Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Layer-wise domain correction for unsupervised domain adaptation*#

Shuang LI, Shi-ji SONG‡, Cheng WU

Automation Department, Tsinghua University, Beijing 100084, China

E-mail: l-s12@mails.tsinghua.edu.cn; shijis@mail.tsinghua.edu.cn; wuc@tsinghua.edu.cn

Received Nov. 19, 2017; Revision accepted Jan. 18, 2018; Crosschecked Jan. 19, 2018

Abstract: Deep neural networks have been successfully applied to numerous machine learning tasks because of their impressive feature abstraction capabilities. However, conventional deep networks assume that the training and test data are sampled from the same distribution, and this assumption is often violated in real-world scenarios. To address the domain shift or data bias problems, we introduce layer-wise domain correction (LDC), a new unsupervised domain adaptation algorithm which adapts an existing deep network through additive correction layers spaced throughout the network. Through the additive layers, the representations of source and target domains can be perfectly aligned. The corrections that are trained via maximum mean discrepancy, adapt to the target domain while increasing the representational capacity of the network. LDC requires no target labels, achieves state-of-the-art performance across several adaptation benchmarks, and requires significantly less training time than existing adaptation methods.

Key words: Unsupervised domain adaptation; Maximum mean discrepancy; Residual network; Deep learning
https://doi.org/10.1631/FITEE.1700774 **CLC number:** TP183

1 Introduction

Past years have witnessed a renaissance of deep neural network architectures (He et al., 2016; Krizhevsky et al., 2017). This has led to dramatic improvements across many diverse machine learning applications (Mikolov et al., 2013; Sutskever et al., 2014; Russakovsky et al., 2015; Gehring et al., 2017). The most compelling aspects of deep networks may be that they learn a feature representation that is especially well suited for the particular prediction task entirely from the data itself. The reliance on large labeled data sets can, however, also constitute a limitation. In the absence of sufficient labeled data, learning the representation and the classifier jointly becomes a difficult problem, and the advantage of deep network architectures vanishes. It may be fair to argue that most application domains do not have the luxury of access to carefully manicured and gigantic labeled data sets, as the collection of labeled data can be a non-trivial and costly task.

Domain adaptation itself is concerned with the question of how to adapt classifiers from a source problem domain to a (similar) target domain (Pan and Yang, 2010; Chen et al., 2011; Pan et al., 2011; Gong et al., 2012). This can provide a promising solution to settings that suffer from shortage of labeled data or where the training and testing data follow substantially different distributions. Actually, domain adaptation has been successfully applied to various real-world machine learning tasks, such as sentiment analysis (Blitzer et al., 2006), text classification (Duan et al., 2012), and computer vision (Duan et al., 2009). To be specific, for example, due

‡ Corresponding author

* Project supported by the National Key R&D Program of China (No. 2016YFB1200203) and the National Natural Science Foundation of China (Nos. 41427806 and 61273233)

Electronic supplementary materials: The online version of this article (https://doi.org/10.1631/FITEE.1700774) contains supplementary materials, which are available to authorized users

ORCID: Shi-ji SONG, http://orcid.org/0000-0001-7361-9283

to the different factors of occlusion, pose, or illumination, for the object recognition problem, the performance of a well trained source model may degrade tremendously on the test images (Long et al., 2016a). Thus, transfer learning methods could effectively extract the domain invariant features for both domains to improve the final classification accuracy. As for another example, to avoid collecting plentiful labeled data in the semantic segmentation problem, we would like to train the source model leveraging a large amount of synthetic data as is generated from graphics game engines. Pixel-level domain adaptation methods will align both domain data not only in the feature space but also in the raw pixel space to maintain semantic consistency (Hoffman et al., 2017).

If a similar data domain exists where labels are readily available in ample quantities, it may be possible to adapt deep networks from this source domain to the actual target domain of interest. The main challenge of domain adaptation for deep networks is that the task is not limited to adapting a classifier to a new data domain, but primarily its data representation. In Fig. 1, if a deep net is trained on a source domain and applied to a target domain, the many consecutive layers of non-linear transformations within the neural network can amplify the differences between domains (Yosinski et al., 2014; Long et al., 2015). Thus, reducing the substantial distribution discrepancy is of vital importance.

For example, Tzeng et al. (2014) introduced a deep domain confusion (DDC) architecture to learn domain-invariant features for the source and target in the last hidden layer. Not restricted to adapt only a single layer, Long et al. (2015) proposed a deep adaptation network (DAN) to learn transferable features across two domains after multiple layer adaptation. Different from DDC and DAN, a novel adversarial domain adaptation network was proposed by Ganin and Lempitsky (2015) to learn both discriminative and invariant features during a backpropagation based training procedure.

Although these methods can achieve promising results, they all suffer from two limitations. First, when confronting a new target domain, these methods must retrain the deep network from scratch, and cannot reuse the well trained network directly. The repeated training process leads to a waste of time and resources. Second, the classification capacity of these architectures is restricted by the selected base networks. This means that the adaptation process cannot improve the generalization ability of the infrastructural networks. The design philosophy will limit the performance of these famous methods.

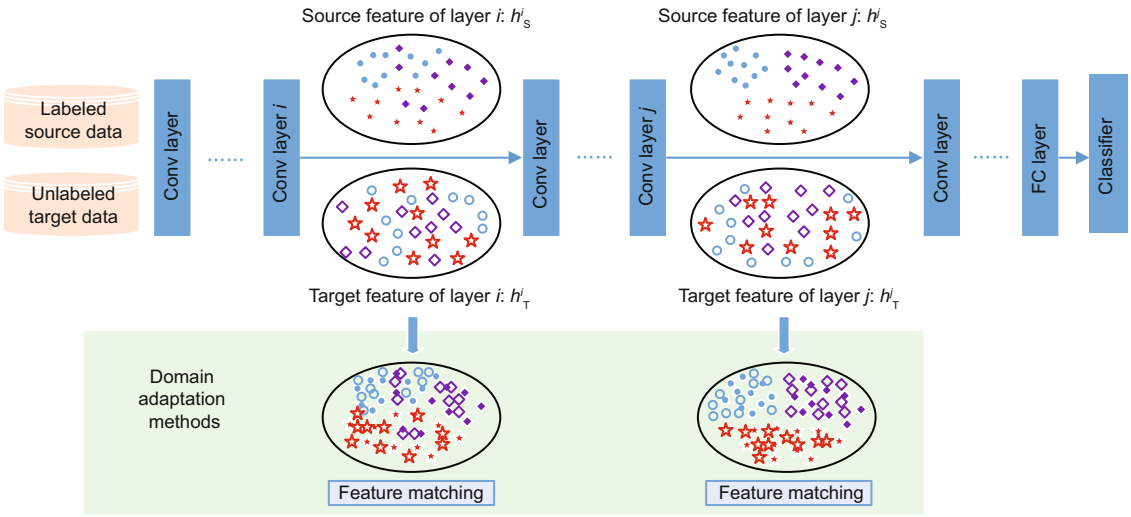

Fig. 1 Illustrating the aims of most deep domain adaptation methods

For the traditional methods, we can use labeled source data to train a perfect deep neural network. However, since the source and target data are under different distributions, for each layer, the representations of both domains, i.e., h^i_S and h^i_T, h^j_S and h^j_T, are divergent, which result in the dramatic degradation of classification performance on target data. Most deep domain adaptation methods (including LDC) aim to mitigate the domain discrepancy by different kinds of feature matching approaches, which can bridge both domains, and benefit the final target classification

