Adaptive Cross-Domain Learning for Generalizable Person Re-Identification

Pengyi Zhang[1], Huanzhang Dou[1], Yunlong Yu[2⋆], and Xi Li[1,3,4⋆]

[1] College of Computer Science & Technology, Zhejiang University
[2] College of Information Science & Electronic Engineering, Zhejiang University
[3] Shanghai Institute for Advanced Study, Zhejiang University
[4] Shanghai AI Laboratory
{pyzhang,hzdou,yuyunlong,xilizju}@zju.edu.cn
https://github.com/peterzpy/ACL-DGReID

Abstract. Domain Generalizable Person Re-Identification (DG-ReID) is a more practical ReID task that is trained from multiple source domains and tested on the unseen target domains. Most existing methods are challenged for dealing with the shared and specific characteristics among different domains, which is called the domain conflict problem. To address this problem, we present an Adaptive Cross-domain Learning (ACL) framework equipped with a CrOss-Domain Embedding Block (CODE-Block) to maintain a common feature space for capturing both the domain-invariant and the domain-specific features, while dynamically mining the relations across different domains. Moreover, our model adaptively adjusts the architecture to focus on learning the corresponding features of a single domain at a time without interference from the biased features of other domains. Specifically, the CODE-Block is composed of two complementary branches, a dynamic branch for extracting domain-adaptive features and a static branch for extracting the domain-invariant features. Extensive experiments demonstrate that the proposed approach achieves state-of-the-art performances on the popular benchmarks. Under Protocol-2, our method outperforms previous SOTA by 7.8% and 7.6% in terms of mAP and rank-1 accuracy.

Keywords: Adaptive cross-domain learning, Common feature space, Dynamic Network, Domain conflict, Domain generalizable person re-identification

1 Introduction

Recently, Domain Generalizable Person Re-Identification (DG-ReID), a more practical ReID task that aims at identifying samples from unseen domains without domain adaptation, has attracted increasing attention due to the conventional ReID approaches suffering from significant performance degradation on the unseen domains. To address this task, some efforts [25, 24, 41, 40, 51, 43, 5, 61]

⋆ Co-corresponding authors.

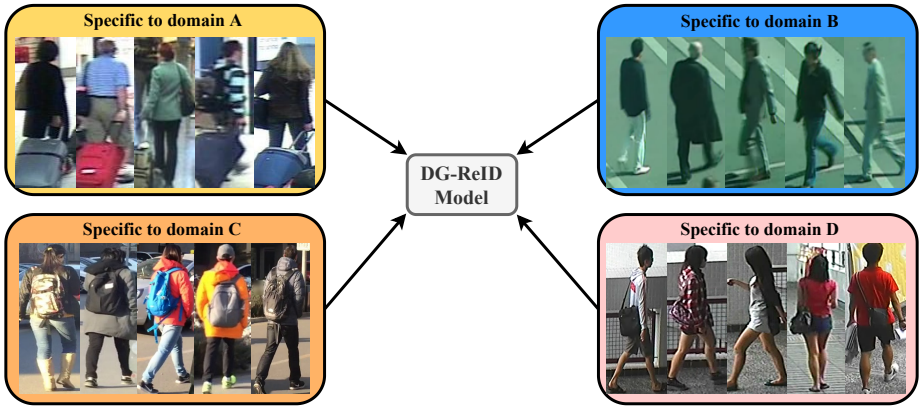

Fig. 1. Examples of multiple person re-identification domains. Different domains are biased on their specific characteristics (e.g., hue, illumination, resolution, clothing style, and carrying objects apart from the common characteristics).

propose to jointly train a static ReID model with multiple seen domains and then directly apply it to the unseen domains. However, as shown in Fig. 1, there is a huge gap between different domains (e.g., illumination, hue, resolution, clothing style, and carrying objects), where each domain has its specific characteristics apart from the common characteristics. During joint training, the specific characteristics of one domain may be useless or even interfere with the learning of other domains, which is called the *domain conflict problem* [45, 28, 27].

