Understanding “Deep transfer learning with Joint Adaptation Networks”

Created by: Eric Fan

Created on: 2017/11/13

yfanal@connect.ust.hk

OUTLINE

Basics of Transfer Learning

Definition:

- Definition 1: (Transfer Learning) Given a source domain D_T and learning task T_T, a target domain D_T and learning task T_T , transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$.
- Domain: In the above definition, a domain is a pair $D = \{X, P(X)\}$. Thus the condition $D_S \neq D_T$ implies that either $X_S \neq X_T$ or $P_S(X) \neq P_T(X)$.
- Task: Similarly, a task is defined as a pair $T = \{Y, P(Y|X)\}$. Thus the condition $T_S \neq T_T$ implies that either $Y_S \neq Y_T$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$.

Basics of Transfer Learning

Learning Target:

Domain Adaptation: Minimize the distance between Source Domain and Target domain.

Basics of Transfer Learning

How to define the distance?

- Manhattan distance
- Euclidean distance
- Cosine Distance
- ...
- MMD(maximum mean discrepancy)

Basics of Transfer Learning

MMD(Maximum Mean Discrepancy)

MMD is introduced for comparing distributions based on the corresponding RKHS distance.

Definition:

Given smaples $X = x_i$ and $Y = y_i$ from two distributions and a kernel-induced feature map: ϕ. The empirical estimate of MMD between X and Y could be defined as:

$$\mathrm{MMD}(X,Y) = ||\frac{1}{n_1}\sum_{i=1}^{n_1}\phi(x_i) - \frac{1}{n_2}\sum_{i=1}^{n_2}\phi(y_i)||_H^2$$

Basics of Transfer Learning

MMD on Transfer Learning:

$$\mathrm{Dist}(X'_S, X'_T) = ||\frac{1}{n_1}\sum_{i=1}^{n_1}\phi(x_{S_i}) - \frac{1}{n_2}\sum_{i=1}^{n_2}\phi(x_{T_i})||_H^2$$

Learning Target:

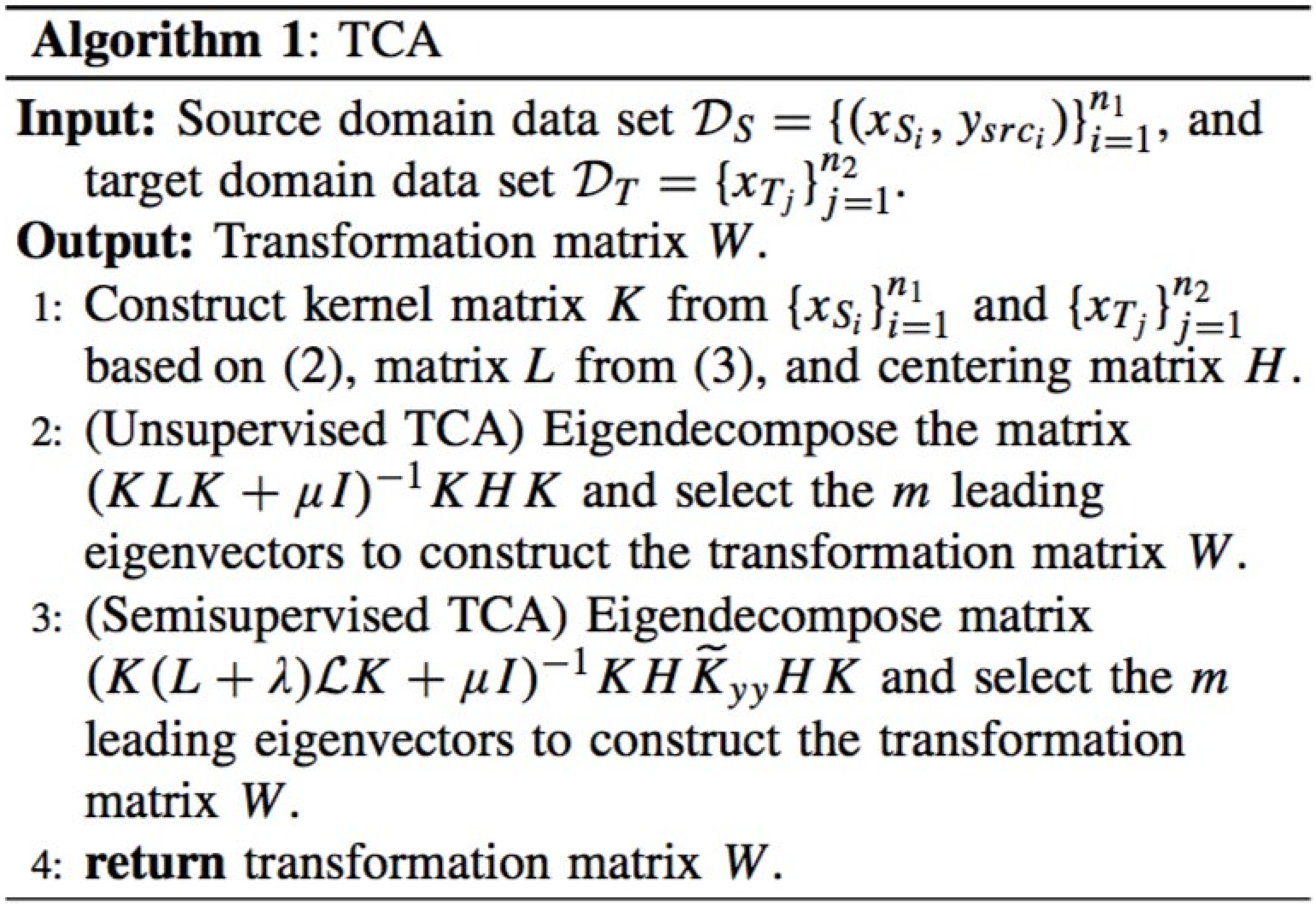
The problem is then converted into a pure mathematical problem:Find a nonlinear mapping ϕ to minimize this quantity.