To address these two limitations and derive perfect domain-invariant representations, in this study, we propose a layer-wise domain correction (LDC) approach, which provides an efficient and effective way to adapt a deep network from a source to a target domain without requiring any labeled target data. LDC corrects the internal representation of layers in the network to compensate for any data shift as samples propagate through the architecture. The correcting layers are additive, in the spirit of residual networks (He et al., 2016). During the adaptation process we keep the entire network fixed and update only the additional corrections. This is computationally very efficient and results in very fast adaptation. Moreover, the well trained source network can be reused for any other newcome target domain, which can dramatically speed up the adaptation process.

We guide the training of the correction layers with maximum mean discrepancy (MMD) (Gretton et al., 2012), which encourages the empirical data distribution of source and target data to match within the hidden layers. We do not require any labeled target data, as we are correcting only the learned representation. On the one hand, the additive layers (residual blocks) can effectively reduce the substantial domain discrepancy. On the other hand, the additive layers can also deepen the based architectures, and improve the representation capacity of the base network.

We demonstrate empirically, on several benchmark data sets, that our method successfully adapts neural networks from source to target domains across a variety of applications. LDC not only outperforms prior work in domain adaptation on almost all data sets, but is substantially faster.

2 Related work and background

Layer-wise domain correction (LDC) builds upon three active areas of research: (1) domain adaptation, (2) residual networks, and (3) maximum mean discrepancy (MMD). Concerning the first two categories, we will discuss the prior work most related to our proposed method, and then review MMD in more detail.

2.1 Deep learning in domain adaptation

Domain adaptation aims to derive an adaptive classifier or predictor under the condition that the training and test data are sampled from different distributions (Pan and Yang, 2010). To transfer knowledge from the source to the target, learning the domain-invariant representation is of vital importance, and a fruitful line of related work has focused on it: geodesic flow kernel (GFK) (Gong et al., 2012), transfer component analysis (TCA) (Pan et al., 2011), joint distribution adaptation (JDA) (Long et al., 2013), and so on (Gong et al., 2013; Long et al., 2014). However, these methods learn mainly shallow features for both domains, and recent studies have demonstrated that deep networks can learn more distinguished and transferable features (Glorot et al., 2011; Chen et al., 2012; Donahue et al., 2014; Yosinski et al., 2014).

When labeled data exists from the target domain, 'fine-tuning' a pretrained neural network is a simple yet effective method for leveraging learned features from a source domain (Donahue et al., 2014; Oquab et al., 2014; Yosinski et al., 2014). Our proposed method does not require target labels and is therefore much more applicable.

If target labels are not available to guide the adaptation process, a common approach is learning hidden feature representations which are invariant across domains. Deep domain confusion (DDC), a method proposed by Tzeng et al. (2014), uses labeled source data and unlabeled target data during training to reduce the MMD between domains in the final hidden layer. Long et al. (2015) generalized this method, matching the source and target representations across multiple layers in the network. The features of these layers are mapped into a reproducing kernel Hilbert space (RKHS) and matched using a multi-kernel MMD objective function. Similar to Long et al. (2015), we use MMD for adaptation on multiple layers for LDC; however, we insert new layers into a source-trained network rather than modify existing layers. In contrast to these methods, our adaptation is much faster as we update only the correction layers. Ganin and Lempitsky (2015) and Ajakan et al. (2014) proposed a different approach to learning domain invariance, using a gradient reversal trick rather than a statistical test. The reverse gradient approach (RevGrad) attaches a domain discriminator to the feature extractor of a classification network, flipping the gradient of the discriminator during backpropogation to increase the similarity between source and target features. This approach is

designed to be trained end to end, learning the classifier and domain invariance simultaneously. Conversely, LDC updates an existing network trained on source data, expanding the network with corrections trained for layer-wise adaptation.

2.2 Deep residual networks

He et al. (2016) modified the structure of deep network layers to compute additive residual functions rather than traditional multiplicative transformations. The authors hypothesized that ResNet layers can easily learn functions which approximate the identity or slight perturbations thereof, as it is far easier to learn a small residual than a direct transformation. While the original motivation of a residual network (ResNet) architecture is increasing the depth of supervised models, the architecture is well suited to layer-wise domain correction. We use residual connections to model the discrepancy between source and target distributions, as we assume that the layer-wise activations of the two domains differ by a small perturbation.

Long et al. (2016b) explored residual functions for domain adaptation. Their approach is designed to modify the final layer of an existing adaptation algorithm, while LDC performs adaptation exclusively through residual layers. By doing so, LDC can increase the depth of the network, while simultaneously locking down source layers for increased efficiency.

2.3 Maximum mean discrepancy

The maximum mean discrepancy (MMD) (Borgwardt et al., 2006; Gretton et al., 2012) statistical test quantitatively measures the similarity of two probability distributions: source p and target q. MMD produces a function f from a class of functions $\mathcal{F}$ which is different for the source and the target, and measures the average distance between domains when transformed by

$$\mathrm{MMD}^2(\mathcal{F},p,q)=\sup_{f\in\mathcal{F}}\|\mathbb{E}_{\boldsymbol{a}\sim p}[f(\boldsymbol{a})]-\mathbb{E}_{\boldsymbol{b}\sim q}[f(\boldsymbol{b})]\|^2. \tag{1}$$

Intuitively, MMD is small in magnitude when it is difficult to differentiate the two domains. Additionally, it has been shown that two distributions p and q are equal if and only if $\mathrm{MMD}^2(\mathcal{F},p,q)=0$ and $\mathcal{F}$ is universal (Gretton et al., 2012). When $\mathcal{F}$ is an RKHS with kernel function k, there is a closed-form solution to Eq. (1):

$$\begin{aligned}\mathrm{MMD}^2(\mathcal{F},p,q)=&\,\mathbb{E}_{\boldsymbol{a},\boldsymbol{a}'\sim p}[k(\boldsymbol{a},\boldsymbol{a}')]\\&-2\mathbb{E}_{\boldsymbol{a}\sim p,\boldsymbol{b}\sim q}[k(\boldsymbol{a},\boldsymbol{b})]+\mathbb{E}_{\boldsymbol{b},\boldsymbol{b}'\sim q}[k(\boldsymbol{b},\boldsymbol{b}')].\end{aligned} \tag{2}$$

Eq. (2) can be empirically estimated from a finite set of source and target samples (Gretton et al., 2012). In this equation, $\boldsymbol{a},\boldsymbol{a}'$ and $\boldsymbol{b},\boldsymbol{b}'$ are samples from the source and target distributions, respectively.

