Assembling Heterogeneous Domain Adaptation Methods for Image Classification

Boris Chidlovskii, Gabriela Csurka and Shalini Gangwar

Xerox Research Centre Europe,
6 chemin Maupertuis, Meylan, France

Abstract. In this paper we report the contribution of XRCE team to the Domain Adaptation Challenge [10] organized in the framework of ImageCLEF 2014 competition [9]. We describe our approach to build an image classification system when a weak image annotation in the target domain is compensated by massively annotated images in source domains. One method is based using several heterogeneous methods for the domain adaptation aimed at the late fusion of the individual predictions. One big class of domain adaptation methods addresses a selective reuse of instances from source domains for target domain. We adopt from this class the adaptive boosting for weighting source instances which learns a combination of weak classifiers in the target domain. Another class of methods aims to transform both target and source domains in a common space. In this class we focused on metric learning approaches aimed at reducing distances between images from the same class and to increase distances of different classes independently if they are from source or target domain. Combined the above approaches with a "brute-force" SVM-based approach we obtain a set of heterogeneous classifiers for class prediction of target instances. In order to improve the overall accuracy, we combine individual classifiers through different versions of majority voting. We describe different series of experiments including those submitted for the official competition and analyze their results.

1 Introduction

The shortage of labeled data is a fundamental problem in machine learning applications. While huge amounts of unlabeled data is generated and made available in many domains, the cost of acquiring data labels remains high. Domain adaptation addresses this problem by leveraging labeled data in one or more related domains, often referred as "source" domains, when learning a classifier for unseen data in a "target" domain. The domains are assumed to be related but not identical.

This situation occurs in domains where machine learning components are intensively deployed, such as event detection in videos, entity recognition across different text corpora, object recognition in images acquired in different conditions (see [21] for a survey of domain adaptation methods and [24] for a survey on the related field of transfer learning).

Domain adaptation has also received a significant attention in computer vision applications [17,18,19,20,22,26,27], Different aspects have been addressed and various approaches to the domain adaptation have been proposed [7,25]; this clearly indicates how complex and multi-faceted the problem is.

1.1 Domain adaptation challenge

Current research in domain adaptation focuses on a scenario where (a) the prior domain (source) consists of one or maximum two databases (b) the labels between the source and the target domain are the same, and (c) the number of annotated training data for the target domain are limited. The goal of this challenge is to push the state of the art towards more realistic settings, relaxing these assumptions[1].

Indeed researchers and teams participating in the ImageCLEF DA 2014 challenge were asked to build recognition systems for the target classes by leveraging the knowledge from four source domains. Both source and target data are provided by exploiting existing available resources.

Specifically, the participants were provided with image features[2] extracted by the organizers from 600 randomly selected images collected from five different image collections: Caltech-256 [2], ImageNet12 [3], PASCAL-VOC12 [4], Bing [1] and SUN [5]. The organizers selected 12 common classes from each datasets, namely, *aeroplane, bike, bird, boat, bottle, bus, car, dog,horse, monitor, motorbike, people*. (see Figure 1 for example images for each class and in all collection, showing the variability of classes between the collections). The first four collections from the list are proposed as source domains for which the image features and all the labels were provided. The SUN dataset served as the target domain, with 60 annotated and 600 non-annotated instances. The task was to provide predictions for the non-annotated target data.

2 XRCE approach to domain adaptation

We started by analyzing the domain adaptation task and the available features for the different collections. Three key elements of our analysis are the following:

1. Original images are not available neither for target nor for sources domains; the participants dispose feature sets only. It makes impossible to leverage the intermediate knowledge generated during the feature extraction process, such as the choice of low or high level features or the vocabulary for bags of visual words.
2. Source and target domains are semantically related, however they are different feature-wise. Figure 2.left compares the PCA projection of four source and target domains made available at phase 1 of the challenge[3]. A similar relationship (see Figure 2.middle) can be found between PCA projections for the source and target domain data at phase 2. Worse, as shown in Figure 2.right, the target feature distribution has been changed between phases 1 and 2. This change is likely due to

[1] From `http://www.imageclef.org/2014/adaptation`.

[2] These image features were the concatenation of four bag-of-visual words [11] built on a 2x2 split of the image where the low level features were SIFT descriptors extracted densely from the corresponding image regions.

[3] At the beginning of the challenge, the participants were provided with a similar problem configuration but different feature sets in order to familiarize with the problem. In addition all labels for the target set were also available in order to allow participants to evaluate their methods. We will refer to this setting as phase 1. The features for the actual training for the submission were released in phase 2.