To mitigate the domain conflict problem, the early approaches [60, 21, 57, 38] focus on modeling the invariant features between different domains with disentanglement learning or meta-learning. However, these methods do not consider the diverse and complementary information of domain-specific features enough, which may limit the generalization capability on the unseen target domain [63]. Recently, in addition to modeling invariant features, some approaches [59, 2, 42] also focus on capturing the domain-specific features via designing specific expert networks for the individual domains. Such ways of maintaining a specific space for each domain help reduce the cross-domain interference during training but hardly transfer the specific knowledge from different spaces and capture the specific features from unseen domains. Besides, these methods are challenged by the problem of linear increasing costs, since the number of individual feature spaces must be consistent with the number of domains.

To overcome the above limitations, in this paper, we propose to improve the existing methods from two perspectives: maintaining a common space for both domain-invariant and domain-specific features, and adaptively capturing features through the dynamically adjusted architecture. First, maintaining a common space could capture the relations between different domains, thus benefiting in the knowledge transfer across different domains and avoiding the redundant modeling among domains. Second, capturing the domain-adaptive fea-

tures through adaptive dynamic architectures enables the model to learn specific features for each novel domain and alleviate the conflicts among domains.

For this purpose, we propose a novel framework called **A**daptive **C**ross-domain **L**earning (ACL) to dynamically capture the adaptive features with both invariant and specific features for each domain. Specifically, we design a **C**r**O**ss-**D**omain **E**mbedding Block (CODE-Block), which is composed of three components, i.e., a dynamic branch, a static branch, and a fusion module. The dynamic branch is designed with a series of parallel feature embedding networks to reduce the cross-domain interference, each of which captures either the fine-grained domain-invariant or domain-specific features, and a domain-aware adapter to produce meta-weights for adaptively capturing domain-adaptive information. The static branch serves as a complement to the dynamic branch to additionally extract the domain-invariant features among domains and makes the whole training process more stable. Following [53], the static branch adopts a style normalization layer for filtering out domain-specific contrast information. By doing this, our framework could focus on learning the corresponding features of a single domain at a time without interference from other domain-specific features. As a result, the domain conflict issue would be alleviated.

In a nutshell, our highlights include:

- To the best of our knowledge, we are the first to adopt the dynamic network to adaptively learn features from different domains for tackling the domain conflict problem in DG-ReID.
- We develop a novel framework equipped with a CODE-Block to adaptively capture both the domain-invariant and the domain-specific features in a common feature space, and aggregate them through a domain-aware adapter to adaptively learn different domains.
- Extensive experiments demonstrate the effectiveness of our framework under multiple testing protocols. The proposed approach obtains 7.8% and 7.6% improvements in terms of mAP and rank-1 accuracy over the SOTA method on the average results of four large-scale benchmarks under Protocol-2.

2 Related Work

2.1 Domain Generalization

Domain Generalization [20, 15, 55, 36, 1, 64, 29, 34, 44, 31, 7] is a solution to the potential domain shift problem in practice, which can be categorized into two types of solutions. 1) Representation learning. These methods reduce the discrepancy between different domains by disentanglement learning or probability distribution alignment. For instance, [54] learns a feature transformation to minimize the variances of the class-conditional distributions among multiple source domains for all classes. 2) Domain augmentation. These methods generate more cross-domain samples to regularize the model for avoiding overfitting and improving generalization ability. For example, [39] trains a domain classifier and use its adversarial gradients to generate the cross-domain samples.

Although previous DG methods obtain remarkable performance on closed-set tasks (e.g., classification), they are based on the premise that all the training and testing domains share the same label space. However, there is no overlap of identities under the retrieval tasks like ReID. As a result, it is difficult to perform well when directly applying these methods to ReID.

2.2 Domain Generalizable Person Re-Identification

With the application of ReID technology in practice, several DG methods tailored for ReID [25, 24, 41, 40, 51, 43, 5, 61] have been proposed recently. There are two main categories of DG-ReID methods. 1) Domain-invariant feature modeling [60, 37, 50, 21, 18, 57, 19, 38, 4]. These methods aim to learn the shared features among multiple source domains, which could reduce the biases in the single domain and obtain a more robust generalization capability. Specifically, these methods could be achieved by domain-adversarial learning [38], feature disentanglement learning [57], joint training with meta-learning [60], or some normalization-based strategies [37]. 2) Domain-specific feature modeling [59, 56, 2, 42, 6]. These methods aim to leverage the diversity of each source domain for complementary learning through modeling the relevance between seen source domains and unseen target domains. Typically, these methods are implemented through Mixture-of-Expert (MoE), where each expert extracts domain-specific features from the corresponding domain. As for the relevance between different domains, it can be calculated using the similarity between features of different domains [56] or the similarity between the IN/BN statistics of different domains [2], as well as directly predicted through a specific network [59].

