Collaborative and Adversarial Network for Unsupervised domain adaptation

Weichen Zhang[1] Wanli Ouyang[1] Wen Li[2] Dong Xu[1]
[1] School of Electrical and Information Engineering, The University of Sydney
[2] Computer Vision Laboratory, ETH Zurich
{weichen.zhang,wanli.ouyang,dong.xu}@sydney.edu.au liwen@vision.ee.ethz.ch

Abstract

In this paper, we propose a new unsupervised domain adaptation approach called Collaborative and Adversarial Network (CAN) through domain-collaborative and domain-adversarial training of neural networks. We add several domain classifiers on multiple CNN feature extraction blocks[1]*, in which each domain classifier is connected to the hidden representations from one block and one loss function is defined based on the hidden presentation and the domain labels (e.g., source and target). We design a new loss function by integrating the losses from all blocks in order to learn domain informative representations from lower blocks through collaborative learning and learn domain uninformative representations from higher blocks through adversarial learning. We further extend our CAN method as Incremental CAN (iCAN), in which we iteratively select a set of pseudo-labelled target samples based on the image classifier and the last domain classifier from the previous training epoch and re-train our CAN model by using the enlarged training set. Comprehensive experiments on two benchmark datasets Office and ImageCLEF-DA clearly demonstrate the effectiveness of our newly proposed approaches CAN and iCAN for unsupervised domain adaptation.*

1. Introduction

In many visual recognition tasks, the training data used to learn a model and the testing data on which the model is applied often have different distributions. In order to enhance the generalization capability of learned models on the testing data, several domain adaptation technologies [1, 9, 5, 22, 11] were proposed to explicitly reduce the data distribution mismatch between the training samples in the source domain and the testing samples in the target domain.

Since deep learning methods have achieved excellent performance for many computer vision tasks including object recognition, a few deep transfer learning methods [21, 30, 25] were recently proposed to learn domain invariant features for visual recognition. These deep transfer learning methods can be roughly categorized as statistics approaches [19, 20, 24, 27], which exploit regularizer based

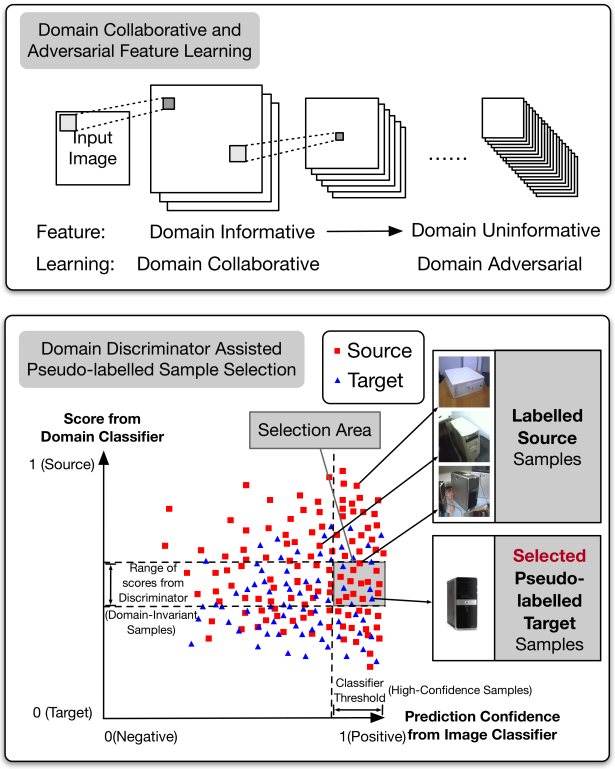

Figure 1. Illustration of two contributions in this work. Firstly, it is beneficial to learn domain informative features through collaborative learning at lower blocks and learn domain uninformative features through adversarial learning at higher blocks. Informative features such as corners and edges are useful for distinguishing not only images from different domains but also images from different classes, while uninformative features are useful for domain adaptation. Secondly, we iteratively select more pseudo-labelled target samples that have high prediction confidence from the image classifier, and are also considered as domain uninformative based on the last domain classifier from the previous training epoch.

[1]In this work, each block consists of several CNN layers.

on Maximum Mean Discrepancy (MMD), and adversarial learning based approaches [2, 3, 12, 13, 18, 26], which learn new representations through adversarial learning processes. To learn domain-uninformative representations, the recent work Domain Adversarial Training of Neural Network (DANN) [13] added one domain classifier at the last block and learn domain invariant features by minimizing the loss of this classifier and utilizing reversed gradient during the backpropagation process. Please refer to Section 2 for a brief review of existing domain adaptation methods.

In Section 3, we propose a new deep transfer learning method called Collaborative and Adversarial Network (CAN) by introducing a set of domain classifiers (also called as domain discriminators) into multiple blocks, in which each domain classifier is connected to the hidden representation from one block. Our work is based on the motivation that some characteristic information from target domain data may be lost after learning domain-invariant features with the recent method DANN [13]. Meanwhile, the representations at lower blocks are often low-level features such as corners and edges, which are expected to be informative for distinguishing images from different domains. To this end, we propose to learn one domain classifier at each block in order to learn domain-informative representations at lower blocks and domain-uninformative representations at higher blocks. After defining one loss function based on each hidden representation and domain labels (e.g., source and target) at one block, we integrate the losses from all blocks as a new loss function, in which the optimal combination weights are to be learnt. For the last block, the combination weight is set as a negative number as in DANN [13] in order to learn domain uninformative representation through adversarial learning. However, the learnt combination weights for lower blocks are often positive through collaborative learning, which indicates that the lower-block representations are domain informative. As a result, our Collaborative and Adversarial Network (CAN) can learn new representations which are not only domain-invariant for domain adaptation but also discriminant for image classification.