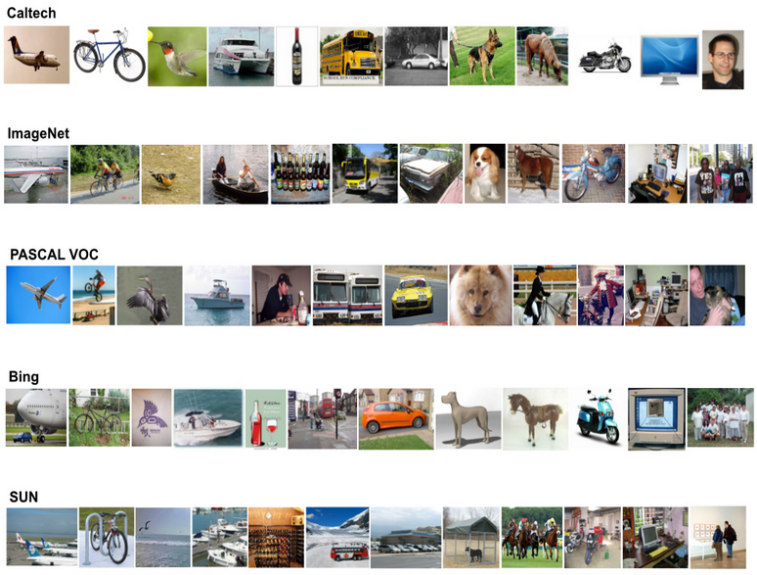

Fig. 1. Examples of 12 classes from four source domains (Caletch, InageNet, Pascal, Bing) and the target domain (SUN).

parameter changes in the feature extraction process hidden from the participants. Consequently, it made impossible any deployment of models learned at phase 1.

3. According to our experiments made with the features released in phase 1, none of existing domain adaptation methods for the visual classification appeared as a perfect match for the challenge task [7,25].

Putting all these elements together, we decided to proceed in two steps. First, we undertook a number of conceptually uncorrelated approaches to domain adaptation. Second, we used ensemble techniques to aggregate individual predictions using different majority voting methods.

More precisely, in the first step we tested methods of the following classes:

Brute force : Cross validation on available training data to identify best combinations of source domains for transferring to the target domain, with the optimal set of parameter values and kernel functions.

Instance weighting : Boosting-based instance-transfer domain adaptation to identify different instances in one or more source domains allowing to boost the learning in the target domain.

Space transformation : Metric learning-based domain adaptation where the idea is to transform the feature space and bring instances from the same class close to each other, independently if they are from source or target domains.

The remainder of this working note is organized as follows. In Section 3 we describe three individual domain adaptation methods. Section 4 describes the ensemble

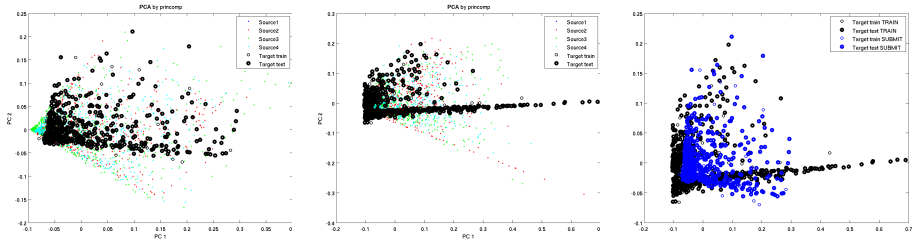

Fig. 2. PCA projections for source and target domains. a) Target and source data at phase 1; b) Target and source data at phase 2; 3) Target data at phases 1 and 2.

methods over heterogeneous classifiers. Section 5 reports evaluation results including the ten submissions to the challenge. We analyze the results and the impact of different components on the overall performance. Finally we outline future work and conclude in Section 6.

3 Individual domain adaptation methods

There are two main cases in domain adaptation, they depend on available data in the target domain. In the unsupervised case, no labeled target instances are available. In the semi-supervised case, a few target instances are labeled. In ImageCLEF DA challenge, 60 labeled target instances are available, this positions the challenge into the second case.

We suppose that the labeled target dataset $\mathcal{T}_l$ is drawn from the same distribution as the unlabeled target set $\mathcal{T}_u$ and hence it will play an important role in building the classification model. However the size of T_l is too small to adequately train a good classifier for the test data $\mathcal{T}_u$.

The training data from source domains are more abundant, but the classifiers learned from these data cannot classify the test data well due to different data distributions. Let $\mathcal{S}_1, \ldots, \mathcal{S}_{N_S}$ denote the source domains, where $N_S \geq 1$ is the number of sources. Let X_{s_k} be an instance space of $\mathcal{S}_k$ (*e.g.* a subset of $\mathcal{S}_k$ used for training). We denote by X_t the instance space of the target domain (that can take elements from $\mathcal{T}_l$, $\mathcal{T}_u$ or both). Further, we denote by $Y = \{c_1, c_2, \ldots, c_{N_c}\}$ the set of category labels common to all domains.

