




A Survey on Adversarial Domain Adaptation

Mahta HassanPour Zonoozi¹ · Vahid Seydi¹ 

Accepted: 20 July 2022 / Published online: 13 August 2022
© The Author(s) 2022

Abstract

Having a lot of labeled data is always a problem in machine learning issues. Even by collecting lots of data hardly, shift in data distribution might emerge because of differences in source and target domains. The shift would make the model to face with problems in test step. Therefore, the necessity of using domain adaptation emerges. There are three techniques in the field of domain adaptation namely discrepancy based, adversarial based and reconstruction based methods. For domain adaptation, adversarial learning approaches showed state-of-the-art performance. Although there are some comprehensive surveys about domain adaptation, we technically focus on adversarial based domain adaptation methods. We examine each proposed method in detail with respect to their structures and objective functions. The common aspect of proposed methods besides domain adaptation is considering the target labels are predicted as accurately as possible. It can be represented by some methods such as metric learning and multi-adversarial discriminators as are used in some of the papers. Also, we address the negative transfer issue for dissimilar distributions and propose the addition of clustering heuristics to the underlying structures for future research.

Keywords Domain adaptation · Adversarial learning · Domain shift

1 Introduction

Over the past few years, machine learning methods have achieved many successes. Real-world applications are an important part of success. Classification, face recognition, object detection, etc. are some of the areas. Training a deep neural network requires enough labeled datasets which have similar distribution. However, collecting dataset has been always expensive and time consuming. Furthermore, in case of testing a model with a dataset that has a different distribution from the training model's dataset, the model would fail; besides, it is expensive and impossible to collect appropriate dataset for each real-world application. Therefore, it is a great idea to be able to use any available dataset without regard to differences in the source and target distributions. Some factors such as image condition, brightness, and image quality leads to differences in distributions or domain shift. When a model is trained on a dataset, and it is tested on another dataset, its efficiency decreases because of domain

✉ Vahid Seydi
V.Seydi@bangor.ac.uk

¹ Faculty of Technical and Engineering, South Tehran Branch, Islamic Azad University, Tehran, Iran

shift [17] resulting from different distributions of each domain. Domain adaptation seeks to solve the domain shift problem [87]. A trained model on a large dataset cannot be generalized well to a new dataset [9] due to dataset bias or domain shift [68]. Convolutional neural networks (CNN) let us extract important features that are useful for classification and image segmentation. Although these features are related to the domain of the dataset and they won't be discriminative in other datasets. This aspect will result in a weak classification problem if the model trains on a dataset and tests on another dataset without adaptation [47].

Domain adaptation is learning an accurate model to transfer the knowledge extracted from source domain to the target domain with the presence of domain shift. Domain adaptation is a particular type of transfer learning where the feature space or the data distribution of source and target domains are different but the label space in the source domain and target domain are the same. Though the goal in domain adaptation is being able to classify unlabeled target data. Domain adaptation methods have achieved great success during past years.

Domain adaptation methods are divided into three categories with respect to the available datasets; these categories are supervised, semi-supervised and unsupervised domain adaptation. In supervised domain adaptation, both source and target data have labels. The source data and some limited number of target data have labels in semi-supervised domain adaptation. Whereas, labeled data are available only in the source data in the unsupervised domain adaptation method. Regarding adversarial domain adaptation researches, the unsupervised domain adaptation has been investigated more than two other categories [39].

The domain adaptation can be divided into two categories due to differences in domain divergences. Distribution shift and feature space differences form homogeneous and heterogeneous domain adaptation respectively. In homogeneous domain adaptation, the feature spaces of the source and target domains ($X_s = X_t$) and dimensions ($d_s = d_t$) are the same while the data distributions are different ($P(X)^s \neq P(X)^t$). In the heterogeneous domain adaptation, the feature spaces of the source and target domains are not the same ($X_s \neq X_t$) and the dimensions of the feature spaces may also be different ($d_s \neq d_t$). Heterogeneous domain adaptation can be divided into two scenarios. In the first scenario the differences are caused by different sensors such as RGB vs. depth images, and also different types of images such as photos, sketches, and paintings. In the second scenario, the differences are caused by different types of media in the source and target domains; for instance, text vs. images. The cross-domain gap between the source and target domains are much larger in the second scenario. There are three different methods to align the domains. Namely minimizing discrepancy, reconstruction, and adversarial training.

Discrepancy Based Domain Adaptation Minimizing divergence is one of the aligning distribution methods which measures divergence or distance between the distributions.

Commonly, in discrepancy based methods, the first n layers are shared or reused by the network between the source and target domains. In order to bound input target feature space near the source. This will cause limitation in feature spaces of the input to the same dimension. In the homogeneous domain adaptation discrepancy based methods can be result in good performance. In the first scenario of heterogeneous domain adaptation, the images can be resized into the same dimensions while in the second scenario, the different features of different media cannot be resized in to the same dimensions. Therefore, discrepancy based methods will fail.

Reconstruction Based Domain Adaptation Ghifary et al. [12] proposed reconstruction learning in which a representation is learned that accurately classifies the labeled source data and reconstructs source and target data. The reconstruction based method can be used in homo-

geneous domain adaptation while the adversarial reconstruction based method can be used in heterogeneous domain adaptation.

Adversarial Based Domain Adaptation Adversarial learning tries to align the distributions by extracting features which are both distinguishable for the labeled source data and indistinguishable for the source and target domains. Using adversarial based domain adaptation can result in good performance in both homogeneous and heterogeneous domain adaptation. In the heterogeneous domain adaptation, adversarial based method can generate heterogeneous target data while transferring source data features to them. Recently, many adversarial domain adaptation approaches have been proposed while they have shown promising results.

Adversarial domain adaptation is based on generative adversarial network; which involves two neural networks competing each other in a minimax game. Generative adversarial network has a generator and a discriminator which takes part in the minimax game [23]. Generator is trained to produce images in order to fool discriminator, while the discriminator tries to discriminate real and fake data correctly. In domain adaptation, the assumption changes in a way that the neural network tries to extract features such that makes the discriminator be fooled whereas discriminator tries to differentiate data domains [82].

Ganin and Lempitsky et al. [10] proposed a method in 2015, in which domain-invariant features are imposed on a classifier [62]. In order to achieve this goal, the classifier is trained in a way that operates well on source data while reduces the gap between the source and target domain extracted features [72]. For this purpose, domain adversarial learning is used [78]. Domain adaptation techniques were developed to cover the source and target domain shifts. Adversarial learning is one of the best techniques we will review them. It makes the source and target domain extracted features indiscriminative. Then features of the target domain are given to the pre-trained classifier in order to predict labels of target data. The classifier is pre-trained on source data [33]. Despite the distance of domain or dataset bias, adversarial learning methods make it possible to identify and detect data. In adversarial learning, two players have two aggressive purposes and try their best to defeat each other. In domain adaptation methods, the decoder related to feature extractor tries to extract features in a way discriminator to be fooled while discriminator tries to distinguish data domains as well as possible. In this survey, we will take into consideration the adversarial domain adaptation methods proposed over the past few years. Specifically, the following contributions are covered in this survey:

- Existing surveys have studied generally all domain adaptation methods, while we technically proposed a specific guide to adversarial domain adaptation methods. We hope to provide a useful resource for the researchers of adversarial domain adaptation. So they can decide to choose their preferred techniques facing the existing domain adaptation challenges.
- We took some articles into consideration that haven't been reviewed in previous surveys. We summarized the main ideas and equations of each approach.
- We categorized the proposed adversarial methods into groups based on their structures.

2 Deep Domain Adaptation

2.1 Adversarial Based Domain Adaptation Method

In this method, a discriminator classifies the domain of samples that are taken from the source or target domains. It is used to encourage domain misclassification with regard to an adversarial objective function which reduces the distance between the source domain

and target domain distributions. Adversarial based deep domain adaptation methods can be divided into two categories [11]. Generative models [77] and non-generative models [71].

- Generative models:

In generative models due to the lack of training data, the generator is able to generate unlimited unreal target domain data by using source domain data [14]. In this model a discriminator is combined with a generator based on GAN [2]. One of the common methods is using source domain images, noise vector, or both to generate simulated instances which are similar to the target domain and preserve the source domain information [32].

- Non-generative models:

Unlike generative models that have input image distribution, non-generative models have a feature extractor component which learns a discriminator representation using source domain labels and maps the target domain data to the same space over a domain-confusion loss [11]. Hence leading to domain-invariant representations [53].

2.2 Datasets

Reviewing methods, indicates that Digit datasets, Office-31, ImageCLEF-DA, VisDA2017 and Home-office datasets are widely used in unsupervised adversarial domain adaptation methods. A closer examination shows that Office-31 dataset and Digit datasets including MNIST, SVHN, and USPS are used more than others.

Digit Datasets These datasets contain digits 0–9 but with different styles. **MNIST dataset** includes grayscale 28*28 images in 60,000 train images and 10,000 test images [30]. **USPS dataset** involves RGB 32*32 images in 7291 train data and 2007 test data [7]. The street view house numbers which is known as **SVHN dataset** has 73,257 train images and 26,032 test images and 531,131 extra training images which are 32*32 RGB images [46]. These are real-world images from Google Street View images. **SYN-DIGITS** [10] dataset consists of 32*32 RGB images in 479,400 number of training samples and 9553 number of test samples. These images belong to the same classes as SVHN. It contains lots of labeled data since generating labeled synthetic data is easier than obtaining large labeled datasets with real-images.

Office-31 This is a standard benchmark for domain adaptation. It consists of 4652 images from 31 classes in 3 domains. 2817 images are from the Amazon domain (A) which are downloaded from Amazon.com website, 795 images from webcam domain (W) which are taken by a web camera and 498 images from DSLR domain (D) are collected by a digital SLR camera. In the experiments all six direction of adaptation are considered; $A \rightarrow W$, $A \rightarrow D$, $W \rightarrow A$, $W \rightarrow D$, $D \rightarrow W$, $D \rightarrow A$ [56].

ImageCLEF-DA This is a benchmark dataset which is collected by selecting the 12 common categories shared by the following three public datasets, each is considered as a domain: Caltech-256 (C), ImageNet ILSVRC 2012 (I) and Pascal VOC 2012 (P). The three domains are of equal size. In each category there are 50 images and in each domain 600 images exist.

Home-office This dataset includes four domains namely, Art (Ar), Clipart (CL), Product (Pr) and Real-World (Rw). Each domain contains 65 common categories [73]. The Art domain are artistic distributions of painting, sketches, etc. The Clipart domain contains clipart images. The Product domain includes images with no background. The Real-World domain are images from the regular camera.

VisDA2017 This is a visual domain adaptation dataset which is a simulation-to-real dataset which consists of 12 classes shared by both domains in 280,000 images in the training, validation and testing domains [51].



Fig. 1 MNIST, USPS, and SVHN datasets [71]

NYUD Dataset [63] The images are bounding boxes around the instances of 19 object classes. It is composed of 2186 labeled RGB images and 2401 unlabeled HHA-encoded depth [18] images. It is worth to know that these are acquired from two different datasets, so that their instances are not as the same.

Some samples of images from Digit datasets are shown in Figs. 1, 2, 3, 4.

3 Overview

The reviewed articles can be categorized based on their approaches.

3.1 Basic Adversarial Structures

The first category includes basic proposed methods, namely DANN and ADDA. Both methods have structures that are previously unexplored. DANN combined domain adaptation and deep feature learning within a single training procedure. ADDA combined untied weight sharing, GAN loss and discriminator modeling to adapt domains based on being optimal on discriminative tasks and exploiting a GAN-based loss.

3.1.1 Adversarial Discriminative Domain Adaptation (ADDA)

Tzeng et al. [71] proposed ADDA in 2017. This discriminative method uses GAN loss and it doesn't include weight sharing; its name is Adversarial Discriminator domain adaptation. Its results showed that the ADDA method is simpler and more efficient than other methods.

In this article he proposed a generalized structure for adversarial domain adaptation. Based on this structure each proposed architecture would be unique by answering three questions. Is it a generator model or a discriminator model? Does the feature extractor shares weight or not? Which adversarial objective function is used?

The ADDA method is not based on weight sharing which means it doesn't use a shared feature extractor. It has a consecutive process; in the first step, classifier loss can be computed based on labeled source data. Feature extractor will be learned in this step. In the next step, a pre-trained source encoder from the previous step is used. A separate encoder is considered for

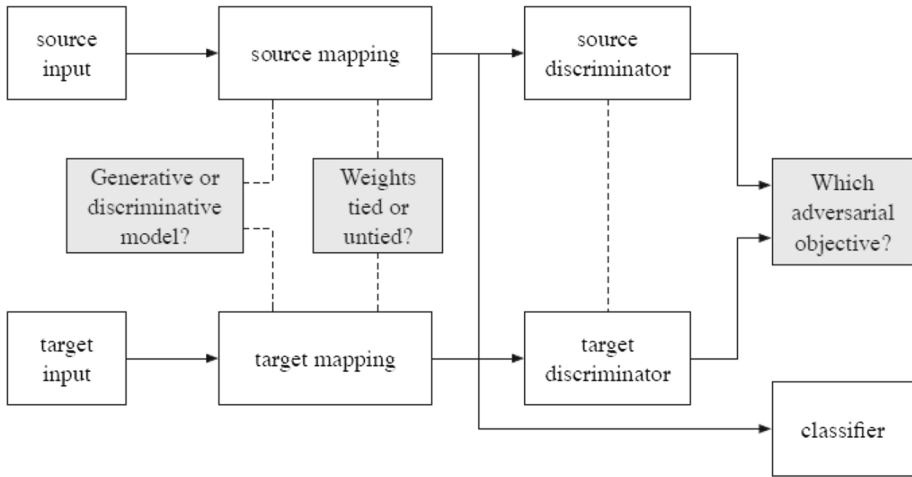


Fig. 2 Generalized structure for adversarial domain adaptation [71]

target data and plays a minimax role against domain discriminator. So adversarial learning progress between target encoder and domain discriminator. Target encoder tries to learn features in a way the discriminator is fooled while discriminator tries to detect the domain of source and target data correctly. GAN loss is used as the objective function of adversarial learning. In the last step, the pre-trained target encoder from the second step and the classifier trained on the source data in step 1, are used to classify target data.

In an unsupervised domain adaptation, source data is shown by X_s and their labels are shown by Y_s which are from source domain distribution $P_s(x, y)$. and target data X_t is gained from target domain distribution $P_t(x, y)$ while their labels are not available. The goal is learning a target representation (M_t) and a classifier (C_t) that can classify target data into k classes in the test phase. Since learning without labels is impossible, domain adaptation method learns a mapping of source representation (M_s) and uses source classifier (C_s), then learns how to adapt that model for the target domain.

The main goal of adversarial domain adaptation methods is reducing the distance between the source and target representations ($M_s(X_s), M_t(X_t)$). By achieving this goal, source classifier (C_s) can be used for target representation. ($C = C_s = C_t$) Since M_s is supposed to be unchanged, source domain labels are used to learn the best M_s .