In Section 4, we further extend our CAN method as Incremental CAN (iCAN), which utilize domain discriminator to enlarge the training set by iteratively selecting pseudo-labelled target samples with different weights and re-training our CAN model with the enlarged training set. In order to select target samples that share similar data distributions as source samples, at each iteration, we select pseudo-labelled target samples that not only have high prediction confidence from the image classifier and but also are considered as domain uninformative based on the last domain classifier from the previous training epoch. By iteratively adding these samples to the training set, the overall data distribution of the enlarged training set is expected to gradually move from the source distribution to the target distribution. As a result, our work iCAN can gradually learn better representations and thus improves visual recognition performance in the target domain.

In Section 5, we perform comprehensive experiments on two benchmark datasets Office-31 and ImageCLEF-DA, and the results clearly demonstrate the effectiveness of our newly proposed approaches CAN and iCAN for unsupervised domain adaptation.

2. Related Works

Existing domain adaptation methods can be roughly categorized as feature (transform) based approaches [1, 11, 17], which aims to seek new domain-invariant features or learn new feature transforms for domain adaptation, and classifier based approaches [8, 9, 4, 7, 5, 29, 10], which directly learn the target classifiers (e.g., the SVM based classifiers) for domain adaptation.

Recently, several deep transfer learning methods were proposed to directly learn domain invariant features, which can be roughly categorized as statistic-based approaches [19, 20, 24, 27] and adversarial learning based approaches [2, 3, 12, 13, 18, 26]. In Deep Adaptation Network [19], the MMD-based regularizer was introduced between two hidden representations from both source and target domains at each higher layer in order to explicitly reduce the data distribution mismatch at multiple layers. This work is further extended by jointly learning transferable features and adaptive classifiers [20] and aligning the joint distributions of multiple hidden representations from several higher layers [21]. In contrast to these works, our method directly learn new representations through domain-collaborative and domain-adversarial training of neural networks without using any statistic-based regularizer.

Among adversarial learning based approaches, most works [2, 18] are based on Generative Adversarial Networks [15] by using generators to synthesize images or representations in different domains to learn domain invariant features. In [14], Grifary et al. proposed the Deep Reconstruction Classification Network (DRCN) method to learn a shared representation for classifying labelled source data and reconstructing unlabelled target data. In contrast to these approaches, our work does not use generators to synthesize new images or reconstruct target images. Our work is more related to DANN [13], which learns domain-uninformative features by inversely back-propagating the gradients from the loss related to the domain classifier. In addition to learn domain uninformative representations at the last layer as in [13], we additionally learn domain informative representations in lower layers through domain collaborative learning, to further improve visual recognition performance.

In our method iCAN, we progressively select pseudo-

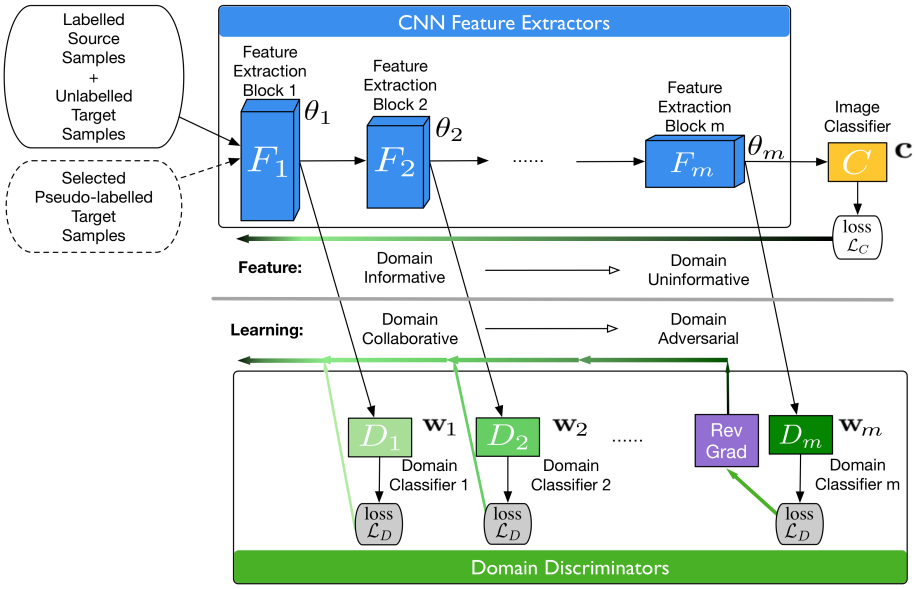

Figure 2. The whole architecture of the CAN model. Each feature extraction block consist of a group of CNN layers. Domain Classifier(also called Domain Discriminator) is composed of several FC layers, which distinguish which domain each sample belongs to. Each block F_k, $k = 1, \ldots, m$, is followed with a domain classifier D_k in order to learn domain informative and domain uninformative features. The pseudo-labelled target samples are selected by the image classifier and the last domain classifier, and then used as training data in iCAN.