How to solve?

Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. Domain adaptation via transfer component analysis IEEE Transactions on Neural Networks (TNN), 22(2):199–210, 2011.

TCA(Transfer Component Analysis)

Algorithm:

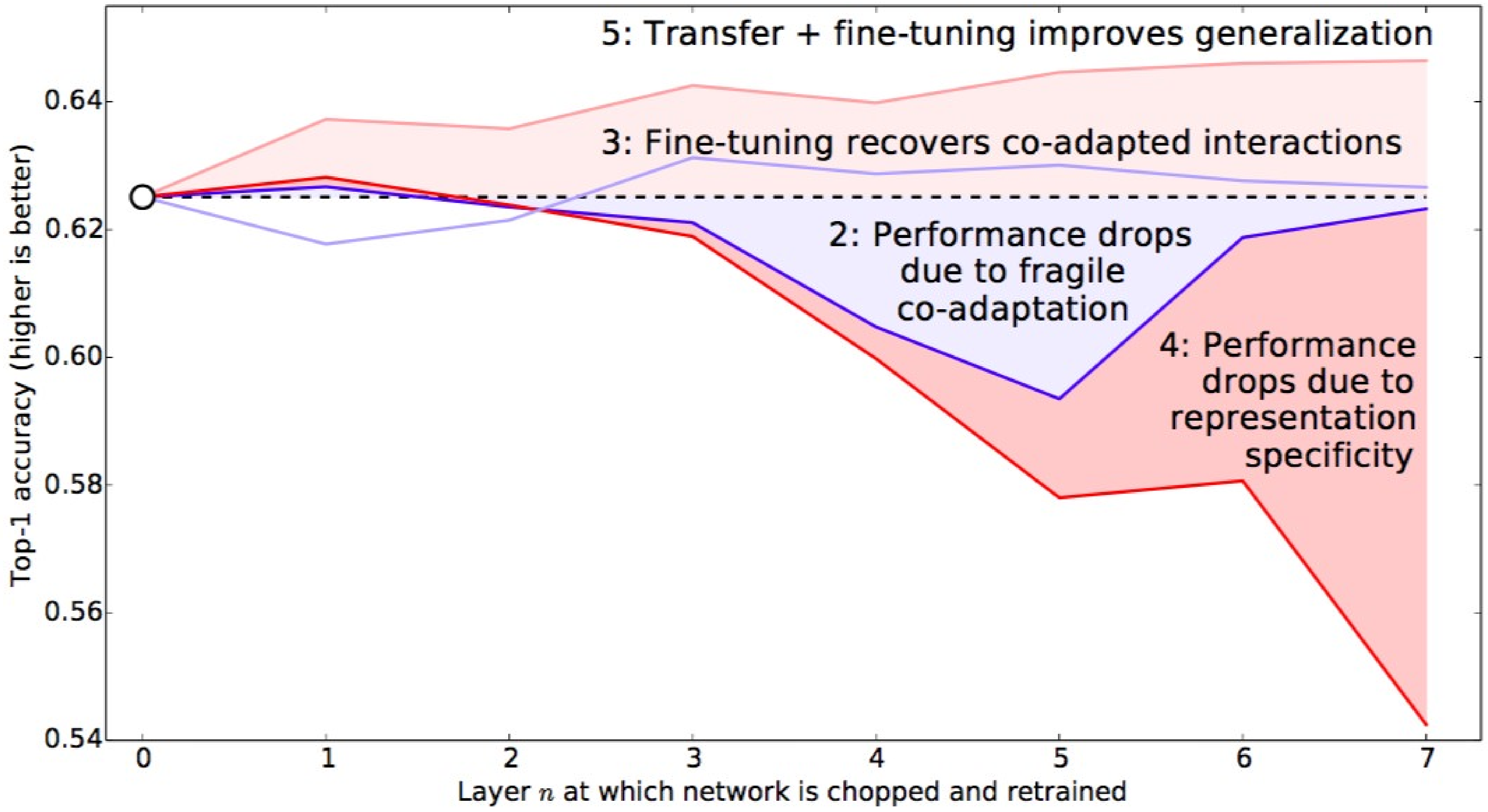


```
Algorithm 1: TCA
Input: Source domain data set $\mathcal{D}_S = \{(x_{S_i}, y_{src_i})\}_{i=1}^{n_1}$, and
    target domain data set $\mathcal{D}_T = \{x_{T_j}\}_{j=1}^{n_2}$.
Output: Transformation matrix $W$.
1: Construct kernel matrix $K$ from $\{x_{S_i}\}_{i=1}^{n_1}$ and $\{x_{T_j}\}_{j=1}^{n_2}$
   based on (2), matrix $L$ from (3), and centering matrix $H$.
2: (Unsupervised TCA) Eigendecompose the matrix
   $(KLK + \mu I)^{-1} KHK$ and select the $m$ leading
   eigenvectors to construct the transformation matrix $W$.
3: (Semisupervised TCA) Eigendecompose matrix
   $(K(L + \lambda)\mathcal{L}K + \mu I)^{-1} KH\widetilde{K}_{yy}HK$ and select the $m$
   leading eigenvectors to construct the transformation
   matrix $W$.
4: return transformation matrix $W$.
```

