Supervised Domain Adaptation: Were we doing Graph Embedding all along?

Lukas Hedegaard*, Omar Ali Sheikh-Omar*, and Alexandros Iosifidis
Department of Engineering, Aarhus University, Denmark
lh@eng.au.dk, sheikhomar@mailbox.org, alexandros.iosifidis@eng.au.dk

***Abstract*—The performance of machine learning models tends to suffer when the distributions of the training and test data differ. Domain Adaptation is the process of closing the distribution gap between datasets. In this paper, we show that existing Domain Adaptation methods can be formulated as Graph Embedding methods in which the domain labels of samples coming from the source and target domains are incorporated into the structure of the intrinsic and penalty graphs used for the embedding. To this end, we define the underlying intrinsic and penalty graphs for three state-of-the-art supervised domain adaptation methods. In addition, we propose the Domain Adaptation via Graph Embedding (DAGE) method as a general solution for supervised Domain Adaptation, that can be combined with various graph structures for encoding pair-wise relationships between source and target domain data. Moreover, we highlight some generalisation and reproducibility issues related to the experimental setup commonly used to evaluate the performance of Domain Adaptation methods. We propose a new evaluation setup for more accurately assessing and comparing different supervised DA methods, and report experiments on the standard benchmark datasets Office31 and MNIST-USPS.**

***Index Terms*—Domain Adaptation, Graph Embedding, Transfer Learning, Domain Shift**

I. Introduction

Deep neural networks have been applied successfully to a variety of applications. However, their performance tends to suffer when a trained model is applied to another domain. This is of no surprise, as statistical learning theory makes the simplifying assumption that both training and test data are generated by the same underlying process; the use of real-world datasets makes the i.i.d. assumption impractical as it requires collecting data and training a model for each domain. The collection and labelling of datasets that is sufficiently large to train a well-performing model from random initialisation, is daunting and costly. Therefore, we often have little data for the task at hand. Moreover, training a deep network with scarce training data can lead to overfitting [1].

The process aiming to alleviate this challenge is commonly referred to as transfer learning. The main idea in transfer learning is to leverage knowledge extracted from one or more source domains to improve the performance on problems defined in a related target domain [2, 3, 4]. In the image classification task, we may want to utilise the large number of labelled training samples in the ImageNet database to improve the performance on another image classification task on a very different domain, e.g. that of fine-grained classification of aquatic macroinvertebrates [5]. This is often done by reusing the parameters of a deep learning model trained on a large source domain dataset.

To clearly define transfer learning, the literature distinguishes between a domain and a task. A domain $\mathscr{D}$ consists of an input space $\mathcal{X}$ and a marginal probability distribution $p(\mathbf{X})$ where $\mathbf{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \in \mathcal{X}$ are N samples from that space. Given a domain, a task $\mathscr{T}$ is composed of an output space $\mathcal{Y}$ and a function $f(\cdot)$ which can predict $y_i \in \mathcal{Y}$ given some input $\boldsymbol{x}_i$. Suppose we have a source domain $\mathscr{D}_\mathcal{S}$ with its associated task $\mathscr{T}_\mathcal{S}$ and a target domain $\mathscr{D}_\mathcal{T}$ with a corresponding task $\mathscr{T}_\mathcal{T}$. Transfer learning is defined as the process of improving the target predictive function $f_\mathcal{T}(\cdot)$ using the knowledge in $\mathscr{D}_\mathcal{S}$ and $\mathscr{T}_\mathcal{S}$ when there is a difference between the domains ($\mathscr{D}_\mathcal{S} \neq \mathscr{D}_\mathcal{T}$) or the tasks ($\mathscr{T}_\mathcal{S} \neq \mathscr{T}_\mathcal{T}$) [2]. Two domains or two tasks are said to be different if their constituent parts are not the same. In some cases, the feature space and the label space of the source and target domains are equal. Then the performance degradation when reusing a model in another domain is caused by a *domain shift* or *covariate shift*, i.e. a difference between the distribution of features in the domains. The process of aligning the distributions between the domains is known as Domain Adaptation.

There are two main strategies for closing the distribution gap; instance reweighting and feature transformation [2, 3, 6]. Instance-based Domain Adaptation methods assign weights to source samples and train the target classifier with it. The feature transformation approach applies a transformation to lower the discrepancy in distributions. One way to learn a feature transformation in a deep neural network is to employ a Siamese network architecture as it allows samples from both domains to be introduced as inputs (see Fig. 1). We can view such a neural network model as a function that is composed of two other functions, i.e. $f(\cdot) = h(\varphi_n(\cdot))$ where $\varphi_n(\cdot)$ is the feature extractor and $h(\cdot)$ is the classifier. Defining a loss function on the output of the feature extractor, we can encourage the network to produce domain-invariant features.

A typical goal in transformation-based Domain Adaptation methods is to map samples belonging to the same class close together in a common (latent) subspace (within-class compactness) while separating samples with different labels (between-class separability) irrespective of the originating domain. This problem can be reformulated as a graph embedding problem [7]. In Graph Embedding [8], the goals for increasing the within-class compactness and the between-

*Equal contribution

class separability are enforced by appropriately connecting samples in two graph structures, the intrinsic and penalty graphs. It has been also shown that when data comes from different views, a Generalized Multi-view Embedding based on extended intrinsic and penalty graph structures, which encode information of the different views, can be applied [9].

In this paper, we show that many existing Domain Adaptation methods can be formulated as Graph Embedding methods in which the domain labels of samples coming from the source and target domains are incorporated in the structure of the intrinsic and penalty graphs. Specifically, we analyse the loss functions of three state-of-the-art supervised domain adaptation methods and show that they are specific instantiations of Graph Embedding. We propose the Domain Adaptation via Graph Embedding (DAGE) method as a general solution for supervised Domain Adaptation that can be combined with various graph structures encoding pair-wise relationships between source and target domain data. Furthermore, we propose a new experimental setup for supervised Domain Adaptation that we believe can serve as a better benchmark of the performance of the methods. To facilitate research in this area to utilise the new setup, we release a new open source library called Dataset Ops[1] which can be used for reproducibility purposes. We also make the source code of our experiments available[2]. This work is a substantial extension of our work in [7].

The remainder of the paper is structured as follows. In Section II, we provide a brief overview of methods in the domain adaptation literature. We introduce graph embedding, how to optimise the graph preserving criterion and multi-view extensions in Section III. Section IV begins by delineating Domain Adaptation via Graph Embedding (DAGE) framework. Then, we analyse three different methods and justify how they are different instances of the DAGE framework. In Section V, we explain the issues with the existing experimental setup used in prior Domain Adaptation work and propose a new setup. Subsequently, we present the results of our experiments on the canonical datasets Office-31 and MNIST-USPS. Finally, in Section VI, we conclude on our findings.

