Domain Generalization and Adaptation using Low Rank Exemplar SVMs

Wen Li, Zheng Xu, Dong Xu, Dengxin Dai, and Luc Van Gool

Abstract—Domain adaptation between diverse source and target domains is a challenging research problem, especially in the real-world visual recognition tasks where the images and videos consist of significant variations in viewpoints, illuminations, qualities, etc. In this paper, we propose a new approach for domain generalization and domain adaptation based on exemplar SVMs. Specifically, we decompose the source domain into many subdomains, each of which contains only one positive training sample and all negative samples. Each subdomain is relatively less diverse, and is expected to have a simpler distribution. By training one exemplar SVM for each subdomain, we obtain a set of exemplar SVMs. To further exploit the inherent structure of source domain, we introduce a nuclear-norm based regularizer into the objective function in order to enforce the exemplar SVMs to produce a low-rank output on training samples. In the prediction process, the confident exemplar SVM classifiers are selected and reweigted according to the distribution mismatch between each subdomain and the test sample in the target domain. We formulate our approach based on the logistic regression and least square SVM algorithms, which are referred to as low rank exemplar SVMs (LRE-SVMs) and low rank exemplar least square SVMs (LRE-LSSVMs), respectively. A fast algorithm is also developed for accelerating the training of LRE-LSSVMs. We further extend Domain Adaptation Machine (DAM) to learn an optimal target classifier for domain adaptation, and show that our approach can also be applied to domain adaptation with evolving target domain, where the target data distribution is gradually changing. The comprehensive experiments for object recognition and action recognition demonstrate the effectiveness of our approach for domain generalization and domain adaptation with fixed and evolving target domains.

Index Terms—latent domains, domain generalization, domain adaptation, exemplar SVMs.

1 Introduction

DOMAIN adaptation techniques, which aim to reduce the domain distribution mismatch when the training and testing samples come from different domains, have been successfully used for a broad range of vision applications such as object recognition and video event recognition [1], [2], [3], [4], [5], [6]. As a related research problem, domain generalization differs from domain adaptation, because it assumes that the target domain samples are not available during the training process. Without focusing on the generalization ability on the specific target domain, domain generalization techniques aim to better classify testing data from any unseen target domain [7], [8]. Please refer to Section 2 for a brief review of existing domain adaptation and domain generalization techniques.

For visual recognition, most existing domain adaptation methods treat a whole dataset as one domain [1], [2], [3], [4], [5], [6]. However, the real world visual data is usually quite diverse. The images and videos could be captured from arbitrary viewpoints, under different illuminations, and using different equipments. In other words, the distribution of visual domain is complex, thus it is challenging to reduce the distribution mismatch between different domains.

Several recent works proposed to partition the source domain into multiple hidden domains [9], [10]. While those works showed the benefits to exploit latent domains in the source data for improving domain adaptation performance, it is a non-trivial task to discover the characteristic latent domains by explicitly partitioning the training samples into multiple clusters because many factors (*e.g.*, pose and illumination) overlap and interact in images and videos in complex ways [10].

In this work, we propose a new approach for domain generalization and adaptation by exploiting the intrinsic structure of positive samples from latent domains without explicitly partitioning the training samples into multiple clusters/domains. Our work builds up the recent ensemble learning method exemplar SVMs, in which we aim to train a set of exemplar classifiers with each classifier learnt by using one positive training sample and all negative training samples. Under the context of domain adaptation, the training set for learning each exemplar classifier (*i.e.*, one positive training sample and all negative training samples) can be regarded as one *subdomain*, which is relatively less diverse and with a simpler distribution. So each learnt exemplar classifier is expected to have good generalization capability for one certain data distribution. When predicting the test sample from an arbitrary distribution, those exemplar classifiers are then combined to properly fit the target domain distribution and produce good prediction results.

To further enhance the discriminative capability of the learnt exemplar classifiers, we exploit the intrinsic latent structure in the source domain, as inspired by the recent latent domain discovery works [9], [10]. In particular, positive samples may come from multiple latent domains characterized by different factors. For the positive samples

- *W. Li, D. Dai and L. Van Gool are with the Computer Vision Laboratory, ETH Zürich, Switzerland.*
  E-mail: {liwen, dai, vangool}@vison.ee.ethz.ch
- *Z. Xu is with the department of Computer Science at the University of Maryland, College Park.*
  Email: xuzhustc@gmail.com
- *D. Xu is with the School of Electrical and Information Engineering, The University of Sydney, Sydney, NSW 2006, Australia.*
  e-mail: dongxudongxu@gmail.com

captured under similar conditions (*e.g.*, frontal-view poses), their predictions from each exemplar classifier are expected to be similar to each other. Using the predictions from all the exemplar classifiers as the feature of each positive sample, we assume the prediction matrix consisting of the features of all positive samples should be low-rank in the ideal case. Based on this assumption, we formulate a new objective function by introducing a nuclear norm based regularizer on the prediction matrix into the objective function of exemplar SVMs in order to learn a set of more robust exemplar classifiers for domain generalization and domain adaptation. Therefore, we refer to our new approach as Low Rank Exemplar SVMs, or LRE-SVMs in short.

During the testing process, we can directly use the whole or a selected set of learnt exemplar classifiers for the domain generalization task when the target domain samples are not available during the training process. For the domain adaptation problem where the unlabeled target domain data is available, we propose an effective method to re-weight the selected set of exemplar classifiers based on the Maximum Mean Discrepancy (MMD) criterion, and further extend the Domain Adaptation Machine (DAM) method to learn an optimal target classifier.

In our preliminary conference paper [11], we have formulated our LRE-SVMs approach by using the logistic regression method to learn each exemplar classifier, which is computational expensive as we need to learn one classifier for each positive training sample. To improve the efficiency in the training process, in this paper, we formulate a new objective function based on the least square SVM method, and develop a fast algorithm for learning each exemplar classifier. For the testing process, we also extend our method to address the domain adaptation problem with evolving target domain, where the test data comes one by one, and the target data distribution may change gradually [12]. We conduct comprehensive experiments for various visual recognition tasks using three benchmark datasets, and the results clearly demonstrate the effectiveness of our approach for domain generalization, and domain adaptation with fixed and evolving target domains.

2 Related Work

Traditional domain adaptation methods can be roughly categorized into feature based approaches and classifier based approaches. The feature based approaches aim to learn domain invariant features for domain adaptation. Kulis *et al.* [1] proposed a distance metric learning method to reduce domain distribution mismatch by learning asymmetric nonlinear transformation. Gopalan *et al.* [2] and Gong *et al.* [3] proposed two domain adaptation methods by interpolating intermediate domains. To reduce domain distribution mismatch, some recent approaches learnt a domain invariant subspace [13] or aligned two subspaces from both domains [14].

Moreover, with the advance of deep learning techniques, a few works have also proposed to learn domain invariant features based on the convolutional neural network (CNN) for image recognition [15], [16], [17]. Long *et al.* proposed to learn CNN features while minimizing the MMD of the features between two domains [18]. Ganin and Lempitsky proposed to simultaneously minimize the classification loss and maximize the domain confusion with a CNN for unsupervised domain adaptation [16], while Tzeng *et al.* employed soft-label generated from source domain to train the CNN for target samples [15]. Li *et al.* [17] proposed to apply the batch normalization method for domain adaptation with the CNN.

Our work is different from those works in two folds. First, those works focus on learning domain-adaptive features for a specifical target domain, while our approaches can be applied for domain generalization problem where the target domain is unseen during the training process. Second, our approaches and those methods are at different stages for visual recognition. Instead of learning feature representations, our approaches aim to learn robust classifiers, thus the learnt feature representations from their method can be used as input of our approaches if there still exists domain distribution mismatch.

Classifier based approaches directly learn the classifiers for domain adaptation, among which SVM based approaches are the most popular ones. Duan *et al.* [4] proposed Adaptive MKL based on multiple kernel learning (MKL), and a multi-domain adaptation method by selecting the most relevant source domains [19]. The work in [20] developed an approach to iteratively learn the SVM classifier by labeling the unlabeled target samples and simultaneously removing some labeled samples in the source domain. For the unsupervised domain adaptation scenario, training domain adaptive classifiers by reweighing the source training samples has been widely exploited for reducing the distribution mismatch between the source and target domains. Various approaches have been proposed to learn the weights (*a.k.a.* density ratios) based on different strategies and criteria in the literature, which includes Kernel Mean Machting (KMM) [21], the MaxNet based methods [22], Kullback-Leibler Importance Estimation Procedure (KLIEP) [23], the Logistic Regression based method (LogReg) [24], and Large Scale KLIEP (LS-KLIEP) [25].

There are a few works specifically designed for domain generalization. Muandet *et al.* proposed to learn domain invariant feature representations [7]. Given multiple source datasets/domains, Khosla *et al.* [8] proposed an SVM based approach, in which the learnt weight vectors that are common to all datasets can be used for domain generalization. Ghifary *et al.* [26] proposed an approach for learning domain-invariant features based on the auto-encoder method. Inspired by our preliminary conference work [11], Niu *et al.* [27] proposed an approach for multi-view domain generalization.

Our work is more related to the recent works for discovering latent domains [9], [10]. In [9], a clustering based approach is proposed to divide the source domain into different latent domains. In [10], the MMD criterion is used to partition the source domain into distinctive latent domains. However, their methods need to decide the number of latent domains beforehand. In contrast, our method exploits the low-rank structure from latent domains without requiring the number of latent domains. Moreover, we directly learn the exemplar classifiers without partitioning the data into clusters/domains.

Our work builds up the recent work on exemplar

SVMs [28]. In contrast to [28], we introduce a nuclear norm based regularizer on the prediction matrix in order to exploit the low-rank structure from latent domains for domain generalization. Recently, inspired by our preliminary conference work [11], Xu *et al.* [29] also employed nuclear norm to exploit the low-rank structure for sub-categorization. In multi-task learning, the nuclear norm based regularizer is also introduced to enforce the related tasks share similar weight vectors when learning the classifiers for multiple tasks [30], [31]. However, their works assume the training and testing samples come from the same distribution without considering the domain generalization or domain adaptation tasks. Moreover, our regularizer is on the prediction matrix such that we can better exploit the structure of positive samples from multiple latent domains.

Domain adaptation with changing distribution also attracts attentions from computer vision researchers. Lee *et al.* [32] studied the change of car styles through the past decades. Hoffman *et al.* [12] studied the domain adaptation problem with evolving target domain, where the distribution of target domain is gradually changing. They proposed an approach to learn the subspaces along manifold for domain adaptation. Lampert [33] studied the time-varying data problem, where the source domain data distribution is varying, and proposed a method for predicting the data in the future.

3 Low rank Exemplar SVMs

In this section, we introduce the formulation of our low rank exemplar SVMs as well as the optimization algorithm. For ease of presentation, in the remainder of this paper, we use a lowercase/uppercase letter in boldface to represent a vector/matrix. The transpose of a vector/matrix is denoted by using the superscript $'$. $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{m\times n}$ defines a matrix $\mathbf{A}$ with a_{ij} being its (i,j)-th element for $i = 1,\dots,m$ and $j = 1,\dots,n$. The element-wise product between two matrices $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{m\times n}$ and $\mathbf{B} = [b_{ij}] \in \mathbb{R}^{m\times n}$ is defined as $\mathbf{C} = \mathbf{A} \circ \mathbf{B}$, where $\mathbf{C} = [c_{ij}] \in \mathbb{R}^{m\times n}$ and $c_{ij} = a_{ij}b_{ij}$.