Basics of Deep Transfer Learning

- Deep Learning models are hot in everything from games to recognizing cats.
- Apply Transfer Learning algorithms on Deap Learning models, such as: LeNet、AlexNet、GoogLeNet、VGG、ResNet.
- It's all about transfering Neural Network features(Layer Transfer).
- The usual transfer learning approach is to train a base network and then copy its first n layers to the first n layers of a target network. The remaining layers of the target network are then randomly initialized and trained toward the target task.

How transferable are features in deep neural networks?



Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In Ad- vances in Neural Information Processing Systems (NIPS), 2014.

How transferable are features in deep neural networks?

- *frozen*
  Transferred feature layers do not change during training on the new task.
- *fine-tune*
  Backpropagate the errors from the new task into the base (copied) features to fine-tune them to the new task

About the experiment result

- Why?

Typically, for a deep neural network, feature layers transition from *general* to *specific* by the last layer of the network.

- What does that mean?

It demonstrates that deep nerual network is transferable and can obtain a good performance if it is fine-tuned.

How to do Deep Transfer Learning?

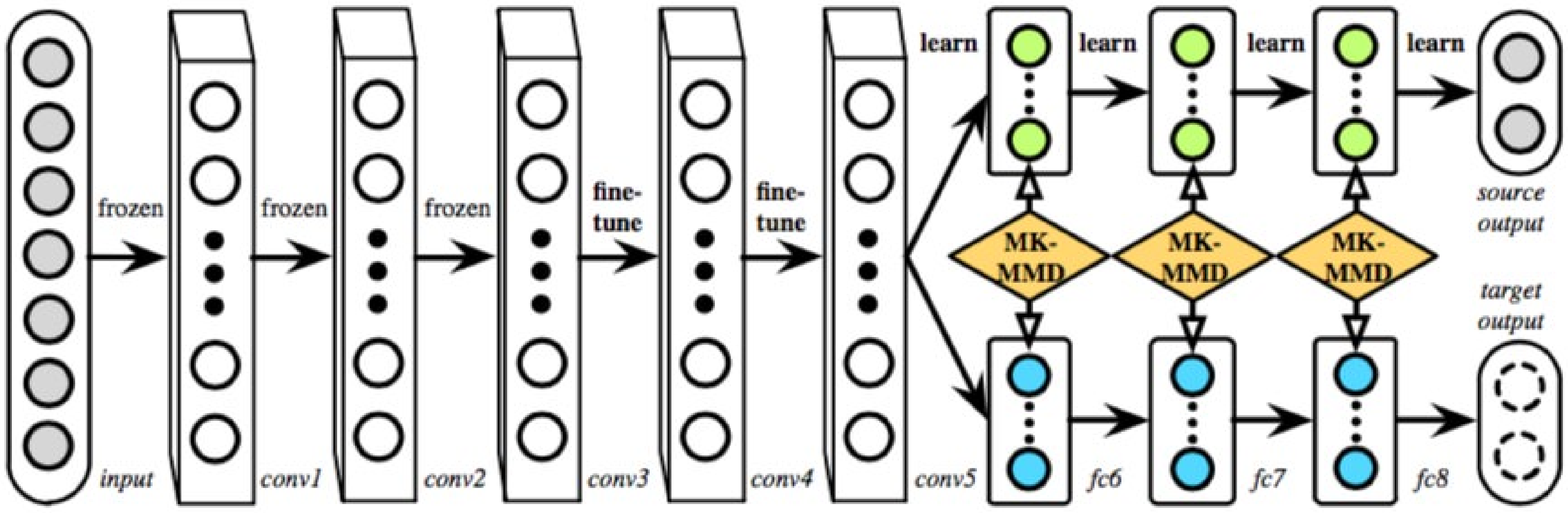
- For general feature layers

Simply copy them, forzen or do fine-tuned.

- How about the specific feature layer?