II. Related Works

Domain adaptation can be broadly categorised by which feature spaces the data resides in, and by which label information is available. When the available data in source and target domains have the same space (e.g. RGB images or VGG-16 [13] features), it is denoted *Homogeneous Domain Adaptation*. This is the most common flavour, and what many authors simply refer to as domain adaptation [2, 10, 11, 12]. Our work also falls into this category, and we will freely use Domain Adaptation to denote Homogeneous Domain Adaptation as well. When features come from different spaces (e.g. SURF [14] features for the source domain data and DeCAF$_6$ [15] features for the target domain), the domain adaptation problem is denoted as heterogeneous [6]. The *Heterogeneous Domain Adaptation* methods are especially useful when the features naturally reside in predominantly disjoint

[1]Dataset Ops: https://github.com/lukashedegaard/datasetops
[2]DAGE: www.github.com/lukashedegaard/dage

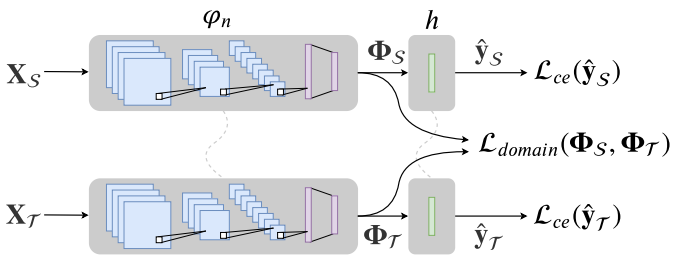


Fig. 1: The two-stream network architecture used in DAGE, CCSA [10], d-SNE [11] and NEM [12]. It allows source domain samples $\mathbf{X}_S$ and target domain samples $\mathbf{X}_T$ to be introduced to a deep convolutional neural network simultaneously. The network is split into a feature extractor $\varphi_n(\cdot)$ and a classifier $h(\cdot)$. A domain adaptation loss $\mathcal{L}_{domain}$ is defined on the output of the feature extractors to encourage the generation of domain-invariant features. At the same time, the categorical cross-entropy loss ensures that the network can discriminate the different classes.

spaces. One such example in Natural Language Processing is when one wants to utilise the vast learning resources available in English to improve performance on a task in a minority language such as Danish. Depending on the tactics used to accomplish this, Heterogeneous Domain Adaptation can be further divided into *asymmetric* methods, when the source space is transformed into the target space, or *symmetric*, when the source and target domains are both transformed into a common space. Some recent contributions using the symmetric approach are [16, 17, 18].

Another view is to categorise methods based on the label availability. Here, domain adaptation methods usually assume that all source data is labelled [6]. When the source data is unlabelled, the methods fall beyond the usual scope of domain adaptation and into the realm of general unsupervised learning [2]. If the source data constitutes a mix of labelled and unlabelled samples, the self-taught learning framework can be applied [19]. In addition to labels in the source domains, *Supervised Domain Adaptation* assumes that all target data is labelled. These methods are especially relevant for data that is costly to obtain, such as classification of medical images, and place their focus on few-shot learning scenarios. Classification and Contrastive Semantic Alignment (CCSA) [10] is one such method, which embeds the contrastive loss introduced by Hadsell et al. [20] as a loss term in a two-stream deep neural network. Effectively, it places a penalty on the distance between samples belonging to the same class across source and target domains, as well as the proximity of samples belonging to different classes, that fall within a distance margin. Domain Adaptation using Stochastic Neighborhood Embedding (d-SNE) [11] uses the same deep two-stream architecture, and finds its inspiration in the dimensionality reduction method of Stochastic Neighbor Embedding (SNE). From it, the work in [11] derives as loss a modified-Hausdorffian distance, which minimises the Euclidean distance in the embedding space between the furthest same-class data pairs, and maximises the distance of the closest different-label pairs. Domain Adapta-

tion With Neural Embedding Matching (NEM) [12] extends the contrastive loss of CCSA with an additional loss term to match the local neighbourhood relations of the target data prior to and after feature embedding. It does so by constructing a graph embedding loss connecting the nearest neighbours of the target data in their original feature space, and adding the weighted sum of distances between corresponding embedded features to the constrastive loss.

When some unlabelled data is available in addition to the labelled target data, *Semi-supervised Domain Adaptation* methods can be used. Examples of this are d-SNE and NEM, both of which provide extensions to include unlabelled data.

Unsupervised Domain Adaptation methods do not assume that any labels are available in the target domain, and use only the label information from the source domain. A recent addition to this space is the Graph Embedding Framework for Maximum Mean Discrepancy-Based Domain Adaptation Algorithm (GEF) [21], which assigns pseudo-labels to target data and solves the generalised eigenvalue problem for a revised Maximum Mean Discrepancy-based graph to compute a linear projection of the source data. The reconstructed source data is then used to train a classifier which in turn updates the psuedo-labels of the target data. Another notable contribution in the space is [22].

III. Graph Embedding and its optimization problem

Graph Embedding [8] is a general dimensionality reduction framework based exploiting graph structures. Suppose we have a data matrix $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{D\times N}$ and we want to obtain its one-dimensional counterpart $\boldsymbol{z} = [z_1, \cdots, z_N] \in \mathbb{R}^{1\times N}$. To encode the data relationships that we want to preserve in the subspace, we can construct a so-called intrinsic graph $G = (\mathbf{X}, \mathbf{W})$ where $\mathbf{W}$ is a non-negative adjacency matrix encoding the (weighted) pair-wise relationships between the representations of the graph vertices included in $\mathbf{X}$. When we want to also suppress relationships between some graph vertices in the embedding space, we can create a penalty graph $G_p = (\mathbf{X}, \mathbf{W}_p)$. The optimal embeddings $\boldsymbol{z}^*$ are found by optimising the graph preserving criterion [8]:

$$\boldsymbol{z}^* = \underset{\boldsymbol{z}^\top\mathbf{B}\boldsymbol{z}=c}{\operatorname{argmin}} \sum_{i\neq j} \|z_i - z_j\|_2^2 \mathbf{W}^{(i,j)} = \underset{\boldsymbol{z}^\top\mathbf{B}\boldsymbol{z}=c}{\operatorname{argmin}} \boldsymbol{z}^\top\mathbf{L}\boldsymbol{z} \quad (1)$$

where c is a constant, $\mathbf{L} = \mathbf{D} - \mathbf{W}$ and $\mathbf{B} = \mathbf{D}_p - \mathbf{W}_p$ are $N\times N$ graph Laplacian matrices of G and G_p, respectively, and $\mathbf{D} = \sum_j \mathbf{W}^{(i,j)}$ and $\mathbf{D}_p = \sum_j \mathbf{W}_p^{(i,j)}$ are the corresponding (diagonal) Degree matrices. When using a linear embedding, $z_i = \boldsymbol{v}^\top\boldsymbol{x}_i$, the above criterion takes the following form:

$$\boldsymbol{z}^* = \underset{\boldsymbol{v}^\top\mathbf{X}\mathbf{B}\mathbf{X}^\top\boldsymbol{v}=c}{\operatorname{argmin}} \boldsymbol{v}^\top\mathbf{X}\mathbf{L}\mathbf{X}^\top\boldsymbol{v}. \quad (2)$$

which is equivalent to maximizing the *trace ratio* problem [23, 24]:

$$\mathcal{J}(\boldsymbol{v}) = \frac{\boldsymbol{v}^\top\mathbf{X}\mathbf{B}\mathbf{X}^\top\boldsymbol{v}}{\boldsymbol{v}^\top\mathbf{X}\mathbf{L}\mathbf{X}^\top\boldsymbol{v}}. \quad (3)$$

Following Lagrange-based optimisation, the optimal projection $\boldsymbol{v} \in \mathbb{R}^D$ is found by solving the generalized eigenanalysis problem $\mathbf{X}\mathbf{B}\mathbf{X}^\top\boldsymbol{v} = \lambda\mathbf{X}\mathbf{L}\mathbf{X}^\top\boldsymbol{v}$ and is given by the eigenvector corresponding to the maximal eigenvalue.

