Domain Adaptive Neural Networks for Object Recognition

3 authors:

Muhammad Ghifary
Weta Digital Ltd.
11 PUBLICATIONS **867** CITATIONS

SEE PROFILE

W. Bastiaan Kleijn
Victoria University of Wellington
360 PUBLICATIONS **6,071** CITATIONS

SEE PROFILE

Mengjie Zhang
Victoria University of Wellington
680 PUBLICATIONS **9,542** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Evolutionary Image Descriptors View project

Generating Benchmark Feature Selection Datasets with Genetic Programming View project

Domain Adaptive Neural Networks for Object Recognition

Muhammad Ghifary, W. Bastiaan Kleijn, and Mengjie Zhang

School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington, New Zealand
{muhammad.ghifary,bastiaan.kleijn,mengjie.zhang}@ecs.vuw.ac.nz

Abstract. We propose a simple neural network model to deal with the domain adaptation problem in object recognition. Our model incorporates the Maximum Mean Discrepancy (MMD) measure as a regularization in the supervised learning to reduce the distribution mismatch between the source and target domains in the latent space. From experiments, we demonstrate that the MMD regularization is an effective tool to provide good domain adaptation models on both SURF features and raw image pixels of a particular image data set. We also show that our proposed model, preceded by the denoising auto-encoder pretraining, achieves better performance than recent benchmark models on the same data sets. This work represents the first study of MMD measure in the context of neural networks.

Keywords: Domain Adaptation, Neural Networks, Representation Learning, Transfer Learning, Maximum Mean Discrepancy

1 Introduction

In learning-based computer vision, the probability distribution mismatch between the training and test samples is an essential problem to overcome for the success in real world scenarios. For example, suppose we have an object recognizer learned from a training set containing objects with specific viewpoints, backgrounds, and transformations. It is then applied to an environment with a similar object category, but different viewpoints, backgrounds, and transformations condition. This situation might happen due to a lack of labeled data representing the target environment or insufficient knowledge regarding to the target condition. A good recognition model on this setting can not be guaranteed if it is trained by using traditional learning techniques.

Methods to address the distribution mismatch have been investigated under the names of *domain adaptation*[1] and *transfer learning*. More specifically, given a training set $\{\mathbf{x}_s^{(i)}, y_s^{(i)}\}_{i=1,\dots,n_s}$ and test set $\{\mathbf{x}_t^{(j)}, y_t^{(j)}\}_{j=1,\dots,n_t}$ sampled from

---

[1] In this area, the term "domain" and "probability distribution" are considered to be identical.

a distribution $\mathcal{D}_s$ and $\mathcal{D}_t$ respectively, the goal is to predict the target labels $y_t^{(j)}$ when $\mathcal{D}_s \neq \mathcal{D}_t$ and the information about $y_t^{(j)}$ is not sufficient. In recent years, many solutions to this problem have been proposed for computer vision applications (Gong et al., 2012, Gopalan et al., 2011, Long et al., 2013, Saenko et al., 2010) and natural language processing (Daumé-III, 2009, Pan and Yang, 2010).

In image recognition, the Office data set (Saenko et al., 2010) has become a standard image set to evaluate the performance of domain adaptation models. The standard evaluation protocol on this data set is based on using the SURF feature descriptor (Bay et al., 2008) as inputs to the model. However, the utilization of such a descriptor usually needs a careful engineering to get good discriminative features. Furthermore, it may bring more complexity in the context of real time feature extraction processes. It is therefore worthwhile to build good models without using any handcrafted feature descriptors.

Representation or feature learning provides a framework to reduce the dependency on manual feature engineering (Bengio et al., 2012). Examples that can be considered as representation learning are Principal Component Analysis (PCA), Independent Component Analysis (ICA), Sparse Coding, Neural Networks, and Deep Learning. In deep learning, the greedy layer-wise unsupervised training, which is known as the *pretraining*, has played an important role for the success of deep neural networks (Bengio et al., 2007, Erhan et al., 2010). Although representation learning-based techniques have brought some successes over many applications, methods to address the distribution mismatch have not yet been well studied.

In this work, we propose a simple neural network model with good domain adaptation performance on raw image pixels. More particularly, we utilize a non-parametric probability distribution distance measure, i.e, the Maximum Mean Discrepancy (MMD), as a regularization embedded in the supervised backpropagation training. MMD is used to reduce the distribution mismatch between two hidden layer representations induced by samples drawn from different domains. Despite its effectiveness, to our best knowledge, the use of MMD in the context of neural networks has not been investigated yet. This work is therefore the first study to use MMD in neural networks. Specifically, we will investigate whether the MMD regularization can indeed improve the discriminative domain adaptation performance of neural networks.

2   Preliminaries

In this section, we will describe several tools related to our proposed method such as MMD measure, feed forward neural network, and denoising auto-encoder. Some reviews about such tools in recent literature will be also included.