In this study, we calculate MMD in a more efficient way. To be specific, given a minibatch containing n source samples $(\boldsymbol{a}_1,\boldsymbol{a}_2,\dots\boldsymbol{a}_n)\sim p$ and n target samples $(\boldsymbol{b}_1,\boldsymbol{b}_2,\dots\boldsymbol{b}_n)\sim q$, MMD can be empirically estimated:

$$\begin{aligned}\mathrm{MMD}^2(\mathcal{F},p,q)\approx\frac{1}{n^2}\Bigg[&\sum_{i,j=1}^{n}k(\boldsymbol{a}_i,\boldsymbol{a}_j)\\&-2\sum_{i=1}^{n}\sum_{j=1}^{n}k(\boldsymbol{a}_i,\boldsymbol{b}_j)+\sum_{i,j=1}^{n}k(\boldsymbol{b}_i,\boldsymbol{b}_j)\Bigg].\end{aligned} \tag{3}$$

However, Eq. (3) is not ideal for an objective function since the estimate incurs an $O(n^2)$ cost. Therefore, during training we calculate MMD using linear-time unbiased approximation:

$$\begin{aligned}\mathrm{MMD}^2(\mathcal{F},p,q)\approx\frac{2}{n}\sum_{i=1}^{n/2}(&k(\boldsymbol{a}_{2i-1},\boldsymbol{a}_{2i})\\&+k(\boldsymbol{b}_{2i-1},\boldsymbol{b}_{2i})-k(\boldsymbol{a}_{2i-1},\boldsymbol{b}_{2i-1})-k(\boldsymbol{a}_{2i},\boldsymbol{b}_{2i})).\end{aligned} \tag{4}$$

To calculate Eq. (4) during training, a minibatch sample can be divided into quad-tuples, each containing two source samples and two target samples. Since Eq. (4) incurs an $O(n)$ cost, this is ideal for an objective function.

MMD is used in a number of deep learning applications, ranging from generative models (Li et al., 2015) to image transformation (Gardner et al., 2015) to domain adaptation (Tzeng et al., 2014; Long et al., 2015). We train our correction layers with an MMD-based loss function based on the success of Tzeng et al. (2014) and Long et al. (2015).

3 Layer-wise domain correction

3.1 Notations and motivation

Consider a classification task with input space $\mathcal{X}$ and label space $\mathcal{Y}$. Let $P_{\mathrm{S}}(\boldsymbol{x},y)$ and $P_{\mathrm{T}}(\boldsymbol{x},y)$

be distributions over the space $\mathcal{X} \times \mathcal{Y}$. We refer to these spaces as the 'source domain' and 'target domain', respectively. We formalize the problem of domain adaptation as a discrepancy between the two distributions:

$$P_{\mathrm{S}}(\boldsymbol{x}, y) \neq P_{\mathrm{T}}(\boldsymbol{x}, y). \tag{5}$$

We are provided with n_{s} labeled source data $\mathcal{D}_{\mathrm{S}} = \{\boldsymbol{x}_{\mathrm{S}i}, y_{\mathrm{S}i}\}_{i=1}^{n_{\mathrm{s}}} \sim P_{\mathrm{S}}(\boldsymbol{x}, y)$ and a corresponding high accuracy classifier f_{S}. We are interested in obtaining a classifier f_{T} for the target domain; however, we have access to only n_{t} unlabeled target data $\mathcal{D}_{\mathrm{T}} = \{\boldsymbol{x}_{\mathrm{T}j}\}_{j=1}^{n_{\mathrm{t}}} \sim P_{\mathrm{T}}(\boldsymbol{x})$. The target domain is different enough from the source such that f_{S} incurs an unacceptably high target error rate on $\mathcal{D}_{\mathrm{T}}$.

We focus on the setting where both f_{S} and f_{T} are deep neural networks. The lack of labeled target data prevents us from training f_{T} directly. Instead, we adapt f_{S} to the target domain through additive 'correction' residual layers for target data. Our adaptation approach leverages the fact that deep neural networks learn their own internal data representation as well as a classifier. We 'correct' this internal representation for target data by adding small correction terms to the hidden representations to make them mimic source data.

Residual correction. Let $\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})$ be the representation of input $\boldsymbol{x}$ at a given layer in f_{S}. To adapt f_{S}, we must learn a hidden representation $\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})$ for target data such that

$$P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x}), y) \approx P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}), y). \tag{6}$$

We adapt the representation $\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})$ to $\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})$ with the help of an additive corrective term $\triangle\boldsymbol{h}$, i.e.,

$$\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}) = \boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x}) + \triangle\boldsymbol{h}(\boldsymbol{x}).$$

This definition is reminiscent of the residual connections in ResNets (He et al., 2016). Similar to He et al. (2016), we model $\triangle\boldsymbol{h}(\boldsymbol{x})$ as a small multi-layer neural network. Fig. 2 illustrates the concept of skip connections. The orange layers represent the small neural network computing $\triangle\boldsymbol{h}(\boldsymbol{x})$, whose output is added to the previous representation $\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})$. During adaptation, only these orange layers are trained, and everything else remains fixed. We present more details on these individual layers at the end of this section.

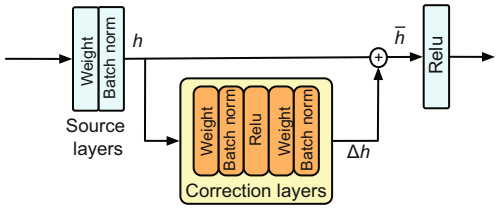

Fig. 2 Architecture of corrections added after a source network weight layer

Original source layers are blue, and correction layers are in the yellow box. References to color refer to the online version of this figure

3.2 Method and formulation

Domain correction. The learned residual function must minimize the discrepancy between $P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x}), y)$ and $P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}), y)$ to reduce the target error. Because these joint distributions are difficult to estimate from the data, we decompose them using the product rule:

$$\begin{cases} P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x}), y) = P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x}))P_{\mathrm{S}}(y|\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})), \\ P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}), y) = P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}))P_{\mathrm{T}}(y|\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})). \end{cases} \tag{7}$$

Thus, to match source and target distributions, we can simultaneously minimize the discrepancy of marginal and conditional distributions. We accomplish each of these through the following objective functions: (1) a marginal MMD measure to match $P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x}))$ and $P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}))$ and (2) a class-conditional MMD measure to match $P_{\mathrm{S}}(y|\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x}))$ and $P_{\mathrm{T}}(y|\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}))$.

