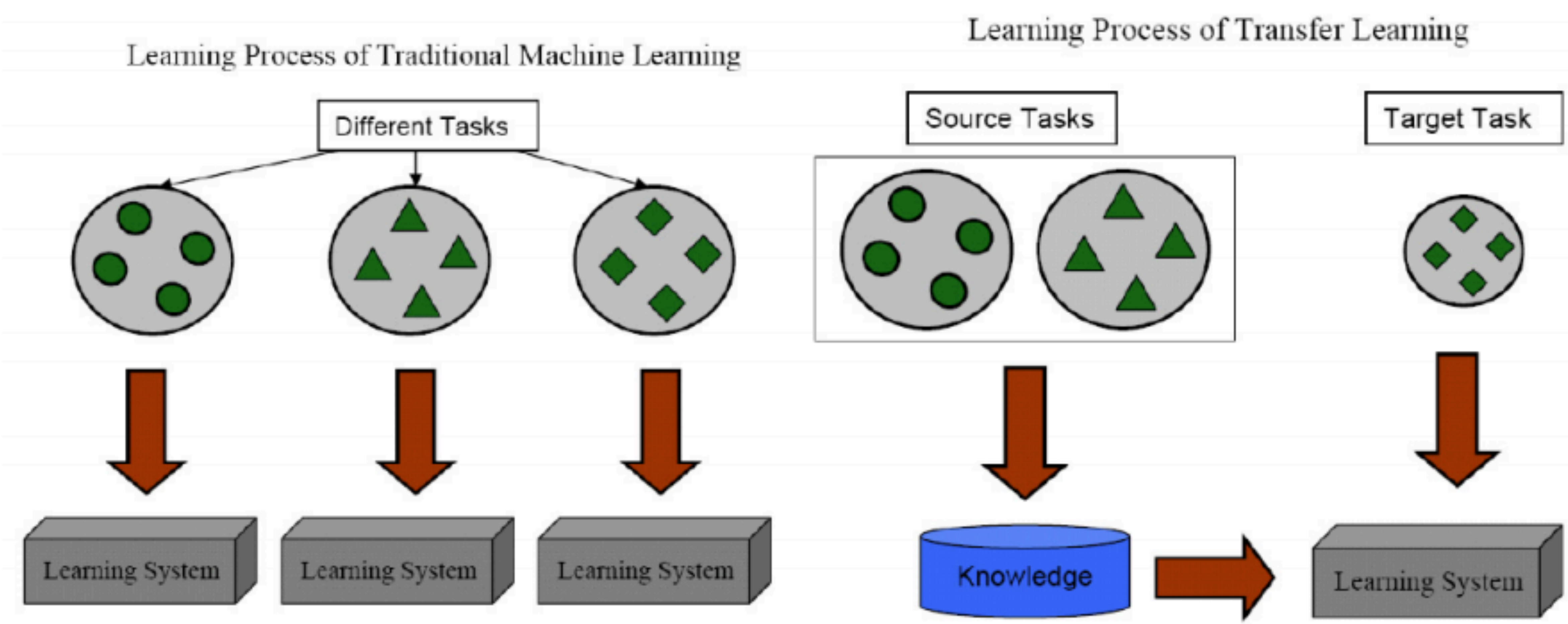
Deep Domain Adaptation Methods

Condy Chen

Outline

- Basic introduction and a simple method（DCC)
- Conditional Adversarial Domain Adaptation
- Self-ensembling for visual domain adaptation

Basic Introduction



(a) Traditional Machine Learning (b) Transfer Learning

Basic Introduction

- Source domain: $\boldsymbol{D}_S = \{\mathbf{X}_S, P(X_S)\}$
- Source task: $\boldsymbol{T}_S = \{Y_S, f_S(\cdot)\}$
- Target domain: $\boldsymbol{D}_T = \{\mathbf{X}_T, P(X_T)\}$
- Target task: $\boldsymbol{T}_T = \{Y_T, f_T(\cdot)\}$
- Goal: $\min \epsilon(f_T(\mathbf{X}_T), Y_T)$
- Conditions: $\boldsymbol{D}_T \neq \boldsymbol{D}_S$ or $\boldsymbol{T}_T \neq \boldsymbol{T}_S$ with $(\boldsymbol{D}_T, \boldsymbol{D}_S, Y_T, Y_S)$ may be unknown, respectively

