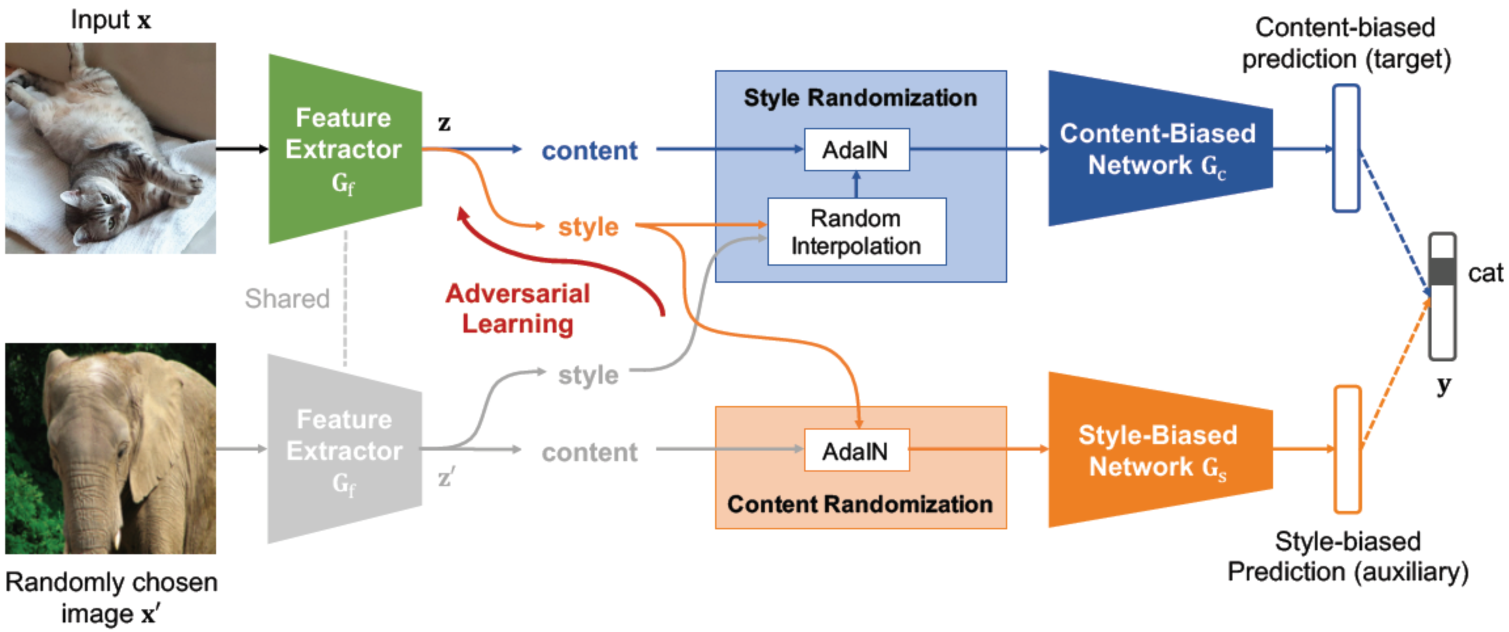
Meta Batch-Instance Normalization for Generalizable Person Re-Identification

Choi et al., CVPR 2021.

Presenter: Yoonki Cho

Recap: Reducing Domain Gap by Reducing Style Bias (CVPR 21, oral)

- Style-Agnostic Network (SagNet)
  - Goal: Content-biased network → Robust under domain shifts.
  - How? → Randomly swap style codes between images.



SGVR Lab

Nam et al. Reducing Domain Gap by Reducing Style Bias. In CVPR 2021.

Contents

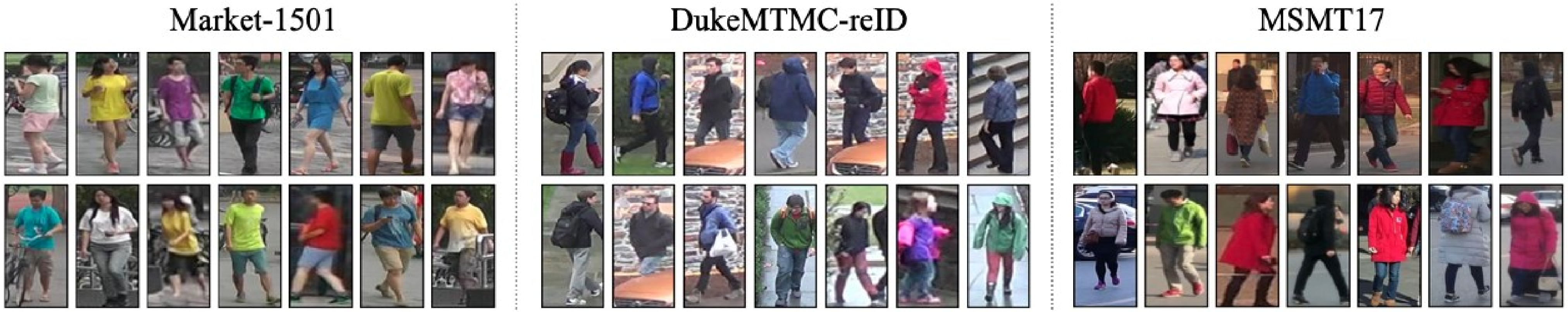
1. Introduction
2. Backgrounds
3. Method
4. Experiments
5. Quiz

Introduction

Motivation & Research Goal

Generalizable Person Re-identification

- Generalizable person re-ID is a “domain generalization (DG)” problem for person retrieval.
  - Training (Source) domain ≠ Testing (Target) domain.
- Style difference between domains makes a domain gap.
  - Season (Weather), Viewpoint, Clothing, etc.



SGVR Lab

Zheng et al. Joint Discriminative and Generative Learning for Person Re-identification. In CVPR 2019.

Generalizable Person Re-identification

- Many works employ an **instance normalization (IN)** to reduce style variations.
  - However, it often loses discriminative information.
  - Also, it requires a lot of trial and error.

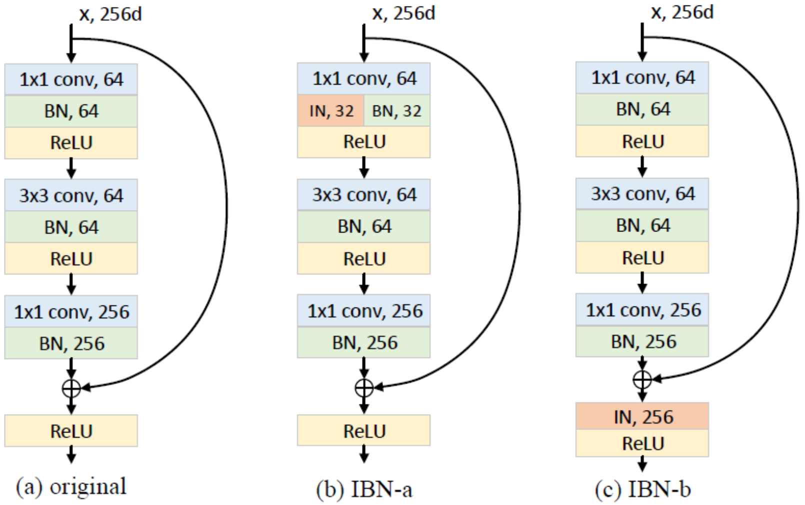
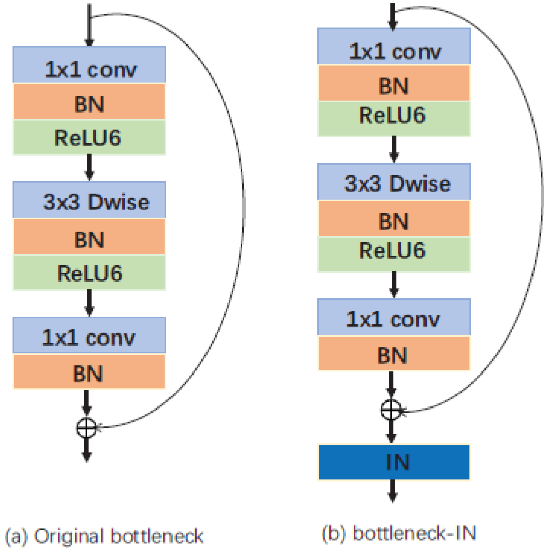


Fig. 3. Instance-batch normalization (IBN) block.

IBN-Net, ECCV 18



DualNorm, BMVC 19

Pan et al. Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net. In ECCV 2018.
Jia et al. Frustratingly Easy Person Re-Identification: Generalizing Person Re-ID in Practice. In BMVC 2019.

Research Goal

- Goal: Solve a generalizable person re-ID using style normalization
  - Preserving discriminative information.
  - Without a trial and error manner.

- To achieve the goal, the proposed method
  - Mimics unsuccessful generalization scenarios in a meta-learning manner.
  - Learn a generalization ability from unsuccessful generalization episodes.

  - Utilizes both IN and BN → learnable batch-instance normalization (BIN).

Research Goal

- Goal: Solve a generalizable person re-ID using style normalization
  - Preserving discriminative information.
  - Without a trial and error manner.

- To achieve the goal, the proposed method
  - Mimics unsuccessful generalization scenarios in a meta-learning manner.
  - Learn a generalization ability from unsuccessful generalization episodes.

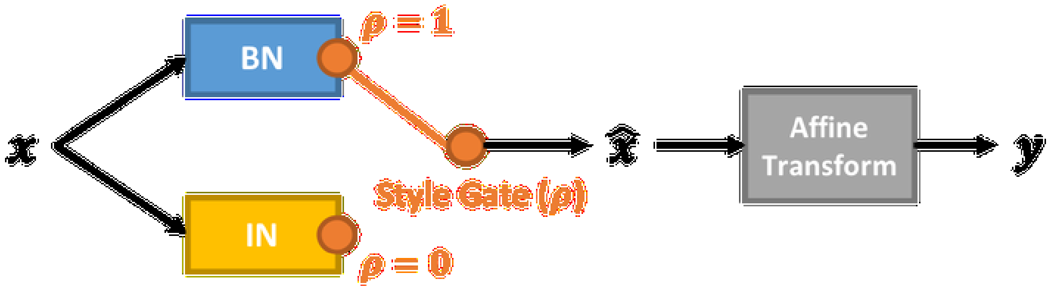
  - Utilizes both IN and BN → learnable batch-instance normalization (BIN).