In this paper separate representations are used for the source and target data. Adversarial learning part is learned only on M_t and source data distribution remains invariant. This concept can be comparable with the GAN concept in which real data distribution remains unchanged and generated data distribution is learned. Minimizing distance of the source and target representations is related to minimizing the following adversarial functions: This adversarial leaning uses non-saturating GAN loss as its optimizer [38].

$$\begin{aligned} \min_{M_s, C} L_{cls}(X_s, Y_s) &= \mathbb{E}_{(x_s, y_s) \sim (X_s, Y_s)} \\ &- \sum_{k=1}^K \mathbb{I}[k = y_s] \log C(M_s x_s) \min_D L_{advD}(X_s, X_t, M_s, M_t) = \\ &- \mathbb{E}_{x_s \sim X_s} [\log D(M_s(X_s))] - \mathbb{E}_{x_t \sim X_t} [\log(1 - D(m_t(x_t)))] \end{aligned}$$

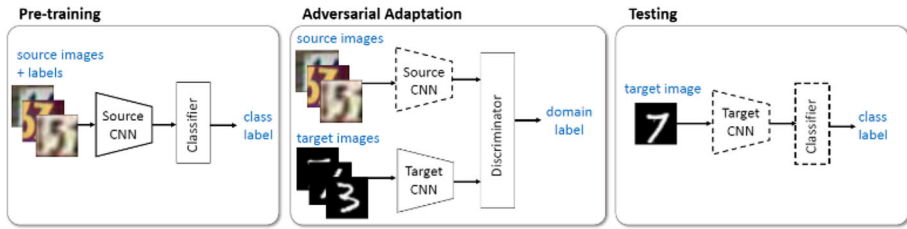


Fig. 3 The Adversarial discriminative domain adaptation (ADDA) architecture [71]

$$\begin{aligned}
 & \min_{M_s, M_t} L_{advM}(X_s, X_t, D) \\
 & = -\mathbb{E}_{x_t \sim X_t} [\log D(M_t(x_t))]
 \end{aligned}
 \tag{1}$$

Where M_s is source representation mapping, and C_s is source classifier. $M_s(X_s), M_t(X_t)$ are source and target mapping distributions. The optimization process starts with optimizing L_{cls} over M_s and using the labeled source data to train C . In second step, M_s is left fixed while training M_t , thus L_{advD} and L_{advM} can be optimized. In testing step, target images are fed into target encoder which is learned in adversarial step then classified by classifier trained in first step.

3.1.2 Domain-Adversarial Training of Neural Networks (DANN)

Ganin et al. [11] proposed DANN in 2016. The goal of the DANN method is domain adaptation; the method extracts features in a way that are classified correctly while their source or target domain is not recognizable.

The training is performed on labeled source data and un-labeled target data (unsupervised domain adaptation). As training progress, discriminative labels and non discriminative domains are followed up as the goal. This structure is provided by a deep neural network and its combination with a GRL layer. Training is done through backpropagation like any other deep structure.

In this paper the combination of domain adaptation and deep feature learning is addressed. The purpose is domain adaptation in a way that features can be classified well while the same features remain domain invariant. This means that features are learnt which are discriminative while their domains cannot be discriminated. This would be possible by optimizing features for (1) label classifier that classifies classes, and (2) domain classifier that discriminates between source and target domains. Domain classifier’s parameters are optimized in order to minimize training error, while deep feature mapping parameters are optimized in order to minimize loss of the label classifier and maximizing of the domain classifier. Maximizing the domain classifier loss(discriminator) and minimizing the feature extractor loss is an adversarial learning between domain classifier and extracted feature representation. Feature extractor seeks to learn domain invariant features that can make the discriminator to be fooled; yet the discriminator tries to discriminate domains of features correctly.

These three learning processes are defined in a feed forward neural network. In the training process, backpropagation algorithms are used based on stochastic gradient descent.

The proposed structure includes a deep feature extractor and a deep label classifier that form a standard feed forward structure. Unsupervised domain adaptation is implemented by adding a domain discriminator. Domain discriminator is connected to a GRL layer. During

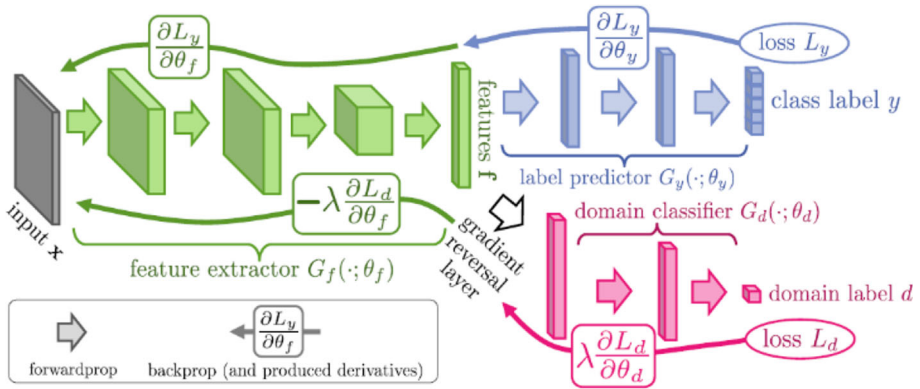


Fig. 4 The domain-adversarial neural network (DANN) structure [11]



Fig. 5 Sample of source and target from different datasets [11]

backpropagation GRL layer multiplies gradient by a negative value which results in maximizing discriminator loss; whereas through feed forward propagation minimizing classifier loss for source data and minimizing discriminator loss for all source and target data is considered. GRL ensures that feature representations over the two domains are made similar; it means features will be domain invariant. The results of the DANN method are investigated on MNIST, SVHN, and Office benchmark datasets.

Samples of source and target domain pairs used in this article are shown in Figs. 5, 6, 7, 8, 9, 10, 11, 12.

By training DANN the following function is going to be optimized:

$$\begin{aligned}
 E^{(\theta_f, \theta_y, \theta_d)} &= \frac{1}{n} \sum_{i=1}^n L_y(G_y(G_f(x_i; \theta_f); \theta_y), y_i) \\
 &\quad - \lambda \left(\frac{1}{n} \sum_{i=1}^n L_d(G_d(G_f(x_i; \theta_f); \theta_d), d_i) \right) \\
 &\quad + \frac{1}{n'} \sum_{i=n+1}^N L_d(G_d(G_f(x_i; \theta_f); \theta_d), d_i)
 \end{aligned} \tag{2}$$

Where $G_f(\cdot; \theta_f)$ is D-dimensional neural network feature extractor with parameter θ_f . And $G_y(\cdot; \theta_y)$ is the label predictor of the DANN’s structure with parameter θ_y . And $G_d(\cdot; \theta_d)$ indicates domain predictor of the neural network with parameter θ_d .

The goal of each component is like below: The parameters are optimized in order to minimize the loss of the feature extractor and classifier with a fixed value of the domain classifier’s parameters. And to maximize the loss of the domain classifier based on fixed

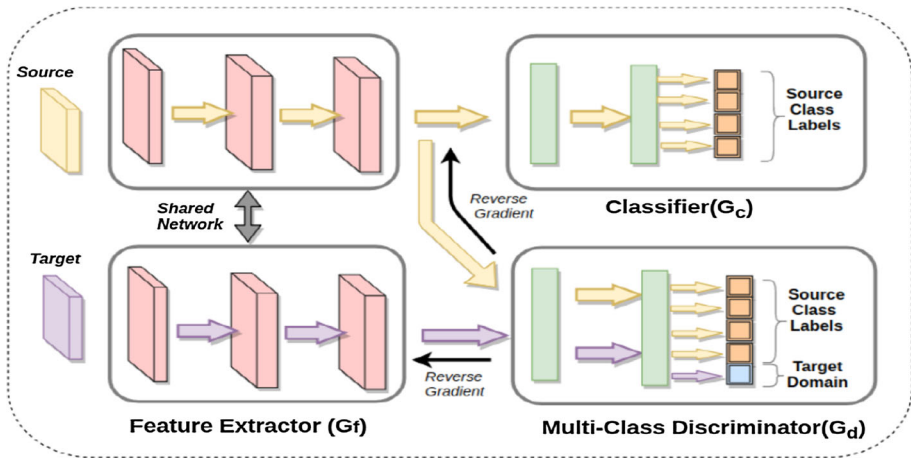


Fig. 8 The proposed architecture of a class based domain adaptation technique [28]

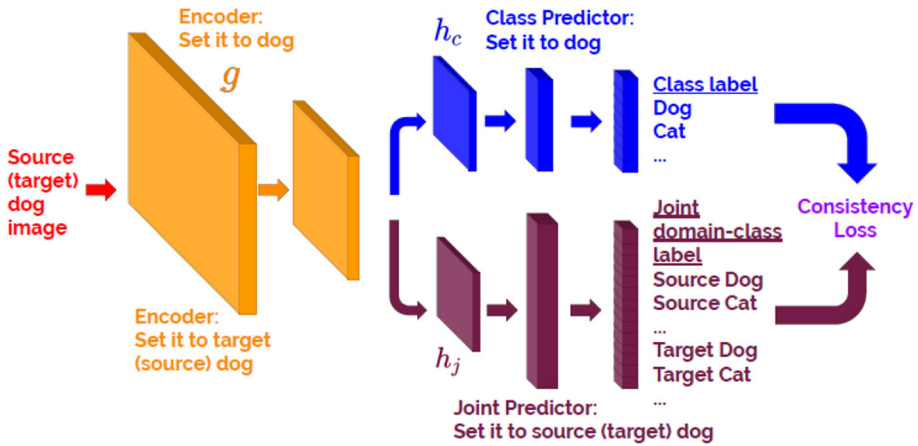


Fig. 9 The proposed structure of UDA [5]

$$\begin{aligned}
 \theta_y &\leftarrow \theta_y - \mu \left(\frac{\partial L_y^i}{\partial \theta_y} \right) \\
 \theta_d &\leftarrow \theta_d + \mu \left(-\lambda \frac{\partial L_d^i}{\partial \theta_d} \right)
 \end{aligned}
 \tag{4}$$

Such that μ is learning rate and λ is adaptation parameter.

The updates equations are similar to the stochastic gradient descent, and for a feed forward deep neural network it includes a feature extractor that fed into label predictor and domain classifier. The difference between the updates above and the SGD updates is that domain and label predictor here are subtracted while in the SGD is being summed. Such reduction can be presented by Gradient reversal layer which perform as an identity transformation in feed forward propagation and multiplies by a -1 during back propagation. GRL is located between feature extractor G_f and domain classifier G_d . GRL is considered as a pseudo function that its behavior is like below:

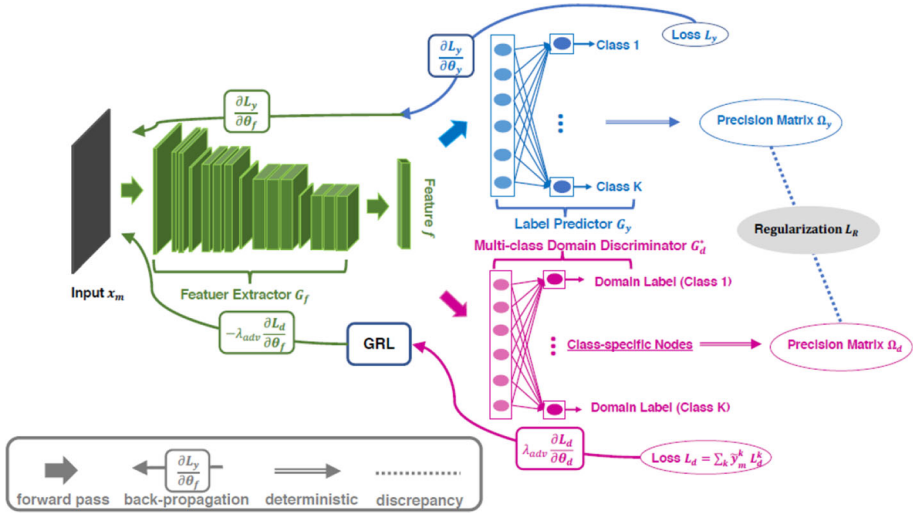


Fig. 10 The architecture of the proposed RADA algorithm built on top of the plain DANN model [79]

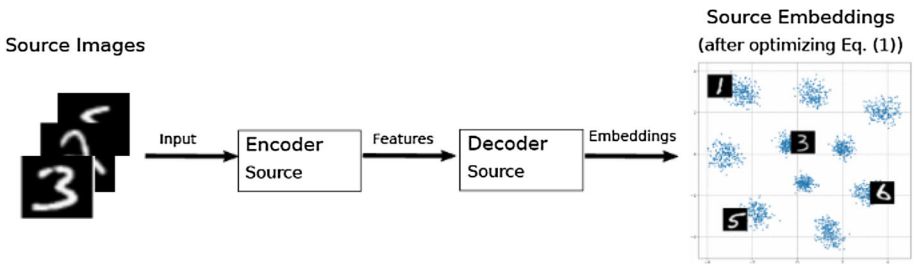


Fig. 11 Training the source model [29]. In this step encoder and decoder are trained with source images. The encoder extracts features of the source data and the decoder maps the extracted features to the embedding in which the clusters are observable

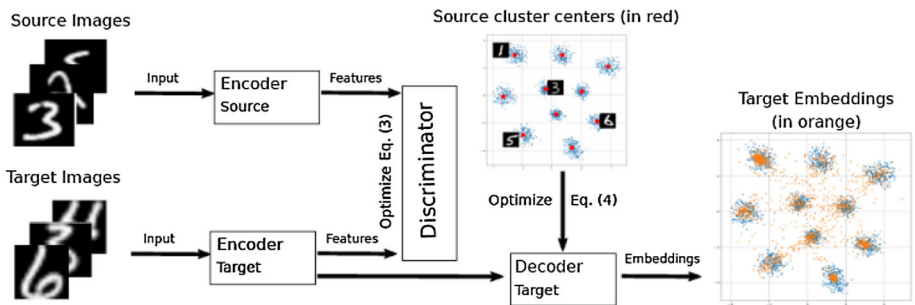


Fig. 12 Training the target model. In this step we adversarially adapt the encoded feature representation among source and target features. In order to optimize the center magnet loss, we use centroids of the source clusters. Target data labels are determined based on the label of the nearest source data [29]

$$\begin{aligned}\mathcal{R}(x) &= x, \\ \frac{d\mathcal{R}}{dx} &= -I,\end{aligned}\tag{5}$$

Where I is identity matrix.

3.2 Multi-class Discriminator

This category includes the proposed methods which explored the inter-class semantic relationships for domain adaptation. They point out that other methods based on single domain discriminators will result in mode collapse.

3.2.1 Multi-Adversarial Domain Adaptation (MADA)

The proposed method is a multi-adversarial structure which tries to enable fine-grained alignment of various data established on multiple domain discriminator, each for a class where k is the number of classes.