2.1   Maximum Mean Discrepancy

The Maximum Mean Discrepancy (MMD) is a measure of the difference between two probability distributions from their samples. It is an effective criterion that

compares distributions without initially estimating their density functions. Given two probability distributions p and q on $\mathcal{X}$, MMD is defined as

$$\mathcal{MMD}(\mathfrak{F}, p, q) = \sup_{f \in \mathfrak{F}} (\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)]), \tag{1}$$

where $\mathfrak{F}$ is a class of functions $f : \mathcal{X} \to \mathbb{R}$. By defining $\mathfrak{F}$ as the set of functions of the unit ball in a universal Reproducing Kernel Hilbert Space (RKHS), denoted by $\mathcal{H}$, it was shown that $\mathcal{MMD}(\mathfrak{F}, p, q) = 0$ will detect any discrepancy between p and q (Borgwardt et al., 2006).

Let $\{\mathbf{x}_s^{(i)}\}_{i=1,\dots,n_s}$ and $\{\mathbf{x}_t^{(j)}\}_{j=1,\dots,n_t}$ be data vectors drawn from distributions $\mathcal{D}_s$ and $\mathcal{D}_t$ on the data space $\mathcal{X}$, respectively. Based on the fact that f is in the unit ball in a universal RKHS, one may rewrite the empirical estimate of MMD as

$$\mathcal{MMD}_{\mathrm{e}}(\mathbf{x}_s, \mathbf{x}_t) = \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \phi(\mathbf{x}_s^{(i)}) - \frac{1}{n_t} \sum_{j=1}^{n_t} \phi(\mathbf{x}_t^{(j)}) \right\|_{\mathcal{H}}, \tag{2}$$

where $\phi(\cdot) : \mathcal{X} \to \mathcal{H}$ is referred to as the *feature space map*.

By casting (2) into a vector-matrix multiplication form, we come up with a *kernelized* equation of the form (Borgwardt et al., 2006)

$$\mathcal{MMD}_{\mathrm{e}}(\mathbf{x}_s, \mathbf{x}_t) = \left(\frac{1}{n_s^2} \sum_{i=1}^{n_s} \sum_{j=1}^{n_s} k(\mathbf{x}_s^{(i)}, \mathbf{x}_s^{(j)}) + \frac{1}{n_t^2} \sum_{i=1}^{n_t} \sum_{j=1}^{n_t} k(\mathbf{x}_t^{(i)}, \mathbf{x}_t^{(j)}) \right.$$

$$\left. - \frac{2}{n_s n_t} \sum_{i=1}^{n_s} \sum_{j=1}^{n_s} k(\mathbf{x}_s^{(i)}, \mathbf{x}_t^{(j)}) \right)^{\frac{1}{2}} \tag{3}$$

$$= \left(\frac{\mathrm{Tr}\,(\mathbf{K}_{xss})}{n_s^2} + \frac{\mathrm{Tr}\,(\mathbf{K}_{xtt})}{n_t^2} - 2 \frac{\mathrm{Tr}\,(\mathbf{K}_{xst})}{n_s n_t} \right)^{\frac{1}{2}}, \tag{4}$$

where $[\mathbf{K}_{x\bullet\bullet}]_{ij} = k(\mathbf{x}_\bullet^{(i)}, \mathbf{x}_\bullet^{(j)})$ is the gram-matrix of all possible kernels in the data space.

In domain adaptation or transfer learning, MMD has been used to reduce the distribution mismatch between the source and target domain. Pan et al. (2009) proposed a PCA-based model referred to as Transfer Component Analysis (TCA) that used MMD to induce a subspace where the data distributions in different domains are closed to each other. Long et al. (2013) presented a Transfer Sparse Coding (TSC) that utilizes MMD in the encoding stage to match the distributions of the sparse codes.

Our work here adopts an idea of incorporating MMD into the learning algorithm similarly to TCA and TSC. The difference is that we carry out the MMD regularization with respect to the supervised criterion while both TCA and TSC are unsupervised learning. We expect that the MMD regularization embedded in the supervised training will induce better discriminative features.

2.2 Feed Forward Neural Networks

The Feed Forward Neural Network (FFNN) has been used extensively for solving many discriminative tasks during the past decades, including object recognition tasks. The standard FFNN structure consists of three types of layer that are the input, hidden, and output layers with weighted *inter-layer* connections. The FFNN training corresponds to adjusting the connection weights with respect to a specific criterion.

Let us consider a single hidden layer neural network with $\mathbf{x} \in \mathbb{R}^{n_x}$, $\mathbf{h} \in \mathbb{R}^{n_h}$, and $\mathbf{o} \in \mathbf{R}^{n_o}$ as the visible, hidden, and output layers, respectively. We denote $\mathbf{W}_1 \in \mathbb{R}^{n_x \times n_h}$ and $\mathbf{W}_2 \in \mathbb{R}^{n_h \times n_o}$ as the connection weights between the adjacent layers. The FFNN can be written in the form of

$$\mathbf{h} = \sigma_1(\mathbf{W}_1^\top \mathbf{x} + \mathbf{b}), \tag{5}$$

$$\mathbf{o} = \sigma_2(\mathbf{W}_2^\top \mathbf{h} + \mathbf{c}), \tag{6}$$

where $\mathbf{b} \in \mathbb{R}^{n_h}$ and $\mathbf{c} \in \mathbb{R}^{n_o}$ are the hidden and output units' biases, respectively.