3.1 Brute force

We start off with the brute force approach which uses available annotated data to assess the relatedness between source domains and the target domain and to test a straightforward domain adaptation scenario. For the $N_S = 4$ different source domains we consider $N_{SC} = 2^N_S - 1 = 15$ source combinations $SC_i, i = 1, \ldots, N_{SC}$ which are generated by an exhaustive enumeration of all possible subsets of source domains, *e.g.*

$SC_1 = \{\mathcal{S}_1\}$, $SC_6 = \{\mathcal{S}_1, \mathcal{S}_3\}$ and $SC_{15} = \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$. For each source combination SC_j, we concatenate the target train set $\mathcal{T}_l$ with the selected sources SC_j and train the SVM models in a cross validation mode to find out the optimal parameter values and kernel functions. Two versions have been implemented, the multi-task SVM and one-against-all binary SVMs, as follows:

Multi-class SVM: We first used the multi-class LIBSVM package[4] for the multi-class classification in the target domain. In the cross validation with $k = 10$ folds, we tested different values of SVM parameters μ, ν, C, the standard kernels available with the package and different source combinations SC_i we found as the best source combination $CS_{13} = \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_4\}$ with the linear ν-SVM and the parameters $\nu = 0.12$ and $C = 0.01$. This classifier denoted by f_{mcsvm} allows a gain of 3% compared to the baseline multi-class SVM f_{bf0} with no source domains and cross validated parameters.

Binarised SVM: We additionally consider the multi-class classification by a composition of one-against-all classifiers running again the LIBSVM package and cross validate to determine the optimal parameters for each of $N_c = 12$ classes in the datasets. Unlike the multi-class SVM above where the optimal parameter values, kernel and the source combination are common for all classes, in the binarized version we identify the best classifier for each class $c_j \in Y$ which comes with a specific set of parameter values, kernels and source combinations for each class c_i. Hence, the multi-class classifier f_{bsvm} is composed of N_c binary classifiers $f_{\text{bsvm}}^{c_j}, j = 1, \ldots, N_c$. For an unseen sample $\mathbf{x}_i$, it proceeds by applying all classifiers and predicting the label $\hat{y}_{\text{bsvm}} = c_j$ for which the corresponding classifier reports the highest confidence score:

$$\hat{y}_{\text{bsvm}} = \underset{c_j \in Y}{\operatorname{argmax}} f_{\text{bsvm}}^{c_j}(\mathbf{x}_i).$$

In a probabilistic setting, all class confidence scores are converted into probabilities $P(y_i = c | f_{\text{bsvm}}(\mathbf{x}_i))$.

3.2 Instance Transfer with Adaboost

We extended the Transfer AdaBoost learning algorithm (TrAdaboost) [12], an extension of the AdaBoost [15] for transfer learning by assuming that there is abundant source training data to learn a classifier, but the target domain is different from the source. Hence this approach is can be easily adopted for the domain adaptation.

AdaBoost aims to boost the accuracy of a weak learner by carefully adjusting the weights of training instances and to learn a classifier accordingly. In TrAdaboost, source and target instances have opposite roles. Target training instances are weighted similarly as in AdaBoost, but when source training instances are wrongly predicted by the learned model due to distribution changes, it is assumed that they could be those that are the most dissimilar to the target instances and therefore TrAdaboost tries to decrease the weights of these instances in order to weaken their impact.

[4] http://ww.csie.ntu.edu.tw/ cjlin/libsvm/.

Algorithm 1 Transfer Adaptive Boosting with one source domain.

Require: Target training set $T_t = (X_t, Y)$ of size N_t; source training set $T_s = (X_s, Y)$, of size N_s; a base learning algorithm *Learner*; the maximum number of iterations M.

Ensure: Target learner $f : X_t \rightarrow Y$.

1: Initialize the initial source and target weight vectors, $\mathbf{w}_T^1 = (w_{t_1}^1, \ldots, w_{t_{N_t}}^1)$, $\mathbf{w}_S^1 = (w_{s_1}^1, \ldots, w_{s_{N_s}}^1)$,

2: Set $\mathbf{w} = (\mathbf{w}_T, \mathbf{w}_S)$, $\beta = 1/(1 + 2\sqrt{\ln\ N_t/M})$ and $T = (T_t, T_s)$.

3: **for** $r = 1, \ldots, M$ **do**

4: Normalize $\mathbf{w}^r = \mathbf{w}^r / |\mathbf{w}^r|$.

5: Call *Learner* on the training set T with the distribution $\mathbf{w}^r$ to find a hypothesis $f_r : X \rightarrow Y$ which minimizes error for $(T, \mathbf{w}^r)$.

6: Calculate the error of h_r on $T_t : \epsilon_r = min\left(\frac{1}{2}, \frac{1}{\sum_{i=1}^{N_t} w_{t_i}^r} \sum_{i=1}^{n} w_{t_i}^r \cdot [\![f_r(\mathbf{x}_i^t) \neq y_i]\!]\right)$.

7: Set $\beta^r = 1/2 \log((1 - \epsilon_r)/\epsilon_r)$.

8: Set $\Gamma^r = 2(1 - \epsilon_r)$.