Basic Introduction

Inductive transfer learning

Given $T_S \neq T_T$ under conditions:

- A lot of labeled D_S or
- No labeled D_S

Transductive transfer learning

Given $T_S = T_T$ under conditions:

- $\mathbf{X}_S \neq \mathbf{X}_T$ or
- $\mathbf{X}_S = \mathbf{X}_T$ and $P(X_S) \neq P(X_T)$

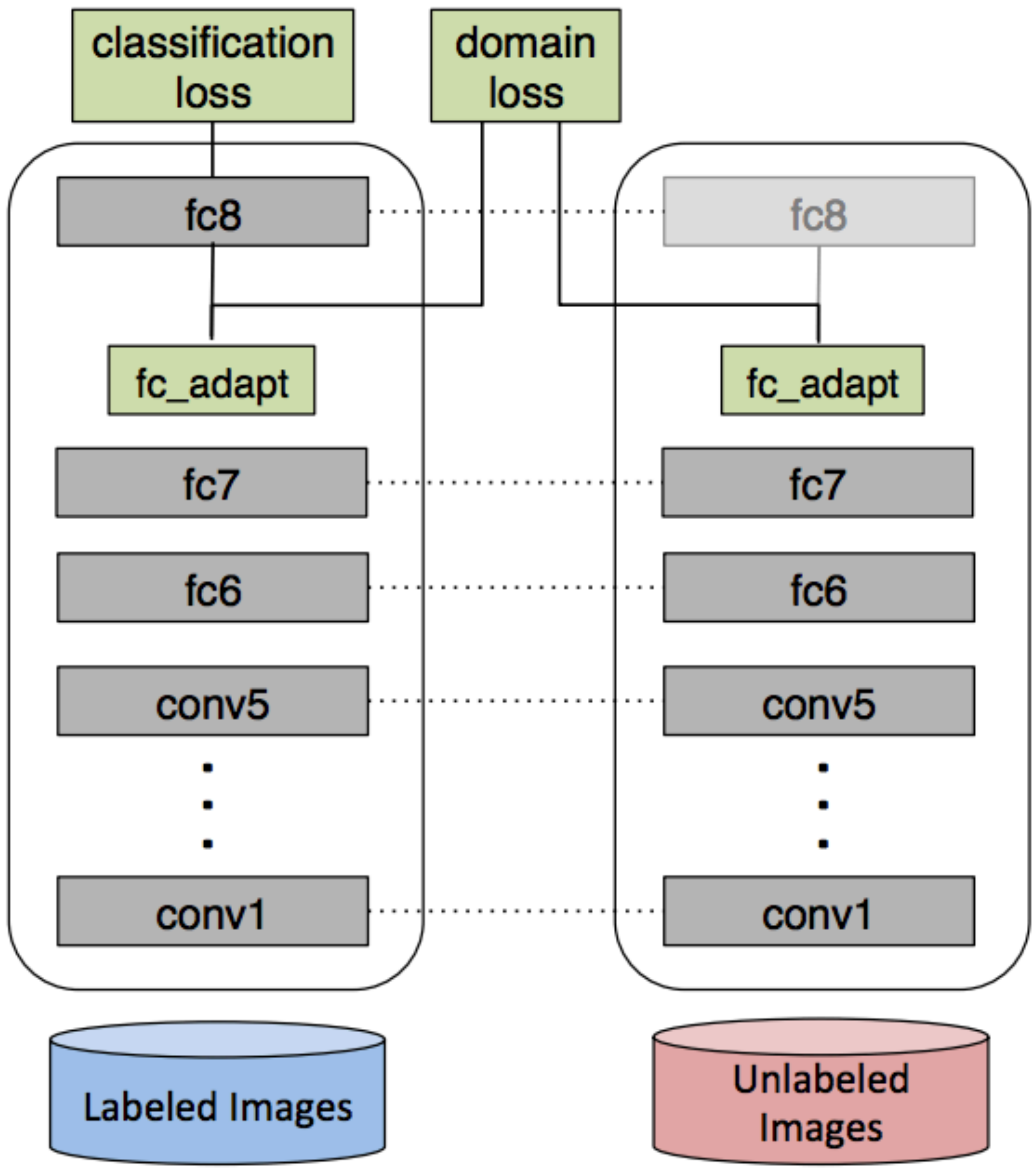
Unsupervised transfer learning

Given $T_S \neq T_T$ under conditions:

- No labeled D_S and D_T

Transfer Learning Settings	Related Areas	Source Domain Labels	Target Domain Labels	Tasks
Inductive Transfer Learning	Multi-task Learning	Available	Available	Regression, Classification
	Self-taught Learning	Unavailable	Available	Regression, Classification
Transductive Transfer Learning	Domain Adaptation, Sample Selection Bias, Co-variate Shift	Available	Unavailable	Regression, Classification
Unsupervised Transfer Learning		Unavailable	Unavailable	Clustering, Dimensionality Reduction

A simple method（DDC)



A simple method (DDC)

$$\mathrm{MMD}(X_S, X_T) = \left\| \frac{1}{|X_S|} \sum_{x_s \in X_S} \phi(x_s) - \frac{1}{|X_T|} \sum_{x_t \in X_T} \phi(x_t) \right\| \quad (1)$$

```python
def mmd_linear(f_of_X, f_of_Y):
    delta = f_of_X - f_of_Y
    loss = torch.mean(torch.mm(delta, torch.transpose(delta, 0, 1)))
    return loss
```

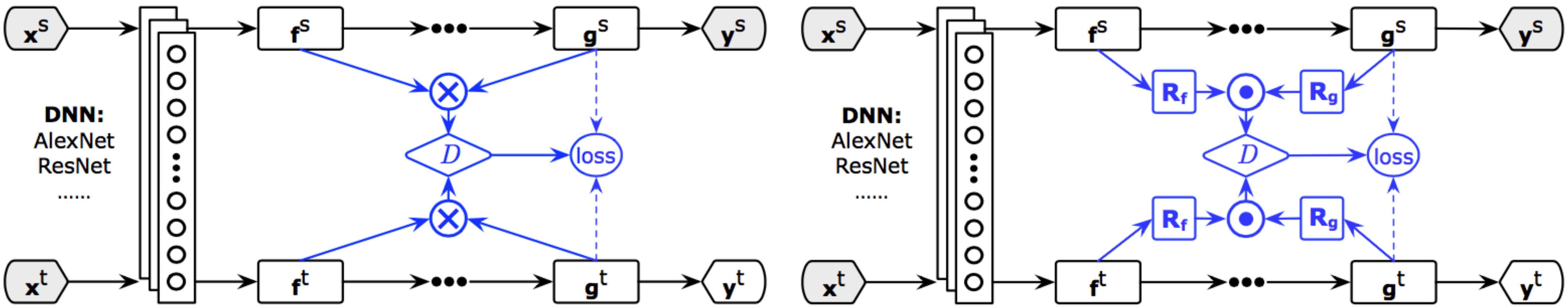
$$\mathcal{L} = \mathcal{L}_C(X_L, y) + \lambda \mathrm{MMD}^2(X_S, X_T)$$

Conditional Adversarial Domain Adaptation

Mingsheng Long[†], **Zhangjie Cao**[†], **Jianmin Wang**[†], **and Michael I. Jordan**[♯]
[†]School of Software, Tsinghua University, China
[†]KLiss, MOE; BNRist; Research Center for Big Data, Tsinghua University, China
[♯]University of California, Berkeley, Berkeley, USA
{mingsheng,jimwang}@tsinghua.edu.cn caozhangjie14@gmail.com
jordan@berkeley.edu

NIPS 2018

Conditional Adversarial Domain Adaptation



(a) Multilinear conditioning (CDAN-M) (b) Randomized Multilinear conditioning (CDAN-RM)

Conditional Adversarial Domain Adaptation

Conditional Discriminator

$$E_G = \frac{1}{n_s} \sum_{i=1}^{n_s} L\left(G\left(\mathbf{x}_i^s\right), \mathbf{y}_i^s\right), \tag{1}$$

$$E_{D,G} = -\frac{1}{n_s} \sum_{i=1}^{n_s} \log\left[D\left(\mathbf{f}_i^s, \mathbf{g}_i^s\right)\right] - \frac{1}{n_t} \sum_{j=1}^{n_t} \log\left[1 - D\left(\mathbf{f}_j^t, \mathbf{g}_j^t\right)\right], \tag{2}$$

Goal

$$\begin{aligned} &\min_G E_G - \lambda E_{D,G} \\ &\min_D E_{D,G} \end{aligned} \tag{3}$$

