# Project4: Domain Adaptation for Image Classification

Jieyu Li, Dongyue Li, Hongbin Chen, Haoxuan Wang
Course: CS245 - Data Science Principles

***Abstract*— This is a report on the experiments we have done for image classification using domain adaptation. Domain adaptation faces the problem of inconsistent training sample space with testing sample space. We try to overcome this problem by using different algorithms, including traditional methods as well as deep methods. Classification is done by SVM or a neural network layer. We further compare the performance of these different methods and discuss their reasons for such performances.**

## I. Introduction

### A. Project Description

Domain adaptation is a widely discussed machine learning problem. We use Office-Home dataset to do image classification using different domain adaptation techniques. Traditional and deep models are all included. Comparison is done between their performances to help us get deeper insight into these algorithms and the dataset.

### B. Dataset

Office-Home dataset consists of $65$ categories of office depot from 4 domains (Art, Clipart, Product, Real-world). Feature extraction step is done by using ResNet50, achieving 2048-dimension deep learning features. The original dataset is split into three settings: $Art \rightarrow RealWorld$, $Clipart \rightarrow RealWorld$, $Product \rightarrow RealWorld$, and evaluations are done separatedly.

### C. Experiment Settings

1) Environment: Python3.6, Matlab
2) Packages: Sklearn, Seaborn, Matplotlib

## II. Algorithms

In this section, we explain the theory of different methods and discuss their performances. Also, comparison between different datasets and methods will also be done.

### A. Baseline

Domain adaptation actually acts as a preprocssing step in the task of classification. Thus, we need a baseline to evaluate how our algorithms effect the final results. Our baseline is simple: Just train SVMs on the source domain and apply it directly to the target domain for accuracy evaluation.

### B. Domain Invariant Projection

DIP [1] encloses the gap between source domain and target domain by projecting them to a common subspace. It learns a projection $W$ to make the projected domains as close to each other as possible. The optimization goal can be written as:

$$\begin{aligned} &min_W tr(K_W L) \\ &s.t. W^T W = I \end{aligned} \tag{1}$$

In DIP, kernelization is done after projection to convert the transformed data space to a more appropriate space.

### C. Transfer Component Analysis

Similar with DIP, TCA [2] also learns a projection from the original space to a latent subspace to minimize the distance between source domain and target domain. However, TCA adopts kernelization before the projection. The optimization goal is:

$$\begin{aligned} &min_W tr(W^T KLKW) \\ &s.t. W^T W = I \end{aligned} \tag{2}$$

### D. Subspace Alignment

SA [3] learns a mapping function to align the source subspace with the target one. The solution to this optimization problem can be obtained in a simple closed form, which also makes the algorithm faster to be solved. In practice, we adopt different dimensions of latent space dimensions

and observe how the performance changes. The goal can be explained in math as follows:

$$min_M||MP_s - P_t||_F^2 = min_M||MP_sP_s^T - P_tP_s^T||_F^2 \\ = min_M||M - P_tP_s^T||_F^2 \quad (3)$$

Where $P_s$ is the result of $X_s$ after PCA, and $P_t$ is the result of $X_t$ after PCA. By solving the equation, $M = P_tP_s^T$. We can further derive the equation $M\widetilde{P_s} = \widetilde{P}_t\widetilde{P_s^T}\widetilde{P_s}$ and we can get the final projection: $\widetilde{X_s} = \widetilde{P}_t\widetilde{P_s^T}\widetilde{P_s}X_s$ and $\widetilde{X_t} = \widetilde{P}_tX_t$.

### E. CORrelation Alignment

Different from aligning the source and target subspaces directly, CORAL [4] aligns the second order statistics (covariances) of target and source domains. By defining $A$ as the projection matrix, we are able to formalize the optimization goal:

$$min_A||\hat{C}_s - C_t||_F^2 \quad (4)$$

where $\hat{C}_s = A^TC_sA$.

The optimal solution can be obtained by using SVD,

$$A* = U_s\Sigma_s^{-\frac{1}{2}}U_s^TU_t[1:r]\Sigma_t[1:r]^{\frac{1}{2}}U_t[1:r]^T \quad (5)$$

### F. Sampling Geodesic Flow

SGF [5] is a typical example of interpolation on the manifold. By inserting points on the manifold connecting the source domain and target domain, we create a path from source to target. This path is useful for making the two domains comparable to each other.