9: Update the weight vectors:

$$w_{s_j}^{r+1} = \Gamma^r w_{s_j}^r \exp(-\beta\ [\![f_r(\mathbf{x}_j^s) \neq y_j]\!]),\ \mathbf{x}_j^s \in X_s,$$
$$w_{t_i}^{r+1} = w_{t_i}^r \exp(2\beta^r\ [\![f_r(\mathbf{x}_i^t) \neq y_i]\!]),\quad \mathbf{x}_i^t \in X_t.$$

10: **end for**

11: Output the aggregated estimate $f_{\text{tra}}(\mathbf{x}) = \left(\sum_{r=1}^{M} \beta^r f_r(\mathbf{x})\right)$. In a probabilistic setting, these scores are converted into probabilities $P(y_i = c | f_{\text{tra}}(\mathbf{x}_i))$.

A formal description of the framework is given in Algorithm 1 for the case of using one source domain, $N_S = 1$, but the extension to multiple sources is straightforward. The main idea is that at each iteration round, if a source training instance is mistakenly predicted, the instance may likely conflict with the target training data. Therefore, we decrease its training weight to reduce its effect through multiplying its weight by a strictly positive factor $\exp(-\beta [\![f_r(\mathbf{x}_i) \neq y_i]\!]) \leq 1$, where $[\![\cdot]\!]$ is the Iverson brackets denoting the indicator function that equals one if its argument is true and zero otherwise. We further add Γ^r term inspired by [6] to address some drawbacks of the initial TrAdaboost [12]. Due to this, in the next round, the mis-classified source training instances, which are dissimilar to the target ones, will affect the learning process less than in the current round. After several iterations, the source training instances that fit the target ones better will have larger training weights, while the source training instances that are dissimilar to the target ones will have much lower weights. These instances with large training weights tend to help the learning algorithm to train better classifiers.

As in the previous section, we consider N_{SC}=15 different compositions of available sources to be used as one. We use Algorithm 1 on different combinations by simply concatenating the sources to form the source training set X_s. Note the existence of an extension of TrAdaboost to cope with multiple source domains [29]. This extension probes at each iteration instances from different sources, in order to identify a source which reduces the error the most. In the context of the ImageClef DA challenge, our experience of using TrAdaboost with multiple sources has not been convincing. Concatenating sources was more beneficial that keeping them separately. This phenomenon can

be explained by the closeness between the source instances proposed for the challenge (see Figure 2). In the cases where source domains have very different distributions, keeping them separately might be a better strategy.

3.3 Metric learning methods

Another class of domain adaptation methods tries to jointly transform the source and target domains, either by trying to align domains using PCA projections [18,16,14] or by learning a metric which can bridge the two domains. For the challenge we experimented with two type of metric learning approaches. On one hand we extended a metric learning technique that optimizes KNN objectives [13,28] that we will denote by KNN_ML. On the other hand, we adopted for domain adaptation the Nearest Class Mean (NCM) and Nearest Class Multiple Centroids (NCMC) classifiers from [23].

Metric learning with KNN objectives. The aim of these approaches is to find a linear transformation $\boldsymbol{W}$ such that, in the new space, distances between examples from the same class are decreased and distances between images from different classes are increased. This is done generally by optimizing the sum of the losses on the training set[5]:

$$L_{qpn} = \left[1 + d_W(\mathbf{x}_q, \mathbf{x}_r) - d_W(\mathbf{x}_q, \mathbf{x}_n)\right]_+ = \left[1 + 2\mathbf{x}_q^\top \boldsymbol{W}^\top \boldsymbol{W}(\mathbf{x}_n - \mathbf{x}_p)\right]_+ \quad (1)$$

where $(\mathbf{x}_q, \mathbf{x}_p, \mathbf{x}_n)$ are triplets such as the image $\mathbf{x}_p$ is from the same class as $\mathbf{x}_q$ while $\mathbf{x}_n$ is from a different class, $d_W(\mathbf{x}_q, \mathbf{x}_i) = \| \boldsymbol{W}\mathbf{x}_q - \boldsymbol{W}\mathbf{x}_i \|_2^2$ is the Euclidean distance between image $\mathbf{x}_q$ and image $\mathbf{x}_i$ in the transformed space and $[a]_+$ denotes $max(0, a)$.

We consider only projections into a lower dimensional space, *i.e*. the dimension of the projected vector $\boldsymbol{W}\mathbf{x}$ is lower than the dimension of $\mathbf{x}$ (in our experiments we used target dimensions of 64 and 128). One advantage of this is to have less parameters to estimate, which is especially important when only relatively small amount of training examples are available; also it generally leads to better performances.

As optimizing the sum of losses over all possible triplets is typically unfeasible, we approximate the solution through stochastic gradient descent (SGD) [8] method, where at each step a set of random triplets are selected and $\boldsymbol{W}$ is updated with a fix learning rate and the sub-gradients:

$$\nabla_W L_{qpn} = 2 \left[\!\left[L_{qpn} > 0 \right]\!\right] \boldsymbol{W}\Big(X_{qnp} + X_{qnp}^\top\Big) \quad (2)$$

where $X_{qnp} = \mathbf{x}_q(\mathbf{x}_n - \mathbf{x}_p)^\top$. If instead of updating only by a single triplet, for each $\mathbf{x}_q$ we select a set of random positives-negative pair $\{(\boldsymbol{x}_{p_i}, \boldsymbol{x}_{n_i}), i = 1..m\}$, we have the same updating formula but $X_{qnp} = X_q X_{np}^\top$, where X_{np} is the matrix concatenating the $(\mathbf{x}_{n_i} - \mathbf{x}_{p_i})$ vectors, and X_q is the matrix concatenating m times the $\mathbf{x}_q$ vector.

