Principles of Data Science Project 4 Domain Adaptation

Hongzhou Liu
517030910214
deanlhz@sjtu.edu.cn

Xuanrui Hong
517030910227
hongxuanrui.1999@sjtu.edu.cn

Qilin Chen
517030910155
1017856853@sjtu.edu.cn

***Abstract*—In this project, we tried different domain adaptation methods on the Office-Home dataset, which contains 65 categories of things from 4 domains. The four domains are Art, Clipart, Product and Real-World. In our experiments, we take Art, Clipart and Product as source domains and Real-World as target domain. For traditional methods, we tried KMM, CORAL, GFK, TCA, and EasyTL. For deep learning methods, we only tried DAN due to the scarce of computation resources and time limitation. We compared performances among those methods and discussed the difference among them.**

***Index Terms*—Domain Adaptation, Transfer Learning**

I. Introduction

In this project, we tried different unsupervised domain adaptation methods on the Office-Home dataset, which contains 65 categories of things from 4 domains. The four domains are Art, Clipart, Product and Real-World. There are two parts in this section. Firstly, we will introduce several traditional transfer learning methods we used in our project, including KMM, CORAL, GFK TCA and EasyTL. Then we will introduce deep transfer learning method DAN to compare with the traditional transfer learning methods.

A. Traditional Transfer Learning Methods

1) Kernel Mean Matching (KMM): Huang et al. proposed Kernel Mean Matching [1] to estimate the probability density of data samples. The ultimate goal is to weight the samples to make the probability distribution of sourch domain and target domain closer. The core of the method is the measurement of the difference in the distribution of the two areas. Specifically, the weighted data of the two areas is mapped into the RKHS, and the difference between the average values of the samples in each area is obtained. This distribution measurement method is called the maximum mean difference. After learning the weights of samples with similar distributions, standard machine learning algorithms can be trained and predicted. For example, using SVM:

$$step1 : \min_{\beta_i} \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \beta_i \phi(x_i^s) - \frac{1}{n_t} \sum_{i=1}^{n_t} \phi(x_i^t) \right\|^2 \tag{1}$$

$$step2 : \min_{w,b,\xi_i} \frac{1}{2} \|w\|^2 + C \sum_i \beta_i \xi_i \tag{2}$$

$$s.t. y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \tag{3}$$

2) CORrelation ALignment (CORAL): The goal of CORAL is to minimize the second-order distance between source domain and target domain, that is, covariance. It linearly transforms the second-order statistics of source distribution and target distribution. Although it is very simple, it works well for unsupervised areas. However, it relies on linear transformation, not end-to-end: it needs to first extract features, apply the transformation, and then train the SVM classifier in a separate step.

The main formula is as follows:

$$\min_A \left\| \widehat{C_s} - C_t \right\|_F^2 = \min_A \left\| A^T C_s A - C_t \right\|_F^2 \tag{4}$$

$$C_s = U_s \Sigma_s U_s^T, C_t = U_t \Sigma_t U_t^T \tag{5}$$

The optimal solution:

$$A^* = U_s \Sigma_s^{-\frac{1}{2}} U_s^T U_t[1:r] \Sigma_t[1:r]^{\frac{1}{2}} U_t[1:r]^T \tag{6}$$

$$r = min(rank(C_s), rank(C_t)) \tag{7}$$

3) Geodesic Flow Kernel (GFK): Geodestic Flow Kernel's [2] main idea based on the Domain Adaption method of geodetic flow cores (preferably displayed in color). They embed the source data set and the target data set into the Glassman manifold. Then, they construct a flow between two points of the geodesic line and integrate the infinite flow $\Phi(t)$ in the subspace. Specifically, the original features are projected into these subspaces to form an infinite dimensional feature vector z_h. The inner product between these feature vectors defines a kernel function that can be calculated in a closed form on the original feature space. The kernel encapsulates incremental changes between subspaces, and these changes are the basis for the differences and commonalities between the two domains. Therefore, the learning algorithm uses this kernel to derive a low-dimensional representation of domain invariance. The main idea is shown in fig .I-A3. And the approach consists of the following steps:

1) Determinetheoptimaldimensionalityofthesubspaces to embed domains.
2) Construct the geodesic curve. Let P_S, P_T denote the two sets of basis of the subsapces for the source and target domains.

$$\Phi(t) = P_S U_1 \Gamma(t) - R_S U_2 \Sigma(t) \tag{8}$$

$$\boldsymbol{P}_{\mathcal{S}}^{\mathrm{T}} \boldsymbol{P}_{\mathcal{T}} = \boldsymbol{U}_1 \boldsymbol{\Gamma} \boldsymbol{V}^{\mathrm{T}}, \quad \boldsymbol{R}_{\mathcal{S}}^{\mathrm{T}} \boldsymbol{P}_{\mathcal{T}} = -\boldsymbol{U}_2 \boldsymbol{\Sigma} \boldsymbol{V}^{\mathrm{T}} \qquad (9)$$

3) Compute the geodesic flow kernel.

$$\left\langle z_i^{\infty}, z_j^{\infty} \right\rangle = \int_0^1 \left(\Phi(t)^{\mathrm{T}} x_i \right)^{\mathrm{T}} \left(\Phi(t)^{\mathrm{T}} x_j \right) dt = x_i^{\mathrm{T}} G x_j \qquad (10)$$

4) Use the kernel to construct a classifier with labeled data.

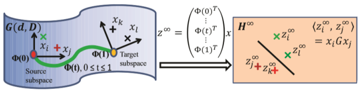


Fig. 1.

4) Transfer Component Analysis (TCA): For domain adaptation, Transfer Component Analysis (TCA) [3] tries to learn some transfer components across domains in a Reproducing Kernel Hilbert Space (RKHS) using Maximum Mean Discrepancy (MMD). It minimizes the distance between domain distributions by projecting data onto the learned transfer components.

The basic assumption of TCA is

$$P\left(X_s\right) \neq P\left(X_t\right)$$

where X_s denotes source domain data and $P\left(X_s\right)$ denotes its marginal distributions, X_t denotes target domain data and $P\left(X_t\right)$ denotes its marginal distributions. The motivation of TCA is to find a map Φ which could preserve the most data properties after projection, which means obtain the most variance, i.e.

$$P\left(\phi\left(\mathbf{x}_s\right)\right) \approx P\left(\phi\left(\mathbf{x}_t\right)\right)$$

or we can find conditional distribution of the two will also be similar as:

$$P\left(y_s \mid \phi\left(\mathbf{x}_s\right)\right)\right) \approx P\left(y_t \mid \phi\left(\mathbf{x}_t\right)\right)\right)$$

We ccan give the maximum mean discrepancy (MMD) formula as:

$$MMD(X, Y) = \left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \Phi\left(x_i\right) - \frac{1}{n_2} \sum_{j=1}^{n_2} \Phi\left(y_j\right) \right\|^2$$

where n_1, n_2 are the number of instances of the two domains. Then by changing the solution of this function to the solution of the kernel function, we can get:

$$\operatorname{Dist}\left(X_S^{\prime}, X_T^{\prime}\right) = \operatorname{tr}(KL)$$

where

$$K = \left[\begin{array}{ll} K_{S,S} & K_{S,T} \\ K_{T,S} & K_{T,T} \end{array} \right] \in \mathbb{R}^{\left(n_1+n_2\right) \times \left(n_1+n_2\right)}$$

, then, Q Yang [3] decomposed K to transfer this problem to:

$$\left\{ \begin{array}{l} \min tr\left(W^T KLKW\right) + \mu tr\left(W^T W\right) \\ \text{ s.t. } W^T KHKW = I_m \end{array} \right.$$

Finally, we can get the solution $W*$ as the m leading eigenvectors of

$$(KLK + \mu I)^{-1} KHK$$

5) Easy Transfer Learning (EasyTL): Most traditional and deep learning migration algorithms are parametric methods, which require a lot of time and money to train those hyperparameters. In order to overcome these drawbacks, Easy Transfer Learning (EasyTL) [4] learns non-parametric transfer features through intra-domain alignment, and learns transmission classification through intra-domain programming. EasyTL can also improve the performance of existing TL methods through in-domain programming as the final classifier, the procedure of EasyTL can be shown in Fig. I-A5.

