, d\omega. \]  

(4)

Gal et al. [10] have recently shown that dropout in DNNs is mathematically identical to approximate variational inference in the deep GP.

**Monte Carlo Dropout** In our method, it is necessary to evaluate the predictive uncertainty of a model on the target samples. Probabilistic Bayesian models can be used to estimate the model uncertainty, but these usually come with a prohibitive computational cost. It was recently shown that dropout is equivalent to an approximation to a deep Bayesian model. Whereas dropout is commonly used to prevent overfitting of a DNN model [28], Gal et al. established a theoretical interpretation of dropout as approximate Bayesian inference to the deep GP model [10, 11]. As a result, dropout can be used to obtain the model uncertainty. This technique, referred to as MC dropout, enables to estimate the model uncertainty using an ensemble of multiple stochastic feedforward passes.

### 3. Proposed Method

#### 3.1. Model Preliminaries

We formulate the UDA problem similarly to recent UDA methods. A feature extractor network \( F \) takes a sample \( x \) and produces a latent feature vector \( \phi = F(x) \). A task-specific classifier network \( C \) takes \( \phi \) and produces a \( K \)-dimensional output. \( F \) and \( C \) are shared by both the source and the target domains. Given \( x \), the classification outcome is \( \tilde{y} = C(F(x)) \), where \( \tilde{y} \) is a \( K \)-dimensional vector containing softmax scores. The decision vector determined by \( \tilde{y} \) is denoted by \( h(x) \).
3.2. Unsupervised Domain Adaptation Based on Model Uncertainty

A domain shift occurs when the source samples fail to represent the target samples. Because such target samples are less discriminative, their predictive uncertainty is likely to be high, as shown in Fig. 1b. High predictive uncertainty indicates that the target samples are outside of the source distribution. This leads to our idea that if we find a feature extractor $F$ that minimizes the predictive uncertainty, it will avoid generating target features outside of the source distribution.

The inequality (1) implies that reducing the source error $\epsilon_S$ and the divergence $d_{H\triangle H}$ is essential for successful UDA. We describe how we accomplish this goal by reducing the model uncertainty. The divergence $d_{H\triangle H}$ in (1) is defined as follows:

\[
d_{H\triangle H}(D_S, D_T) = 2 \sup_{h, h' \in H} \left| \mathbb{E}_{x \sim D_S} [\mathbb{1}(h(x) \neq h'(x))] - \mathbb{E}_{x \sim D_T} [\mathbb{1}(h(x) \neq h'(x))] \right|.
\]

(5)

If $h$ and $h'$ can correctly classify the source samples, we can consider that they will agree on the source samples. This enables us to safely neglect the term $\mathbb{E}_{x \sim D_S} [\mathbb{1}(h(x) \neq h'(x))]$.

Because $\mathbb{1}(h(x) \neq h'(x)) = (h(x) - h'(x))^2$ for the binary classification $h(x) \in \{0, 1\}$, we can approximate $d_{H\triangle H}$ by

\[
d_{H\triangle H}(D_S, D_T) \approx 2 \sup_{h, h' \in H} \mathbb{E}_{x \sim D_T} [\mathbb{1}(h(x) \neq h'(x))]
\]

(6)

\[
= 2 \sup_{h, h' \in H} \mathbb{E}_{x \sim D_T} [(h(x) - h'(x))^2].
\]

(7)

Hence, the UDA is simplified to the problem of minimizing the supremum of the expected disagreement between two hypotheses on the target samples. Though this objective involves a supremum over all hypothesis functions in $H$, we argue that we can achieve the same goal by narrowing our attention to the set of hypotheses that minimizes $\epsilon_S$.

To this end, we follow the Bayesian approach and define the posterior distribution $D_H = p(h|X_s, Y_s)$ of $h$ conditioned on the labeled source samples. We also manipulate equation (7) by replacing the supremum with the expectation with respect to the posterior $D_H$ to obtain (see Appendix)

\[
2 \mathbb{E}_{h, h' \sim D_H} \mathbb{E}_{x \sim D_T} [(h(x) - h'(x))^2]
\]

(8)

\[
= 4 \mathbb{E}_{x \sim D_T} \mathbb{E}_{h \sim D_H} [(h(x) - \mathbb{E}_{h \sim D_H}[h(x)])^2].
\]

(9)

Therefore, we can reformulate the problem of minimizing the divergence $d_{H\triangle H}$ into minimizing the predictive variance of a hypothesis (model uncertainty) on target samples. This replacement will no longer ensure the modified divergence in equation (9) as an upper bound of the target error. However, we can easily imagine that minimizing the mean hypothesis disagreement will lead to similar consequences as minimizing the supremum hypothesis disagreement. In section
we provide more discussion of this reformulation. Consequently, our objective is to find a feature extractor $F$ that minimizes this model uncertainty:

$$\min_F E_{x \sim D_T} E_{h \sim D_H} [(h(x) - E_{h \sim D_H} [h(x)])^2]$$  \hspace{1cm} (10)$$

while minimizing the source error $\epsilon_S$:

$$\min_{F,C} \epsilon_S(h).$$  \hspace{1cm} (11)$$

We note that the extension to $K$-way classification is straightforward, where $h(x)$ is a $K$-dimensional decision vector.