[5] The second equality is true if we normalize our features to have L2 norm in the original space, what we have done in all our experiments

Metric Learning for Nearest Class Mean. The nearest class mean NCM classifier assigns a document to the class $c^* \in Y$ with the closest mean:

$$c^* = \underset{c \in Y}{\operatorname{argmin}}\, d_W(\mathbf{x}, \boldsymbol{\mu}_c) \quad \text{where} \quad d_W(\mathbf{x}, \boldsymbol{\mu}_c) = \| \boldsymbol{W}\mathbf{x} - \boldsymbol{W}\boldsymbol{\mu}_c \|_2^2, \tag{3}$$

where $d_W(\mathbf{x}, \boldsymbol{\mu}_c)$ is the Euclidean distance between an image $\mathbf{x}$ and the class mean $\boldsymbol{\mu}_c$ in the transformed space.

The main idea behind the metric learning for Nearest Class Mean (MLNCM) is to learn a linear projection matrix $\boldsymbol{W}$ such that, in this new space, the instances of the same class are closer to each other, and hence to the correct class mean than to the class centers of the other classes. To optimize such NCM performance objective in the projected space, Mensink *et al*. [23] proposed to formulate the NCM classifier as a multi-class soft-max regression problem using a probabilistic model where the probability for a class c given a feature vector $\mathbf{x}_i$ is defined as:

$$P(y_i = c|\mathbf{x}_i) = \frac{\exp\big(-.5 d_W(\mathbf{x}_i, \boldsymbol{\mu}_c)\big)}{\sum_{c' \in Y} \exp\big(-.5 d_W(\mathbf{x}_i, \boldsymbol{\mu}_{c'})\big)}. \tag{4}$$

Then, to learn the projection matrix W, the log-likelihood of the correct predictions are optimized over the training set using the stochastic gradient descend (SGD) algorithms [8]. At each step, $\boldsymbol{W}$ is updated with a fixed learning rate in the direction given by the gradient:

$$\nabla_W \mathcal{L} = \frac{1}{N} \sum_i \sum_{c \in Y} \big(P(y_i = c|\mathbf{x}_i) - [\![y_i = c]\!]\big) \cdot \boldsymbol{W} \cdot (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^\top, \tag{5}$$

where N is the number of training examples. At each iteration we are using only a random subset of the training data to update $\boldsymbol{W}$.

In [23], Mensink *et al*. proposed an extension to this method, the Nearest Class Multiple Centroids (NCMC) classifier that allows more flexible class representations by considering multiple cluster means (centroids) for each class. In their case, NCMC represents each class by a set of centroids instead of a single class mean. In our multiple domain case, instead of cluster means, we consider domain specific class means $\boldsymbol{\mu}_c^d$ (averaging over instances from the same class c and same domain d) and we assign a test instance to a given class based on a weighted soft-max distance to these domain specific class means:

$$P(y_i = c|\mathbf{x}_i, \boldsymbol{\mu}_c^d) = \frac{1}{Z} \sum_{d=1}^{N_d} w_d \exp\big(-.5 d_W(\mathbf{x}_i, \boldsymbol{\mu}_c^d)\big) \tag{6}$$

where N_d is the number of domains and $Z = \sum_c \sum_d w_d \exp\big(-.5 d_W(\mathbf{x}_i, \boldsymbol{\mu}_c^d)\big)$ is the normalizer. Adding domain specific weights w_d in Eq. 6 allows giving different importance to each domain (in our experiments we used $w_t = 2$ for the target domain $\mathcal{T}_l$ and $w_{s_k} = 1$ for the source domains).

To learn the projection matrix W, we maximize the log-likelihood of correct classification in Eq. 6, for which the gradient w.r.t. $\boldsymbol{W}$ is given by:

$$\nabla_W \mathcal{L} = \frac{1}{N} \sum_{i,c,d} \Big(p(\boldsymbol{\mu}_d^c|\mathbf{x}_i) - [\![c = y_i]\!] \frac{p(\boldsymbol{\mu}_d^c|\mathbf{x}_i)}{\sum_{d'} p(\boldsymbol{\mu}_c^{d'}|\mathbf{x}_i)} \Big) \boldsymbol{W}(\mathbf{x}_i - \boldsymbol{\mu}_c^d)(\mathbf{x}_i - \boldsymbol{\mu}_c^d)^\top, \tag{7}$$

where $p(\boldsymbol{\mu}_d^c|\mathbf{x}_i) = \frac{w_d}{Z} \exp\big(-.5 d_W(\mathbf{x}_i, \boldsymbol{\mu}_d^c)\big)$. Again, we use Stochastic Gradient Descent [8] with fixed learning rate where at each iteration we sample a random subset from the training set ($2 * N_C$ was used in the experiments). We denote this learning method by NCMC_ML.