Backgrounds

Batch-Instance Normalization (BIN)

Batch-Instance Normalization (BIN)

- Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks. In NeurIPS 2018.
  - BIN learns to selectively normalize a disturbing style while preserving an useful style.
  - The learnable parameter $\rho \in [0,1]^C$ controls how much to normalize style for each channel



$$\mathbf{y} = \left(\rho \cdot \hat{\mathbf{x}}^{(B)} + (1-\rho) \cdot \hat{\mathbf{x}}^{(I)}\right) \cdot \gamma + \beta$$

Batch-normalized feature map

Style-normalized feature map

Image credit: © Lunit Tech Blog. https://blog.lunit.io/2018/05/25/batch-instance-normalization/

Method

Meta Batch-Instance Normalization (MetaBIN)

Meta Batch-Instance Normalization (MetaBIN)

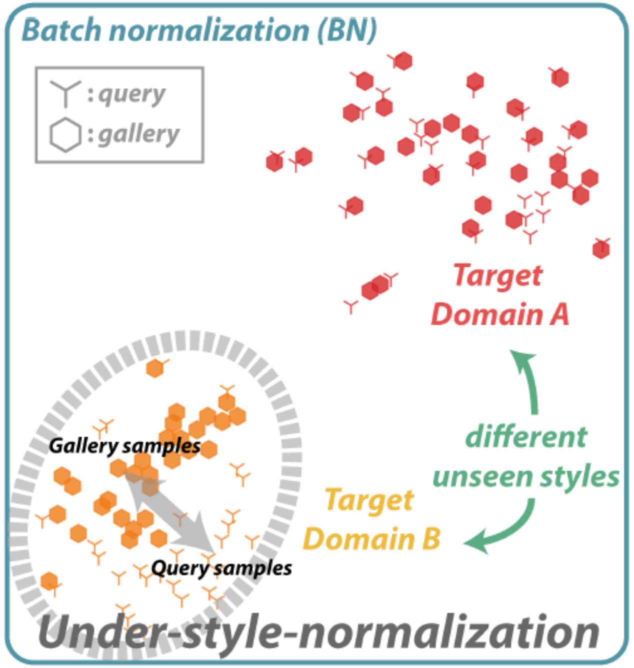
- The paper proposes a novel generalizable re-ID framework called MetaBIN that prevents overfitting to given source domain styles.

- The proposed method selectively normalize disturbing style by **unsuccessful generalization scenarios in a meta-learning manner.**

- To diversify the virtual simulations (i.e., unsuccessful generalization scenarios), the paper proposes **meta-train loss.**

Experimental Observation

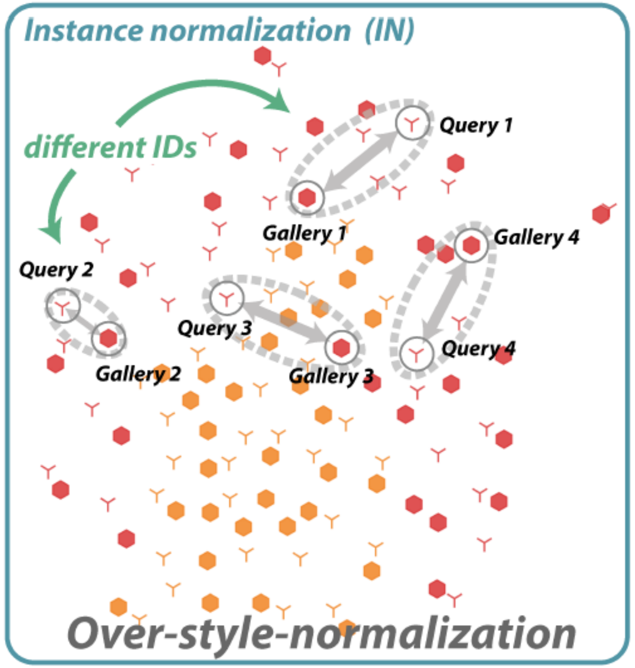
- Train a model with only BNs across multiple source domains.
- Under-style-normalization by BN.
  - When unexpected styles are given from unseen target domain, the model often fails to distinguish inputs' IDs.



SGVR Lab

Experimental Observation

- Train a model with only INs across multiple source domains.
- Over-style-normalization by IN.
  - It can remove unseen styles in the target domain.
  - However, it can also removes some discriminative information for re-identifying a person.



Meta Batch-Instance Normalization (MetaBIN)

- To address under- and over-style-normalization problems, the proposed method utilizes a batch-instance normalization (BIN).

$$\mathbf{y} = \rho\left(\gamma_B \cdot \underset{\substack{\uparrow \\ \text{Batch-normalized} \\ \text{feature map}}}{\hat{\mathbf{x}}_B} + \beta_B\right) + (1-\rho)\left(\gamma_I \cdot \underset{\substack{\uparrow \\ \text{Style-normalized} \\ \text{feature map}}}{\hat{\mathbf{x}}_I} + \beta_I\right)$$

- So how can we train a learnable balancing parameter $\rho \in [0,1]^C$?
  - If we directly train ρ with a end-to-end manner, it can easily overfit to the source domain's style.

MetaBIN Framework

- The proposed method trains a learnable balancing parameter ρ using a meta-learning pipeline.

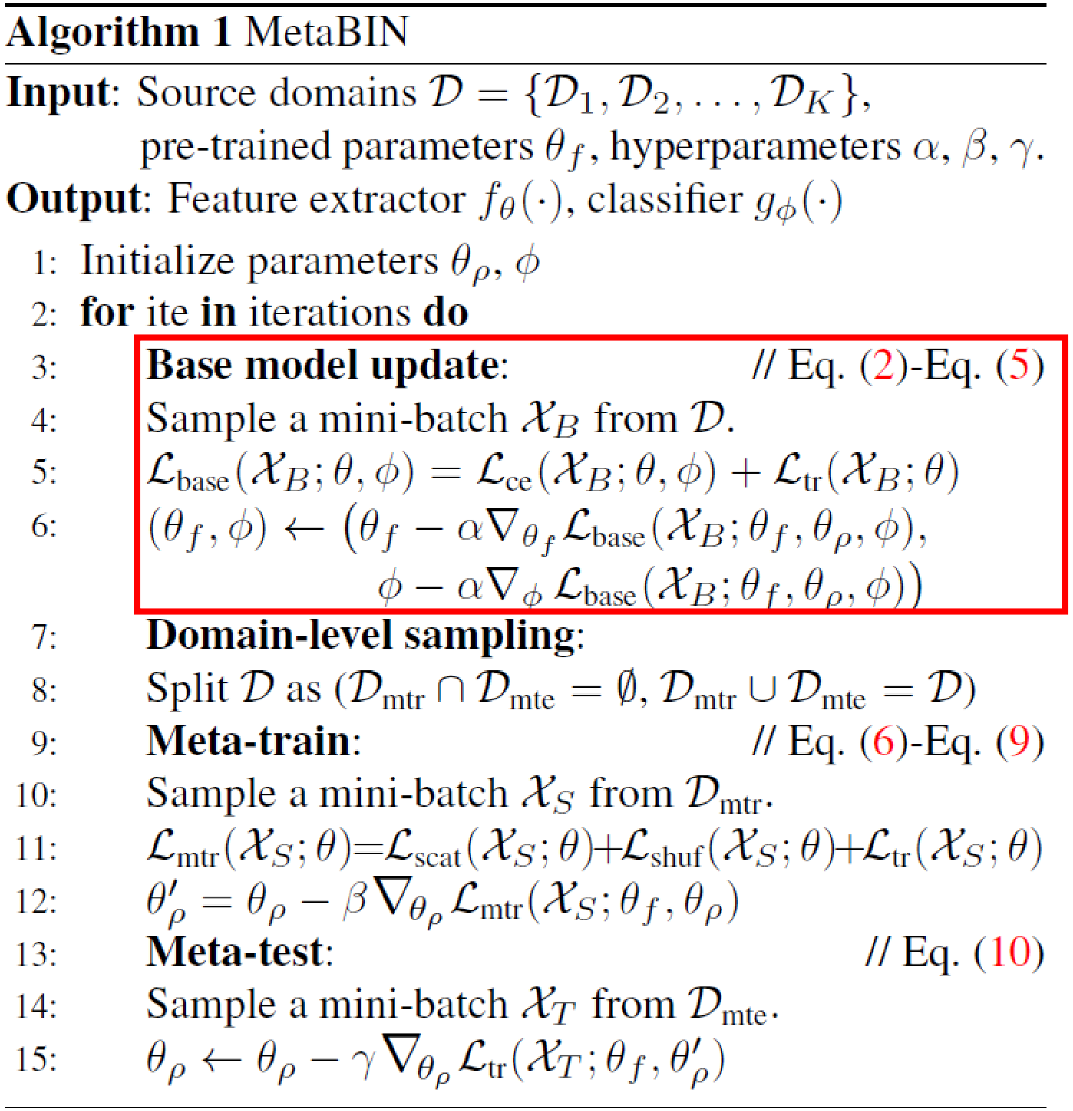
- Separate the training procedure to two episodes, then alternate both episodes.
  - Base model (Feature extractor) training.
  - Balancing parameter training.

- In the balancing parameter training, it mimics unsuccessful generalization scenarios.
  - Learn a generalization from generalization episodes.

MetaBIN Framework

- Base model update (Train a feature extractor and a classifier).
    - Utilizes cross-entropy loss and triplet loss.
    - Learn to re-identify a person.

$$\mathcal{L}_{\text{base}}(\mathcal{X}_B;\underset{\substack{\uparrow\\ \text{Feature extractor's}\\ \text{params}}}{\theta},\overset{\substack{\text{Classifier's}\\ \text{params}\\ \downarrow}}{\phi}) = \underbrace{\mathcal{L}_{\text{ce}}(\mathcal{X}_B;\theta,\phi)}_{\text{Cross-entropy loss}} + \underbrace{\mathcal{L}_{\text{tr}}(\mathcal{X}_B;\theta)}_{\text{Triplet loss}}.$$



Algorithm 1 MetaBIN

Input: Source domains $\mathcal{D}=\{\mathcal{D}_1,\mathcal{D}_2,\dots,\mathcal{D}_K\}$,
pre-trained parameters θ_f, hyperparameters α, β, γ.