Marginal MMD. The marginal distributions $P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x}))$ and $P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}))$ can be aligned by minimizing MMD, using the definition provided in the previous section:

$$\ell^{\mathrm{m}} = \mathrm{MMD}^2(\mathcal{F}, P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})), P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}))), \tag{8}$$

where $\mathcal{F}$ is the RKHS defined by the Gaussian kernel $k(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\|\boldsymbol{a} - \boldsymbol{b}\|^2/\gamma)$ with width γ. Intuitively, this term ensures that the source features and target features resemble. Similar to Long et al. (2015), we calculate the marginal distributions empirically from source and target samples in each mini-batch, and estimate Eq. (8) using a linear-time empirical approximation as Eq. (4). We set γ to the median pairwise distance of each minibatch sample (Gretton et al., 2012), which is continuously recomputed during training.

Class-conditional MDD. Actually, it is difficult to directly align $P_{\mathrm{S}}(y|\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x}))$ and $P_{\mathrm{T}}(y|\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x}))$, as

these distributions cannot be directly estimated from the data. Instead, one can align the class-conditional likelihood, which is easier to estimate using data. By Bayes' theorem, we have

$$\begin{cases} P_{\mathrm{S}}(y|\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})) \propto P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})|y)P_{\mathrm{S}}(y), \\ P_{\mathrm{T}}(y|\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})) \propto P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})|y)P_{\mathrm{T}}(y). \end{cases} \tag{9}$$

We can assume that the two class distributions are similar, i.e., $P_{\mathrm{S}}(y) \approx P_{\mathrm{T}}(y)$, and we are therefore left with minimizing the discrepancy between $P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})|c)$ and $P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})|c)$. If we knew the class c of target inputs, we could match the likelihood distributions through a class-conditional MMD loss term:

$$\ell^{\mathrm{c}} = \frac{1}{|C|}\sum_{c\in C} \mathrm{MMD}^2\big(\mathcal{F}, P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})|c), P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})|c)\big), \tag{10}$$

where C is the set of classes, and $P_{\mathrm{S}}(\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})|c)$ and $P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})|c)$ are the distributions of features for source and target samples of class c, respectively. Intuitively, Eq. (10) ensures that any learned transformation preserves class information; e.g., it does not transform a target dog into a source cat. We can ensure that each minibatch has sufficient source and target samples of each class to estimate this objective.

However, because we do not have access to the labels of the target data, we cannot estimate $P_{\mathrm{T}}(\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})|c)$ accurately. We can use the pseudo target label as a surrogate. We initially predict the target data from the source network, and then use the target network estimates after $k = 50$ training iterations. Though the estimates will be noisy, a majority of the predictions will be accurate since the unadapted network performs much better than random at the target task. These correct samples guide the first iterations of training, improving accuracy over future epochs.

Layer-wise correction. Neural networks learn multiple stages of hidden representations across their layers, which can amplify the divergence between the two data distributions. If the divergence is too large, it may become impossible to correct. Imagine, hypothetically, that data from the source domain leads to all positive activations, whereas data from the target domain to only negative ones. A ReLU transition function, $\max(h, 0)$, would set the representation of the target samples to zeros, making it impossible to recover any information about the input.

It is therefore important to keep the divergence between the source and the target relatively small throughout the network, which we achieve through corrections at multiple layers. Introducing more residual layers has the additional beneficial effect of increasing the depth of the network and boosting model capacity to absorb target data.

Regularization. Although the corrected features $\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})$ are trained in an unsupervised manner, we can incorporate additional knowledge through regularization.

The source data provides such prior knowledge as we know (1) the labels for all source samples, and (2) that source samples require no feature correction. Therefore, we train our target classifier f^{T}, to correctly classify source data in addition to target data as a form of regularization. Furthermore, we restrict the additive terms $\triangle\boldsymbol{h}(\boldsymbol{x})$ to be near-zero for source data. This is accomplished with the following regularization term:

$$\mathcal{R}_{\mathrm{S}}(\boldsymbol{x}) = \begin{cases} \ell_{\mathrm{S}}(\boldsymbol{x}) + \frac{1}{p}\|\triangle\boldsymbol{h}(\boldsymbol{x})\|^2, & \boldsymbol{x} \in \mathcal{D}_{\mathrm{S}}, \\ 0, & \boldsymbol{x} \in \mathcal{D}_{\mathrm{T}}, \end{cases} \tag{11}$$

where ℓ_{S} is a softmax loss for source data, and p is the number of parameters in $\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})$. The first term preserves source class information, while the second term explicitly minimizes the correction.

Full objective function. The final objective function combines the two MMD terms and regularization for each layer i:

$$\ell = \mathcal{R}_{\mathrm{S}} + \sum_{i\in\mathcal{L}} \alpha\ell_i^{\mathrm{m}} + \beta\ell_i^{\mathrm{c}}, \tag{12}$$

where i indexes the layer and $\mathcal{L}$ denotes the set of layers with corrections in the network. α and β are hyperparameters. While these hyperparameters could be specific to each correction, we assign the same parameters to all cases to avoid an expensive search. (Training details will be described in Section 4).

Correction architecture. Our correction layers follow an architecture inspired by He et al. (2016), which can be seen in Fig. 2. The residual function $\triangle\boldsymbol{h}(\boldsymbol{x})$ is learned by two weight layers, each followed by batch normalization (Ioffe and Szegedy, 2015), and a ReLU nonlinearity in between. The weight layers are either 3×3 convolution layers or fully connected layers, depending on the source layers. The

full transformation $\boldsymbol{h}_{\mathrm{T}}(\boldsymbol{x})$ is computed through an element-wise addition of $\boldsymbol{h}_{\mathrm{S}}(\boldsymbol{x})$ and the final residual layer output. Network diagrams can be found in the supplementary materials.

4 Experiments and results

We compare LDC with the state-of-the-art deep domain adaptation and traditional shallow domain adaptation methods on several popular cross-domain datasets. Example images from the datasets are shown in Fig. 3. In the following, we will introduce the datasets in detail.

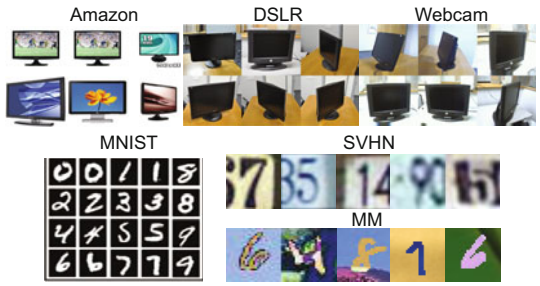

Fig. 3 Example images from different datasets (Amazon (A), DSLR (D), Webcam (W), MNIST, MM, SVHN)

4.1 Datasets

Digits. The MNIST dataset (LeCun et al., 1998) is a widely used grayscale hand-written digit dataset, containing 60 000 training images and 10 000 testing images, and MNIST-M (MM) (Ajakan et al., 2014) is a dataset of MNIST digit superimposed on random image backgrounds. Street view house numbers (SVHN) (Netzer et al., 2011) contains various crops of house numbers from Google Street View. Obviously, SVHN comes from a significantly harder real-world problem, which contains 73 257 images for training and 26 032 digits for testing. We test LDC adaptation across the following digit classification tasks: SVHN→MNIST, SVHN→MM, and MNIST→MM. In each task, the images are rescaled to a resolution of 28×28.

