Transferable Normalization: Towards Improving Transferability of Deep Neural Networks

Ximei Wang, Ying Jin, Mingsheng Long (✉)[1], Jianmin Wang, and Michael I. Jordan[♯]

School of Software, BNRist, Tsinghua University, China
Research Center for Big Data, Tsinghua University, China
National Engineering Laboratory for Big Data Software
[♯]University of California, Berkeley, Berkeley, USA

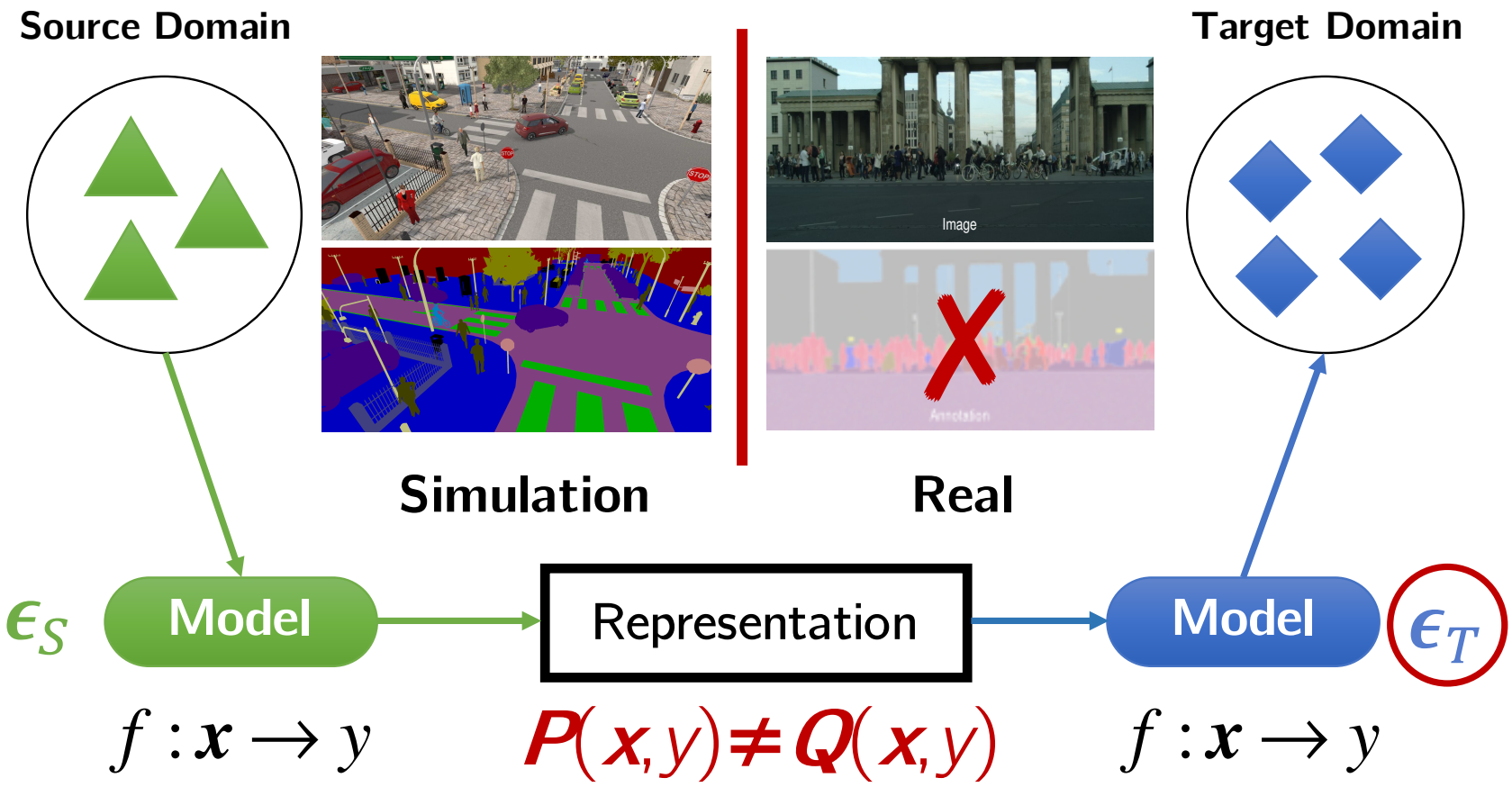
Neural Information Processing Systems (NeurIPS), 2019

[1]Corresponding author: Mingsheng Long (mingsheng@tsinghua.edu.cn)