When more than one-dimensional embedding spaces are needed, i.e. when the mapping takes the form of $\mathbb{R}^D \to \mathbb{R}^d$ with $1 < d \leq D$, trace ratio problem in Eq. (3) takes the form:

$$\mathcal{J}(\mathbf{V}) = \frac{\operatorname{Tr}\left(\mathbf{V}^\top\mathbf{X}\mathbf{B}\mathbf{X}^\top\mathbf{V}\right)}{\operatorname{Tr}\left(\mathbf{V}^\top\mathbf{X}\mathbf{L}\mathbf{X}^\top\mathbf{V}\right)}. \quad (4)$$

where $\operatorname{Tr}(\cdot)$ is the trace operator and $\mathbf{V} \in \mathbb{R}^{D\times d}$ is the corresponding projection matrix. The trace ratio problem in Eq. (4) does not have a closed-form solution. Therefore, it is conventionally approximated by solving the *ratio trace* problem, $\tilde{\mathcal{J}}(\mathbf{V}) = \operatorname{Tr}[(\mathbf{V}^\top\mathbf{X}\mathbf{L}\mathbf{X}^\top\mathbf{V})^{-1}(\mathbf{V}^\top\mathbf{X}\mathbf{B}\mathbf{X}^\top\mathbf{V})]$, which is equivalent to the optimization problem $\mathbf{X}\mathbf{B}\mathbf{X}^\top\boldsymbol{v} = \lambda\mathbf{X}\mathbf{L}\mathbf{X}^\top\boldsymbol{v}$, $\lambda \neq 0$, and the columns of $\mathbf{V}$ are given by the eigenvectors of the matrix $\left((\mathbf{X}\mathbf{L}\mathbf{X}^\top)^{-1}(\mathbf{X}\mathbf{B}\mathbf{X}^\top)\right)$ corresponding to the d maximal eigenvalues. Although the trace ratio problem in Eq. (3) does not have a closed form solution, it was shown in [23] that it can be converted to an equivalent *trace difference* problem:

$$\bar{\mathcal{J}}(\mathbf{V}, \lambda) = \operatorname{Tr}\left(\mathbf{V}^\top(\mathbf{X}\mathbf{B}\mathbf{X}^\top - \lambda\mathbf{X}\mathbf{L}\mathbf{X}^\top)\mathbf{V}\right), \quad (5)$$

where λ is the trace ratio calculated by applying an iterative process. After obtaining the trace ratio value λ^*, the optimal projection matrix $\mathbf{V}^*$ is obtained by substituting λ^* to the trace difference problem in Eq. (5) and maximizing its value.

Non-linear mappings from $\boldsymbol{x}_i \in \mathbb{R}^D$ to $\boldsymbol{z}_i \in \mathbb{R}^d$ can be obtained by exploiting the Representer Theorem, i.e. by using an implicit nonlinear mapping $\phi : \mathbb{R}^D \to \mathcal{F}$, with $\mathcal{F}$ being a reproducing kernel space, leading to $\boldsymbol{x}_i \in \mathbb{R}^D \to \phi(\boldsymbol{x}_i) \in \mathcal{F}$. We can then express the mapping in the form of $\boldsymbol{z}_i = \boldsymbol{\alpha}^\top\boldsymbol{\Phi}^\top\phi(\boldsymbol{x}_i)$ where $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_N)]$ are the training data representations in $\mathcal{F}$ and the projection matrix is given by $\mathbf{V} = \boldsymbol{\Phi}\mathbf{A}$. In that case, the problems in Eqs. (4) and (5) are transformed by substituting $\mathbf{X}$ with $\mathbf{K} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}$, which is the *kernel matrix* calculated using the so-called kernel function $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbf{K}^{(i,j)}$.

Multi-view extensions using intrinsic and penalty graphs for jointly determining data transformations for data coming from multiple input spaces (views) have also been proposed. As was shown in [9], several standard multi-view methods such as Multi-View Fisher Discriminant Analysis [25], Partial Least Squares [26], (deep) Canonical Correlation Analysis [27], and Multi-view Discriminant Analysis [28] can be expressed as specific instantiations of the problem in Eq. (4), which exploit the view label information to define corresponding intrinsic and penalty graphs. Moreover, the Multi-view Nonparametric Discriminant Analysis [29] and Multi-view Learning to Rank [30] methods have been formulated based on the problem in Eq. (4) for retrieval and ranking problems.

IV. Domain Adaptation via Graph Embedding

Given the versatility of graph embedding, we derive the proposed Domain Adaptation via Graph Embedding (DAGE) framework. In this section, we detail DAGE and an instantiation of it inspired by Linear Discriminant Analysis. Next, we analyse three state-of-the-art supervised Domain Adaptation methods and show that they are specific instances of the DAGE framework.

A. DAGE Framework

The aim of transformation-based Domain Adaptation methods is to learn a common subspace where the distribution gap between source domain data and target domain data is as small as possible. In the supervised setting, we want a transformation $\varphi(\cdot)$ which places samples belonging to the same class close together without regard to the originating domain to achieve within-class compactness. On the other hand, we want $\varphi(\cdot)$ to clearly separate samples with different labels irrespective of the domain, gaining between-class separability.

Let $\mathbf{X}_{\mathcal{S}} \in \mathbb{R}^{D\times N_{\mathcal{S}}}$ and $\mathbf{X}_{\mathcal{T}} \in \mathbb{R}^{D\times N_{\mathcal{T}}}$ be two data matrices from the source and target domains, respectively, and let $N = N_{\mathcal{S}} + N_{\mathcal{T}}$. Suppose we have a transformation $\varphi(\cdot)$ which can produce d-dimensional vectors from D-dimensional data. Then we can construct a matrix $\mathbf{\Phi} = [\varphi(\mathbf{X}_{\mathcal{S}})\varphi(\mathbf{X}_{\mathcal{T}})] \in \mathbb{R}^{d\times N}$ containing the transformed data from both domains. By encoding the desired pair-wise data relationships in an intrinsic graph $G = (\mathbf{X}, \mathbf{W})$ and computing its graph Laplacian matrix $\mathbf{L}$, we can formulate a measure of within-class spread as:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \left\| \mathbf{\Phi}^{(i)} - \mathbf{\Phi}^{(j)} \right\|_2^2 \mathbf{W}^{(i,j)} = \text{Tr}\left(\mathbf{\Phi L \Phi}^\top\right) \tag{6}$$

Similarly, we can create a penalty graph $G_p = (\mathbf{X}, \mathbf{W}_p)$ and express the between-class separability using:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \left\| \mathbf{\Phi}^{(i)} - \mathbf{\Phi}^{(j)} \right\|_2^2 \mathbf{W}_p^{(i,j)} = \text{Tr}\left(\mathbf{\Phi B \Phi}^\top\right) \tag{7}$$

Since the goal is to minimise the within-class compactness and maximise the between-class separability, DAGE optimises the following objective function:

$$\varphi^* = \underset{\varphi}{\text{argmin}} \frac{\text{Tr}\left(\mathbf{\Phi L \Phi}^\top\right)}{\text{Tr}\left(\mathbf{\Phi B \Phi}^\top\right)} \tag{8}$$

Note that since Eq. (8) corresponds to a minimization problem, the graphs Laplacian matrices of the intrinsic and the penalty graphs are placed respectively in the numerator and denominator of the trace ratio problem.