Objects. Office-31 (Saenko et al., 2010) is a standard for evaluating domain adaptation algorithms. It consists of 4110 images, each with resolution of 300×300, across 31 distinct classes of office items, ranging from a computer monitor to a letter tray. The images were collected from three distinct domains: amazon.com (A), a DSLR camera (D), and a webcam (W). All Amazon photos were taken in a studio environment against a white background. DSLR and Webcam photos were captured 'in-the-wild' in a typical office setting; however, the two environments differ in terms of image quality. We test adaptation against all permutations of the domains: A→D, A→W, D→A, D→W, W→A, and W→D.

4.2 Baselines

In this study, we test our approach against two conventional transfer learning methods, i.e., transfer component analysis (TCA) (Pan et al., 2011) and geodesic flow kernel (GFK) (Gong et al., 2012), and four recent deep domain adaptation approaches, i.e., domain adversarial networks (RevGrad) (Ganin and Lempitsky, 2015), deep domain confusion (DDC) (Tzeng et al., 2014), deep adaptation networks (DAN) (Long et al., 2015), and residual transfer learning (RTN) (Long et al., 2016b).

TCA decreases the distance between source and target domains via the learned transfer components underlying both domains, and GFK aims to bridge both domains by interpolating across an infinite number of intermediate subspaces. For both TCA and GFK, we adopt DeCAF_7 (Donahue et al., 2014) features for Objects datasets.

RevGrad uses a gradient-reversal trick to ensure that the features learned by the network cannot distinguish source and target data. DDC and DAN learn domain invariance through MMD-based objective functions, similar to LDC. RTN modifies DAN with the addition of a gated residual block on the final classification layer. Finally, as a lower bound, we feed target data through an unadapted source network (source).

4.3 Network architectures and training

In each experiment, we use the same source network architecture as used in Ganin and Lempitsky (2015). For MNIST→MM, our source network is based on the five-layer LeNet architecture (LeCun et al., 1998), modified with batch normalization and ReLU non-linearities. For SVHN→MNIST and SVHN→MM, the source network is based on the SVHNNet architecture proposed by Srivastava et al.

(2014), with batch-normalization in place of dropout. For the Office experiments, we test our method on two architectures, both of which are pretrained on the ImageNet corpus (Russakovsky et al., 2015): (1) AlexNet, the source network architecture used by Krizhevsky et al. (2017), and (2) ResNet18, a pretrained 18-layer residual network (He et al., 2016).

LeNet, SVHNNet, and AlexNet. Because the first layers of neural networks are known to be sufficiently general across a variety of tasks (Yosinski et al., 2014), we add corrections only after later source network layers. We outfit LeNet with corrections after layers 2–5, SVHN after layers 3–6, and AlexNet after layers 4–8. We use minibatches of size 256 which contain an equal number of source and target samples. Network specifics can be found in the supplementary materials.

ResNet18. We hypothesize that LDC is especially suited for deep modern architectures, such as ResNets. In particular, the increased depth of such architectures provides three key advantages:

1. Reduced task-specificity in convolutional layers: while shorter networks such as AlexNet are known to suffer from a lack of transferability in later layers (Yosinski et al., 2014), deeper architectures are hypothesized to learn more general features due to increased depth (Simonyan and Zisserman, 2014). As a result, we modify the ResNet18 network with only two corrections: one after the final pooled feature map, and one after the fully connected classification layer.

2. Layers with fewer parameters: The final pooled feature map of ResNet18 has 512 variables, compared with the 9216 of AlexNet. Thus, our two corrections only add 1M parameters to the 11.1M-parameter source network.

3. Extremely efficient backproagation: Since the parameters of the source network are fixed, backpropagation is only required through the first correction layer. This results in a significant training speedup for ResNet18: a majority of the backward pass can be eliminated because the corrections are added to the end of the network. Additionally, this efficient backpropagation has the added benefit of requiring much less GPU memory during training. As a result, we can train the adapted ResNet18 model with a minibatch of size 512, ensuring better estimates of MMD and class-conditional MMD each iteration. Network specifics can be found in the supplementary materials.

Training details. We add class-conditional MMD loss only to the final two residual blocks of networks. Each minibatch contains equal numbers of source and target samples. We set the minibatch size to 256 for the digit experiments (on average, 11 source samples and 11 target ones per class), and size 512 for the object experiments (eight source samples and eight target ones per class).

All weighted corrections are initialized with the scheme described in He et al. (2015). The source models are all trained with stochastic gradient descent with the learning rate set to 0.1, weight decay set to 0.0001, and Nesterov momentum (Sutskever et al., 2013) set to 0.9. We drop the learning rate at 40% and 80% of the training. The adapted LeNet and SVHNNet experiments follow the same scheme, while the adapted AlexNet and ResNet are trained with Adam (Kingma and Ba, 2014) and a base learning rate of 0.01.

The objective function contains two hyperparameters: α, which weights the marginal MMD; β, which weights the class-conditional MMD. To avoid a costly grid search, we use simple settings for the two hyperparameters: on layers with class-conditional MMD, we set α to 0.5 and β to 1; on all other layers, α is set to 1.

4.4 Results

Objects. We test adaptation approaches in a fully transductive setup, where all unlabeled target data is used for training. Table 1 shows the results of the Office adaptation experiments. For most tasks, the deep domain adaptation methods are superior to the shallow ones, i.e., GFK and TCA. This shows that deep domain adaptation methods can learn more domain-invariant and discriminative representations to improve the final classification performance.

From the performance of the unadapted source models, it can be seen that the most challenging adaptations involve the Amazon domain. The unadapted ResNet18 models outperform the unadapted AlexNet models on these tasks, but still suffer from high target errors. This shows that adaptation is necessary, even for deeper source networks. After adaptation with LDC, both the AlexNet and ResNet18 models exhibit better target accuracy. Both LDC varieties match or reduce the target error of the

baseline MMD methods (DDC and DAN) and the residual-based method (RTN). We attribute this performance to increased capacity resulting from multiple corrections and a layer-wise MMD loss.

The LDC-ResNet18 models achieve remarkable target accuracy, producing state-of-the-art results in all the tasks (six out of the six adaptations). In most experiments, ResNet18 models reduce the target error by 30% or more (significantly more than the LDC-AlexNet models). This performance gap can be attributed to the thinner layers of ResNet18. This reduces the number of correction parameters, which may make the layers easy to train. Additionally, this reduces the dimensionality of the feature maps, simplifying the MMD estimates. As current trends favor deep and thin architectures (He et al., 2016), we expect that LDC will be well suited for modern neural networks.