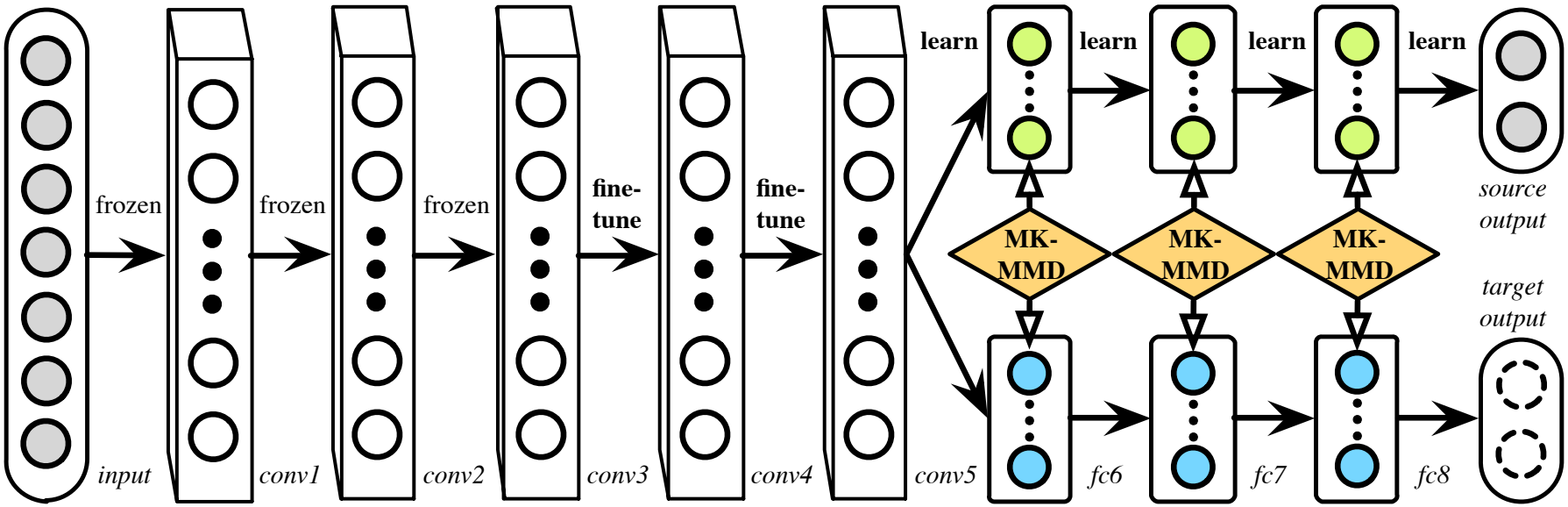
Domain Adaptation

Machine learning across domains of different distributions: $P \neq Q$

- Independent and Differently Distributed (IDD)



Deep Adaptation Network[2]

