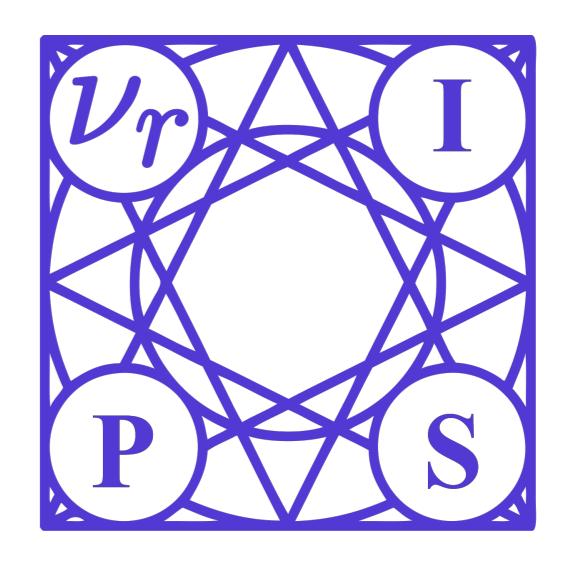




Transferable Normalization: Towards Improving Transferability of Deep Neural Networks

Ximei Wang[†], Ying Jin[†], Mingsheng Long(\star)[†], Jianmin Wang[†], and Michael I. Jordan[#]



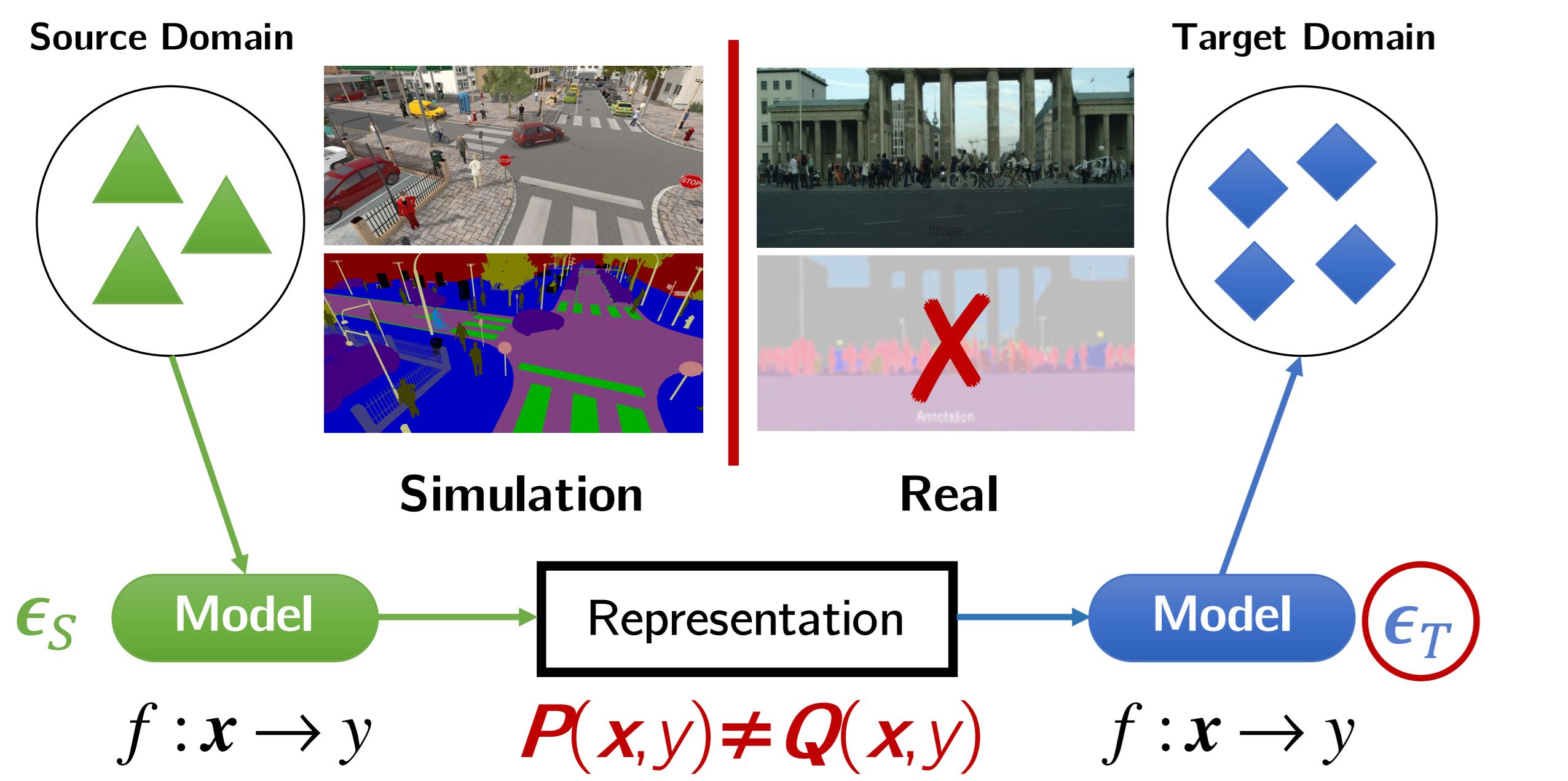
[†] School of Software, BNRIst, NEL-BDS, Tsinghua University, China, [#]University of California, Berkeley, USA