3.1 Exemplar SVMs

The exemplar SVMs model was first introduced in [28] for object detection. In exemplar SVMs, each exemplar classifier is learnt by using one positive training sample and all the negative training samples. Let $\mathcal{X}_s = \mathcal{X}^+ \cup \mathcal{X}^-$ denote the set of training samples, in which $\mathcal{X}^+ = \{\mathbf{x}_1^+,\dots,\mathbf{x}_n^+\}$ is the set of positive training samples, and $\mathcal{X}^- = \{\mathbf{x}_1^-,\dots,\mathbf{x}_m^-\}$ is the set of negative training samples. Each training sample $\mathbf{x}^+$ or $\mathbf{x}^-$ is a d-dimensional column vector, *i.e.*, $\mathbf{x}^+, \mathbf{x}^- \in \mathbb{R}^d$. We first develop our LRE-SVMs approach based on the logistic regression method, and then introduce a variant based on the formulation of least square SVMs for improving the efficiency of the training process. Specifically, given any sample $\mathbf{x} \in \mathbb{R}^d$, the prediction function using logistic regression can be written as:

$$p(\mathbf{x}|\mathbf{w}_i) = \frac{1}{1+\exp(-\mathbf{w}_i'\mathbf{x})} \quad , \tag{1}$$

where $\mathbf{w}_i \in \mathbb{R}^d$ is the weight vector in the i-th exemplar classifier trained by using the positive training sample $\mathbf{x}_i^+$ and all negative training samples[1]. By defining a weight matrix $\mathbf{W} = [\mathbf{w}_1,\dots,\mathbf{w}_n] \in \mathbb{R}^{d\times n}$, we formulate the learning problem as follows,

$$\min_{\mathbf{W}} \|\mathbf{W}\|_F^2 + C_1\sum_{i=1}^n l(\mathbf{w}_i,\mathbf{x}_i^+) + C_2\sum_{i=1}^n\sum_{j=1}^m l(\mathbf{w}_i,\mathbf{x}_j^-), \tag{2}$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix, C_1 and C_2 are the tradeoff parameters analogous to C in SVM, and $l(\mathbf{w},\mathbf{x})$ is the logistic loss, which is defined as:

$$l(\mathbf{w}_i,\mathbf{x}_i^+) = \log(1+\exp(-\mathbf{w}_i'\mathbf{x}_i^+)), \tag{3}$$
$$l(\mathbf{w}_i,\mathbf{x}_j^-) = \log(1+\exp(\mathbf{w}_i'\mathbf{x}_j^-)). \tag{4}$$

3.2 Exemplar SVMs for Domain Generalization and Adaptation

Each exemplar classifier is learnt to discriminate a unique exemplar positive training sample from all the negative samples, so the differences between exemplar SVM classifiers represent the differences between positive training samples. In other words, the exemplar classifier also learns the unique property of each exemplar positive training sample. Under the context of domain adaptation, the training data for learning each exemplar SVM classifier (*i.e.*, one positive training sample and all negative training samples) can be considered as a small subdomain. By learning an exemplar classifier for each subdomain, we expect that the classifier encodes not only the corresponding category information of the exemplar positive sample, but also the domain property (*e.g.*, pose, background, illumination, *etc.*) of the exemplar positive training sample. In the testing process, for any given target sample or target domain, we combine the exemplar classifiers with proper weights, and expect the domain properties of training data are also similar to those of target samples, thus producing better predictions by using the combined classifiers (see Section 5.2 for more details).

Formally, let us denote $\mathcal{D}_i^s$ as the underlying distribution of a subdomain formed by $\{\mathbf{x}_i^+, \mathbf{x}_j^-|_{j=1}^m\}$, and denote f_i as the learnt exemplar classifier. We also denote the distribution of the test data as $\mathcal{D}^t$. It has been shown in the previous works [34], [35] that the error of f_i on the test data can be bounded as $err_t(f_i) \le err_s(f_i) + dist(\mathcal{D}_i^s,\mathcal{D}^t) + \Delta$, where $err_t(f_i)$ is the test error, $err_s(f_i)$ is the training error of f_i, $dist(\mathcal{D}_i^s,\mathcal{D}^t)$ is the distance between two distributions under certain measurement, and Δ is a constant term. The distribution distance term $dist(\mathcal{D}_i^s,\mathcal{D}^t)$ plays an important role in the generalization bound for domain adaptation. Given two different exemplar classifiers f_i and f_j, their training sets share the same negative training samples, so the difference between the underlying distributions $\mathcal{D}_i$ and $\mathcal{D}_j$ is from the two exemplar positive training samples $\mathbf{x}_i$ and $\mathbf{x}_j$. This means that $dist(\mathcal{D}_i^s,\mathcal{D}^t)$ (*resp.*, $dist(\mathcal{D}_j^s,\mathcal{D}^t)$) tends to be decided by whether $\mathbf{x}_i$ (*resp.*, $\mathbf{x}_j$) is closer to the distribution of the test data.

The exemplar SVMs can be directly used for domain generalization, as the target domain unlabeled data is not required in the training stage. Generally, in the domain

1. Although we do not explicitly use the bias term in the prediction formulation, in our experiments we append 1 to the feature vector of each training sample.

generalization task, the test data is assumed to come from an arbitrary target domain that is unseen in the training procedure. However, based on the above analysis, the generalization error of exemplar classifiers tends to be huge if the target domain distribution is considerably different from those of all the subdomains (*i.e.*, $dist(\mathcal{D}_i^s, \mathcal{D}^t)$ is huge $\forall i$). For domain adaptation, the traditional domain adaptation methods [1], [2], [3] usually aim to learn a common subspace or an adaptive classifier to handle the domain distribution mismatch. Those methods can be regarded as using the global distribution information from the source and target domains for domain adaptation. However, for visual recognition applications where the training and test data are with large variances, the data distribution of one domain is usually complex, so it might be challenging to match two distributions by using global distribution information. In contrast, our approach focuses on each positive sample by using exemplar SVMs, and learns the local information of each exemplar. Each exemplar classifier encodes certain local distribution information related to this exemplar. So it is more flexible to match a single target sample or even a target domain with different data distributions.

3.3 Low-Rank Exemplar SVMs

A drawback of exemplar SVMs is that training exemplar classifier with only one exemplar positive training sample might be sensitive to the noise. For example, the exemplar classifiers of two images with similar objects might produce different predictions for the same test image. To this end, we consider to discover the intrinsic latent structure in the source training data to improve discriminate capacity of exemplar classifiers, as inspired by the recent latent domain discovery methods [9], [10].

Intuitively, if there are multiple latent domains in the training data (*e.g.*, poses, backgrounds, illuminations, *etc.*), the positive training samples should also come from several latent domains. For the positive samples captured under similar conditions (*e.g.*, frontal-view poses), the prediction from each exemplar classifier is expected to be similar to each other. Using the predictions from all the exemplar classifiers as the feature of each positive sample, we assume the prediction matrix consisting of the predictions of all positive samples should be low-rank in the ideal case. Formally, we denote the prediction matrix as $\mathbf{G}(\mathbf{W}) = [g_{ij}] \in \mathbb{R}^{n\times n}$, where each $g_{ij} = p(\mathbf{x}_i^+|\mathbf{w}_j)$ is the prediction of the i-th positive training sample by using the j-th exemplar classifier. We also denote the objective of exemplar SVMs in (2) as $J(\mathbf{W}) = \|\mathbf{W}\|_F^2 + C_1\sum_{i=1}^n l(\mathbf{w}_i, \mathbf{x}_i^+) + C_2\sum_{i=1}^n\sum_{j=1}^m l(\mathbf{w}_i, \mathbf{x}_j^-)$. To exploit those latent domains, we thus enforce the prediction matrix $\mathbf{G}(\mathbf{W})$ to be low-rank when we learn those exemplar SVMs, namely, we arrive at the following objective function,

$$\min_{\mathbf{W}} J(\mathbf{W}) + \lambda\|\mathbf{G}(\mathbf{W})\|_*, \tag{5}$$

where we use the nuclear norm based regularizer $\|\mathbf{G}(\mathbf{W})\|_*$ to approximate the rank of $\mathbf{G}(\mathbf{W})$. It has been shown that the nuclear norm is the best convex approximation of the rank function over the unit ball of matrices [36]. However, it is a nontrivial task to solve the problem in (5), because

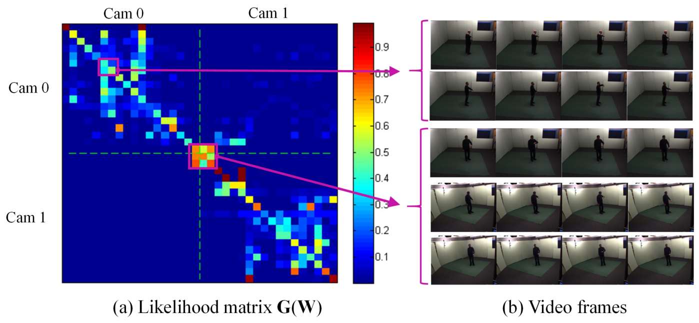


Fig. 1. An illustration of the prediction matrix $\mathbf{G}(\mathbf{W})$, where we observe the block diagonal property of $\mathbf{G}(\mathbf{W})$ in (a). The frames from the videos corresponding to the two blocks with large values in $\mathbf{G}(\mathbf{W})$ are also visually similar to each other in (b).

the last term is a nuclear norm based regularizer on the prediction matrix $\mathbf{G}(\mathbf{W})$ and $\mathbf{G}(\mathbf{W})$ is non-linear w.r.t. $\mathbf{W}$.

To solve the optimization problem in (5), we introduce an intermediate matrix $\mathbf{F} \in \mathbb{R}^{n\times n}$ to model the ideal $\mathbf{G}(\mathbf{W})$ such that we can decompose the last term in (5) into two parts: on one hand, we expect the intermediate matrix $\mathbf{F}$ should be low-rank as we discussed above; on the other hand, we enforce the prediction matrix $\mathbf{G}(\mathbf{W})$ to be close to the intermediate matrix $\mathbf{F}$. Therefore, we reformulate the objective function as follows,

$$\min_{\mathbf{W},\mathbf{F}} J(\mathbf{W}) + \lambda_1\|\mathbf{F}\|_* + \lambda_2\|\mathbf{F} - \mathbf{G}(\mathbf{W})\|_F^2, \tag{6}$$

which can be solved by alternatingly optimizing two subproblems w.r.t. $\mathbf{W}$ and $\mathbf{F}$. Specifically, the optimization problem w.r.t. $\mathbf{W}$ does not contain the nuclear norm based regularizer, which makes the optimization much easier. Also, the nuclear norm based regularizer only depends on the intermediate matrix $\mathbf{F}$ rather than a non-linear term w.r.t. $\mathbf{W}$ (*i.e.*, the prediction matrix $\mathbf{G}(\mathbf{W})$) as in (5), thus the optimization problem w.r.t. $\mathbf{F}$ can be readily solved by using the Singular Value Threshold (SVT) method [37] (see Section 3.4 for the details).

Discussions: The existing latent domain discovery methods [9], [10] aim to explicitly partition the source domain data into several clusters, each corresponding to one latent domain. However, it is nontrivial to determine the number of latent domains, or to cluster the training samples into multiple subsets, because many factors (*e.g.*, pose and illumination) overlap and interact in images and videos in complex ways [10]. In contrast, our proposed approach exploits the latent domain structure in an implicit way based on the low-rank regularizer defined on the prediction matrix.