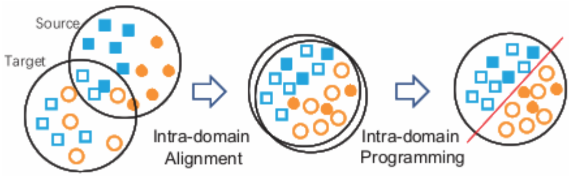


Fig. 2. procedure of EasyTL [4]

Instead of learning $\mathbf{y}^t$ directly, Easy Transfer Learning algorithm focuses on learning the probability annotation matrix $\mathbf{M}$. In this way, the cost function can be formalized as:

$$\mathcal{J} = \sum_j^{n_t} \sum_c^C D_{cj} M_{cj}$$

where the distance value D_{cj} is an entry in a distance matrix D. D_{cj} denotes the distance between $\mathbf{x}_j^t$ and the c -th class center of the source domain $\Omega_s^{(c)}$.

We can use constrains optimazition to get the softmax function, which give the label estimation value:

$$y_j^t = \arg\max_r \frac{M_{rj}}{\sum_c^C M_{cj}} \text{ for } r \in \{1, \cdots, C\}$$

It is noticeable that this classifier does not involve any parameters to tune explicitly. This is significantly different from well-established classifiers such as SVM that needs to tune numerous hyperparameters. In fact, intra-domain programming can be used alone for TL problems.

B. Deep Transfer Learning Methods

1) Deep Adaptation Network (DAN): Recent research shows that deep neural networks can learn transferable features, which can be well extended to new fields to adapt to tasks. Deep Adaptation Network (DAN) use deep net to optimize the loss function and distribution distance in Regenerative Nuclear Hilbert Space (RKHS) [5].

Denote $\mathcal{H}_k$ as the reproducing kernel Hilbert space (RKHS) endowed with a characteristic kernel k. The average embedding of the distribution p in $\mathcal{H}_k$ is a unique element $k(p)$, making $\mathbf{E}_{\mathbf{x}\sim p} f(\mathbf{x}) = \langle f(\mathbf{x}), \mu_k(p)\rangle_{\mathcal{H}_k}$ for all $f \in \mathcal{H}_k$. Define the MK-MMD $d_k(p, q)$between the probability distributions p and q as the average embedding distance RKHS of p and q, and define the square formula of MK-MMD as

$$d_k^2(p, q) \triangleq \left\| \mathbf{E}_p\left[\phi\left(\mathbf{x}^s\right)\right] - \mathbf{E}_q\left[\phi\left(\mathbf{x}^t\right)\right] \right\|_{\mathcal{H}_k}^2$$

and the kernel defined by the multiple cores is

$$\mathcal{K} \triangleq \left\{ k = \sum_{u=1}^{m} \beta_u k_u : \sum_{u=1}^{m} \beta_u = 1, \beta_u \geqslant 0, \forall u \right\}$$

Global optimization goal consists of two parts: loss function and distribution distance. The loss function is used to measure the difference between the predicted value and the true value.

DAN use adaptive method based on mk-mmd and CNNs to onercome that the target domain has no or only limited label information, so it is impossible to adapt CNN directly to the target domain through fine-tuning, or it is easy to overfit. Fig. I-B1 gives a description of the proposed DAN model.