Summary

- TransNorm: a simple, general and end-to-end trainable normalization layer towards improving transferability of Deep Neural Networks (DNNs).
- TransNorm can be easily plugged into various DNNs and DA methods, without introducing any extra learnable parameters or hyper-parameters.
- Improves classification accuracies and accelerates convergence for mainstream DNN-based DA methods.
- Code available @ github.com/thuml/TransNorm

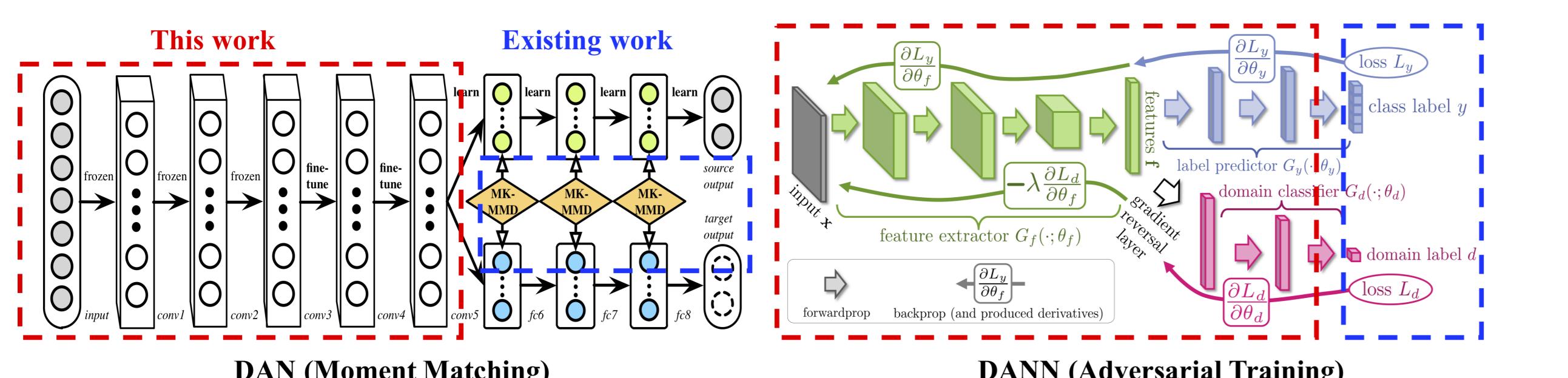
Domain Adaptation (DA)

- Deep learning across domains: ($P \neq Q$)
 - Independent and Differently Distributed (IDD) distributions



Mainstream Approaches to Domain Adaptation

- Existing works tackle domain adaptation by reducing the domain shift from the perspective of loss function designs.
- Rare attention has been paid to the other side of the coin, i.e. network design, an indispensable part of DA.