When the transformation is linear using a projection matrix $\mathbf{V}$, i.e. $\varphi(\mathbf{X}) = \mathbf{V}^\top\mathbf{X}$, then the DAGE criterion becomes:

$$\mathbf{V}^* = \underset{\mathbf{V}}{\text{argmin}} \frac{\text{Tr}\left(\mathbf{V}^\top\mathbf{X L X}^\top\mathbf{V}\right)}{\text{Tr}\left(\mathbf{V}^\top\mathbf{X B X}^\top\mathbf{V}\right)} \tag{9}$$

where $\mathbf{X} = [\mathbf{X}_{\mathcal{S}}, \mathbf{X}_{\mathcal{T}}]$. The optimal transformation matrix $\mathbf{V}^*$ is obtained by solving the ratio trace problem. Its solution is formed by the eigenvectors corresponding to the d largest eigenvalues of the generalised eigenvalue problem $\mathbf{X B X}^\top\boldsymbol{v}^* = \lambda\mathbf{X L X}^\top\boldsymbol{v}^*$, or by minimising the trace difference problem as described in Section III:

$$\bar{\mathcal{J}}(\mathbf{V}, \lambda) = \text{Tr}\left(\mathbf{V}^\top(\mathbf{X L X}^\top - \lambda\mathbf{X B X}^\top)\mathbf{V}\right) \tag{10}$$

The linear DAGE criterion in Eq. (9) can also be formulated using the kernel trick for deriving non-linear mappings. Suppose $\phi : \mathbb{R}^D \to \mathcal{F}$ is a nonlinear function mapping the input data into a reproducing kernel Hilbert space $\mathcal{F}$. Let the matrix $\mathbf{\Phi} = [\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_N)]$ be composed of data in $\mathcal{F}$. Based on the Representer Theorem, we let $\mathbf{V} = \mathbf{\Phi A}$ and get:

$$\mathbf{A}^* = \underset{\mathbf{A}}{\text{argmin}} \frac{\text{Tr}\left(\mathbf{A}^\top\mathbf{K L K A}\right)}{\text{Tr}\left(\mathbf{A}^\top\mathbf{K B K A}\right)} \tag{11}$$

where $\mathbf{K} = \mathbf{\Phi}^\top\mathbf{\Phi}$ has elements equal to $\mathbf{K}^{(i,j)} = \phi(\boldsymbol{x}_i)^\top \cdot \phi(\boldsymbol{x}_j)$. The solution of the kernelised DAGE formulation in Eq. (11) can be found via generalised eigenvalue decomposition or applying an iterative process similar to the linear case.

Eigenvalue decomposition for nonlinear DAGE is intractable for large datasets as the computational complexity is in the order of $\mathcal{O}(N^3)$ [31]. An alternative solution is to express the DAGE criterion as part of the loss function in a deep neural network. For supervised domain adaptation problems in the visual domain, the first layers of a neural network architecture can be seen as a non-linear parametric function $\varphi_n(\cdot)$ taking as input the raw image data and giving as output vector representations. This allows the DAGE objective to be optimised using gradient descent-based approaches. Moreover, the DAGE loss can be optimised together with a classification loss (e.g. cross-entropy) in an end-to-end manner. Given a mini-batch b of data, the DAGE loss can be computed:

$$\mathcal{L}_{\text{DAGE}} = \frac{\text{Tr}\left(\mathbf{\Phi}_b\mathbf{L}_b\mathbf{\Phi}_b^\top\right)}{\text{Tr}\left(\mathbf{\Phi}_b\mathbf{B}_b\mathbf{\Phi}_b^\top\right)}, \tag{12}$$

where $\mathbf{\Phi}_b = \left[\varphi_n\left(\mathbf{X}_{\mathcal{S}}^{(b)}\right), \varphi_n\left(\mathbf{X}_{\mathcal{T}}^{(b)}\right)\right]$ is a matrix formed by the transformed features in the mini-batch b and the graph Laplacian matrices $\mathbf{L}_b$ and $\mathbf{B}_b$ are computed on the data forming the mini-batch. The loss function to be optimised is the sum of the DAGE loss and classification losses for source and target domain data:

$$\underset{\theta_\varphi, \theta_h}{\text{argmin}}\ \mathcal{L}_{\text{DAGE}} + \beta\ \mathcal{L}_{\text{CE}}^{\mathcal{S}} + \gamma\ \mathcal{L}_{\text{CE}}^{\mathcal{T}} \tag{13}$$

where θ_φ and θ_h denote the parameters of the parametric functions $\varphi_n(\cdot)$ (feature extractor) and $h(\cdot)$ (classifier), respectively. β and γ indicate the weight of the classification losses for the source and target data, respectively.

B. DAGE-LDA

The DAGE criterion in Eq. (8) is a generic criterion which can lead to a multitude of Domain Adaptation solutions. Constructing the two graphs G and G_p in different ways gives rise to different properties to be optimised in the subspace $\mathbb{R}^d$. A simple instantiation of DAGE inspired by Linear Discriminant Analysis is obtained by using an intrinsic graph structure connecting the samples belonging to the same class:

$$\mathbf{W}^{(i,j)} = \begin{cases} 1, & \text{if } \ell_i = \ell_j \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where ℓ_i and ℓ_j are the labels associated with the i-th and j-th samples, respectively. The corresponding penalty graph structure connects samples belonging to different classes:

$$\mathbf{W}_p^{(i,j)} = \begin{cases} 1, & \text{if } \ell_i \neq \ell_j \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

Despite the simplicity of the above-described DAGE instantiation, the resulting method performs on par with state-of-the-art Domain Adaptation methods, as will be shown in Section V.

C. Existing supervised Domain Adaptation Methods as DAGE

To illustrate the general nature of the DAGE framework, we analyse three recent state-of-the-art supervised Domain Adaptation methods and show that they are specific instantiations of DAGE. Here we should note that several other Domain Adaptation methods such as [32], which are formulated based on pair-wise data relationships, can also be formulated as specific instantiations of DAGE. In the subsequent sections, we focus on the Domain Adaptation terms included in the optimisation function of each method, while we omit the corresponding cross-entropy terms of each method for simplicity.

1) Classification and Contrastive Semantic Alignment: The contrastive semantic alignment loss of CCSA [10] is constructed from two terms: A similarity loss $\mathcal{L}_S$, which penalises the distance between within-class samples of different domains, and a dissimilarity loss $\mathcal{L}_D$, which penalises the proximity of between-class samples if they come within a distance margin ϵ, i.e.:

$$\mathcal{L}_{\text{CSA}} = \mathcal{L}_{\text{S}} + \mathcal{L}_{\text{D}}. \tag{16}$$

Using as notational shorthand $d_{ij} = \|\varphi_n(\boldsymbol{x}_i) - \varphi_n(\boldsymbol{x}_j)\|_2$, the partial losses are defined as follows:

$$\mathcal{L}_{\text{S}} = \sum_{\substack{\boldsymbol{x}_i \in \mathscr{D}_{\mathcal{S}} \\ \boldsymbol{x}_j \in \mathscr{D}_{\mathcal{T}} \\ \ell_i = \ell_j}} \frac{1}{2} d_{ij}^2 \tag{17}$$

$$\mathcal{L}_{\text{D}} = \sum_{\substack{\boldsymbol{x}_i \in \mathscr{D}_{\mathcal{S}} \\ \boldsymbol{x}_j \in \mathscr{D}_{\mathcal{T}} \\ \ell_i \neq \ell_j}} \frac{1}{2} \max\{0, \epsilon - d_{ij}\}^2 . \tag{18}$$