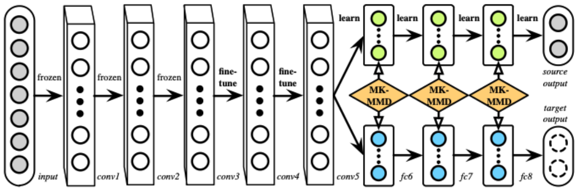


Fig. 3. The DAN architecture for learning transferable features.Since deep features eventually transition from general to specific along the network, (1) the features extracted by convolutional layers conv1–conv3 are general, hence these layers are frozen, (2) the features extracted by layers conv4–conv5 are slightly less transferable, hence these layers are learned via fine-tuning, and (3) fully connected layers fc6–fc8 are tailored to fit specific tasks, hence they are not transferable and should be adapted with MK-MMD. [5]

DAN fine-tuned the source of the labeled examples, requiring that under the hidden representation of the fully connected layers $f\ c6\ f\ c8$, the distribution of the source and target becomes similar. This can be achieved by adding a multi-layer adaptive regularizer (1) based on mk-mmd to the risk (3) of CNN:

$$\min_{\Theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J\left(\theta\left(\mathbf{x}_i^a\right), y_i^a\right) + \lambda \sum_{\ell=l_1}^{l_2} d_k^2\left(\mathcal{D}_s^\ell, \mathcal{D}_t^\ell\right)$$

where $\lambda > 0$ is a penalty parameter, l_1 and l_2 are layer indices between which the regularizer is effective.

II. Experiments

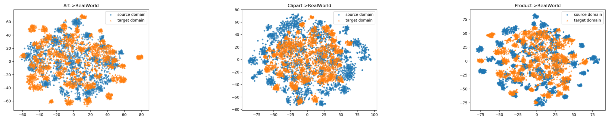
In this part, we will show the experimental results and comparative analysis of the results through two types of traditional transfer learning methods and deep transfer learning methods.

A. Baseline

Before using transfer learning methods, we applied linear SVM and rbf kernel SVM directly and used the results as baseline of this experiment. Besides, we used $GridSearchCV$ to find the optimal value of parameters. The results are shown in Table .I. And we visualize the source domain and target domain, the result is shown in the fig. 4.

TABLE I
THE RESULT OF BASELINE

result dataset	accuracy	C	kernel
Art->RealWorld	0.7484	10.0	rbf
Clipart->RealWorld	0.6577	0.01	linear
Product->RealWorld	0.7294	10.0	rbf



(a) A->RealWorld (b) C->RealWorld (c) P->RealWorld

Fig. 4. baseline

B. Traditional Transfer Learning Methods

1) KMM: In this experiment, we used the KMM method to assign different weights on source domain samples in order to make the probability distribution of the weighted source and target domains as close as possible. And our experiment results are shown in Table II.

From the results, we can see that rbf and linear KMM perform similarly, and there is almost no improvement compared to baseline. So we visualized the result of KMM for Art-Realworld in fig .II-B1.We speculate that the reason why the experimental performance has hardly improved is that the weights given by KMM to most samples are similar, which may not help to narrow the sample distribution of the source domain and target domain. In addition, the sample distribution of the source domain and target domain is not particularly different, and may be one of the reasons.

TABLE II
THE RESULT OF KMM

result dataset	accuracy	kernel
Art->RealWorld	**0.7528**	linear
Clipart->RealWorld	0.6518	linear
Product->RealWorld	0.7296	linear
Art->RealWorld	**0.7528**	rbf
Clipart->RealWorld	0.6513	rbf
Product->RealWorld	0.7294	rbf

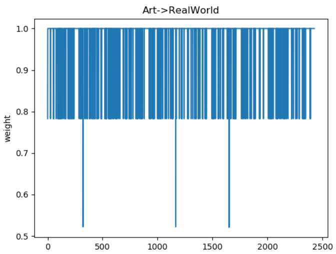


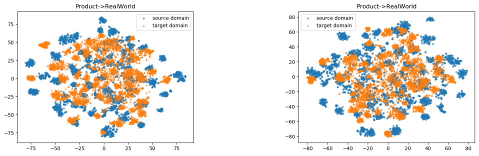
Fig. 5. Samlpe weights of Art-Realword of RBF

2) CORAL: In this experiment, we used the CORAL method to minimize the second-order distance between source domain and target domain. And our experiment results are shown in Table III.