labelled target samples in each iteration. Chen et al.[6] proposed a domain adaptation method based on co-training, in which the target samples with high prediction scores are selected for retraining in the next iteration. In [4], Bruzzone et al. proposed the Domain Adaptation Support Vector Machine (DASVM) method to iteratively select the unlabelled target domain data while simultaneously remove some labelled source samples. In contrast to these methods[4, 6], our work is a deep learning based approach that specifically learns new representations instead of the classifiers. Moreover, the domain classifiers are also used to help select and reweigh more reliable target samples.

3. CAN: Collaborative and Adversarial Network

In unsupervised domain adaptation, we are given a set labeled data from the *source domain* and a set of unlabeled data from the *target domain*, where the data distributions of two domains are different. The goal is to learn a target classifier which performs well on test data in the target domain.

Formally, we denote the source domain data as $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)|_{i=1}^{N_s}\}$, where $\mathbf{x}_i^s$ is the i-th source domain image, y_i^s is its category label, and N_s is the number of images in the source domain. Similarly, the target domain data is represented as $\mathcal{D}_t = \{\mathbf{x}_i^t|_{i=1}^{N_t}\}$, where $\mathbf{x}_i^t$ is the i-th target domain image, and N_t is the number of images in the target domain. For convenience, we also use $\mathcal{D} = \mathcal{D}_s \cup \mathcal{D}_t = \{(\mathbf{x}_i, d_i)|_{i=1}^{N}\}$ to denote the training images from both domains, where N is the total number of images, $d_i \in \{0, 1\}$ is the domain label for the i-th image with $d_i = 0$ as the source domain, and $d_i = 1$ as the target domain.

Many deep transfer learning approaches were proposed to learn domain-invariant representations for unsupervised domain adaptation [27, 19, 13]. Usually, a multi-task scheme is employed, where they aim to minimize the classification loss on the labeled source data, and also align the source and target domain distributions with a MMD-based regularizer.

However, the network aims to minimize the source domain classification loss, the learned representation may be less discriminative for classifying images in the target domain. To this end, we propose a new Collaborative and Adversarial Network (CAN) model, which simultaneously learn both domain-informative and domain uninformative features through domain collaborative and domain adversarial learning, such that the final representation is discriminant and also domain invariant for image classification. We introduce our new CAN model below.

3.1. Domain Informative Feature Learning

Domain informative feature such as corners and edges from lower layers are often useful for distinguishing which domain each sample belongs to. To keep the informative features when training the network, we propose to incorporate a domain classifier on each of the low-level blocks.

In particular, given an image $\mathbf{x}$, let us denote its feature representation extracted from a certain block as $\mathbf{f}$. We also

denote the feature extraction network before this block (inclusive) as F, then $\mathbf{f}$ can be deemed as the output of F, *i.e.*, $\mathbf{f} = F(\mathbf{x}; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the parameters for F.

To let $\mathbf{f}$ encode as much target information as possible, we propose to learn a domain classifier $D : \mathbf{f} \rightarrow \{0, 1\}$, which is used to predict whether the input image $\mathbf{x}$ belongs to the source domain or the target domain. Intuitively, if the feature representation $\mathbf{f}$ can be used to well distinguish which domain $\mathbf{x}$ comes from, sufficient target information should be encoded within $\mathbf{f}$. Then the objective for learning D can be written as,

$$\min_{\boldsymbol{\theta}, \mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_D \left(D(F(\mathbf{x}_i; \boldsymbol{\theta}); \mathbf{w}), d_i \right) \tag{1}$$

where $\mathbf{w}$ is the parameters for the domain discriminator D, d_i is the domain label, and $\mathcal{L}_D$ is the classification loss which is the cross entropy loss in this paper.

3.2. Domain Uninformative Feature Learning

On the other hand, domain uninformative features cannot clearly distinguish whether the sample is from the source domain or the target domain. By learning the domain discriminator and reversely back-propagating the domain gradient through the Gradient Reversal Layer [12, 13], the domain uninformative features can be learned. Specifically, they proposed to learn domain-uninformative features by confusing a domain discriminator through the domain adversarial training strategy. Following the terminology in the last subsection, the objective of DANN can be written as follows,

$$\max_{\boldsymbol{\theta}} \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_D \left(D(F(\mathbf{x}_i; \boldsymbol{\theta}); \mathbf{w}), d_i \right) \tag{2}$$

where the difference between the above equation and Eqn. (1) is that the network is trained to minimize the domain classification loss $\mathcal{L}_D$ in Eqn. (1) for learning domain informative features, while the network is trained to maximize $\mathcal{L}_D$ in Eqn. (2) for learning domain uninformative features.

As shown in DANN [12, 13], the maxmin problem in (2) can be implemented by using a gradient reversal layer, where the sign of gradient is reversed before passing into the network in the back-prorogation procedure. Thus, one can easily train the network by minimizing the loss with the conventional optimization techniques like the stochastic gradient descent method.