The similarity loss can be expressed equivalently in terms of the weighted summation over graph edges:

$$\mathcal{L}_{\text{S}} = \sum_{\substack{\boldsymbol{x}_i \in \mathscr{D}_{\mathcal{S}} \\ \boldsymbol{x}_j \in \mathscr{D}_{\mathcal{T}}}} \|\varphi_n(\boldsymbol{x}_i) - \varphi_n(\boldsymbol{x}_j)\|_2^2 \, \mathbf{W}^{(i,j)} = \text{Tr}(\boldsymbol{\Phi} \mathbf{L} \boldsymbol{\Phi}^\top) \tag{19}$$

where the graph weight matrix $\mathbf{W}$ has an edge for sample-pairs with the same label but different originating domains

$$\mathbf{W}^{(i,j)} = \begin{cases} \frac{1}{2}, & \text{if } \ell_i = \ell_j \text{ and } \mathscr{D}_i \neq \mathscr{D}_l \\ 0, & \text{otherwise,} \end{cases} \tag{20}$$

and $\mathbf{L}$ is the graph Laplacian matrix associated with $\mathbf{W}$. Using the fact that $\max\{f(x)\} = -\min\{-f(x)\}$, the dissimilarity loss can likewise be expressed in terms of a summation over graph edges:

$$\begin{aligned} \mathcal{L}_{\text{D}} &= -\sum_{\substack{\boldsymbol{x}_i \in \mathscr{D}_{\mathcal{S}} \\ \boldsymbol{x}_j \in \mathscr{D}_{\mathcal{T}} \\ \ell_i \neq \ell_j \\ d_{ij} < \epsilon}} \frac{1}{2} (d_{ij} - \epsilon)^2 = -\sum_{\substack{\boldsymbol{x}_i \in \mathscr{D}_{\mathcal{S}} \\ \boldsymbol{x}_j \in \mathscr{D}_{\mathcal{T}} \\ \ell_i \neq \ell_j \\ d_{ij} < \epsilon}} d_{ij}^2 \frac{1}{2} \left(1 + \frac{\epsilon^2}{d_{ij}^2} - \frac{2\epsilon}{d_{ij}}\right) \\ &= -\sum_{\substack{\boldsymbol{x}_i \in \mathscr{D}_{\mathcal{S}} \\ \boldsymbol{x}_j \in \mathscr{D}_{\mathcal{T}}}} \|\varphi_n(\boldsymbol{x}_i) - \varphi_n(\boldsymbol{x}_j)\|_2^2 \, \mathbf{W}_p^{(i,j)} = -\text{Tr}(\boldsymbol{\Phi} \mathbf{B} \boldsymbol{\Phi}^\top) \end{aligned} \tag{21}$$

where

$$\mathbf{W}_p^{(i,j)} = \begin{cases} \frac{1}{2} + \frac{\epsilon^2}{2d_{ij}^2} - \frac{\epsilon}{d_{ij}}, & \text{if } d_{ij} < \epsilon \text{ and } \ell_i = \ell_j \\ & \text{and } \mathscr{D}_i \neq \mathscr{D}_l \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

and $\mathbf{B}$ is the graph Laplacian matrix associated with the corresponding weight matrix $\mathbf{W}_p$. Note that the weight matrix of Eq. (22) constitutes an ϵ-distance margin rule for graph embedding. The partial similarity and dissimilarity losses can thus be expressed using graph Laplacian matrices encoding the within-class and between-class relations. Combining Eqs. (19) and (21), we see that the contrastive semantic alignment loss of CCSA is equivalent to:

$$\mathcal{L}_{\text{CSA}} = \text{Tr}\Big(\boldsymbol{\Phi} \mathbf{L} \boldsymbol{\Phi}^\top - \lambda \boldsymbol{\Phi} \mathbf{B} \boldsymbol{\Phi}^\top\Big) \tag{23}$$

which is equivalent to the *trace difference problem* in Eq. (10) used in the DAGE framework. While CCSA employs a value of $\lambda = 1$, one can also determine an optimised value for λ.

2) Domain Adaptation using Stochastic Neighborhood Embedding: Following the procedure outlined above, it is straightforward to show that d-SNE [11] can also be viewed as a graph embedding. For each target sample, the domain adaptation loss term of d-SNE penalises the furthest distance to a within-class source sample, and encourages the distance for the closest between-class to source sample to be maximised:

$$\mathcal{L}_{d\text{-SNE}} = \sum_{\boldsymbol{x}_j \in \mathscr{D}_{\mathcal{T}}} \max_{\substack{\boldsymbol{x}_i \in \mathscr{D}_{\mathcal{S}} \\ \ell_i = \ell_j}} \{a | a \in d_{ij}^2\} - \min_{\substack{\boldsymbol{x}_i \in \mathscr{D}_{\mathcal{S}} \\ \ell_i \neq \ell_j}} \{b | b \in d_{ij}^2\} \tag{24}$$

We can readily express this using the trace difference formulation:

$$\mathcal{L}_{d\text{-SNE}} = \text{Tr}\Big(\boldsymbol{\Phi} \mathbf{L} \boldsymbol{\Phi}^\top - \lambda \boldsymbol{\Phi} \mathbf{B} \boldsymbol{\Phi}^\top\Big) \tag{25}$$

with $\lambda = 1$ and $\mathbf{L}$ and $\mathbf{B}$ being the Graph Laplacian matrices corresponding to the following weight matrices:

$$\mathbf{W}^{(i,j)} = \begin{cases} 1, & \text{if } d_{ij} = \max\limits_{\boldsymbol{x}_k \in \mathscr{D}_{\mathcal{S}}} \{a \mid a \in d_{kj}\} \\ & \text{and } \ell_j = \ell_i = \ell_k \text{ and } \mathscr{D}_i \neq \mathscr{D}_j \\ 0, & \text{otherwise,} \end{cases} \tag{26}$$

$$\mathbf{W}_p^{(i,j)} = \begin{cases} 1, & \text{if } d_{ij} = \min\limits_{\boldsymbol{x}_k \in \mathscr{D}_{\mathcal{S}}} \{b \mid b \in d_{kj}\} \\ & \text{and } \ell_j \neq \ell_i = \ell_k \text{ and } \mathscr{D}_i \neq \mathscr{D}_j \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

Because only a single edge is specified for each source sample per graph Laplacian, it is worth noting that the resulting graph connectivity for d-SNE is highly dependent on the batch size used during optimisation. Small batch sizes will result in more densely connected graphs than large batch sizes.

3) Neural Embedding Matching: NEM [12] extends the contrastive loss of CCSA with an additional term designed to maintain the neighbour relationship of target data throughout the feature embedding:

$$\mathcal{L}_{\text{NEM}} = \mathcal{L}_{\text{CSA}} + \nu \mathcal{L}_{\text{neighbour}} \tag{28}$$

Here, ν is a hyperparameter weighting the importance of the neighbour matching loss, which is specified as the loss over a neighbourhood graph with edges between each target sample

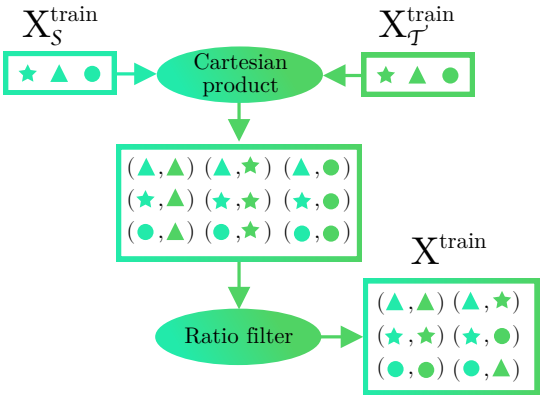
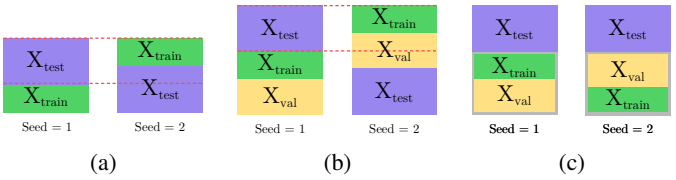


Fig. 2: Cartesian product of two sets, each with three samples. Sample labels are indicated by their shape, while the colour indicates their origin. The Cartesian product produces all pairwise combinations of samples with one sample from each set. A ratio filter (here with a 1:1 ratio) can be used to limit the ratio of same-class samples to different-class samples.