Long M, Cao Y, Wang J, et al. Learning transferable features with deep adaptation networks International Conference on Machine Learning. 2015 97-105.

Deep Adaptation Network(DAN)



- The features extracted by convolutional layers $conv1$-$conv3$ are general, hence these layers are frozen.
- The features extracted by layers $conv4$–$conv5$ are slightly less transferable, hence these layers are learned via fine-tuning.
- Fully connected layers $fc6$–$fc8$ are tailored to fit specific tasks, not transferable and should be adapted with MK-MMD.

Deep Adaptation Network(DAN)

Main innovations:

- Multi-Layer adaptation

The last 3 full connected layers are adapted with MK-MMD.

- MK-MMD(Multiple Kernel Variant of MMD)

MK-MMD is introduced to measure the distance rather than MMD.

MK-MMD(Multiple Kernel Variant of MMD)

Generally speaking, the MMD chooses a pre-defined Kernel function to do the feature mapping. However, in terms of performance, that kernel function we choosed may not be the best one. Thus, we should try different Kernel functions(Gaussian, Polynomial, Linear...) to find the best one. That is MK-MMD.

Deep Adaptation Network(DAN)

Optimization Objective

$$\min_{\Theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(X_i^a), y_i^a) + \lambda \sum_{l=l_1}^{l_2} d_k^2(D_s^l, D_t^l)$$

- Minmize cross-entropy loss function:
- MK-MMD-based multi-layer adaptation regularizer：

DAN Optimization Objective

Minmize Loss Function:

$$\min_{\Theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(X_i^a), y_i^a)$$

where

- $\Theta = \{W^l, b^l\}_{l=1}^{l}$
- J is the cross-entropy loss function: $J(p, q) = -\sum_x p(x)\ \log q(x)$
- $\theta(X_i^a)$ is the conditional probability that the CNN assigns X_i^a to label y_i^a

DAN Optimization Objective

MK-MMD-based multi-layer adaptation regularizer：

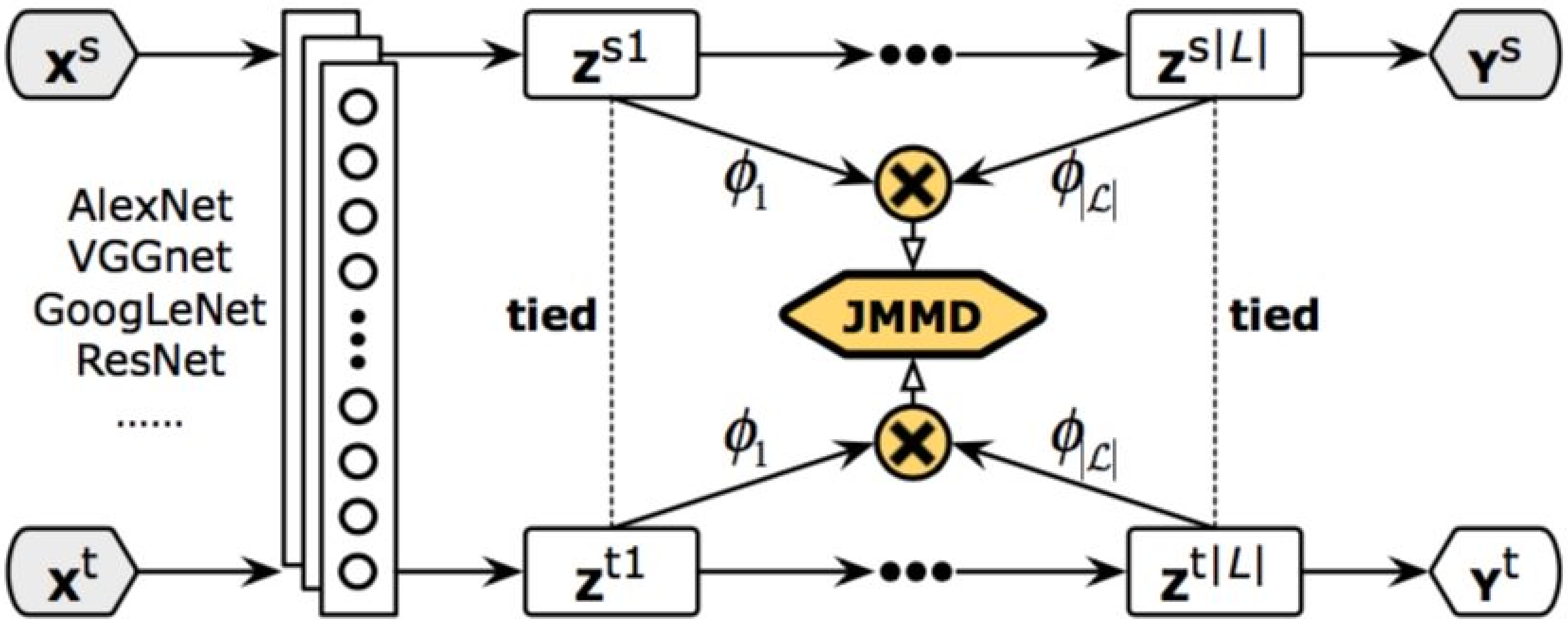
$$\lambda \sum_{l=l_1}^{l_2} d_k^2(D_s^l, D_t^l)$$

where $d_k^2(D_s^l, D_t^l)$ is a MK-MMD based on a certain kernel k.