3.3. Model Uncertainty Loss

Though we can model the posterior $D_H = p(h|X_s, Y_s)$ with a GP, its evaluation is often intractable. Variational inference instead defines an approximate posterior $q(\omega)$ and yields an approximate predictive distribution for a new target sample $x \sim D_T$, as follows:

$$q(y|x) = \int p(y|x, \omega) q(\omega) \, d\omega$$  \hspace{1cm} (12)$$

where $\omega$ is a set of random parameters for $h$.

Since dropout applied to a DNN is shown to be equivalent to the approximate variational inference of the deep GP [10], we can evaluate the predictive variance in equation (10) by means of MC dropout sampling. By performing $M$ stochastic forward passes through the DNN for $h$ (equivalently, $F$ and/or $C$), we obtain $\{\tilde{y}_1 \ldots \tilde{y}_M\}$, where $\tilde{y}_m$ is a $K$-dimensional softmax score. These computations can be done in parallel. Our estimate of the predictive variance then becomes

$$\hat{\sigma}^2_{\text{MC}}(x) = \text{diag}(\frac{1}{M} \sum_{m=1}^M \tilde{y}_m \tilde{y}_m^\top - \overline{y}_{\text{MC}} \overline{y}_{\text{MC}}^\top),$$  \hspace{1cm} (13)$$

where $\overline{y}_{\text{MC}} = \frac{1}{M} \sum_{m=1}^M \tilde{y}_m.$  \hspace{1cm} (14)$$

Therefore, we define our model uncertainty loss with the norm of this predictive variance:

$$L_{\text{div}}(D_T) = E_{x \sim D_T} ||\hat{\sigma}_{\text{MC}}(x)||$$  \hspace{1cm} (15)$$

as an approximation to the divergence $d_{H\triangle H}$.

3.4. Classification Loss

Keeping the source error $\epsilon_S(h)$ minimized is essential for successful UDA. To measure the source error, we use the cross entropy loss

$$L_{\text{cls}}(D_S) = -E_{(x_s,y_s) \sim D_S} [y_s^\top \log C(F(x_s))]$$  \hspace{1cm} (16)$$
Algorithm 1 Learning algorithm for MUDA

\textbf{Input:} Labeled source data \((X_s, Y_s)\); unlabeled target data \(X_t\); MC dropout sample size \(M\); MC dropout rate \((\rho_F, \rho_C)\).

\textbf{Output:} Domain-adapted feature extractor \(F\) and classifier \(C\).

1: Learn initial weights of \(F\) and \(C\) on \((X_s, Y_s)\).
\quad \quad \quad F, C \leftarrow \text{arg min}_{F, C} L_{\text{cls}}(D_S)

2: \textbf{repeat}
3: \quad Sample mini-batch \((X_s, Y_s)^{\text{mini}}\) from \((X_s, Y_s)\).
4: \quad Sample mini-batch \(X_{t}^{\text{mini}}\) from \(X_t\).
5: \quad Compute \(L_{\text{cls}}(D_S)\) on \((X_s, Y_s)^{\text{mini}}\) by equation \([16]\).
6: \quad \textbf{for} \(m = 1\) to \(M\) \textbf{do}
7: \quad \quad Perform stochastic feedforward on \(X_{t}^{\text{mini}}\).
\quad \quad \quad \hat{Y}_{t}^{\text{mini}} \leftarrow C(F(X_{t}^{\text{mini}}))
8: \quad \textbf{end for}
9: \quad Deactivate dropout of \(F\) and \(C\).
10: \quad Compute \(\hat{\sigma}_{MC}^2\) using \(\hat{Y}_{t}^{\text{mini}}, \ldots, \hat{Y}_{t}^{\text{mini}}\) by equation \([13]\).
11: \quad Compute \(L_{\text{div}}(D_T)\) on \(X_{t}^{\text{mini}}\) by equation \([15]\).
12: \quad Update \(C\): \quad \quad \quad C \leftarrow \text{arg min}_C L_{\text{cls}}(D_S)
13: \quad Update \(F\): \quad \quad \quad F \leftarrow \text{arg min}_F L_{\text{cls}}(D_S) + L_{\text{div}}(D_T)
14: \textbf{until} \text{Stopping conditions are satisfied}

for our \(K\)-way classification task. Here, \(y_s\) denotes the one-hot encoded label vector of a source sample \(x_s\).

We note that this task-specific loss can be replaced according to the purpose of the given task. For example, one might use the mean squared loss for a regression task or the pixel-wise cross entropy loss for a semantic segmentation task.

3.5. Optimization Strategy

As described above, our \(d_{H\Delta H}\)-divergence approximation and the source error in equations \([10]\) and \([11]\) can be evaluated with the model uncertainty loss \(L_{\text{div}}(D_T)\) and the cross entropy loss \(L_{\text{cls}}(D_S)\), respectively. This leads to two optimization problems:

\begin{itemize}
  \item \(\text{min}_C L_{\text{cls}}(D_S)\) \hfill (17)
  \item \(\text{min}_F L_{\text{cls}}(D_S) + L_{\text{div}}(D_T)\). \hfill (18)
\end{itemize}

Therefore, our domain adaptation method based on model uncertainty (MUDA) alternates to train \(F\) and \(C\). The proposed method is outlined in Algorithm 1.