(a) (b) (c)

Fig. 3: (a) Current domain adaptation setup in [10, 11] leads to dependent splits. (b) Drawing a validation does not ensure test set independence. (c) To produce an independent test split, an initial fixed train-rest split should be made followed by train-val splits for each experimental run.

i and its k nearest neighbours $\mathcal{N}(i)$ in the original feature space:

$$\mathcal{L}_{\text{neighbour}} = \sum_{\substack{\boldsymbol{x}_i \in \mathscr{D}_\mathcal{T} \\ \boldsymbol{x}_j \in \mathcal{N}(i)}} \|\varphi_n(\boldsymbol{x}_i) - \varphi_n(\boldsymbol{x}_j)\|_2 \, \kappa_{\text{RBF}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad (29)$$

where $\kappa_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 / 2\sigma^2\right)$ is the Radial Basis Function kernel used to assign a weight to the edge between any pair of vertices. To express the NEM loss in terms of a graph embedding, the neighbour term can be incorporated into the similarity weight matrix by extending the encoding rule from Eq. (20):

$$\mathbf{W}^{(i,j)} = \begin{cases} \nu \frac{\kappa_{\text{RBF}}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{d_{ij}}, & \text{if } j \in \mathcal{N}(i) \text{ and } \mathscr{D}_i = \mathscr{D}_j = \mathscr{D}_\mathcal{T} \\ \frac{1}{2}, & \text{if } \ell_i = \ell_j \text{ and } \mathscr{D}_i \neq \mathscr{D}_l \\ 0, & \text{otherwise,} \end{cases} \quad (30)$$

where ν is a hyper-parameter weighting the influence of the neighbour term. The penalty weight matrix for NEM is the same as for CCSA in Eq. (22) and the final graph embedding problem is a trace difference problem as in Eqs. (23) and (25).

V. Experiments and Results

In this section, we describe the experimental setup that is normally used to evaluate and compare supervised Domain Adaptation methods. We showcase issues related to non-exclusive use of data in model selection and testing phases and we describe how the evaluation process can be improved by proposing a new experimental setup. Finally, we conduct experiments on the Office31 and MINST-USPS datasets using our new experimental setup.

A. Datasets

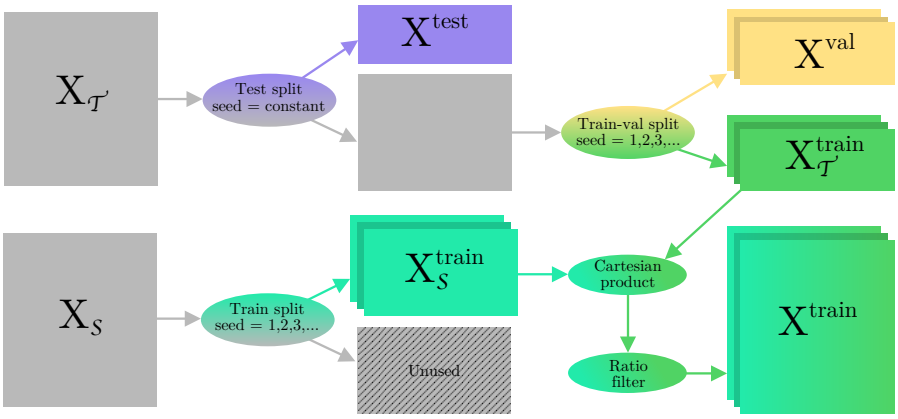
The Office31 dataset [33] contains images of 31 object classes found in the modern office. It has three visual domains: Amazon ($\mathcal{A}$) consists of 2.817 images found on the e-commerce site `www.amazon.com`. These images are generally characterised by their white background and studio-lighting conditions. DSLR ($\mathcal{D}$) contains 498 high resolution images taken using a digital single-lens reflex camera. Here, multiple photos are taken of each object in an office setting. Finally, Webcam ($\mathcal{W}$) has 795 images captured using a cheap web-camera. The objects photographed are the same as for DSLR, but the images in this case are low-resolution and suffer from visual artefacts such as colour imbalances and optical distortion. A sample of the Office31 images is shown in Fig. 5.

The MNIST [34] and USPS [35] datasets contain handwritten digits from 0 to 9 captured in grayscale. MNIST consists of 70,000 images with a 28×28 resolution, and USPS has 11,000 images in a 16×16 format.

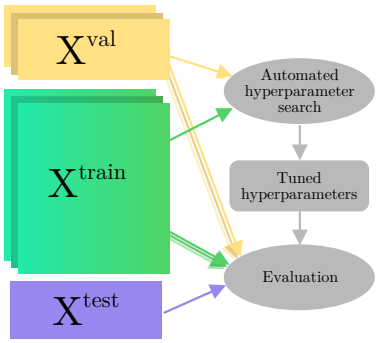
B. Traditional Experimental Setup

The experimental setup used to evaluate the performance of Domain Adaptation methods, e.g. [10, 11], is as follows: A number of samples of each class are drawn from the source domain, and a few samples per class are drawn from the target domain to be used for training. For instance, in experiments using the Amazon data as source domain and the Webcam data as target domain, the number of samples per class forming the training set is equal to twenty and three, respectively. The remaining target data is used for testing. The sampled data from both source and target domains are paired up as the Cartesian product of the two sets, producing as the resulting dataset all combinations of two samples from either domain. To limit the size and redundancy, the dataset is filtered to have a predefined ratio of same-class samples (where both samples in a pair have the same label) to different-class samples. This ratio is commonly set equal to 1:3. An illustration of this is found in Fig. 2.

This combined dataset is then used to train a model with a Domain Adaptation technique e.g. using the two stream architecture as illustrated in Fig. 1. The final evaluation is conducted on the test set coming from the target domain. Because very few unique samples from the target domain are used for training in each experiment, the results will usually vary significantly between runs and will depend on the random seed used for creating the training and test splits. Therefore, each experiment is repeated multiple times, each time with a new seed value, and the mean accuracy alongside the standard deviation over the runs is reported. The absence of validation data on each experiment has the risk of performing



(a) Data preparation procedure. Test data is a constant subset of target data, whereas training and validation data are sampled with different seeds for each experiment. Training data is the Cartesian product of training samples from target and source domain, filtered to have a predefined ratio of same-class to different class pairs. Here, ovals represent operations and rectangles represent data.



(b) Automated hyperparameter search is performed using a single train-validation split, producing the tuned hyperparameters to be used for evaluation with other splits.

Fig. 4: Rectified experimental setup

model selection (including hyper-parameter search) based on the performance on the test data. One could try to avoid the problem by performing model selection and hyper-parameter search using training/test splits from seed values which are not used for the final training/test splits. This, however, is not enough to guarantee that the test performance generalises to unseen data, since it is probable that test data is used for model selection and hyper-parameter search, as illustrated in Fig. 3.