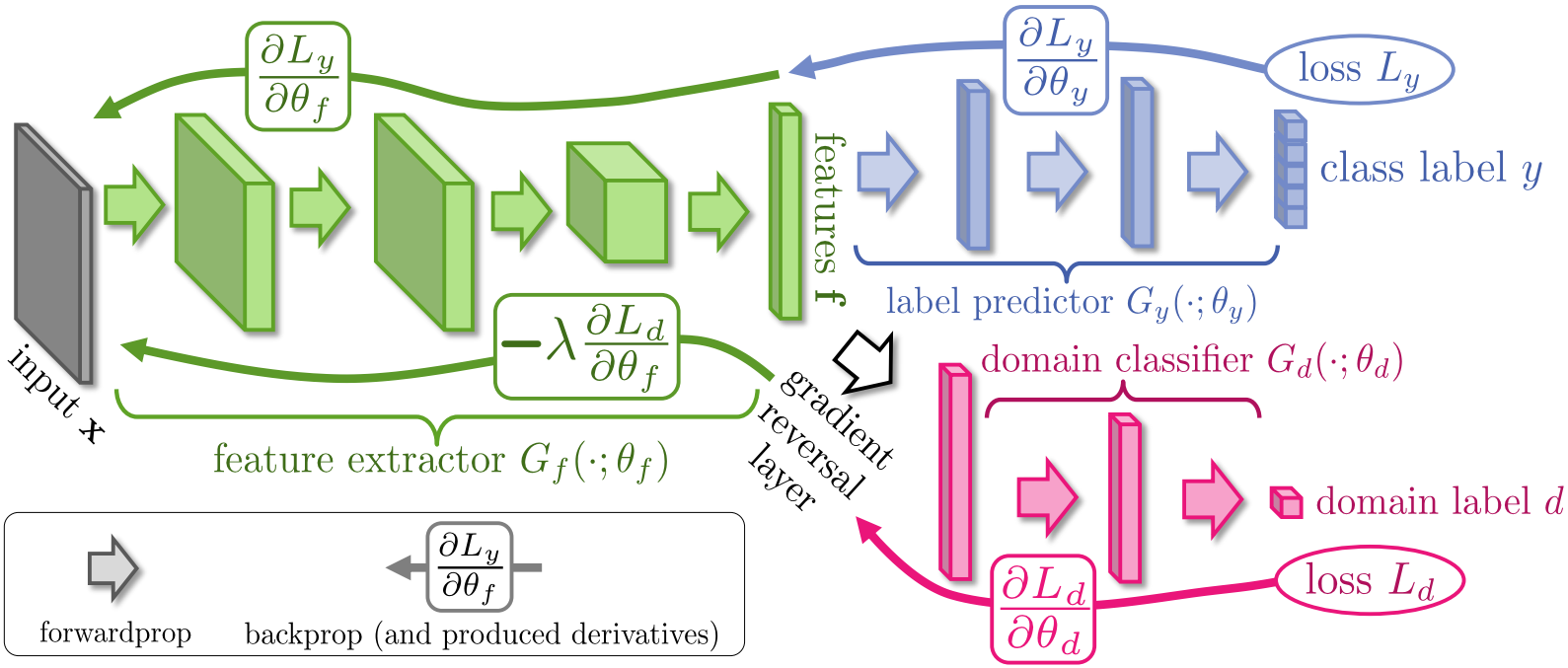


$$d_k^2(P,Q) \triangleq \|\boldsymbol{E}_P[\phi(\boldsymbol{x}^s)] - \boldsymbol{E}_Q[\phi(\boldsymbol{x}^t)]\|_{\mathcal{H}_k}^2$$

$$\min_{\theta\in\Theta} \max_{k\in\mathcal{K}} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(\boldsymbol{x}_i^a), y_i^a) + \lambda \sum_{l=l_1}^{l_2} d_k^2(\mathcal{D}_s^l, \mathcal{D}_t^l) \tag{1}$$

[2] Long et al. Learning Transferable Features with Deep Adaptation Networks. ICML 15

Domain Adversarial Network[3]

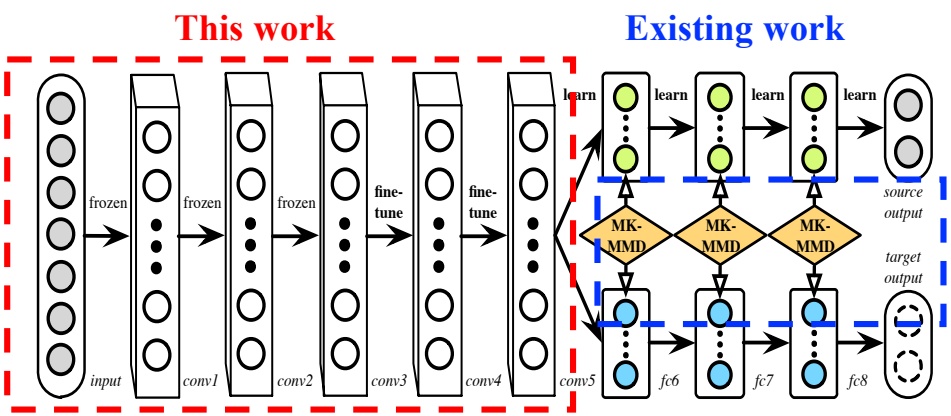


$$C_0\left(\theta_f,\theta_y,\theta_d\right)=\frac{1}{n_s}\sum_{\mathbf{x}_i\in\mathcal{D}_s}L_y\left(G_y\left(G_f\left(\mathbf{x}_i\right)\right),y_i\right)$$
$$-\frac{\lambda}{n}\sum_{\mathbf{x}_i\in(\mathcal{D}_s\cup\mathcal{D}_t)}L_d\left(G_d\left(G_f\left(\mathbf{x}_i\right)\right),d_i\right) \tag{2}$$

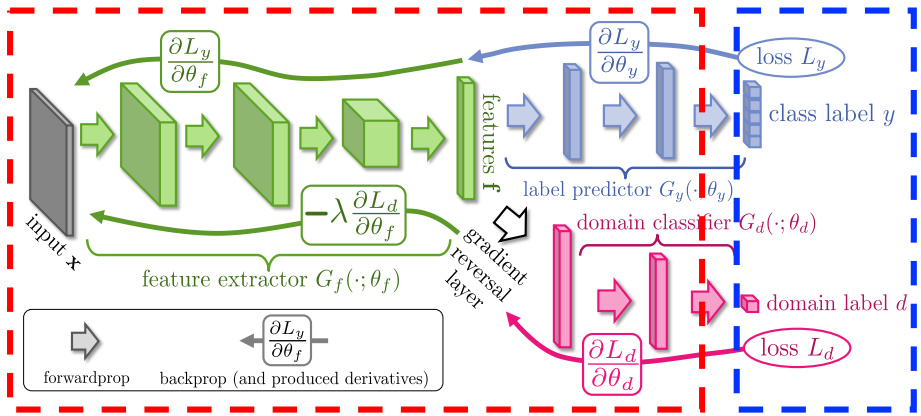
[3] Ganin et al. Domain-adversarial training of neural networks. JMLR 16