Intuitively, optimizing Eqn. (1) is to distinguish the samples from both domains, such that the feature representation is domain informative (or domain specific). On the other hand, optimizing Eqn. (2) tends to remove domain specific information, such that the samples from the two domains are similar to each other using the learned feature representations. Thus, optimizing Eqn. (2) is inevitably to make target informative information get lost, since such information does not help to confuse the domain discriminator. To learn domain informative representation at lower layers and domain uninformative representation at higher layers, we propose a collaborative and adversarial learning scheme below, which accommodates those opposite tasks into one framework, such that the learned feature is domain-invariant and also discriminant for image classification.

3.3. A Collaborative and adversarial Learning Scheme

To accommodate the two opposite tasks, domain informative and domain uninformative feature learning should be performed together. We propose a collaborative and adversarial learning scheme, in which we encourage the feature representation to keep as much domain informative feature as possible in lower layers/blocks, while the feature representation is enforced to be domain uninformative in higher layers/blocks. The two tasks are applied to multiple layers/blocks with different weights, such that feature representation goes smoothly from domain informative to uninformative when the samples are forwarded in the network from lower layers/blocks to higher layers/blocks.

In particular, suppose in total m blocks are used, where each block consist of a group of CNN layers, we build a domain discriminator after each block, leading to m domain discriminators. We assign a weight λ_k ($k = 1, \ldots, m$) for each domain discriminator, and automatically optimize the weights when we back-propagate the losses. Intuitively, a higher λ_k indicates the network tends to learn domain informative features at the k-th block. When $\lambda_k < 0$, the network tends to learn domain uninformative features at the k-th block.

Formally, let us denote $\boldsymbol{\theta}_k$ ($k = 1, \ldots, m$) as the network parameters before the k-th block (inclusive), and $\mathbf{w}_k$ is the parameters of the domain discriminator at the k-th block. For m blocks, we denote $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$, $\Theta_F = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m\}$. We also denote the loss term from a domain discriminator as $\mathcal{L}_D(\boldsymbol{\theta}, \mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_D(D(F(\mathbf{x}_i; \boldsymbol{\theta}); \mathbf{w}), d_i)$. The collaborative and adversarial learning scheme can be written as follows:

$$\begin{aligned} \min_{\Theta_F, \boldsymbol{\lambda}} \mathcal{L}_{CAN} = & \sum_{k=1}^{m-1} \lambda_k \min_{\mathbf{w}_k} \mathcal{L}_D(\boldsymbol{\theta}_k, \mathbf{w}_k) \\ & + \lambda_m \min_{\mathbf{w}_m} \mathcal{L}_D(\boldsymbol{\theta}_m, \mathbf{w}_m), \\ \text{s.t.} \quad & \sum_{k=1}^{m-1} \lambda_k = \lambda_0, \quad |\lambda_k| \leq \lambda_0, \end{aligned} \tag{3}$$

where $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_{m-1}\}$. λ_k is the weight for each block with $k = 1, \ldots, m-1$, which is automatically optimized during back-propogation. λ_0 and λ_m are the trade-off parameters, which will be discussed in Section 5.4. Automatically optimizing the loss weights λ_k's reduces the num-

ber of hyper-parameters, and more importantly, also allows multiple informative feature learning tasks to well collaborate with each other. When $\lambda_k \geq 0$, the corresponding subproblem is similar to the optimization problem in Eqn. (1), so we disable the gradient reverse layer, and encourage the corresponding discriminator to learn domain informative features. On the other hand, when $\lambda_k < 0$, the corresponding subproblem is similar to the max-min problem in Eqn. (2), so we enable the gradient reverse layer, and encourage the discriminator to learn domain uninformative features.

We can incorporate the loss L_{CAN} in (3) into any popular deep convolutional neural networks (CNNs) architecture (*e.g.*, AlexNet, VGG, ResNet, DenseNet, *etc*.) to learn robust features for unsupervised domain adaptation. Therefore, we jointly optimize the above loss with the conventional classification loss. Let an image classifier $C : \mathbf{f} \rightarrow \tilde{y_i}$. The image classification loss can be denoted as,

$$\mathcal{L}_{src} = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathcal{L}_C(C(F(\mathbf{x}_i^s; \Theta_F); \mathbf{c}), y_i^s), \tag{4}$$

where $\mathbf{c}$ is the parameters for the classifier C and $\mathcal{L}_C$ is the cross entropy loss for the classification task. Then, the final objective for our collaborative and adversarial network (CAN) can be written as,

$$\min_{\Theta_F, \mathbf{c}, \lambda_k \in \Lambda} \mathcal{L} = \mathcal{L}_{src} + \mathcal{L}_{CAN}, \tag{5}$$

where $\Lambda = \{\lambda_k | \sum_{k=1}^{m-1} \lambda_k = \lambda_0, |\lambda_k| \leq \lambda_0, k = 1, \ldots, m-1\}$ is the feasible set of λ_k's.

4. iCAN: Incremental CAN Model

In unsupervised domain adaptation, we have only unlabeled training samples in the target domain. To effectively use the unlabeled target samples for learning better representation, we further propose an incremental CAN model, in which we progressively select a set of pseudo-labeled target samples, and reweigh their classification losses for training the classifier in our CAN model. Those samples are selected and reweighted according to their prediction confidence scores by the image classifier and the domain discriminator, respectively, which are introduced in details as below.