4. Experiments

To demonstrate our approach, we conduct experiments on various datasets. A proof-of-concept experiment on a toy dataset (section 4.1) is followed by an
extensive evaluation on the digits and traffic signs (section 4.2), the Office-31 (section 4.3), and the VisDA-17 (section 4.4) datasets. Fig. 2 shows a few sample images from each of these benchmark datasets.

We extend our UDA method to problems in which more than one source domain dataset is available. Digits-five is a collection of digit images from five different domains, and miniDomainNet [29] contains object images from four different domains. MUDA shows competitive results compared to recent multi-source domain adaptation methods (section 4.5).

We also qualitatively analyze our method by visualizing the changes of feature embeddings and by comparing class activation maps with or without MUDA (section 5). All experiments are implemented using PyTorch [30]. Our code is available at [https://github.com/joonholee-research/MUDA](https://github.com/joonholee-research/MUDA).

### 4.1. Proof of Concept

**Setup** We conduct a proof-of-concept experiment on the interleaving two-moons dataset. Fig. 3 shows the labeled source samples (red, green) and the unlabeled target samples (blue). The target distribution is obtained by rotating the source distribution by 30°. For the experiment, we generate 1000 source and target samples, respectively, and split them into 500 training samples and 500 testing samples for each domain. The test samples from both domains are depicted in Fig. 3.

We use a five-layer fully-connected (FC) network with 15 hidden neurons for every hidden layer. Each of the first three layers is followed by batch normalization, and the fourth layer is set to dropout at a rate 0.5. ReLU is used for activation. The mini-batch size is 128.

**Results** Fig. 3 compares the model adapted using our MUDA method with a model trained only on source samples. The decision boundary of the source-only model in Fig. 3 crosses the support of the target distribution and misclassifies many target samples. On the other hand, the proposed MUDA adapts the model to the target samples and successfully separates them with high accuracy as shown in Fig. 3. MUDA achieves this by finding the decision boundary that minimizes the predictive uncertainty in the target domain. Note that the decision boundary is moved away from the target samples. Fig. 3 illustrates
Figure 3: (Best viewed in color) Red and green dots are color-labeled source samples, and blue dots represent unlabeled target samples. The black curve represents a decision boundary. The pink and light green areas are the decision regions of the classifier. (a) shows the domain shift. The adapted decision boundary in (b) correctly classifies target samples. (c) displays multiple boundaries from MC dropout in our MUDA method. Note that accumulated boundary is thicker near the target samples far from the source distribution.

how MUDA measures the predictive uncertainty. Twenty classifiers from MC dropout sampling are overlaid. The accumulated boundary is thicker where the boundary is closer to the target samples and farther from the source samples, indicating that the predictive uncertainty is higher.

4.2. Digits and Traffic Signs Datasets

Setup We compare the proposed method with recent UDA methods in the literature on standard benchmark datasets. The digit classification task consists of three datasets: MNIST [31], SVHN [32], and USPS [33]. Its aim is to classify an image into one of ten digit classes. We also conduct an evaluation with a traffic sign classification task. SYNSIG [34] and GTSRB [35] contain 43 types of synthetic and actual signs, respectively.

We utilize the experimental setup with the dataset splits and the network architectures of prior works [2, 5, 6] for a fair comparison. For MNIST* → USPS*, we use all of the target samples during training. For all other tasks, we split both the source and the target samples into the training and test datasets.

Three convolution layers for $F$ and two FC layers for $C$ are used with intermediate dropout layers. The input image size is set to 28×28 for MNIST → USPS, 32×32 for SVHN → MNIST, and 40×40 for SYNSIG → GTSRB. The Adam [36] optimizer is used with a learning rate $2.0 \times 10^{-4}$ and weight decay $5.0 \times 10^{-4}$. The dropout rate is set to 0.4 for $C$ and 0.1 for $F$. The mini-batch size is 64.

Results Table 1 summarizes the results on the digits and traffic signs datasets. The mean and the standard deviation of ten independent experiments are reported. The table also compares recent state-of-the-art UDA methods. As can be seen, MUDA exhibits superior performance for all tasks. The improvement over the source-only model ranges from 13.5% to 33.3%. In particular on a difficult task, SVHN → MNIST, MUDA increases the accuracy by more than 30%. Whereas no labels are used, the performance is nearly perfect. Though some methods like STAR [9] and RWOT [23] show slightly higher accuracies for USPS → MNIST, the differences are not significant.
Table 1: Classification accuracy (%) on the digits and traffic signs datasets (S: SVHN, M: MNIST, U: USPS, S → G: SYN SIG → GTSRB). The results are cited from each study. We indicate the best in **bold** and the second best in *bold italic*.