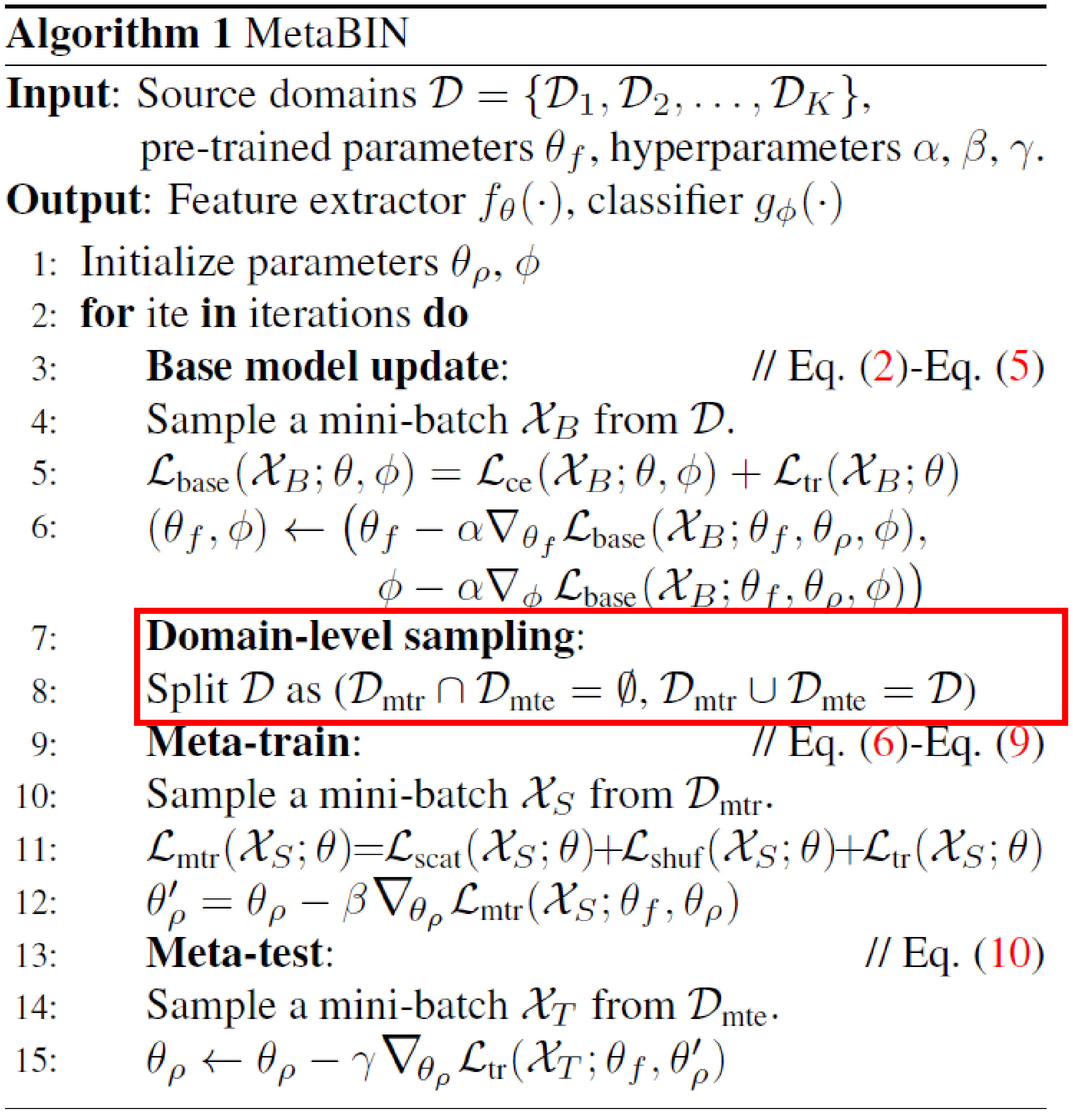
Output: Feature extractor $f_\theta(\cdot)$, classifier $g_\phi(\cdot)$

```
Initialize parameters $\theta_\rho$, $\phi$
for ite in iterations do
    Base model update:                                   // Eq. (2)-Eq. (5)
    Sample a mini-batch $\mathcal{X}_B$ from $\mathcal{D}$.
    $\mathcal{L}_{\text{base}}(\mathcal{X}_B;\theta,\phi) = \mathcal{L}_{\text{ce}}(\mathcal{X}_B;\theta,\phi) + \mathcal{L}_{\text{tr}}(\mathcal{X}_B;\theta)$
    $(\theta_f,\phi) \leftarrow (\theta_f - \alpha\nabla_{\theta_f}\mathcal{L}_{\text{base}}(\mathcal{X}_B;\theta_f,\theta_\rho,\phi),$
        $\phi - \alpha\nabla_\phi\mathcal{L}_{\text{base}}(\mathcal{X}_B;\theta_f,\theta_\rho,\phi))$
    Domain-level sampling:
    Split $\mathcal{D}$ as $(\mathcal{D}_{\text{mtr}} \cap \mathcal{D}_{\text{mte}} = \emptyset, \mathcal{D}_{\text{mtr}} \cup \mathcal{D}_{\text{mte}} = \mathcal{D})$
    Meta-train:                                          // Eq. (6)-Eq. (9)
    Sample a mini-batch $\mathcal{X}_S$ from $\mathcal{D}_{\text{mtr}}$.
    $\mathcal{L}_{\text{mtr}}(\mathcal{X}_S;\theta)=\mathcal{L}_{\text{scat}}(\mathcal{X}_S;\theta)+\mathcal{L}_{\text{shuf}}(\mathcal{X}_S;\theta)+\mathcal{L}_{\text{tr}}(\mathcal{X}_S;\theta)$
    $\theta'_\rho = \theta_\rho - \beta\nabla_{\theta_\rho}\mathcal{L}_{\text{mtr}}(\mathcal{X}_S;\theta_f,\theta_\rho)$
    Meta-test:                                           // Eq. (10)
    Sample a mini-batch $\mathcal{X}_T$ from $\mathcal{D}_{\text{mte}}$.
    $\theta_\rho \leftarrow \theta_\rho - \gamma\nabla_{\theta_\rho}\mathcal{L}_{\text{tr}}(\mathcal{X}_T;\theta_f,\theta'_\rho)$
```

MetaBIN Framework

- Domain-level sampling
- Split the given domains to
  - meta-train domains $\mathcal{D}_{mtr}$.
  - meta-test domains $\mathcal{D}_{mte}$.

- Now, we can mimic a domain generalization scenario!
  - Train on $\mathcal{D}_{mtr}$, then generalize to $\mathcal{D}_{mte}$.



Algorithm 1 MetaBIN

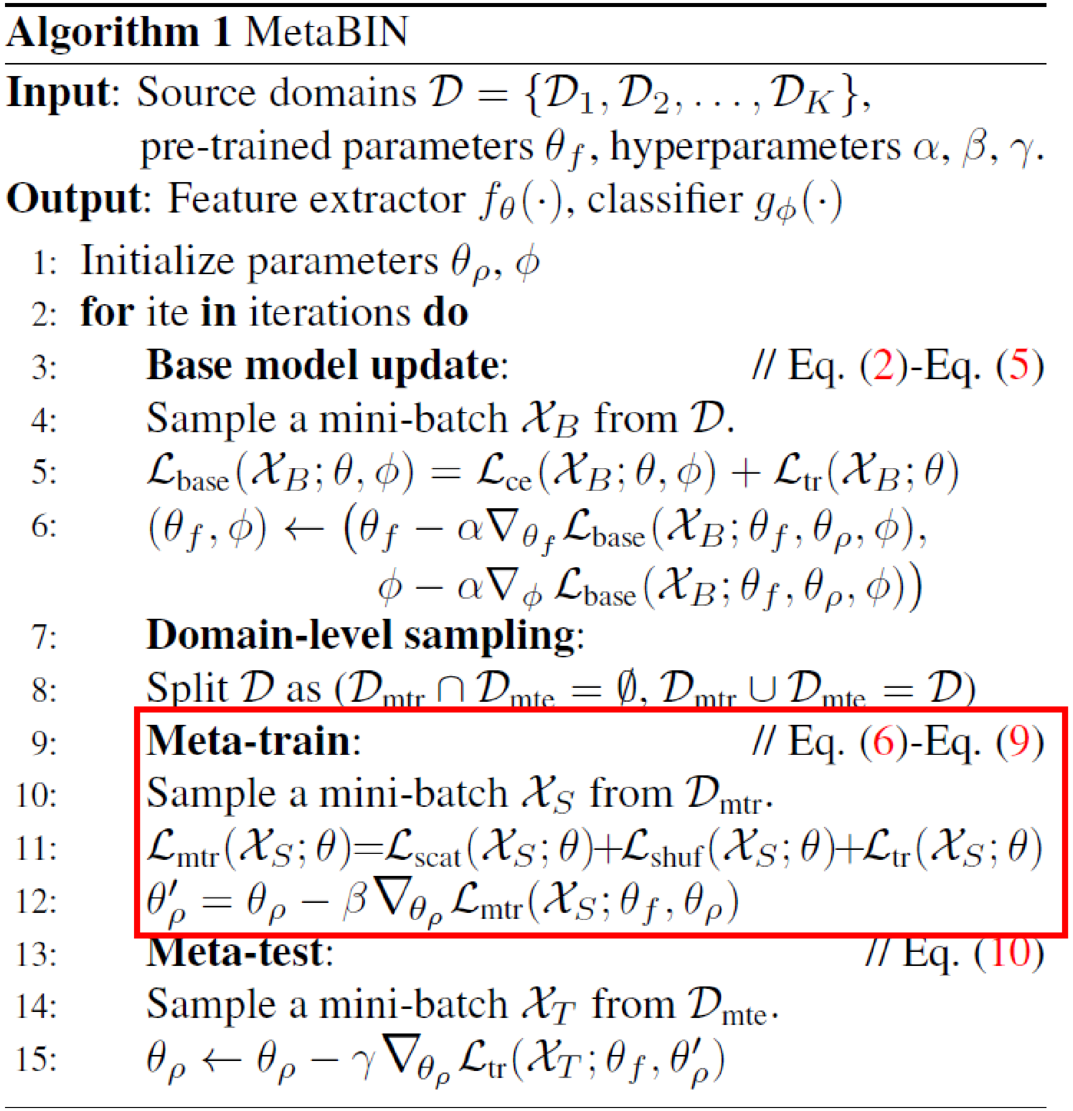
```
Input: Source domains $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K\}$,
       pre-trained parameters $\theta_f$, hyperparameters $\alpha$, $\beta$, $\gamma$.
Output: Feature extractor $f_\theta(\cdot)$, classifier $g_\phi(\cdot)$
Initialize parameters $\theta_\rho$, $\phi$
for ite in iterations do
    Base model update:                                  // Eq. (2)-Eq. (5)
    Sample a mini-batch $\mathcal{X}_B$ from $\mathcal{D}$.
    $\mathcal{L}_{\text{base}}(\mathcal{X}_B; \theta, \phi) = \mathcal{L}_{\text{ce}}(\mathcal{X}_B; \theta, \phi) + \mathcal{L}_{\text{tr}}(\mathcal{X}_B; \theta)$
    $(\theta_f, \phi) \leftarrow (\theta_f - \alpha \nabla_{\theta_f} \mathcal{L}_{\text{base}}(\mathcal{X}_B; \theta_f, \theta_\rho, \phi),$
                   $\phi - \alpha \nabla_\phi \mathcal{L}_{\text{base}}(\mathcal{X}_B; \theta_f, \theta_\rho, \phi))$
    Domain-level sampling:
    Split $\mathcal{D}$ as $(\mathcal{D}_{\text{mtr}} \cap \mathcal{D}_{\text{mte}} = \emptyset, \mathcal{D}_{\text{mtr}} \cup \mathcal{D}_{\text{mte}} = \mathcal{D})$
    Meta-train:                                         // Eq. (6)-Eq. (9)
    Sample a mini-batch $\mathcal{X}_S$ from $\mathcal{D}_{\text{mtr}}$.
    $\mathcal{L}_{\text{mtr}}(\mathcal{X}_S; \theta) = \mathcal{L}_{\text{scat}}(\mathcal{X}_S; \theta) + \mathcal{L}_{\text{shuf}}(\mathcal{X}_S; \theta) + \mathcal{L}_{\text{tr}}(\mathcal{X}_S; \theta)$
    $\theta'_\rho = \theta_\rho - \beta \nabla_{\theta_\rho} \mathcal{L}_{\text{mtr}}(\mathcal{X}_S; \theta_f, \theta_\rho)$
    Meta-test:                                          // Eq. (10)
    Sample a mini-batch $\mathcal{X}_T$ from $\mathcal{D}_{\text{mte}}$.
    $\theta_\rho \leftarrow \theta_\rho - \gamma \nabla_{\theta_\rho} \mathcal{L}_{\text{tr}}(\mathcal{X}_T; \theta_f, \theta'_\rho)$
```