Digits. Table 2 displays the adaptation accuracy on a withheld target test set for each digit experiment. First, we observe that the unadapted source networks are not well suited to classify target data, while all two adaptation approaches successfully increase the target accuracy in all tasks. RevGrad performs very well on MNIST→MM adaptation, with LDC slightly outperforming it. For the two SVHN adaptations, LDC achieves higher accuracy than the existing baseline models by a wide margin. We hypothesize that this performance can be attributed to the increased representational power of LDC. The corrections add significant depth to the original network. This may be necessary to learn such a complex adaptation.

5 Analytic experiments

5.1 Feature visualization

Fig. 4 shows a t-SNE embedding (Maaten and Hinton, 2008) of source and target representations from the SVHN→MNIST adapted network during different stages of training. We can make the following observations: (1) The unadapted features are disjoint before adaptation yet become increasingly aligned throughout the adaptation process. (2) The representations in corrected layers 5 and 6 gradually cluster into 10 distinct classes, which may correspond to the 10 digits. We attribute this clustering to class-conditional MMD loss, which is added to these final layers. (3) By the end of training, source and target samples are nearly indistinguishable in the final layer.

5.2 Network convergence

In Fig. 5, we compare the convergence speed of LDC with those of DAN and RevGrad on the

Table 1 Classification accuracy on Office-31 dataset*

Method	Classification accuracy						
	A→W	D→W	W→D	A→D	D→A	W→A	Average
TCA (Pan et al., 2011)	59.0±0.0	90.2±0.0	88.2±0.0	57.8±0.0	51.6±0.0	47.9±0.0	65.8
GFK (Gong et al., 2012)	58.4±0.0	93.6±0.0	91.0±0.0	58.6±0.0	52.4±0.0	46.1±0.0	66.7
AlexNet (Krizhevsky et al., 2017)	60.6±0.4	95.4±0.2	99.0±0.1	64.2±0.3	45.5±0.5	48.3±0.5	68.8
ResNet18 (He et al., 2016)	65.7±0.2	95.9±0.1	98.3±0.2	69.8±0.2	51.7±0.1	49.0±0.3	71.7
DDC (Tzeng et al., 2014)	61.0±0.5	95.0±0.3	98.5±0.3	64.9±0.4	47.2±0.5	49.4±0.6	69.3
RevGrad (Ganin and Lempitsky, 2015)	73.0±0.6	96.4±0.4	99.2±0.3	72.8±0.4	54.4±0.3	53.6±0.2	74.9
DAN (Long et al., 2015)	68.5±0.3	96.0±0.1	99.0±0.1	66.8±0.2	50.0±0.4	49.8±0.3	71.7
RTN (Long et al., 2016b)	73.3±0.3	96.8±0.2	99.6±0.1	71.0±0.2	50.5±0.3	51.0±0.1	73.7
LDC-AlexNet	73.7±0.4	95.6±0.6	96.9±0.3	65.3±0.2	55.1±0.6	53.3±0.2	73.3
LDC-ResNet18	**78.6±0.5**	**98.7±0.1**	**100.0±0.0**	**79.1±0.3**	**61.9±0.3**	**59.6±0.5**	**79.7**

*: AlexNet is an AlexNet model, and ResNet18 is a ResNet18 model

Table 2 Classification accuracy on MNIST, MM, and SVHN datasets

Method	Classification accuracy			
	MNIST→MM	SVHN→MNIST	SVHN→MM	Average
Source-network	49.0±3.4	65.6±2.2	49.7±0.2	54.8
RevGrad (Ganin and Lempitsky, 2015)	81.0±1.0	73.5±0.5	54.9±1.1	69.8
LDC	**84.8±1.2**	**89.5±2.1**	**71.4±0.3**	**81.9**

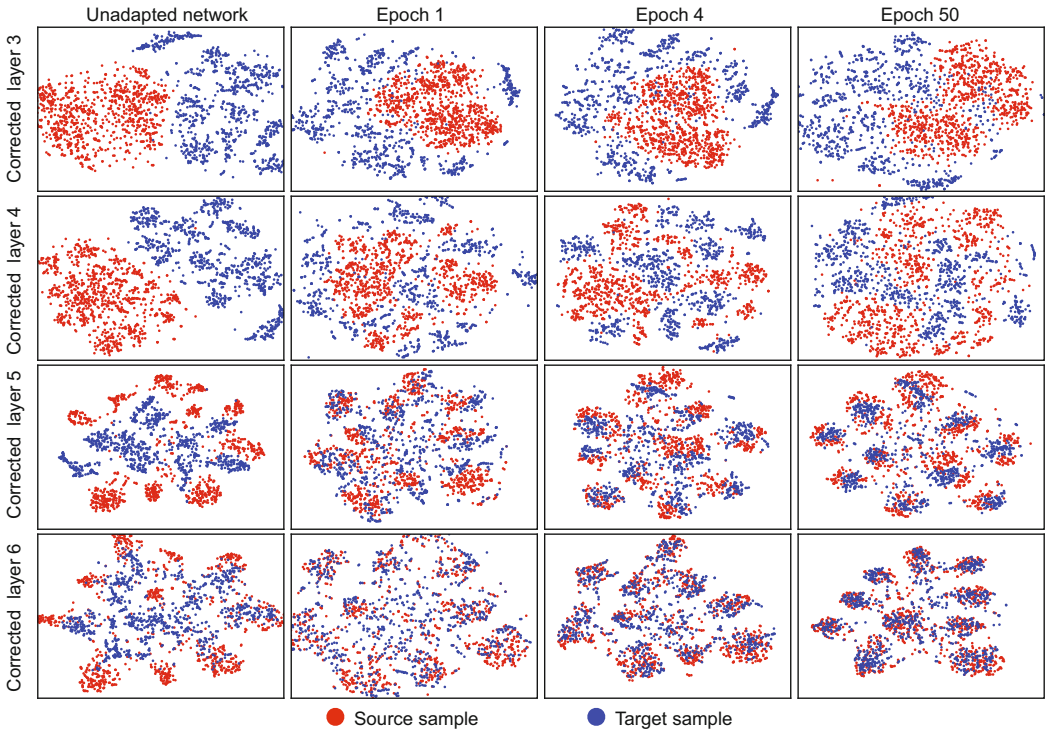

Fig. 4 t-SNE visualizations of different adapted layers at different stages of training SVHN→MNIST

Source samples are red and target samples are blue. As training proceeds, the source and target data are aligned, becoming nearly indistinguishable in the final layer by the end of training. References to color refer to the online version of this figure

A→W experiment, as measured by CPU clock time. The LDC model adapts remarkably fast, despite being significantly deeper than the baseline methods. It reaches the 67% accuracy of DAN in one-tenth the training time, and reaches the 74% accuracy of RevGrad similarly in one-tenth the time. This fast convergence is a direct consequence of training only two residual functions during adaptation rather than training an entire network.