| Methods | S → M | M → U | M* → U* | U → M | S → G |
|---------|-------|-------|----------|-------|-------|
| Source-only | 67.1  | 76.7  | 79.4  | 63.4  | 85.1  |
| DAN [14] | 71.1  | 81.1  | 91.1  |       |       |
| DANN [2] | 71.1  | 77.1   | 85.1  | 73.0   | 88.7  |
| DSN [17] | 82.7  | 91.3  | -     | -     | 93.1  |
| ADDA [5] | 76.0   | 89.4   | -     | 90.1   | -     |
| G2A [38] | 92.4  | 92.8  | 95.3  | 90.8  | -     |
| MCDA [6] | 96.2   | 94.2   | 94.1  | 94.4   | 94.3  |
| GPDA(2019) [8] | 82.7  | 91.3  | -     | -     | 93.1  |
| ADDA [5] | 76.0   | 89.4   | -     | 90.1   | -     |
| MUDA (ours) | 99.1  | 97.9  | 98.5  | 96.7  | 98.6   |

Table 2: Classification accuracy (%) on Office-31 (A: Amazon, W: Webcam, D: DSLR). The results are cited from each study. **Bold** is the best and *bold italic* is the second best.

| Methods | A → W | D → W | W → D | A → D | D → A | W → A | Average |
|---------|-------|-------|-------|-------|-------|-------|---------|
| Source-only | 68.4  | 96.7  | 99.3  | 68.9  | 62.5  | 60.7  | 76.1   |
| DAN [2]  | 82.0  | 96.9  | 99.1  | 79.7  | 68.2  | 67.4  | 82.2   |
| JAN [16] | 86.2  | 96.2  | 98.4  | 77.8  | 69.5  | 68.9  | 82.8   |
| MUDA (ours) | 90.0  | 97.4  | 99.6  | 87.8  | 70.3  | 66.4  | 85.3   |
| GPDA(2019) [8] | 83.9  | 97.3  | 100.0 | 85.5  | 72.3  | 68.8  | 84.6   |
| SAFN [20] | 88.8  | 98.4  | 99.8  | 87.7  | 69.8  | 69.7  | 85.7   |
| GPDA(2021) [25] | 87.4  | 98.4  | 99.4  | 85.8  | 70.6  | 74.1  | 86.1   |

4.3. Office-31 Dataset

**Setup** The Office-31 dataset [39] contains 4,652 images across 31 categories collected from three different domains: *Amazon* (A), *Webcam* (W), and *DSLR* (D). Whereas each image is larger, the dataset is smaller than the digits and traffic signs datasets. Because the Office-31 dataset is relatively small, we evaluate MUDA fully transductively as in previous works [16, 5, 8]. We use all labeled source samples and all unlabeled target samples for training.

We employ the pre-trained ResNet-50 [40] as our feature extractor $F$ and FC layers with 1,000 neurons as the classifier $C$. Every image is resized to $256 \times 256$, randomly flipped, and then cropped on center to $224 \times 224$. Images are standardized using ImageNet [41] statistics before they are fed into the DNNs. We optimize using Adam with a learning rate $2.0 \times 10^{-5}$, weight decay $5.0 \times 10^{-4}$, and mini-batch size 32. The dropout is applied only to $C$ at a fixed rate of 0.4.

**Results** The results are presented in Table 2. We find that MUDA outper-
Table 3: Classification accuracy per category (%) on VisDA-17. The results are cited from each study. The mean is reported from 5 random runs, and each run updates for 5 epochs. We indicate the best in **bold** and the second best in **bold italic**.

| Methods       | plane | bcycl | bus  | car  | horse | knife | mcycl | person | plant  | sktbrd | train  | truck  | Average |
|---------------|-------|-------|------|------|-------|-------|-------|--------|--------|--------|--------|--------|--------|
| Source-only   | 55.1  | 53.3  | 61.9 | 50.1 | 80.6  | 17.9  | 79.7  | 31.2   | 81.0   | 26.5   | 73.5   | 8.5    | 52.4   |
| DAN[14]       | 87.1  | 63.0  | 76.5 | 42.0 | 90.3  | 42.9  | 85.9  | 53.1   | 49.7   | 36.3   | 85.8   | 20.7   | 61.1   |
| DANN[2]       | 81.9  | 77.7  | 82.8 | 44.3 | 81.2  | 29.5  | 65.1  | 26.6   | 51.9   | 54.6   | 82.8   | 7.8    | 57.4   |
| MCDA[6]       | 87.0  | 60.9  | 83.7 | 64.0 | 88.9  | 79.6  | 84.7  | 76.9   | 86.6   | 40.3   | 83.0   | 25.8   | 71.9   |
| DeepJDOT[22] | 85.4  | 73.4  | 77.1 | 87.3 | 84.1  | 64.7  | 91.5  | 79.2   | 91.9   | 44.4   | 88.5   | 61.8   | 77.4   |
| GPDA(2019)[8] | 83.0  | 74.3  | 80.4 | 66.0 | 87.6  | 75.3  | 83.8  | 73.1   | 90.1   | 57.3   | 80.2   | 37.9   | 73.3   |
| SAFN[20]      | 93.6  | 61.3  | 84.1 | 70.6 | 94.1  | 79.0  | 91.8  | 79.6   | 89.9   | 55.6   | 89.0   | 24.4   | 76.1   |
| SWD[21]       | 90.8  | 82.5  | 81.7 | 70.5 | 91.7  | 69.5  | 86.3  | 77.5   | 87.4   | 63.6   | 85.6   | 29.2   | 76.4   |
| DMRL[24]      | -     | -     | -    | -    | -     | -     | -     | -      | -      | -      | -      | -      | 75.5   |
| DMAT+2M[26]  | 86.0  | 61.5  | 88.3 | 61.6 | 83.8  | 6.7   | 92.9  | 56.8   | 89.0   | 68.8   | 87.2   | 23.0   | 69.2   |
| MUDA (ours)   | 92.2  | 79.5  | 80.8 | 70.2 | 91.9  | 78.5  | 90.8  | 81.9   | 93.0   | 62.5   | 88.7   | 31.9   | 78.5   |