SGVR Lab

MetaBIN Framework

- Domain-level sampling
- Split the given domains to
  - meta-train domains $\mathcal{D}_{mtr}$.
  - meta-test domains $\mathcal{D}_{mte}$.

- Now, we can mimic a domain generalization scenario!
  - *Train on $\mathcal{D}_{mtr}$,* then generalize to $\mathcal{D}_{mte}$.



Algorithm 1 MetaBIN

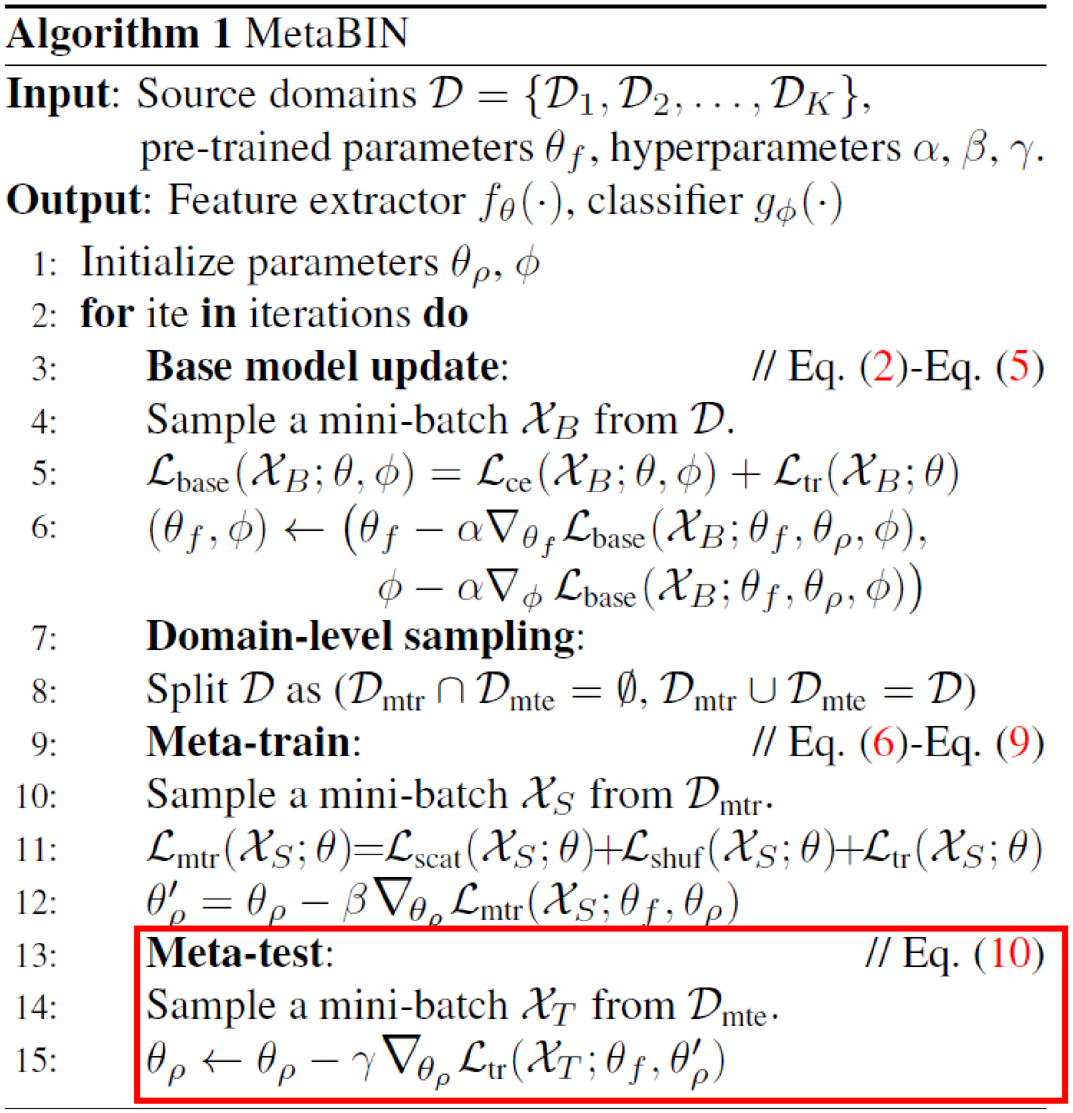
Input: Source domains $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$,
pre-trained parameters θ_f, hyperparameters α, β, γ.

Output: Feature extractor $f_\theta(\cdot)$, classifier $g_\phi(\cdot)$

```
Initialize parameters $\theta_\rho$, $\phi$
for ite in iterations do
    Base model update:                                   // Eq. (2)-Eq. (5)
    Sample a mini-batch $\mathcal{X}_B$ from $\mathcal{D}$.
    $\mathcal{L}_{\text{base}}(\mathcal{X}_B;\theta,\phi) = \mathcal{L}_{\text{ce}}(\mathcal{X}_B;\theta,\phi) + \mathcal{L}_{\text{tr}}(\mathcal{X}_B;\theta)$
    $(\theta_f,\phi) \leftarrow (\theta_f - \alpha\nabla_{\theta_f}\mathcal{L}_{\text{base}}(\mathcal{X}_B;\theta_f,\theta_\rho,\phi),$
                  $\phi - \alpha\nabla_\phi\mathcal{L}_{\text{base}}(\mathcal{X}_B;\theta_f,\theta_\rho,\phi))$
    Domain-level sampling:
    Split $\mathcal{D}$ as $(\mathcal{D}_{\text{mtr}} \cap \mathcal{D}_{\text{mte}} = \emptyset, \mathcal{D}_{\text{mtr}} \cup \mathcal{D}_{\text{mte}} = \mathcal{D})$
    Meta-train:                                          // Eq. (6)-Eq. (9)
    Sample a mini-batch $\mathcal{X}_S$ from $\mathcal{D}_{\text{mtr}}$.
    $\mathcal{L}_{\text{mtr}}(\mathcal{X}_S;\theta)=\mathcal{L}_{\text{scat}}(\mathcal{X}_S;\theta)+\mathcal{L}_{\text{shuf}}(\mathcal{X}_S;\theta)+\mathcal{L}_{\text{tr}}(\mathcal{X}_S;\theta)$
    $\theta'_\rho = \theta_\rho - \beta\nabla_{\theta_\rho}\mathcal{L}_{\text{mtr}}(\mathcal{X}_S;\theta_f,\theta_\rho)$
    Meta-test:                                           // Eq. (10)
    Sample a mini-batch $\mathcal{X}_T$ from $\mathcal{D}_{\text{mte}}$.
    $\theta_\rho \leftarrow \theta_\rho - \gamma\nabla_{\theta_\rho}\mathcal{L}_{\text{tr}}(\mathcal{X}_T;\theta_f,\theta'_\rho)$
```

MetaBIN Framework

- Domain-level sampling
- Split the given domains to
  - meta-train domains $\mathcal{D}_{mtr}$.
  - meta-test domains $\mathcal{D}_{mte}$.
- Now, we can mimic a domain generalization scenario!
  - Train on $\mathcal{D}_{mtr}$, ***then generalize to $\mathcal{D}_{mte}$.***