Motivations

- Existing works tackle domain adaptation by reducing the domain shift from the perspective of loss function designs.
  - Moment Matching: DAN, DDC, RTN, JAN,...
  - Adversarial Training: DANN, MADA, CDAN, MCD, MDD, ...
- Rare attention has been paid to the other side of the coin, i.e. network design, an indispensable part of domain adaptation



DAN (Moment Matching)



DANN (Adversarial Training)

Motivations

Each internal component of ResNet [4]

- *Convolutional layers* contribute to feature transformation and abstraction
- *Pooling layers* contribute to enhancing the translation invariance
- **Key points**: The invariance they achieve cannot bridge the domain gap

Batch Normalization (BN):

- First it normalizes inputs to be zero-mean and univariance
- Then it scales and shifts the normalized vectors with two trainable parameters to maintain expressiveness
- **Key points**: BN shares the spirit of statistics (moments) matching with domain adaptation

$$\widehat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sqrt{\sigma^{2(j)} + \epsilon}}, \quad y^{(j)} = \gamma^{(j)}\widehat{x}^{(j)} + \beta^{(j)} \tag{3}$$

[4] K. He et al. Deep residual learning for image recognition. In CVPR, 2016.

Limitations of BN for Domain Adaptation

A brute-force sharing of the mean and variance across domains:

- The representations from the source and target domains are *combined* into a single batch and fed as the input of the BN layer
- Distort the feature distributions and thereby deteriorate the network transferability

All channels in BN are equally important is unsuitable for DA:

- Some channels are easier to transfer than others since they constitute the sharing patterns of both domains
- Those more transferable channels should be highlighted

Questions: How to implement it in an effective way, retaining the *simplicity* of BN while not changing other components of network architectures?

Transferable Normalization (TransNorm)

- Previous: Batch Normalization (BN)

$$\widehat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sqrt{\sigma^{2(j)} + \epsilon}}, \quad y^{(j)} = \gamma^{(j)}\widehat{x}^{(j)} + \beta^{(j)} \tag{4}$$

- **Domain Specific Mean and Variance**

$$\widehat{x}_s = \frac{x_s - u_s}{\sqrt{\sigma_s^2 + \epsilon}}, \quad \widehat{x}_t = \frac{x_t - u_t}{\sqrt{\sigma_t^2 + \epsilon}} \tag{5}$$

- **Domain Sharing Gamma and Beta**

$$z_s = \gamma\widehat{x}_s + \beta, \quad z_t = \gamma\widehat{x}_t + \beta, \tag{6}$$

- **Domain Adaptive Alpha**
  - Can we quantify the transferability of different channels and adapt them differently across domains?

Transferable Normalization (TransNorm)

- **Domain Adaptive Alpha**
  - Calculating Distance: Means normalized by variance

$$d^{(j)} = \left| \frac{\mu_s^{(j)}}{\sqrt{\sigma_s^{2(j)} + \epsilon}} - \frac{\mu_t^{(j)}}{\sqrt{\sigma_t^{2(j)} + \epsilon}} \right|, \quad j = 1, 2, ..., c, \tag{7}$$