Note that both $\sigma_1 : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ and $\sigma_2 : \mathbb{R}^{n_o} \to \mathbb{R}^{n_o}$ are the non-linear activation functions. In this work, we use the rectifier function approximated by the softplus function $\sigma_1(\mathbf{u})_j = \log(1 + \exp(\mathbf{u}_j))$ and the softmax function $\sigma_2(\mathbf{v})_l = \frac{\exp(\mathbf{v}_l)}{\sum_k \exp(\mathbf{v}_k)}$, where $\mathbf{u} \in \mathbb{R}^{n_h}$ and $\mathbf{v} \in \mathbb{R}^{n_o}$. The rectifier function $\sigma_1(\cdot)$ has been argued to be more biologically plausible than the logistic function (Glorot et al., 2011). More importantly, several experimental works proved that the rectifier activation function can improve the performance of neural network models (Nair and Hinton, 2010). Furthermore, the use of the softmax function induces a probabilistic interpretation of the FFNN output.

Given the n labeled training data $\{\mathbf{x}^{i)}, \mathbf{y}^{(i)}\}_{i=1,\ldots,n}$, where $\mathbf{y} \in \{0, 1\}^{n_o}$ represents the label with one active output node per class, the objective function of FFNN in the form of the empirical log-likelihood loss function is given as

$$J_{\mathrm{NN}} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{l} \mathbf{y}_k^{(i)} \log\left([g(\mathbf{x}^{(i)})]_k\right) \tag{7}$$

which is typically minimized by the back-propagation algorithm.

2.3 Denoising Auto-encoder

An auto-encoder refers to an unsupervised neural network used for learning efficient codings. In deep learning research, it is known as an effective technique for pretraining deep neural networks (Bengio et al., 2007). In terms of the structure, the auto-encoder is very similar to the standard feed-forward neural network except that its output layer has an equal number of nodes as the input layer. The objective of the auto-encoder is to reconstruct its own inputs by means of a reconstruction loss function.

A denoising auto-encoder (DAE) is a variant of the auto-encoder model that captures robust representations by reconstructing clean inputs given their noisy counterparts (Vincent et al., 2010). Qualitatively, the use of several types of

noise such as zero masking, Gaussian, and salt-and-pepper noises characterizes particular "filters" that correspond to the first hidden layer parameters (Vincent et al., 2010). DAEs have been considered better than standard auto-encoders and comparable to restricted Boltzmann machines in the context of deep learning discriminative performance (Erhan et al., 2010, Vincent et al., 2010).

In this work, we consider DAE as the pretraining stage of our proposed domain adaptive model. Unlabeled images from both source and target domains are considered as inputs to the DAE pretraining. We will investigate the effect with and without the DAE pretraining regarding to the domain adaptation performance.

3   Domain Adaptive Neural Networks

We propose a variant of the standard feed forward neural network that we refer to as the *Domain Adaptive Neural Network* (DaNN). This model incorporates MMD measure (2) as a regularization embedded in the supervised back-propagation training. By using such a regularization, we aim to train the network parameters such that the supervised criterion is optimized and the hidden layer representations are encouraged to be invariant across different domains.

Given the labeled *source* data $\{\mathbf{x}_s^{(i)}, \mathbf{y}_s^{(i)}\}_{i=1,\dots,n_s}$ and the unlabeled *target* data $\{\mathbf{x}_t^{(j)}\}_{j=1,\dots,n_t}$, the loss function of a single layer DaNN is given by

$$J_{\mathrm{DaNN}} = J_{\mathrm{NNs}} + \gamma \mathcal{MMD}_e^2(\mathbf{q}_s, \bar{\mathbf{q}}_t), \tag{8}$$

where $J_{\mathrm{NNs}} = -\frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{k=1}^{l} ([\mathbf{y}_s^{(i)}]_k \log([f(\mathbf{x}_s^{(i)})]_k))$ is the same loss function as shown in (7) but applied only over the source data, $\mathbf{q}_s = \mathbf{W}_1^\top \mathbf{x}_s + \mathbf{b}$, $\bar{\mathbf{q}}_t = \mathbf{W}_1^\top \mathbf{x}_t + \mathbf{b}$ are the linear combination outputs before the activation, and γ is the regularization constant controlling the importance of MMD contribution to the loss function.

To minimize (8), we need the gradient of J_{DaNN}. While computing the gradient of J_{NNs} over $\{\mathbf{W}_1, \mathbf{b}, \mathbf{c}\}$ is trivial, computing the gradient of $\mathcal{MMD}_e^2(\mathbf{q}_s, \bar{\mathbf{q}}_t)$ depends on the choice of the kernel function. We choose the Gaussian kernel, which is considered as a universal kernel (Steinwart, 2002), as the kernel function of the form $k_G(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2s^2}\right)$, where s is the standard deviation.