Algorithm 1 MetaBIN

```
Input: Source domains $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$,
       pre-trained parameters $\theta_f$, hyperparameters $\alpha$, $\beta$, $\gamma$.
Output: Feature extractor $f_\theta(\cdot)$, classifier $g_\phi(\cdot)$
Initialize parameters $\theta_\rho$, $\phi$
for ite in iterations do
    Base model update:                                   // Eq. (2)-Eq. (5)
    Sample a mini-batch $\mathcal{X}_B$ from $\mathcal{D}$.
    $\mathcal{L}_{\text{base}}(\mathcal{X}_B;\theta,\phi) = \mathcal{L}_{\text{ce}}(\mathcal{X}_B;\theta,\phi) + \mathcal{L}_{\text{tr}}(\mathcal{X}_B;\theta)$
    $(\theta_f,\phi) \leftarrow (\theta_f - \alpha\nabla_{\theta_f}\mathcal{L}_{\text{base}}(\mathcal{X}_B;\theta_f,\theta_\rho,\phi),$
        $\phi - \alpha\nabla_{\phi}\mathcal{L}_{\text{base}}(\mathcal{X}_B;\theta_f,\theta_\rho,\phi))$
    Domain-level sampling:
    Split $\mathcal{D}$ as $(\mathcal{D}_{\text{mtr}} \cap \mathcal{D}_{\text{mte}} = \emptyset, \mathcal{D}_{\text{mtr}} \cup \mathcal{D}_{\text{mte}} = \mathcal{D})$
    Meta-train:                                          // Eq. (6)-Eq. (9)
    Sample a mini-batch $\mathcal{X}_S$ from $\mathcal{D}_{\text{mtr}}$.
    $\mathcal{L}_{\text{mtr}}(\mathcal{X}_S;\theta)=\mathcal{L}_{\text{scat}}(\mathcal{X}_S;\theta)+\mathcal{L}_{\text{shuf}}(\mathcal{X}_S;\theta)+\mathcal{L}_{\text{tr}}(\mathcal{X}_S;\theta)$
    $\theta'_\rho = \theta_\rho - \beta\nabla_{\theta_\rho}\mathcal{L}_{\text{mtr}}(\mathcal{X}_S;\theta_f,\theta_\rho)$
    Meta-test:                                           // Eq. (10)
    Sample a mini-batch $\mathcal{X}_T$ from $\mathcal{D}_{\text{mte}}$.
    $\theta_\rho \leftarrow \theta_\rho - \gamma\nabla_{\theta_\rho}\mathcal{L}_{\text{tr}}(\mathcal{X}_T;\theta_f,\theta'_\rho)$
```

Meta-train stage

- In this stage, MetaBIN **trains the balancing parameter ρ to mimic unsuccessful generalization scenarios.**
- Unsuccessful generalization scenarios = under- or over-style-normalization

- To make under-&over-style-normalization, the paper proposes the meta-train loss.

$$\mathcal{L}_{\mathrm{mtr}}(\mathcal{X}_S;\theta) = \underbrace{\mathcal{L}_{\mathrm{scat}}(\mathcal{X}_S;\theta)+\mathcal{L}_{\mathrm{shuf}}(\mathcal{X}_S;\theta)}_{\text{for over-style-normalization}}+\underbrace{\mathcal{L}_{\mathrm{tr}}(\mathcal{X}_S;\theta)}_{\text{for under-style-normalization}}.$$

K_S : number of domains in a mini-batch
N_S^k : number of samples in domain k
N_S : number of samples in a mini-batch

Meta-train stage (over-style-normalization)

- Intra-domain scatter loss: ***Each feature should be far from its own domain's centroid feature.***

$$\mathcal{L}_{\text{scat}}(\mathcal{X}_S;\theta) = \frac{1}{N_S}\sum_{k=1}^{K_S}\sum_{i=1}^{N_S^k} cos(\boldsymbol{f}_i^k, \bar{\boldsymbol{f}}^k)$$

mean feature vector
(centroid)

- Inter-domain shuffle loss: ***Inter-domain features should be closer than intra-domain features.***

$$\mathcal{L}_{\text{shuf}}(\mathcal{X}_S;\theta) = \frac{1}{N_S}\sum_{i=1}^{N_S} l_s\big(d(\boldsymbol{f}_i^a, \boldsymbol{f}_i^{n-}) - d(\boldsymbol{f}_i^a, \boldsymbol{f}_i^{n+})\big)$$

negative sample
from the inter-domain

negative sample
from the intra-domain

K_S : number of domains in a mini-batch
N_S^k : number of samples in domain k
N_S : number of samples in a mini-batch

Meta-train stage (over-style-normalization)

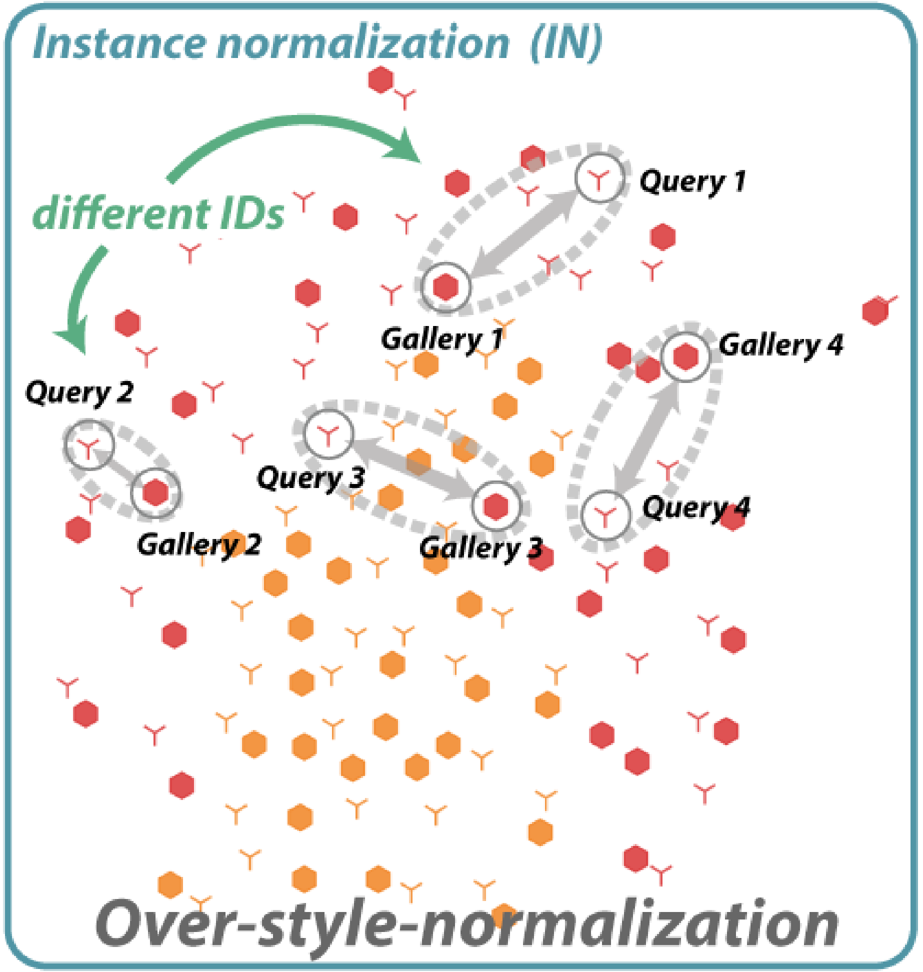
- Intra-domain scatter los *wn domain's centroid* ***feature.***

$\mathcal{L}_{\mathrm{sc}}$

ctor

- Inter-domain shuffle los ***er than intra-domain features.***

$\mathcal{L}_{\mathrm{shuf}}(\mathcal{X}_S$ $f_i^{n+}))$

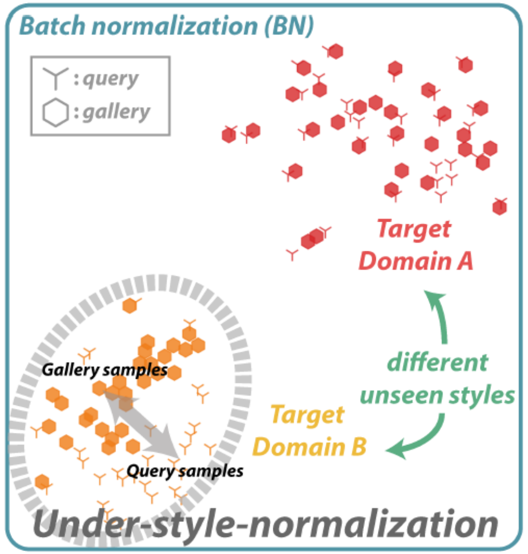


negative sample
om the intra-domain

SGVR Lab

Meta-train stage (under-style-normalization)

- Triplet loss for under-style-normalization
  - It leads to overfitting to the styles of the meta-train domains $\mathcal{D}_{mtr}$.



$$\mathcal{L}_{\text{mtr}}(\mathcal{X}_S;\theta) = \mathcal{L}_{\text{scat}}(\mathcal{X}_S;\theta) + \mathcal{L}_{\text{shuf}}(\mathcal{X}_S;\theta) + \mathcal{L}_{\text{tr}}(\mathcal{X}_S;\theta).$$

$$\theta'_\rho = \theta_\rho - \beta \nabla_{\theta_\rho} \mathcal{L}_{\text{mtr}}(\mathcal{X}_S;\theta_f,\theta_\rho)$$ Update the balancing parameter!

SGVR Lab

Meta-test stage

- In this stage, MetaBIN mimics a domain generalization scenario.
    - Evaluate the model with updated balancing parameter on meta-test domains $\mathcal{D}_{mte}$.
    - Employ the triplet loss on a mini-batch X_T from meta-test domains $\mathcal{D}_{mte}$.