4.1. Sample Selection with Classification Confidence Scores

We select the target samples according to their classification confidence scores. Specifically, let us denote $\{p_c(\mathbf{x}_i^t)|_{c=1}^{N_c}\}$ as the output from the softmax layer of the classifier in CNN, in which each $p_c(\mathbf{x}_i^t)$ is the probability that $\mathbf{x}_i^t$ belongs to the c-th category, and N_c is the total number of categories. Then, the pseudo-label of $\mathbf{x}_i^t$ can be obtained by choosing the category with the highest probability, *i.e.*, $\tilde{y_i^t} = \arg\max_c p_c(\mathbf{x}_i^t)$. We refer to the probability $p_{\tilde{y_i^t}}(\mathbf{x}_i^t)$ as the *classification confidence score*.

Intuitively, the higher the classification confidence score is, the more likely the target sample is correctly predicted. A common strategy for sample selection is to select target samples with their classification confidence scores above a certain threshold T. However, when training CNN, the classifier tends to always output low classification confidence scores at the initial stage, while gives high classification confidence scores at the later stage. Thus we propose to use the logistic function to adaptively adjust the threshold as follows,

$$T_C = \frac{1}{1 + e^{-\rho * A}}, \tag{6}$$

where ρ is a constant set as 3 in our experiments, A is the classification accuracy of the current image classifier measured by using the labeled source data as follows,

$$\begin{aligned} A &= \frac{1}{N_s} \sum_{i=1}^{N_s} I(y_i^s, \arg\max_c p_c(\mathbf{x}_i^s)), \\ I(a, b) &= \begin{cases} 1, & \text{if } a = b. \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{7}$$

With the above adaptive threshold scheme, we then define our sample selection function as follows,

$$\begin{aligned} s(\mathbf{x}_i^t) &= \sigma(p_{\tilde{y_i^t}}(\mathbf{x}_i^t), T_C), \\ \sigma(a, b) &= \begin{cases} 1, & \text{if } a > b. \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

4.2. Sample Selection and Weighting with Discriminator Prediction Scores

Moreover, considering there is a distribution difference between the target domain and the source domain, the classification prediction score is usually biased. As a result, the selected samples based on Eqn. (8) are likely to be biased to the the source distribution. To alleviate such bias, we further propose a sample reweighing strategy, such that a selected sample is assigned a higher weight when it is not close to the source samples, and vice versa. In particular, we use the prediction scores from the domain classifier trained in our CAN model. Recall that a domain classifier aims to predict an input sample as 0 if it is from the source domain, and 1 if it is from the target domain. So, if the probability output by the domain classifier is close to 0.5, the domain classifier is difficult to determine which domain the

sample belongs to. In other words, the sample in this case is domain-indistinguishable, which should be selected and reweighed with higher weights when learning our model.

To this end, we design a weighting function to assign higher weights to samples having a domain probability close to 0.5, and to assign lower weights to other samples. Let us denote the output from a domain classifier as $d(\mathbf{x}_i^t)$. Inspired by [28], we employ the sample weights function as follows,

$$h(\mathbf{x}_i^t; z) = -|z * (d(\mathbf{x}_i^t) - 0.5)|^\alpha + 1 \quad (9)$$

where $|\cdot|$ is the absolute value function, α is a constant that controls the curvature, z is a learnable parameter that controls the range of the threshold function. Based on the above function, we design our target sample selection and weighting function as follows,

$$w(\mathbf{x}_i^t; z) = \beta\sigma(h(\mathbf{x}_i^t; z), 0) + \max(h(\mathbf{x}_i^t; z), 0) \quad (10)$$

where $\sigma(\cdot)$ is a gating function as in Eqn. (8), β is constant that controls the minimum weight.

Basically, the target sample selection and the weighting function in (9) gives a smooth response for the input scores from the domain classifier which is centered at 0.5, and weighting function in (10) further constrains the range of weights as $[\beta, 1+\beta]$ when $h(\mathbf{x}_i^t; z) > 0$. Note that the sample is unselected when $h(\mathbf{x}_i^t; z) \leq 0$. The constants α and β controls the shape of the function. We use $\alpha = 4$ and $\beta = 0.3$ in our experiments.

4.3. Overview

After performing pseudo-labeled target sample selection and reweighing, then the classification loss using pseudo-labeled target samples can be written as,

$$\mathcal{L}_{tar} = \frac{1}{N_t}\sum_{i=1}^{N_t} s(\mathbf{x}_i^t) w(\mathbf{x}_i^t; z) \mathcal{L}_C(C(F(\mathbf{x}_i^t; \Theta_F); \mathbf{c}), \tilde{y_i^t}) \quad (11)$$

where $\mathcal{L}_C$ is the cross entropy loss in Eqn. (4), and $s(\mathbf{x}_i^t)$ and $w(\mathbf{x}_i^t; z)$ are respectively the selection indicator and weight obtained using (8) and (10). $\mathcal{L}_{tar}$ is defined at the same location as $\mathcal{L}_C$ in the network.