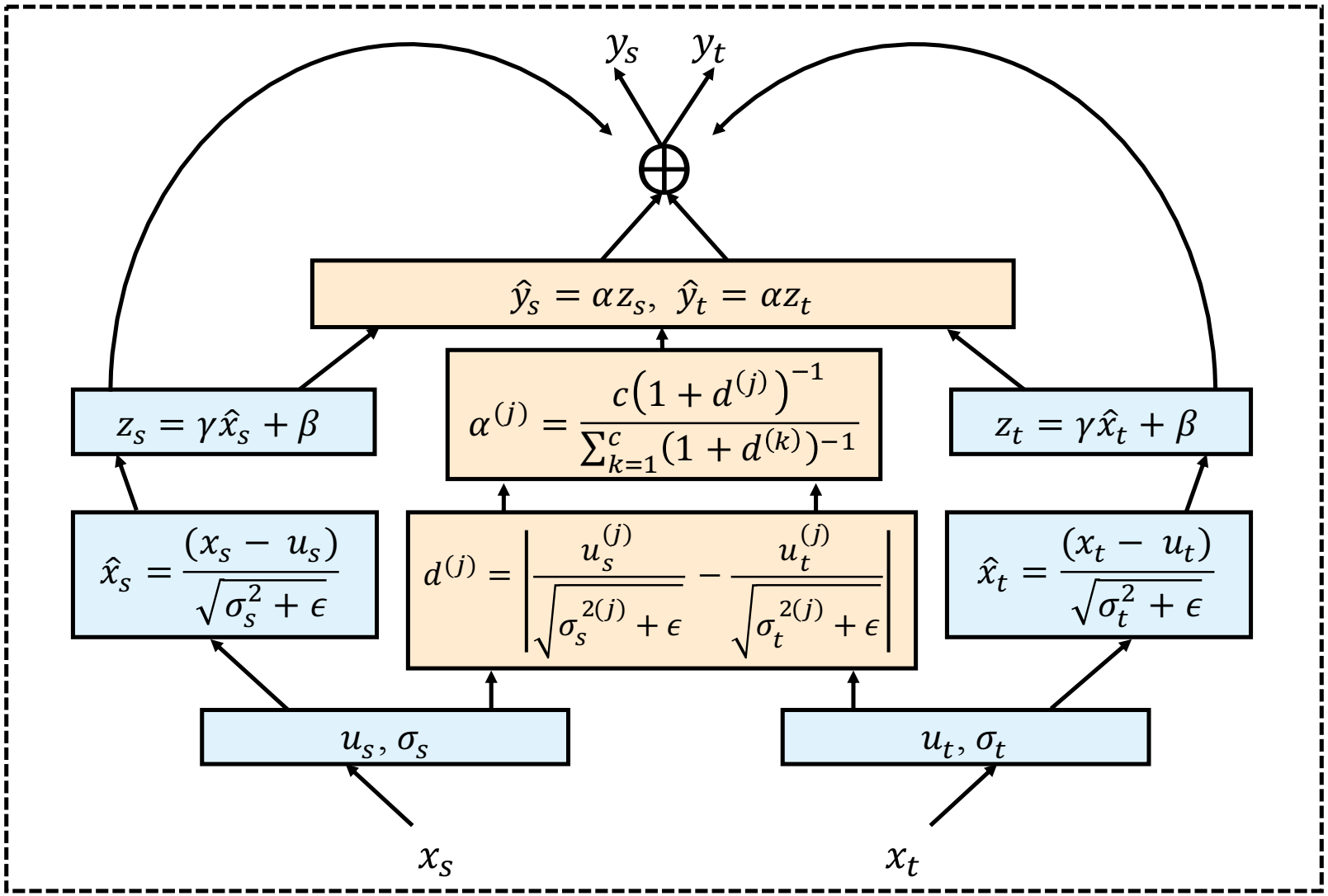
  - Generating Probability: Student-t distribution with heavier tails

$$\alpha^{(j)} = \frac{c(1 + d^{(j)})^{-1}}{\sum_{k=1}^{c} (1 + d^{(k)})^{-1}}, \quad j = 1, 2, ..., c, \tag{8}$$

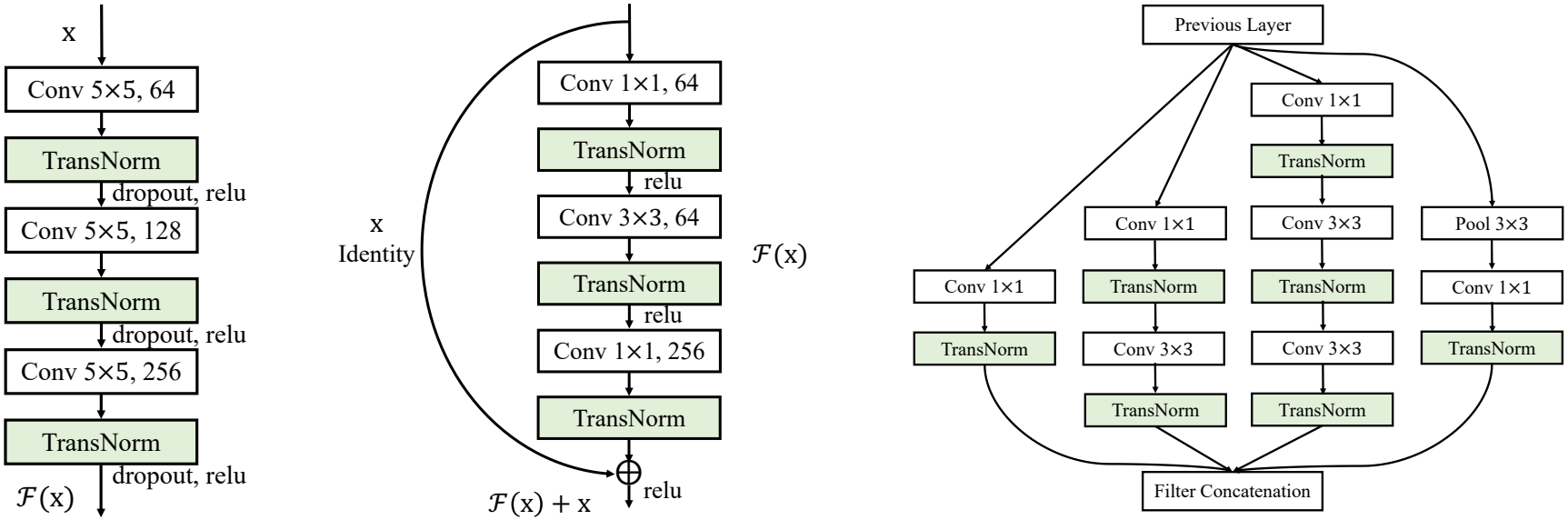
  - Residual Connection: Avoid overly penalizing the informative channels

$$y_s = (1 + \alpha) z_s, \quad y_t = (1 + \alpha) z_t. \tag{9}$$

Transferable Normalization (TransNorm)



TransNorm Layer



(a) CNN (b) Residual Block (c) Inception Block

Figure: Improving transferability of mainstream deep backbones by replacing BN with TransNorm.