To better understand the effect of the low-rank regularizer, in Figure 1, we show an example of the learnt prediction matrix $\mathbf{G}(\mathbf{W})$ from the "check watch" category in the IXMAS multi-view dataset by using Cam 0 and Cam 1 as the source domain. After using the nuclear norm based regularizer, we observe that the block diagonal property of the prediction matrix $\mathbf{G}(\mathbf{W})$ in Figure 1(a). In Figure 1(b), we also display some frames from the videos corresponding to the two blocks with large values in $\mathbf{G}(\mathbf{W})$. We observe that the videos sharing higher values in the matrix $\mathbf{G}(\mathbf{W})$ are also visually similar to each other. For example, the first two rows in Figure 1(b) are the videos from similar poses.

More interestingly, we also observe that our algorithm can group similar videos from different views into one block (*e.g.*, the last three rows in Figure 1(b) are the videos from the same actor), which demonstrates it is beneficial to exploit the latent source domains by using our approach.

3.4 Optimization

In this section, we discuss how to optimize the problem in (6). We optimize (6) by iteratively updating $\mathbf{W}$ and $\mathbf{F}$. The two subproblems w.r.t. $\mathbf{W}$ and $\mathbf{F}$ are described in detail as follows.

3.4.1 Update $\mathbf{W}$*:*

When $\mathbf{F}$ is fixed, the subproblem w.r.t. $\mathbf{W}$ can be written as,

$$\min_{\mathbf{W}} J(\mathbf{W}) + \lambda_2 \|\mathbf{G}(\mathbf{W}) - \mathbf{F}\|_F^2, \tag{7}$$

where the matrix $\mathbf{F}$ is obtained at the k-th iteration, and $\mathbf{G}(\mathbf{W})$ is defined as in Section 3.3. We optimize the above subproblem by using the gradient descent technique. Let us respectively define $\mathbf{X}_+ = [\mathbf{x}_1^+, \ldots, \mathbf{x}_n^+] \in \mathbb{R}^{d\times n}$ and $\mathbf{X}_- = [\mathbf{x}_1^-, \ldots, \mathbf{x}_m^-] \in \mathbb{R}^{d\times m}$ as the data matrices of positive and negative training samples, and also denote $H(\mathbf{W}) = \|\mathbf{G}(\mathbf{W}) - \mathbf{F}\|_F^2$. Then, the gradients of the two terms in (7) can be derived as follows,

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = 2\mathbf{W} + C_1\mathbf{X}_+(\mathbf{P}_+ - \mathbf{I}) + C_2\mathbf{X}_-\mathbf{P}_-,$$
$$\frac{\partial H(\mathbf{W})}{\partial \mathbf{W}} = 2\mathbf{X}_+\left(\mathbf{G}(\mathbf{W}) \circ (\mathbf{1}\mathbf{1}' - \mathbf{G}(\mathbf{W})) \circ (\mathbf{G}(\mathbf{W}) - \mathbf{F})\right),$$

where $\mathbf{P}_+ = diag\left(p(\mathbf{x}_i^+|\mathbf{w}_i)\right) \in \mathbb{R}^{n\times n}$ is a diagonal matrix with each diagonal entry being the prediction on one positive sample by using its corresponding exemplar classifier, $\mathbf{P}_- = [p(\mathbf{x}_i^-|\mathbf{w}_j)] \in \mathbb{R}^{m\times n}$ is the prediction matrix on all negative training samples by using all exemplar classifiers, $\mathbf{I} \in \mathbb{R}^{n\times n}$ is an identity matrix, and $\mathbf{1} \in \mathbb{R}^n$ is a vector with all entries being 1.

3.4.2 Update $\mathbf{F}$*:*

When $\mathbf{W}$ is fixed, we calculate the matrix $\mathbf{G} = \mathbf{G}(\mathbf{W})$ at first, then the subproblem w.r.t. $\mathbf{F}$ becomes,

$$\min_{\mathbf{F}} \lambda_1\|\mathbf{F}\|_* + \lambda_2\|\mathbf{F} - \mathbf{G}\|_F^2, \tag{8}$$

which can be readily solved by using the singular value thresholding (SVT) method [37]. Specifically, let us denote the singular value decomposition of $\mathbf{G}$ as $\mathbf{G} = \mathbf{U\Sigma V}'$, where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{n\times n}$ are two orthogonal matrices, and $\mathbf{\Sigma} = diag(\sigma_i) \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing all the singular values. The singular value thresholding operator on $\mathbf{G}$ can be represented as $\mathbf{U}\mathcal{D}(\mathbf{\Sigma})\mathbf{V}'$, where $\mathcal{D}(\mathbf{\Sigma}) = diag((\sigma_i - \frac{\lambda_1}{2\lambda_2})_+)$, and $(\cdot)_+$ is a thresholding operator by assigning the negative elements to be zeros.

3.4.3 Algorithm:

We summarize the optimization procedure in Algorithm 1 and name our method as *Low-rank Exemplar SVMs (LRE-SVMs)*. Specifically, we first initialize the weight matrix $\mathbf{W}$ as $\mathbf{W}^0$, where $\mathbf{W}^0$ is obtained by solving the traditional exemplar SVMs formulation in (2). Then we calculate the prediction matrix $\mathbf{G}(\mathbf{W})$ by applying the learnt classifiers on all positive samples. Next, we obtain the matrix $\mathbf{F}$ by solving the problem in (8) with the SVT method. After that, we use the gradient descent method to update the weight matrix $\mathbf{W}$. The above steps are repeated until the algorithm converges.

Algorithm 1 Optimization Algorithm for Low-rank Exemplar SVMs (LRE-SVMs)

```
Input: Training data $\mathcal{X}_s$, and the parameters $C_1, C_2, \lambda_1, \lambda_2$.
1: Initialize $\mathbf{W} \leftarrow \mathbf{W}^0$, where $\mathbf{W}^0$ is obtained by solving (2).
2: repeat
3:     Calculate the prediction matrix $\mathbf{G}(\mathbf{W})$ based on the current $\mathbf{W}$.
4:     Solve for $\mathbf{F}$ by optimizing the problem in (8) with the SVT method.
5:     Update $\mathbf{W}$ by solving the problem in (7) with the gradient descent method.
6: until The objective converges or the maximum number of iterations is reached.
Output: The weight matrix $\mathbf{W}$.
```

4 Low-Rank Exemplar Least Square SVMs

While being able to exploit subdomains, LRE-SVMs is computationally inefficient. This is inherited from the exemplar SVMs method, in which an exemplar SVM is required to be trained for each positive training sample. In this section, we develop an efficient algorithm with a new LRE-SVMs formulation based on the least square SVMs, which is referred to as Low-Rank Exemplar Least Square SVMs or LRE-LSSVMs in short.

4.1 The Formulation

In the same spirit of exemplar SVMs, we learn an SVM classifier with the least square SVM method by using each positive sample and all the negative samples. In particular, let us denote the decision function as $g(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x})$. Based on the least square SVM method, we formulate the exemplar least square SVMs problem as follows,

$$\begin{aligned} \min_{\mathbf{W},\eta_i,\xi_{i,j}} \quad & \frac{1}{2}\|\mathbf{W}\|^2 + \frac{1}{2}C_1\sum_{i=1}^{n}\eta_i^2 + \frac{1}{2}C_2\sum_{i=1}^{n}\sum_{j=1}^{m}\xi_{i,j}^2 \\ \text{s.t.} \quad & \mathbf{w}_i'\phi(\mathbf{x}_i^+) + \eta_i = 1 \\ & \mathbf{w}_i'\phi(\mathbf{x}_j^-) + \xi_{i,j} = -1, \end{aligned} \tag{9}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_n] \in \mathbb{R}^{d\times n}$, and $\mathbf{w}_i \in \mathbb{R}^d$ is the weight vector for the decision function of the i-th exemplar least square SVM.

Let us denote the objective as $J_2(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|^2 + \frac{1}{2}C_1\sum_{i=1}^n \eta_i^2 + \frac{1}{2}C_2\sum_{i=1}^n\sum_{j=1}^m \xi_{i,j}^2$. By substituting it into the objective function of (6), we arrive at our new LRE-LSSVMs formulation based on the exemplar least square SVMs as follows,

$$\min_{\mathbf{W},\mathbf{F}} J_2(\mathbf{W}) + \lambda_1\|\mathbf{F}\|_* + \lambda_2\|\mathbf{F} - \mathbf{G}(\mathbf{W})\|_F^2, \tag{10}$$

where $\mathbf{G}(\mathbf{W})$ consists of the decision values of the exemplar classifiers on all positive training samples, *i.e.*, $\mathbf{G}(\mathbf{W}) =$

$[g_{ij}] \in \mathbb{R}^{n\times n}$ with $g_{ij} = \mathbf{w}_j'\phi(\mathbf{x}_i^+)$. Similarly, we also optimize the new problem by alternatingly update two matrices $\mathbf{F}$ and $\mathbf{W}$. When updating $\mathbf{F}$, the algorithm is the same as that in Section 3.4.2 based on the new definition of the matrix $\mathbf{G}$, so we omit the details here.

4.2 Fast Solution for Exemplar LSSVM

Now we discuss how to update $\mathbf{W}$, which is also the most computationally expensive part for training LRE-SVMs. When $\mathbf{F}$ is fixed, the subproblem for updating $\mathbf{W}$ can be written as

$$\min_{\mathbf{W}} J(\mathbf{W}) + \lambda_2\|\mathbf{G}(\mathbf{W}) - \mathbf{F}\|_F^2, \tag{11}$$

which can be further decomposed into n independent subproblems, with each subproblem for one exemplar SVM. Specifically, the k-th subproblem can be written as

$$\min_{\mathbf{w}_k,\eta_k,\xi_{k,j}} \quad \frac{1}{2}\|\mathbf{w}_k\|^2+\frac{1}{2}C_1\eta_k^2+\frac{1}{2}C_2\sum_{j=1}^{m}\xi_{k,j}^2+\frac{1}{2}\lambda_2\sum_{i=1}^{n}\zeta_i^2 \tag{12}$$

$$\text{s.t.} \quad \mathbf{w}_k'\phi(\mathbf{x}_k^+) + \eta_k = 1 \tag{13}$$

$$\mathbf{w}_k'\phi(\mathbf{x}_j^-) + \xi_{k,j} = -1, \tag{14}$$

$$\mathbf{w}_k'\phi(\mathbf{x}_i^+) + \zeta_i = f_{i,k}, \tag{15}$$

where $f_{i,k}$ is the (i,k)-th entry of $\mathbf{F}$. Moreover, the first constraint is on the exemplar sample $\mathbf{x}_k^+$ (*i.e.*, the k-th positive training sample), the second constraint is on each negative training sample $\mathbf{x}_j^-$, and the last constraint is on the i-th positive training sample $\mathbf{x}_i^+$.