C. Rectified Experimental Setup

To avoid the above described issues of the experimental setup used in evaluating the performance of Domain Adaptation methods, we need to conduct our sampling in two steps: First, we need to define the data in the target domain that will be used for evaluating the performance of the Domain Adaptation method in all the runs. The remaining data in the target domain will be used to form the training and validation sets in the target domain in different runs. This can be done exactly as described in Section V-B: We draw few samples from the source domain and the training set of the target domain, and combine them using the Cartesian Product with an optional ratio for filtering. This way, we ensure that independent test data is used for method evaluation, and a validation set is available for model selection and hyper-parameter search. This data splitting procedure is illustrated in Fig. 4a.

D. Results

In our experiments on the Office31 dataset, we used a model consisting of the convolutional layers of a VGG16 [13] network pretrained on ImageNet [36] with randomly initialised dense layers of 1024 and 128 neurons, respectively, as done in [10, 11]. This network is subsequently fine-tuned on all source data (FT-Source). We found a gradual-unfreeze procedure [37], where four pretrained layers are unfrozen each time the model converges, to work well.

TABLE I: Employed hyper-parameter search space.

Hyper-Parameter	Lower	Upper	Prior
Learning Rate	10^{-6}	0.1	Log-Uniform
Learning Rate Decay	10^{-7}	0.01	Log-Uniform
Momentum	0.5	0.99	Inv Log-Uniform
Dropout	0.1	0.8	Uniform
L2 Regularisation	10^{-7}	10^{-3}	Log-Uniform
Batch Norm	`False`	`True`	Uniform
Margin, ϵ §	10^{-3}	10	Log-Uniform
No. Unfrozen Base-Layers ¶	0	16	Uniform
DA-CE Loss Ratio, $\frac{\beta+\gamma}{1+\beta+\gamma}$	0.01	0.99	Uniform
$\mathcal{S}$-$\mathcal{T}$ CE Loss Ratio, $\frac{\beta}{\beta+\gamma}$	0.0	1.0	Uniform

§Only relevant for CCSA and d-SNE.
¶Only relevant for the experiments in Office31 dataset.

We follow the experimental procedure described in Section V-C. After first splitting off 30% of the target data to form the test set, we create the training set using twenty source samples per class for the Amazon domain, and eight source samples per class for DSLR and Webcam. From the target domain, three samples per class are drawn in each case. The remaining target data is used as a validation set[3]. Thus, we employ the same number of samples for training as in the traditional split [10, 11, 38], but ensure an independent test split as well as a well-defined validation split. The model is duplicated across two streams with shared weights as depicted in Fig. 1 and trained on the combined training data, with one domain entering each stream. This experiment is performed for all six combinations of source and target domain in $\{\mathcal{A}, \mathcal{D}, \mathcal{W}\}$, and each combination is run five times using different seeds. We re-implemented CCSA and d-SNE using their publicly available source code and included them in our experiments. Prior to executing the five runs, an independent hyper-parameter search on the space summarised in Table I was conducted for each method using Bayesian Optimisation

[3]Splits are available at www.github.com/lukashedegaard/office31

TABLE II: Office-31 macro average classification accuracy (%) using a VGG16 network that was pretrained on ImageNet. The results are reported as the mean and standard deviation across five runs. Top rows: The published results by the respective authors. Bottom rows: Our results using the revised experimental methodology.

	$\mathcal{A} \rightarrow \mathcal{D}$	$\mathcal{A} \rightarrow \mathcal{W}$	$\mathcal{D} \rightarrow \mathcal{A}$	$\mathcal{D} \rightarrow \mathcal{W}$	$\mathcal{W} \rightarrow \mathcal{A}$	$\mathcal{W} \rightarrow \mathcal{D}$	Avg.
FT-Source [10]	62.3 ± 0.8	61.2 ± 0.9	58.5 ± 0.8	80.1 ± 0.6	51.6 ± 0.9	95.6 ± 0.7	68.20
CCSA [10]	89.0 ± 1.2	88.2 ± 1.0	71.8 ± 0.5	96.4 ± 0.8	72.1 ± 1.0	97.6 ± 0.4	85.80
FT-Source [11]	62.4 ± 0.4	61.5 ± 0.8	48.9 ± 1.0	82.2 ± 1.4	47.5 ± 0.9	90.4 ± 1.0	65.49
d-SNE [11]	91.4 ± 0.2	90.1 ± 0.1	71.1 ± 0.2	97.1 ± 0.1	71.7 ± 0.4	97.5 ± 0.2	86.49
FT-Source	66.6 ± 3.0	59.8 ± 2.1	42.8 ± 5.2	92.3 ± 2.8	44.0 ± 0.7	98.5 ± 1.2	67.35
FT-Target	71.4 ± 2.0	74.0 ± 4.9	56.2 ± 3.6	95.9 ± 1.2	50.2 ± 2.6	99.1 ± 0.8	74.46
CCSA	$\mathbf{86.4 \pm 2.5}$	$\mathbf{84.5 \pm 2.1}$	$\mathbf{65.5 \pm 1.2}$	97.5 ± 0.9	60.8 ± 1.5	98.4 ± 1.0	82.17
d-SNE	84.7 ± 1.3	82.3 ± 2.4	65.1 ± 0.9	$\mathbf{98.2 \pm 0.4}$	59.9 ± 1.6	$\mathbf{99.7 \pm 0.4}$	81.63
DAGE-LDA	85.4 ± 2.6	84.3 ± 1.7	64.9 ± 1.2	98.0 ± 0.3	$\mathbf{65.5 \pm 1.2}$	98.7 ± 0.5	**82.77**

TABLE III: MNIST $\rightarrow$ USPS classification accuracy (%) for a varying number of available target samples per class, and 200 source samples per class. The mean and standard deviation is reported across ten runs. Top rows: The results published in other works Bottom rows: Our results using the revised experimental methodology.

Samples/class	1	3	5	7	Avg.
CCSA [10]	85.0	90.1	92.4	92.9	90.10
d-SNE [11]	92.9	93.6	95.1	96.1	94.43
CCSA [12]	69.0	78.8	83.4	85.2	79.08
NEM [12]	72.2	86.6	91.4	91.8	85.49
CCSA	$\mathbf{89.1 \pm 1.1}$	91.2 ± 0.9	$\mathbf{93.8 \pm 0.4}$	$\mathbf{94.3 \pm 0.4}$	92.07
d-SNE	88.3 ± 1.7	91.4 ± 1.2	93.1 ± 0.5	93.6 ± 0.6	91.61
DAGE-LDA	88.8 ± 1.8	$\mathbf{92.4 \pm 0.5}$	93.4 ± 0.4	94.1 ± 0.3	**92.15**

Fig. 5: Samples from Office31 (Amazon, DSLR, Webcam) as well as MNIST and USPS.

with the Expected Improvement acquisition function [39] given 100 trials. For the final tests, we used data augmentation with random modifications of colour hue and saturation, image brightness and contrast, as well as rotation and zoom. For a fair comparison, all hyper-parameter tuning and tests are performed with the exact same computational budget and data available for all methods tested.

The results for Office31 are shown in Table II. Comparing the CCSA and d-SNE results of the traditional experimental setup with the revised one, we see that the achieved macro accuracy is generally lower: -3.63% for CCSA and -4.86% for d-SNE. This is in-line with our expectations, and may signify that the results of [10] and [11] did suffer from generalisation issues as described in Section V-B. Comparing CCSA, d-SNE, and DAGE-LDA in the rectified experimental setup, we see that there is surprisingly little difference in performance. While DAGE-LDA only outperforms the other methods on a single adaptation ($\mathcal{W} \rightarrow \mathcal{A}$), it has the highest average score across all six adaptations. CCSA performs next best, and d-SNE performs worst of the three. This suggests, that the higher accuracy originally reported in [11] as compared to [10] may be due to better hyper-parameter optimisation rather than a better domain adaptation loss. In addition to the results of CCSA, d-SNE and DAGE-LDA on the revised experimental setup, we have included our results for fine-tuning the FT-Source model on target data (FT-Target). While improving upon the FT-Source baseline, it scored significantly lower than DAGE-LDA, CCSA and d-SNE, reaffirming the utility of the domain adaptation methods.