The entire procedure of training our incremental CAN (iCAN) model is depicted in Algorithm 1. In the first stage, we train our CAN model by optimizing the problem in Eqn. (5). In the second stage, we fine-tune the pretraind CAN model by adding our newly selected pseudo-labeled target samples. At each iteration, we first respectively calculate the sample selection indicator $s(\mathbf{x}_i^t)$ and the sample weight $w(\mathbf{x}_i^t; z)$ in the forward process. Then, we perform back-propagation to jointly optimize the CAN loss in (5), and the target sample classification loss in (11) by optimizing the total objective function

$$\min_{\Theta_F, \mathbf{c}, z, \lambda_k \in \Lambda} \mathcal{L}_{total} = \mathcal{L}_{src} + \mathcal{L}_{tar} + \mathcal{L}_{CAN} \quad (12)$$

The learnable weight z in the weight function is also optimized. We repeat this process until the model converges.

Algorithm 1 Incremental Collaborative and Adversarial Network (iCAN).

1: **Input**: source domain labeled samples $\{(x_i^s, y_i^s)|_{i=1}^{N_s}\}$, target domain unlabeled samples $\{(x_i^t)|_{i=1}^{N_t}\}$.
Stage-1:
2: Train an initial CAN model by optimizing Eqn. (5).
Stage-2:
3: **loop** until max_iter is reached:
4: Calculate the average accuracy from the current model on the source samples in the mini-batch by using Eqn. (7).
5: Perform sample selection on pseudo-labelled target samples in the mini-batch by using Eqn. (8).
6: Calculate sample weights on the selected pseudo-labelled target samples in the mini-batch by using Eqn. (10).
7: Train the model by optimizing Eqn. (12) with all samples.

5. Experiments

We evaluate the proposed unsupervised domain adaptation method on two datasets Office-31 [23] and ImageCLEF-DA [21], and investigate the components of our models in details.

5.1. Implementation Details

Our implementation is based on the PyTorch framework[2]. We utilize the pre-trained ResNet50 model as the CNN feature extractor and learn the parameters of the feature extraction layers with 0.1 times learning rate of the domain discriminators and the image classifier. In the network, the feature extractor layers are grouped into four blocks. We add four domain discriminators after the 10th, 22th, 40th and 49th layer. We use stochastic gradient descent(SGD) for optimization.

The learning rate η_0 decreases gradually after each iteration from $\eta_0 = 0.0015$, and we use the same INV learning rate decrease strategy as in DANN [13]. We also progressively change the learning rate after each iteration from 0 to η_0 instead of setting the learning rate to be η_0 for the first iteration. Following the setting in DANN [12, 13], an adaptation factor is used for controlling the learning rate of the domain discriminator. In our experiment, we set the batch size, momentum, and weight decay as 16, 0.9 and 3×10^{-4}, respectively. In each domain adaptation task, we utilize all samples from both source and target domains.

[2]https://github.com/pytorch/pytorch

Due to different dataset size in different tasks, we utilize different total training epoch length and different number of pseudo-labelled target samples in one batch. For training epoch length, we decide it based on the fixed total iteration for different tasks, similar to other settings in [13, 20, 21]. As in DANN, we keep the same number of labelled and unlabelled samples in one batch. Suppose the number of samples in one batch is 16, the number of labelled source data and the unlabelled target data in one batch are both 8 in DANN. In our iCAN model, let us denote N_s and N_p as the number of source samples and the number of selected pseudo-labelled target samples in one training epoch, respectively. In one mini-batch, we use $8(1-\frac{N_p}{N_s+N_p})$ labelled source samples, $8\frac{N_p}{N_s+N_p}$ pseudo-labelled target samples, $8(1-\frac{N_p}{N_s+N_p})$ unlabelled target samples and $8\frac{N_p}{N_s+N_p}$ source samples without using their category label. In this way, the number of labelled/pseudo-labelled samples from both domains is equal to the number of unlabelled data from both domains, and the number of samples from each domain is also equal, which is helpful for model convergence. We use the domain labels of all the source and target samples.

5.2. Datasets and State-of-the-art Approaches

Office-31 [23]. Office-31 dataset is a benchmark dataset for evaluating different domain adaptation methods for object recognition, which contains 4,110 real images from 31 classes. It contains three domains Amazon (A), Webcam (W) and Dslr (D) subset. We utilize the common evaluation protocol on all six settings as in [19].

ImageCLEF-DA [21]. This dataset is built for ImageCLEF 2014 domain adaptation challenge[3]. It contains 4 subsets, including Caltech-256 (C), ImageNet ILSVRC 2012 (I), Bing (B) and Pascal VOC 2012 (P). Each of the subset contains 12 classes and each class has 50 images, which results in total 600 images for one subset. Similarly, we follow [21] to report the results for six settings.

We compare our method with the basic deep learning methods(ResNet50) and the existing deep domain adaptation learning methods based on ResNet50. For the basic deep learning methods, we use only source samples to finetune the ResNet50 [16] model that is pretrained based on ImageNet. For deep transfer learning methods, we report the results of Deep Domain Confusion (DDC) [27], Deep Adaptation Network (DAN) [19], Residual Transfer Network (RTN) [20] and Joint Adaptation Network (JAN) [21]. In addition, we also report the results of DANN [13] using our own implementation.