Let us introduce the dual variable vector $\boldsymbol{\alpha} = [\alpha^+, \alpha_1^-, \ldots, \alpha_m^-, \alpha_1^f, \ldots, \alpha_n^f]' \in \mathbb{R}^{n+m+1}$, where each of α^+, α_j^-'s and α_i^f's are the dual variables for those three types of constraints in (13), (14) and (15), respectively. We also use a vector $\tilde{\mathbf{y}} = [+1, -1, \ldots, -1, f_{1,k}, \ldots, f_{n,k}]' \in \mathbb{R}^{n+m+1}$ to denote the values on the right hand side of the above constraints. Then, the dual problem of (12) can be written as,

$$(\tilde{\mathbf{K}} + \tilde{\mathbf{D}})\boldsymbol{\alpha} = \tilde{\mathbf{y}} \tag{16}$$

where the matrix $\tilde{\mathbf{K}} \in \mathbb{R}^{(n+m+1)\times(n+m+1)}$ is an extended kernel matrix. Specifically, the elements in the first row of $\tilde{\mathbf{K}}$ are the inner products between the exemplar training sample and all samples, *i.e.*, $[\phi(\mathbf{x}_k^+)'\phi(\mathbf{x}_k^+), \phi(\mathbf{x}_k^+)'\phi(\mathbf{x}_1^-), \ldots, \phi(\mathbf{x}_k^+)'\phi(\mathbf{x}_m^-), \phi(\mathbf{x}_k^+)'\phi(\mathbf{x}_1^+), \ldots, \phi(\mathbf{x}_k^+)'\phi(\mathbf{x}_n^+)]$. Similarly, the elements in the subsequent m rows are similarly defined as the inner products between all the negative samples and all samples, and the elements in remaining n rows are the inner products between all the positive training samples and all samples. The matrix $\tilde{\mathbf{D}}$ is a diagonal matrix, with the first element as $\frac{1}{C_1}$, the following m elements as $\frac{1}{C_2}$, and the remaining n elements ans $\frac{1}{\lambda_2}$. The solution can be obtained in close form as $\boldsymbol{\alpha} = (\tilde{\mathbf{K}} + \tilde{\mathbf{D}})^{-1}\tilde{\mathbf{y}}$. For the linear kernel case, the weight vector in the decision function can be recovered as $\mathbf{w}_k = \tilde{\mathbf{X}}\boldsymbol{\alpha}$, where $\tilde{\mathbf{X}} = [\mathbf{x}_k, \mathbf{x}_1^-, \ldots, \mathbf{x}_m^-, \mathbf{x}_1^+, \ldots, \mathbf{x}_n^+]$.

Although there is a closed form solution, it is still expensive to iteratively solve $\mathbf{w}$ for each exemplar SVM. The main cost is from the matrix inverse operation $(\tilde{\mathbf{K}} + \tilde{\mathbf{D}})^{-1}$, which is $O((n+m+1)^3)$ in time complexity. Let us define $\tilde{\mathbf{M}} = \tilde{\mathbf{K}} + \tilde{\mathbf{D}}$. For different exemplar samples, the resultant

Algorithm 2 Optimization Algorithm for Low-rank Least Square Exemplar SVMs (LRE-LSSVMs)

```
Input: Training data $\mathcal{X}_s$, and the parameters $C_1, C_2, \lambda_1, \lambda_2$.
1: Initialize $\mathbf{W} \leftarrow \mathbf{W}^0$, where $\mathbf{W}^0$ is obtained by solving (9), and calculate the matrix $\mathbf{M}^{-1}$.
2: repeat
3:     Calculate the prediction matrix $\mathbf{G}(\mathbf{W})$ based on the current $\mathbf{W}$.
4:     Solve for $\mathbf{F}$ by optimizing the problem in (8) with the SVT method.
5:     Update $\mathbf{W}$ by iteratively solving $n$ subproblems in (12) based on the precomputed $\mathbf{M}^{-1}$.
6: until The objective converges or the maximum number of iterations is reached.
Output: The weight matrix $\mathbf{W}$.
```

matrices $\tilde{\mathbf{M}}$'s are only different in the elements from the first row and the first column, which makes it possible to efficiently calculate the inverse of those matrices $\tilde{\mathbf{M}}$'s. In particular, each matrix $\tilde{\mathbf{M}}$ can be decomposed as

$$\tilde{\mathbf{M}} = \begin{bmatrix} m_{11} & \mathbf{m}_1' \\ \mathbf{m}_1 & \mathbf{M} \end{bmatrix} \tag{17}$$

where m_{11} is the $(1,1)$-th element of $\tilde{\mathbf{M}}$, $\mathbf{m}_1 \in \mathbb{R}^{n+m}$ is the vector consists of the remaining elements in the first column of $\tilde{\mathbf{M}}$, and $\mathbf{M} \in \mathbb{R}^{(n+m)\times(n+m)}$ is a submatrix of $\tilde{\mathbf{M}}$ by eliminating the elements in the first row and the first column. Using the block matrix inverse lemma, we arrive at

$$\tilde{\mathbf{M}}^{-1} = \begin{bmatrix} \mu & -\mu(\mathbf{M}^{-1}\mathbf{m}_1)' \\ -\mu\mathbf{M}^{-1}\mathbf{m}_1 & \mathbf{M}^{-1} + \mu\mathbf{M}^{-1}\mathbf{m}_1\mathbf{m}_1'\mathbf{M}^{-1} \end{bmatrix} \tag{18}$$

where $\mu = \frac{1}{m_{11} - \mathbf{m}_1'\mathbf{M}^{-1}\mathbf{m}_1}$. After calculating $\mathbf{M}^{-1}$, the time complexity of calculating $\tilde{\mathbf{M}}^{-1}$ for each exemplar SVM is reduce to $O((n+m)^2)$ only.

We summarize the optimization algorithm for LRE-LSSVMs in Algorithm 2. It is similar to Algorithm 1, except that we optimize a set of exemplar least square SVMs with the aforementioned fast algorithm at step 5.

5 Ensemble Exemplar Classifiers

After training the low-rank exemplar SVMs, we obtain n exemplar classifiers. To predict the test data, we discuss how to effectively use those learnt classifiers in three situations. The first one is the domain generalization scenario, where the target domain samples are not available during the training process. The second one is the domain adaptation scenario with fixed target domain, where we have unlabeled data in the target domain during the training process. And the third one is the domain adaptation scenario with evolving target domain, where the target test sample comes one by one, and is sampled from an unknown and gradually changing data distribution.

5.1 Domain Generalization

In the domain generalization scenario, we have no prior information about the target domain. A simple way is to equally fuse those n exemplar classifiers. Given any test sample $\mathbf{x}$, the prediction $p(\mathbf{x}|\mathbf{W})$ can be calculated as,

$$p(\mathbf{x}|\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n} p(\mathbf{x}|\mathbf{w}_i), \tag{19}$$

where $p(\mathbf{x}|\mathbf{w}_i)$ is the prediction from the i-th exemplar classifier. It can be the likelihood value (*resp.*, the decision value) when using the logistic regression method (*resp.*, the least square SVM method) for learning the exemplar classifier.

Recalling the training samples may come from several latent domains, a more practical way is to only use the exemplar classifiers from the latent domain that the test data likely belongs to. As aforementioned, each exemplar classifier encodes certain local information of the exemplar positive training sample. As a result, an exemplar classifier tends to output relatively higher prediction scores for the positive samples from the same latent domain, and relatively lower prediction scores for the positive samples from different latent domains. On the other hand, all exemplar classifiers are expected to output low prediction scores for the negative samples.

Therefore, given the test sample $\mathbf{x}$ during the test process, it is beneficial to fuse only the exemplar classifiers that output higher predictions, such that we output a higher prediction score if $\mathbf{x}$ is positive, and a low prediction score if $\mathbf{x}$ is negative. Let us define $\mathcal{T}(\mathbf{x}) = \{\, i \mid p(\mathbf{x}|\mathbf{w}_i) \text{ is one of the top } K \text{ prediction scores for } \mathbf{x}\}$ as the set of indices of those selected exemplar classifiers, then the prediction on this test sample can be obtained as,

$$p(\mathbf{x}|\mathbf{W}) = \frac{1}{K}\sum_{i:i\in\mathcal{T}(\mathbf{x})} p(\mathbf{x}|\mathbf{w}_i), \tag{20}$$

where K is the predefined number of exemplar classifiers that output high prediction scores for the test sample $\mathbf{x}$.

5.2 Domain Adaptation with Fixed Target Domain

When we have unlabeled data in the target domain during the training process, we can further assign different weights to the learnt exemplar classifiers and better fuse the exemplar classifiers for predicting the test data from the target domain. Intuitively, when the training data of one exemplar classifier is closer to the target domain, we should assign a higher weight to this classifier and vice versa.

Let us denote the target domain samples as $\{\mathbf{x}_1, \ldots, \mathbf{x}_u\}$, where u is the number of samples in the target domain. Based on the Maximum Mean Discrepancy (MMD) criterion [38], we define the distance between the target domain and the training samples for learning each exemplar classifier as follows,

$$d_i = \|\frac{1}{n+m}\left(n\phi(\mathbf{x}_i^+) + \sum_{j=1}^{m}\phi(x_j^-)\right) - \frac{1}{u}\sum_{j=1}^{u}\phi(\mathbf{x}_j)\|^2, \tag{21}$$

where $\phi(\cdot)$ is a nonlinear feature mapping function induced by the Gaussian kernel. We assign a higher weight n to the positive sample x_i^+ when calculating the mean of source domain samples, since we only use one positive sample for training the exemplar classifier at each time. In other words, we duplicate the positive sample $\mathbf{x}_i^+$ for n times and then combine the duplicated positive samples with all the negative samples to calculate the distance with the target domain.

With the above distance, we then obtain the weight for each exemplar classifier by using the RBF function as $v_i = \exp(-d_i/\sigma)$, where σ is the bandwidth parameter, and is set to be the median value of all distances. Then, the prediction on a test sample $\mathbf{x}$ can be obtained as,

$$p(\mathbf{x}|\mathbf{W}) = \sum_{i:i\in\mathcal{T}(\mathbf{x})} \tilde{v}_i p(\mathbf{x}|\mathbf{w}_i), \tag{22}$$

where $\mathcal{T}(\mathbf{x})$ is defined as in Section 5.1, and $\tilde{v}_i = v_i / \sum_{i:i\in\mathcal{T}(\mathbf{x})} v_i$.