Conditional Adversarial Domain Adaptation

Multilinear Conditioning

Taking the advantage of multilinear map, in this paper, we condition D on $\mathbf{g}$ with the multilinear map

$$T_{\otimes}(\mathbf{f}, \mathbf{g}) = \mathbf{f} \otimes \mathbf{g}, \tag{4}$$

where $T_{\otimes}$ is a multilinear map and $D(\mathbf{f}, \mathbf{g}) = D(\mathbf{f} \otimes \mathbf{g})$. As such, the conditional domain discrimi-

Dimension Explosion？ ？

$$T(\mathbf{h}) = \begin{cases} T_{\otimes}(\mathbf{f}, \mathbf{g}) & \text{if } d_f \times d_g \leqslant 4096 \\ T_{\odot}(\mathbf{f}, \mathbf{g}) & \text{otherwise,} \end{cases}$$

$$T_{\odot}(\mathbf{f}, \mathbf{g}) = \frac{1}{\sqrt{d}} (\mathbf{R_f f}) \odot (\mathbf{R_g g}),$$

Conditional Adversarial Domain Adaptation

Entropy Conditioning

$$H(\mathbf{g}) = -\sum_{c=1}^{C} g_c \log g_c$$

The certainty of predictions can be computed by $e^{-H(\mathbf{g})} \in [\frac{1}{C}, 1]$.

$$E_{D,G} = -\frac{1}{n_s}\sum_{i=1}^{n_s} e^{-H(\mathbf{g}_i^s)} \log\left[D\left(T\left(\mathbf{h}_i^s\right)\right)\right] - \frac{1}{n_t}\sum_{j=1}^{n_t} e^{-H\left(\mathbf{g}_j^t\right)} \log\left[1 - D\left(T\left(\mathbf{h}_j^t\right)\right)\right].$$

Conditional Adversarial Domain Adaptation

Conditional Domain Adversarial Network

$$\min_{G} \frac{1}{n_s}\sum_{i=1}^{n_s} L\left(G\left(\mathbf{x}_i^s\right), \mathbf{y}_i^s\right)$$

$$+\frac{\lambda}{n_s}\sum_{i=1}^{n_s} e^{-H\left(\mathbf{g}_i^s\right)} \log\left[D\left(T\left(\mathbf{h}_i^s\right)\right)\right] + \frac{\lambda}{n_t}\sum_{j=1}^{n_t} e^{-H\left(\mathbf{g}_j^t\right)} \log\left[1 - D\left(T\left(\mathbf{h}_j^t\right)\right)\right]$$

$$\max_{D} \frac{1}{n_s}\sum_{i=1}^{n_s} e^{-H\left(\mathbf{g}_i^s\right)} \log\left[D\left(T\left(\mathbf{h}_i^s\right)\right)\right] + \frac{1}{n_t}\sum_{j=1}^{n_t} e^{-H\left(\mathbf{g}_j^t\right)} \log\left[1 - D\left(T\left(\mathbf{h}_j^t\right)\right)\right]$$

Conditional Adversarial Domain Adaptation

Experience Result

Table 1: Accuracy (%) on *Office-31* for unsupervised domain adaptation (AlexNet and ResNet)

Method	A → W	D → W	W → D	A → D	D → A	W → A	Avg
AlexNet [27]	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
DAN [29]	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 [31]	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
DANN [13]	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
ADDA [51]	73.5±0.6	96.2±0.4	98.8±0.4	71.6±0.4	54.6±0.5	53.5±0.6	74.7
JAN [30]	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
CDAN-RM	77.9±0.3	96.9±0.2	**100.0**±.0	75.1±0.2	54.5±0.3	**57.5**±0.4	77.0
CDAN-M	**78.3**±0.2	**97.2**±0.1	**100.0**±.0	**76.3**±0.1	57.3±0.2	57.3±0.3	**77.7**
ResNet-50 [20]	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
DAN [29]	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 [31]	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
DANN [13]	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
ADDA [51]	86.2±0.5	96.2±0.3	98.4±0.3	77.8±0.3	69.5±0.4	68.9±0.5	82.9
JAN [30]	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
GTA [43]	89.5±0.5	97.9±0.3	99.8±0.4	87.7±0.5	**72.8**±0.3	**71.4**±0.4	86.5
CDAN-RM	93.0±0.2	98.4±0.2	**100.0**±.0	89.2±0.3	70.2±0.4	67.4±0.4	86.4
CDAN-M	**93.1**±0.1	**98.6**±0.1	**100.0**±.0	**92.9**±0.2	71.0±0.3	69.3±0.3	**87.5**