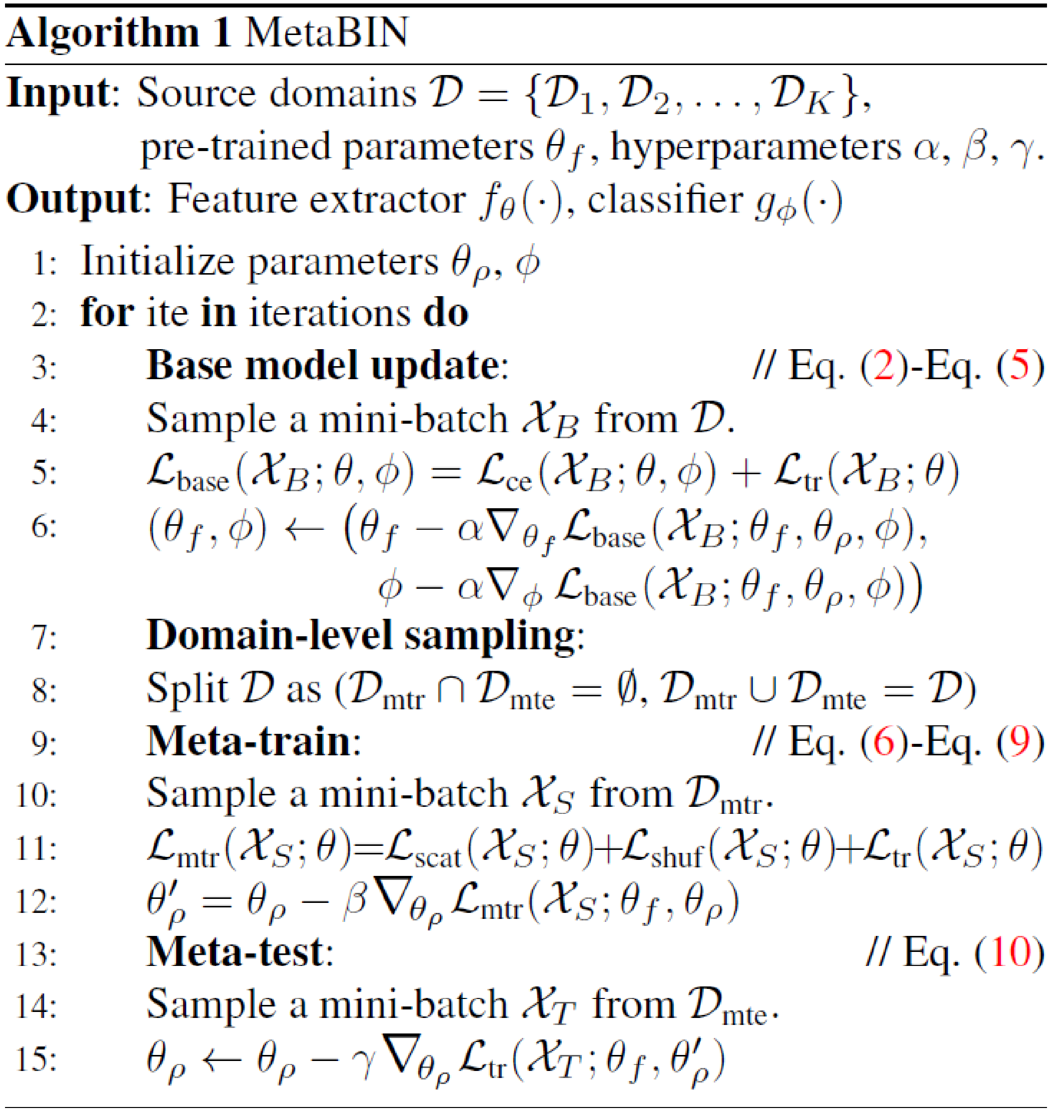
$$\theta_\rho \leftarrow \theta_\rho - \gamma \nabla_{\theta_\rho} \mathcal{L}_{\mathrm{tr}}(\mathcal{X}_T; \theta_f, \underset{\uparrow}{\theta'_\rho})$$

Updated parameter by meta-train stage

- Meta-update the balancing parameter to overcome the virtual simulations.
    - Learn a generalization from unsuccessful generalization scenarios!

Summary

- MetaBIN learns the balancing parameter of BIN in a meta-learning manner.
- It mimics unsuccessful generalization scenarios in a meta-learning manner.
- It proposes a meta-train loss to induce over-/under- style-normalization.
  - Meta-train loss collapses the balancing parameter.



Algorithm 1 MetaBIN

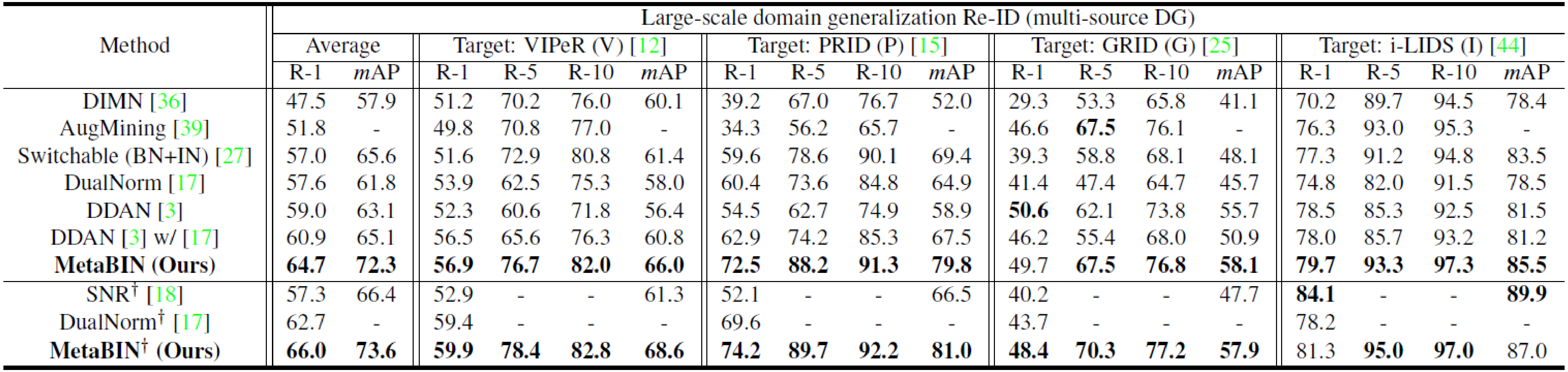
```
Input: Source domains $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K\}$,
       pre-trained parameters $\theta_f$, hyperparameters $\alpha$, $\beta$, $\gamma$.
Output: Feature extractor $f_\theta(\cdot)$, classifier $g_\phi(\cdot)$
Initialize parameters $\theta_\rho$, $\phi$
for ite in iterations do
    Base model update:                                          // Eq. (2)-Eq. (5)
    Sample a mini-batch $\mathcal{X}_B$ from $\mathcal{D}$.
    $\mathcal{L}_{\text{base}}(\mathcal{X}_B; \theta, \phi) = \mathcal{L}_{\text{ce}}(\mathcal{X}_B; \theta, \phi) + \mathcal{L}_{\text{tr}}(\mathcal{X}_B; \theta)$
    $(\theta_f, \phi) \leftarrow (\theta_f - \alpha \nabla_{\theta_f} \mathcal{L}_{\text{base}}(\mathcal{X}_B; \theta_f, \theta_\rho, \phi),$
                   $\phi - \alpha \nabla_\phi \mathcal{L}_{\text{base}}(\mathcal{X}_B; \theta_f, \theta_\rho, \phi))$
    Domain-level sampling:
    Split $\mathcal{D}$ as $(\mathcal{D}_{\text{mtr}} \cap \mathcal{D}_{\text{mte}} = \emptyset, \mathcal{D}_{\text{mtr}} \cup \mathcal{D}_{\text{mte}} = \mathcal{D})$
    Meta-train:                                                 // Eq. (6)-Eq. (9)
    Sample a mini-batch $\mathcal{X}_S$ from $\mathcal{D}_{\text{mtr}}$.
    $\mathcal{L}_{\text{mtr}}(\mathcal{X}_S; \theta) = \mathcal{L}_{\text{scat}}(\mathcal{X}_S; \theta) + \mathcal{L}_{\text{shuf}}(\mathcal{X}_S; \theta) + \mathcal{L}_{\text{tr}}(\mathcal{X}_S; \theta)$
    $\theta'_\rho = \theta_\rho - \beta \nabla_{\theta_\rho} \mathcal{L}_{\text{mtr}}(\mathcal{X}_S; \theta_f, \theta_\rho)$
    Meta-test:                                                  // Eq. (10)
    Sample a mini-batch $\mathcal{X}_T$ from $\mathcal{D}_{\text{mte}}$.
    $\theta_\rho \leftarrow \theta_\rho - \gamma \nabla_{\theta_\rho} \mathcal{L}_{\text{tr}}(\mathcal{X}_T; \theta_f, \theta'_\rho)$
```

Experiments

Comparison with SOTAs

- Multi-source domain generalization
  - Learn from multiple domains, then generalize to the other single domain.

Table 1. Performance (%) comparison with the state-of-the-arts on the large-scale DG Re-ID benchmark, where '†' is based on ResNet-50.



Method	Large-scale domain generalization Re-ID (multi-source DG)																	
	Average		Target: VIPeR (V) [12]				Target: PRID (P) [15]				Target: GRID (G) [25]				Target: i-LIDS (I) [44]			
	R-1	*m*AP	R-1	R-5	R-10	*m*AP	R-1	R-5	R-10	*m*AP	R-1	R-5	R-10	*m*AP	R-1	R-5	R-10	*m*AP
DIMN [36]	47.5	57.9	51.2	70.2	76.0	60.1	39.2	67.0	76.7	52.0	29.3	53.3	65.8	41.1	70.2	89.7	94.5	78.4
AugMining [39]	51.8	-	49.8	70.8	77.0	-	34.3	56.2	65.7	-	46.6	**67.5**	76.1	-	76.3	93.0	95.3	-
Switchable (BN+IN) [27]	57.0	65.6	51.6	72.9	80.8	61.4	59.6	78.6	90.1	69.4	39.3	58.8	68.1	48.1	77.3	91.2	94.8	83.5
DualNorm [17]	57.6	61.8	53.9	62.5	75.3	58.0	60.4	73.6	84.8	64.9	41.4	47.4	64.7	45.7	74.8	82.0	91.5	78.5
DDAN [3]	59.0	63.1	52.3	60.6	71.8	56.4	54.5	62.7	74.9	58.9	**50.6**	62.1	73.8	55.7	78.5	85.3	92.5	81.5
DDAN [3] w/ [17]	60.9	65.1	56.5	65.6	76.3	60.8	62.9	74.2	85.3	67.5	46.2	55.4	68.0	50.9	78.0	85.7	93.2	81.2
MetaBIN (Ours)	**64.7**	**72.3**	**56.9**	**76.7**	**82.0**	**66.0**	**72.5**	**88.2**	**91.3**	**79.8**	49.7	**67.5**	**76.8**	**58.1**	**79.7**	**93.3**	**97.3**	**85.5**
SNR† [18]	57.3	66.4	52.9	-	-	61.3	52.1	-	-	66.5	40.2	-	-	47.7	**84.1**	-	-	**89.9**
DualNorm† [17]	62.7	-	59.4	-	-	-	69.6	-	-	-	43.7	-	-	-	78.2	-	-	-
MetaBIN† (Ours)	**66.0**	**73.6**	**59.9**	**78.4**	**82.8**	**68.6**	**74.2**	**89.7**	**92.2**	**81.0**	**48.4**	**70.3**	**77.2**	**57.9**	81.3	**95.0**	**97.0**	87.0

Comparison with SOTAs

- Cross-domain generalization
  - Learn from single domains, then generalize to the other single domain.