For our experiments in the MNIST to USPS domain adap-

tation problem, we used a network architecture which has two streams with shared weights, with two convolutional layers containing 6 and 16 5×5 filters respectively, max-pooling, and two dense layers of size 120 and 84 prior to the classification layer. This architecture is the same as the one used in [10]. We trained the network from random initialisation using 2,000 randomly sampled images per class from MNIST (source) and a varying number of USPS (target) samples per class. Experiments using 1, 3, 5 and 7 target samples per class were conducted and each experiment was repeated 10 times. Here, we used the predefined test-train splits from TorchVision Datasets, sampling the training and validation data from the train split. Though our implementation uses Tensorflow, the datasets were made compatible by using the Dataset Ops library[4]. Aside from following the rectified sampling, the experiments use the procedure from [10, 11, 40]. Prior to conducting the final experiment runs, a hyper-parameter search was conducted using the same settings as for Office31, and for testing, similar data augmentation was employed. The results obtained by running the experiments are shown in Table III. Comparing CCSA, d-SNE and DAGE-LDA, we find the same trend as for the Office31 experiments: DAGE-LDA has the highest average accuracy, closely followed by CCSA and then d-SNE. While [12] did not conduct experiments on the Office31 dataset nor published the source code for NEM, it included experiments on the MNIST to USPS domain adaptation problem, the findings of which we repeated in Table III. Their results were substantially lower than reported in both [10], [11], and our work.

VI. Conclusion

In this paper, we proposed a unified framework for supervised Domain Adaptation by viewing the Domain Adaptation process as a graph embedding problem. We showed that existing state-of-the-art methods correspond to specific instantiations of the proposed Domain Adaptation via Graph Embedding (DAGE) framework, and we proposed a simple instantiation of DAGE that performs on par with the current state-of-the-art on the standard benchmark datasets Office31 and MNIST-USPS. Moreover, we highlighted some generalisation and reproducibility issues related to the experimental setup commonly used to evaluate the performance of Domain Adaptation methods and proposed a new experimental setup for more accurately assessing and comparing different supervised DA methods. Alongside our source code, we made the training/test splits for the standard benchmark datasets Office31 and MNIST-USPS separately available to facilitate fair comparisons of supervised Domain Adaptation methods in future research.

Acknowledgement

This work has received partial funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

[4]Splits are available at www.github.com/lukashedegaard/mnist-usps

References

[1] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," in *International Conference on Machine Learning*, vol. 70. JMLR. org, 2017, pp. 233–242.

[2] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[3] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, p. 9, 2016.

[4] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 2010, pp. 242–264.

[5] J. Raitoharju, E. Riabchenko, K. Meissner, I. Ahmad, A. Iosifidis, M. Gabbouj, and S. Kiranyaz, "Data enrichment in fine-grained classification of aquatic macroinvertebrates," in *ICPR 2nd Workshop on Computer Vision for Analysis of Underwater Imagery*, 2016, pp. 43–48.

[6] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[7] L. Hedegaard, O. A. Sheikh-Omar, and A. Iosifidis, "Supervised domain adaptation using graph embedding," 2020.

[8] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, 2006.

[9] G. Cao, A. Iosifidis, and M. Gabbouj, "Generalized multi-view embedding for visual recognition and cross-modal retrieval," *IEEE Transactions on Cybernetics*, vol. 48, no. 9, pp. 2542–2555, 2018.

[10] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, "Unified deep supervised domain adaptation and generalization," in *IEEE International Conference on Computer Vision*, 2017, pp. 5715–5725.

[11] X. Xu, X. Zhou, R. Venkatesan, G. Swaminathan, and O. Majumder, "d-sne: Domain adaptation using stochastic neighborhood embedding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2497–2506.

[12] Z. Wang, B. Du, and Y. Guo, "Domain adaptation with neural embedding matching," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2019.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[14] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2066–2073.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM International Conference on Multimedia*, 2014, pp. 675–678.

[16] J. Li, K. Lu, Z. Huang, L. Zhu, and H. T. Shen, "Heterogeneous domain adaptation through progressive alignment," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1381–1391, 2019.

[17] F. Liu, G. Zhang, and J. Lu, "Heterogeneous domain adaptation: An unsupervised approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, 2020.

[18] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Heterogeneous domain adaptation via nonlinear matrix factorization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 3, pp. 984–996, 2020.

[19] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in

International Conference on Machine Learning, 2007, pp. 759–766.

[20] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1735–1742.

[21] Y. Chen, S. Song, S. Li, and C. Wu, "A graph embedding framework for maximum mean discrepancy-based domain adaptation algorithms," *IEEE Transactions on Image Processing*, vol. 29, pp. 199–213, 2019.

[22] G. Cai, Y. Wang, L. He, and M. Zhou, "Unsupervised domain adaptation with adversarial residual transform networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2019.

[23] Y. Jia, F. Nie, and C. Zhang, "Trace ratio problem revisited," *IEEE Transactions on Neural Networks*, vol. 20, no. 4, pp. 729–735, 2009.

[24] A. Iosifidis, A. Tefas, and I. Pitas, "On the optimal class representation in linear discriminant analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 9, pp. 1491–1497, 2013.

[25] T. Diethe, D. Hardoon, and J. Shawe-Taylor, "Multiview fsher discriminant analysis," in *Neural Information Processing Systems*, 2008.

[26] S. Wold, A. Ruhe, H. Wold, and W. Dunn, "The collinearity problem in linear regression: The partial least squares (pls) approach to generalized inverses," *SIAM Journal on Scientic and Statistical Computing*, vol. 5, no. 3, pp. 735–743, 1984.

[27] G. Andrew, R. Arona, J. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *International Conference on Machine Learning*, 2013.

[28] M. Kan, S. Shan, H. Zhang, S. Lao, and X. Chen, "Multi-view discriminant analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 188–194, 2016.

[29] G. Cao, A. Iosifidis, and Gabbouj, "Multi-view nonparametric discriminant analysis for image retrieval and recognition," *IEEE Signal Processing Letters*, vol. 24, no. 10, pp. 1537–1541, 2017.

[30] G. Cao, A. Iosifidis, M. Gabbouj, V. Raghavan, and R. Gottumukkala, "Deep multi-view learning to rank," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[31] V. Y. Pan and Z. Q. Chen, "The complexity of the matrix eigenproblem," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 1999, pp. 507–516.

[32] D. Das and S. G. Lee, "Graph matching and pseudo-label guided deep unsupervised domain adaptation," in *International Conference on Artificial Neural Networks*, 2018, pp. 342–352.

[33] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European Conference on Computer Vision*, 2010, pp. 213–226.

[34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[35] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed., 1990, pp. 396–404.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[37] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 328–339.

[38] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *IEEE International Conference on Computer Vision*, 2015, pp. 4068–4076.

[39] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *CoRR*, vol. abs/1012.2599, 2010.

[40] B. Fernando, T. Tommasi, and T. Tuytelaars, "Joint cross-domain classification and subspace learning," *CoRR*, vol. abs/1411.4491, 2014.