Batch Normalization (BN)

$$\mu^{(j)} = \frac{1}{m} \sum_{i=1}^m x_i^{(j)}, \quad \sigma^{2(j)} = \frac{1}{m} \sum_{i=1}^m (x_i^{(j)} - \mu^{(j)})^2 \quad (1)$$

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

Transferable Normalization (TransNorm)

Domain Specific Mean and Variance

$$\hat{x}_s = \frac{x_s - u_s}{\sqrt{\sigma_s^2 + \epsilon}}, \quad \hat{x}_t = \frac{x_t - u_t}{\sqrt{\sigma_t^2 + \epsilon}} \quad (3)$$

Domain Specific Mean and Variance

$$z_s = \gamma \hat{x}_s + \beta, \quad z_t = \gamma \hat{x}_t + \beta, \quad (4)$$

Domain Adaptive Alpha

Calculating Distance

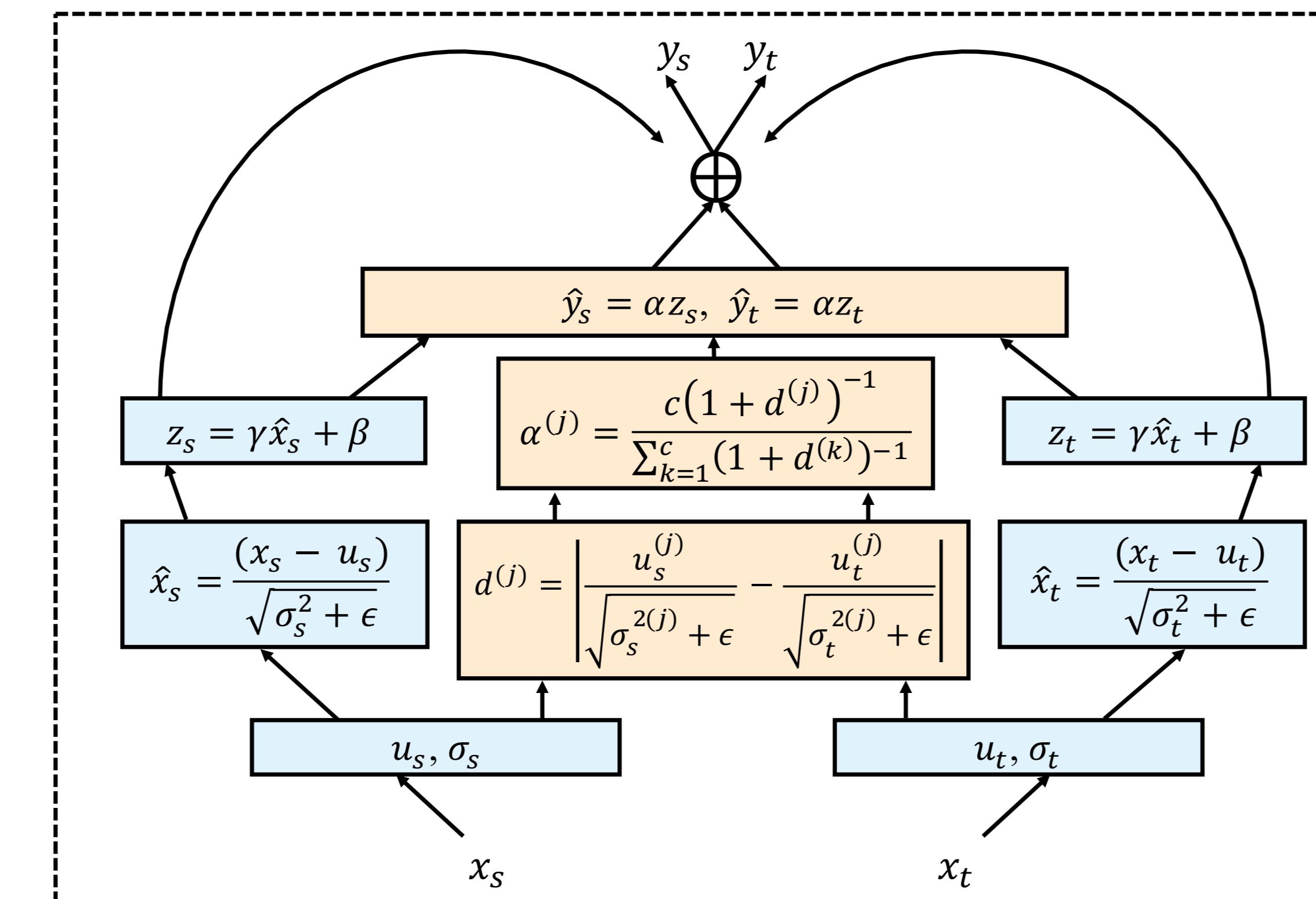
$$d^{(j)} = \left| \frac{\mu_s^{(j)}}{\sqrt{\sigma_s^2 + \epsilon}} - \frac{\mu_t^{(j)}}{\sqrt{\sigma_t^2 + \epsilon}} \right|, \quad j = 1, 2, \dots, c, \quad (5)$$