We can rewrite the $\mathcal{MMD}_e^2(\cdot, \cdot)$ function (8) in terms of the Gaussian kernel by a matrix-vector form. Let us denote the sample vectors $\tilde{\mathbf{x}}_s^{(\mathbf{i})} = \begin{bmatrix} 1 \\ \mathbf{x}_s^{(i)} \end{bmatrix} \in \mathbb{R}^{(d+1)}, \forall i = 1, \dots, n_s$ and $\tilde{\mathbf{x}}_t^{(\mathbf{j})} = \begin{bmatrix} 1 \\ \mathbf{x}_t^{(j)} \end{bmatrix} \in \mathbb{R}^{(d+1)}, \forall j = 1, \dots, n_t$. The additional element of 1 in each sample is utilized to incorporate the computation with the biases. Let us define the parameter matrices $\mathbf{U}_1 = \begin{bmatrix} \mathbf{b}^\top \\ \mathbf{W}_1 \end{bmatrix} \in \mathbb{R}^{(d+1)\times k}$ and $\mathbf{U}_2 = \begin{bmatrix} \mathbf{c}^\top \\ \mathbf{W}_2 \end{bmatrix} \in \mathbb{R}^{(k+1)\times l}$. Hence, the $\mathcal{MMD}_e^2(\cdot, \cdot)$ function can be rewritten as

$$\mathcal{MMD}_e^2(\mathbf{U}_1^\top \mathbf{X}_s, \mathbf{U}_1^\top \mathbf{X}_t) = \frac{1}{n_s^2} \sum_{i,j=1}^{n_s} \exp\left(-\frac{(\mathbf{x}_s^{(i)} - \mathbf{x}_s^{(j)})^\top \mathbf{U}_1 \mathbf{U}_1^\top (\mathbf{x}_s^{(i)} - \mathbf{x}_s^{(i)})}{2s^2}\right)$$

$$+ \frac{1}{n_t^2} \sum_{i,j=1}^{n_t} \exp\left(-\frac{(\mathbf{x}_t^{(i)} - \mathbf{x}_t^{(j)})^\top \mathbf{U}_1 \mathbf{U}_1^\top (\mathbf{x}_t^{(i)} - \mathbf{x}_t^{(i)})}{2s^2}\right)$$

$$- \frac{2}{n_s n_t} \sum_{i,j=1}^{n_s, n_t} \exp\left(-\frac{(\mathbf{x}_s^{(i)} - \mathbf{x}_t^{(j)})^\top \mathbf{U}_1 \mathbf{U}_1^\top (\mathbf{x}_s^{(i)} - \mathbf{x}_t^{(i)})}{2s^2}\right). \quad (9)$$

Let $G_{\bullet\bullet}(i,j)$ be the gradient of $k_G(\mathbf{U}_1^\top \mathbf{x}_\bullet^{(i)}, \mathbf{U}_1^\top \mathbf{x}_\bullet^{(j)})$, where the symbol $\bullet$ can be either s or t, with respect to $\mathbf{U}_1$. Then, $G_{\bullet\bullet}(i,j)$ takes the form

$$G_{\bullet\bullet}(i,j) = -\frac{1}{s^2} k_G(\mathbf{x}_\bullet^{(i)}, \mathbf{x}_\bullet^{(j)})(\mathbf{x}_\bullet^{(i)} - \mathbf{x}_\bullet^{(j)})(\mathbf{x}_\bullet^{(i)} - \mathbf{x}_\bullet^{(j)})^\top \mathbf{U}_1. \quad (10)$$

Now it is straightforward to see that the gradient of $\mathcal{MMD}_e^2(\mathbf{U}_1^\top \mathbf{X}_s, \mathbf{U}_1^\top \mathbf{X}_t)$ w.r.t $\mathbf{U}_1$ ($\frac{\partial \mathcal{M}_{st}^2}{\partial \mathbf{U}_1}$ for short) is given by

$$\frac{\partial \mathcal{M}_{st}^2}{\partial \mathbf{U}_1} = \frac{1}{n_s^2} \sum_{i,j=1}^{n_s} G_{ss}(i,j) + \frac{1}{n_t^2} \sum_{i,j=1}^{n_t} G_{tt}(i,j) - \sum_{i,j=1}^{n_s, n_t} \frac{2}{n_s n_t} G_{st}(i,j). \quad (11)$$

The main reason for choosing the Gaussian kernel is that it has been well studied and proven to make MMD useful in practice (Gretton et al., 2012). Furthermore, it is worth noting that MMD here is applied to linear combination outputs before we put on the non-linear activation function. This means that MMD provides a biased estimate with respect to an actual distribution discrepancy of the hidden representations. However, since we use the rectifier activation function that is close to linear, we expect that the measure in (9) would be able to produce good approximation of the true distribution discrepancy.

In the implementation, we separate the minimization of J_{NNs} and $\mathcal{MMD}_e^2(\cdot, \cdot)$ into two steps. Firstly, J_{NNs} is minimized using a *mini-batched* stochastic gradient descent with respect to $\mathbf{U}_1$ update. The mini-batched setting has become a standard practice in neural network training to establish a compromise between speed and accuracy. Then, $\mathcal{MMD}_e^2(\cdot, \cdot)$ is minimized by re-updating $\mathbf{U}_1$ with respect to the gradient (11). The latter step is accomplished by a *full-batched* gradient descent. The detail of this procedure are summarized in Algorithm 1.

Algorithm 1: The DaNN supervised back-propagation algorithm.

