Project 4 of CS245: Domain Adaptation for Image Classification

515030910369, Sicheng Zuo, Ginga479726995@sjtu.edu.cn
516030910044, Yujie Yang, yangyujie@sjtu.edu.cn
516030910021, Hongxiang Yu, sjtu.yuhongxiang@sjtu.edu.cn
5130309210, Zheng Gong, 573965625@qq.com

May 28, 2019

1 Introduction

In this project, we try to use different unsupervised domain adaption methods to improve the accuracy of image classification. The domain adaption dataset we use is Office-Home dataset which consists of 65 categories of office depot from four domains (i.e., A: Art, C: Clipart, P: Product, R: Real-world). There are two parts in this project. Firstly, we tried several traditional transfer learning methods, including KMM, TCA, CORAL, GFK and EasyTL. Secondly, we use some deep transfer learning methods to compare with the traditional transfer learning methods, including DAN and CDAN. Our results shows that the deep transfer learning methods works much better than traditional methods. But tranditional method EasyTL also demonstrates a good performance comparable to the performance of deep transfer learning methods.

2 Baseline

Before using transfer learning methods, we first apply linear SVM and 1-NN (KNN, *n_neighbors=1*) directly, and use the results as baseline. For linear SVM model, *GridSearchCV* is used to find the optimal value of parameter **C**. According to the result of project-2, we know that using *cosine* distance metric can achieve the best performance, so the *cosine* distance metric was used in this project. The result of baseline is shown in table-1.

Table 1: Results of Baseline

Model	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time
SVM	74.41%	34.62s	65.62%	58.37s	72.53%	47.11s
1NN	67.78%	6.15s	60.89%	13.97s	69.89%	14.59s

[1] A: Art, C: Clipart, P: Product, R: Real-world

[2] 1NN: K nearest neighbor when $K = 1$

We find that although the performance of 1NN is not so good as SVM, the speed of 1NN is much faster than SVM. In the following tests, we will use the result in table-1 as the baseline, and compare the results of different transfer learning method with the baseline.

3 Traditional transfer learning methods

In this part, we try Kernel Mean Matching (KMM), Transfer Component Analysis (TCA), CORrelation ALignment (CORAL), Geodesic Flow Kernel (GFK) and Easy Transfer Learning (EasyTL). We will introduce these methods one by one, and the results we have achieved. From the results of baseline, we find that the performance of SVM model is much better than 1NN. Since we mainly focus on the accuracy, we only use SVM model during the process of classification. The SVM model's parameter will be fine tuned by using *GridSearchCV*.

3.1 KMM

Kernel Mean Matching assigns different weights on source domain samples, the goal is to make the probability distribution of the weighted source and target domains as close as possible. In the algorithm of KMM, there are two kinds of kernel to choose (linear kernel and rbf kernel). Figure-1 shows the results of our experiments.

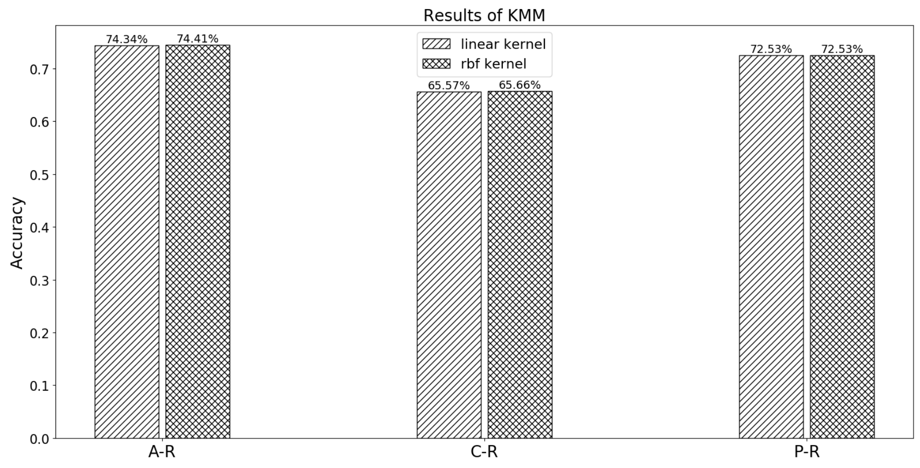

Figure 1: KMM results

Apparently, the performance of rbf kernel is slightly better than the linear kernel, but we are disappointed to find that KMM does not have much improvement on the accuracy compared with the baseline. To find the reason of why KMM is not effective, we visualize the sample weights of the case Art-Realword in figure-2.