In typical domain adaptation problem, the source and target domain of data distributions include a complex multi-state structure that in supervised learning this structure includes class boundaries and in unsupervised learning it includes cluster boundaries. However, existing adversarial learning methods adapts data domain distributions without considering multistate structure, and it doesn't prevent negative transfer. The proposed MADA structure considers the following challenges in domain adaptation: (1) enhancing positive transfer by maximally matching multiclass structure on domain distribution of each domain, and (2) alleviating negative transfer by preventing false adaptation of classes in different distributions across domains.

Pei et al. [50] proposed a multi adversarial domain adaptation (MADA) which adapts source and target distributions based on multistate domain discriminator. A discriminator is assigned to each class. The purpose of this allocation is to reduce the negative transform and prevent wrong classification. Each data is assigned to a discriminator by hard assignment in order to prevent wrong classification of target data. Since each data is adapted to the most relevant class. Irrelevant classes will not be placed in the relevant domain discriminators and will be filtered. Studies showed that the proposed method outperform on standard domain adaptation datasets.

Labeled data of source domain is used to consider multi-state structure for source and target domain adaptation. The domain discriminator divides G_d into k classes, which is the number of labels of source data. $G_d^k, k = 1, \dots, k$ such that each k is each class for source and target domain adaptation.

Since the target domain has no label, it is hard to decide about G_d^k domain discriminator for each target data. It is investigated that output of label predictor $y'_i = G_y(x_i)$ for each data x_i is distribution probability over the label space of k classes. Thus it is a good idea to use y'_i as the probability that shows how much each data point x_i belongs to the domain discriminator. The attention of each data x_i to domain discriminator G_d^k can be modeled through feature weighting $G_f(x_i)$ with y_i^k probability. Applying it to all k domain discriminators $G_d^k, k = 1, \dots, K$ we will have:

$$L_d = \frac{1}{n} \sum_{k=1}^K \sum_{x_i \in D_s \cup D_t} L_d^k(G_d^k(y_i^k G_f(x_i)), d_i) \tag{6}$$

Where G_d^k is k-th domain discriminator, while L_d^k is its cross entropy loss and d_i is the domain label of point x_i .

Compared with the single-discriminator domain adversarial neural networks, the proposed multi adversarial domain adaptation enables adaptation where each data x_i is related only to those domain discriminators with y_i^k probability. The fine-grained adaptation may have three benefits: (1) it prevents hard assignment to a single domain discriminator which leads to inaccurate target domain data. (2) it prevents negative transform, since each data is adapted to the most relevant class while irrelevant classes are filtered out by probabilities and won't be involved in the corresponding domain discriminators. (3) multiple domain discriminators are trained by weighted probabilities, $y_i^k G_f(x_i)$ which learn multiple domain discriminators with different parameters θ_d^k . These domain discriminators with different parameters enhance positive transfer for each sample.

The objective function of multi adversarial domain adaptation is:

$n = n_s + n_t$ and $D = D_s \cup D_t$. λ is a hyper-parameter that trade-off two objective functions of the problem. The optimized function is finding parameters θ'_f, θ'_y and θ^k_d for $k = 1, \dots, K$ in a way to satisfy simultaneously both following loss functions:

3.2.2 Dual Adversarial Domain Adaptation(DADA)

Du et al. [84] proposed a method called Dual Adversarial Domain Adaptation (DADA). This method simultaneously performs domain-level and class-level alignment using a common discriminator. He proposed a mechanism based on multi-view learning [27] and domain adaptation [58], including two discriminators which pit against each other. Furthermore, SSL regularization is used to make the representations more discriminative.

In this article, Du referred to the adversarial domain adaptation methods which use a minimax game between feature extractor and domain discriminator. He emphasizes that these methods focus on domain-level distribution while class-level distribution matching is not guaranteed. Then, class-level alignment methods are proposed. Some of them use class-wise domain discriminators that each discriminator stands for one class [50] and some of them use discriminators with k-dimensional output in which k is the number of classes [79]. Furthermore, some methods proposed structures including both domain-level and class-level alignments in a single discriminator [28]. Experiments show that these methods can preserve the multimodal information in the source and target domain [22]. To have binary or k-dimensional output, many discriminators can be designed. Cicek et al. [5] proposed a method that contains a discriminator with 2k-dimensional output that can discriminate both class and domain information concurrently. The first k-dimensional outputs are the source classes and the second k-dimensional outputs are the target classes.

The method includes three steps. In step 1, the feature extractor (G), the class predictor (F), and the joint discriminators (D1, D2) are training using the source data. Source classification loss and joint discriminators losses are minimized in this step. Its objective is shown as below:

$$\min_{G, F, D_1, D_2} \ell_{sc}(F) + \lambda_{dsc1} \ell_{dsc}(D_1) + \lambda_{dsc2} \ell_{dsc}(D_2) \tag{7}$$

In step 2, feature extractor is fixed while class predictor and joint discriminators are updated using both source and target data. This step contains three sub objective functions, the first one is minimizing the source and target classification loss of joint discriminators. The second one is minimizing the source classification loss of the class predictor and also the SSL regularization loss. The last one is maximizing the discrepancy between discriminators. Objective function of this step is as follow:

$$\min_{F, D_1, D_2} \ell_F + \ell_{D_1} + \ell_{D_2} - \lambda_d \ell_d \tag{8}$$

$$\ell_{D_1} = \lambda_{dsc1} \ell_{dsc}(D_1) + \lambda_{dte1} \ell_{dte}(D_1) \tag{9}$$

$$\ell_{D_2} = \lambda_{dsc2} \ell_{dsc}(D_2) + \lambda_{dte2} \ell_{dte}(D_2) \tag{10}$$

$$\begin{aligned} \ell_f = \ell_{sc}(F) + \lambda_{svat} \ell_{svat}(F) + \lambda_{te} \ell_{te}(F) \\ + \lambda_{tvat} \ell_{tvat}(F) \end{aligned} \tag{11}$$

In step 3, the class predictor and joint discriminators are fixed while feature extractor is updated using source and target data. Source and target alignment loss of joint discriminators besides the discrepancy between discriminators are minimized in this step. The objective is guided as follows:

$$\begin{aligned} \min_G \lambda_{dsa1} \ell_{dsa1}(G) + \lambda_{dta1} \ell_{dta1}(G) \\ + \lambda_{dsa2} \ell_{dsa2}(G) + \lambda_{dta2} \ell_{dta2}(G) \\ + \lambda_d \ell_d \end{aligned} \tag{12}$$

Step 2 and step 3 are repeated interchangeably.

According to the above objective functions, it is good to know some details. ℓ_{dsc} and ℓ_{dte} are single discriminator loss and target classification loss functions respectively shown as below:

$$\ell_{dsc}(D_1) = E_{(x_s, y_s) \sim P} \ell_{CE}(D_1(G(x_s)), [y_s, 0]) \tag{13}$$

Where 0 is the zero vector to make the last k joint probabilities 0 for source samples. Predicted label for the target sample x_t is:

$$\hat{y} = \text{argmax}_k f(x_t)[k] \tag{14}$$

And target classification loss is:

$$\ell_{dte}(D_1) = E_{x_t \sim q_t} \ell_{CE}(D_1(x_t), [0, \hat{y}_t]) \tag{15}$$

The source classification loss of class predictor is shown as follow. This loss is used during training to minimize the cross entropy loss.

$$\ell_{sc}(F) = E_{(x_s, y_s) \sim P} \ell_{CE}(f(x), y) \tag{16}$$

$$\ell_{CE}(f(x), y) = - \langle y, \log f(x) \rangle \tag{17}$$

Where CE is cross entropy loss and it is calculated with one hot ground-truth labels $y \in \{0, 1\}^K$

Source alignment loss of the joint discriminator for labeled source data is:

$$\ell_{dsa1}(G) = E_{(x_s, y_s) \sim P} \ell_{CE}(D_1(G(x_s)), [0, y_s]) \tag{18}$$

And target alignment loss of the joint discriminator by changing the pseudo labels from $[0, y^t]$ to $[y^t, 0]$ is defined as below:

$$\ell_{dta1}(G) = E_{x_t \sim q_t} \ell_{CE}(D_1(G(x_t)), [\hat{y}_t, 0]) \tag{19}$$

Utilizing the absolute value of the difference between the probabilistic output is defined as the two joint discriminators' discrepancy.

$$d(f_{D1}(x), f_{D2}(x)) = \frac{1}{K} \sum_{k=1}^K |f_{D1}(x)[k] - f_{D2}(x)[k]| \tag{20}$$

Firstly, discriminators are trained to increase their discrepancy.

$$\max_{D1, D2} \ell_d \tag{21}$$

where

$$\ell_d = E_{x_s \sim p_s} [d(f_{D1}(x_s), f_{D2}(x_s))] + E_{x_t \sim q_t} [d(f_{D1}(x_t), f_{D2}(x_t))] \tag{22}$$

Then to make the two joint distributions similar, the feature extractor is trained and the discrepancy for fixed discriminator is minimized.

$$\min_G \ell_d \tag{23}$$

The discrepancy between domains can be smaller by using SSL regularization. The class predictor is trained to minimized the target entropy loss:

$$\ell_{te}(F) = E_{(x_t, y_t) \sim q} \ell_E(f(x)) \tag{24}$$

Where $\ell_E(f(x)) = -\sum_k f(x)[k].\log f(x)[k]$ Minimizing entropy is only applicable to locally-Lipschitz classifiers [44]. It causes adding following virtual adversarial training losses to the objective:

$$\begin{aligned} \ell_{svat}(F) &= E_{x_s \sim p_s} [\max_{\|r\| \leq \epsilon} \ell_{CE}(f_i(x_s) | f_i(x_s + r))] \\ \ell_{tvat}(F) &= E_{x_t \sim q_t} [\max_{\|r\| \leq \epsilon} \ell_{CE}(f_i(x_t) | f_i(x_t + r))] \end{aligned} \tag{25}$$

The results of this method on the Office-31 dataset are shown that it outperforms the methods which are compared with. It is compared with DANN and ADDA which only consider domain-level alignment and this method outperforms them. Also, it outperforms MADA which considers domain-level and class-level alignment. It uses a discriminator responsible for each class while the DADA method uses 2k-dimensional discriminator for classes.

3.2.3 Looking Back at Labels: A Class Based Domain Adaptation Technique (IDDA)

Kurmi et al. [28] proposed an adversarial model which contains a feature extractor, classifier and informative discriminator. Unlike [10, 50] use binary discriminator, IDDA method uses a multi-class discriminator. While binary discriminator is trained in a way to be fooled by feature extractor, multi-class discriminator in the proposed method classify source samples

with their labels and classify target samples with fake labels. Feature extractor is trained in a way to misclassify target samples as one of the source labels. The multi-modal structure is used to prevent misclassification of the source samples.

There are two objectives for training. First one is for minimizing label prediction loss on the source data while optimizing the parameters of feature extractor. The second one is about making the source and target extracted features indistinguishable.

$$\begin{aligned}
 loss(\theta_f, \theta_y, \theta_d) &= \frac{1}{n_s} \sum_{x_i \in D_s} L_y(G_y(G_f(x_i)), y_i) + \\
 &\frac{\lambda}{n_s + n_t} \sum_{x_i \in D_s \cup D_t} L_d(G_d(G_f(x_i)), d_i)
 \end{aligned}
 \tag{26}$$

where

$$d_i = \begin{cases} y_i, & \text{if } x_i \in D_s. \\ |C| + 1, & \text{if } x_i \in D_t. \end{cases}
 \tag{27}$$

Where λ is used as a trade-off parameter, C is the number of the source classes, L_y and L_d are respectively cross entropy loss as the classification loss and discriminator loss. D_s shows the source domain and D_t indicates the target domain. G_f , G_c , G_d are feature extractor, classifier, and discriminator.

IDDA model is evaluated on datasets such as Office-31 [56], Office-Home datasets [73] and Caltech- Bing datasets [1]. DAN [70], RTN [33], deep CORAL [65], I2I [45], Autodial [3], CDAN[35], DRCN [12] and DAH [73] are the methods which IDDA compared with.

3.2.4 Unsupervised Domain Adaptation via Regularized Conditional Alignment (UDA)

Cicek et al. [5] proposed UDA method which aligns joint distribution. A two-folded label space is proposed to improve target classification and domain confusion. The key idea of this method is to use a 2k dimensional discriminator instead of using a binary discriminator. It means that known source classes are assigned to the source domain and unknown target classes are assigned to the target domain. In other words, the first k outputs are source classes and the second k outputs are target classes. The feature extractor tries to fool the joint predictor such that the classifier loss will be minimized between two samples from source and target domains while the predicted target sample has the same label as the source sample. To align the target samples to the joint predictor, pseudo labeling is used. Since the first k labels in the joint predictor for target samples converge to zero, it is necessary to use semi-supervised learning regularization (SSL).

The overall objective function is as follow:

$$L_{adv}(g) = \lambda_{jsa} L_{jsa}(g) + \lambda_{jta} L_{jta}(g)
 \tag{28}$$

$$L(g, h_j, h_c) = L_s(g, h_j, h_c) + \lambda_t L_t(g, h_j, h_c)
 \tag{29}$$

Where

$$\begin{aligned}
 L_s(g, h_j, h_c) &= L_{sc}(f_c) + \lambda_{svat} L_{svat}(f_c) + \\
 &\lambda_{jsc} L_{jsc}(h_j) \\
 L_t(g, h_j, h_c) &= L_{te}(f_c) + \lambda_{tvat} L_{tvat}(f_c) + \\
 &\lambda_{jtc} L_{jtc}(h_j)
 \end{aligned}
 \tag{30}$$

The proposed method minimizes 1, and 2 interchangeably. The class predictor is written for the labeled source data as follow:

$$L_{sc}(f_c) = E_{(x,y) \sim P^s} \ell_{CE}(f_c(x), y) \tag{31}$$

Where ℓ_{CE} is cross entropy loss. The joint source classification loss is

$$L_{jsc}(h_j) = E_{(x,y) \sim P^s} \ell_{CE}(h_j(g(x)), [y, 0]) \tag{32}$$

Where 0 is the zero vector to make the last k joint probabilities 0 for source samples. And joint target classification loss is as below:

$$L_{jtc}(h_j) = E_{x \sim P_x^t} \ell_{CE}(h_j(g(x)), [0, \hat{y}]) \tag{33}$$

Where \hat{y} is in place of ground-truth labels for target data. $\hat{y} = e_k$ and $k = \text{argmax}_k f_c(x)[k] = \text{argmax}_k h_c(g(x))[k]$

The joint source alignment loss is minimized to train the encoder to make the joint label from $[y, 0]$ to $[0, y]$. It is shown as below:

$$L_{jsa}(g) = E_{(x,y) \sim P^s} \ell_{CE}(h_j(g(x)), [0, y]) \tag{34}$$

The joint target alignment loss is:

$$L_{jta}(g) = E_{x \sim P_x^t} \ell_{CE}(h_j(g(x)), [\hat{y}, 0]) \tag{35}$$

The encoder is trained to change the pseudo-labels from $[0, \hat{y}]$ to $[\hat{y}, 0]$ by minimizing the loss function above. The target entropy loss is minimized by the training of class predictor:

$$L_{te}(f_c) = E_{x \sim P_x^t} \ell_E(h_c(g(x))) \tag{36}$$

Where

$$\ell_E(f(x)) := -\langle f(x), \log f(x) \rangle \tag{37}$$

The regularization loss of Lipschitz condition [43, 44] on the source and target data are as follows:

$$\begin{aligned} L_{svat}(f_c) &= E_{(x,y) \sim P^s} \ell_{VAT}(f_c(x)) \\ L_{tvat}(f_c) &= E_{x \sim P_x^t} \ell_{VAT}(f_c(x)) \end{aligned} \tag{38}$$

The results of the UDA method are compared with DANN [10], DRCN [12], kNN-Ad [61], ATT [57], VADA [62], DIRT-T [62] and Co-DA [27]. The comparison shows it outperforms other methods and has the second-best performance after Co-DA. MNIST [30], SVHN [46], CIFAR10 [26], STL [6] and SYN-DIGITS [10] are the datasets used in its experiments.