Generating Probability

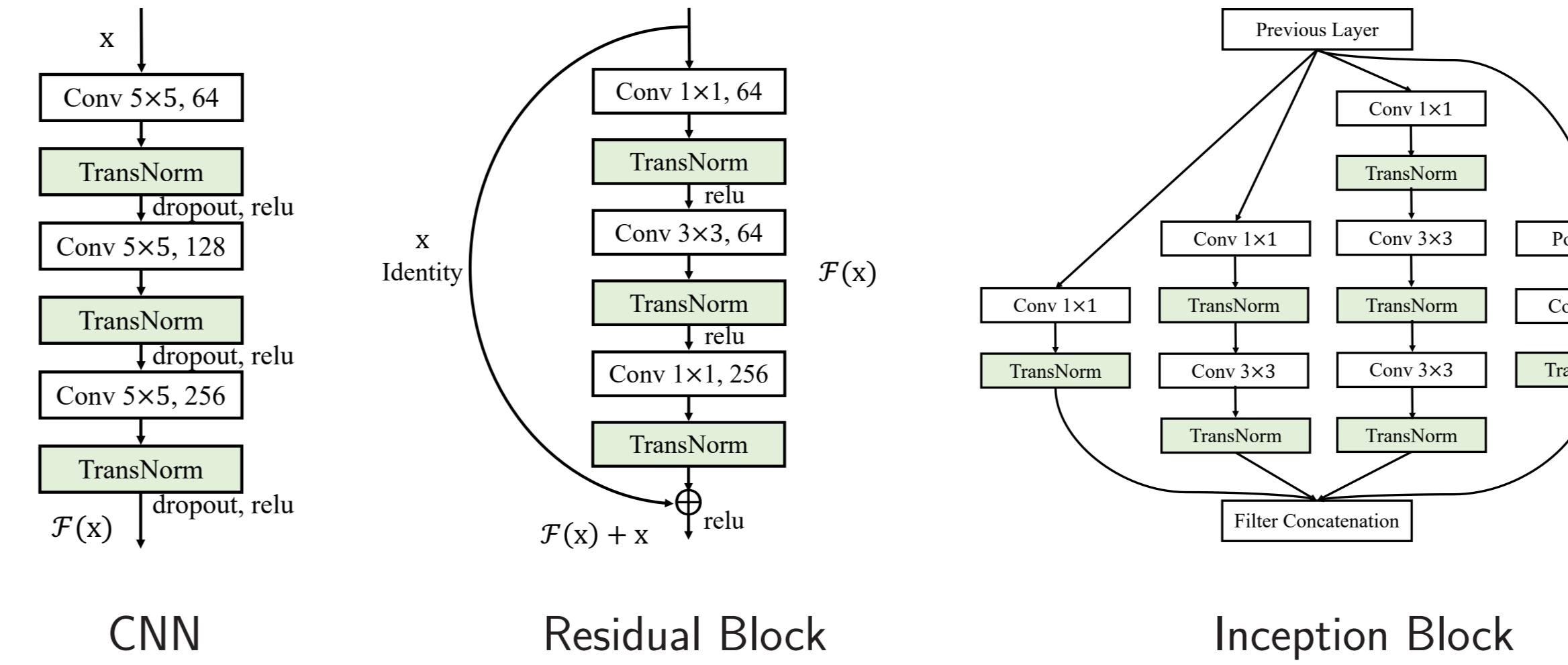
$$\alpha^{(j)} = \frac{c(1 + d^{(j)})^{-1}}{\sum_{k=1}^c (1 + d^{(k)})^{-1}}, \quad j = 1, 2, \dots, c, \quad (6)$$