One potential drawback with the above ensemble method is that we need to perform the predictions for n times, and then fuse the top K prediction scores. Inspired by Domain Adaptation Machine [19], we also propose to learn a single target classifier on the target domain by leveraging the predictions from the exemplar classifiers. Specifically, let us denote the target classifier as $f(\mathbf{x}) = \tilde{\mathbf{w}}'\phi(\mathbf{x}) + b$. We formulate our learning problem as follows,

$$\begin{aligned}
\min_{\tilde{\mathbf{w}}, b, \xi_i, \xi_i^*, \mathbf{f}} \quad & \frac{1}{2}\|\tilde{\mathbf{w}}\|^2 + C\sum_{i=1}^{u}(\xi_i + \xi_i^*) + \frac{\lambda}{2}\Omega(\mathbf{f}), && (23)\\
\text{s.t.} \quad & \tilde{\mathbf{w}}'\phi(\mathbf{x}_i) + b - f_i \le \epsilon + \xi_i, \quad \xi_i \ge 0, && (24)\\
& f_i - \tilde{\mathbf{w}}'\phi(\mathbf{x}_i) - b \le \epsilon + \xi_i^*, \quad \xi_i^* \ge 0, && (25)
\end{aligned}$$

where $\mathbf{f} = [f_1, \ldots, f_u]'$ is an intermediate variable, λ and C are the tradeoff parameters, ξ_i and ξ_i^* are the slack variables in the ϵ-insensitive loss similarly as in SVR, and ϵ is a predifined small positive value in the ϵ-insensitive loss. The regularizer $\Omega(\mathbf{f})$ is a smoothness function defined as follows,

$$\Omega(\mathbf{f}) = \sum_{j=1}^{u}\sum_{i:i\in\mathcal{T}(\mathbf{x}_j)} \tilde{v}_i \left(f_j - p(\mathbf{x}_j|\mathbf{w}_i)\right)^2, \tag{26}$$

where we enforce each intermediate variable f_j to be similar to the prediction scores from the selected exemplar classifiers in $\mathcal{T}(\mathbf{x}_j)$ for the target sample $\mathbf{x}_j$. In the above problem, we use the ϵ-insensitive loss to enforce the prediction score from target classifier $f(\mathbf{x}_j) = \tilde{\mathbf{w}}'\phi(\mathbf{x}_j) + b$ to be close to the intermediate variable f_j. At the same time, we also use a smoothness regularizer to enforce the intermediate variable f_j to be close to the prediction scores of the selected exemplar classifiers in $\mathcal{T}(\mathbf{x}_j)$ on the target sample $\mathbf{x}_j$. Intuitively, when $\tilde{v}_i$ is large, we enforce the intermediate variable f_j to be closer to $p(\mathbf{x}_j|\mathbf{w}_i)$, and vice versa. Recall the weight $\tilde{v}_i$ models the importance of the i-th exemplar classifier for predicting the target sample, so we expect the learnt classifier $f(\mathbf{x})$ performs well for predicting the target domain samples.

By introducing the dual variables $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_u]'$ and $\boldsymbol{\alpha}^* = [\alpha_1^*, \ldots, \alpha_u^*]'$ for the constraints in (24) and (25), we arrive at its dual form as follows,

$$\begin{aligned}
\min_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} \quad & \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\tilde{\mathbf{K}}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + \mathbf{p}'(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \\
& + \epsilon\mathbf{1}_u'(\boldsymbol{\alpha} + \boldsymbol{\alpha}^*), && (27)\\
\text{s.t.} \quad & \mathbf{1}'\boldsymbol{\alpha} = \mathbf{1}'\boldsymbol{\alpha}^*, \mathbf{0} \le \boldsymbol{\alpha}, \boldsymbol{\alpha}^* \le C\mathbf{1}, && (28)
\end{aligned}$$

where $\tilde{\mathbf{K}} = \mathbf{K} + \frac{1}{\lambda}\mathbf{I} \in \mathbb{R}^{u \times u}$ with $\mathbf{K}$ being the kernel matrix of the target domain samples, $\mathbf{p} = [p(\mathbf{x}_1|\mathbf{W}), \ldots, p(\mathbf{x}_u|\mathbf{W})]'$ with each entry $p(\mathbf{x}_j|\mathbf{W})$ defined in (22) being the "virtual label" for the target sample $\mathbf{x}_j$ as in DAM [19]. In DAM [19], the virtual labels of all the target samples are obtained by fusing the same set of source classifiers. In contrast, we use the predictions from different selected exemplar classifiers to obtain the virtual labels of different target samples. Therefore, DAM can be treated as a special case of our work by using the same classifiers for all test samples.

5.3 Domain Adaptation with Evolving Target Domain

The domain adaptation with evolving target domain is a learning problem between domain generalization and domain adaptation. On one hand, the unlabeled target data is also unseen during the training process. On the other hand, we have prior information that the unlabeled target samples arrive sequentially, which are assumed to be sampled from an unknown distribution that changes gradually in the sequential order.

As discussed in [12], the traditional domain adaptation methods are not applicable to this problem, because we usually cannot access a set of unlabeled target samples, which are required by those methods for reducing the domain distribution mismatch. Moreover, even if we use the existing unlabeled target domain samples as input, the traditional domain adaptation methods may not work well, because the underlying data distribution is dynamically changing.

Since the target domain samples are not available during the training process, a possible way is to treat it as a domain generalization problem. More interestingly, recall our LRE-SVMs and LRE-LSSVMs encode the local domain properties into those exemplar classifiers. For any given test target sample, by using the ensemble method in (20), we can dynamically select multiple exemplar classifiers with appropriate domain properties that well match the target sample, so our LRE-SVMs and LRE-LSSVMs approaches are also expected to perform well on the domain adaptation task with evolving target domain by simply treating it as an domain generalization problem, which is verified in our experiments (see Section 6.3).

However, if we simply apply LRE-SVMs or LRE-LSSVMs, we would have ignored the prior information that the underlying distribution of target test samples is gradually changing. Therefore, to further improve the stability of domain adaptation, we propose to learn a single classifier to predict the test sample in the target domain, and gradually update the classifier when the test samples in the target domain come sequentially. In particular, we first consider the traditional domain adaptation problem where we are given a batch of unlabeled target samples, and then adapt it to the scenario when the unlabeled target samples come one by one.

Given a set of unlabeled target samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_u\}$, we employ the similar formulation in (23) to learn a unified classifier $\mathbf{f}(\mathbf{x}) = \tilde{\mathbf{w}}'\mathbf{x}$ for predicting target domain samples, except replacing the ϵ-insensitive loss with the least square loss. In particular, we arrive at the following objective function,

Algorithm 3 Low-rank Exemplar SVMs (LRE-SVMs) for Domain Adaptation with Evolving Target Domain

```
Input: Sequential target data $\{\mathbf{x}_1, \ldots\}$, weight matrix of LRE-SVMs $\mathbf{W}$, and the tradeoff parameter $C$.
1:  Set $t = 1$.
2:  repeat
3:    if $t = 1$ then
4:      Calculate $\mathbf{H}^{-1} = (\mathbf{x}_t\mathbf{x}_t' + \frac{1}{C}\mathbf{I})^{-1}$.
5:      Calculate $\tilde{\mathbf{w}} = \mathbf{H}^{-1}\mathbf{x}_t p_t$.
6:    else
7:      Calculate $\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} + \frac{\mathbf{H}^{-1}\mathbf{x}_t(p_t - \tilde{\mathbf{w}}'\mathbf{x}_t)}{1+\mathbf{x}_t'\mathbf{H}^{-1}\mathbf{x}_t}$
8:      Calculate $\mathbf{H}^{-1} \leftarrow \mathbf{H}^{-1} - \frac{\mathbf{H}^{-1}\mathbf{x}_t\mathbf{x}_t'\mathbf{H}^{-1}}{1+\mathbf{x}_t'\mathbf{H}^{-1}\mathbf{x}_t}$.
9:    end if
10:   Predict the test sample $f(\mathbf{x}_t) = \tilde{\mathbf{w}}'\mathbf{x}_t$.
11:   Set $t \leftarrow t + 1$.
12: until The end of the input sequence.
Output: Decision values $\{\mathbf{f}(\mathbf{x}_1), \ldots\}$.
```

$$\min_{\tilde{\mathbf{w}}} \quad \frac{1}{2}\|\tilde{\mathbf{w}}\|^2 + C\sum_{i=1}^{u}(\tilde{\mathbf{w}}'\mathbf{x}_i - p_i)^2, \tag{29}$$

which has a closed form solution $\tilde{\mathbf{w}} = (\mathbf{X}\mathbf{X}' + \frac{1}{C}\mathbf{I})^{-1}\mathbf{X}\mathbf{p}$, where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_u] \in \mathbb{R}^{d \times u}$ is the data matrix of the unlabled target samples, $\mathbf{I} \in \mathbb{R}^{d \times d}$ is an identity matrix, and $\mathbf{p} = [p_1, \ldots, p_u]'$ with each $p_i = p(\mathbf{x}_i|\mathbf{W})$ being the prediction from LRE-SVMs as defined in (22).

When the target domain samples come sequentially, the problem in (29) can also be solved in an online fashion. In particular, let us denote $\mathbf{X}_t = [\mathbf{x}_1, \ldots, \mathbf{x}_t] \in \mathbb{R}^{d \times t}$ as the data matrix consisting of the first t samples, and denote $\mathbf{w}_t$ as the the weight vector of the prediction function at the t-th time. We also define $\mathbf{H}_t = \mathbf{X}_t{\mathbf{X}_t}' + \frac{1}{C}\mathbf{I}$. According to the Woodbury formula, the inverse of $\mathbf{H}_{t+1}$ can be written as

$$\mathbf{H}_{t+1}{}^{-1} = \mathbf{H}_t{}^{-1} - \frac{\mathbf{H}_t^{-1}\mathbf{x}_{t+1}\mathbf{x}_{t+1}'\mathbf{H}_t^{-1}}{1 + \mathbf{x}_{t+1}'\mathbf{H}_t{}^{-1}\mathbf{x}_{t+1}}. \tag{30}$$

Therefore, the weight vector $\tilde{\mathbf{w}}_{t+1}$ at time $(t+1)$, can be updated as,

$$\begin{aligned}\tilde{\mathbf{w}}_{t+1} &= (\mathbf{X}_{t+1}\mathbf{X}_{t+1}' + \frac{1}{C}\mathbf{I})^{-1}\mathbf{X}_{t+1}\mathbf{p}_{t+1} \\ &= \tilde{\mathbf{w}}_t + \frac{\mathbf{H}_t^{-1}\mathbf{x}_{t+1}(p_{t+1} - \tilde{\mathbf{w}}_t'\mathbf{x}_{t+1})}{1 + \mathbf{x}_{t+1}'\mathbf{H}_t^{-1}\mathbf{x}_{t+1}}.\end{aligned} \tag{31}$$

We observe that the calculation of $\tilde{\mathbf{w}}_{t+1}$ and $\mathbf{H}_{t+1}$ only relies on the $(t+1)$-th sample $\mathbf{x}_{t+1}$ and the matrix $\mathbf{H}_t$. Therefore, we can gradually update $\mathbf{H}_t$ and $\tilde{\mathbf{w}}_t$, and predict the target sample sequentially without storing those target samples.

We summarize the algorithm for domain adaptation with evolving target domain in Algorithm 3. Specifically, for the first target sample, we directly learn the least square SVM classifier using the closed form solution for (29). For the subsequent target samples, we update $\tilde{\mathbf{w}}$ and $\mathbf{H}^{-1}$ using the equations in (30) and (31). The target sample is predicted by using the latest classifier. The process is repeated until no more target samples arrive.

6 Experiments

In this section, we evaluate our two low-rank exemplar SVMs (LRE-SVMs) approaches for domain generalization, and domain adaptation with fixed and evolving target domains, respectively.

6.1 Domain Generalization

In domain generalization scenario, only the source domain samples are available in the training process. The task is to train classifiers that generalize well to the unseen target domain.

6.1.1 Experimental Setup

Following the work in [10], we use the Office-Caltech dataset [3], [39] for visual object recognition and the IXMAS dataset [40] for multi-view action recognition.

Office-Caltech [3], [39] dataset contains the images from four domains, Amazon (A), Caltech (C), DSLR (D), and Webcam (W), in which the images are from Amazon, Caltech-256, and two more datasets captured with digital SLR camera and webcam, respectively. The ten common categories among the 4 domains are utilized in our evaluation. Following the recent works on unsupervised domain adaptation [16], [18], [41], we extract the AlexNet $fc7$ feature [42] for the images in the Office-Caltech dataset.