Joint Adaptation Networks(JAN)

- Joint Adaptation Networks (JAN) learn a transfer network by aligning the joint distributions of multiple domain-specific layers across domains based on a joint maximum mean discrepancy (JMMD) criterion.
- Learning can be performed by stochastic gradient descent with the gradients computed by back-propagation in linear-time.

Joint Adaptation Networks(JAN)



(a) Joint Adaptation Network (JAN)

Mingsheng Long, Han Zhu, Jianmin Wang, Michael I. Jordan Deep Transfer Learning with Joint Adaptation Networks.International Conference on Machine Learning (ICML), 2017

JMMD(Joint Maximum Mean Discrepancy)

Following the virtue of MMD, we use the Hilbert space embeddings of joint distributions to measure the discrepancy of two joint distributions $P(Z^{s1},..,Z^{s|L|})$ and $Q(Z^{t1},..,Z^{t|L|})$,The resulting measure is called Joint Maximum Mean Discrepancy (JMMD), which is defined as:

$$D_L(P,Q) = ||C_{Z^{S,1:|L|}}(P) - C_{Z^{T,1:|L|}}(Q)||_{\bigotimes_{l=1}^{|L|} H^l}.$$

where

$$C_{X^{1:m}}(P) = E_{X^{1:m}}[\bigotimes_{l=1}^{m} \phi^l(X^l)]$$

Empirical estimate of $D_L(P, Q)$

The empirical estimate of $D_L(P, Q)$ is computed as the squared distance between the empirical kernel mean embeddings as:

$$\hat{D}_L(P, Q) = \frac{1}{n_s^2} \sum_{i=1}^{n_s} \sum_{j=1}^{n_s} \prod_{l \in L} k^l(Z_i^{sl}, Z_j^{sl})$$

$$+ \frac{1}{n_t^2} \sum_{i=1}^{n_t} \sum_{j=1}^{n_t} \prod_{l \in L} k^l(Z_i^{tl}, Z_j^{tl})$$

$$- \frac{2}{n_s n_t} \sum_{i=1}^{n_s} \sum_{j=1}^{n_t} \prod_{l \in L} k^l(Z_i^{sl}, Z_j^{tl})$$

A limitation of JMMD

- JMMD has quadratic complexity $O(n^2)$, which is inefficient for scalable deep transfer learning
- Motivated by the unbiased estimate of MMD (Gretton et al., 2012), we derive a similar linear-time estimate of JMMD as follows:

$$\begin{aligned}\widehat{D}_{\mathcal{L}}(P,Q) = \frac{2}{n}\sum_{i=1}^{n/2}\left(\prod_{\ell\in\mathcal{L}} k^{\ell}(\mathbf{z}_{2i-1}^{s\ell},\mathbf{z}_{2i}^{s\ell}) + \prod_{\ell\in\mathcal{L}} k^{\ell}(\mathbf{z}_{2i-1}^{t\ell},\mathbf{z}_{2i}^{t\ell})\right) \\ - \frac{2}{n}\sum_{i=1}^{n/2}\left(\prod_{\ell\in\mathcal{L}} k^{\ell}(\mathbf{z}_{2i-1}^{s\ell},\mathbf{z}_{2i}^{t\ell}) + \prod_{\ell\in\mathcal{L}} k^{\ell}(\mathbf{z}_{2i-1}^{t\ell},\mathbf{z}_{2i}^{s\ell})\right)\end{aligned}$$

Difference between MMD and JMMD

$$\widehat{D}_{\mathcal{H}}(P,Q) = \frac{1}{n_s^2}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s} k\left(\mathbf{x}_i^s, \mathbf{x}_j^s\right) + \frac{1}{n_t^2}\sum_{i=1}^{n_t}\sum_{j=1}^{n_t} k\left(\mathbf{x}_i^t, \mathbf{x}_j^t\right) - \frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_t} k\left(\mathbf{x}_i^s, \mathbf{x}_j^t\right), \tag{6}$$

$$\widehat{D}_{\mathcal{L}}(P,Q) = \frac{1}{n_s^2}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s}\prod_{\ell\in\mathcal{L}} k^{\ell}\left(\mathbf{z}_i^{s\ell}, \mathbf{z}_j^{s\ell}\right) + \frac{1}{n_t^2}\sum_{i=1}^{n_t}\sum_{j=1}^{n_t}\prod_{\ell\in\mathcal{L}} k^{\ell}\left(\mathbf{z}_i^{t\ell}, \mathbf{z}_j^{t\ell}\right) - \frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_t}\prod_{\ell\in\mathcal{L}} k^{\ell}\left(\mathbf{z}_i^{s\ell}, \mathbf{z}_j^{t\ell}\right). \tag{9}$$

JAN Optimization Objective

$$\min_f \frac{1}{n_s} \sum_{i=1}^{n_s} J(f(X_i^s), y_s^s) + \lambda \hat{D}_L(P, Q)$$

We set $L = \{fc6, fc7, fc8\}$ for the JAN model based on AlexNet (last three layers) while we set $L = \{pool5, fc\}$ for the JAN model based on ResNet (last two layers).