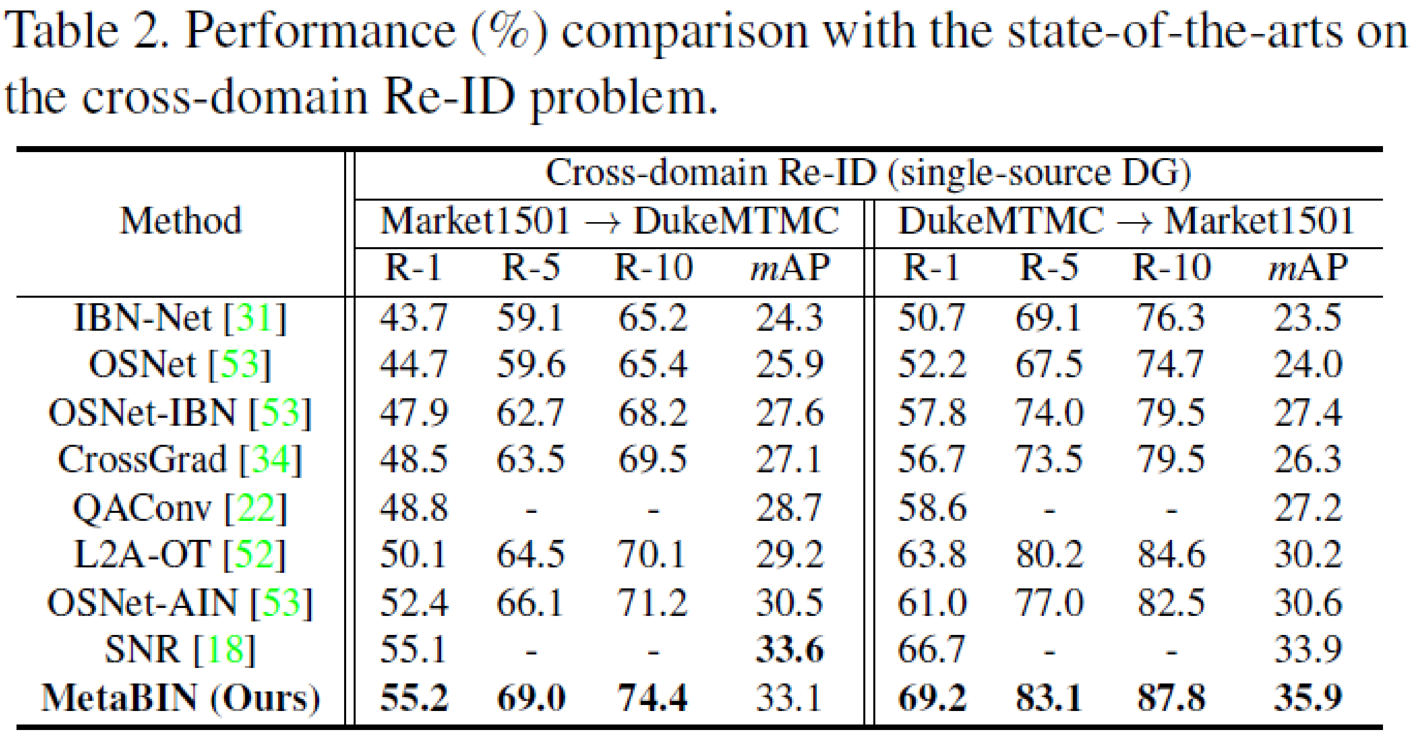


Table 2. Performance (%) comparison with the state-of-the-arts on the cross-domain Re-ID problem.

Method	Cross-domain Re-ID (single-source DG)							
	Market1501 → DukeMTMC				DukeMTMC → Market1501			
	R-1	R-5	R-10	*m*AP	R-1	R-5	R-10	*m*AP
IBN-Net [31]	43.7	59.1	65.2	24.3	50.7	69.1	76.3	23.5
OSNet [53]	44.7	59.6	65.4	25.9	52.2	67.5	74.7	24.0
OSNet-IBN [53]	47.9	62.7	68.2	27.6	57.8	74.0	79.5	27.4
CrossGrad [34]	48.5	63.5	69.5	27.1	56.7	73.5	79.5	26.3
QAConv [22]	48.8	-	-	28.7	58.6	-	-	27.2
L2A-OT [52]	50.1	64.5	70.1	29.2	63.8	80.2	84.6	30.2
OSNet-AIN [53]	52.4	66.1	71.2	30.5	61.0	77.0	82.5	30.6
SNR [18]	55.1	-	-	**33.6**	66.7	-	-	33.9
MetaBIN (Ours)	**55.2**	**69.0**	**74.4**	33.1	**69.2**	**83.1**	**87.8**	**35.9**

Ablation Study

- Meta-learning pipeline is matter.

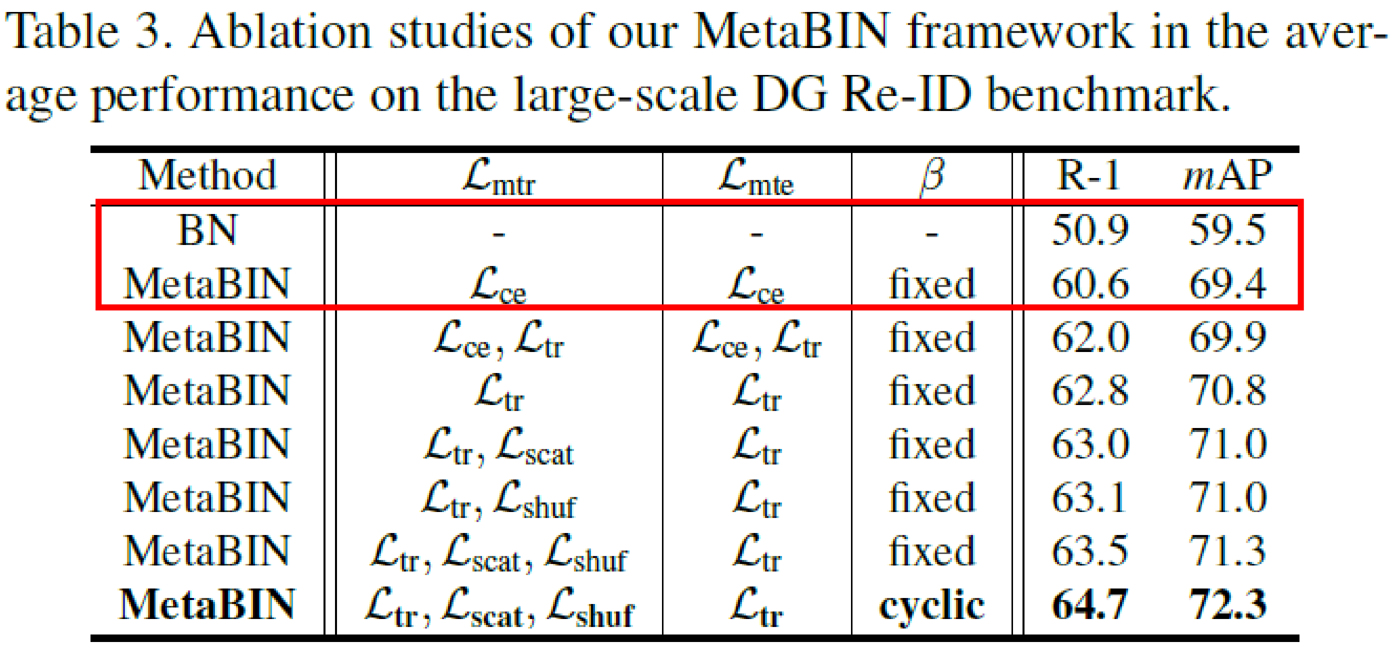
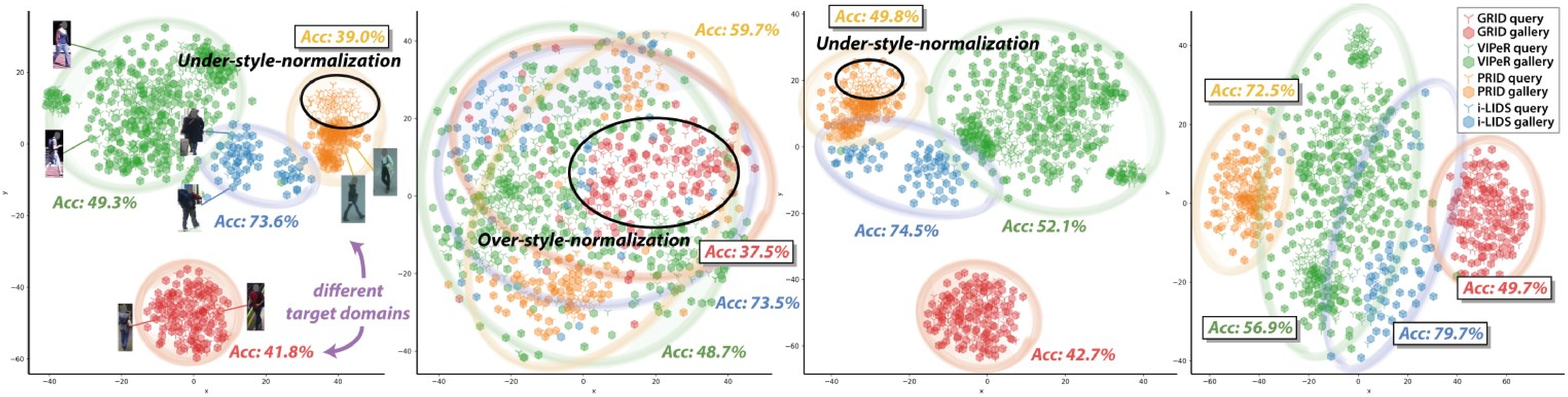


Table 3. Ablation studies of our MetaBIN framework in the average performance on the large-scale DG Re-ID benchmark.

Method	$\mathcal{L}_{\text{mtr}}$	$\mathcal{L}_{\text{mte}}$	β	R-1	mAP
BN	-	-	-	50.9	59.5
MetaBIN	$\mathcal{L}_{\text{ce}}$	$\mathcal{L}_{\text{ce}}$	fixed	60.6	69.4
MetaBIN	$\mathcal{L}_{\text{ce}}, \mathcal{L}_{\text{tr}}$	$\mathcal{L}_{\text{ce}}, \mathcal{L}_{\text{tr}}$	fixed	62.0	69.9
MetaBIN	$\mathcal{L}_{\text{tr}}$	$\mathcal{L}_{\text{tr}}$	fixed	62.8	70.8
MetaBIN	$\mathcal{L}_{\text{tr}}, \mathcal{L}_{\text{scat}}$	$\mathcal{L}_{\text{tr}}$	fixed	63.0	71.0
MetaBIN	$\mathcal{L}_{\text{tr}}, \mathcal{L}_{\text{shuf}}$	$\mathcal{L}_{\text{tr}}$	fixed	63.1	71.0
MetaBIN	$\mathcal{L}_{\text{tr}}, \mathcal{L}_{\text{scat}}, \mathcal{L}_{\text{shuf}}$	$\mathcal{L}_{\text{tr}}$	fixed	63.5	71.3
MetaBIN	$\mathcal{L}_{\text{tr}}, \mathcal{L}_{\text{scat}}, \mathcal{L}_{\text{shuf}}$	$\mathcal{L}_{\text{tr}}$	**cyclic**	**64.7**	**72.3**

T-SNE Visualization

- BN, BIN: under-style-normalization.
- IN: over-style-normalization.