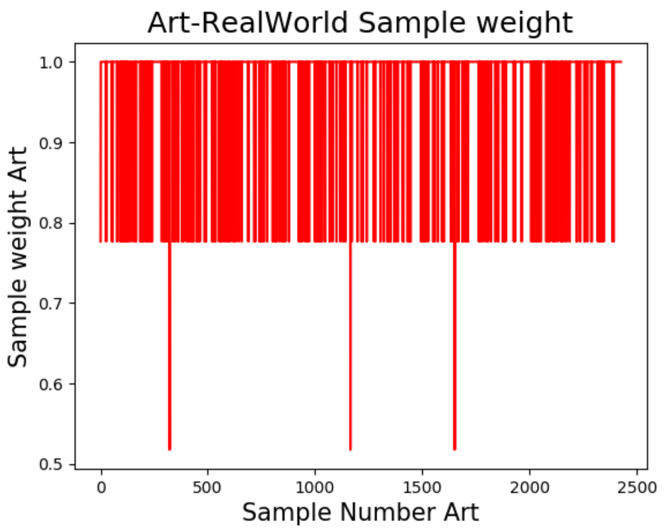

Figure 2: Sample weights of case Art-Realword

KMM does assign different weights to the sample, but most samples have a weight of either 1 or 0.78. In order to have a better view of the data distribution of source domain and target domain, we use T-SNE to visualize the data. In order to reduce time consuming, we use LDA for dimension reduction before applying T-SNE. The visualization of data distribution is displayed in figure-3.

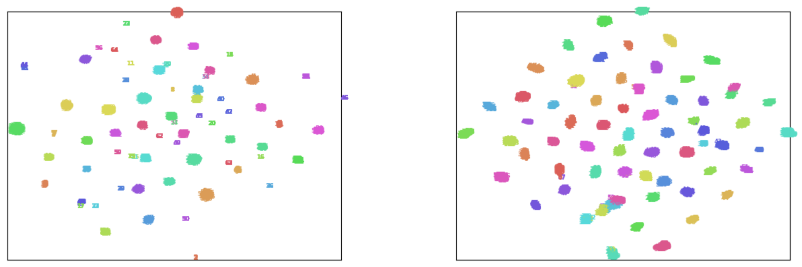

Figure 3: Data distribution of source domain (Art) and target domain (Real-world)

We can see that, samples belonging to the same class are grouped together, this because we use the LDA. However, the data distribution of source domain and target domain are completely different. I think there are three reasons to explain why KMM can not improve the classification performance.

1. KMM makes the kernel mean of source domain smaples and target domain smaples as close as possible, but this does not mean that the data distribution will be close.
2. Although we can assign different weights to samples, since there are only two main weight values here, this may not be helpful in getting the data distribution as close as possible.
3. The data in target domain is closer to each other compared with data in source domain, and there are more overlapping parts, which is disadvantageous for improving the classification performance.

Finally, the results and time consuming information is shown in table-2.

Table 2: Results of KMM

Model	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time
KMM	74.41%	33.98s	65.66%	71.64s	72.53%	58.07s

3.2 TCA

Transfer component Analysis focuses on minimizing the margin probability distribution distance between the source domain and the target domain by projecting the source domain data and target domain data to a common subspace. In the algorithm of TCA, we can choose the kernel's type, the feature dimension after projection and λ. What's more, in order to get a good result, the parameters of SVM should also be tuned. Since there are too many parameters to be adjusted, the parameter tuning process can be divided into following steps:

1. Firstly, fix the data dimension after projection to 2048 and try three different kernel types. *GridSearchCV* is used for SVM model.
2. Secondly, use the optimal type of kernel that we find in step 1 and tune the data dimension after projection.
3. Thirdly, adjuest parameter λ to get a better result.