Basic Results

Table 1: Accuracy (%) on Office-31 for unsupervised domain adaption (ResNet-50).

Method	A→W	D→W	W→D	A→D	D→A	W→A	Avg
ResNet-50 [9]	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 [17]	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 [19]	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
ADDA [37]	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 [20]	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
MADA [26]	90.0 ± 0.1	97.4 ± 0.1	99.6 ± 0.1	87.8 ± 0.2	70.3 ± 0.3	66.4 ± 0.3	85.2
SimNet [28]	88.6 ± 0.5	98.2 ± 0.2	99.7 ± 0.2	85.3 ± 0.3	73.4 ± 0.8	71.6 ± 0.6	86.2
GTA [33]	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
DANN+BN [4]	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
DANN+TransNorm	**91.8** ± 0.4	**97.7** ± 0.2	**100.0** ± 0.0	**88.0** ± 0.2	**68.2** ± 0.2	**70.4** ± 0.3	**86.0**
CDAN+BN [18]	94.1 ± 0.1	98.6 ± 0.1	**100.0** ± 0.0	92.9 ± 0.2	71.0 ± 0.3	69.3 ± 0.3	87.7
CDAN+TransNorm	**95.7** ± 0.3	**98.7** ± 0.2	**100.0** ± 0.0	**94.0** ± 0.2	**73.4** ± 0.4	**74.2** ± 0.3	**89.3**

Table 2: Accuracy (%) on Image-CLEF for unsupervised domain adaption (ResNet-50).

Method	I→P	P→I	I→C	C→I	C→P	P→C	Avg
ResNet-50 [9]	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 [17]	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+BN [4]	75.0 ± 0.3	86.0 ± 0.3	**96.2** ± 0.4	87.0 ± 0.5	74.3 ± 0.5	91.5 ± 0.6	85.0
DANN+TransNorm	**78.2** ± 0.4	**89.5**± 0.2	95.5 ± 0.2	**91.0** ± 0.2	**76.0** ± 0.2	**91.5** ± 0.3	**87.0**
CDAN+BN [18]	77.7 ± 0.3	90.7 ± 0.2	**97.7** ± 0.3	91.3 ± 0.3	74.2 ± 0.2	94.3 ± 0.3	87.7
CDAN+TransNorm	**78.3** ± 0.3	**90.8** ± 0.2	96.7 ± 0.4	**92.3** ± 0.2	**78.0** ± 0.1	**94.8** ± 0.3	**88.5**

Generalized Results

Table 3: Accuracy (%) on Office-Home for unsupervised domain adaption (ResNet-50).

Method (S:T)	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
ResNet-50 [9]	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 [17]	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
JAN [20]	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
DANN+BN [4]	**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
DANN+TransNorm	43.5	**60.9**	**72.1**	**51.0**	**61.5**	**62.5**	**49.6**	**46.8**	**70.4**	**63.7**	**52.2**	**77.9**	**59.3**
CDAN+BN [18]	**50.7**	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
CDAN+TransNorm	50.2	**71.4**	**77.4**	**59.3**	**72.7**	**73.1**	**61.0**	**53.1**	**79.5**	**71.9**	**59.0**	**82.9**	**67.6**

Table 4: Accuracy (%) on more adaptation methods, network backbones, and datasets.

Adaptation Method		Network Backbone				Dataset	
Method	Avg	Method (Inception)	Avg	Method (DTN)	S→M	Method	VisDA
DANN [4]	82.2	Inception-BN [12]	75.5	CyCADA [10]	90.4	GTA [33]	69.5
CDAN [18]	87.7	AdaBN [15]	76.7	MCD [32]	94.2	MCD [32]	69.7
MDD+BN [46]	88.8	**DANN+BN** [4]	82.1	**DANN+BN** [4]	73.9	**DANN+BN** [4]	63.7
		DANN+TransNorm	**83.2**	**DANN+TransNorm**	**77.0**	**DANN+TransNorm**	**66.3**
MDD+TransNorm	**89.5**	**CDAN+BN** [18]	84.9	**CDAN+BN** [18]	89.2	**CDAN+BN** [18]	70.0
		CDAN+TransNorm	**86.0**	**CDAN+TransNorm**	**94.7**	**CDAN+TransNorm**	**71.4**

Comparison and Ablation Study

Table 5: Accuracy comparison with BN, AdaBN and AutoDIAL (%) on Office-31 (ResNet-50).