Learning strategies. Let us now suppose that we have the labeled target dataset $\mathcal{T}_l$, the unlabeled target data set $\mathcal{T}_u$, and the source training set (that can contain one or several source domains $\mathcal{S}_1, \ldots, \mathcal{S}_{N_s}$). We experimented with the above mentioned metric learning approaches using 3 different strategies and different source configurations SC_i.

1. In the first case we simply merged the source combination SC_i with the target training data $\mathcal{T}_l$ and directly optimized the loss in Eq. 2, Eq. 5 or Eq. 7 using all training examples. The rationale behind this is the following. By decreasing interclass distances independently from the domains, allows to exploit more efficiently labeled images from source domains in the projected space to classify target examples and hence it yield to increased performance of distance based classifiers. The methods from this first strategy are denoted by KNN_ML, MLNCM_ML respectively MLNCMC_ML.
2. As we have much less labeled target examples than source ones, their presence is much rearer in the above mentioned random sampling processes. Hence in the iterative approaches they will have less influence in the learning process than the much larger amount of source images. Therefore as a second strategy, we propose to refine $\boldsymbol{W}$ obtained with the first strategy such that we impose at each iteration to have in the sample examples from $\mathcal{T}_l$. In the case of KNN, this is done by considering at each step $\mathbf{x}_q^t \in \mathcal{T}_l$ and sampling randomly a set of randomly sampled positive/negative image pairs from the source and then updating $\boldsymbol{W}$ using the triplet sets $(\mathbf{x}_q^t, \mathbf{x}_{r_j}, \mathbf{x}_{n_j}), j = 1..N_c$. In the case of NCM and NCMC we simply ensure that at least one target image from $\mathcal{T}_l$ is present in the selected batch. The methods corresponding to this second strategy are denoted KNN_MLt, NCM_MLt and NCMC_MLt respectively.
3. Finally, inspired by the adaptive learning proposed in [27], we consider an iterative learning strategy where we further adjust the learned metric with the second strategy. To do this, at each step and for each class we add target instances from the unlabeled set $\mathcal{T}_u$ and remove source instances as follows. For each class c we add the target instance $\mathbf{x}_i^t$ for which $P(y_i = c_i^*|\mathbf{x}_i^t) - P(y_i = c_i^\ddagger|\mathbf{x}_i^t)$ according to Eq. 4 is the largest difference, where $c_i^* = c$ is the predicted label of $\mathbf{x}_i^t$ and $c_i^\ddagger$ is the second predicted label of $\mathbf{x}_i^t$. Also, for each class c we remove the source image $\mathbf{x}_j^s$ from SC_i for which $c_j^* = c$ and $P(y_j = c_j^*|\mathbf{x}_j^s) - P(y_j = c_j^\ddagger|\mathbf{x}_j^s)$ is the

lowest difference. Then we refine $\boldsymbol{W}$ using the updated training set. We do several rounds until the stopping criteria is reached[6]. The methods corresponding to the third strategies are denoted by KNN_MLDA, NCM_MLDA and NCMC_MLDA respectively.

Classification strategies. In the projected space we have different options to predict the class labels of the target images. As we learned a metric, we can use any distance based classifiers such as KNN, NCM or NCMC based classifications as they do not require further parameter tuning. The only parameter k of the KNN can be fixed (*e.g.* we used 3 in all experiments) or cross validated. NCM has no parameter and the parameters of the NCMD are the weights w_d that can be fixed or tuned. In our experiments we used fixed weights. Note that we experimented also with training Multi-Class SVM in the projected space, but the performance was similar to or below the results obtained in the original space. As our experiments on the dataset provided in phase 1[7] have shown that NCMC in general performs better than KNN and NCM (up to 3-5%) we only retained the NCMC classifier to compute the class probabilities (using Eq. 6) independently which metric learning method or strategy was used to compute $\boldsymbol{W}$.

Furthermore, from our experiments in phase 1 it was not clear which source combinations are better when we evaluated the NCMC performance in the projected space. Furthermore, the combinations that yielded best accuracies in the original space (using SVM classifiers) were not always the ones that proved to be the best after projection. Therefore, for each of the above described methods, we learned a metric using each source combination SC_i individually and computed class predictions $P(y_i = c^*|f_{\text{ml}}^{SC_i}(\mathbf{x}_i^t))$ for the test set T_u in the corresponding projected space. Then we computed the late fusion of class predictions over all source combination to get final predictions given a metric learning method. These classifiers denoted by f_{ml}, were used in two modes. Two runs have been used in submission to predict directly the labels, namely, MLNCM_MLDA_128_it200_e0.1_p025 and MLNCMC_ML_128_it200_e0.1_p025. Also they were used by ensemble methods in combination with other models to generate the combin_* runs.