forms on this dataset as well. MUDA achieves the best or second best on most UDA tasks and records the highest overall average accuracy. Compared to the source-only model, the average accuracy across all tasks increases from 76.1% to 86.1%. For certain tasks such as Amazon → DSLR and Amazon → Webcam, the accuracy increases significantly by nearly 20%. Even when the number of source instances is not sufficiently large, as in DSLR → Amazon and Webcam → Amazon, the improvement exceeds 8%. Because Amazon is known to be substantially different from the other two domains[39], this performance gain indicates that MUDA is effective on challenging real-world adaptation tasks.

4.4. VisDA-17 Dataset

**Setup** The VisDA dataset[42] is designed to evaluate an adaptation from synthetic-object to real-object images. The source images are 3D object models rendered under different angles and different lighting conditions. This dataset contains 152,397 synthetic images across 12 categories. The target domain images are actual objects in the same categories. They are collected from MSCOCO[43] and number 55,388 in total. The image samples are displayed in Fig. 2. In our experiment, each image is randomly cropped and resized to 224×224 before being randomly flipped.

We use the pre-trained ResNet-101[40] for the feature extractor $F$. For the three FC layers of the classifier $C$, we set the number of hidden neurons to 1,000. We also incorporate the class balance loss into the objective functions to account for the class imbalance of the VisDA dataset. The configured setup is similar to[8] for a fair comparison. For optimization, we use SGD with a learning rate $10^{-4}$, weight decay $5.0 \times 10^{-4}$ and momentum 0.9. The mini-batch size is 32. The dropout rate of the classifier network $C$ is set to 0.4.

**Results** The adaptation results in Table 3 clearly demonstrate that the proposed MUDA greatly outperforms other UDA methods. Our method records the highest average per-category accuracy. In addition, MUDA achieves the best or the second best outcome for seven out of 12 categories. Even for the most challenging categories such as truck, knife, skateboard and person, the improvements over the source-only model exceed from 20% to 60%.
We find these results on the VisDA dataset encouraging. For many applications, adaptation from synthetic to real data can reduce the annotation costs considerably. However, the large domain difference makes the adaptation task challenging. In this regard, the performance improvements on the VisDA dataset along with the previous SYNSIG $\rightarrow$ GTSRB task provide convincing evidence that MUDA can overcome such difficulties.

4.5. Multi-Source Domain Adaptation

We extend MUDA to more complex multi-source domain adaptation problems. Unlike typical single-source UDA, multi-source DA considers adaptation from more than one source domains to one unlabeled target domain. To apply MUDA to a multi-source problem, we simply adopt the source combine strategy, in which all the source domains are combined to train a single model.

Setup

We evaluate MUDA on Digits-five and miniDomainNet [29]. Example images are illustrated in Fig. 4. Digits-five is a collection of digit images from five different domains: MNIST, MNIST-M [2], USPS, SVHN, and SYN [2]. We follow [44, 29] to sample 25,000 images for training and 9,000 for testing from each domain, except USPS. For the smaller USPS dataset, all 9,298 images are used. We choose one of the five domains for the target domain and the rest for the source domains. The experiment is repeated five times in turn.

MiniDomainNet [29] is a subset of DomainNet [44] and contains 140,006 $96 \times 96$ images of 126 classes from four domains: Clipart, Painting, Real, and Sketch. From each domain of miniDomainNet, 630 test images are held out. Similarly, we repeat the experiment four times in turn.

For Digits-five, the same network architectures for $F$ and $C$ and the hyper-parameters are used as in section 4.2. Each color image is randomly cropped and resized to $32 \times 32$. For miniDomainNet, ResNet-18 is used as in [29] and trained with the same hyperparameters in section 4.3.

Results

Table 4 and Table 5 show the mean and standard deviation of the classification accuracy after running five random experiments. The proposed MUDA achieves 95.74% (Digits-five) and 60.28% (miniDomainNet) average accuracy and outperforms the other single-source UDA methods by large margins. The tables also compare with recent multi-source methods. We observe that MUDA
Table 4: Classification accuracy (%) on Digits-five multi-source experiment (mt: MNIST, mm: MNIST-M, up: USPS, sv: SVHN, sy: SYN, R: the rest). The results are cited from each study. We indicate the best in bold in each standard.