From the results we can see that only the experiment results of Product-Realworld are slightly better than the baseline. We visualize the source domain processed by the CORAL method and compare it with the baseline. We can't find that the obvious data distribution is getting closer. We speculate that the reason CORAL is not performing well is that it is not suitable for no-deep models. In addition, we checked the dataset and found that there are few pictures in some categories, which may lead to insufficient training of the model and affect the accuracy of the test. Hoffman J and others proposed the deep CORAL method based on CORAL [6], which uses the CORAL loss function for neural network training, which greatly improves the experimental results. Unfortunately, due to limited computing resources, we did not use this method for experiment.

TABLE III
THE RESULT OF CORAL

result dataset	accuracy
Art->RealWorld	0.7381
Clipart->RealWorld	0.6513
Product->RealWorld	0.7369



(a) baseline

(b) CORAL

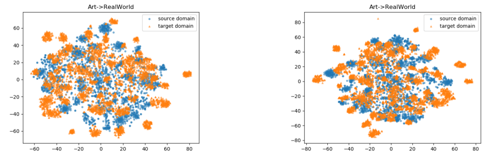
Fig. 6. baseline and CORAL for Product-Realworld

3) GFK: In this experiment we changed the dimension of geodesic flow kernel for comparative experiment. The experimental results are shown in the Table IV.

From the results we can see that the dimension of geodesic flow kernel will affect the experiment, and the optimal dimension in this project is 128, that is, the source domain and target domain can maintain maximum consistency in the 128-dimensional subspace. We speculate that when d is too small, the subspace cannot maintain the variance of the source domain to construct a good classifier. But when d is too large, the two subspaces start to appear in orthogonal directions, which will destroy the experimental results. For the three data sets, GFK's experimental results have improved compared to the baseline. We visualize the source domain and target domain after GFK processing, and compare with the baseline, as shown in the Fig .7. We can find that GFK makes the sample distribution of source domain and target domain indeed closer.

TABLE IV
THE RESULT OF GFK

result dataset	accuracy	dimension
Art->RealWorld	0.7190	32
Clipart->RealWorld	0.6336	
Product->RealWorld	0.7117	
Art->RealWorld	0.7456	64
Clipart->RealWorld	0.6561	
Product->RealWorld	**0.7335**	
Art->RealWorld	**0.7537**	128
Clipart->RealWorld	**0.6598**	
Product->RealWorld	0.7310	
Art->RealWorld	0.7482	256
Clipart->RealWorld	0.6566	
Product->RealWorld	0.7300	
Art->RealWorld	0.7475	512
Clipart->RealWorld	0.6575	
Product->RealWorld	0.7284	
Art->RealWorld	0.7475	1024
Clipart->RealWorld	0.6580	
Product->RealWorld	0.7296	



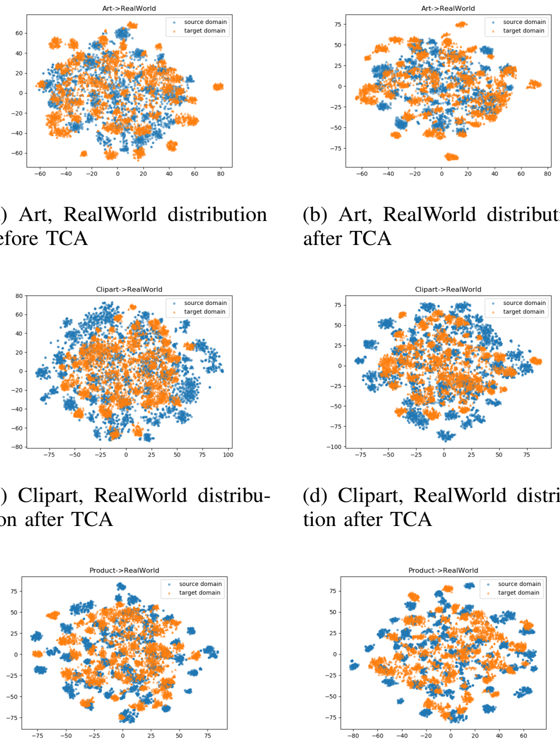
(a) baseline

(b) GFK(128)