Table 2: Accuracy (%) on *ImageCLEF-DA* for unsupervised domain adaptation (AlexNet and ResNet)

Method	I → P	P → I	I → C	C → I	C → P	P → C	Avg
AlexNet [27]	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 [29]	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
DANN [13]	66.5±0.6	81.8±0.3	89.0±0.4	79.8±0.6	63.5±0.5	88.7±0.3	78.2
JAN [30]	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
CDAN-RM	67±0.4	**84.8**±0.2	**92.4**±0.3	81.3±0.3	**64.7**±0.3	**91.6**±0.4	**80.3**
CDAN-M	**67.7**±0.3	83.3±0.1	91.8±0.2	**81.5**±0.2	63.0±0.2	91.5±0.3	79.8
ResNet-50 [20]	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 [29]	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
DANN [13]	75.0±0.6	86.0±0.3	96.2±0.4	87.0±0.5	74.3±0.5	91.5±0.6	85.0
JAN [30]	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
CDAN-RM	77.2±0.3	88.3±0.3	**98.3**±0.4	90.7±0.4	76.7±0.3	**94.0**±0.4	87.5
CDAN-M	**78.3**±0.3	**91.2**±0.2	96.7±0.3	**91.2**±0.3	**77.2**±0.2	93.7±0.3	**88.1**

Table 3: Accuracy (%) on *Office-Home* for unsupervised domain adaptation (AlexNet and ResNet)

Method	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg
AlexNet [27]	26.4	32.6	41.3	22.1	41.7	42.1	20.5	20.3	51.1	31.0	27.9	54.9	34.3
DAN [29]	31.7	43.2	55.1	33.8	48.6	50.8	30.1	35.1	57.7	44.6	39.3	63.7	44.5
DANN [13]	36.4	45.2	54.7	35.2	51.8	55.1	31.6	39.7	59.3	45.7	46.4	65.9	47.3
JAN [30]	35.5	46.1	57.7	36.4	53.3	54.5	33.4	40.3	60.1	45.9	47.4	67.9	48.2
CDAN-RM	36.2	47.3	58.6	37.3	54.4	**58.3**	33.2	**43.9**	62.1	48.2	48.1	70.7	49.9
CDAN-M	**38.1**	**50.3**	**60.3**	**39.7**	**56.4**	57.8	**35.5**	43.1	**63.2**	**48.4**	**48.5**	**71.1**	**51.0**
ResNet-50 [20]	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DAN [29]	43.6	57.0	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3
DANN [13]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
JAN [30]	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
CDAN-RM	49.2	64.8	72.9	53.8	62.4	62.9	49.8	48.8	71.5	65.8	56.4	79.2	61.5
CDAN-M	**50.6**	**65.9**	**73.4**	**55.7**	**62.7**	**64.2**	**51.8**	**49.1**	**74.5**	**68.2**	**56.9**	**80.7**	**62.8**

Table 4: Accuracy (%) on *Digits* and *VisDA-2017* for unsupervised domain adaptation (ResNet)

Method	M → U	U → M	S → M	Avg	Method	Synthetic → Real
UNIT [28]	96.0	93.6	**90.5**	93.4	JAN [30]	61.6
CyCADA [22]	95.6	96.5	90.4	94.2	GTA [43]	69.5
CDAN-M	**96.5**	**97.1**	89.2	**94.3**	**CDAN-M**	**70.3**