| Standards | Methods       | R → mt | R → mm | R → up | R → sv | R → sy | Average |
|-----------|---------------|--------|--------|--------|--------|--------|---------|
| Oracle    | Target-only   | 99.65  | 98.30  | 99.52  | 94.30  | 99.22  | 98.20   |
| Source    | DANN [6]      | 98.46  | 83.44  | 94.19  | 84.08  | 92.91  | 90.61   |
|           | MCDA [6]      | 99.22  | 80.65  | 98.32  | 81.87  | 95.42  | 91.09   |
| Combine   | MUDA (ours)   | 99.18  | 93.21  | 98.89  | 90.39  | 97.62  | 95.74   |
| Multi-source | MDDA [13]   | 98.80  | 78.60  | 93.90  | 79.30  | 89.70  | 88.06   |
| UDA       | DCTN [10]     | 99.38  | 76.20  | 94.39  | 86.37  | 86.78  | 88.63   |
|           | M3SDA [12]    | 99.38  | 82.15  | 98.71  | 88.44  | 96.10  | 92.96   |
|           | DAEL [20]     | 99.45  | 93.77  | 98.09  | 92.50  | 97.04  | 96.47   |

Table 5: Classification accuracy (%) on miniDomainNet (clp: Clipart, pnt: Painting, rel: Real, skt: Sketch, R: the rest). The best in each standard is in bold.

| Standards | Methods       | R → clp | R → pnt | R → rel | R → skt | Average |
|-----------|---------------|--------|--------|--------|---------|---------|
| Oracle    | Target-only   | 72.59  | 60.53  | 80.47  | 63.44   | 69.26   |
| Source    | DANN [6]      | 65.55  | 46.27  | 61.54  | 44.12   | 54.76   |
|           | MCDA [6]      | 62.91  | 45.77  | 57.57  | 45.88   | 53.03   |
| Combine   | MUDA (ours)   | 67.62  | 52.38  | 63.41  | 57.72   | 60.28   |
| Multi-source | DCTN [10]   | 62.96  | 48.79  | 58.85  | 48.25   | 54.49   |
| UDA       | M3SDA [12]    | 64.18  | 49.05  | 57.70  | 49.21   | 55.03   |
|           | DAEL [20]     | 68.69  | 47.14  | 63.33  | 43.50   | 55.52   |
|           | MUDA (ours)   | 69.95  | 55.13  | 66.11  | 55.72   | 61.73   |

presents comparable and often better performance compared to the state-of-the-art multi-source UDA methods. Whereas many multi-source methods demand complex training procedures to take advantage of multi-source data, our MUDA can produce competitive results despite the considerably simpler source-combine approach.

5. Analysis and Discussion

Feature Visualization In the adaptation from SYNSIG to GTSRB in section 4.2, we visualize the learned features using t-SNE [15]. The embeddings in Fig. 5 contrast our MUDA with the source-only model. Red and blue points in the upper row represent the source and the target data, respectively. Different colors in the lower row indicate the 43 categories of the target samples. The adaptation results in the figure show desirable patterns. In Fig. 5b, the feature distributions of both domains align similarly after adaptation. This demonstrates that MUDA significantly reduces the divergence between the source and the target domains. Furthermore, the adapted features in the same category more tightly group together than those in different categories, as can be seen in Fig. 5c. Hence, MUDA makes the target samples become clearer and thus easier to classify. This clustering effect is mainly attributed to the model uncertainty minimization.
Figure 5: (Best viewed in color) SYNSIG→GTSRB Analysis. Feature embeddings obtained from the last pooling layer of $F$ are visualized using t-SNE [48]. Red and blue dots represent the testing samples of the source and the target domain, respectively. We observed that MUDA makes the target samples more discriminative as shown in (b) and (d).

Figure 6: (Best viewed in color) MUDA learning curve for SYNSIG to GTSRB.

approach of MUDA. To reduce the predictive variance, MUDA enforces classes so that they contain more consistent features.

**Learning Curve** For the task SYNSIG → GTSRB, we also present the learning curve of MUDA. Fig. 6 depicts how the classification accuracy and the model uncertainty loss change during training. As the model uncertainty loss drops rapidly in the early phase and begins to diminish slowly, the target accuracy
increases sharply and draws a gentle upward curve. In the meantime, the source error is kept small. This relationship between the model uncertainty and accuracy confirms that minimizing the model uncertainty on the target samples can improve the domain adaptation outcome.

Class Activation Maps To verify that MUDA can effectively capture semantically meaningful features, we examine the class activation maps on several images in the VisDA-17 dataset. In Fig. 7, the activation maps generated by Grad-CAM [49] highlight the regions that most contribute to the category prediction. Red (blue) regions correspond to high (low) scores for class. The first and the second rows show images of 3D models from the source domain and real images from the target domain, respectively. On the target images, the activation maps with or without adaptation are compared. Grad-CAM overlays from the source-only model are in the third row. The adapted results from the proposed MUDA are presented in the last row.

As shown in the Fig. 7, the source-only model often fails to locate the relevant regions properly. It pays attention to only a portion of the target objects (Fig. 7a, Fig. 7b), identifies target objects but with less accuracy (Fig. 7c, Fig. 7d, Fig. 7f), or even misses the target objects (Fig. 7b). On the other hand, the adapted model by MUDA produces more faithful localization maps and focuses on the class-discriminative regions of the target objects. These observations indicate that the proposed method MUDA learns domain-invariant
features. Thus, the model can successfully adapt to target samples and generate robust classification results.