3.2.5 Adversarial Domain Adaptation Being Aware of Class Relationships (RADA)

Wang et al. [79] proposed a novel relational-aware adversarial domain adaptation (RADA) method inspired by rDNN [24] and based on DANN [11]. He believes that the inter-class semantic relationship hasn't been much considered. RADA method makes use of a single multi-class domain discriminator in order to learn the inter-class dependencies. By training the label predictor on source domain samples, inter-class dependencies are distinguished. So there will be a discrepancy between inter-class dependencies evaluated from label predictor and domain discriminator; that would be fine by implementing a regularization term. This

method’s alignment provides class relationship awareness of the adversarial domain adaptation. The adversarial domain adaptation being aware of class relationships comprises two phases; class relationships are modeled based on DNNs at first, then the class specific domain discriminator is implemented.

The training objective is as below:

$$\begin{aligned} \min & \frac{1}{M_s} \sum_{x_m \in D_s} L_y(G_y(G_f(x_m)), y_m) \\ & + \lambda_R L_R + \frac{\lambda_{adv}}{M_s + M_t} \\ & \sum_{k=1}^K \sum_{x_m \in D_s \cup D_t} \tilde{y}_m^k L_D^k(G_d^*(\mathcal{R}(G_f(x_m))), d_m) \end{aligned} \tag{39}$$

Where λ_R is balancing parameter for relational aware regularization term. Ω is used to model the inter-class dependency, it is interjected to adversarial training process to make adversarial domain adaptation aware of class relationships. G_f and G_y are extracted features to predict class labels. G_d^* aligns the source and target data distributions. A regularization is designed to minimize the discrepancy of class relationships between G_y and G_d^* as below:

$$\begin{aligned} d \rightarrow y : L_R &= Tr(W_y^{[L]}(W_d^{[L]T} W_d^{[L]})^{-1} W_y^{[L]T}) \\ &- \frac{d_y}{d_d} \{ \log \det(W_y^{[L]T} W_y^{[L]}) - \log \det(W_d^{[L]T} W_d^{[L]}) \} \end{aligned} \tag{40}$$

And

$$\begin{aligned} y \rightarrow d : L_R &= Tr(W_d^{[L]}(W_y^{[L]T} W_y^{[L]})^{-1} W_d^{[L]T}) \\ &- \frac{d_d}{d_y} \{ \log \det(W_d^{[L]T} W_d^{[L]}) - \log \det(W_y^{[L]T} W_y^{[L]}) \} \end{aligned} \tag{41}$$

These equations are consequence of inserting Ω_y and Ω_d into $D_{KL}(\Omega_y \parallel \Omega_d) = Tr(\Omega_y^{-1} \Omega_d) - \log \det(\Omega_y^{-1} \Omega_d) - K$ and $D_{KL}(\Omega_d \parallel \Omega_y) = Tr(\Omega_d^{-1} \Omega_y) - \log \det(\Omega_d^{-1} \Omega_y) - K$ respectively which minimize the divergence from Ω_d to Ω_y and Ω_y to Ω_d .

Ω_y and Ω_d are denoted as the precision metric regarding weight metrics $W_y^{[L]}$ and $W_d^{[L]}$ of the output layers in G_y and G_d^* . Where

$$\begin{aligned} \Omega_y &= d_y (W_y^{[L]T} W_y^{[L]})^{-1} \\ \Omega_d &= d_d (W_d^{[L]T} W_d^{[L]})^{-1} \end{aligned} \tag{42}$$

Where Ω_y can be optimized by the training object:

$$\begin{aligned} \min_{\Omega_y} & -d_y \log \det(\Omega_y) + Tr(W_y^{[L]} \Omega_y W_y^{[L]T}); \\ \text{s.t. } & \Omega_y \succeq 0. \end{aligned} \tag{43}$$

And Ω_d can be optimized by objective below:

$$\begin{aligned} \min_{\Omega_d} & -d_d \log \det(\Omega_d) + Tr(W_d^{[L]} \Omega_d W_d^{[L]T}); \\ \text{s.t. } & \Omega_d \succeq 0. \end{aligned} \tag{44}$$

The mean classification accuracy of RADA method is compared with ResNet [20], TCA [48], GFK [15], DDC [70], DAN [34], RTN [33], DANN [11], ADDA [71], JAN [37], JDDA [4], CAN [86] and MADA [50] on ImageCLEF-DA and Office-31 datasets. Reports show it provides comparable results.

3.3 Using Clustering

Metric-based methods are listed as the third category. These methods try to improve discrimination of target representation by learning clusters. They are minimizing the distance between cluster centers and target sample embeddings.

3.3.1 M-ADDA: Unsupervised Domain Adaptation with Deep Metric Learning

Sometimes due to unsupervised domain adaptation, some difficulties may come to emerge. For instance, source and target feature representations are adapted but the data are mistakenly placed close together and incorrectly classified. Metric based methods are proposed for solving this problem.

In 2018, Issam et al. [29] proposed the M-ADDA method based on the ADDA method. This method emphasizes tasks in unsupervised domain adaptation. It is shown that using metric learning has been successful in different classification tasks [80]. Learning a metric based on distance is metric learning methods' goal; which brings examples with the same labels close as much as possible and keep the samples with different labels apart as far as possible. It can be both used in unsupervised learning methods such as clustering [83] and supervised learning such as k-nearest neighbor algorithms [19].

Two basic steps of the M-ADDA model:

1. Source model is clustered based on metric learning on source dataset by triplet loss function.
2. Source and target feature extracted distributions are adapted, and simultaneously predicted target dataset embeddings are formed as clusters.

Previous methods were discrepancy based, while recent studies show better domain adaptation performance of adversarial learning. The method is implemented on MNIST and USPS Digit datasets. It is shown that the method outperforms ADDA on MNIST and USPS Digit datasets [81].

Training the Source Model The source model $f_{\theta_s}(\cdot)$ with θ_s parameters is trained based on triplet loss on source dataset:

$$L(\theta_s) = \sum_{(\alpha_i, p_i, n_i)} \max(\|f_{\theta_s}(\alpha_i) - f_{\theta_s}(p_i)\|^2 - \|f_{\theta_s}(\alpha_i) - f_{\theta_s}(n_i)\|^2 + m, 0) \quad (45)$$

In which α_i is the anchor sample that is selected randomly. p_i is a sample with the same label as α_i . And n_i is a sample with a label different from α_i . The above equation results in approaching the samples α_i to p_i by the margin, m .

Training the Target Model In the training the target model section, cluster centroids of source feature representations are considered as C . Each C center is a unique label of the source dataset. The center is calculated using the average of the data in the source cluster belonging

to a label. Then target model is trained by parameter θ_T and optimizing the following loss functions:

$$L(\theta_T, \theta_D) = L_A(\theta_{TE}, \theta_D) + L_C(\theta_T) \tag{46}$$

$L_A(\theta_{TE}, \theta_D)$ is the loss function of the adaptation section. And $L_C(\theta_T)$ is the center magnet. The source model encoder parameters are shown by θ_{SE} and the target model encoder parameters are shown by θ_{TE} . θ_D shows parameters of the discriminator model which are used for adapting the source and target extracted features representation. (The adversarial part):

$$L_A(\theta_{TE}, \theta_D) = \min_{\theta_D} \max_{\theta_{TE}} \left[- \sum_{i \in S} \log D_{\theta_D}(E_{\theta_{SE}}(X_{S_i})) - \sum_{i \in T} \log(1 - D_{\theta_D}(E_{\theta_{TE}}(X_{T_i}))) \right] \tag{47}$$

$D(\cdot)$ is the discriminator model to maximize the probability that the extracted features in the source model came from the source dataset and the extracted features in the target model came from the target dataset. In other words, discriminator $D(\cdot)$ tries to detect the domain of extracted features correctly. The discriminator $D(\cdot)$ assigns greater values (approximately 1) to the extracted features from the source domain and smaller values (approximately 0) to the extracted features from the target domain.

Simultaneously target encoder is trained to make the discriminator to be fooled. This adversarial learning makes the extracted features $E_{\theta_{SE}}(X_{S_i})$ and $E_{\theta_{TE}}(X_{T_i})$ domain invariant.

In parallel center magnet loss is minimized:

$$L_C(\theta_T) = \sum_{i \in T} \min_j \|f_{\theta_T}(x_i) - C_j\|^2 \tag{48}$$

Which takes x_i samples to the nearest center of cluster C . Since there are 10 classes in MNIST dataset, there will be 10 cluster centroids. $|C| = 10$

This regularization causes the target data representation to form clusters similar to source clusters. In fact, adding $L_C(\theta_T)$ loss function creates a better form of target clusters.

3.3.2 Domain-Invariant Adversarial Learning for Unsupervised Domain Adaption (DIAL)

In existing adversarial-based domain adaptation methods, only marginal distribution adaptation through $P(X)$ distribution is considered. However, in some previous studies the conditional distribution $P(Y|X)$ of the two domains might be different. Since there is no label for target data, direct determination of $P(Y|X)$ is challenging. Inspired by references such as [36, 75], some pseudo labels for target samples are considered and class conditional distribution $P(X|Y)$ will be explored which helps target features to be placed in correct clusters.

In the previous studies, feature extractor component between source and target was separated or partially tied, while a common feature extractor between source and target domain is considered. Also by the proposed structure, feature representations of both domains are learned simultaneously unlike previous methods where source features were fixed during adaptation.

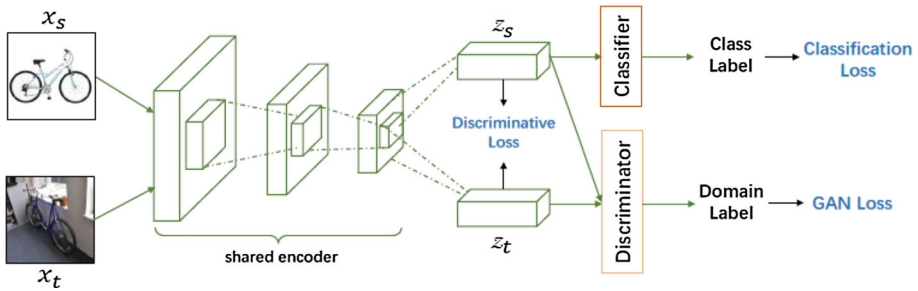


Fig. 13 DIAL proposed structure for unsupervised domain adaptation [87]

Zhang et al. [87] Proposed DIAL method in 2018. This is a simple but effective method for unsupervised domain adaptation. The method is inspired by the fact that humans recognize an object correctly without knowing its domain. Domain-Invariant Adversarial Learning for Unsupervised Domain Adaption method includes an encoder, a classifier, and a discriminator for both domain invariant and discriminative representations. Unlike models including two separate [71] or partially tied feature extractor components for source and target domain [54], DIAL shares an encoder between two domains, and there is no need to know the source of images during the test.

Then extracted features are sent to the adversarial discriminator. Encoder and discriminator play a minimax game with the goal that the source of features cannot be discriminated. It is expected the encoder learns domain invariant representations and ignores the domain specifications. Furthermore, to emphasize the power of discriminating feature representations, the center loss is introduced using source domain labeled data.

Part One: Domain Invariant Feature Extraction In unsupervised domain adaptation, domain invariant feature extraction is intended to reduce the effect of domain shifts.

As it is shown in Figs. 13, 14, 15, 16, 17, 18, 19, images from source and target domains are given to a common encoder and there is an aim to extract features which contain information about the content of the image, namely domain invariant features. Here the adversarial learning and a discriminator are adopted to discriminate which domain the extracted feature is from, and simultaneously encoder tries to extract indistinguishable features for the discriminator. Adversarial loss would be as follow:

$$\min_{\theta_E} \max_{\theta_D} L_{GAN} = \sum_{x_i \in X_S} \log(D(E(x_i))) + \sum_{x_i \in X_T} \log(1 - D(E(x_i))) \tag{49}$$

Where $D(\cdot)$ is the probability of being source domain predicted by discriminator D . θ_E and θ_D are parameters of encoder E and discriminator D . X_S and X_T are distributions of samples in the source and target domain. Although the input of the decoder is from two different domains, the domain of extracted features cannot be distinguished by the discriminator. By this limited condition, it is expected the encoder extracts the content information of images and ignores the domain's specification. In addition, a common encoder is used for a model of source and target domain and there is no need to know the source of images during testing.

Part Two: Discriminative Feature Extraction Since the source domain samples have labels, it is possible to classify source domain features by classifier C which is a fully connected softmax layer that its size depends on each task. Optimization function for classification of

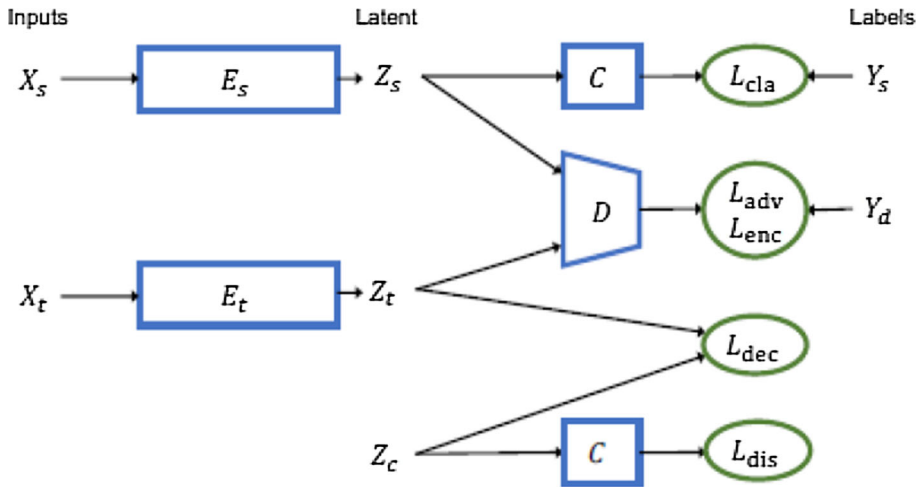


Fig. 14 Robust unsupervised domain adaptation architecture. Y_s and Y_d are source and domain labels. Latent features for source, target and target cluster centroids are shown by Z_s , Z_t and Z_c . Model blocks are indicated by rectangles and losses by ellipses [78]

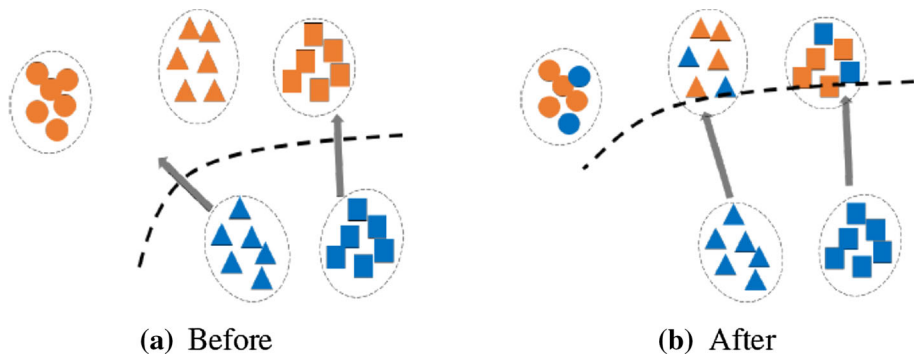


Fig. 15 Features before and after target generation. Orange and blue items show source and target features respectively. The dotted line is the boundary separating the source and the target, learned by the discriminator. Arrows shows the direction of the adaptation. After target generation, discriminator illustrates a softer boundary for target domain. Since target samples tend to be moved vertically relative to the boundary through gradient updates, this change can reduce the impact of negative transfer from the target toward outlier source classes [78]

labeled source data is defined as below:

$$\min_{\theta_E, \theta_C} L_S = \sum_{(x_i, y_i) \sim (X_S, Y_S)} H(C(E(x_i)), y_i) \tag{50}$$

Where $H(\cdot)$ is cross entropy loss which is used in the softmax layer. X_S and Y_S are respectively data distribution and labels in the source domain. θ_C shows classifier parameters.