Conditional Adversarial Domain Adaptation

Experience Result

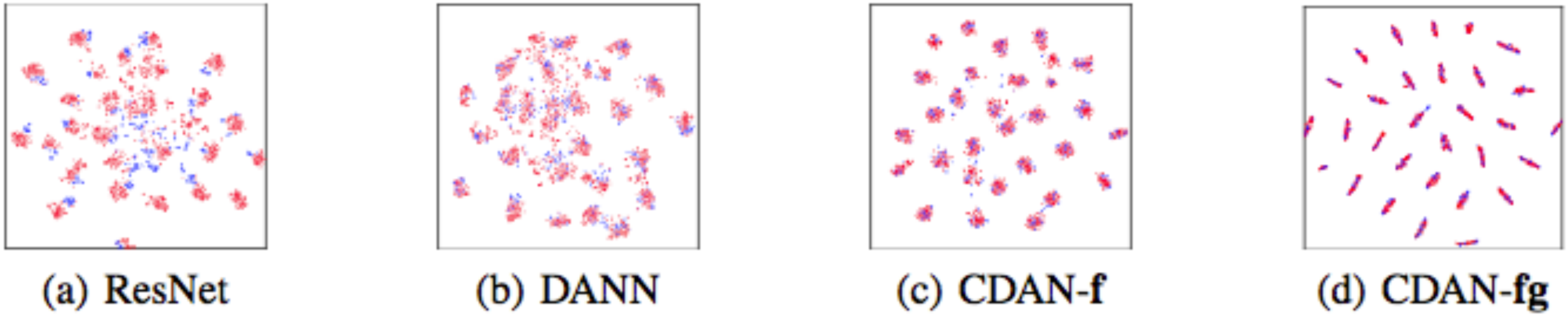


Figure 3: T-SNE of features by (a) ResNet, (b) DANN, (c) CDAN-f, (d) CDAN-fg (red: **A**; blue: **W**).

Self-ensembling for visual domain adaptation

French, G.
g.french@uea.ac.uk

Mackiewicz, M.
m.mackiewicz@uea.ac.uk

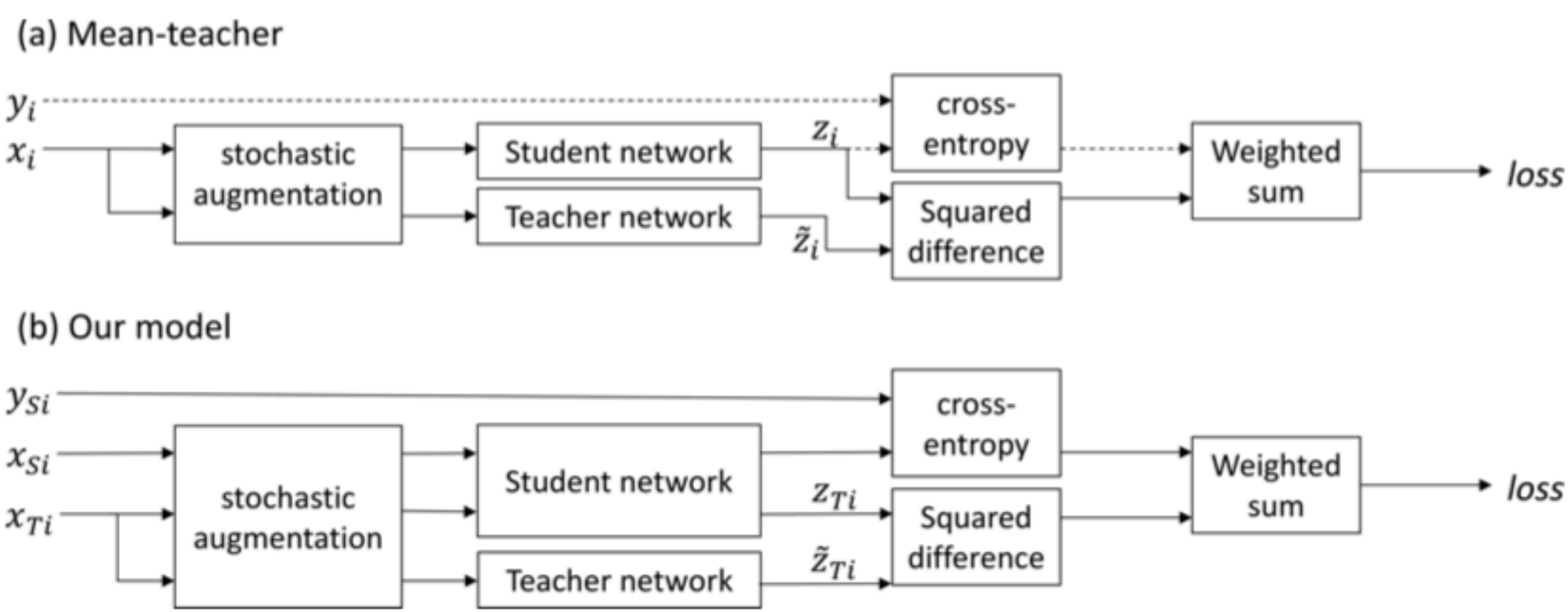
Fisher, M.
mark.fisher@uea.ac.uk

September 25, 2018

ICLR 2018

Self-ensembling for visual domain adaptation

Model Structure



Self-ensembling for visual domain adaptation

Confidence thresholding

$\tilde{f}_{Ti} = \max_{j \in C}(\tilde{z}_{Tij})$; the predicted probability of the predicted class of the sample. If $\tilde{f}_{Ti}$ is below the confidence threshold (a parameter search found 0.968 to be an effective value for small image benchmarks), the self-ensembling loss for the sample x_i is masked to 0.