Fig. 7. baseline and GFK for Art-Realworld

TABLE V
ACCURACY OF TCA-BASED METHOD

Task	Art->RealWorld			Clipart->RealWorld			Product->RealWorld		
Kn	primal	linear	rbf	primal	linear	rbf	primal	linear	rbf
32	71.59	69.82	70.74	64.45	62.61	63.44	71.29	69.34	70.69
64	75.28	72.89	73.74	65.92	63.71	65.32	73.35	71.33	72.78
128	75.49	73.28	74.27	65.32	63.78	65.27	73.67	71.56	73.24
256	75.74	73.40	74.18	66.12	63.85	65.23	73.63	71.70	73.24
512	75.58	73.40	74.27	66.12	63.81	65.25	73.56	71.70	73.12
1024	75.53	73.40	74.23	66.10	63.81	65.25	73.63	71.70	73.16
2048	75.60	73.40	74.27	66.08	63.81	65.25	73.65	71.70	73.14



(a) Art, RealWorld distribution before TCA

(b) Art, RealWorld distribution after TCA

(c) Clipart, RealWorld distribution after TCA

(d) Clipart, RealWorld distribution after TCA

(e) Product, RealWorld distribution after TCA

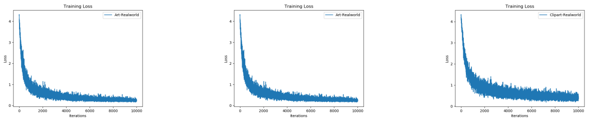
(f) Product, RealWorld distribution after TCA

Fig. 8. TCA and baselines

4) Transfer Component Analysis (TCA): In this experiment, we test TCA method on kernels amount $[primal, linear, rbf]$, and various aim dimension in $[32, 64, 128, 256, 512, 1024, 2048]$, and different domain transfer tasks in [Art->RealWorld, Clipart->RealWorld, Product->RealWorld]. We can show our experimental results in Tab. V . To figure out why TCA can have better performance, we plot the source domain data and target domain data in the same figure. Here, we can see the data distribution of case A-R, C-R and P-R in Fig. 8.

Unlike the sample-based transfer learning method like KMM, TCA is a feature-based transfer learning method. As we can see from Fig. 8, after using TCA, the area where the source domain data distribution coincides with the target domain data distribution is larger. And it's clear that TCA has different degrees of improvement for the accuracy of the three cases (A-R, C-R, P-R) and TCA performance is better than KMM performance.

5) Easy Transfer Learning (EasyTL): In the previous experiments, we find that the model performs good need extensive parameter tuing and a large mount of time to fix or train. With regard to Easy Transfer Learning (EasyTL), it's proved to maintain good performance while avoiding extensive parameter tuning. According to previous related work, there are four kinds of methods for Intra-domain alignment, they are RAW(do not use Intra-domain alignment), PCA, CORAL, GFK. We test Raw and Coral in this experiment.



(a) A-R loss via DAN

(b) P-R loss via DAN

(c) C-R loss via DAN

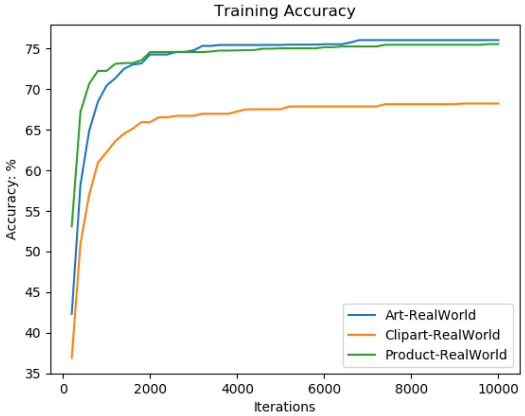
Fig. 9. Loss tendency in DAN

In this experiment, we test EasyTL method on intra_align amount $[raw, coral]$, and various dist metrics in $[euclidean, ma, cosine, rbf]$, and different domain transfer tasks in [Art->RealWorld, Clipart->RealWorld, Product->RealWorld]. We can show our experimental results in Tab. VI.