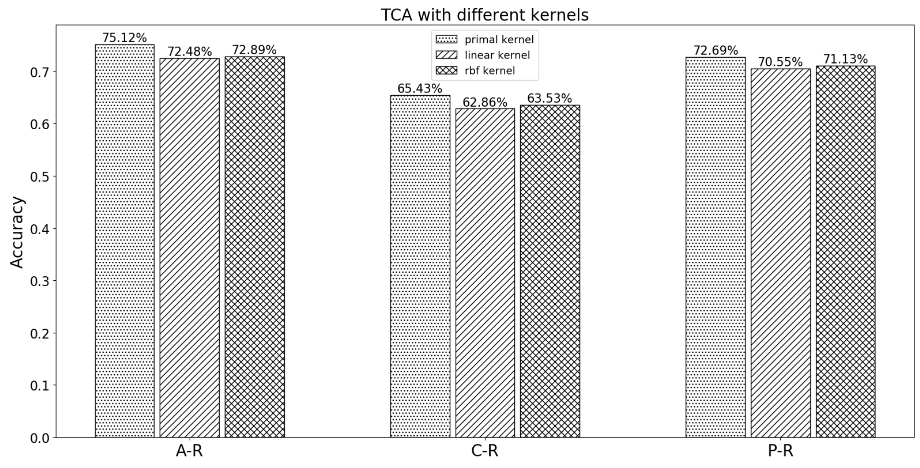

Figure 4: Different types of kernel in TCA

From figure-4 we can see that the primal kernel outperforms the other two kernels. Therefore, the primal kernel will be used in the following tests. After we get the optimal kernel type, data dimension after projection is adjuested in order to get a better performance. The original data dimension is 2048, so we try the data dimension with 32, 64, 128, 256, 512, 1024, 2048. The performance of different data dimension is shown in figure-5.

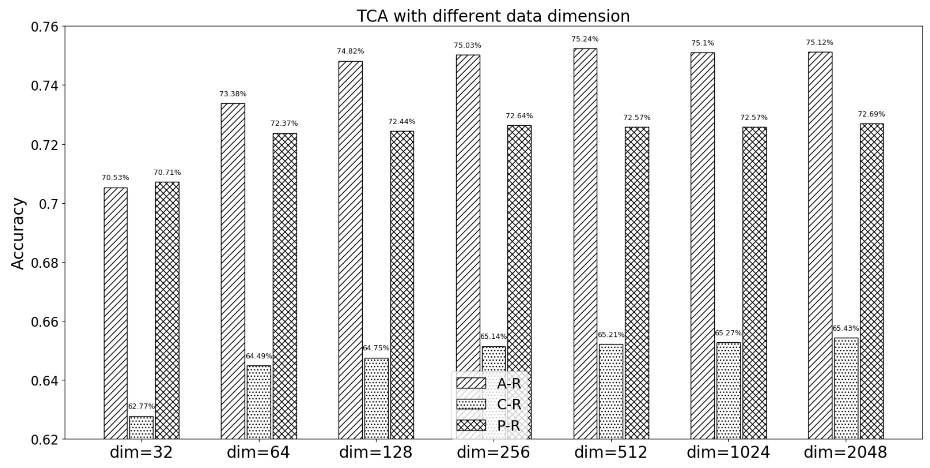

Figure 5: Different data dimension in TCA

As figure-5 shows, increasing the data dimension after projection can improve the performance. Larger data dimension means more useful information. The optimal data dimension can be selected according to the results in figure-5. Finally, we carefully tune the parameter λ to get a better result. Our result of TCA is shown in table-3.

Table 3: Results of TCA

Model	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time
TCA	75.24%	158.52s	65.66%	291.23s	72.83%	191.77s

TCA has different degrees of improvement for the accuracy of the three cases (A-R, C-R, P-R). Obviously, TCA performance is better than KMM performance. 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 in figure-6.

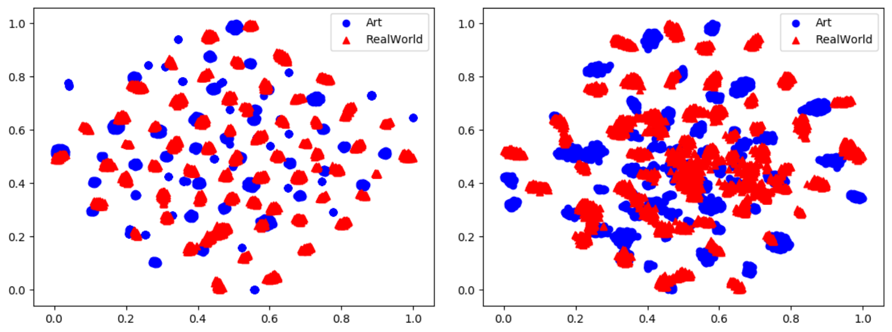

Figure 6: Left: source and target domain data distribution before applying TCA. Right: source and target domain data distribution after applying TCA. (A-R case)

Unlike the sample-based transfer learning method like KMM, TCA is a feature-based transfer learning method. As we can see from figure-6, after using TCA, the area where the source domain data distribution coincides with the target domain data distribution is larger. For comparison, let's look at the data distribution of the C-R case in figure-7.