The Choice of $M$ We investigate how the number of MC dropout samples $M$ affects the performance. Fig. 8 shows the change in the classification accuracy on the task SVHN $\rightarrow$ MNIST as $M$ is varied from 3 to 30. Though there is some fluctuation, we find that the performance increases slightly as $M$ increases. This occurs because the estimate of model uncertainty becomes more accurate with larger values of $M$. However, we can choose a smaller value for $M$ in practice to reduce the computational burden.

Analysis on Supremum and Expectation In our formulation of model uncertainty in section 3, the supremum of the $\mathcal{H} \Delta \mathcal{H}$ divergence is replaced with the expectation. We investigate this modification with a numerical analysis. We compare the supremum of the disagreement rate in equation 6 with the expected disagreement rate. To evaluate these quantities, we compute the disagreement rate $\mathbb{E}_{D_T} \left[ \mathbb{I}(h(x) \neq h'(x)) \right]$ between every pair of hypotheses. Then, we compare the supremum and the expectation of the target error in this analysis. In the
computation, MC dropout samples are used. Fig. 9 illustrates the supremum (blue) and the expectation (green) as MUDA progresses. Though the supremum is not directly minimized by MUDA, we can observe that the supremum decreases as the expectation decreases. The supremum is within the 95% confidence interval of the expectation all the time, and that their difference gets smaller. This empirical analysis supports our intuition and explains why the modified divergence works in practice.

6. Conclusion

In this paper, we presented a novel approach for unsupervised DA. Our method generates a domain-adaptive classifier that effectively generalizes to target domain. The proposed method uses the Bayesian approach and learns feature representations that reduce the divergence between the source and the target domains by minimizing the model uncertainty. We demonstrated that our approach outperforms current state-of-the-art methods on challenging image classification benchmarks.

We associated the proposed model uncertainty objective with the classifier-induced domain divergence. In the derivation, we replaced the supremum in $H\triangle H$-divergence with the expectation. This reformulation no longer makes the divergence as a valid upper bound of the target error. However, our intuition and empirical analysis suggest that the target error practically reduces as we minimize this modified divergence. We leave a more rigorous theoretical justification as future work.

Appendix A. Derivation of Predictive Variance of Hypotheses

Let $\bar{h}(x) = \mathbb{E}_{h \sim D_{h}}[h(x)]$. Then, we can rewrite

$$
\begin{align*}
\mathbb{E}_{h,h' \sim D_{h}}[(h(x) - h'(x))^2] \\
= \mathbb{E}_{h,h' \sim D_{h}}[(h(x) - \bar{h}(x) + \bar{h}(x) - h'(x))^2] \\
= 2 \mathbb{E}_{h \sim D_{h}}[(h(x) - \bar{h}(x))^2] - 2 \mathbb{E}_{h,h' \sim D_{h}}[(h(x) - \bar{h}(x))(h'(x) - \bar{h}(x))].
\end{align*}
$$

(A.1)

Here, the second term vanishes because

$$
\begin{align*}
\mathbb{E}_{h,h' \sim D_{h}}[(h(x) - \bar{h}(x))(h'(x) - \bar{h}(x))] \\
= \mathbb{E}_{h \sim D_{h}}[(h(x) - \bar{h}(x)) \mathbb{E}_{h' \sim D_{h}}[(h'(x) - \bar{h}(x)]] \\
= 0.
\end{align*}
$$

(A.2)

Therefore, we have

$$
2 \mathbb{E}_{h,h' \sim D_{h}} \mathbb{E}_{x \sim D_{T}}[(h(x) - h'(x))^2] = 4 \mathbb{E}_{x \sim D_{T}} \mathbb{E}_{h \sim D_{h}}[(h(x) - \bar{h}(x))^2].
$$

(A.3)
References

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

[2] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, Domain adversarial training of neural networks, Journal of Machine Learning Research 17(59) (2016) 1–35.

[3] E. Tzeng, J. Hoffman, T. Darrell, K. Saenko, Simultaneous deep transfer across domains and tasks, in: Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV), 2015, pp. 4068–4076.

[4] M.-Y. Liu, O. Tuzel, Coupled generative adversarial networks, in: Advances in Neural Information Processing System (NeurIPS), Vol. 29, 2016, pp. 469–477.

[5] E. Tzeng, J. Hoffman, K. Saenko, T. Darrell, Adversarial discriminative domain adaptation, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 7167–7176.

[6] K. Saito, K. Watanabe, Y. Ushiku, T. Harada, Maximum classifier discrepancy for unsupervised domain adaptation, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 3723–3732.

[7] H. Zhao, R. T. D. Combes, K. Zhang, G. Gordon, On learning invariant representations for domain adaptation, in: K. Chaudhuri, R. Salakhutdinov (Eds.), ICML, Vol. 97 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 7523–7532.

[8] M. Kim, P. Sahu, B. Gholami, V. Pavlovic, Unsupervised visual domain adaptation: a deep max-margin gaussian process approach, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4380–4390.

[9] Z. Lu, Y. Yang, X. Zhu, C. Liu, Y.-Z. Song, T. Xiang, Stochastic classifiers for unsupervised domain adaptation, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 9111–9120.

[10] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: Proceedings of International Conference on Machine Learning (ICML), 2016, pp. 1050–1059.