Experiments

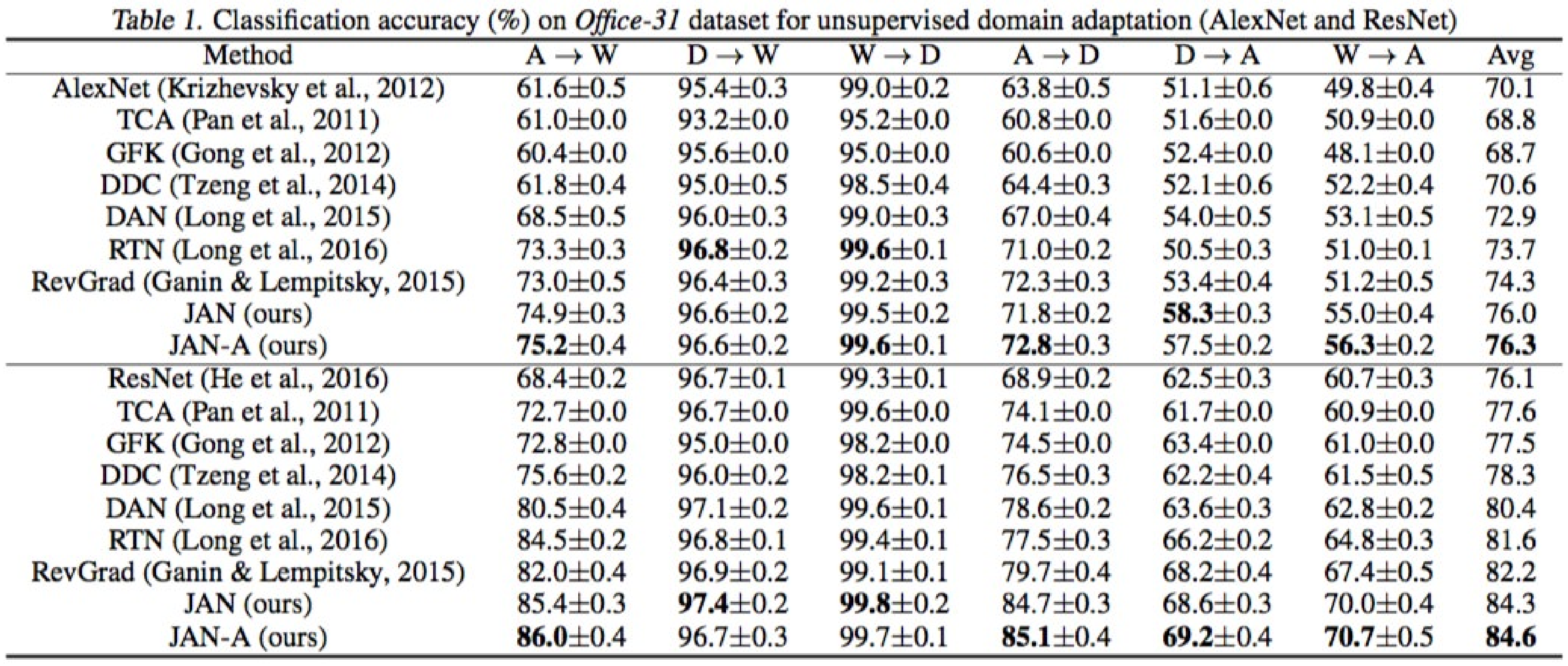


Table 1. Classification accuracy (%) on *Office-31* dataset for unsupervised domain adaptation (AlexNet and ResNet)