5.3. Experimental Results

The results on the Office-31 and ImageCLEF-DA datasets are reported in Table 1 and Table 2, respectively. In terms of the average accuracy, our newly proposed method CAN outperforms several baseline methods, including the ResNet50 [16], DDC [27], DAN [19], RTN [20] and DANN [21] on both datasets, and our CAN is also comparable with the recent work JAN [21] on the ImageCLEF-DA dataset. The results demonstrate the effectiveness of our method CAN by simultaneously learning domain informative and domain uninformative representations through collaborative and adversarial training of neural networks. Moreover, our work iCAN achieves the best average accuracies on both datasets, which indicates that it is beneficial to iteratively select pseudo-labelled target samples with the guidance of image classifier and domain classifiers and re-train the CAN model by using the enlarged training dataset.

[3]http://imageclef.org/2014/adaptation

Model	A→W	W→A	A→D	D→A	W→D	D→W	Avg.
ResNet50[16]	73.5	59.8	76.5	56.7	99.0	93.6	76.5
DDC[27]	76.0	63.7	77.5	67.0	98.2	94.8	79.5
DAN[19]	80.5	62.8	78.6	63.6	99.6	97.1	80.4
RTN[20]	84.5	64.8	77.5	66.2	99.4	96.8	81.6
DANN[13]	79.3	63.2	80.7	65.3	99.6	97.3	80.9
JAN[21]	86.0	70.7	85.1	69.2	99.7	96.7	84.6
CAN(ours)	81.5	63.4	85.5	65.9	99.7	98.2	82.4
iCAN(ours)	92.5	69.9	90.1	72.1	100.0	98.8	87.2

Table 1. Comparison of different methods for unsupervised domain adaptation on the Office-31 dataset.

Model	I→P	P→I	I→C	C→I	C→P	P→C	Avg.
ResNet50[16]	74.6	82.9	91.2	79.8	66.8	86.9	80.4
DAN[19]	74.5	82.2	92.8	86.3	69.2	89.8	82.5
RTN[20]	74.6	85.8	94.3	85.9	71.7	91.2	83.9
DANN[13]	75.6	84.0	93.0	86.0	71.7	87.5	83.0
JAN[21]	76.8	88.0	94.7	89.7	74.2	91.7	85.8
CAN(ours)	78.2	87.5	94.2	89.5	75.8	89.2	85.7
iCAN(ours)	79.5	89.7	94.7	89.9	78.5	92.0	87.4

Table 2. Comparison of different methods for unsupervised domain adaptation on the ImageCLEF-DA dataset.

		λ_m					
		-0.1	-0.2	-0.4	-0.5	-0.6	-0.8
λ_0	0.1	88.0	88.1	88.7	89.1	91.3	90.6
	0.2	87.0	89.8	91.4	90.9	90.6	90.5
	0.3	84.9	88.5	92.1	91.5	91.3	90.8
	0.4	85.2	89.1	91.3	92.5	92.5	92.4
	0.5	85.1	89.9	90.4	91.8	91.9	90.3
	0.6	84.3	87.8	88.2	90.8	91.2	88.2

Table 3. Performance of our iCAN on the Office-31 dataset (A→W) by using different parameters λ_0 and λ_m.

5.4. Analysis of our method

In this subsection, we take the Office-31 dataset (A→W) as an example to analyse our method. In our method

Model	A→W	W→A	A→D	D→A	W→D	D→W	Avg.
DANN	79.3	63.2	80.7	65.3	99.6	97.3	80.9
CAN	81.5	63.4	85.5	65.9	99.7	98.2	82.4
DANN+TSS(I)	80.0	65.9	81.9	66.5	99.7	97.8	82.0
CAN+TSS(I)	86.6	67.7	86.8	69.9	99.8	98.3	84.9
DANN+TSS(I+D)	85.3	69.6	86.1	70.7	99.8	98.3	85.0
iCAN	92.5	69.9	90.1	72.1	100.0	98.8	87.2

Table 4. Evaluation of different methods for selecting pseudo-labelled target samples on the Office-31 dataset (A→W).

CAN, we use two parameters λ_0 and λ_m to control λ_k $(k = 1, \ldots, m-1)$ and balance the losses from domain classifiers at different blocks. In this section, we take the Office-31 dataset (A→W) as an example to evaluate our iCAN when using different parameters λ_0 and λ_m. Specifically, we set λ_0 in the range of {0.1, 0.2, 0.3, 0.4, 0.5, 0.6} and λ_m in the range of {-0.1, -0.2, -0.4, -0.5, -0.6, -0.8}. For fair comparison, we still employ ResNet50 as the CNN feature extractor and use the same parameter settings for other parameters. From the results shown in Table 3, we observe that the results are relatively stable when setting λ_0 between 0.2 and 0.5 and λ_m between -0.4 and -0.6. We also observe that the learned λ_k's at lower blocks are often larger than those at higher blocks, This shows that the learned representations can be gradually changed from informative representations at lower blocks to uninformative representations at higher blocks.