4 Ensemble Method

The combination of the decisions of several classifiers has been proposed as a means of improving the accuracy achieved by any of them. The reasons for combining the outputs of multiple classifiers are compelling, because different classifiers may implicitly represent different useful aspects of a problem, or of the input data, while none of them represents all useful aspects. In the context of instance classification, the idea of combining the decisions of several classifiers has been well explored.

[6] The classification performance on the initial training set incurs a stronger degradation than a predefined tolerance threshold or no more target image can be added or source removed.

[7] We did 11 fold cross validation on the training set in phase 1. We split the provided the 600 test instances into 10 folds such that each folds contained 5 document per class and we added the training set as 11th fold.

In contrast to approaches which combine models derived from multiple versions of the same learning method, the specific focus of this paper is on ensemble methods that are able to combine the decisions of multiple classifiers of different types, so-called heterogeneous sets of classifiers.

Previous sections described three domain adaptation methods which generate one or more classifiers for target instances, namely, obtained by the brute force (f_{mcsvm} and f_{bsvm}), transfer adaptive boosting (f_{tra}) and metric learning (f_{ml}). They form a pool F of several classifiers from each group $F = \{f_1, \ldots, f_{N_f}\}$ where any classifier's output is class scores or class probabilities. We will denote by $g(f_k, \mathbf{x}_i^t)$ the predicted label of f_k for $\mathbf{x}_i^t$ and by $p(y_i = c|f_k(\mathbf{x}_i^t))$ the class probability scores. When the classifier does not provide such score we can define $p(y_i = c|f_k(\mathbf{x}_i^t)) = 1$ when $c = g(f_k, \mathbf{x}_i^t)$ and zero otherwise.

The first method we tested was the (unweighted) majority voting that combines decisions of individual classifiers, so the global ensemble prediction for target instance $\mathbf{x}^t$ is:

$$c^* = \text{argmax}_{c \in Y} \sum_{f_k \in F} [\![g_k(f_k, \mathbf{x}_i^t) = c]\!]$$

In the probabilistic setting, we sum up class probabilities and report the class with the highest probability:

$$c^* = \text{argmax}_{c \in Y} \sum_{f_k \in F} P(y_i = c|f_k(\mathbf{x}_i^t))$$

The majority voting method is based solely on the output label or probabilities computed by each classifier. In the case of weighting majority voting, classifiers' opinions may be taken differently, depending on how accurate they have been in the past. Below we establish weights proportional to each classifier's accuracy, so each classifier's output is considered according to its past performance.

In the probabilistic setting, the fact that instance $\mathbf{x}_i^t$ belongs to class c given that classifier f_k output label c' for $\mathbf{x}$, *i.e.* $g(f_k, \mathbf{x}_i^t) = c'$, has an uncertainty which can be expressed as the conditional probability $p(y_i = c|g(f_k, \mathbf{x}_i^t) = c')$. We may use the confusion matrices to approximate the conditional probabilities for all classes $c \in Y$ given a classifier $f_k \in F$.

Making assumption of the classifier independence, we use the Bayesian rule to estimate the posterior probability of the class c for the target instance $\mathbf{x}_i$ from the output of classifiers $f_k \in F$:

$$P(y_i = c|\mathbf{x}_i^t) = \frac{\prod_{c'' \in Y} P\big(y_i = c|g(f_k, \mathbf{x}_i^t) = c''\big)}{\sum_{c' \in Y} \prod_{c'' \in Y} P\big(y_i = c'|g(f_k, \mathbf{x}_i^t) = c''\big)}$$

Finally, the class $c^* = \text{argmax}_{c \in Y} P(y_i = c|\mathbf{x}_i^t)$ (with the highest probability) is assigned as the weighted majority vote to instance $\mathbf{x}_i$.

5 Evaluation results

Both individual and ensemble methods have been tuned by XRCE team during the phase 1 to select best strategies of the challenge and applied to the submission data

(phase 2). Among multiple evaluation runs, ten most promising results have been selected for the submission, they all are reported in Table 1. For each submission, the table reports the accuracy, score, divergence and a comment on the method used.

The divergence has been proposed to measure the deviation of the submitted prediction vector from a vector with equi-weighted classes:

$$div = \sum_{c \in Y} \left| \mathrm{Card}(\{i | g(f, \mathbf{x}_i^t) = c\}) - \frac{N}{N_c} \right|$$

where N is the number of test images, N_c the number of classes, $\{i|g(f, \mathbf{x}_i^t) = c\}$ is the set of target instances for which the classifier f predicts class c, $\mathrm{Card}(A)$ is the cardinality of the set A and $|a|$ is the absolute value of a. Under the assumption on the equal class distribution in the submission dataset, it may make sense to choose a run whose prediction tends to minimize the divergence.

Table 1. Ten runs submitted by XRCE team.