Method	A → W	D → W	W → D	A → D	D → A	W → A	Avg
AlexNet (Krizhevsky et al., 2012)	61.6±0.5	95.4±0.3	99.0±0.2	63.8±0.5	51.1±0.6	49.8±0.4	70.1
TCA (Pan et al., 2011)	61.0±0.0	93.2±0.0	95.2±0.0	60.8±0.0	51.6±0.0	50.9±0.0	68.8
GFK (Gong et al., 2012)	60.4±0.0	95.6±0.0	95.0±0.0	60.6±0.0	52.4±0.0	48.1±0.0	68.7
DDC (Tzeng et al., 2014)	61.8±0.4	95.0±0.5	98.5±0.4	64.4±0.3	52.1±0.6	52.2±0.4	70.6
DAN (Long et al., 2015)	68.5±0.5	96.0±0.3	99.0±0.3	67.0±0.4	54.0±0.5	53.1±0.5	72.9
RTN (Long et al., 2016)	73.3±0.3	**96.8**±0.2	**99.6**±0.1	71.0±0.2	50.5±0.3	51.0±0.1	73.7
RevGrad (Ganin & Lempitsky, 2015)	73.0±0.5	96.4±0.3	99.2±0.3	72.3±0.3	53.4±0.4	51.2±0.5	74.3
JAN (ours)	74.9±0.3	96.6±0.2	99.5±0.2	71.8±0.2	**58.3**±0.3	55.0±0.4	76.0
JAN-A (ours)	**75.2**±0.4	96.6±0.2	**99.6**±0.1	**72.8**±0.3	57.5±0.2	**56.3**±0.2	**76.3**
ResNet (He et al., 2016)	68.4±0.2	96.7±0.1	99.3±0.1	68.9±0.2	62.5±0.3	60.7±0.3	76.1
TCA (Pan et al., 2011)	72.7±0.0	96.7±0.0	99.6±0.0	74.1±0.0	61.7±0.0	60.9±0.0	77.6
GFK (Gong et al., 2012)	72.8±0.0	95.0±0.0	98.2±0.0	74.5±0.0	63.4±0.0	61.0±0.0	77.5
DDC (Tzeng et al., 2014)	75.6±0.2	96.0±0.2	98.2±0.1	76.5±0.3	62.2±0.4	61.5±0.5	78.3
DAN (Long et al., 2015)	80.5±0.4	97.1±0.2	99.6±0.1	78.6±0.2	63.6±0.3	62.8±0.2	80.4
RTN (Long et al., 2016)	84.5±0.2	96.8±0.1	99.4±0.1	77.5±0.3	66.2±0.2	64.8±0.3	81.6
RevGrad (Ganin & Lempitsky, 2015)	82.0±0.4	96.9±0.2	99.1±0.1	79.7±0.4	68.2±0.4	67.4±0.5	82.2
JAN (ours)	85.4±0.3	**97.4**±0.2	**99.8**±0.2	84.7±0.3	68.6±0.3	70.0±0.4	84.3
JAN-A (ours)	**86.0**±0.4	96.7±0.3	99.7±0.1	**85.1**±0.4	**69.2**±0.4	**70.7**±0.5	**84.6**

Experiments

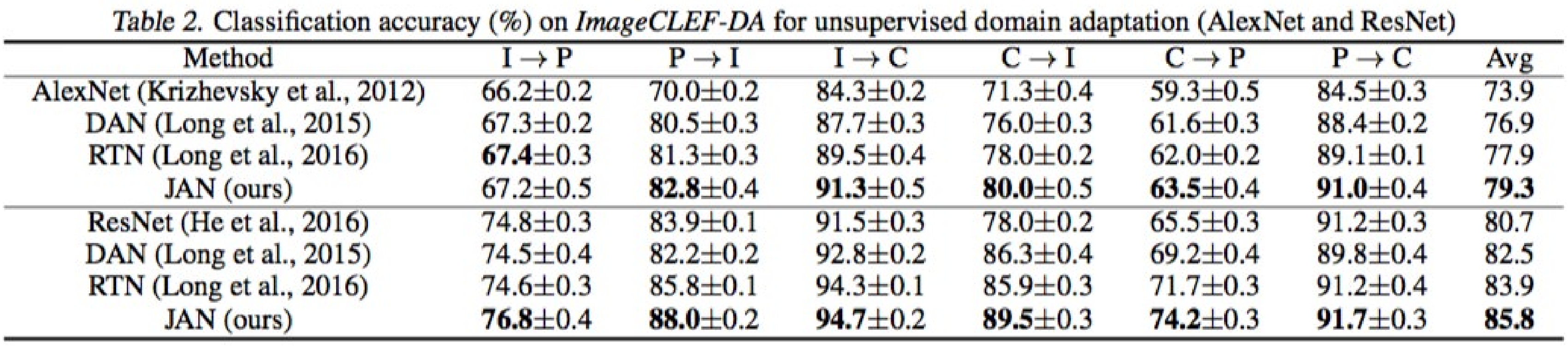
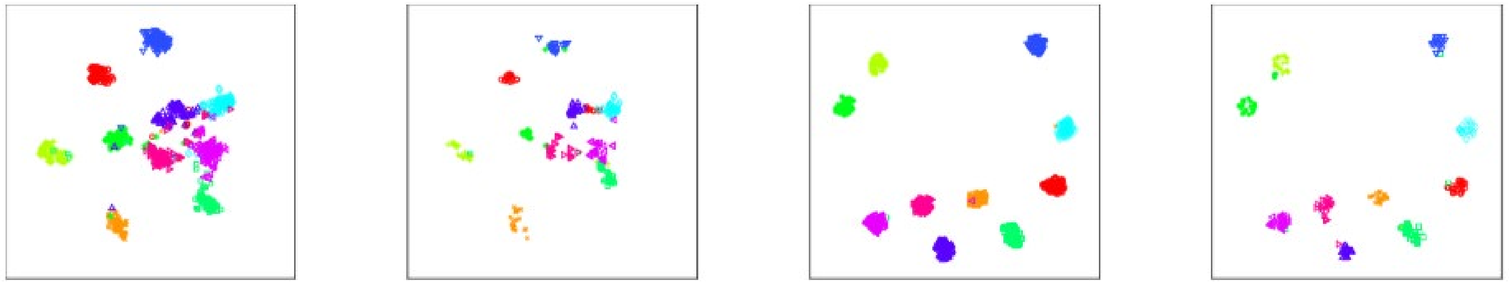