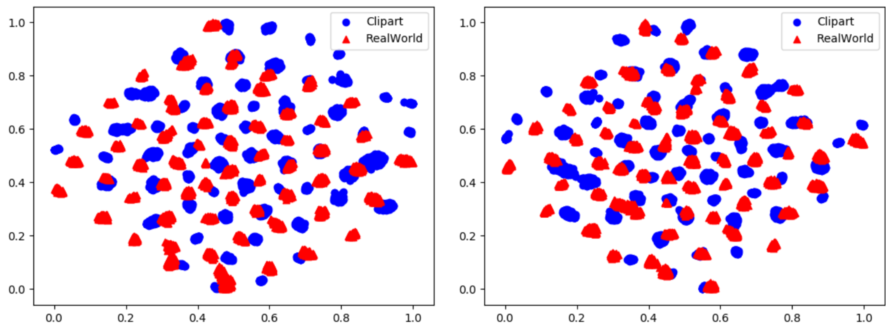

Figure 7: Left: source and target domain data distribution before applying TCA. Right: source and target domain data distribution after applying TCA. (C-R case)

Obviously, compared with A-R case, the area where the data distribution of the source domain and the target domain coincide after applying TCA is not so large. What's more, compared with basline, the accuracy of A-R case increase by 0.83% after using TCA, while C-R case only increase by 0.04%. Therefore, we can reach the conclusion that the larger the area of the coincident part, the more similar the data distribution of the source and target domains is, and we can get higher accuracy.

3.3 CORAL

CORrelation ALignment perform second-order feature alignment on the source and target domain. Suppose C_s and C_t are covariance matrix of the source domain and target domain. The CORAL method learns a second-order eigen transform $\mathbf{A}$, which minimizes the feature distance between C_s and C_t. The CORAL method is simple and efficient, and does not require any adjustment of parameters. Therefore, we just simply show the results of CORAL in table-4.

Table 4: Results of CORAL

Model	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time
CORAL	73.35%	88.92s	64.98%	196.09s	72.21%	185.05s

We are disappointed to find that the performance becomes even worse after applying CORAL. To find the reason, we first visualize the data in figure-8.

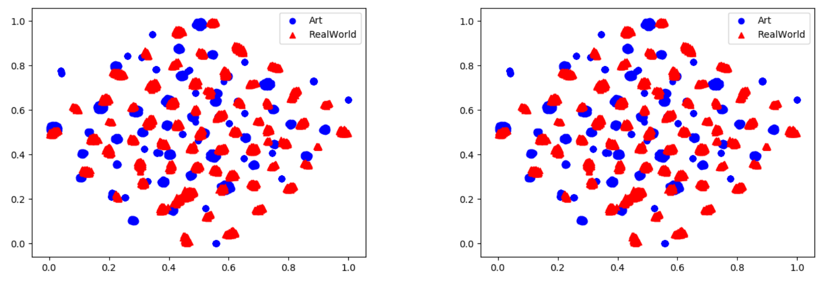

Figure 8: Left: source and target domain data distribution before applying CORAL. Right: source and target domain data distribution after applying CORAL. (A-R case)

We are surprised to find that the data distribution before and after applying CORAL has little difference. For further analysis, we do some extra experiments. Here are the main steps:

- Train SVM with the source domain data and labels, use 5-fold cross-validation get the best model 1 and the best score 1. Apply the best model 1 to the target domain data, get transfer accuracy 1.

- Use the data and labels of target domain to train SVM with 5-fold cross-validation to get the max score 1.
- Apply the CORAL model and get the newly generated source domain data.
- Use the newly generated source domain data and labels to train SVM, then get the best model 2 and the best score 2. Apply the best model 2 to the target domain data, get transfer accuracy 2.

The results of extra experiments are shown in table-5.

Table 5: Results of CORAL extra experiments

results / cases	best score 1	transfer acc 1	max score 1	best score 2	transfer acc 2
A-R	79.56%	74.41%	85.08%	79.36%	73.36%
C-R	88.48%	65.62%	82.07%	87.97%	64.98%
P-R	95.88%	72.53%	84.21%	95.81%	72.21%

[1] The definition of best score 1, best score 2, transfer acc 1, transfer acc 2 and max score 1 can be found in the experiment main steps above.

By comparing best score 1 and best score 2, we can see that applying CORAL to source domain data will make the accuracy drop slightly. What's more, CORAL does not reduce the accuracy gap between the source domain and target domain. We also find that case C-R's performance gap and case P-R's performance gap are really big. To find the reason for CORAL's poor performance and big performance gap, we combine the results in table-5 and visualization of data distribution in figure-8. And there are some reasons:

* The second-order features of source domain and target domain are sufficiently similar, using the CORAL model to further align second-order features will do harm to the accuracy.