Moreover, it is important to preserve discriminative feature representation during domain adaptation. Although the source and target domain distributions are adapted, some instances might be placed in the wrong interclass space, which introduces the need to learn discriminative features. There are different methods for learning discriminative features, such as triplet

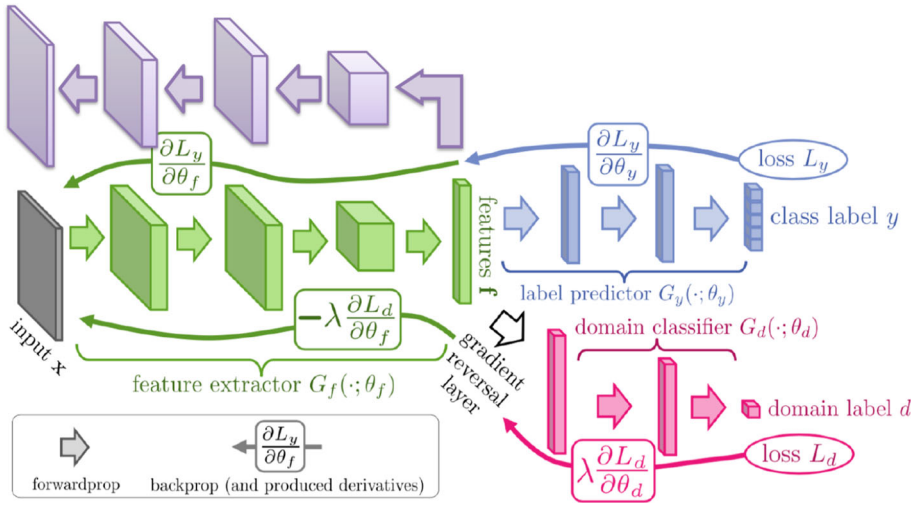


Fig. 16 DAUTO. A modified model of DANN [88]

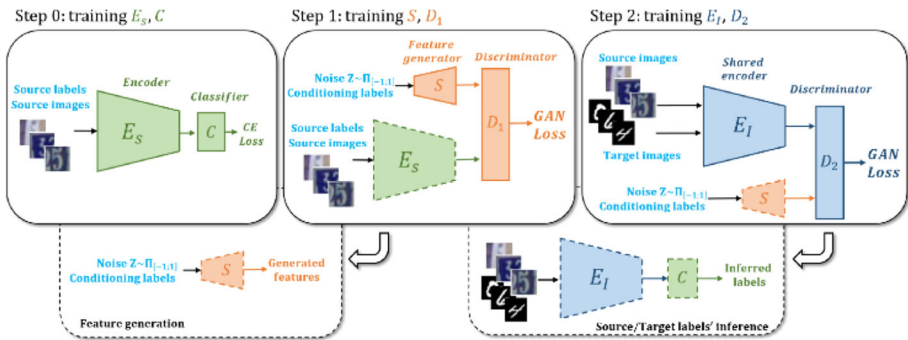


Fig. 17 DIFA training. The solid line indicate that the module is training. Dash lines indicate that the module has been trained in the previous steps. All of the modules are neural networks. Panels in the dash boxes are: left one, indicating feature generation, and the right shows predicting labels of source and target data with pre-trained encoder and pre-trained classifier [75]

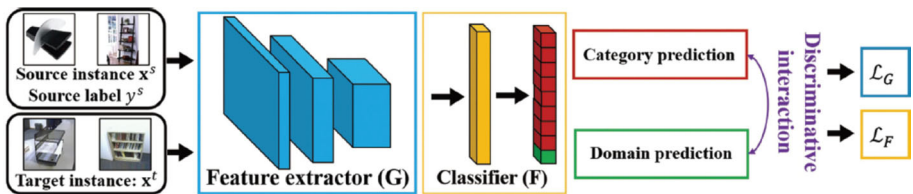


Fig. 18 The structure of discriminative adversarial domain adaptation (DADA) [66]

loss [60], the contrastive loss [64] and the center loss [81]. Both triplet loss and contrastive loss require the creation of a large number of pairs of images and the calculation of the distance between the images of each pair that are computationally complex. Therefore, in this study, center loss is introduced which can be easily combined with classification loss.

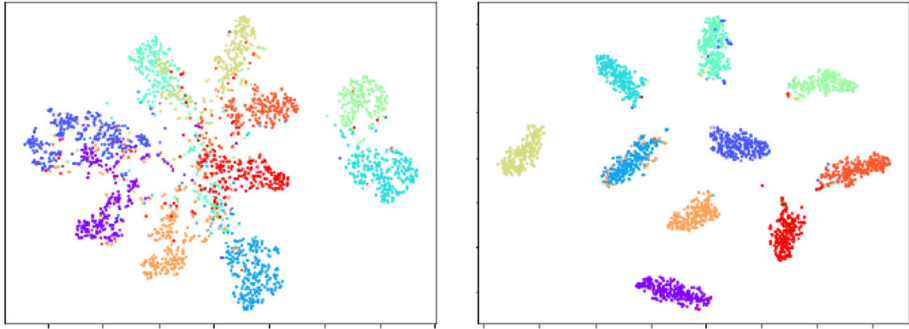


Fig. 19 t-SNE plot of the feature domains for SVHN→MNIST. ADDA is shown in the left Fig and “Discriminative clustering for Robust Unsupervised Domain Adaptation” in the right [78]

The following loss function is used to cluster features belong to a class for labeled source domain samples:

$$\min_{\theta_E} L_S = \sum_{(x_i, y_i) \sim (X_S, Y_S)} \|E(x_i) - C_{y_i}\|_2^2 \tag{51}$$

Where C_{y_i} is a d-dimensional vector representing the center of the $y_i - th$ class. Ideally, center of each class is calculated based on all features of samples belong to the class. But since the model is optimized for each mini-batch, it is difficult to compute the average of all samples. So at first the center of the class is initialized by the batch in the first iteration, and then centers are updated with the following strategy:

$$C_k^{t+1} = C_k^t - \lambda \Delta C_k^t, k = 1, \dots, K \tag{52}$$

Where C_k^t is the center of k-th class in t-th iteration. γ is the learning rate for updates of centers. K is the number of classes.

$$\Delta C_k^t = \frac{\sum_{(x_i, y_i) \in \beta^t} \mathbb{1}(y_i = k)(C_k^t - E(x_i))}{1 + N_k} \tag{53}$$

And β^t shows mini-batch in t-th iteration. $\mathbb{1}(\cdot)$ is an indicator function. N_k is the number of samples in the β^t batch which belongs to class k:

$$N_k = \sum_{(x_i, y_i) \in \beta^t} \mathbb{1}(y_i = k) \tag{54}$$

Part Three: Conditional Distribution Alignment

To maintain domain invariance, center loss used for source data is also used for target data to enforce them into the clusters. After applying L_{ct} loss function, the number of misclustered points decreases, and more points are placed in the appropriate space. This is because of the conditional distribution $P(Y|X)$ which leads target data to correct clusters due to correct labels. But since target data don't have labels, $P(Y|X)$ cannot be used. By using class conditional distribution, pseudo labels are allocated to target data. Then source classifier is applied to target data. To find out how well it worked, the center loss is used. However, in the center loss computation procedure, not all predicted labels are useful. Rather, just a part of the data is considered that its probability to belong to a predicted class is more than a particular threshold. Therefore, for un-labeled samples in the target domain, a pseudo label

that is predicted by a source classifier is allocated to a sample. Center loss for target domain is defined as below:

$$\min_{\theta_E} L_{ct} = \sum_{x_i \in \phi(X_t)} \|E(x_i) - c_{\hat{y}_i}\|_2^2 \tag{55}$$

Where \hat{y}_i is a predicted label of x_i by the classifier C. Since not all predicted labels are necessarily correct, L_{ct} is computed only on $\phi(X_t)$ that is a subset of x_i :

$$\phi(X_t) = \{x_i | x_i \in X_t \text{ and } \max(p(x_i)) \geq T\} \tag{56}$$

Where $p(x_i)$ is a k-dimensional vector with the i-th dimension that is the predicted probability of belonging to the i-th class. $\max(p(x_i))$ is the probability sample x_i belongs to a class. T is the threshold. Thus, the samples of target domain are expected to get in the corresponding clusters.

Total objective function of the model would be as follow:

$$\min_{\theta_E, \theta_c} \max_{\theta_D} L_{GAN} + \alpha L_s + \beta_1 L_{cs} + \beta_2 L_{ct} \tag{57}$$

α, β_1 and β_2 are weighted parameters.

3.3.3 Discriminative Clustering for Robust Unsupervised Domain Adaptation

Wang et al. [78] Proposed the discriminative clustering for robust unsupervised domain adaptation method in 2019. In this paper it is explained that ADDA and DANN domain adaptation methods are performed on the assumption that the source and target domain distributions are balanced, which means source and target domain labels are equal ($y_s = y_t$) and the probability of labels are equal also ($P(y_s) = P(y_t)$). But domain adaptation is possible in the other two spaces. Imbalanced domain adaptation, that have equal labels in the source and target domain while their probabilities are different. It means that data with a specific label is seen in different values in the source domain and the target. Partial domain adaptation, that target labels are a subset of source labels ($y_t \subset y_s$). The purpose of this paper is to extend domain adaptation to imbalanced and partial setting in addition to balanced scenario [49].

An important part of the representation, is improvement the ability to recognize target latent representation simultaneously with (1) learning tightly clustered target representation, (2) Emphasizing that each cluster is assigned to a unique and different classes of the source domain. And (3) minimizing the distance of the source and target representation through distribution adaptation. Studies showed that these three metrics reduce the effect of negative transform in partial and imbalanced domain adaptation tasks. And generally, the proposed method yielded good results in all three scenarios.

The unsupervised domain invariant proposed model is trained by classification on the source data $P(Y_s = k | Z_s) = C(Z_s), k = 1, 2, \dots, K_s$. While the classifier is optimized, source encoder will be used beside pre-trained source classifier to adapt domains. At this step, the adaptation is due to updates of discriminator, target encoder and cluster centroids with regard to objective functions.

A discriminator is considered to have latent features which possess no information about the difference of source and target domain and it just has information about labels $p(Y_{domain} = s | Z_d) = D(Z_d)$, where Z_d is about source and target domain $d \in s, t$, and Y_{domain} indicates being from source or target domain $Y_{domain} \in s, t$.

This means that learning a discriminator is desirable which considers label 1 for the source domain ($p(Y_{domain} = s|Z_s) = D(Z_s) \rightarrow 1$) and label 0 for target domain ($p(Y_{domain} = s|Z_t) = D(Z_t) \rightarrow 0$). This discriminator is obtained through adversarial learning of ADDA method. Furthermore, we want to set the target latent representation Z_t to clusters to the number of source classes K_s . Centers of clusters are shown by Z_c for $c = 1, 2, \dots, K_s$.

The objective function of this model is as below consist of five terms, namely, classification L_{cla} , adversarial L_{adv} , encoder L_{enc} , clustering L_{dec} and dissimilarity L_{dis} .

Classification Objective Function Supervised models include source encoder $Z_s = E_s(X_s)$ and classifier $P(Y_s = k|Z_s) = C(Z_s)$ for $k = 1, 2, \dots, K_s$. These two components are pre-trained on source data $\{X_s, Y_s\}_{s=1}^{N_s}$ through maximizing the following objective function:

$$L_{cla} = \mathbb{E}_{(x,y) \sim P(X_s, Y_s)} [y^T \log\{C(E_s(x))\}] \tag{58}$$

Where y is K_s -dimensional one-hot vector representation for source labels Y_s . $C(Z_s)$ has softmax activation function; and $P(X_s, Y_s)$ is the source domain joint distribution. After training on source dataset, $E_s(X_s)$ and $C(Z_s)$ will be maintained fixed during adaptation.

Adversarial Objective Function To minimize the impact of variation caused by difference of source and target domain, a standard adversarial objective function is used, L_{adv} . Discriminator $D(\cdot)$ is trained by maximizing the following objective function:

$$L_{adv} = \mathbb{E}_{x \sim p(X_s)} \log D(E_s(x)) + \mathbb{E}_{x \sim p(X_t)} \log(1 - D(E_t(x))) \tag{59}$$

Where $p(X_s)$ and $p(X_t)$ are source and target marginal distribution. The following objective function is maximized over target encoder separately for generator:

$$L_{enc} = \mathbb{E}_{x \sim p(X_t)} \log D(E_t(x)) \tag{60}$$

Clustering Objective Function It is assumed that in latent space, there are k clusters which is equal to the number of distinct labels of source domain ($K = K_s$). Centroids of the clusters are indicated as Z_c where $c = 1, 2, \dots, K_s$. Citing the article Deep Unsupervised Embedding, the following KL divergence [89] is minimized as clustering objective function:

$$L_{dec} = KL(P||Q) = \sum_{i=1}^N \sum_{c=1}^{K_s} p_{ic} \log \frac{p_{ic}}{q_{ic}} \tag{61}$$

Where Q and P are respectively soft assignment and auxiliary distributions. Student's t distribution with α degrees of freedom [49] is used as soft assignment for Q :

$$q_{ic} = \frac{(1 + \frac{\|Z_i - Z_c\|^2}{\alpha})^{-\frac{\alpha+1}{2}}}{\sum_{c'} (1 + \frac{\|Z_i - Z_{c'}\|^2}{\alpha})^{-\frac{\alpha+1}{2}}} \tag{62}$$

Where q_{ic} estimates the probability of belonging sample i to cluster c . For all experiments, $\alpha = 1$ is set.