We also compare our new method based on both image classifier and domain classifiers for selecting pseudo-labelled target samples with a simple target sample selection method using only image classifier as in Eqn. (8). Such a simple target sample selection (TSS) method can be used together with DANN and our CAN, leading to two baseline methods DANN+TSS(I) and CAN+TSS(I), respectively. Similarly, our new target sample selection method can also be used together with DANN and our CAN, which result in DANN+TSS(I+D) and iCAN, respectively. The results are reported in Table 4. From the results, we observe that DANN+TSS(I) outperforms DANN and CAN+TSS(I) is better than CAN, which indicates that it is useful to select pseudo-labelled target samples to improve the classification performance. Moreover, DANN+TSS(I+D) is better than DANN+TSS(I) and iCAN achieves the best results, which shows the effectiveness of our newly proposed method for selecting pseudo-labelled target samples.

In Table 5, we also evaluate our iCAN by using our new target sample selection method with different parameters α and β, in which we set α in the range of {1, 2, 3, 4} and β in the range of {0.1, 0.2, 0.3, 0.5}. Using different values of α and β, different weights will be assigned to the selected pseudo-labelled target samples. We observe that our method iCAN using different values of α and β outperform the baseline algorithms CAN and CAN+TSS(I) (note

		β			
		0.1	0.2	0.3	0.5
α	1	86.9	87.6	87.9	88.6
	2	90.4	90.0	87.6	89.9
	3	88.1	91.8	90.2	90.9
	4	89.3	90.6	92.5	89.6

Table 5. Performance of our method iCAN on the Office-31 dataset (A→W) by using different parameters α and β. By using different values of α and β, different weights will be assigned to the selected pseudo-labelled target samples.

their results are 81.5% and 86.6%, receptively, as reported in Table 4). Our iCAN with the best setting of α and β has accuracy 92.5%. This again shows the effectiveness of our new target sample selection criterion. In addition, we also observe that the results of our method iCAN are relatively stable when setting α in [3, 4] and β in [0.2, 0.3].

6. Conclusion

In this paper, we have proposed a new deep learning method called Collaborative and Adversarial Network (CAN) for unsupervised domain adaptation. Different from the existing works that learn only domain uninformative representations through domain adversarial learning, our method CAN additionally learns domain informative representations through domain collaborative learning. We have also proposed a new criterion to select pseudo-labelled target samples and developed an iterative approach called incremental CAN (iCAN), in which we iteratively perform sample selection and re-train our network by using the enlarged training set. Our newly proposed approaches have achieved better performance than several state-of-the-art methods on two benchmark datasets, which clearly demonstrates the effectiveness of our newly proposed approaches for unsupervised domain adaptation.

7. Acknowledgement

Wanli Ouyang is supported by SenseTime Group Limited.

References

[1] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Unsupervised domain adaptation by domain invariant projection. In *ICCV*, pages 769–776, 2013.

[2] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, pages 3722–3731, 2017.

[3] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *NIPS*, pages 343–351, 2016.

[4] L. Bruzzone and M. Marconcini. Domain adaptation problems: A DASVM classification technique and a circular validation strategy. *TPAMI*, 32(5):770–787, 2010.

[5] L. Chen, W. Li, and D. Xu. Recognizing RGB images by learning from RGB-D data. In *CVPR*, pages 1418–1425, 2014.

[6] M. Chen, K. Q. Weinberger, and J. Blitzer. Co-training for domain adaptation. In *NIPS*, pages 2456–2464, 2011.

[7] L. Duan, I. W. Tsang, and D. Xu. Domain transfer multiple kernel learning. *TPAMI*, 34(3):465–479, 2012.

[8] L. Duan, I. W. Tsang, D. Xu, and T.-S. Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*, pages 289–296. ACM, 2009.

[9] L. Duan, D. Xu, and I. W.-H. Tsang. Domain adaptation from multiple sources: A domain-dependent regularization approach. *T-NNLS*, 23(3):504–518, 2012.

[10] L. Duan, D. Xu, I. W.-H. Tsang, and J. Luo. Visual event recognition in videos by learning from web data. *T-PAMI*, 34(9):1667–1680, 2012.

[11] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *CVPR*, pages 2960–2967, 2013.

[12] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015.

[13] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(59):1–35, 2016.

[14] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV*, pages 597–613, 2016.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[17] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR*, pages 1785–1792, 2011.

[18] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *NIPS*, pages 469–477, 2016.

[19] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105, 2015.

[20] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *NIPS*, pages 136–144, 2016.

[21] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, pages 2208–2217, 2017.

[22] L. Niu, W. Li, and D. Xu. Exploiting privileged information from web data for action and event recognition. *IJCV*, 118(2):130–150, 2016.

[23] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, pages 213–226, 2010.

[24] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, pages 443–450, 2016.

[25] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, pages 4068–4076. IEEE, 2015.

[26] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pages 2962–2971, 2017.

[27] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[28] Z. Wang, H. Li, W. Ouyang, and X. Wang. Learnable histogram: Statistical context features for deep neural networks. In *ECCV*, pages 246–262, 2016.

[29] Z. Xu, W. Li, L. Niu, and D. Xu. Exploiting low-rank structure from latent domains for domain generalization. In *ECCV*, pages 628–643, 2014.

[30] X. Zeng, W. Ouyang, M. Wang, and X. Wang. Deep learning of scene-specific classifier for pedestrian detection. In *ECCV*, pages 472–487. Springer, 2014.