* The features of the image extracted by ResNet50 are not the intrinsic features of the object. For example, the marker (the class with the lowest classification accuracy).

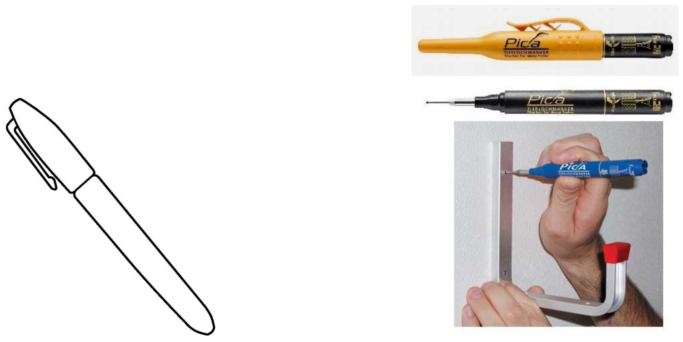
Figure 9: Left: marker of Clipart domain. Right: marker of Realworld domain

The marker images in Clipart domain are mostly composed of simple lines. But the the marker images in Realworld domain are much more complex and the objects involved are also more complicated. Hence, the feature in two domains are totally different. The figures in Clipart domain with succinct style is closer to the abstract description of the object, and their features is closer to the intrinsic features.

* There are unrelated objects in the picture, which will interfere with the extraction of features. For example in figure-10, the main content of the figure is not the marker, but the face portrait.

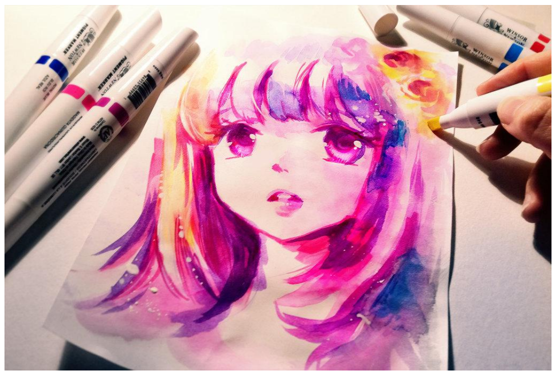
Figure 10: Figure with irrelevant objects

* In some categories, there are few samples. For example, in Art domain, there are only 20 figures in the marker categories. This may lead to insufficient training of the model.

3.4 GFK

Geodesic Flow Kernel model is a kernel-based method that focuses on exploiting low-dimensional structures that are intrinsic to many vision datasets. It domain shift by integrating an integrating an infinite number of subspaces that characterize changes in geometric and statistical properties from the source to the target domain. GFK approach is computationally advantageous, automatically inferring important algorithmic parameters without requiring extensive cross-validation or labeled data from either domain. The only parameter we have to tune is the feature dimension.

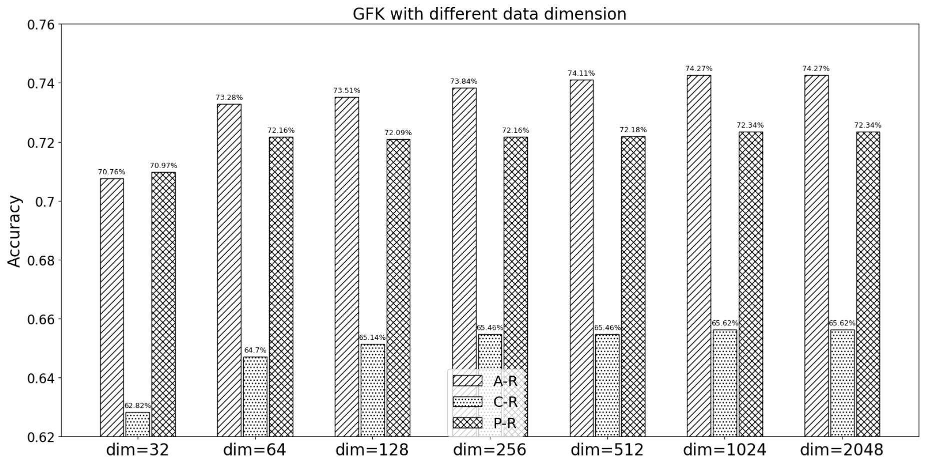

Figure 11: Different data dimension in GFK

The performance of different feature dimension is displayed in figure-11. And the optimal accuracy of GFK methods is shown in table-6. For the three cases (A-R, C-R, P-R), the accuracy is the same for 1024-dimension and the 2048-dimension, but the small dimension can reduce the time overhead, so the best dimension of the three cases is 1024.