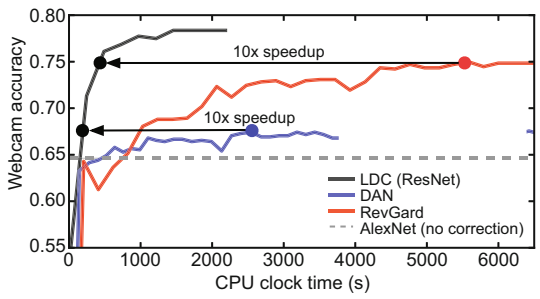

Fig. 5 Test accuracy on A→W as a function of training time

5.3 Layer activations

In Fig. 6, we measure the standard deviation of layer responses as a proxy for how much information is encoded in each layer (He et al., 2016). For each adapted network, the original layers are plotted in gray, while the correction layers are plotted in red. Note that there are two correction layers for every original layer, corresponding to the two weight layers in the residual block.

In each plot, the leftmost layer is the first weight layer of the network. We can observe several trends: first, the response strength of the final correction layer is, in all networks, significantly weaker than other corrections. This highlights the importance of a layer-wise adaptation approach. Additionally, it can be seen that the response strength of the correction layers is comparable to that of the original source layers, suggesting that each adaptive layer is increasing the expressive power of the network. Notably, the correction layers in the ResNet18 model respond much more strongly than its original layers. This is the further evidence that LDC is designed to work well with modern network architectures.

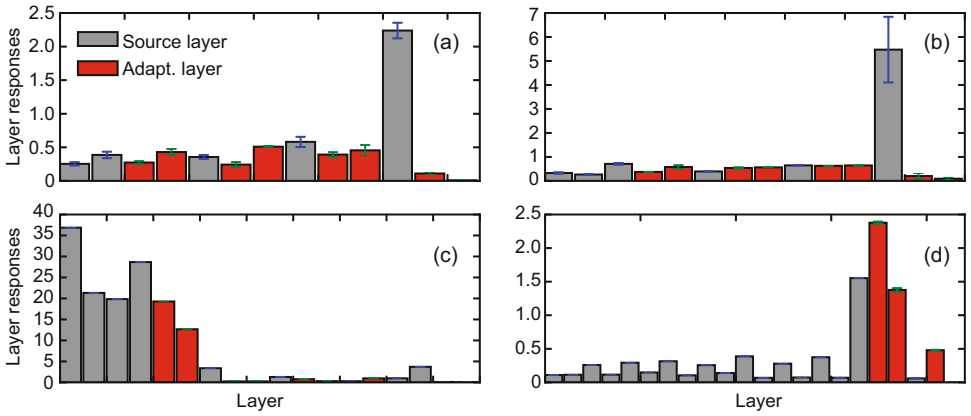

Fig. 6 Standard deviations of activations in each layer of adapted networks: (a) LeNet (MNIST-MM); (b) SVHNNet (SVHN-MNIST); (c) AlexNet (A-D); (d) ResNet18 (A-D)

5.4 Proxy $\mathcal{A}$-distance

Based on the results, we hypothesize that LDC indeed learns domain-invariant features, and finds a representation in which the source and target samples are similar. $\mathcal{A}$-distance is a measure of distance between two representations (Ben-David et al., 2010).Given a collection $\mathcal{A}$ of subsets of $\mathcal{X}$, let P_{S} and P_{T} be distributions on $\mathcal{X}$. Then the $\mathcal{A}$-distance between P_{S} and P_{T} is

$$d_{\mathcal{A}}(P_{\mathrm{S}}, P_{\mathrm{T}}) = 2 \sup_{A \in \mathcal{A}} |P_{\mathrm{S}}(A) - P_{\mathrm{T}}(A)|. \tag{13}$$

A larger $\mathcal{A}$-distance indicates that it is easier to discriminate between two representations. Hence, the domain discrepancy is larger. As directly computing $\mathcal{A}$-distance is intractable, we resolve to the proxy $\mathcal{A}$-distance (PAD). PAD is defined as

$$\widehat{d}_{\mathcal{A}} = 2(1 - 2\epsilon), \tag{14}$$

where ϵ is the test error of classifying representations into the source and target domains. Fig. 7 shows the PAD of three different representations in the task SVHN→MNIST. We choose the activations of the layer right before the classification layer as the unadapted source network and LDC adapted network representations of data. We also compare them with the preprocessed raw features. From Fig. 7 we can see that the raw features and source network representations have PAD close to 2.0, which indicates that the network can discriminate source and target samples nearly perfectly, and that LDC can confuse the classifier to a certain degree. This result shows that LDC decreases the domain discrepancy by learning indistinguishable representations of source and target samples.

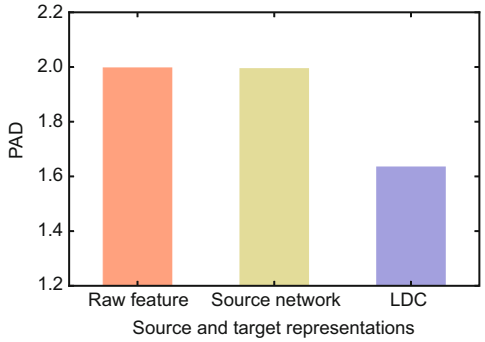

Fig. 7 Proxy $\mathcal{A}$-distance (PAD) between source and target representations of raw features (after preprocessing), source network, and LDC, on the SVHN→MNIST task

6 Conclusions

We have proposed layer-wise domain correction, a new method for unsupervised domain adaptation. LDC leverages the power of residual networks to learn small layer-wise transformations which reduce the propagation of domain differences. In contrast to prior work, it does not re-train the weights of the original classifier from scratch and is therefore substantially faster during the adaptation time. In addition to the speedup, LDC requires substantially less storage for each target domain: only the weights of the corrected layers have to be stored in addition to the original network. Thus, LDC is especially well suited for adapting deep modern architectures, such as residual networks. The additional layers increase the capacity of the neural network, which may be a

reason for its excellent generalization performance. The feature visualization and proxy $\mathcal{A}$ distance experiments demonstrate that LDC can learn perfect domain invariant representations for domain adaptation problems, and comprehensive experiments show that LDC is superior to previous methods.

References

Ajakan H, Germain P, Larochelle H, et al., 2014. Domain-adversarial neural networks. https://arxiv.org/abs/1412.4446

Ben-David S, Blitzer J, Crammer K, et al., 2010. A theory of learning from different domains. *Mach Learn*, 79(1-2):151-175. https://doi.org/10.1007/s10994-009-5152-4

Blitzer J, McDonald R, Pereira F, 2006. Domain adaptation with structural correspondence learning. Proc Conf on Empirical Methods in Natural Language Processing, p.120-128. https://doi.org/10.3115/1610075.1610094

Borgwardt KM, Gretton A, Rasch MJ, et al., 2006. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49-e57. https://doi.org/10.1093/bioinformatics/btl242

Chen MM, Weinberger KQ, Blitzer JC, 2011. Co-training for domain adaptation. Advances in Neural Information Processing Systems, p.2456-2464.