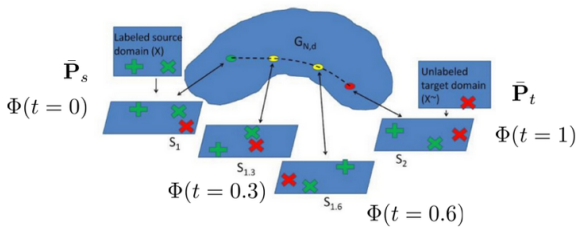



Fig. 1. Theory of SGF

### G. Kernel Mean Matching

KMM [6] is a method using the idea of sample selection. It assigns different weights to the source domain samples so that the samples more close to target domain are considered to be more important. According maximum mean discrepancy, this is equivalent to minimizing the distance between two centers. Two steps are need while solving the problem, in which the goal of the first one is a condition of the second one.

The first step can be formulated as:

$$min_{\beta_i} = ||\frac{1}{n_s}\sum_{i=1}^{n_s}\beta_i\phi(x_i^s) - \frac{1}{n_t}\sum_{i=1}^{n_t}\phi(x_i^t)||^2 \quad (6)$$

After obtaining $\beta_i$, we can further reach step 2, which is:

$$min_{W,b,\epsilon_i}\frac{1}{2}||W||^2 + C\sum_i\beta_i\epsilon_i \\ s.t. y_i(W^T\phi(x_i) + b) \geq 1 - \epsilon_i, \epsilon_i \geq 0, \forall i \quad (7)$$

### H. Deep Methods

The deep methods we use are various, including: DAN (Deep Adaptation Networks), DeepCORAL, TCP (Transfer Channel Pruning). The results can be seen in Table I.

## III. Experiment Results

TABLE I
CLASSIFICATION ACCURACY

| | $A \rightarrow R$ | $C \rightarrow R$ | $P \rightarrow R$ |
|---|---|---|---|
| Baseline | 75.48 | 66.46 | 72.96 |
| DIP | 74.10 | 62.07 | 69.81 |
| TCA | 69.02 | 61.42 | 67.51 |
| SA (512-d) | 75.25 | 65.08 | 72.64 |
| SA (1024-d) | 75.60 | 66.00 | 72.91 |
| SA (2048-d) | 75.48 | 66.46 | 72.96 |
| CORAL | 74.87 | 66.49 | 72.69 |
| SGF | 72.56 | 61.89 | 69.14 |
| KMM | 75.53 | 66.55 | 74.36 |
| DAN | 75.52 | 66.24 | 72.14 |
| DeepCORAL | **75.66** | **66.79** | 73.05 |
| TCP | 75.94 | 66.72 | **73.91** |

## IV. Further Discussions

### A. Discussion of DIP, TCA, SGF

The first observation for these three methods is that: Their accuracy on the $A \rightarrow R$, $C \rightarrow R$ and $P \rightarrow R$ datasets all decreased. TCA is an extension