Method	DANN [4]				CDAN [18]			
Normalization	BN	AdaBN	AutoDIAL	**TransNorm**	BN	AdaBN	AutoDIAL	**TransNorm**
A→W	82.0	82.4	84.8	**91.8**	94.1	88.8	92.3	**95.7**
D→W	96.9	**97.7**	**97.7**	**97.7**	98.6	98.6	98.6	**98.7**
W→D	99.1	99.8	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
A→D	79.7	81.0	85.7	**88.0**	92.9	92.7	93.0	**94.0**
D→A	**68.2**	67.2	63.9	**68.2**	71.5	70.8	71.5	**73.4**
W→A	67.4	68.2	68.7	**70.4**	69.3	70.0	72.2	**74.2**
Avg	82.2	82.7	83.5	**86.0**	87.7	86.8	87.9	**89.3**

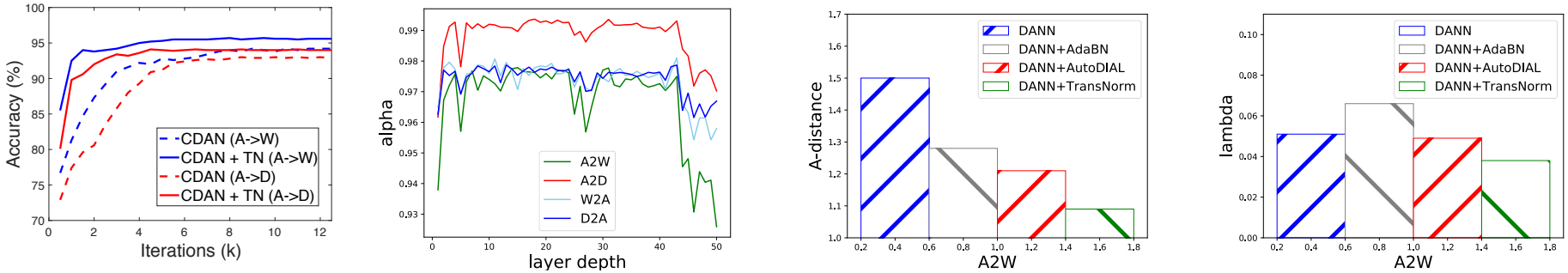
Table 6: Ablation study (%) on Office-31 with CDAN+TransNorm (ResNet-50).

Type	Weight	Distance Type			Probability Type		
Ablation	$\alpha = 1$	μ	Wasserstein Distance	$\mu/\sqrt{\sigma^2+\epsilon}$	Softmax	Gaussian	Student-t
A→W	94.6	95.0	94.5	**95.7**	94.9	94.8	**95.7**
D→W	98.6	**98.7**	**98.7**	**98.7**	97.9	98.6	**98.7**
W→D	**100.0**	**100.0**	**100.0**	**100.0**	99.8	**100.0**	**100.0**
A→D	93.4	93.0	91.0	**94.0**	91.4	93.1	**94.0**
D→A	71.5	72.8	72.6	**73.4**	69.7	72.4	**73.4**
W→A	72.9	73.7	73.6	**74.2**	73.2	73.0	**74.2**
Avg	88.5	88.9	88.4	**89.3**	87.9	88.7	**89.3**

Theoretical Analysis

Ben-David *et al.* $\mathcal{E}_{\mathcal{T}}(h) \leq \mathcal{E}_{\mathcal{S}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) + \lambda$