For the auxiliary distribution P , tightness of clusters are encouraged by raising q_{ic} to the power of 2 and normalizing:

$$p_{ic} = \frac{q_{ic}^2 / f_c}{\sum_{c'} q_{ic'}^2 / f_c} \tag{63}$$

Where $f_c = \sum_i q_{ic} \cdot p_{ic}$ above equation naturally leads to the self-reinforcement mechanism that encourage latent features Z_t to get close to the cluster centroids $\{Z_c\}_{c=1}^{K_s}$. In the experiments, for the balanced and imbalanced adaptations, K_s each cluster centroid Z_c is initialized by the mean of target samples in latent representation, and they are predicted by classifier $C(\cdot)$ as class c where $c = 1, 2, \dots, K_s$. So each cluster is known with one of the labels in the source domain.

Cluster Dissimilarity Objective Function One of the limitations of the above clustering objective function is that while it emphasizes clusters to be tight, but not necessarily emphasizes their purity. Cluster purity means each cluster includes samples from one specific class. Furthermore, it may result in domain collapse, meaning that clusters of a singular class place close together. To avoid these problems, it is desired to find a cluster-level solution that can predict centroids of different clusters as different classes. So $A = [C(Z_1) \dots C(Z_{K_s})]$ is defined as a $K_s * K_s$ matrix, its columns are label prediction distributions for K_s centroid of cluster Z_c using classifier $C(Z_c)$. Then cluster dissimilarity objective function is minimized as follow:

$$L_{dis} = \|A^T A - I\|_F \tag{64}$$

Where $\|\cdot\|_F$ is the Frobenius norm. based on the function above, $A^T A$ inputs are the similarity of the probability of class membership for all pairs of cluster centroids. It is assumed that $A^T A$ diagonal values force A columns to have unit norms. As a result, column A which represents the probability vectors will change into one-hot vectors. Minimizing the objective function will lead the K_s one-hot vector to be the membership vector of different predicted classes. Each cluster centroid will be allocated to a different class with high probability. The objective function will be as follow:

$$L_{dis} = \left(\sum_c \sum_{j \neq i} (a_c^T a_j)^2 \right)^{\frac{1}{2}} \tag{65}$$

Where $a_c = C(Z_c)$ is a column of A, and it is compared with $L_{dis} = \|A^T A - I\|_F$. The diagonal elements of $A^T A$ are not considered, since experiments resulted in more stabilize training with excluding diagonal elements.

- Partial domain adaptation:

In this type of adaptation, it is necessary to know the number of clusters (K) to initialize the center of clusters. Based on the previous assumption $K = K_s$, so source and target domains share the same label space. Although this assumption is not correct in partial domain adaptation since the correct number of target classes K_t are unknown. If k is considered a value between K_s and K_t ($K_t < K < K_s$), at least one cluster will be assigned to one of the labels in the source, not the target. Therefore, it seems difficult to guess the number of target classes.

A simple strategy is presented instead of trying to estimate the number of K_t , which includes target generation with several source data. In can be seen in Fig. 14 especially when a mini-batch of target data is used to update parameters of the model $\{\theta_{E_t}, \theta_D, Z_c\}$, data are generated by a sample of source data. For example, 50% of minibatch includes samples from the target and 50% of minibatch includes samples from source domain (without using labels). In this case, it is important to make sure that generated target data has instances from all classes of the source domain. As a result, partial domain adaptation will be converted to pseudo unbalanced domain adaptation, so that $K_t = K = K_s$, which would be appropriate for the rule we considered.

3.4 Auto-Encoder Regularizer

This category indicates a method which is motivated by stacked denoising auto-encoder. It added auto-encoder as a regularizer to force feature learning part to become robust.

3.4.1 Domain Adaptation with Adversarial Neural Networks and Auto-encoders (Dauto)

Zhao et al. [88] proposed the Dauto method based on DANN and denoising auto-encoder. DANN and denoising auto-encoder are the representation learning methods with different objective function. In the proposed method denoising auto-encoder is used as a regularizer to make features noise invariant. The goal is to learn feature representations that are invariant to the shift of source and target domains. In this method, labeled source data are available while the number of labeled target data is scarce.

The design combines two domain adaptation methods into a single framework. Two methods are representation based adversarial learning and auto-encoder is used for unsupervised pre-training. The proposed structure combines the advantages of both methods and results in a better generalization of the target domain.

The experiments are done based on Amazon benchmark for sentiment analysis. The proposed model is compared with similar methods and is shown that the model has more accurate classification than other methods. While in standard computational learning theory structure, train and test samples are extracted from a common distribution.

In this method, a situation is examined which contains two domains; the source and the target domains. Source domain has lots of labeled data while target domain has scarce labeled data. The goal of the domain adaptation algorithm is to generalize lots of source labeled data to the target domain. In this method domain adaptation problem solving is followed by a unified framework based on recent adversarial networks' advances. One solution is to develop domain invariant feature representations which have similar source and target distributions. Another solution is invariant feature representation through a stacked denoising auto encoder [74]. The encoders are neural networks that have an auto-encoder in each layer. It's proven that both of these methods are effective for generalizing learning unlabeled target data with the help of source labeled data. Both of the methods are representation based with different objective functions [88].

In this paper a unit neural network is proposed; it can learn invariant representations on the source and target domain shift and it is also reconstruction loss invariant. The model is based on DANN. The goal of the DANN method is learning representations that have enough information for source domain learning task and are simultaneously invariant to source and target domain shift. Being invariant to domain shift means the same information of the source domain can be used for the target domain as well. In this paper, the proposed method adds an auxiliary auto-encoder component to the structure which leads to learning representations as an unsupervised regularizer. The proposed model combines two complementary methods into a single structure. It is designed to achieve the following three goals simultaneously: (1) It learns representations that have useful information for the main learning task of the source domain. (2) It learns domain invariant features to be indistinguishable into source and target domains. (3) It is able to reconstruct the original input samples. To validate the effect of the proposed method on domain adaptation, the model is compared with the similar models on the Amazon Benchmark dataset for sentiment analysis. It is shown that while the model bears some computational overhead, it outperforms the compared models. DAuto structure has four main components. Feature extractor with θf parameter which is shown in green

in the Fig. 16. classification component with θ_y that is shown in blue. Domain classifier (discriminator) with θ_d parameter that is shown in red. And auto-encoder component with θ_r parameters is shown in purple. Feature extractor provides a common representation for all other components and then gradients obtained from the classification, domain classification, and auto-encoder are combined to update parameters of the feature learning part.

$G_f(\cdot; \theta_f)$ feature learning component. Which map input samples from \mathbb{R}^d to hidden representation \mathbb{R}^D .

$G_y(\cdot; \theta_y)$: classification component which is a function from \mathbb{R}^D to output representation Δ^k when prediction has k classes.

$G_d(\cdot; \theta_d)$: domain classification component which is a function from \mathbb{R}^D to $[0,1]$ representation.

$G_r(\cdot; \theta_r)$: auto-encoder component. The decoding function is $\mathbb{R}^D \rightarrow \mathbb{R}^d$. While the $G_f(\cdot; \theta_f)$ is the encoding function. $G_r(\cdot; \theta_r)$ has the same structure as $G_f(\cdot; \theta_f)$. $x, f^{(1)}, f^{(2)}, f^{(3)}$ are input samples, first layer, second layer, and third layer of feature extractor neural network. $f'^{(3)}, f'^{(2)}, f'^{(1)}, x$ are reconstructed features and reconstructed input samples from auto-encoder. For example, suppose the training part of the representation consists of three fc layers; the first layer has 100 units, the second layer has 200 and the third layer has 300 units. Therefore, the decoding section has exactly 300, 200 and 100 units. According to the design, the reconstruction loss function is defined as the regularizer of the DAUTO model is as below:

$$L_r = \frac{1}{2n} \sum_{i=1}^n L_r(x_i; \theta_f, \theta_r) + \frac{1}{2n} \sum_{i=n+1}^{2n} L_r(x_i; \theta_f, \theta_r) \tag{66}$$

The first n samples come from the source domain and second, n samples are from the target domain.

According to the layers, reconstruction loss is defined as below:

$$L_r(x; \theta_f, \theta_r) = \|x' - x\|_2^2 + \|f'^{(1)} - f^{(1)}\|_2^2 + \|f'^{(2)} - f^{(2)}\|_2^2 + \|f'^{(3)} - f^{(3)}\|_2^2 \tag{67}$$

Squared L2 distance is used as reconstruction loss; (other distance metrics can also be used). keep in mind L_r is depended on θ_f , since reconstructed hidden features are obtained from last layer of the encoder. This means, the decoder input features are output features of the feature extractor. When maximum information of data is preserved during encoding, the reconstruction loss is minimized. Finally, reconstruction loss is added to objective function:

$$\min_{\theta_f, \theta_y, \theta_r} \max_{\theta_d} \frac{1}{n} \sum_{i=1}^n L_y(x_i, y_i; \theta_f, \theta_y) - \lambda \left(\frac{1}{2n} \sum_{i=1}^n L_d(x_i; \theta_f, \theta_d) \right)$$

$$\begin{aligned}
& + \frac{1}{2n} \sum_{i=n+1}^{2n} L_d(x_i; \theta_f, \theta_d) \\
& + \mu \left(\frac{1}{2n} \sum_{i=1}^n L_r(x_i; \theta_f, \theta_r) \right. \\
& \left. + \frac{1}{2n} \sum_{i=n+1}^{2n} L_r(x_i; \theta_f, \theta_r) \right) \tag{68}
\end{aligned}$$

Note that second and third terms, L_d and L_r are considered as regularizers. This helps to learn feature representations that are informative for the main learning task while they are domain invariant feature representations at the same time.

3.5 Feature Augmentation

This category includes a generative adversarial method that used GANs to perform feature augmentation. This technique is used to improve the usage of GAN in unsupervised domain adaptation framework.

3.5.1 Adversarial Feature Augmentation for Unsupervised Domain Adaptation (DIFA)

Volpi et al. [75] proposed the DIFA method in 2018. This method has three steps in the training part in order to classify source and target data. Then in the test section, source and target data are given to pre-trained encoder, and labels of data are predicted by the classifier which is pre-trained in step 0. This paper's goal is expanding the following framework: (1) The trained extracted features should be domain invariant, and (2) Its training should be during feature augmentation. While the data augmentation is a well-established technique in the field of deep learning, feature augmentation has not yet reached that position. In this paper features are generated through a feature generator which is trained in a minimax game between GAN and source features. The results showed that both objectives, namely feature generation and being domain invariant has better or comparable performance compared to other unsupervised domain adaptation methods.

More specifically, features are generated through a feature generator trained by conditional GAN (CGAN) [41]. Minimax game is played with features instead of images and makes it possible to generate features based on the desired class. So CGAN generator can learn class distributions in feature space, therefore, generate the desired number of labeled feature vectors. The results of this study show that forcing to be domain invariant and feature generation in unsupervised domain adaptation leads to an increase in classifier accuracy.

The invariant feature extractor is inspired by the ADDA method, but it has two fundamental differences. First, the minimax game is played with generated features from the pre-trained model which results in feature generation. Second, the feature extractor component is trained to perform correctly on the source and target samples which means being domain invariant.

The proposed method: The goal of the method is to train a feature extractor component E_I . Its training procedure becomes more robust by generating source data in the feature space. The training procedure has three steps which are shown in Fig. 17. In step 0, a feature extractor component is trained based on source data ($C \circ E_S$). This step is necessary because a reference feature space and a reference classifier is needed. E_S is a ConvNet feature extractor component and C is a fully connected softmax layer whose size depends on the problem.

The following cross entropy loss should be minimized:

$$\min_{\theta_{E_S}, \theta_C} L_0 = E_{(x_i, y_i) \sim (X_S, Y_S)} H(C \circ E_S(x_i), y_i) \tag{69}$$

θ_{E_S}, θ_C are E_S and C parameters. X_S, Y_S are source samples distributions x_i and source labels y_i respectively. H indicated softmax cross entropy function.

In step 1, Model S is trained to generate features similar to the source features. It is possible to train it through minimax game between GAN and the extracted features from E_S . Using CGAN structure, the minimax game is defined as below:

$$\begin{aligned} \min_{\theta_S} \max_{\theta_{D1}} L_1 = & \\ & E_{(z, y_i) \sim (P_z(z), Y_S)} \|D_1(S(z \| y_i) \| y_i) - 1\|^2 \\ & + E_{(x_i, y_i) \sim (X_S, Y_S)} D_1 \|E_S(x_i) \| y_i\|^2 \end{aligned} \tag{70}$$

θ_S, θ_{D1} are D_1 and S parameters. $P_z(z)$ is a distribution from which noise samples are taken from. The least square GANs is used in this step and the next since it had better stability [40].

Eventually, in step 2, it is possible to train a domain invariant feature extractor component E_I . This training is possible through a minimax game between GAN and the extracted features from S . Then this module is combined with the pre-trained softmax layer (classifier) ($C \circ E_I$) so that the results can be checked on both source and target samples. All modules are neural networks that are trained by backpropagation [55]

E_I domain invariant encoder is trained through the following minimax game:

$$\begin{aligned} \min_{\theta_{E_I}} \max_{\theta_{D2}} L_2 = & \\ & E_{x_i \sim (X_s, X_t)} \|D_2(E_I(x_i)) - 1\|^2 \\ & + E_{(z, y_i) \sim (P_z(z), Y_S)} \|D_2(S(z \| y_i))\|^2 \end{aligned} \tag{71}$$

$\theta_{E_I}, \theta_{D2}$ are E_I and D_2 parameters. Since E_I is trained by both source and target domains, the feature extractor component would be domain invariant. This component maps source and target samples to a common feature space in which the features are indiscriminate from those generated by S . In order to generate indistinguishable features from source features, the E_I feature extractor component should be combined with the classifier layer of step 0 (C):

$$\tilde{y}_i = C \circ E_I(x_i) \tag{72}$$

x_i is source and target data and \tilde{y}_i is obtained labels.

3.6 Integrated Category and Domain Classifiers

The category includes a method that proposed an integrated category and domain classifier. It points out that the separate design of task and domain classifier has some shortcomings which this method outperforms them.

3.6.1 Discriminative Adversarial Domain Adaptation (DADA)

Tang et al. [66] referring to the DAN and DANN as examples of domain adversarial learning methods believes that separate design of domain classifiers and tasks cause a mode collapse.

Mode collapse happens when there is not good alignment of feature and category across the source domain and target domain. This limits the methods in aligning features' and tasks' distributions however they perform well in domain shift reduction. To solve this limitation, he proposed the discriminative adversarial domain adaptation (DADA) method. [28] proposed integrated models including tasks and domain classifiers in order to resolve the problem. To advance this path, based on this model's classifier Tang proposed the DADA model. The adversarial objective of DADA leads to discriminative mutual action between domain and category predictions. Tran et al. [69] implicitly aligns the joint distributions while DADA explicitly aligns them which results in target classification improvement.