Data:
$\mathbf{U}_1 \in \mathbb{R}^{(d+1) \times k}$ and $\mathbf{U}_2 \in \mathbb{R}^{(k+1) \times l}$ are the weight-bias matrices in the first and second layers, respectively.
$\mathbf{h} \in \mathbb{R}^k$ is the hidden layer vector.
$\mathbf{o} \in \mathbb{R}^l$ is the output layer vector.
α, γ are the learning rate and the MMD regularization constant.
begin

   1. Initialize $\mathbf{U}_1$ and $\mathbf{U}_2$ with small random real values;
   2. Update $\mathbf{U}_2$ and $\mathbf{U}_1$ using the batched stochastic gradient descent by the standard forward - backward pass w.r.t. J_{NNs};
   3. Update $\mathbf{U}_1$ by the offline gradient descent as follows

$$\mathbf{U}_1(t) := \mathbf{U}_1(t-1) - \alpha\gamma \frac{\partial \mathcal{M}_{st}^2}{\partial \mathbf{U}_1}$$

   4. Repeat Steps 2 and 3 until the end of the epoch;

end

---

4   Experiments and Analysis

We evaluated our proposed method in the context of object recognition over several domain mismatches. We first compared the DaNN to baselines and other recent domain adaptation methods. The results in terms of the recognition accuracy represented by the mean and standard deviation over 30 independent runs are then reported. At last, we investigated the effect of the MMD regularization by measuring the difference of the first hidden layer activations between one domain to another domain.

4.1   Setup

Our experiments used the Office data set (Saenko et al., 2010) that contains images of 31 object classes from three different domains: *amazon*, *webcam*, and *dslr*. In *amazon*, the images contain a single centered object, while for the others the images were acquired in unconstrained settings with some variations such as lighting and background changes. Here we only used 10 object classes following the protocol designed by Gong et al. (2012), which ends up with 1410 instances in total. The number of images for *amazon*, *webcam*, and *dslr*, respectively, are 958, 295, and 157. *Webcam* and *dslr* are known to be more similar to each other based on the Rank of Domain (ROD) measure (Gong et al., 2012). Examples of the Office images can be seen in Figure 1.

Fig. 1. The Office data set (Saenko et al., 2010) samples from *amazon* and *dslr* domains.

The DaNN model used in the experiments has only one hidden layer, i.e., a shallow network of 256 hidden nodes.[2] The input layer of the DaNN can be either raw pixels or SURF features. The output layer contains ten nodes corresponding to the ten classes.

In all our experiments, we used the parameter setting for the supervised back-propagation learning specified in Table 1. Note that we employed the *dropout* regularization introduced by Hinton et al. (2012), the regularization of which randomly omits a hidden node for each training case with a certain probability.

---

[2] This number was to obtain dimensionality reduction. We tried other values such as 100, 300, and 500. Eventually, the number of 256 hidden nodes gave us the best performance among other values.

It has been proven to produce better performance in the sense of reducing the overfitting if a neural network is trained from a small training set.

Table 1. The standard parameter setting of the DaNN.

Learning rate (α)	0.02
Iterations	900
Momentum	0.05
L2 weight regularization	0.003
Dropout fraction	0.5

For the MMD regularization, we set the standard deviation s of the Gaussian kernel by the following calculation: $s = \sqrt{\frac{MSD}{2}}$ (Baktashmotlagh et al., 2013), where MSD is the median squared distance between all source samples. The MMD regularization constant γ was set to be sufficiently large ($\gamma = 10^3$) to accommodate small values of (11) compared to $\frac{J_{\text{NNs}}}{\mathbf{U}_1}$ for each iteration.

We conducted six domain shift settings, each of which is a domain pair, based on three domains originated from the Office data set ($A \rightarrow W$, $W \rightarrow A$, $A \rightarrow D$, $D \rightarrow A$, $W \rightarrow D$, and $D \rightarrow W$). The evaluation was divided into two settings: 1) *unsupervised* adaptation, and 2) *semi-supervised* adaptation. The unsupervised adaptation corresponds to the setting when we can use both labeled images from the source domain and unlabeled images from the target domain during the training, but no labels from the target domain are incorporated. In the semi-supervised adaptation, we incorporate a few labeled images from the target domain as additional training images. First three images per object category from the target domain are selected. Differently from what was conducted in the initial work (Saenko et al., 2010), we used all labeled images from the source domain instead of randomly sampled from it.

The performance of our model was then compared to SVM-based baselines, two existing domain adaptation methods, and a simple neural network as follows:
L-SVM: an SVM (Cortes and Vapnik, 1995) model with a linear kernel that was applied to the original features.[3]
L-SVM + PCA: the same model as the L-SVM but preceded by PCA to reduce feature dimensionality.
GFK (Gong et al., 2012): the Geodesic Flow Kernel approach by considering an infinite number of intermediate subspaces between the source and target domains followed by k-NN classification.[4]
TSC (Long et al., 2013): the Transfer Sparse Coding technique based on the combination of the graph regularized sparse coding, the MMD regularization, and the logistic regression.[5]
NN: a single layer neural network with the same structure and parameter setting (Table 1) used in our DaNN, but without the MMD regularization.[6]

---

[3] `http://www.csie.ntu.edu.tw/~cjlin/liblinear`
[4] Here we used the subspaces constructed by PCA only
[5] `http://learn.tsinghua.edu.cn:8080/2011310560/long.html`
[6] It is basically Algorithm 1 without Step 3.