Table 6: Results of GFK

Model	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time
GFK	74.27%	136.37s	65.62%	184.93s	72.34%	142.96s

In figure-12, although data distribution has been changed after using GFK model, but GFK contributes no improvement to the accuracy.

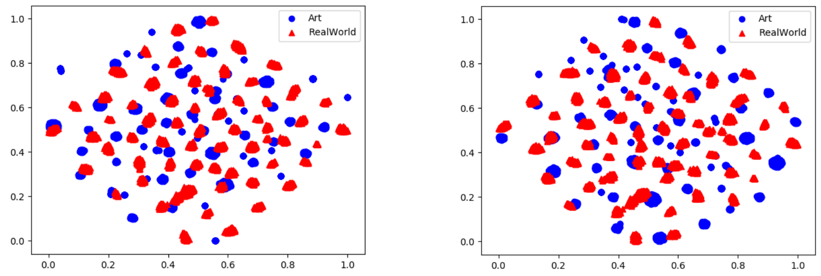

Figure 12: Left: source and target domain data distribution before applying GFK. Right: source and target domain data distribution after applying GFK. (A-R case)

As what we do in CORAL part, we also do some extra experiments here. The main steps are almost the same. The only difference is that we only get newly generated source domain data with CORAL method, but we can get both newly generated source domain data and target domain data with GFK method. Therefore, a new item called *max score 2* will be added to the results table. The *max score 2* will be generated by using newly generated target domain data to train SVM with 5-fold cross-validation. The results is shown in table-7.

Table 7: Results of GFK extra experiments

results	best	transfer	max	best	transfer	max
cases	score 1	acc 1	score 1	score 2	acc 2	score 2
A-R	79.56%	74.41%	85.08%	79.56%	74.27%	85.15%
C-R	88.48%	65.62%	82.07%	88.41%	65.62%	82.03%
P-R	95.88%	72.53%	84.21%	95.92%	72.34%	84.12%

By comparing the best score 1 and the best score 2, max score 1 and max score 2, we find that applying GFK to source and target domain data will not hurt the accuracy like applying CORAL. But GFK still lead to negative transfer. In this regard, our existing knowledge is not enough to explain, deeper understanding of GFK model's algorithm is needed. Maybe GFK is not suitable for this problem or the parameters of GFK are not fine tuned.

3.5 EasyTL

In the above, we disussed four tranditional transfer learning methods(KMM, TCA, CORAL, GFK). We find that the model performs good need extensive parameter tuing, but the model which does not need extensive adjustment of parameters performs bad. Hence, someone proposed the Easy Transfer leanring model to maintain good performance while avoiding extensive parameter tuning. EasyTL focuses on exploiting the intra-domain structure. The main part of EasyTL is a novel non-parametric Intra-domain programming classifier, while remains open for adopting existing methods for Intra-domain alignment. In the code written by the author of EasyTL, there are four kinds of methods for Intra-domain alignment, they are RAW(do not use Intra-domain alignment), PCA, CORAL, GFK. We tried all four kinds of methods, here are the results:

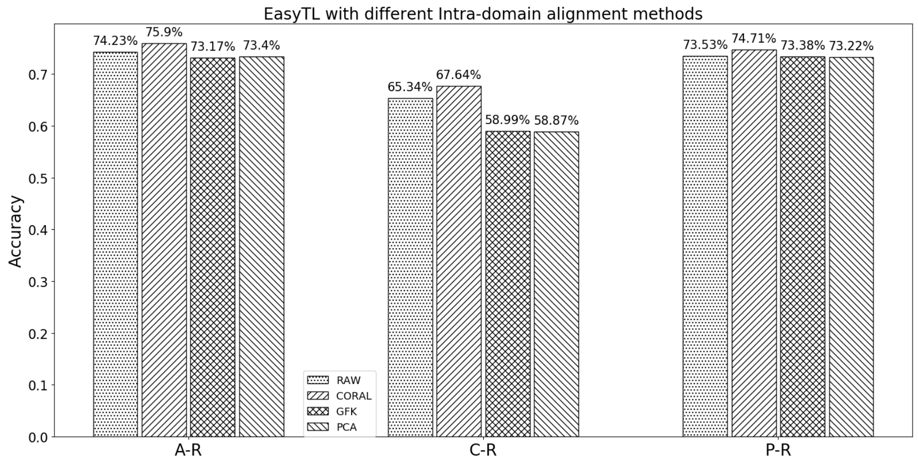

Figure 13: Different Intra-domain alignment methods of EasyTL

The performance of CORAL Intra-domain alignment methods is significantly better than other methods. We also notice that EasyTL provides different distance metrics. According the result of project-2, we know that the cosine distance metric achieves the best result in most situations. Hence, we try to use cosine distance metric to further improve the accuracy. Here are our results in table-8:

Table 8: Results of EasyTL

metrics \ results	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time
euclidean	**75.90%**	8079.14s	67.64%	8232.09s	74.71%	7885.10s
cosine	75.76%	7935.23s	**68.17%**	8079.55s	**75.44%**	7959.01s

The same as what we expected, the cosine distance has a good effect on improving accuracy. By comparing with the basline, we find that the accuracy of P-R case increased the most. So, we visualize the data distribution of P-R case in figure-14.

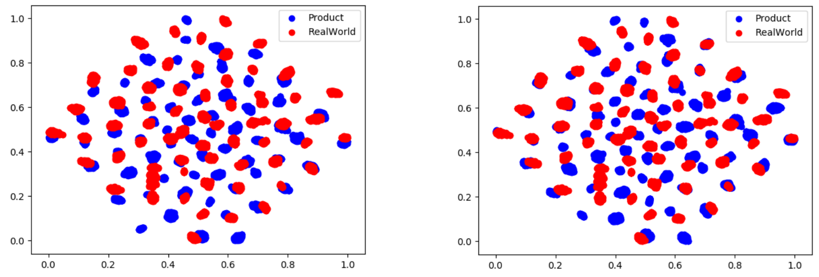

Figure 14: Left: source and target domain data distribution before applying EasyTL. Right: source and target domain data distribution after applying EasyTL. (P-R case)

Obviously, after using EasyTL, the source domain data distribution and the target data distribution are more coincident (more overlapping parts). This also demonstrates the effectiveness of EasyTL. The final result of EasyTL method is shown in tabel-9.

Table 9: Results of EasyTL

Model	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time
EasyTL	75.90%	8079.14s	68.17%	8079.55s	75.44%	7959.01s

4 deep transfer learning methods

In this section, we tried two deep learning methods, they are Deep Adaptation Network (DAN) and Conditional Adversarial Domain Adaptation (CDAN). Due to the limitations of time and knowledge level, we do not delve into these two methods here.

4.1 DAN

DAN is a kind of deep network adaption method. Deep network adaption achieves the adaption of source and target domain data by adding an adaption layer. The loss function of the whole network is also composed of two parts: the classification error ℓ_C on the source domain data with label, and the discriminant error ℓ_D on the data of the two fields. By this way, we can make the data distribution of the source domain and the target domain closer, which makes the network's performance better. The DAN we used is implemented by pytorch. The DAN is trained on one Tesla K80 GPU core. Here is our configuration during the training process:

Table 10: DAN configuration

batch size	epochs	learning rate
32	50	0.01

The loss and accuracy of the training process are shown in figure-15. The smaller the loss, the higher the accuracy. The loss and accuracy is satble after 10 epochs.

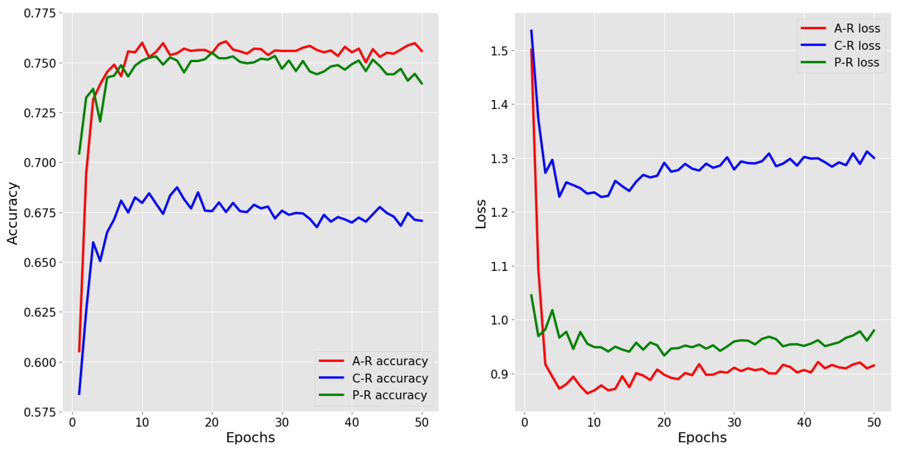

Figure 15: Loss and accuracy during the training of DAN

The final results of DAN model is shown in table-11.