Table 2. Classification accuracy (%) on *ImageCLEF-DA* for unsupervised domain adaptation (AlexNet and ResNet)

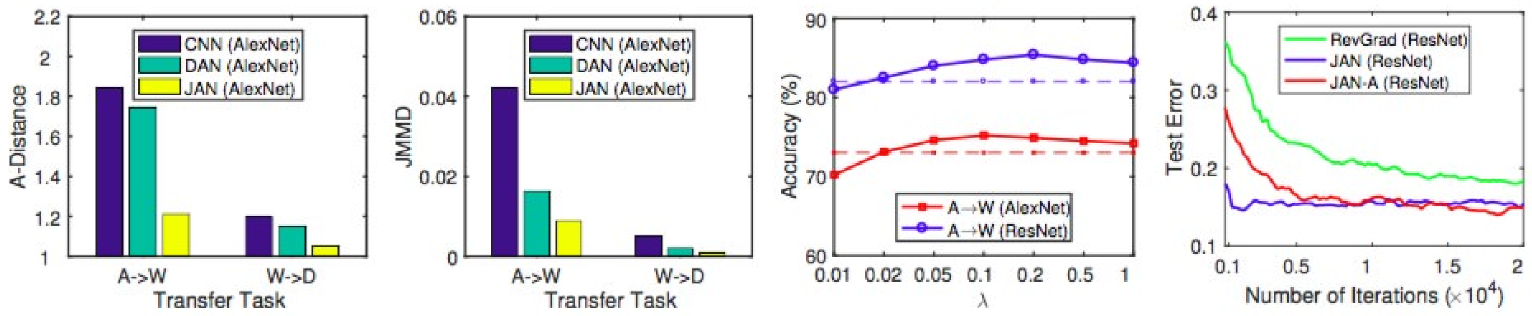
Method	$I \rightarrow P$	$P \rightarrow I$	$I \rightarrow C$	$C \rightarrow I$	$C \rightarrow P$	$P \rightarrow C$	Avg
AlexNet (Krizhevsky et al., 2012)	66.2±0.2	70.0±0.2	84.3±0.2	71.3±0.4	59.3±0.5	84.5±0.3	73.9
DAN (Long et al., 2015)	67.3±0.2	80.5±0.3	87.7±0.3	76.0±0.3	61.6±0.3	88.4±0.2	76.9
RTN (Long et al., 2016)	**67.4**±0.3	81.3±0.3	89.5±0.4	78.0±0.2	62.0±0.2	89.1±0.1	77.9
JAN (ours)	67.2±0.5	**82.8**±0.4	**91.3**±0.5	**80.0**±0.5	**63.5**±0.4	**91.0**±0.4	**79.3**
ResNet (He et al., 2016)	74.8±0.3	83.9±0.1	91.5±0.3	78.0±0.2	65.5±0.3	91.2±0.3	80.7
DAN (Long et al., 2015)	74.5±0.4	82.2±0.2	92.8±0.2	86.3±0.4	69.2±0.4	89.8±0.4	82.5
RTN (Long et al., 2016)	74.6±0.3	85.8±0.1	94.3±0.1	85.9±0.3	71.7±0.3	91.2±0.4	83.9
JAN (ours)	**76.8**±0.4	**88.0**±0.2	**94.7**±0.2	**89.5**±0.3	**74.2**±0.3	**91.7**±0.3	**85.8**

Experiments



(a) DAN: *Source*=**A** (b) DAN: *Target*=**W** (c) JAN: *Source*=**A** (d) JAN: *Target*=**W**

Figure 2. The t-SNE visualization of network activations (ResNet) generated by DAN (a)(b) and JAN (c)(d), respectively.



(a) $\mathcal{A}$-distance (b) JMMD (c) Accuracy w.r.t. λ (d) Convergence

Figure 3. Analysis: (a) $\mathcal{A}$-distance; (b) JMMD; (c) parameter sensitivity of λ; (d) convergence (dashed lines show best baseline results).

References:

- Pan, S. J. and Yang, Q. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering (TKDE), 22(10):1345–1359, 2010
- Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. Domain adaptation via transfer component analysis IEEE Transactions on Neural Networks (TNN), 22(2):199–210, 2011.
- Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In Ad- vances in Neural Information Processing Systems (NIPS), 2014.
- Long M, Cao Y, Wang J, et al. Learning transferable features with deep adaptation networks. International Conference on Machine Learning. 2015 97-105.
- Mingsheng Long, Han Zhu, Jianmin Wang, Michael I. Jordan Deep Transfer Learning with Joint Adaptation Networks.International Conference on Machine Learning (ICML), 2017

Thank You