(a) Batch Normalization (BN) (b) Instance Normalization (IN) (c) Batch-Instance Normalization (BIN) (d) MetaBIN (Ours)

Conclusion

- The paper proposes MetaBIN that improve the model generalization ability by unsuccessful generalization scenarios in a meta-learning manner.

- Pros
    - Nice observation (over-/under-style-normalization) to motivates the proposed method.
    - Intuitive method to overcome their observation.
    - Extensive experimental results (e.g., visualization, a lot of ablation study) & analysis.
- Cons
    - The proposed method is quite complex.
    - Too many hyperparameters (e.g., there are three learning rate).

Thank you for listening!

Appendix

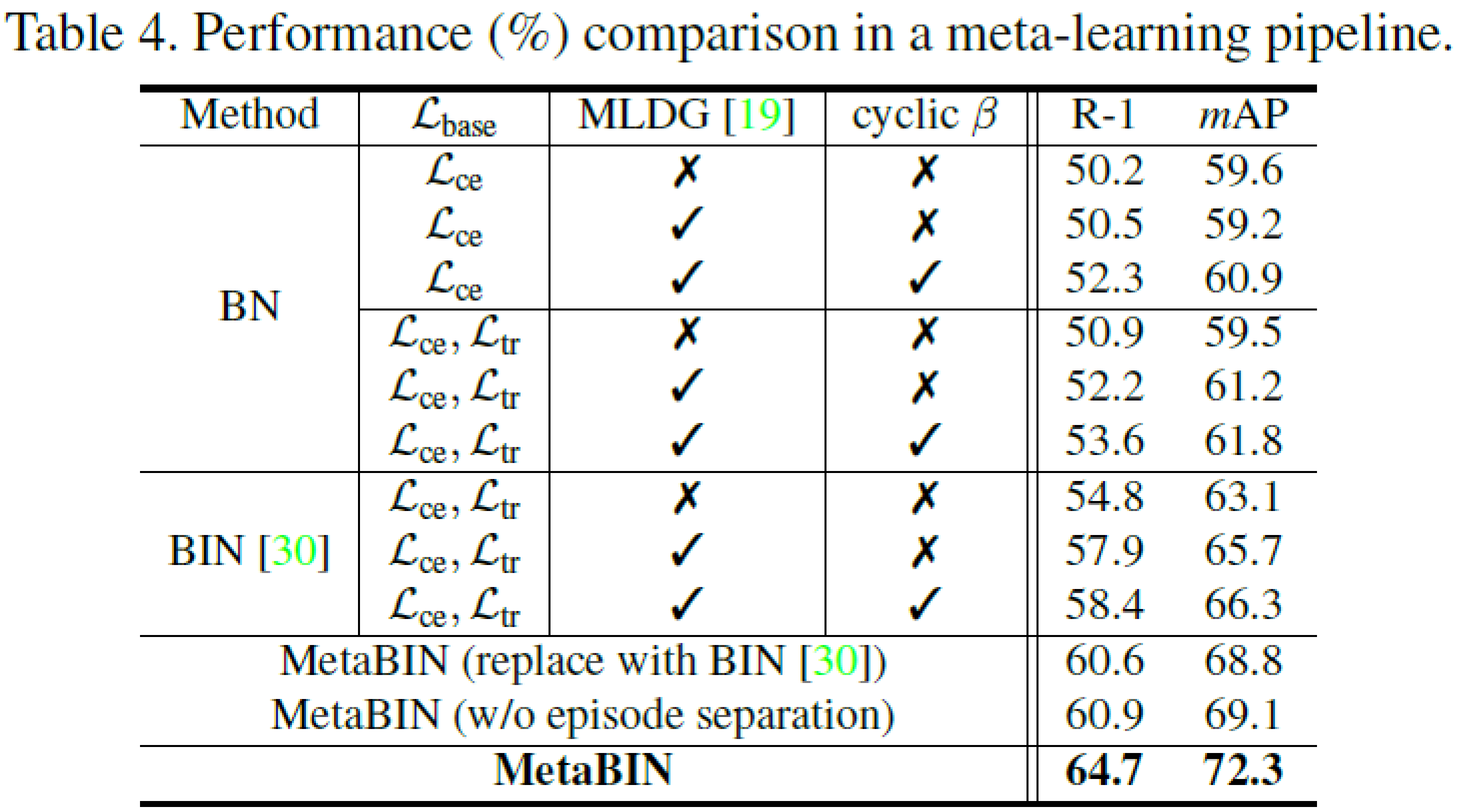


Table 4. Performance (%) comparison in a meta-learning pipeline.

Method	$\mathcal{L}_{\text{base}}$	MLDG [19]	cyclic β	R-1	*m*AP
BN	$\mathcal{L}_{\text{ce}}$	✗	✗	50.2	59.6
	$\mathcal{L}_{\text{ce}}$	✓	✗	50.5	59.2
	$\mathcal{L}_{\text{ce}}$	✓	✓	52.3	60.9
	$\mathcal{L}_{\text{ce}}, \mathcal{L}_{\text{tr}}$	✗	✗	50.9	59.5
	$\mathcal{L}_{\text{ce}}, \mathcal{L}_{\text{tr}}$	✓	✗	52.2	61.2
	$\mathcal{L}_{\text{ce}}, \mathcal{L}_{\text{tr}}$	✓	✓	53.6	61.8
BIN [30]	$\mathcal{L}_{\text{ce}}, \mathcal{L}_{\text{tr}}$	✗	✗	54.8	63.1
	$\mathcal{L}_{\text{ce}}, \mathcal{L}_{\text{tr}}$	✓	✗	57.9	65.7
	$\mathcal{L}_{\text{ce}}, \mathcal{L}_{\text{tr}}$	✓	✓	58.4	66.3
MetaBIN (replace with BIN [30])				60.6	68.8
MetaBIN (w/o episode separation)				60.9	69.1
MetaBIN				**64.7**	**72.3**

Appendix

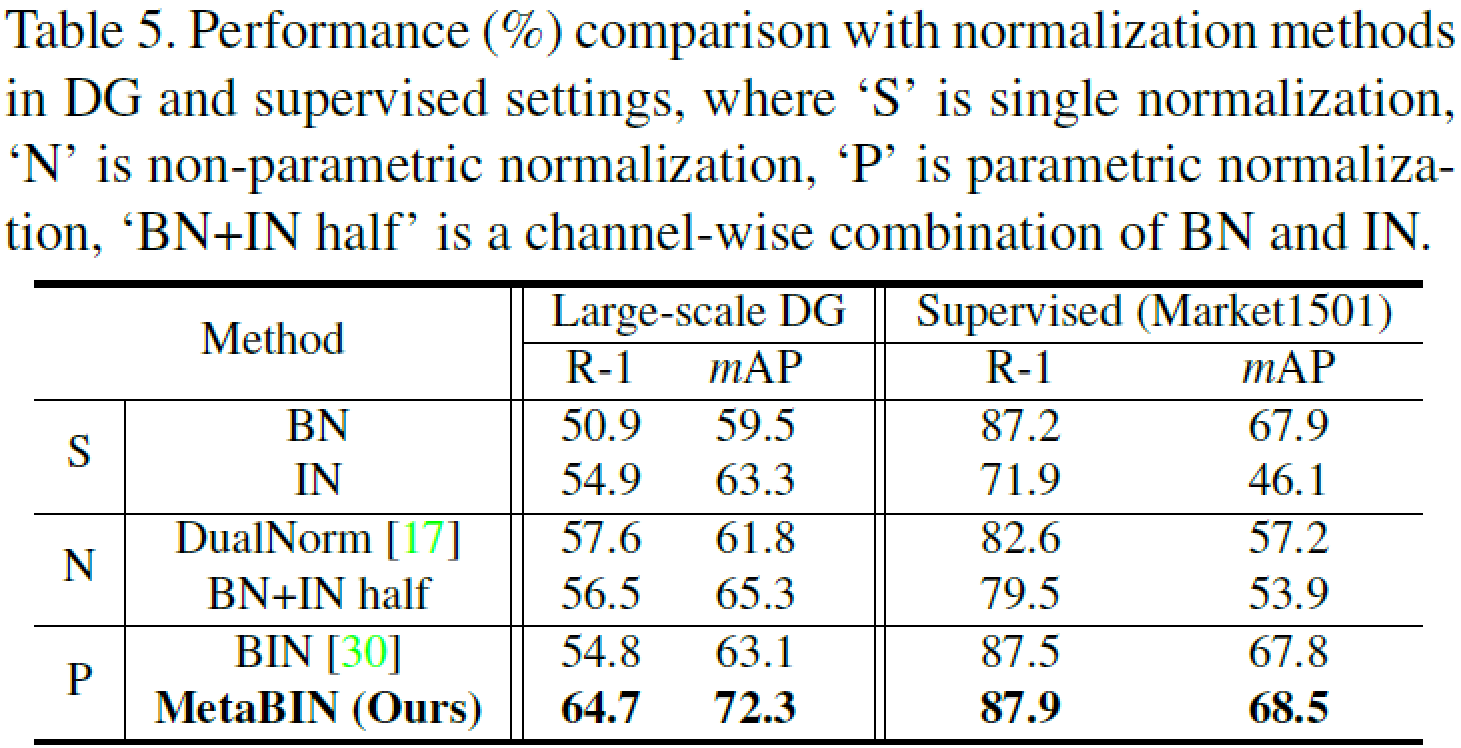


Table 5. Performance (%) comparison with normalization methods in DG and supervised settings, where 'S' is single normalization, 'N' is non-parametric normalization, 'P' is parametric normalization, 'BN+IN half' is a channel-wise combination of BN and IN.

Method		Large-scale DG		Supervised (Market1501)	
		R-1	*m*AP	R-1	*m*AP
S	BN	50.9	59.5	87.2	67.9
	IN	54.9	63.3	71.9	46.1
N	DualNorm [17]	57.6	61.8	82.6	57.2
	BN+IN half	56.5	65.3	79.5	53.9
P	BIN [30]	54.8	63.1	87.5	67.8
	MetaBIN (Ours)	**64.7**	**72.3**	**87.9**	**68.5**