The objective is as following minimax:

$$\begin{aligned} \min_F \mathcal{L}_F &= \lambda(\mathcal{L}^S + \mathcal{L}_F^t) - \mathcal{L}_{em}^t \\ \max_G \mathcal{L}_G &= \lambda(\mathcal{L}^S + \mathcal{L}_G^t) - \mathcal{L}_{em}^t \end{aligned} \tag{73}$$

Where λ is the trade-off hyper-parameter. The proposed source discriminator adversarial loss is inspired by binary cross entropy loss; it is shown as below:

$$\begin{aligned} \mathcal{L}^S(G, F) &= -\frac{1}{n_s} \sum_{i=1}^{n_s} [(1 - p_{K+1}(x_i^s)) \log p_{y_i^s}(x_i^s) \\ &\quad + p_{K+1}(x_i^s) \log(1 - p_{y_i^s}(x_i^s))] \end{aligned} \tag{74}$$

Where it establishes a relation between $p_{y^s}(x^s)$ of the prediction on the true category of x^s and $p_{k+1}(x^s)$ of the prediction on the domain of x^s . By using categorical probabilities, a target discriminative adversarial loss based on the integrated classifier. (F) is proposed as below:

$$\begin{aligned} \mathcal{L}_F^t(G, F) &= -\frac{1}{n_t} \sum_{j=1}^{n_t} \sum_{k=1}^K \bar{p}_k(x_j^t) \log \hat{p}_{K+1}^k(x_j^t) \\ \mathcal{L}_G^t(G, F) &= \frac{1}{n_t} \sum_{j=1}^{n_t} \sum_{k=1}^K \bar{p}_k(x_j^t) \log(1 - \hat{p}_{K+1}^k(x_j^t)) \end{aligned} \tag{75}$$

Where the k^{th} element of the vector \hat{p}^k for the k^{th} category is written as follows:

$$\hat{p}_{k'}^k(x) = \begin{cases} \frac{p_{k'}(x)}{p_k(x) + p_{k+1}(x)}, & k' = k, K + 1 \\ 0, & \text{otherwise} \end{cases} \tag{76}$$

Entropy minimization principle is as follow [16]:

$$\mathcal{L}_{em}^t(G, F) = \frac{1}{n_t} \sum_{j=1}^{n_t} \mathcal{H}(\bar{p}(x_j^t)) \tag{77}$$

Where $\mathcal{H}(\cdot)$ computes the entropy. Combination of 74 and 75 and 77 gives the proposed DADA objective.

Results of DADA method on Office-31 dataset is compared with No-adaptation, DANN, and DANN-CA, the proposed method is also compared with No-adaptation [20], DAN [13], DANN [11], DANN-CA [69], ADDA [71], MADA [50], VADA [62], GTA [59], MCD [58], and some other methods on Syn2Real-C dataset [52].

4 Results

The reviewed adversarial methods are categorized as shown in Table 1 based on their approaches. The performance of the reviewed methods, which has been experimented on two domain adaptation benchmark datasets namely Digit dataset and Office-31, are directly copied to Tables 2 and 3 respectively. Since we follow the experimental settings of the reviewed methods, we directly copied the results from corresponding papers. Their results demonstrate that some methods had improvement in comparison with others. The row of “source” in Tables 2 and 3 is a criterion which the accuracy values of the reviewed methods are compared. It reports the accuracies on target data based on training a classifier on the source data without applying any domain adaptation techniques.

Results on Digital classification For experiments on Digit datasets, the proposed approaches are compared with state-of-the-art methods. The results shown in Table 2 is verified due to the comparison of proposed methods and state-of-the-art methods. Non adapted classifier trained on the source data and the accuracies on target data indicates the source row of the table. The results of approaches are copied from published articles to observe correct comparison. As it is shown in Table 2, the method “Discriminative clustering for robust unsupervised domain adaptation” outperforms other methods in two direction adaptations. MNIST \rightarrow USPS, and USPS \rightarrow MNIST. Its average accuracy in all the directions shows 0.965 that is a bit higher than DIAL which its accuracy is 0.960.

Results on Office-31 In contrast to the large dataset of digital classification, the Office-31 has small number of data on both source and target domains. As shown on Table 3, DIAL method outperforms other proposed methods with average accuracy 0.868. As DIAL achieves comparable results to state-of-the-art methods on all six directions of adaptation; it shows that the method can be effective even with small number of samples. To have fair comparison of methods, the results of fine-tuning the AlexNet are shown on Table 3, and also since we want to compare experimental results, the accuracies are copied from corresponding articles.

Superior performance of DIAL on $A \rightarrow W$ and $A \rightarrow D$ tasks indicating improvement since their samples of the source and target domains are very different. While DIAL reached the accuracies 0.917 and 0.893 respectively on tasks mentioned, the second best performance is related to IDDA method which achieves the accuracy 0.822 on $A \rightarrow W$ and 0.824 on $A \rightarrow D$. The gap demonstrating how well DIAL performed. It also outperformed on $W \rightarrow A$ and $D \rightarrow A$ tasks where the size of the source domain is very small. The second best performance on these tasks is the “Discriminative clustering for robust unsupervised domain adaptation” method. As it is shown on Table 3, “Discriminative clustering for robust unsupervised domain adaptation” reached 0.595 accuracy on $W \rightarrow A$ task and 0.553 on $D \rightarrow A$ task which are improved to 0.714 and 0.717 respectively in DIAL method.

Since the t-sne visualization of data distribution is not shown on a common task and common dataset for all the reviewed papers, it is not possible to compare their t-sne plots which are shown in the articles. For instance, DIAL indicates the t-sne visualization for $A \rightarrow D$ task on Office-31 dataset, while DIFA plotted t-sne visualization for Digit datasets, and DADA (Dual) showed t-sne visualization of features of task $A \rightarrow W$ on Office-31 dataset. In case of having powerful computers and enough time, it can be a good criterion for a visualized comparison. Although two outperformed methods are depicted in Figs. 19 and 20. Figure 19 shows the t-sne embedding of “Discriminative clustering for Robust Unsupervised Domain Adaptation” which has better performance in comparison with “ADDA” on Digit dataset benchmark which consists of 10 classes from digits 0–9 but with different styles for task MNIST \rightarrow USPS. Figure 20 illustrates the t-sne visualization of DIAL adaptation

Table 1 Categories of reviewed methods based on their approaches

Methods	Added part to adversarial approach	Loss function distance	Considering negative-transfer	Considering partial-non-partial
ADDA	Basic adversarial structures			
DANN	Basic adversarial Structures			
RADA	Multi-class discriminator		✓	✓
M-ADDA	Using clustering	Triplet loss	✓	
Dauto	Auto-encoder regularizer			
DIFA	Feature augmentation			
MADA	Multi-class discriminator		✓	
DADA(disc)	Integrated category and Domain classifiers		✓	✓
IDDA	Multi-class discriminator		✓	
DADA(Dual)	Multi-class discriminator		✓	
UDA	Multi-class discriminator		✓	
DIAL	Using clustering	Center loss	✓	
Robust UDA ^a	Using clustering	KL-divergence		✓

^a Robust UDA is an abbreviation to the Discriminative Clustering for Robust Unsupervised Domain Adaptation.

Table 2 Accuracy on Digit datasets for unsupervised domain adaptation

Method	MNIST→USPS	USPS→MNIST	SVHN→MNIST	Avg
Source	0.771	0.634	0.598	0.667
ADDA	0.894	0.901	0.760	0.851
DANN	0.771	0.730	0.739	0.746
M-ADDA	0.952	0.940	–	0.630
DIFA	0.923	0.897	0.897	0.905
UDA	–	–	0.993	0.331
DIAL	0.950	0.972	0.958	0.960
Robust UDA ⁵	0.952	0.979	0.965	0.965

Table 3 Accuracy on Office-31 for unsupervised domain adaptation (AlexNet)

Method	A→W	D→W	W→D	A→D	D→A	W→A	Avg
Source	0.629	0.951	0.982	0.604	0.504	0.477	0.691
ADDA	0.751	0.970	0.996	0.677	0.525	0.573	0.748
DANN	0.730	0.964	0.992	0.719	0.507	0.535	0.741
MADA	0.785	0.998	1	0.741	0.560	0.545	0.771
IDDA	0.822	0.998	1	0.824	0.541	0.525	0.785
DIAL	0.917	0.971	0.998	0.893	0.717	0.714	0.868
Robust UDA ⁸	0.810	0.979	0.998	0.727	0.553	0.595	0.777

on Office dataset for A→D task, where “A” refers to Amazon domain (downloaded from Amazon.com), and “D” refers to DSLR domain (collected by a digital SLR camera). Both domains possess 31 classes; including backpack, pen, bike, bookcase, monitor, etc. For this task, as it is shown in Table 3, “DIAL” has the best accuracy among others.

Considering all shown accuracies, it is valuable to point out the desired performance. Although the proposed methods outperform state-of-the-art methods, it is important to reach superior performance on both Digit dataset and Office-31 on all directions and even on more benchmark datasets.

5 Conclusion

A model which is well-trained with a lot of data cannot be generalized from new data in case of domain shift existence. Domain adaptation methods reduce the effect of domain shift. Proposed domain shift reduction methods divide into three main categories: discrepancy-based methods, reconstruction-based methods, and adversarial domain adaptation methods. Adversarial domain adaptation method tries reducing the distance between two domains by the adversarial learning algorithm and with the help of a domain discriminator.

Adversarial domain adaptation methods are divided into two categories, generative models and non-generative models. Although generative models generate data close to real data but only in the case of existing slight differences in domains can have good discriminating performance; however, non-generative models propose better performance in the face of

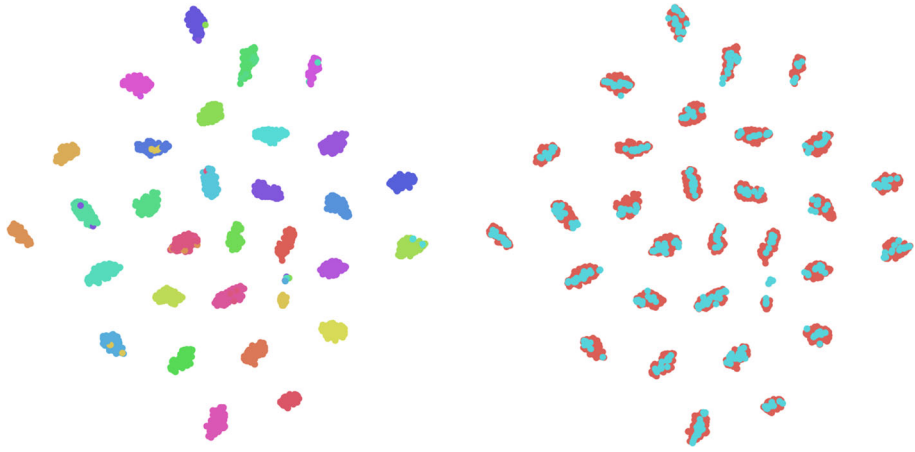


Fig. 20 The t-SNE visualization of features extracted by model for A→D task. Left Fig shows the distribution from category perspective and each color represents one class. Right Fig shows the distribution from domain perspective and points in red color represents samples from the source domain and blue points are samples from target domain [87]

greater domain differences. In this survey, we reviewed the articles proposed in the field of adversarial domain adaptation, and we had insight into their proposed structures.

Adversarial domain adaptation is formed based on the generative adversarial networks method which is made up of two networks competing against each other. In domain adaptation, GAN's principle is changed how a network extracts features that domain discriminator can be fooled while the discriminator is trying to discriminate data domains correctly as an opponent network in minimax game. Adversarial learning methods are promising methods for training deep neural networks. This method is able to generate samples within diverse domains, and it can also improve the ability to discriminate samples despite domain shift or data variation. Tzeng proposed a generalized structure for adversarial domain adaptation. Based on this structure, the proposed architecture would be unique by answering three questions. Is it a generator model or a discriminator model? Is feature extractor with weight sharing or without weight sharing? Which adversarial objective function is used?

Ganin et al. [11] proposed DANN method based on combining domain adaptation and deep feature learning within one training process. Tzeng et al. [71] proposed ADDA which was introduced an effective model among other competing domain-adversarial methods. Although its prior generative models showed persuasive visualizations, they hadn't optimal discriminative tasks and could be adapted to limited small shifts. On the other hand, prior discriminative models could handle some large domain shifts while didn't exploit a GAN-based loss. These two approaches are a forerunner among other adversarial domain adaptation methods. ADDA is surprisingly simple. The adaptation component of DANN can be added easily to any feed forward neural network model that is also trainable with back propagation. The main limitation of DIFA is similar to the limitation which exists in DANN and ADDA methods. Practically these methods make source and target features to be domain indistinguishable, while they don't guarantee that target samples will be mapped in the correct region of feature spaces. In other words, these methods focus on domain-level distribution alignment, while they don't guarantee the class-level distribution how is going to be aligned. The following adversarial domain adaptation methods are proposed to

encounter the mode collapse challenge; M-ADDA prevents false alignment to alleviate negative transfer. Discriminator Adversarial Domain Adaptation (DADA) method tries to learn domain invariant features, in a manner to apply a task classifier to both source and target domains. IDDA uses a multi-class discriminator to cover mode collapse. Dual Adversarial domain adaptation (DADA) approach is able to arrange domain level and class-level alignment simultaneously. Unsupervised DA is proposed to perform the joint distribution alignment. The conditional distributions of source and target domains are aligned by DIAL method. Discriminative clustering for robust unsupervised domain adaptation learns discriminative presentation of source and target practically. Some metric learning methods are proposed in adversarial domain adaptation approaches such as M-ADDA which its major strength is being non-parametric. DADA (Dual) method makes the extracted features more discriminative by using semi-supervised learning (SSL) regularization. Discriminative clustering for robust unsupervised domain adaptation is proposed to be robust to the differences in source and target labels distributions; that is known as partial domain adaptation where the set of target labels can be a subset of source labels. This scenario is challenging for the methods in real applications and may result in negative transfer.