Chen MM, Xu ZX, Weinberger K, et al., 2012. Marginalized denoising autoencoders for domain adaptation. https://arxiv.org/abs/1206.4683

Donahue J, Jia YQ, Vinyals O, et al., 2014. Decaf: a deep convolutional activation feature for generic visual recognition. Proc 31st Int Conf on Machine Learning, p.647-655.

Duan LX, Tsang IW, Xu D, et al., 2009. Domain transfer SVM for video concept detection. IEEE Conf on Computer Vision and Pattern Recognition, p.1375-1381. https://doi.org/10.1109/CVPR.2009.5206747

Duan LX, Tsang IW, Xu D, 2012. Domain transfer multiple kernel learning. *IEEE Trans Patt Anal Mach Intell*, 34(3):465-479. https://doi.org/10.1109/TPAMI.2011.114

Ganin Y, Lempitsky V, 2015. Unsupervised domain adaptation by backpropagation. Proc 32nd Int Conf on Machine Learning, p.1180-1189.

Gardner JR, Upchurch P, Kusner MJ, et al., 2015. Deep manifold traversal: changing labels with convolutional features. https://arxiv.org/abs/1511.06421

Gehring J, Auli M, Grangier D, et al., 2017. Convolutional sequence to sequence learning. https://arxiv.org/abs/1705.03122

Glorot X, Bordes A, Bengio Y, 2011. Domain adaptation for large-scale sentiment classification: a deep learning approach. Proc 28th Int Conf on Machine Learning, p.513-520.

Gong BQ, Shi Y, Sha F, et al., 2012. Geodesic flow kernel for unsupervised domain adaptation. IEEE Conf on Computer Vision and Pattern Recognition, p.2066-2073. https://doi.org/10.1109/CVPR.2012.6247911

Gong BQ, Grauman K, Sha F, 2013. Connecting the dots with landmarks: discriminatively learning domain-invariant features for unsupervised domain adaptation. Proc 30th Int Conf on Machine Learning, p.222-230.

Gretton A, Borgwardt KM, Rasch MJ, et al., 2012. A kernel two-sample test. *J Mach Learn Res*, 13(1):723-773.

He KM, Zhang XY, Ren SQ, et al., 2015. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. IEEE Int Conf on Computer Vision, p.1026-1034. https://doi.org/10.1109/ICCV.2015.123

He KM, Zhang XY, Ren SQ, et al., 2016. Deep residual learning for image recognition. IEEE Conf on Computer Vision and Pattern Recognition, p.770-778. https://doi.org/10.1109/CVPR.2016.90

Hoffman J, Tzeng E, Park T, et al., 2017. CyCADA: cycle-consistent adversarial domain adaptation. https://arxiv.org/abs/1711.03213

Ioffe S, Szegedy C, 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. Proc 32nd Int Conf on Machine Learning, p.448-456.

Kingma DP, Ba J, 2014. Adam: a method for stochastic optimization. https://arxiv.org/abs/1412.6980

Krizhevsky A, Sutskever I, Hinton GE, 2017. ImageNet classification with deep convolutional neural networks. *Commun ACM*, 60(6):84-90. https://doi.org/10.1145/3065386

LeCun Y, Bottou L, Bengio Y, et al., 1998. Gradient-based learning applied to document recognition. *Proc IEEE*, 86(11):2278-2324. https://doi.org/10.1109/5.726791

Li YJ, Swersky K, Zemel R, 2015. Generative moment matching networks. Proc 32nd Int Conf on Machine Learning, p.1718-1727.

Long MS, Wang JM, Ding GG, et al., 2013. Transfer feature learning with joint distribution adaptation. Proc IEEE Int Conf on Computer Vision, p.2200-2207. https://doi.org/10.1109/ICCV.2013.274

Long MS, Wang JM, Ding GG, et al., 2014. Transfer joint matching for unsupervised domain adaptation. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.1410-1417. https://doi.org/10.1109/CVPR.2014.183

Long MS, Cao Y, Wang JM, et al., 2015. Learning transferable features with deep adaptation networks. Proc 32nd Int Conf on Machine Learning, p.97-105.

Long MS, Wang JM, Cao Y, et al., 2016a. Deep learning of transferable representation for scalable domain adaptation. *IEEE Trans Knowl Data Eng*, 28(8):2027-2040. https://doi.org/10.1109/TKDE.2016.2554549

Long MS, Zhu H, Wang JM, et al., 2016b. Unsupervised domain adaptation with residual transfer networks. Advances in Neural Information Processing Systems, p.136-144.

Mikolov T, Sutskever I, Chen K, et al., 2013. Distributed representations of words and phrases and their compositionality. Proc 26th Int Conf on Neural Information Processing Systems, p.3111-3119.

Netzer Y, Wang T, Coates A, et al., 2011. Reading digits in natural images with unsupervised feature learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning, p.1-9.

Oquab M, Bottou L, Laptev I, et al., 2014. Learning and transferring mid-level image representations using convolutional neural networks. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.1717-1724. https://doi.org/10.1109/CVPR.2014.222

Pan SJL, Yang Q, 2010. A survey on transfer learning. *IEEE Trans Knowl Data Eng*, 22(10):1345-1359. https://doi.org/10.1109/TKDE.2009.191

Pan SJL, Tsang IW, Kwok JT, et al., 2011. Domain adaptation via transfer component analysis. *IEEE Trans Neur Netw*, 22(2):199-210. https://doi.org/10.1109/TNN.2010.2091281

Russakovsky O, Deng J, Su H, et al., 2015. ImageNet large scale visual recognition challenge. *Int J Comput Vis*, 115(3):211-252. https://doi.org/10.1007/s11263-015-0816-y

Saenko K, Kulis B, Fritz M, et al., 2010. Adapting visual category models to new domains. *LNCS*, 6314:213-226. https://doi.org/10.1007/978-3-642-15561-1_16

Simonyan K, Zisserman A, 2014. Very deep convolutional networks for large-scale image recognition. https://arxiv.org/abs/1409.1556

Srivastava N, Hinton G, Krizhevsky A, et al., 2014. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res*, 15(1):1929-1958.

Sutskever I, Martens J, Dahl G, et al., 2013. On the importance of initialization and momentum in deep learning. Proc 30$^{\text{th}}$ Int Conf on Machine Learning, p.1139-1147.

Sutskever I, Vinyals O, Le Q, 2014. Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems, p.3104-3112.

Tzeng E, Hoffman J, Zhang N, et al., 2014. Deep domain confusion: maximizing for domain invariance. https://arxiv.org/abs/1412.3474

van der Maaten L, Hinton G, 2008. Visualizing data using t-SNE. *J Mach Learn Res*, 9(11):2579-2605.

Yosinski J, Clune J, Bengio Y, et al., 2014. How transferable are features in deep neural networks? Proc 27$^{\text{th}}$ Int Conf on Neural Information Processing Systems, p.3320-3328.

Lists of supplementary materials

Fig. S1 The LDC network architectures: (a) SVHNNet; (b) LeNet

Fig. S2 The LDC network architectures: (a) AlexNet; (b) ResNet18