IXMAS dataset [40] contains the videos from eleven actions captured by five cameras (Cam 0, Cam 1, ..., Cam 4) from different viewpoints. Each of the eleven actions is performed three times by twelve actors. To exclude the irregularly performed actions, we keep the first five actions (check watch, cross arms, scratch head, sit down, get up) performed by six actors (*Alba, Andreas, Daniel, Hedlena, Julien, Nicolas*), as suggested in [10]. We extract the dense trajectories features [43] from the videos, and use K-means clustering to build a codebook with $1,000$ clusters for each of the five descriptors (*i.e.*, trajectory, HOG, HOF, MBHx, MBHy). The bag-of-words features are then concatenated to a $5,000$ dimensional feature for each video sequence.

Following [10], we treat the images from different sources in the Office-Caltech dataset as different domains, and treat the videos from different viewpoints in the IXMAS dataset as different domains, respectively. In our experiments, we mix several domains as the source domain for training the classifiers and use the remaining domains as the target domain for testing. For the domain generalization task, the samples from the target domain are not available during the training process.

We compare our two low-rank exemplar SVMs approaches with the recent two latent domain discovering methods [9], [10]. We additionally report the results from the discriminative sub-categorization(Sub-C) method [44], as it can also be applied to our application. For all methods, we mix multiple domains as the source domain for training the classifiers following the experimental setup in [9], [10].

For those two latent domain discovering methods [9], [10], after partitioning the source domain data into different domains using their methods, we train an SVM classifier on each domain, and then fuse those classifiers for predicting the test samples. We employ two strategies to fuse the learnt classifiers as suggested in [10], which are referred to as the *ensemble* strategy and the *match* strategy, respectively. The ensemble strategy is to re-weight the decision values from different SVM classifiers by using the domain probabilities learnt with the method in [9]. In the match strategy, we first select the most relevant domain based on the MMD criterion, and then use the SVM classifier from this domain to predict the test samples.

TABLE 1
Recognition accuracies (%) of different methods for domain generalization. Our LRE-SVMs and LRE-LSSVMs approaches do not require domain labels or target domain data during the training process. The results of our LRE-SVMs and LRE-LSSVMs approaches are denoted in boldface.

Source	A,C	D,W	C,D,W	Cam 0,1	Cam 2,3,4	Cam 0,1,2,3
Target	D,W	A,C	A	Cam 2,3,4	Cam 0,1	Cam 4
SVM	84.96	84.09	92.28	71.70	63.83	56.61
Sub-C [44]	85.28	84.33	92.17	78.11	76.90	64.04
[9](Ensemble)	83.41	83.37	89.67	71.55	51.02	49.70
[9](Match)	81.86	79.29	88.10	63.81	60.04	48.91
[10](Ensemble)	85.07	84.39	91.82	75.04	68.98	57.64
[10](Match)	84.71	84.22	92.14	71.59	60.73	55.37
E-SVMs	85.24	84.64	92.47	76.86	68.04	72.98
LRE-SVMs	**85.29**	**85.01**	**92.66**	**79.96**	**80.15**	**74.97**
E-LSSVMs	85.85	84.02	92.46	80.68	70.99	71.58
LRE-LSSVMs	**87.56**	**84.72**	**92.99**	**81.05**	**81.81**	**72.75**

Moreover, we also report the results from the baseline SVM method, which is trained by using all training samples in the source domain. The results from exemplar SVMs (E-SVMs) (*resp.*, exemplar least square SVMs (E-LSSVMs)) are also reported, which is a special case of our proposed LRE-SVMs (*resp.*, LRE-LSSVMs) without considering the nuclear-norm based regularizer, and we also use the method in (20) to fuse the selected top K exemplar classifiers for the prediction. We fix $K = 5$ for all our experiments. For other learning parameters, we fix the regularizer parameters as $\lambda_1 = \lambda_2 = 10$, and set the loss weight parameter $C_1 = 10$ and $C_2 = 1$ on the Office-Caltech dataset, and $C_1 = 0.1$ and $C_2 = 0.001$ on the IXMAS dataset. For the baseline methods, we choose the optimal parameters according to their recognition accuracies on the test dataset.

6.1.2 Experimental Results

The experimental results on two datasets are summarized in Table 1. For the two latent domain discovering methods [9], [10], we observe that the recently published method by Gong *et al.* [10] achieves quite competitive results when using the ensemble strategy (*i.e.*, [10](Ensemble) in Table 1). It achieves better results in five cases when compared with SVM, which demonstrates it is beneficial to discover latent domains in the source domain. However, the method in [9] is not as effective as [10]. We also observe the match strategy generally achieves worse results than the ensemble strategy for those latent domain discovering methods, although the target domain information is used to select the most relevant discovered source domain in the testing process. Moreover, the Sub-C method also achieves better results on five cases when compared with SVM, because it also exploits the latent structure within each category, which could be helpful for improving the generalization ability similarly as the latent domain discovering methods.

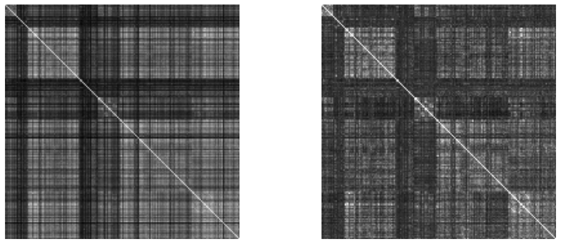
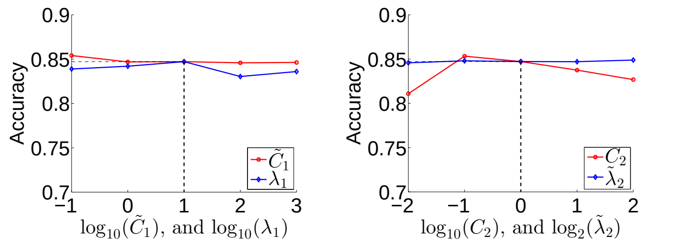

(a) LRE-LSSVMs (b) E-LSSVMs

Fig. 2. Visualization of the prediction matrix $\mathbf{G}(\mathbf{W})$.



(a) (b)

Fig. 3. The performance of our LRE-LSSVMs method when varying different parameters: (a) varying parameters $\tilde{C}_1$ and λ_1, and (b) varying parameters $\mathbf{C}_2$ and $\tilde{\lambda}_2$.

Compared with the baselines, our proposed LRE-SVMs and LRE-LSSVMs approaches achieve the best results in all six cases on two datasets, which clearly demonstrates the effectiveness of our method for domain generalization by exploiting the low-rank structure in the source domain. We also observe that our special cases (*i.e.*, exemplar SVMs (E-SVMs) and exemplar least square SVMs (E-LSSVMs)) also achieve comparable or better results than SVM. Note we also apply the prediction method using (20) for E-SVMs and E-LSSVMs. By selecting the most relevant classifiers, we combine a subset of exemplar classifiers for predicting each test sample, leading to good results. By further exploiting the low-rank structure in the source domain, we implicitly employ the information from latent domains in our LRE-SVMs and LRE-LSSVMs methods. In this way, the selected top K exemplar classifiers are more likely from the same latent domain that the test sample belongs to. Thus, our LRE-SVMs (*resp.*, LRE-LSSVMs) method outperforms its special case E-SVMs (*resp.*, E-LSSVMs) in all six cases for domain generalization.

6.1.3 Effect of the Low-Rank Regularizer

In this section, we provide some qualitative results to better understand the effect of the low-rank based regularizer in our our proposed LRE-LSSVMs and LRE-SVMs methods. Intuitively, the low-rank regularizer aims to enforce the predictions on positive training samples from the exemplar classifiers to be low-rank. In Figure 2, we show the prediction matrix $\mathbf{G}(\mathbf{W})$ for our LRE-LSSVMs method and its corresponding special case the E-LSSVMs method in the setting of “C, D, W $\rightarrow$ A” on the Office-Caltech dataset for the category “laptop computer”. As shown in Figure 2(b), the prediction matrix $\mathbf{G}$ of E-LSSVMs roughly exhibits the block diagonal property, which verifies our motivation that the predictions using the exemplar classifiers from the same latent domains tend to be similar. In Figure 2(a), we also observe that the low-rank property of the matrix $\mathbf{G}$ becomes more obvious after using the low-rank regularizer in our LRE-LSSVMs method.

6.1.4 Parameter Sensitivity Analysis

We show the performance changes of our LRE-LSSVMs method with respect to different parameters in Figure 3. In particular, we take the Office-Caltech dataset with the case $D, W \rightarrow A, C$ as a study case, and run our method by varying each of the four parameters (*i.e.*, C_1, C_2, λ_1, and λ_2) in a certain range when fixing the others. In particular, the parameters can be divided into two groups, the loss tradeoff parameters C_1 and C_2, and the weights for low-rank based regularizer λ_1 and λ_2. We set $C_1 = \tilde{C}_1 * C_2$, and $\lambda_2 = \tilde{\lambda}_2\lambda_1$, where $\tilde{C}_1$ is used to balance different loss terms for the exemplar positive sample and negative samples, and $\tilde{\lambda}_2$ indicates how much we relax the low-rank based regularizer. We vary those parameters by setting $C_2 \in [10^{-2}, 10^{-1}, 10^0, 10^1, 10^2]$, $\tilde{C}_1 \in [10^{-1}, 10^0, 10^1, 10^2, 10^3]$, $\lambda_1 \in [10^{-2}, 10^{-1}, 10^0, 10^1, 10^2]$, $\tilde{\lambda}_2 \in [2^{-1}, 2^0, 2^1, 2^2, 2^3]$.

It can be observed that our method is relatively stable when varying each parameter in a certain range. While we fix the parameter as described above, further tuning the parameter will give better performance. For example, the performance goes up when we set a smaller $\tilde{C}_1$ or C_2. Generally, it is still an open problem to automatically select the optimal parameters for domain adaptation and domain generalization methods, because of the lack of validation data in the target domain. How to use the cross-validation strategy to choose the optimal parameters for our proposed methods without having any labeled target domain data will be an interesting research issue in the future.

6.1.5 Comparison of Two LRE-SVMs Approaches

Compared with the LRE-SVMs method developed in our preliminary conference work [11], our newly developed LRE-LSSVMs method uses the least square loss to replace the logistic regression loss for learning each exemplar classifier. We have also developed a fast algorithm for solving a batch of exemplar least square SVMs. In terms of recognition accuracies, we observe from Table 1 that our new LRE-LSSVMs method generally achieve comparable results with the original LRE-SVMS method, where LRE-LSSVMs is better than LRE-SVMS in four cases, and worse in two cases only.

Besides the recognition accuracy comparison, the main advantage of the newly proposed LRE-LSSVMs method over the LRE-SVMS method is the time cost for training exemplar SVMs. We take the first case of the IXMAS dataset as an example to compare the training time of two approaches. Both methods are implemented with MATLAB, and we conduct the experiments on a workstation with Intel(R) Core i7-3770K CPU@3.50GHz and 16GB RAM. For each method, the total training time over all five actions is recorded. The experiments are repeated for 10 times for each method, and the average training time is reported in Table 2. It can be observed that the newly proposed LRE-LSSVMs approach is much faster (more than 80 times) than the original LRE-

TABLE 2
Average training time and standard deviation (in seconds) of LRE-SVMs and LRE-LSSVMs over 10 rounds of experiments on the IXMAS dataset (Cam 0,1 $\rightarrow$ Cam 2,3,4).