Considering available research in the field of adversarial domain adaptation, it seems the major challenge is preventing negative transfer when the source and target come from different distributions. Since there is no guarantee of facing the same distribution for domains in real-world applications, for the future works It is more reassuring to utilize complicated benchmarks which include different label distributions such as testbed cross dataset [67], 3D Caricshop dataset [25], CASIA NIR-VIS dataset [31], e-PRIP VIS-sketch dataset [42], and also CUHK face sketch [76] and CUFSF from FERET dataset [85]. Further, reviewed methods indicate that adding some heuristics as a domain adaptation booster to the basic structure has resulted in higher accuracies. These heuristics such as multi-class discriminator, auto-encoder regularizer, feature augmentation and clustering part are added to the basic ADDA or DANN structures leading to improvements in performance, accuracy and robustness to negative transfer. Among the reviewed models, “DIAL” and “Robust UDA” that benefit from clustering approaches, could be a fruitful route for future research.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Bergamo A, Torresani L (2010) Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In: *Advances in neural information processing systems*. 181–189

2. Bousmalis K, Silberman N., Dohan D, Erhan D, Krishnan D (2017) Unsupervised pixel-level domain adaptation with generative adversarial networks. In: Proceedings of the 30th IEEE conference on computing vision pattern recognition, CVPR 2017. vol. 2017, 95–104
3. Carlucci FM, Porzi L, Caputo B, Ricci E, Bul'ov SR (2017) Autodial: automatic domain alignment layers. In: International conference on computer vision
4. Chen Ch, Chen Zh, Jiang B, Jin X (2019) Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation. In: AAAI conference on artificial intelligence
5. Cicek S, Soatto S (2019) Unsupervised domain adaptation via regularized conditional alignment. In: 2019 IEEE/CVF international conference on computer vision (ICCV)
6. Coates A, Ng Andrew, Lee H (2011) An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, 215–223
7. Denker JS, Gardner WR, Graf HP, Henderson D, Howard RE, Hubbard W, Jackel LD, Baird HS, Guyon I (1989) Advances in neural information processing systems. 1. chapter Neural Network Recognizer for Handwritten Zip Code Digits. 323–331
8. Ding Z, Fu Y (2018) Deep transfer low-rank coding for cross-domain learning. *IEEE Trans Neural Netw Learn Syst* 30(6):1768–1779
9. Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, Darrell T (2014) Decaf: A deep convolutional activation feature for generic visual recognition. In: International conference on machine learning (ICML). 647–655
10. Ganin Y, Lempitsky V (2015) Unsupervised domain adaptation by backpropagation. In: 32nd international conference on machine learning ICML 2015, vol 2, no. i 1180–1189
11. Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, Marchand M, Lempitsky V (2017) Domain-adversarial training of neural networks. *Adv Comput Vis Pattern Recognit* 17(9783319583464):189–209
12. Ghifary M, Kleijn WB, Zhang M, Balduzzi D, Li W (2016) Deep reconstruction classification networks for unsupervised domain adaptation. In: European conference on computer vision, 597–613
13. Long M, Cao Y, Cao Z, Wang J, Jordan MI (2018) Transferable representation learning with deep adaptation networks. In: IEEE transactions on pattern analysis and machine intelligence
14. Goodfellow I, Pouget-Abadie J, Mirza M, Balduzzi D, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. *Commun ACM* 63(11):139–144. <https://doi.org/10.1145/3422622>
15. Gong B, Shi Y, Sha F, Grauman K (2012) Geodesic flow kernel for unsupervised domain adaptation. In: IEEE conference on computer vision and pattern recognition. 2066–2073
16. Grandvalet Y, Bengio (2005) Semi-supervised learning by entropy minimization. In: Saul LK, Weiss Y, Bottou L (eds) Advances in neural information processing systems 17. MIT Press, 529–536
17. Gretton A, Smola AJ, Huang J, Schmittfull M, Borgwardt KM, Schölkopf B (2009) Covariate shift and local learning by distribution matching. MIT Press, Cambridge, MA, pp 131–160
18. Gupta S, Girshick R, aez PA, Malik J (2014) Learning rich features from rgb-d images for object detection and segmentation. In: European conference on computer vision (ECCV)
19. Han EHS, Karypis G, Kumar V (2001) Text categorization using weight adjusted k-nearest neighbor classification. In: PAKDD
20. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Computer vision and pattern recognition
21. Hoffer E, Ailon N (2015) Deep metric learning using triplet network. In: International workshop on similarity-based pattern recognition
22. Hu L, Kan M., Shan Sh, Chen X (2018) Duplex generative adversarial network for unsupervised domain adaptation. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 1498–1507
23. Isola P, Zhu JY, Zhou T, Efros A (2017) Image-to-image translation with conditional adversarial networks. In: Proceedings of the 30th IEEE conference on computing vision pattern recognition, CVPR. Vol. 2017, 5967–5976
24. Jiang Y, Wu Z, Wang J, Xue X, Chang S (2018) Exploiting feature and class relationships in video categorization with regularized deep neural networks. *IEEE Trans Pattern Anal Mach Intell* 40(2):352–364
25. Klare B (2012) Towards automated caricature recognition. In: 2012 5th IAPR international conference on biometrics (ICB). 139–146
26. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. University of Toronto
27. Kumar A, Sattigeri P, Wadhawan K, Karlinsky L, Feris RS, Freeman B, Wornell GW (2018) Co-regularized alignment for unsupervised domain adaptation, *NeurIPS*

28. Kurmi V. K., and Namboodiri V. P. Looking back at labels: a class based domain adaptation technique. In: 2019 international joint conference on neural networks (IJCNN), pp 1–8 (2019)
29. Radadji IH, Babanezhad R (2018) M-ADDA: unsupervised domain adaptation with deep metric learning. *Domain Adapt Vis Underst* 17–31
30. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324
31. Li S, Yi D, Lei Z, Liao S (2013) The CASIA NIR-VIS 2.0 face database. In: *Computer vision and pattern recognition workshops*
32. Liu MY, Tuzel O (2016) Coupled generative adversarial networks. *Adv Neural Inf Process Syst No. Nips*, 469–477
33. Long M, Zhu H, Wang J, Jordan MI (2016) Unsupervised domain adaptation with residual transfer networks. *Adv Neural Inf Process Syst Nips*. 136–144
34. Long M, Cao Y, Wang J, Jordan M (2015) Learning transferable features with deep adaptation networks. In: *International conference on machine learning*, pp 97–10
35. Long M, Cao Z, Wang J, Jordan MI (2018) Conditional adversarial domain adaptation. In: *Advances in neural information processing systems*, pp 1645–1655
36. Long M, Wang J, Ding G, Sun J, Philip SY (2013) Transfer feature learning with joint distribution adaptation. In: *2013 IEEE international conference on computer vision*, pp 2200–2207
37. Long M, Zhu H., Wang J, Jordan M (2017) Deep transfer learning with joint adaptation networks. In: *International conference on machine learning*. JMLR. org. 2208-2217
38. Lucic M, Kurach M, Michalski M, Bousquet O, Gelly S (2018) Are Gans created equal? A large-scale study. *Adv Neural Inf Process Syst vol. 2018*, pp 597–613
39. Madadi Y, Seydi V, Nasrollahi K, Hosseini R, Moeslund T (2020) Deep visual unsupervised domain adaptation for classification tasks: a survey. *IET Image Proc* 14(19):3283–3299
40. Mao X, Li Q, Xie H, Lau RYK, Wang Z (2016) Multiclass generative adversarial networks with the L2 loss function. *CoRR*, [arXiv:1611.04076](https://arxiv.org/abs/1611.04076)
41. Mirza M, and Osindero S (2014) Conditional generative adversarial nets. *CoRR*, [arXiv:1411.1784](https://arxiv.org/abs/1411.1784)
42. Mittal P, Jain A, Goswami G, Singh R, Vatsa M (2014) Recognizing composite sketches with digital face images via ssd dictionary. In: *International joint conference on biometrics*
43. Miyato T, Maeda S, Koyama M, Nakae K, Ishii S (2015) Distributional smoothing with virtual adversarial training. *arXiv preprint* [arXiv:1507.00677](https://arxiv.org/abs/1507.00677)
44. Miyato T, Maeda S, Koyama M, Ishii S (2017) Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Trans Pattern Anal Mach Intell* 41:1979–1993
45. Murez Z, Kolouri S, Kriegman D, Ramamoorthi R, Kim K (2018) Image to image translation for domain adaptation. In: *IEEE conference on computer vision and pattern recognition*, pp 4500–4509
46. Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. In: *NIPS workshop on deep learning and unsupervised feature learning*
47. Pan SJ, Yang Q (2010) A survey on transfer learning. *IEEE Trans Knowl Data Eng* 22(10):1345–1359
48. Pan SJ, Tsang TW, Kwok JT, Yang Q (2011) Domain adaptation via transfer component analysis. *IEEE Trans Neural Netw* 22(2):199–210
49. Peel D, McLachlan GJ (2000) Robust mixture modelling using the t distribution. *Stat Comput* 10(4):339–348
50. Pei Z, Cao Z, Long M, Wang J (2018) Multi-adversarial domain adaptation. In: *32nd AAAI conference on artificial intelligence AAAI 2018*, pp 3934–394
51. Peng X, Usman B, Kaushik N, Hoffman J, Wang D, Saenko K (2017) VisDA: The visual domain adaptation challenge. [arXiv:1710.06924](https://arxiv.org/abs/1710.06924)
52. Peng X, Usman B, Saito K, Kaushik N, Hoffman J, Saenko K (2018) Syn2real: A new benchmark for synthetic-to-real visual domain adaptation. [arXiv:1806.09755](https://arxiv.org/abs/1806.09755)
53. Peng KC, Wu Z, Ernst J (2018) Zero-shot deep domain adaptation. *Lect Notes Comput Sci (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*. Vol. 11215 LNCS, 793-810. https://doi.org/10.1007/978-3-030-01252-6_47
54. Rozantsev A, Salzmann M, Fua P (2018) Beyond sharing weights for deep domain adaptation. *IEEE Trans Pattern Anal Mach Intell*
55. Rumelhart DE, Hinton GE, Williams RJ (1986) Parallel distributed processing: explorations in the microstructure of cognition. *Chapter Learning Internal Representations by Error Propagation*. MIT Press, Cambridge, MA, USA, pp 318–362
56. Saenko K, Kulis B, Fritz M, Darrell T (2010) Adapting visual category models to new domains. In: *European conference on computer vision*. Springer, pp 213–22
57. Saito K, Ushiku Y, Harada T (2017) Asymmetric tri-training for unsupervised domain adaptation. *arXiv preprint* [arXiv:1702.08400](https://arxiv.org/abs/1702.08400)

58. Saito K, Watanabe K, Ushiku Y, Harada T (2017) Maximum classifier discrepancy for unsupervised domain adaptation. In: IEEE/CVF conference on computer vision and pattern recognition, 3723–3732
59. Sankaranarayanan S, Balaji Y, Castillo CD, Chellappa R (2018) Generate to adapt: aligning domains using generative adversarial networks. In: Computer vision and pattern recognition
60. Schroff F, Kalenichenko D, Philbin J (2015) FaceNet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. vol 07-12-June, pp 815–823
61. Sener O, Song HO, Saxena A, Savarese S (2016) Learning transferrable representations for unsupervised domain adaptation. In: Advances in neural information processing systems, pp 2110–2118
62. Shu R, Bui HH, Narui H, Ermon S (2018) A DIRT-t approach to unsupervised domain adaptation. In: 6th international conference on learning representation. ICLR 2018 - Conf. Track Proc. 1-19
63. Silberman N, Hoiem D, Kohli P, Fergus R (2012) Indoor segmentation and support inference from rgbd images. In: European conference on computer vision (ECCV)
64. Sun Y, Chen Y, Wang X, Tang X (2014) Deep learning face representation by joint identification-verification. In: Advances in neural information processing systems, pp. 1988–1996
65. Sun B, Saenko K (2016) Deep coral: correlation alignment for deep domain adaptation. In: ECCV. Springer, Berlin, 443–450
66. Tang H, Jia J (2020) Discriminative adversarial domain adaptation. In: AAAI 2020 - 34th AAAI conference on artificial intelligence, pp 5940–5947
67. Tommasi T, Tuytelaars T (2014) A testbed for cross-dataset analysis. In: ECCV workshop on transferring and adapting source knowledge in computer vision (TASK-CV)
68. Torralba A, Efros A (2011) Unbiased look at dataset bias. In: CVPR'11 (2011)
69. Tran L, Sohn K, Yu X, Liu X, Chandraker MK (2019) Gotta adapt 'em all: Joint pixel and feature-level domain adaptation for recognition in the wild. In: Computer vision and pattern recognition
70. Tzeng E, Hoffman J., Zhang N, Saenko K, Darrell T (2014) Deep domain confusion: maximizing for domain invariance. arXiv preprint [arXiv:1412.3474](https://arxiv.org/abs/1412.3474)
71. Tzeng E, Hoffman J, Saenko K, Darrell T (2017) Adversarial discriminative domain adaptation. In: Proceedings of the 30th IEEE conference on computer vision pattern recognition, CVPR 2017. Vol. 2017, pp 2962–2971. <https://doi.org/10.1109/CVPR.2017.316>
72. Tzeng E, Devin C., Hoffman J, Finn C, Abbeel P, Levine S, Seanko K Darrell T (2020) Adapting deep visuomotor representations with weak pairwise constraints. Published in WAFR 2016 Computer Science. 688-703. https://doi.org/10.1007/978-3-030-43089-4_44
73. Venkateswara H, Eusebio J, Chakraborty S, Panchanathan S (2017) Deep hashing network for unsupervised domain adaptation. In: Proceedings of the CVPR, pp. 5018–5027
74. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11:3371–3408
75. Volpi R, Morerio P, Savarese S, Murino V (2018) Adversarial feature augmentation for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5495–5504. <https://doi.org/10.1109/CVPR.2018.00576>
76. Wang X, Tang X (2009) Face photo-sketch synthesis and recognition. TPAMI 31(11):1955–1967
77. Wang M, Deng W (2018) Deep visual domain adaptation: a survey. Neurocomputing 312:135–153. <https://doi.org/10.1016/j.neucom.2018.05.083>
78. Wang R, Wang G, Heno R (2019) Discriminative clustering for robust unsupervised domain adaptation. arxiv
79. Wang Z, Jing B, Ni Y, Dong N, Xie P, Xing EP (2020) Adversarial domain adaptation being aware of class relationships. In: ECAI
80. Weinberger KQ, Saul LK (2009) Distance metric learning for large margin nearest neighbor classification. In: JMLR
81. Wen Y, Zhang K, Zhang MLBZ, Qiao Y (2016) A discriminative feature learning approach. In: Eccv, pp 499–515
82. Wilson G, Cook DJ (2018) A survey of unsupervised deep domain adaptation. arXiv
83. Xing Eric P, Zhang M, Ng Andrew Y, Jordan M, Russell S (2003) Distance metric learning with application to clustering with side-information. MIT Press, Cambridge, MA
84. Yuntao D, Zhiwen T., Qian Ch, Xiaowen Z, Yirong Y, Chongjun W (2020) Dual adversarial domain adaptation. arXiv preprint [arXiv:2001.00153](https://arxiv.org/abs/2001.00153)
85. Zhang W, Wang X, and Tang X (2011) Coupled information theoretic encoding for face photo-sketch recognition. In: 2011 IEEE CVPR, pp 513–520. IEEE

86. Zhang W, Ouyang W., Li W, Xu D (2018) Collaborative and adversarial network for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3801–3809
87. Zhang Y, Wang Y, Tian Q (2018) Domain-invariant adversarial learning for unsupervised domain adaption arXiv
88. Zhao H (2017) Domain adaptation with adversarial neural networks and auto-encoders
89. Zhuang F, Cheng X, Luo P, Pan SJ, He Q (2015) Supervised representation learning: transfer learning with deep autoencoders. In: IJCAI, pp 4119–4125

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.