4.2 Results on SURF Features

We first investigated the performance of our model on the standard image features provided by Gong et al. (2012). Briefly, the image features were acquired by first utilizing the SURF descriptor on resized and grayscaled images to detect local scale-invariant interest points. It was then followed by encoding the data points into 800-bin histograms using a codebook trained from a subset of *amazon* images (Saenko et al., 2010). The final features were then normalized and z-scored to have zero mean and unit variance. We conducted the unsupervised setting evaluation with the results shown in Table 2.

We found that DaNN and TSC have better performance than the other approaches on these standard features. More specifically, DaNN performs well when there is the *amazon* set in a particular domain pairs. In the case of *webcam-dslr* shifts, the TSC, which has not been tested on the Office dataset in the previous work, is surprisingly the best model. Despite its effectiveness, TSC has longer feature extraction time than, for example, neural network-based approaches so that it is less efficient in real world situation. We also noted that the GFK, which incorporates multiple intermediate subspaces, fails to surpass the baselines in several cases. This indicates that the projection onto the subspaces generated by GFK is insufficient to reduce the domain mismatch.

Table 2. The unsupervised setting performances on the Office data set ($A : amazon$, $W : webcam$, $D : dslr$) for each domain pair using SURF-based features as inputs. Each column header starting from the second column indicates one domain pair, e.g., $A \rightarrow W$ represents the *amazon* and *webcam* as the training and test sets.

Methods	$A \rightarrow W$	$W \rightarrow A$	$A \rightarrow D$	$D \rightarrow A$	$W \rightarrow D$	$D \rightarrow W$
L-SVM	24.1 ± 0.0	35.8 ± 0.0	28.0 ± 0.0	32.7 ± 0.0	77.7 ± 0.0	78.0 ± 0.0
PCA + L-SVM	34.9 ± 0.0	34.8 ± 0.0	35.0 ± 0.0	32.2 ± 0.0	63.7 ± 0.0	65.4 ± 0.0
GFK Gong et al. (2012)	39.0 ± 0.0	29.8 ± 0.0	36.3 ± 0.0	31.8 ± 0.0	80.3 ± 0.0	75.6 ± 0.0
TSC Long et al. (2013)	$\mathbf{47.4 \pm 1.7}$	$\mathbf{39.1 \pm 0.4}$	46.2 ± 1.4	$\mathbf{41.6 \pm 0.8}$	$\mathbf{93.6 \pm 0.5}$	$\mathbf{93.5 \pm 0.6}$
NN	44.4 ± 0.6	37.3 ± 0.1	47.8 ± 0.9	34.8 ± 0.2	81.5 ± 0.0	78.9 ± 0.0
DaNN	$\mathbf{45.4 \pm 0.8}$	$\mathbf{38.7 \pm 0.2}$	$\mathbf{49.0 \pm 0.7}$	38.1 ± 0.3	83.4 ± 0.0	81.0 ± 0.0

4.3 Results on Raw Pixels

We also conducted the evaluation against the raw pixels of the Office images. Previous works on the Office image set were mostly done using the SURF-based features. It is worth investigating the performance on the Office raw pixels directly since good models on raw pixels are preferable in the sense of reducing the needs of handcrafted feature extractors. We first converted the pixels of the Office images in 2D RGB values into grayscaled pixels and resized them into a dimension of 28×28. They were then z-scored to have zero mean and unit variance.

Domain Adaptation Setting

In this experiment, we ran both the *unsupervised* and *semi-supervised* adaptation setting for all domain pairs. In addition, we also investigated the effect of DAE pretraining that precedes the NN and DaNN supervised training with respect to the performance. The DAE pretraining will slightly change Step 1 of Algorithm 1. We denoted these models as **DAE + NN** and **DAE + DaNN**. Examples of the pretrained weights are depicted in Figure 2. The complete accuracy rates on the Office raw pixels for all domain pairs are presented in Table 3.
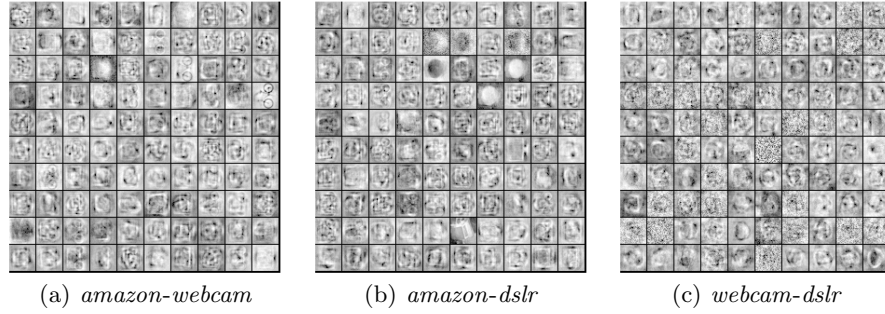

(a) *amazon-webcam*   (b) *amazon-dslr*   (c) *webcam-dslr*

Fig. 2. The 2D visualization of 100 randomly chosen weights after the DAE pretraining for each domain pairs from the Office image set. The white, gray, and black pixels in each box indicate the high-positive, close-to-zero or zero, and high-negative values of a particular connection weight. The zero-masking noise is used with 30% destruction.