	LRE-SVMS	LRE-LSSVMs
Time	236.34 ±1.93	2.70 ±0.03

SVM approach on the IXMAS dataset for the case (Cam 0,1 $\rightarrow$ Cam 2,3,4). This shows the efficiency of our fast algorithm for learning the exemplar least square SVMs when updating $\mathbf{W}$ at the step 5 of Algorithm 2. Therefore, we conduct the following experiments by using LRE-LSSVMs due to its efficiency and effectiveness.

6.2 Domain Adaptation with Fixed Target Domain

In this section, we further evaluate our proposed method for the domain adaptation task, in which the unlabeled samples from the target domain are available in the training process. For domain adaptation, we adopt the approach proposed in Section 5.2 to fuse the exemplar classifiers learnt by using our LRE-LSSVMs method, and we refer to our approach for domain adaptation as *LRE-LSSVMs-DA*. We compare our method with other domain adaptation methods on two tasks, action recognition and object recognition. Because the deep transfer learning methods are assumed to take images as the input, they were mainly applied to the image based object recognition task [16], [18]. We divide our experiments into two parts, comparisons with traditional domain adaptation methods for the video based action recognition on the IXMAS dataset, and comparisons with the CNN based domain adaptation methods for image based object recognition the Office-Caltech dataset [3]. The experiments for image recognition on the benchmark Office dataset can also be found in the Supplementary.

6.2.1 Video Based Action Recognition

We first investigate the state-of-the-art unsupervised domain adaptation methods, including Kernel Mean Matching (KMM) [21], Sampling Geodesic Flow (SGF) [2], Geodesic Flow Kernel (GFK) [3], Selective Transfer Machine (STM) [45], Domain Invariant Projection (DIP) [13], and Subspace Alignment (SA) [14]. For all those methods, we combine the videos captured from multiple cameras to form one combined source domain, and use the remaining samples as the target domain. Then we apply all the methods for domain adaptation. For the feature-based approaches (*i.e.*, SGF, GFK, DIP and SA), we train an SVM classifier after obtaining the domain invariant features/kernels with those methods. We also select the best parameters for those baseline methods according to the test results.

The results of those baseline methods are reported in Table 3. We also include the baseline SVM method trained by using all the source domain samples as training data for the comparison. Cross-view action recognition is a challenging task. As a result, most unsupervised domain adaptation methods cannot achieve promising results on this dataset, and they are worse than SVM in many cases. The recently proposed methods SA [14] and DIP [13] achieve relatively better results, which are better than SVM in two out of three cases.

TABLE 3
Recognition accuracies (%) of different methods for domain adaptation. The best results are denoted in boldface.

Source		Cam 0,1	Cam 2,3,4	Cam 0,1,2,3
Target		Cam 2,3,4	Cam 0,1	Cam 4
SVM		71.70	63.83	56.61
KMM		73.92	42.22	52.57
SGF		60.37	69.04	28.66
GFK		64.87	55.53	42.16
STM		68.69	70.53	51.05
DIP		65.20	70.03	62.92
SA		73.35	77.92	49.59
GFK (latent)	[9] (Match)	61.33	58.77	46.62
	[9] (Ensemble)	65.32	55.01	42.09
	[10] (Match)	65.32	64.43	47.22
	[10] (Ensemble)	69.12	68.87	51.30
SA (latent)	[9] (Match)	58.49	56.27	55.87
	[9] (Ensemble)	63.01	62.05	62.69
	[10] (Match)	66.27	67.00	63.01
	[10] (Ensemble)	71.04	76.64	72.26
DAM (latent)	[9]	77.92	76.99	53.76
	[10]	77.32	73.94	62.47
LRE-LSSVMs-DA		**81.79**	**81.84**	**73.04**

We further investigate the latent domain discovering methods [9], [10]. We use their methods to divide the source domain into several latent domains. Then, we follow [10] to perform the GFK [3] method between each discovered latent domain and the target domain to learn a new kernel for reducing the domain distribution mismatch, and train SVM classifiers using the learnt kernels. Then we also use the two strategies (*i.e.*, ensemble and match) to fuse the SVM classifiers learnt from different latent domains. Moreover, as the SA method achieves better results than GFK on the combined source domain, we further use the SA method to replace the GFK method for reducing the domain distribution mismatch between each latent domain and the target domain. The other steps are the same as those when using the GFK method. We report the results using latent domain discovering methods [9], [10] combined with GFK and SA in Table 3, which are denoted as *GFK(latent)* and *SA(latent)*, respectively. As our method is also related to the DAM method [19], we also report the results of the DAM method by treating the discovered latent domains with [9], [10] as multiple source domains, which is referred to as *DAM(latent)*.

From Table 3, we observe GFK(latent) using the latent domains discovered by [10] is generally better when compared with GFK(latent) using the latent domains discovered by [9]. By using the latent domains discovered by [10], the results of GFK(latent) using both match and ensemble strategies are better than those of GFK on the combined source domain. However, most results from GFK(latent) are still worse than SVM, possibly because the GFK method cannot effectively handle the domain distribution mismatch between each discovered latent domain and the target domain. When using the SA method to replace GFK, we observe the results from SA(latent) in all three cases are improved when compared with their corresponding results from GFK(latent) by using the latent domains discovered by [10]. Moreover, we also observe DAM(latent) outperforms SVM in all cases or most cases when using the latent source domains discovered by [10] or [9].

TABLE 4
Recognition accuracies (%) of different methods for domain adaptation on the Office-Caltech dataset. The best results are denoted in bold.

Source	A,C	D,W	C,D,W
Target	D,W	A,C	A
SVM	85.40	83.76	92.28
DAN [18]	92.92	82.08	92.48
ReverseGrad [16]	92.25	88.90	93.11
LRE-LSSVMs-DA	89.55	88.15	93.27
LRE-LSSVMs(DAN)	**94.54**	83.24	93.07
LRE-LSSVMs(ReverseGrad)	92.55	**89.32**	**93.57**

Our method achieves the best results in all three cases, which again demonstrates the effectiveness of our proposed LRE-LSSVMs-DA for exploiting the low-rank structure in the source domain. Moreover, our method LRE-LSSVMs-DA outperforms LRE-LSSVMs on three cases (see Table 1). Note LRE-LSSVMs does not use the target domain unlabeled samples during the training process. The results further demonstrate the effectiveness of our domain adaptation approach LRE-LSSVMs-DA for coping with the domain distribution mismatch in the domain adaptation task.

6.2.2 *Image Based Object Recognition*

We compare our LRE-LSSVMs-DA method with the recently proposed deep transfer learning methods DAN [18], and ReverseGrad [16] for object recognition on the Office-Caltech dataset. We strictly follow the experimental setup in [16], [18]. We first fine-tune pretrained AlexNet model based on the ImageNet dataset by using the labeled samples in the source domain, and then use the fine-tuned CNN model to extract the features from the images in both source and target domains. For DAN and ReverseGrad, we use their released source codes, and fine-tune the pretrained AlexNet model by using the suggested parameters in [16], [18].

The results are shown in Table 4, in which the results of the baseline SVM method are also included for comparison. Comparing with the baseline SVM method in Table 1 that directly uses AlexNet $fc7$ features, fine-tuning the networks using the labeled data from the source domain does not gain much improvements on this dataset. We also observe that DAN and ReverseGrad generally achieve quite good results when using multiple datasets as one source domain. However, there is no consistent winner when comparing these two methods and our method LRE-LSSVMs-DA. In particular, our method achieves the best result on the last case, while the DAN method and the ReverseGrad method win on the first and second case, respectively.

Moreover, the deep transfer learning methods are proposed to learn domain-invariant features, while our proposed LRE-SVMs and LRE-LSSVMs methods aim to improve the cross-domain generalization ability by learning robust exemplar classifiers, namely, their methods focus on feature learning, while our work focuses on classification. So our proposed LRE-LSSVMs method can be used to further improve the recognition accuracies by learning the exemplar classifiers with the features extracted by DAN and ReveseGrad, which are denoted by LRE-LSSVMs(DAN) and LRELSSVMs(ReverseGrad), respectively. From the last two rows of Table 4, it can be observed that LRE-LSSVMs(DAN) consistently outperforms DAN, while LRELSSVMs(ReverseGrad) is consistently better than ReveseGrad, which demonstrates that our LRELSSVMs method is complementary to the two deep transfer learning methods DAN and ReveseGrad by exploiting the local statistics to further enhance the generalization ability across domains.

6.3 Domain Adaptation with Evolving Target Domain

In the domain adaptation scenario with evolving target domain, unlabeled samples in the target domain are provided sequentially. Each target sample is assumed to be sampled from an unknown and gradually changing distribution. In other words, unlabeled samples in the target domain are unseen when learning the classifiers using labeled data in the source domain, and the adaptation is performed in an online fashion during the testing process.

6.3.1 *Experimental Setup*

Similarly as in [33], we conduct experiments on the CarEvolution dataset [46] [2], which consists of $1,086$ images of different cars. Each car is annotated with one of three manufactories (*i.e.,* BMW, Mercedes or VW), as well as the year in which the car model was introduced (from 1972 to 2013). The task is to predict the manufactory of the car in each image.

As discussed in [32], the car style is gradually varying during the past decades. Therefore, the car images in the CarEvolution dataset can be assumed to be sampled from an underlying distribution, which is gradually changed chronologically. Different from the task in [33], we aim to investigate the classification performance when the target data is varying, so we split as many images as possible into the test set. We conduct the experiments in two settings. In the first setting, we use the images of cars after 1980 as the target domain and the remaining images of cars before 1980 as the source domain (*i.e.,* $\leq 1980 \rightarrow > 1980$), while in the second setting we use the images of cars after 1990 as the target domain and the remaining images of cars before 1990 as the source domain (*i.e.,* $\leq 1990 \rightarrow > 1990$). The target images are ordered chronologically, and are assumed to be provided one by one.

We extract the CNN features using the output from the "fc6" layer of the CAFFE reference model [47], which leads to a $4,096$ dimension feature vector for each image. Each feature vector is further normalized such that its ℓ_2 norm equals 1. When predicting the target samples, we employ Algorithm 3 based on the pre-learnt LRE-LSSVMs classifiers trained based the training samples from the source domain, and refer to our approach as *LRE-LSSVMS-EDA*. We compare our work with the recently proposed Continuous Manifold Adaptation (CMA) method [48] [3]. Following [48], we integrate the CMA approach with two subspace based domain adaptation methods, the GFK method and the SA method, and refer to them as *CMA+GFK* and *CMA+SA*, respectively. The SVM method is used to train the classifiers for predicting the target samples after applying CMA+GFK

2. http://homes.esat.kuleuven.be/~krematas/VisDA/CarEvolution.html
3. Codes are downloaded from http://cma.berkeleyvision.org/

TABLE 5
Recognition accuracies (%) of different methods for domain adaptation with evolving target domain, where the target domain distribution is gradually changing. The best results are denoted in boldface.

	Setting 1	Setting 2
Source	≤ 1980	≤ 1990
Target	> 1980	> 1990
SVM	39.66	46.08
CMA+GFK	42.95	47.93
CMA+SA	42.73	44.39
LRE-LSSVMs	43.61	50.53
LRE-LSSVMs-EDA	**44.33**	**51.01**

or CMA+SA. We also include the SVM method without domain adaptation as a baseline for comparison. For the baseline methods, the optimal parameters are chosen according to their best recognition accuracies on the test dataset.