TABLE VI
THE RESULT OF EASYTL

result dataset	accuracy	dist	intra_align
Art->RealWorld	0.7448	euclidean	raw
Art->RealWorld	0.7565	euclidean	coral
Art->RealWorld	0.0592	ma	raw
Art->RealWorld	0.0566	ma	coral
Art->RealWorld	0.7372	cosine	raw
Art->RealWorld	0.7539	cosine	coral
Art->RealWorld	0.0153	rbf	raw
Art->RealWorld	0.0153	rbf	coral
Clipart->RealWorld	0.6520	euclidean	raw
Clipart->RealWorld	0.6768	euclidean	coral
Clipart->RealWorld	0.0433	ma	raw
Clipart->RealWorld	0.0527	ma	coral
Clipart->RealWorld	0.6628	cosine	raw
Clipart->RealWorld	0.6823	cosine	coral
Clipart->RealWorld	0.0146	rbf	raw
Clipart->RealWorld	0.0146	rbf	coral
Product->RealWorld	0.7378	euclidean	raw
Product->RealWorld	0.7514	euclidean	coral
Product->RealWorld	0.0658	ma	raw
Product->RealWorld	0.0530	ma	coral
Product->RealWorld	0.7376	cosine	raw
Product->RealWorld	0.7502	cosine	coral
Product->RealWorld	0.0174	rbf	raw
Product->RealWorld	0.0174	rbf	coral

We can find with ma and rbf distance metrics the EasyTL performs poor, and we deem the reason that EasyTL can't performs well with complicated model like rbf. And we can find with coral intra_align, EasyTL performs better than raw one. Compared with othor methods in our work, we can find EasyTL have relatively better performance while cost less time, we deem the reason that EasyTL obtains the softmax probability (floating point weight) through linear programming. In this way, EasyTL not only considers the relationship between the sample and the center, but also considers the relationship with other samples.



(a) Accuracy comprison via DAN

Fig. 10. Accuracy tendency in DAN

C. Deep Transfer Learning Methods

1) Deep Adaptation Network (DAN): In this experiment, we test DAN method via different domain transfer tasks in [Art->RealWorld, Clipart->RealWorld, Product->RealWorld]. We can show our experimental results in Tab. VII . To figure out The loss and accuracy of the training process, we can see them in Fig. 9 and Fig. 10.

TABLE VII
THE RESULT OF KMM

result dataset	accuracy(%)
Art->RealWorld	75.63
Clipart->RealWorld	67.68
Product->RealWorld	75.14

We know that with regard to DAN, global optimization goal consists of two parts: loss function and distribution distance. The loss function is used to measure the difference between the predicted value and the true value. Therefore every iteration makes the distribution between source domain and target domain closer. We can see it in the iteration acuracy tendency, but it's time-consuming and for some case, it performs poor due to the interval between two aim domain.

III. Conclusion

In this project, we try totally 6 methods about transfer learning. We can find deep network method DAN achieves best performance and hardly affected by the initial parameters settings. TCA is fast and gives a not bad result. compared with above, EasyTL is a good choice because it can achieve a very good performance without extensive parameters tuning and a lot of training time.

References

[1] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," *Dataset shift in machine learning*, vol. 3, no. 4, p. 5, 2009.

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

[3] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[4] J. Wang, Y. Chen, H. Yu, M. Huang, and Q. Yang, "Easy transfer learning by exploiting intra-domain structures," 2019.

[5] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," 2015.

[6] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.