[11] Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with bernoulli approximate variational inference, in: Proceedings of International Conference on Learning Representations (ICLR), 2016.
[12] J. Lee, G. Lee, Model uncertainty for unsupervised domain adaptation, in: Proceedings of IEEE International Conference on Image Processing (ICIP), 2020, pp. 1841–1845.

[13] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, T. Darrell, Deep domain confusion: Maximizing for domain invariance, CoRR abs/1412.3474 (2014). arXiv:1412.3474.

[14] M. Long, Y. Cao, J. Wang, M. Jordan, Learning transferable features with deep adaptation networks, in: Proceedings of International Conference on Machine Learning (ICML), 2015, pp. 97–105.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing System (NeurIPS), Vol. 27, 2014.

[16] M. Long, H. Zhu, J. Wang, M. I. Jordan, Deep transfer learning with joint adaptation networks, in: Proceedings of International Conference on Machine Learning (ICML), 2017, pp. 2208–2217.

[17] Z. Pei, Z. Cao, M. Long, J. Wang, Multi-adversarial domain adaptation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[18] H. Zou, Y. Zhou, J. Yang, H. Liu, H. P. Das, C. J. Spanos, Consensus adversarial domain adaptation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 5997–6004.

[19] K. Saito, Y. Ushiku, T. Harada, K. Saenko, Adversarial dropout regularization, in: Proceedings of International Conference on Learning Representations (ICLR), 2018.

[20] R. Xu, G. Li, J. Yang, L. Lin, Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation, in: Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 1426–1435.

[21] C.-Y. Lee, T. Batra, M. H. Baig, D. Ulbricht, Sliced wasserstein discrepancy for unsupervised domain adaptation, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10285–10295.

[22] B. B. Damodaran, B. Kellenberger, R. Flamary, D. Tuia, N. Courty, Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation, in: Proceedings of European Conference on Computer Vision (ECCV), 2018, pp. 447–463.

[23] R. Xu, P. Liu, L. Wang, C. Chen, J. Wang, Reliable weighted optimal transport for unsupervised domain adaptation, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 4394–4403.
[24] Y. Wu, D. Inkpen, A. El-Roby, Dual mixup regularized learning for adversarial domain adaptation, in: ECCV, 2020, pp. 540–555.

[25] J. Sun, Z. Wang, W. Wang, H. Li, F. Sun, Domain adaptation with geometrical preservation and distribution alignment, Neurocomputing 454 (2021) 152–167.

[26] Y. Yang, T. Zhang, G. Li, T. Kim, G. Wang, An unsupervised domain adaptation model based on dual-module adversarial training, Neurocomputing 475 (2022) 102–111.

[27] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[29] K. Zhou, Y. Yang, Y. Qiao, T. Xiang, Domain adaptive ensemble learning, IEEE Transactions on Image Processing 30 (2021) 8008–8018.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing System (NeurIPS), Vol. 32, 2019, pp. 8026–8037.

[31] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient based learning applied to document recognition, in: Proceedings of the IEEE, Vol. 86(11), 1998, pp. 2278–2324.

[32] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning, in: Neural Information Processing System Workshop on deep learning and unsupervised feature learning, 2011, p. 5.

[33] J. J. Hull, A database for handwritten text recognition research, IEEE Transactions on Pattern Analysis and Machine (1994) 550–554.

[34] B. Moiseev, A. Konev, A. Chigorin, A. Konushin, Evaluation of traffic sign recognition methods trained on synthetically generated data, in: International Conference on Advanced Concepts for Intelligent Vision Systems, Springer, 2013, pp. 576–583.

[35] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, The german traffic sign recognition benchmark: a multi-class classification competition, in: IEEE International Joint Conference on Neural Networks (IJCNN), 2011, pp. 1453–1460.

[36] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of International Conference on Learning Representations (ICLR), 2015.
[37] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, D. Erhan, Domain separation networks, in: Advances in Neural Information Processing System (NeurIPS), Vol. 29, 2016, pp. 343–351.

[38] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, R. Chellappa, Generate to adapt: Aligning domains using generative adversarial networks, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 8503–8512.

[39] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: Proceedings of European Conference on Computer Vision (ECCV), 2010, pp. 213–226.

[40] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 248–255.

[42] X. Peng, B. Usman, N. Kaushik, D. Wang, J. Hoffman, K. Saenko, Visda: A synthetic-to-real benchmark for visual domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 2021–2026.

[43] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: Proceedings of European Conference on Computer Vision (ECCV), 2014, pp. 740–755.

[44] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, B. Wang, Moment matching for multi-source domain adaptation, in: Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 1406–1415.

[45] S. Zhao, G. Wang, S. Zhang, Y. Gu, Y. Li, Z. Song, P. Xu, R. Hu, H. Chai, K. Keutzer, Multi-source distilling domain adaptation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 12975–12983.

[46] R. Xu, Z. Chen, W. Zuo, J. Yan, L. Lin, Deep cocktail network: Multi-source unsupervised domain adaptation with category shift, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 3964–3973.

[47] K. Saito, D. Kim, S. Sclaroff, T. Darrell, K. Saenko, Semi-supervised domain adaptation via minimax entropy, in: Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV), 2019.
[48] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of Machine Learning Research 9 (2008) 2579–2605.

[49] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV), 2017, pp. 618–626.