Place	Score	Accuracy	Run Name	Divergence	Comment
1	228	38.0	combin6_Np20	108	UMV
2	228	38.0	combin3_Np18	108	UMV
3	226	37.67	combinAll6_Np19	164	UMV
4	217	36.17	combin6A_Np19	78	UMV + min divergence
5	214	35.67	MLNCM_MLDA_128_it200_e0.1_p025	174	ML
6	212	35.33	combinAll7A_Np19	134	WMV
7	208	34.67	combin8A_Random_Np25	78	WMV + min divergence
8	185	30.83	MLNCMC_ML_128_it200_e0.1_p025	168	ML
9	182	30.33	combin2_Np10	134	TrA+UMV
10	158	26.33	svmBoost_Mul_Power_f60	186	TrA

Analyzing the ten submissions made (as well as the non-submitted runs), allowed us to make the following conclusions on the selected strategy and performance of different components of the image classification system we have built for the challenge:

1. Among individual domain adaptation methods, the brute force performed poorly as expected. It did not merit any individual submission, but participated in various ensembles of classifiers.
2. TrAdaboost (TrA) with SVM as a week learner and the Metric Learning (ML) performed reasonably well, each merited two individual submissions.
3. Ensembles of heterogeneous classifiers has turned to be a right strategy to boost the overall performance.
4. Unweighted majority vote (UMV) on a small selection of classifiers performed the best; these selections include 3 to 6 top performing classifiers from each group of classifiers.

5. Weighted majority vote (WMV) works well on large sets of classifiers but underperformed against the manual classifier selection[8] with unweighted majority vote.
6. Divergence minimization did not play any important role; making hypotheses on the class distribution and incorporating this prior knowledge in the classification system should be done in a more systematic way.

6 Conclusion

The strategy of using heterogeneous methods for domain adaptation turned to be a right one. It has allowed our team to build a image classification system in the target domain with an important knowledge transfer from available source domains. Ensembles of heterogeneous classifiers aggregated with different majority voting scenarios has allowed to get high accuracy in the submission runs and to eventually win the ImageCLEF Domain Adaptation competition. The thoughtful analysis of obtained results has also allowed to identify new directions in domain adaptation for image classification.

References

1. Bing. `http://vlg.cs.dartmouth.edu/projects/domainadapt/`.
2. Caltech-256. `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`.
3. Imagenet12. `http://image-net.org/challenges/LSVRC/2012/index`.
4. Pascal-voc12. `http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012/index.html`.
5. Sun. `http://groups.csail.mit.edu/vision/SUN/`.
6. Samir Al-Stouhi and ChandanK. Reddy. Adaptive boosting for transfer learning using dynamic updates. In D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, volume LNCS 6911, pages 60–75. 2011.
7. Oscar Beijbom. Domain adaptations for computer vision applications. Technical report, 2012.
8. L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010.
9. B. Caputo, H. Müller, J. Martinez-Gomez, M. Villegas, B. Acar, N. Patricia, N. Marvasti, S. Üsküdarlı, R. Paredes, M. Cazorla, I. Garcia-Varea, and V. Morell. ImageCLEF 2014: Overview and analysis of the results. In *CLEF proceedings*. 2014.
10. Barbara Caputo and Novi Patricia. Overview of the ImageCLEF 2014 Domain Adaptation Task. In *CLEF 2014 Evaluation Labs and Workshop, Online Working Notes*, 2014.
11. G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.
12. Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *ICML*, pages 193–200. ACM, 2007.
13. Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML*, 2007.

[8] Under manual selection we meant that the selection was made based on our intuitions according to some of the lessons we learned in phase 1.

14. Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, 2013.
15. Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.
16. B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.
17. Boqing Gong, Kristen Grauman, and Fei Sha. Reshaping visual datasets for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1286–1294, 2013.
18. Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *ICCV*, pages 999–1006. IEEE, 2011.
19. Judy Hoffman, Brian Kulis, Trevor Darrell, and Kate Saenko. Discovering latent domains for multisource domain adaptation. In *Computer Vision–ECCV 2012*, pages 702–715. Springer, 2012.
20. I-Hong Jhuo, Dong Liu, D. T. Lee, and Shih-Fu Chang. Robust visual domain adaptation with low-rank reconstruction. In *CVPR*, pages 2168–2175, 2012.
21. Jing Jiang. A literature survey on domain adaptation of statistical classifiers. Technical report, 2008.
22. B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR*, pages 1785–1792, June 2011.
23. T Mensink, J Verbeek, F Perronnin, and G Csurka. Distance-based image classification: Generalizing to new classes at near zero cost. *PAMI*, 11, 2013.
24. Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.
25. Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: An overview of recent advances. *IEEE Transactions on Geoscience and Remote Sensing*, 52(2):1019–1029, 2007.
26. Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *ECCV*, volume 6314 of *Lecture Notes in Computer Science*, pages 213–226. Springer Berlin Heidelberg, 2010.
27. Tatiana Tommasi and Barbara Caputo. Frustratingly easy nbnn domain adaptation. In *ICCV*, 2013.
28. K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009.
29. Zhijie Xu and Shiliang Sun. Multi-source transfer learning with multi-view adaboost. In Tingwen Huang, Zhigang Zeng, Chuandong Li, and ChiSing Leung, editors, *NIPS*, volume 7665 of *Lecture Notes in Computer Science*, pages 332–339. Springer Berlin Heidelberg, 2012.