Table 3. The performances on the Office dataset (A : *amazon*, W : *webcam*, D : *dslr*) using the raw pixels as inputs.

Methods	$A \to W$	$W \to A$	$A \to D$	$D \to A$	$W \to D$	$D \to W$
Unsupervised Setting						
L-SVM	14.9 ± 0.0	14.7 ± 0.0	19.1 ± 0.0	13.7 ± 0.0	36.0 ± 0.0	40.3 ± 0.0
PCA + L-SVM	20.3 ± 0.0	18.1 ± 0.0	16.9 ± 0.0	17.4 ± 0.0	40.4 ± 0.0	37.0 ± 0.0
GFK (Gong et al., 2012)	21.4 ± 0.0	15.0 ± 0.0	30.2 ± 0.0	13.8 ± 0.0	69.1 ± 0.0	65.0 ± 0.0
TSC (Long et al., 2013)	22.3 ± 1.0	15.7 ± 1.1	25.6 ± 1.6	19.6 ± 0.7	$\mathbf{74.1 \pm 1.9}$	67.5 ± 1.5
NN	29.2 ± 0.6	17.0 ± 0.3	32.5 ± 0.7	15.0 ± 0.3	63.7 ± 0.0	57.3 ± 0.0
DAE + NN	32.5 ± 0.2	18.7 ± 0.0	37.8 ± 0.2	17.4 ± 0.0	72.1 ± 0.0	65.9 ± 0.0
DaNN	34.1 ± 0.3	21.2 ± 0.2	34.0 ± 0.8	20.1 ± 0.5	64.4 ± 0.0	62.0 ± 0.0
DAE + DaNN	$\mathbf{35.0 \pm 0.2}$	$\mathbf{23.1 \pm 0.0}$	$\mathbf{39.4 \pm 0.3}$	$\mathbf{22.5 \pm 0.0}$	$\mathbf{74.3 \pm 0.0}$	$\mathbf{70.5 \pm 0.0}$
Semi-supervised Setting						
L-SVM	18.9 ± 0.0	29.0 ± 0.0	25.2 ± 0.0	35.2 ± 0.0	45.7 ± 0.0	52.5 ± 0.0
PCA + L-SVM	20.8 ± 0.0	31.0 ± 0.0	25.6 ± 0.0	35.1 ± 0.0	50.4 ± 0.0	50.2 ± 0.0
GFK (Gong et al., 2012)	47.9 ± 0.0	33.1 ± 0.0	52.0 ± 0.0	31.8 ± 0.0	80.3 ± 0.0	$\mathbf{74.7 \pm 0.0}$
TSC (Long et al., 2013)	42.4 ± 2.1	34.1 ± 0.8	49.3 ± 2.2	36.4 ± 0.9	76.3 ± 1.4	71.1 ± 1.1
NN	48.7 ± 0.3	34.5 ± 0.3	52.8 ± 0.6	36.2 ± 0.4	75.6 ± 0.1	67.2 ± 0.0
DAE + NN	52.8 ± 0.1	36.8 ± 0.0	57.5 ± 0.1	36.5 ± 0.0	$\mathbf{83.5 \pm 0.0}$	69.4 ± 0.0
DaNN	51.3 ± 0.5	36.6 ± 0.4	55.9 ± 0.3	$\mathbf{37.9 \pm 0.3}$	78.0 ± 0.2	70.2 ± 0.0
DAE + DaNN	$\mathbf{53.6 \pm 0.2}$	$\mathbf{37.3 \pm 0.0}$	$\mathbf{59.9 \pm 0.1}$	$\mathbf{38.2 \pm 0.0}$	$\mathbf{83.5 \pm 0.0}$	71.2 ± 0.0

It is clear that our DaNN always provides accuracy improvements in all domain pairs compared to the SVM-based baselines and the NN model. In other words, the MMD regularization indeed improves the performance of neural networks. Compared to TSC that also employs the MMD regularization in the unsupervised training stage, our DaNN performs better in most cases. However, TSC can match the DaNN performance on *webcam-dslr* couples, which has lower level mismatch than the other couples. This indicates that the utilization of the MMD regularization in the supervised training might gain more adaptation ability than that in the unsupervised training for pairs with more difficult mismatches to solve.

The DAE pretraining applied to NN and DaNN indeed improves the performances for all couples of domains. The improvements are quite significant for several cases, especially for *webcam-dslr* couples. In general, the DAE pretraining also produces more stable models in the sense of resulting in lower standard deviations over 30 independent runs. Furthermore, the combination of DAE pretraining and DaNN performs best among other methods in these experiments in almost all cases. In the sense of qualitative analysis, as can be seen in Figure 2, the DAE pretraining captures more distinctive "filters" from local blob detectors to object parts detectors, especially when the *amazon* images are included. This effect is somewhat consistent with what was found in the initial DAE work (Vincent et al., 2010) suggesting that the DAE pretraining provides more useful neural network representations.