Experiment Result

	USPS – MNIST	MNIST – USPS	SVHN – MNIST	MNIST – SVHN	CIFAR – STL	STL – CIFAR	Syn Digits – SVHN	Syn Signs – GTSRB
TRAIN ON SOURCE								
SupSrc*	77.55 ±0.8	82.03 ±1.16	66.5 ±1.93	25.44 ±2.8	72.84 ±0.61	51.88 ±1.44	86.86 ±0.86	96.95 ±0.36
SupSrc+TF	77.53 ±4.63	95.39 ±0.93	68.65 ±1.5	24.86 ±3.29	75.2 ±0.28	59.06 ±1.02	87.45 ±0.65	97.3 ±0.16
SupSrc+TFA	91.97 ±2.15	96.25 ±0.54	71.73 ±5.73	28.69 ±1.59	75.18 ±0.76	59.38 ±0.58	87.16 ±0.85	98.02 ±0.20
Specific aug.[b]	–	–	–	61.99 ±3.9	–	–	–	–
RevGrad[a] [1]	74.01	91.11	73.91	35.67	66.12	56.91	91.09	88.65
DCRN [2]	73.67	91.8	81.97	40.05	66.37	58.65	–	–
G2A [3]	90.8	92.5	84.70	36.4	–	–	–	–
ADDA [4]	90.1	89.4	76.00	–	–	–	–	–
ATT [5]	–	–	86.20	52.8	–	–	93.1	96.2
SBADA-GAN [6]	97.60	95.04	76.14	61.08	–	–	–	–
ADA [7]	–	–	97.6	–	–	–	91.86	97.66
OUR RESULTS								
MT+TF	98.07 ±2.82	**98.26** ±0.11	99.18 ±0.12	13.96[c] ±4.41	80.08 ±0.25	18.3 ±9.03	15.94 ±0.0	98.63 ±0.09
MT+CT*	92.35 ±8.61	88.14 ±0.34	93.33 ±5.88	33.87[c] ±4.02	77.53 ±0.11	71.65 ±0.67	96.01 ±0.08	98.53 ±0.15
MT+CT+TF	97.28 ±2.74	98.13 ±0.17	98.64 ±0.42	34.15[c] ±3.56	79.73 ±0.45	**74.24** ±0.46	96.51 ±0.08	98.66 ±0.12
MT+CT+TFA	**99.54** ±0.04	98.23 ±0.13	**99.26** ±0.05	37.49[c] ±2.44	**80.09** ±0.31	69.86 ±1.97	**97.11** ±0.04	**99.37** ±0.09
Specific aug.[b]	–	–	–	**97.0**[c] ±0.06	–	–	–	–
TRAIN ON TARGET								
SupTgt*	99.53 ±0.02	97.29 ±0.2	99.59 ±0.08	95.7 ±0.13	67.75 ±2.23	88.86 ±0.38	95.62 ±0.2	98.49 ±0.32
SupTgt+TF	99.62 ±0.04	97.65 ±0.17	99.61 ±0.04	96.19 ±0.1	70.98 ±0.79	89.83 ±0.39	96.18 ±0.09	98.64 ±0.09
SupTgt+TFA	99.62 ±0.03	97.83 ±0.17	99.59 ±0.06	96.65 ±0.11	70.03 ±1.13	90.44 ±0.38	96.59 ±0.09	99.22 ±0.22
Specific aug.[b]	–	–	–	97.16 ±0.05	–	–	–	–

Conclusions

Contributions

- CDAN presented a novel approach to domain adaption
- CDAN doesn’t match the feature representation across domains which is prone to under-matching like previous adversarial adaptation methods
- CDAN thinks about dimension explosion and gives a solution

Shortages

- CDAN lacked doing some experiments on transfer learning standard datasets such as cifar to STL and STL to cifar
- It is not clear in 《self-emsembling …》 why using MSE, instead of other difference function

Thanks!