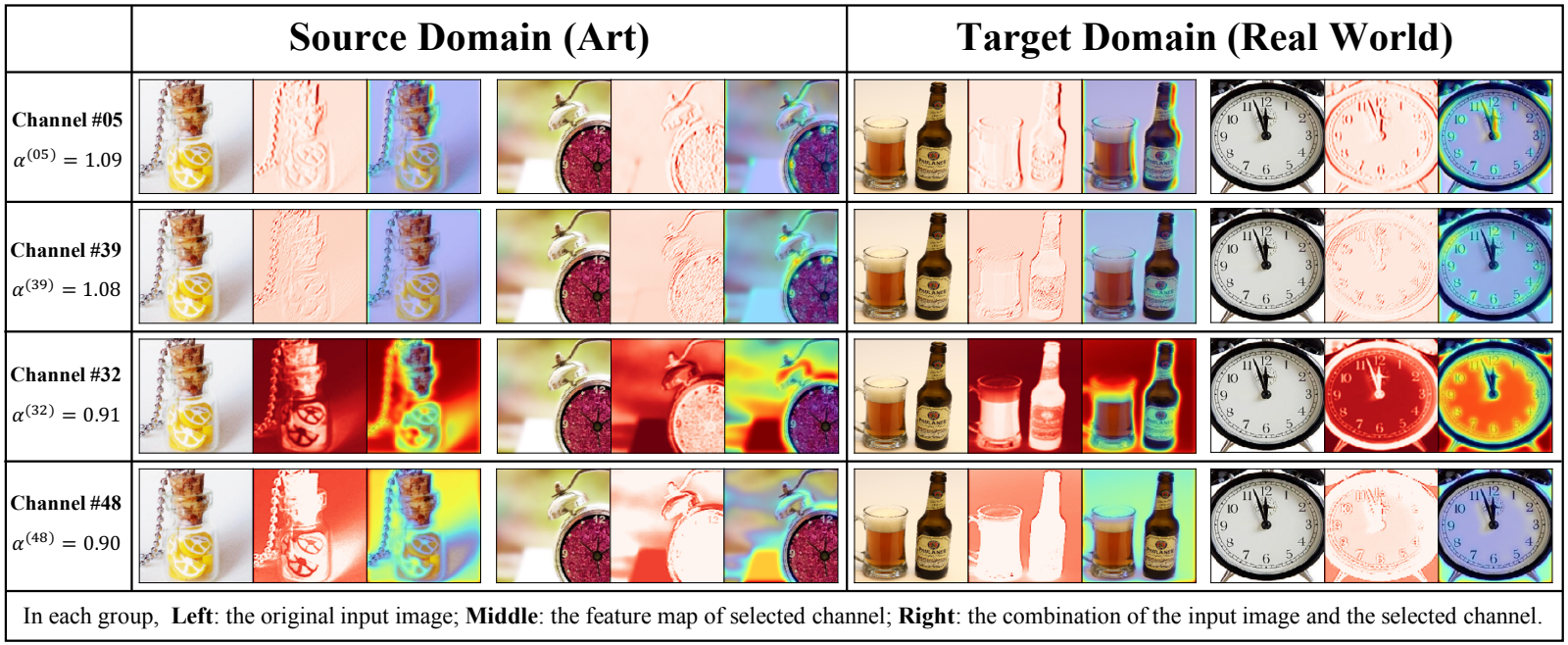
- (a) The expected error of h on the source domain, $\mathcal{E}_{\mathcal{S}}(h)$;
- (b) A-distance $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) = 2(1 - 2\epsilon)$, a measure of domain discrepancy, where ϵ is the error rate of a domain classifier trained to discriminate source domain and target domain;
- (c) The error λ of the ideal joint hypothesis h^* on both domains.

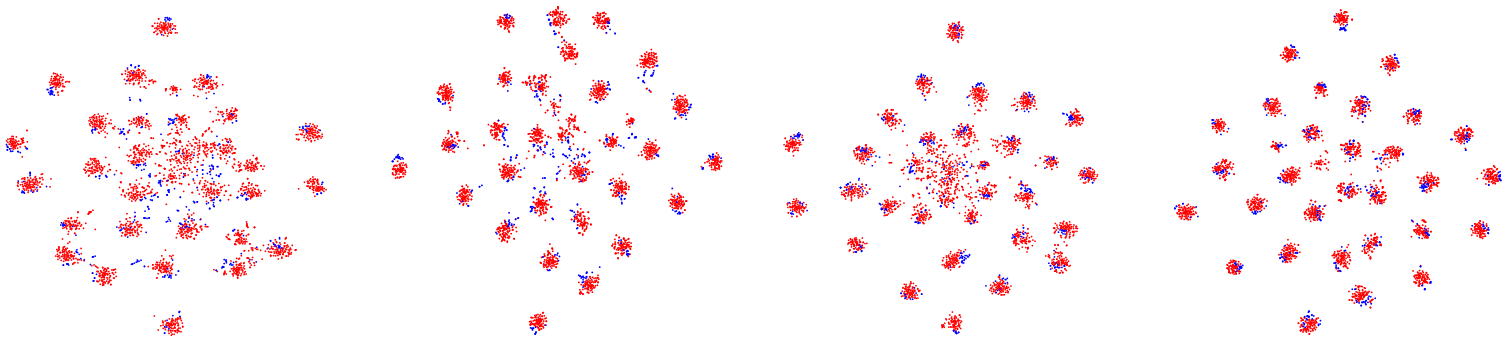


(a) Convergence (b) α in AutoDIAL (c) A-distance (d) lambda

Figure 3: Visualization of convergence speed, α in AutoDIAL [22], A-distance and λ.

Channel Visualization and Feature Visualization





(a) DANN (BN) (b) DANN (TransNorm) (c) CDAN (BN) (d) CDAN (TransNorm)

Figure 5: Visualization of features from the models with BN and TransNorm. (red: **A**; blue: **D**).

Summary

- TransNorm: a simple, general and end-to-end trainable normalization layer towards improving transferability of DNNs.
- TransNorm can be easily plugged into various DNNs and DA methods by simply replacing their internal BN layers with our TransNorm layers.
- As a general method, TransNorm does not introduce any extra learnable parameters or hyper-parameters .
- TransNorm improves classification accuracies and accelerates convergence for mainstream DNN-based DA methods.
- Code available @ `http://github.com/thuml/TransNorm`