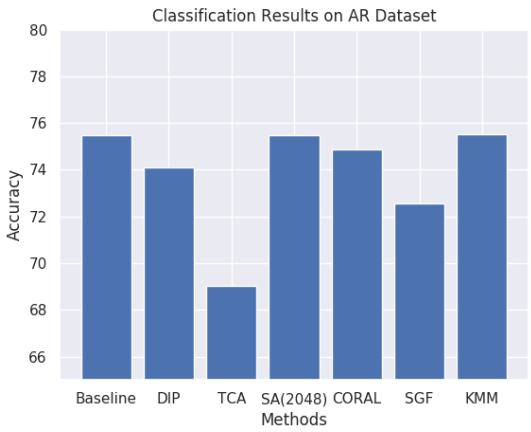



Fig. 2. $A \rightarrow R$ Dataset Results

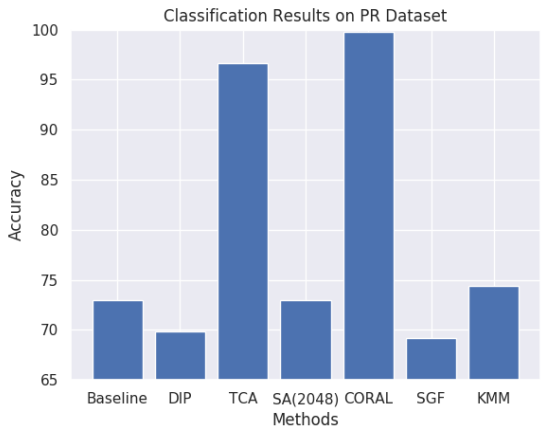



Fig. 4. $P \rightarrow R$ Dataset Results

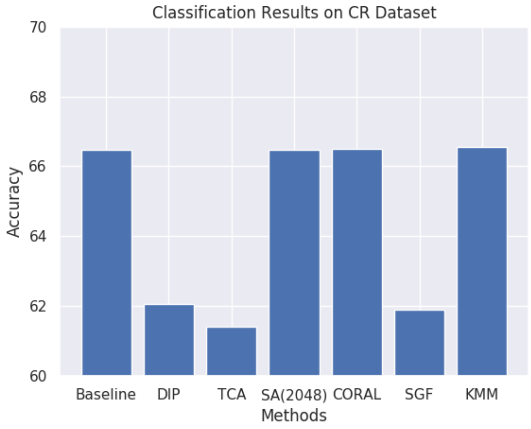



Fig. 3. $C \rightarrow R$ Dataset Results

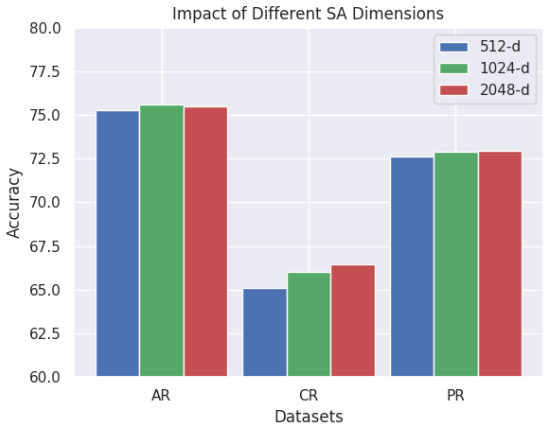



Fig. 5. Impact of SA's Different Dimensions

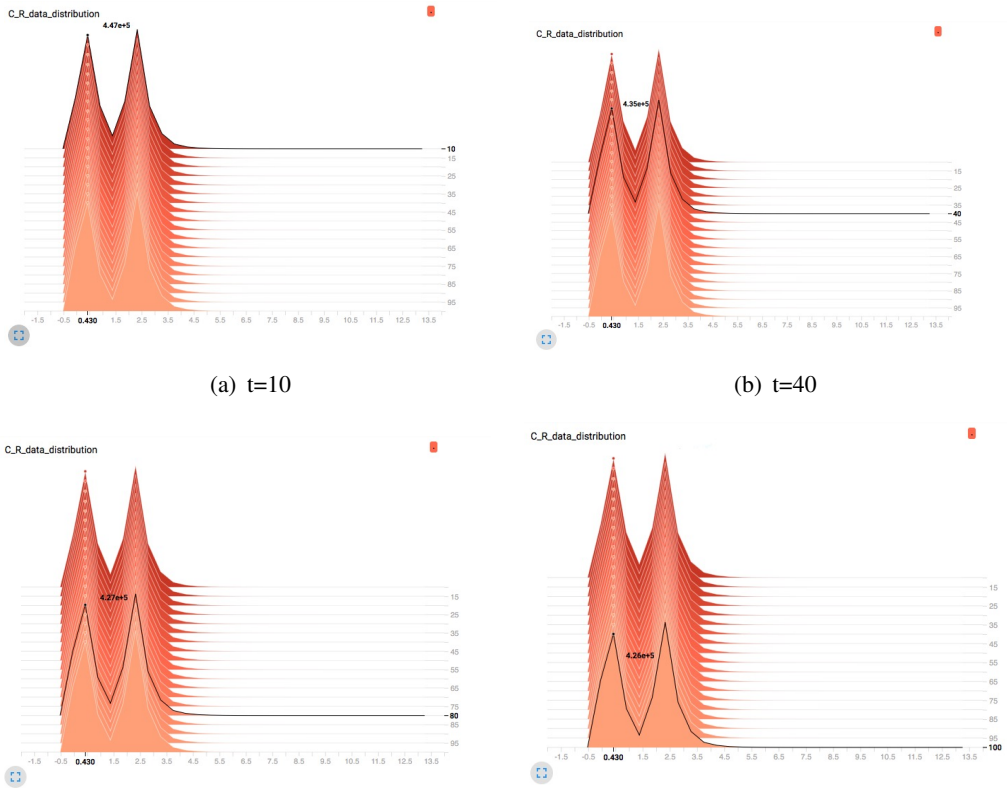
of DIP, SGF and DIP all use kernel tricks, thus we guess that the reason for the worse performance is that the features extracted by ResNet are already well structured. When we are doing kernelization, some important information of the original data is lost, correlations are ignored among various dimensions, thus the performance worsed.