In the semi-supervised setting, the performance trend is somewhat similar to the unsupervised setting. However, the performance discrepancies between NN and DaNN here becomes smaller than those in the unsupervised setting. This outcome also holds for the case of the DAE pretraining. This suggests that both the MMD regularization and DAE pretraining might be less impactful when some labeled images from the target domain can be acquired.

In-domain Setting

One may ask whether the domain adaptation results shown in Table 3 are reasonable compared to the standard learning setting. We refer this standard setting to as the *in-domain* setting, where the training and test samples come from the same domain. The *in-domain* performance can be considered as a reference that indicates the effectiveness of domain adaptation models in dealing with the domain mismatch.

We investigated the *in-domain* performances of non-domain adaptive models described in Section 4.1, i.e., L-SVM, PCA+L-SVM, and NN on raw pixels of the Office images. For each domain, we conducted 10-fold cross-validation. The complete *in-domain* results in terms of the mean and standard deviation are shown in Table 4. In general, we can see that the best *in-domain* model is the NN model on both training and test images.

In comparison to the domain adaptation results, the highest *in-domain* accuracies are better than the results with domain mismatches when the *amazon* or *webcam* are used as the target sets (see the highest accuracy rates in column

Table 4. The *in-domain* performances on the Office data set using 10-fold cross-validation on each domain.

Methods	amazon		webcam		dslr	
	Training	Test	Training	Test	Training	Test
L-SVM	99.0 ± 0.3	52.0 ± 4.6	100.0 ± 0.0	57.7 ± 13.9	100.0 ± 0.0	51.0 ± 14.1
PCA+L-SVM	64.4 ± 0.8	60.6 ± 6.4	72.0 ± 1.5	62.8 ± 8.7	75.6 ± 2.1	55.2 ± 13.1
NN	99.3 ± 0.1	$\mathbf{74.2 \pm 3.2}$	100.0 ± 0.0	$\mathbf{87.2 \pm 5.4}$	100.0 ± 0.0	$\mathbf{77.9 \pm 8.8}$

$D \rightarrow A$ and $D \rightarrow W$ on Table 3). This indicates that a better domain adaptation model might be necessary to overcome those mismatches. However, this is not the case for the *dslr* as the target set where the *in-domain* accuracy is even lower than the best domain adaptation result on $W \rightarrow D$ pair. Knowing the facts that the *webcam* and *dslr* images are quite similar and the *webcam* set has more images, this shows that the domain adaptation indeed helps to produce a better object recognition model for this kind of setting.

5 Conclusions and Future Work

This paper aimed to reduce the domain mismatch problem in object recognition using a simple neural network model, which we refer to as the Domain Adaptive Neural Network (DaNN). In this work, we utilized the MMD measure as a regularization in the supervised back-propagation training. This regularization encouraged the hidden layer representation distributions to be similar to each other. We demonstrated that the DaNN performs well on the Office image set, especially on raw image pixels as inputs. Furthermore, the DaNN preceded by the denoising auto-encoder (DAE) pretraining has better performance compared to SVM-based baselines, GFK (Gong et al., 2012), and TSC (Long et al., 2013) on the Office image set (Saenko et al., 2010) in almost all domain pairs.

Despite the effectiveness of the MMD regularization, there are still many aspects that can be further improved. We have seen that the performance on raw pixels, which is a main concern in representation learning approach, is still not as good as that on SURF features. We note that good models that perform well without any preceding handcrafted feature extractors are preferable to reduce complexity. A better model on raw pixels might be achieved by using deeper neural network layers with a similar strategy since deep architectures have brought some successes in many applications in recent years (Bengio, 2013). Our initial work using a standard deep neural network with the DAE pretraining, which is not shown here due to page limit, suggested that deeper representations do not always improve the performance against the domain mismatch.

In addition, a study on the kernel choice for computing MMD regarding to the domain adaptation problem might be worth addressing. We assumed that the universal Gaussian kernel function can detect any underlying distribution mismatches in the Office data set, which might be not true. A better understanding about the relationship between a kernel function and a particular image mismatch, e.g., background, lighting, affine transformation changes, would induce a great impact in this field of research.

Bibliography

M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of International Conference on Computer Vision*, pages 769–776, 2013.

H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

Y. Bengio. Deep learning of representations: Looking forward. In *Statistical Language and Speech Processing*, volume 7978 of *Lecture Notes in Computer Science*, pages 1–37. Springer, 2013.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 19, page 153, 2007.

Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Computing Research Repository*, abs/1206.5538, 2012.

K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

C. Cortes and V. N. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3): 273–297, 1995.

H. Daumé-III. Frustratingly easy domain adaptation. *CoRR*, abs/0907.1815, 2009.

D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, and P. Vincent. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural network. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 315–323, 2011.

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

R. Gopalan, R. Li, and R. Chellapa. Domain adaptation for object recognition: An unsupervised approach. In *IEEE International Conference on Computer Vision*, pages 999–1006, 2011.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Sch'olkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, pages 723–773, 2012.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

M. Long, G. Ding, J. Wang, J. Sun, Y. Guo, and P. S. Yu. Transfer sparse coding for robust image representation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 404–414, 2013.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1187–1192, 2009.

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

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2002.

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.