Residual Connection

$$y_s = (1 + \alpha)z_s, \quad y_t = (1 + \alpha)z_t. \quad (7)$$



TransNorm Layer

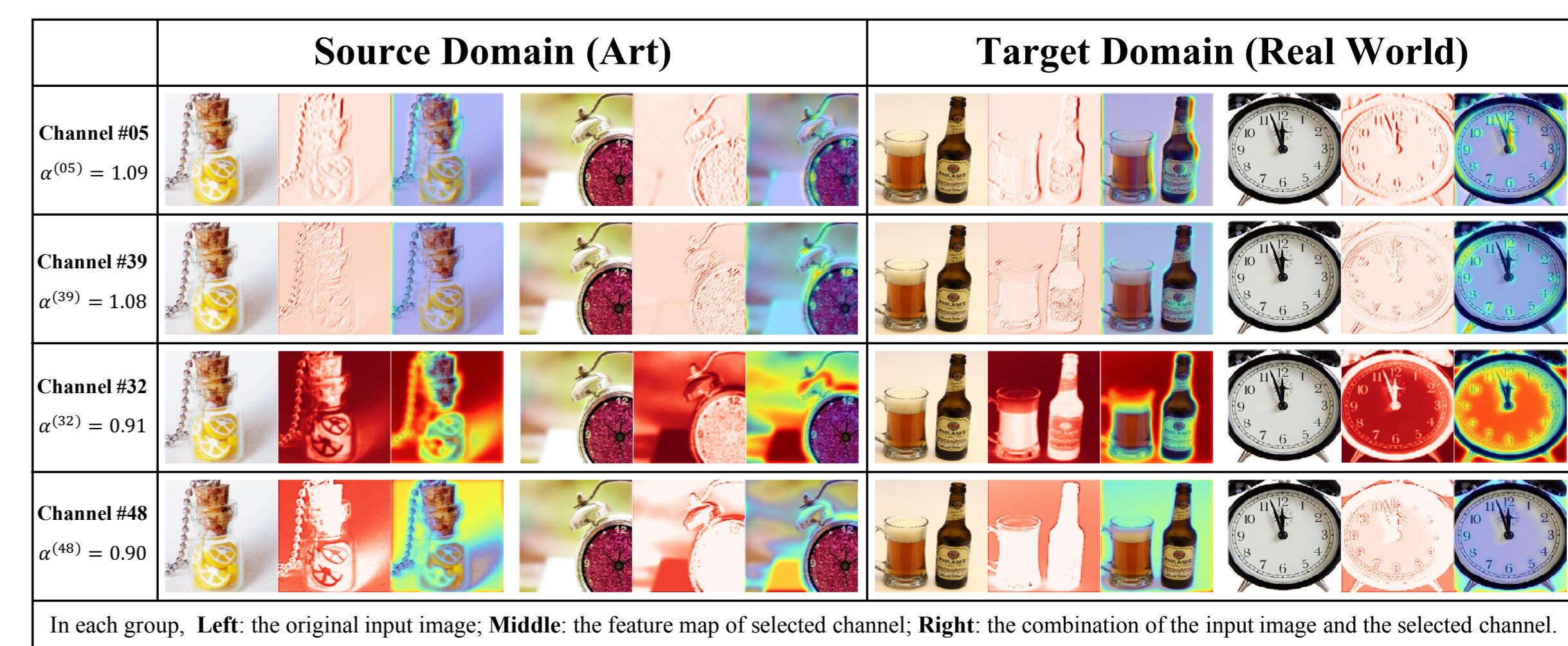


Comparison and Ablation Study

Method	DANN			CDAN				
	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

Type	Weight	Distance Type		Probability Type			
		μ	Wasserstein Distance	$\mu/\sqrt{\sigma^2 + \epsilon}$	Softmax	Gaussian	
Ablation	$\alpha = 1$	μ	Wasserstein Distance	$\mu/\sqrt{\sigma^2 + \epsilon}$	Softmax	Gaussian	
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

Channel Visualization



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 source and target domains.