The second observation is that in our experiments, we found that the peformance for the methods are sensitive to the parameters we choose. The first GridSearch scope only lead us to accuracy of around $5\%$, but when we changed the scope to another range, the performance increases dramatically, leading to our present results.

Thirdly, though the accuracy of all these four methods decreased, the performance of TCA decreased the most. This might be due to the fact that TCA uses a kernel that closes the distance between the source domain and the target domain. This kernel might produce much noise to the actual distribution, resulting in bad performance.

### *B. Discussion of KMM*

KMM's performance is relatively better. This is because the second step of KMM is "aimed at" doing SVM. The first step of KMM (all the mathematical operations) is a preparation for doing SVM. Thus, the kernelization tricks is instead helpful to the classification. Also, KMM has a step that is similar to doing SVM. In this way, we are actually doing 2 SVMs in our experiments, which makes the results more robust.




(a) t=10 (b) t=40

(c) t=80 (d) t=100

Fig. 6. Change of Data Distribution with t

### C. Discussion of SA

SA uses PCA to find new subspaces of source and target domain first, and then project source domain to the target domain. The features extracted by the ResNet might be too well organized, thus not much useful operations are done when doing PCA on the features. Thus the results does not change too much.

We have also done experiments disucssing the impact of different dimensions of the SA on the performance accuracy. The best results are mainly achieved when the final dimension is $2048$. Thus, we reach the conclusion that the more dimensionality reduction we do, the more information is lost from the original features.

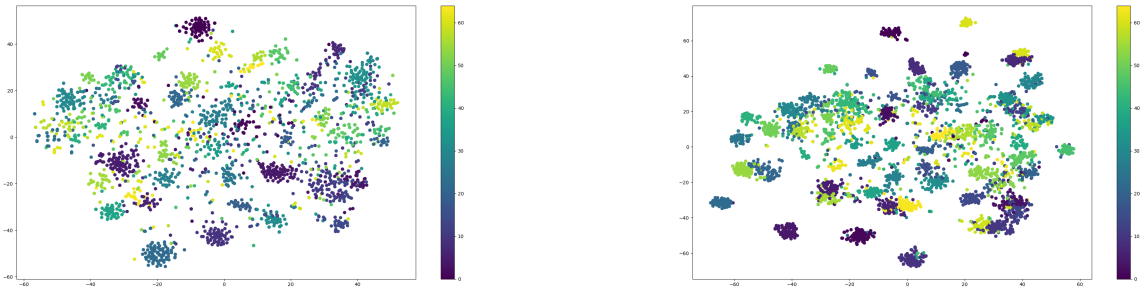
We also plot the data distribution of source domain and target domain of the $A \rightarrow R$ dataset. For better visualization, we use TSNE to do dimension reduction and graph generation. The original distirbution over $65$ categories are shown in Fig.7., while the distribution of data domain after performing SA is shown in Fig.8. From the graphs, we find that:

- The categories in target domain are more closely clustered wit each other, which indicates there are large categories (such as birds) and small categories in these large ones (such as eagles).
- After performing SA, the categories are more closely clustered
- We can see that SA does not ”shift” source domain to target domain significantly, which explains why SA does not perform so well.

### D. Discussion of CORAL

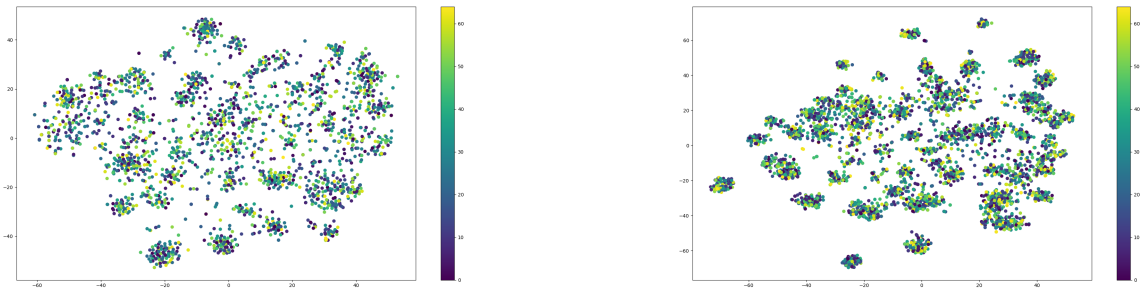
CORAL does not uses the kernel trick and uses the second-order information of the features. Thus the information lost from these methods are less, and the performance is similar with the baseline.

The variance makes the changes more observable, and we can see that the distribution (described by black lines) are changing via the change of $t$.