6.3.2 Experimental Results

The recognition accuracies of all methods on the CarEvolution dataset are summarized in Table 5. Generally, the task become more challenging, when we split more samples into the test set. For example, the recognition accuracies of all methods in the first setting ($\leq 1980 \rightarrow > 1980$) are lower than their corresponding results in the second setting ($\leq 1990 \rightarrow > 1990$). The CMA+GFK method achieves better results than the baseline SVM method in both settings by considering the evolving distribution of target domain samples, while the CMA+SA method does not perform well in the second setting. Our LRE-LSSVMs method achieves better results than two CMA methods, which demonstrates excellent generalization ability of our approach by combining the exemplar classifiers for domain generalization. By further considering the distribution changing on the target domain, our LRE-LSSVMs-EDA method achieves the best results in both settings, which clearly demonstrates the effectiveness of our LRE-LSSVMs-EDA approach for domain adaptation with evolving target domain.

7 Conclusions

In this paper, we have proposed a new approach called Low-rank Exemplar SVMs (LRE-SVMs) for domain generalization by exploiting the low-rank structure of positive training samples from multiple latent source domains. Specifically, based on the recent work on exemplar SVMs, we propose to exploit the low-rank structure in the source domain by introducing a nuclear-norm based regularizer on the prediction matrix consisting of the predictions of all positive samples from all exemplar classifiers. We develop a new LRE-SVMs approach based on the least square SVMs (referred to as LRE-LSSVMs), which is much faster than the original LRE-SVMs method. To additionally handle the domain distribution mismatch between the training and test data, we further develop an effective method to reweight the selected set of exemplar classifiers based on the Maximum Mean Discrepancy (MMD) criterion, and extend the Domain Adaptation Machine (DAM) method to learn a unified target classifier. We also develop a new algorithm for the domain adaptation problem with evolving target domain, where the data distribution of target domain is gradually changing. The comprehensive experiments have demonstrated the effectiveness of our approach for domain generalization, and domain adaptation with fixed and evolving target domains.

References

[1] B. Kulis, K. Saenko, and T. Darrell, “What you saw is not what you get: Domain adaptation using asymmetric kernel transforms,” in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[2] R. Gopalan, R. Li, and R. Chellappa, “Domain adaptation for object recognition: An unsupervised approach,” in *International Conference on Computer Vision (ICCV)*, 2011.

[3] B. Gong, Y. Shi, F. Sha, and K. Grauman, “Geodesic flow kernel for unsupervised domain adaptation,” in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[4] L. Duan, D. Xu, I. W. Tsang, and J. Luo, “Visual event recognition in videos by learning from web data,” *IEEE Transactions on Patter Analysis and Machine Intelligence (T-PAMI)*, vol. 34, no. 9, pp. 1667–1680, 2012.

[5] L. Duan, I. W. Tsang, and D. Xu, “Domain transfer multiple kernel learning,” *IEEE Transactions on Patter Analysis and Machine Intelligence (T-PAMI)*, vol. 34, no. 3, pp. 465–479, 2012.

[6] W. Li, L. Duan, D. Xu, and I. W. Tsang, “Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation,” *IEEE Transactions on Patter Analysis and Machine Intelligence (T-PAMI)*, vol. 36, no. 6, pp. 1134–1148, 2014.

[7] K. Muandet, D. Balduzzi, and B. Schölkopf, “Domain generalization via invariant feature representation,” in *International Conference on Machine Learning (ICML)*, 2013.

[8] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba, “Undoing the damage of dataset bias,” in *European Conference on Computer Vision (ECCV)*, 2012.

[9] J. Hoffman, B. Kulis, T. Darrell, and K. Saenko, “Discovering latent domains for multisource domain adaptation,” in *European Conference on Computer Vision (ECCV)*, 2012.

[10] B. Gong, K. Grauman, and F. Sha, “Reshaping visual datasets for domain adaptation,” in *Neural Information Processing Systems (NIPS)*, 2013.

[11] Z. Xu, W. Li, L. Niu, and D. Xu, “Exploiting low-rank structure from latent domains for domain generalization,” in *European Conference on Computer Vision (ECCV)*, 2014.

[12] J. Hoffman, T. Darrell, and K. Saenko, “Continuous manifold based adaptation for evolving visual domains,” in *CVPR*, 2014.

[13] M. Baktashmotlagh, M. Harandi, and M. S. Brian Lovell, “Unsupervised domain adaptation by domain invariant projection,” in *International Conference on Computer Vision (ICCV)*, 2013.

[14] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, “Unsupervised visual domain adaptation using subspace alignment,” in *International Conference on Computer Vision (ICCV)*, 2013.

[15] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, “Simultaneous deep transfer across domains and tasks,” in *International Coference on Computer Vision (ICCV)*, 2015.

[16] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in *International Conference on Machine Learning (ICML)*, 2015.

[17] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, “Revisiting batch normalization for practical domain adaptation,” *arXiv [cs.CV]*, 15 Mar. 2016.

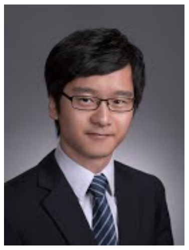
[18] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning transferable features with deep adaptation networks,” in *International Conference on Machine Learning (ICML)*, 2015.

[19] L. Duan, D. Xu, and I. W. Tsang, “Domain adaptation from multiple sources: A domain-dependent regularization approach,” *IEEE Transactions on Neural Networks and Learning Systems (T-NNLS)*, vol. 23, no. 3, pp. 504–518, March 2012.

[20] L. Bruzzone and M. Marconcini, “Domain adaptation problems: A DASVM classification technique and a circular validation strategy,” *IEEE Transactions on Patter Analysis and Machine Intelligence (T-PAMI)*, vol. 32, no. 5, pp. 770–787, 2010.

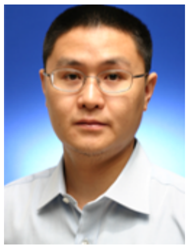
[21] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Scholkopf, “Correcting sample selection bias by unlabeled data,” in *Neural Information Processing Systems (NIPS)*, 2007.

[22] M. Dudik, R. E. Schapire, and S. J. Phillip, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *NIPS*, 2005.
[23] M. Sugiyama, S. Nakajima, H. Kashima, P. von Bnau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *NIPS*, 2008.
[24] S. Bickel, M. Brckner, and T. Scheffer, "Discriminative learning for differing training and test distributions," in *ICML*, 2007.
[25] Y. Tsuboi, H. Kashima, S. Hido, S. Bickel, and M. Sugiyama, "Direct density ratio estimation for large-scale covariate shift adaptation," *JIP*, vol. 17, pp. 138–155.
[26] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi, "Domain generalization for object recognition with multi-task autoencoders," in *International Coference on Computer Vision (ICCV)*, 2015.
[27] L. Niu, W. Li, and D. Xu, "Multi-view domain generalization for visual recognition," in *International Conference on Computer Vision (ICCV)*, 2015.
[28] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-SVMs for object detection and beyond," in *International Conference on Computer Vision (ICCV)*, 2011.
[29] Z. Xu, X. Li, K. Yang, and T. Goldstein, "Exploiting low-rank structure for discriminative sub-categorization," in *British Machine Vision Conference (BMCV)*, 2015.
[30] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," *Neural Information Processing Systems (NIPS)*, 2007.
[31] J. Chen, J. Zhou, and J. Ye, "Integrating low-rank and group-sparse structures for robust multi-task learning," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2011.
[32] Y. J. Lee, A. A. Efros, and M. Hebert, "Style-aware mid-level representation for discovering visual connections in space and time," in *International Conference on Computer Vision (ICCV)*, 2013.
[33] C. H. Lampert, "Predicting the future behavior of a time-varying probability distribution," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
[34] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1, pp. 151–175, 2010.
[35] C. Zhang, L. Zhang, and J. Ye, "Generalization bounds for domain adaptation," in *NIPS*, 2012.
[36] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.
[37] J. Cai, E. J. Cands, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.
[38] A. Gretton, K. M. Gorgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Reserarch (JMLR)*, vol. 23, pp. 723–773, 2012.
[39] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European Conference on Computer Vision (ECCV)*, 2010.
[40] D. Weinland, E. Boyer, and R. Ronfard, "Action recognition from arbitrary views using 3d exemplars," in *International Conference on Computer Vision (ICCV)*, 2007.
[41] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance." *CoRR*, vol. abs/1412.3474, 2014.
[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.
[43] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International journal of computer vision*, vol. 103, no. 1, pp. 60–79, 2013.
[44] M. Hoai and A. Zisserman, "Discriminative sub-categorization," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.
[45] W.-S. Chu, F. D. la Torre, and J. F. Cohn, "Selective transfer machine for personalized facial action unit detection." in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.
[46] K. Rematas, B. Fernando, T. Tommasi, and T. Tuytelaars, "Does evolution cause a domain shift?" 2013.
[47] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.
[48] J. Hoffman, T. Darrell, and K. Saenko, "Continuous manifold based adaptation for evolving visual domains," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.



Wen Li is a postdoctoral researcher with the Computer Vision Laboratory, ETH Zürich, Switzerland. He received the Ph.D. degree from the Nanyang Technological University, Singapore in 2015. Before that, he received the B.S. and M.Eng degrees from the Beijing Normal University, Beijing, China, in 2007 and 2010, respectively. His main interests include transfer learning, multi-view learning, multiple kernel learning, and their applications in computer vision.

Zheng Xu is a Ph.D. student with the Department of Computer Science, University of Maryland, College Park, USA. Before that, he was a Project Officer at the Visual Computing Research Group, School of Computer Engineering, Nanyang Technological University, Singapore. He received his M.Eng. and B.Eng. degrees in the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC). During his master study, he worked as an intern at Microsoft Research Asia (MSRA). His current research focuses on optimization and machine learning.



Dong Xu (M'07) received the B.E. and Ph.D. degrees from University of Science and Technology of China, in 2001 and 2005, respectively. While pursuing his Ph.D., he was with Microsoft Research Asia, Beijing, China, and the Chinese University of Hong Kong, Shatin, Hong Kong, for more than two years. He was a Post-Doctoral Research Scientist with Columbia University, New York, NY, for one year. He worked as a faculty member with Nanyang Technological University, Singapore. Currently, he is a Professor and Chair in Computer Engineering with the School of Electrical and Information Engineering, the University of Sydney, Australia. His current research interests include computer vision, statistical learning, and multimedia content analysis. Dr. Xu was the co-author of a paper that won the Best Student Paper Award in the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) in 2010, and a paper that won the Prize Paper Award in IEEE Transactions on Multimedia (T-MM) in 2014.

Dengxin Dai is currently a postdoctoral researcher with the Computer Vision Laboratory, ETH Zürich, Switzerland, where he received his Ph.D. degree in 2016. Before that, he received his M.Sc. degree in Signal Processing and his B.Sc. degree in Information Science and Technology from Wuhan University, China, in 2010 and 2008, respectively. His current research interests include visual recognition with limited supervision, urban/driving scene understanding, and the integration of vision and language.

Luc Van Gool got a degree in electromechanical engineering at the Katholieke Universiteit Leuven in 1981. Currently, he is a professor at the Katholieke Universiteit Leuven in Belgium and the ETH in Zürich, Switzerland. He leads computer vision research at both places, where he also teaches computer vision. He has authored over 200 papers in this field. He has been a program committee member of several major computer vision conferences. His main interests include 3D reconstruction and modeling, object recognition, tracking, and gesture analysis. He received several Best Paper awards. He is a co-founder of 5 spin-off companies.