Table 11: Results of DAN

Model	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time
DAN	76.06%	>10000s	68.74%	>10000s	75.49%	>10000s

4.2 CDAN

CDAN is a method that make use of deep adversarial network to do transfer learning. The generator continuously learn the domain data features until the discriminator can not distinguish the data from the two domains. similar to the deep network adaption method, the loss of deep adversarial network is also composed of two parts: loss of network training ℓ_C and domain discriminant loss ℓ_D. The CDAN is also implemented by pytorch. We trained CDAN on one Tesla K80 GPU core. Here is our configuration during the training process:

Table 12: CDAN configuration

batch size	epochs	learning rate
36	100000	0.001

The accuracy during training is shown in figure-16. Obviously, CDAN has achieved the best results so far. The results of CDAN is shown in table-13.

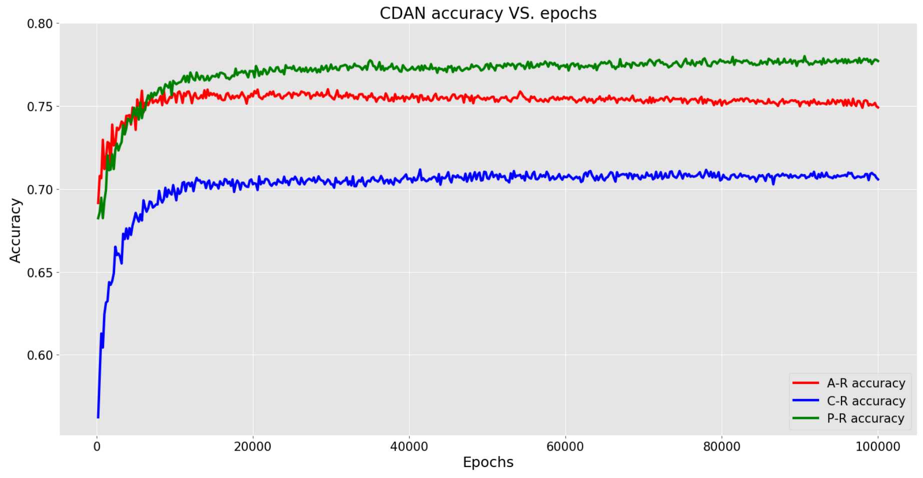

Figure 16: Accuracy during the training of CDAN

Table 13: Results of CDAN

Model	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time
CDAN	76.02%	>10000s	71.17%	>10000s	78.01%	>10000s

5 Summary

In this project, we try totally 7 methods about transfer learning. The result shows that deep transfer methods works significantly better than traditional transfer methods. Here we list the results of all the methods we tried in table-14.

Table 14: Results of all the methods

Model	A-R acc	A-R time	C-R acc	C-R time	P-R acc	P-R time	Avg time	Avg acc	rank
Baseline SVM	74.41%	34.62s	65.62%	58.37s	72.53%	47.11s	46.7s	70.85%	6
KMM	74.41%	33.98s	65.66%	71.64s	72.53%	58.07s	54.56s	70.87%	5
TCA	75.24%	158.52s	65.66%	291.23s	72.83%	191.77s	213.84s	71.24%	4
CORAL	73.55%	88.92s	64.98%	196.09s	72.21%	185.05s	156.69s	70.18%	8
GFK	74.27%	136.37s	65.62%	184.93s	72.34%	142.96s	154.75s	70.74%	7
EasyTL	75.90%	8079.14s	68.17%	8079.55s	75.44%	7959.01s	8039.23s	73.17%	3
DAN	**76.06%**	>10000s	68.74%	>10000s	75.49%	>10000s	>10000s	73.20%	2
CDAN	76.02%	>10000s	**71.17%**	>10000s	**78.01%**	>10000s	>10000s	75.07%	1

From the perspective of accuracy, the CDAN performs much better than other methods. From the perspective of speed, TCA is fast and gives a not bad result. But if we take both the speed and accuracy into account, we think the EasyTL is the best choice. Although transfer methods with deep adversarial network can achieve best results, it takes a lot of time to train the model and tune the parameters. But EasyTL can achieve a very good performance without extensive parameters tuning and a lot of time to train the model. This is very important in practical application situations.