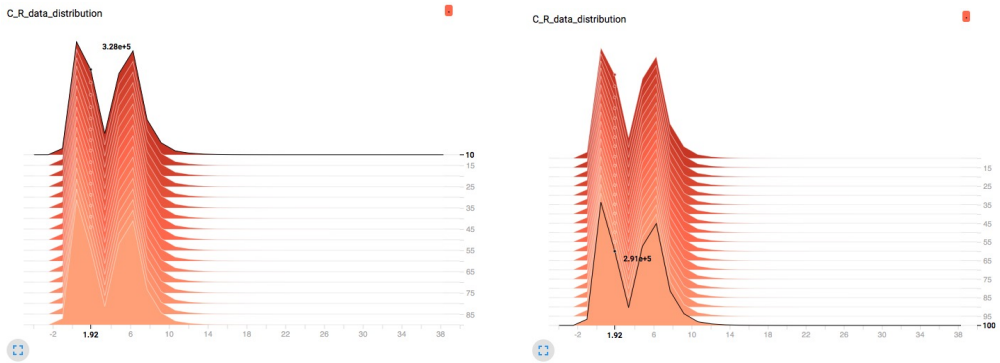
(a) Source Domain (b) Target Domain

Fig. 7. TSNE of Source Domain and Target Domain



(a) Source Domain (b) Target Domain

Fig. 8. TSNE of Source Domain and Target Domain After SA




(a) t=10 (b) t=100

Fig. 9. Change of Data Distribution with t (Larger Variance)

### *E. Discussion of Deep Methods*

Deep methods proved their superiority, with their effectiveness in representing information and finding hidden correlations. However, due to the chracteristics of our features, the deep methods are not performing very satisfying.

### *F. Concluding Discussion*

There are conditions when the amount of data from source domain is less than the amount of data of target domain. Thus, we conducted an

experiment to exmaine the impact of the amount of data from source domain towards the target domain. We used $t\%$ ($t = 10, 20, 30...$)of source data domain to do domain adaptation on the target domain and see the distribution of projected source domain data.

For simplicity of observation, we have the following settings:

- We useat most $400$ data samples from the projected source domain, and $400$ data samples from the target domain. We do not add data with the increment of $t$ so that the top point of the graph we plot do not rise with $t$, and the distributions can be properly compared.
- In the graphs, we do translation on the distributions to set them apart for observation, and the translation does not change the inner relationships of the distribution.

The experiment results are shown in Fig.6.

From the graph, we find that with the increase of $t$, the distribution is monotonically increasing. Thus, we reach the conclusion that the more data samples we train on, the better performance we can achieve.

To further prove our conclusions, we conducted another experiment. In this second experiment, we tried to change the variance over the distributions. The variance in the new distribution is enlarged, and we achieve the result in Fig.9.

## V. Conclusion

In this project, we do domain adpatation on three different datasets. We use different methods for classification, including traditional methods such as DIP as well as deep methods such as DeepCORAL. Some of them showed slight improvement in performance while most of them did not. The main reason for this phenomenon is that the features are extracted from ResNet50 and are already very descriptive and informative. Thus, our domain adaptation methods are usually not doing so well since not much useful information can be further extracted. Whlie our experiment results are not satisfying, we still learned much from applying these well-known domain adaptation methods to real datasets, which is interesting and inspiring.

## References


[1] Baktashmotlagh M , Harandi M T , Lovell B C , et al. Unsupervised Domain Adaptation by Domain Invariant Projection[C]. ICCV. IEEE Computer Society, 2013.

[2] S. J. Pan, I. W. Tsang, J. T. Kwok and Q. Yang, "Domain Adaptation via Transfer Component Analysis," in IEEE Transactions on Neural Networks, vol. 22, no. 2, pp. 199-210, Feb. 2011. doi: 10.1109/TNN.2010.2091281

[3] Fernando B , Habrard A , Sebban M , et al. Unsupervised Visual Domain Adaptation Using Subspace Alignment[C]. International Conference on Computer Vision (ICCV). IEEE, 2013.

[4] Morerio, Pietro & Murino, Vittorio. (2017). Correlation Alignment by Riemannian Metric for Domain Adaptation.

[5] Gopalan R , Li R , Chellappa R . Domain adaptation for object recognition: An unsupervised approach[C]. 2011 International Conference on Computer Vision. IEEE, 2012.

[6] Gretton A , Bernhard Schlkopf, Huang J . Correcting Sample Selection Bias by Unlabeled Data[J]. Advances in Neural Information Processing Systems, 2007, 19:601-608.