Compared with the previous methods, our approach can model both the invariant/specific features in a common feature space. Besides, our dynamic network with the cross-domain embedding can better model the cross-domain relevance and reduce the redundant modeling for each domain than previous MoE-based methods.

2.3 Dynamic Neural Networks

Dynamic neural networks [58], including dynamic architecture [13, 16, 65, 22] and dynamic parameters [33, 52, 8, 35, 11, 9], are wildly used in many fields for their satisfactory representation capability, adaptiveness, and generality. These methods usually adjust the model architectures to allocate appropriate computation based on each sample, which is more adaptive to the specific sample. Motivated by this, we build a dynamic network to solve the domain conflicts in DG-ReID through domain-adaptive architecture adjusting, then learn the meta-knowledge among different domains implicitly for better generalization performance.

3 Adaptive Cross-Domain Learning Framework

In this work, we present a novel **A**daptive **C**ross-domain **L**earning (ACL) framework to mitigate the domain conflict issue for DG-ReID via adaptively modeling

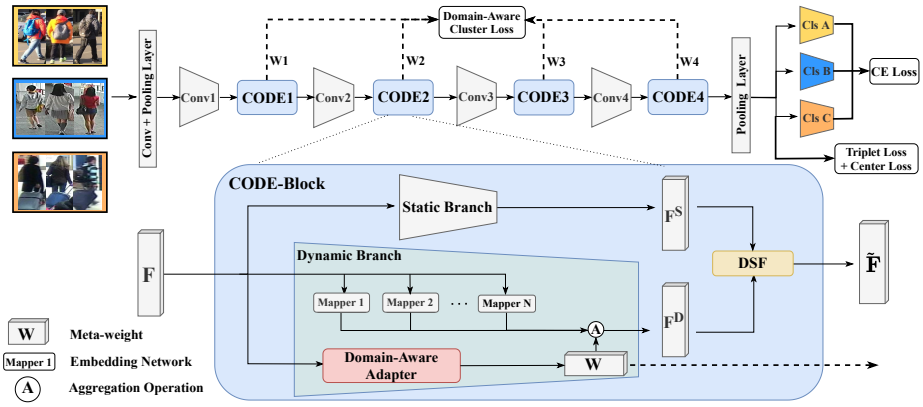

Fig. 2. Illustration of our method. Each conv block means a stage of convolution blocks without the last one. We replace the last convolutional block in each stage with our CODE-Block. In the CODE-Block, we extract a dynamic domain-adaptive feature F^D and a static domain-invariant feature F^S, then we fuse these two features through a dynamic-static fusion module (DSF). Notably, to reduce the domain conflicts, we calculate the cross-entropy loss for each domain by individual classifiers, respectively.

both the common features and specific features from different input domains in a common feature space with a dynamic architecture. In this section, we first give an overview of ACL and then introduce the core component, a novel **Cr**ross-**D**omain **E**mbedding Block (CODE-Block), followed by the detailed objective functions and the optimization process of our ACL framework.

3.1 Overview

As illustrated in Fig. 2, our ACL framework is designed by plugging the CODE-Block into the different layers of the feature extractor architecture, respectively for extracting different levels of semantics (e.g., hue, and contrast in the low-level, and carrying objects, viewpoint, and clothing style in the high-level). To reduce the computation cost, the CODE-Block is plugged into the architecture by replacing the final convolutional blocks at each stage.

3.2 Cross-Domain Embedding Block

As shown in Fig. 2, the CODE-Block is designed for modeling both the domain-invariant/specific features from different domains, which consists of a dynamic branch, a static branch, and a fusion module. Let $F \in \mathbb{R}^{H\times W\times C}$ be the input feature map of CODE-Block, where H, W, and C indicate the height, width, and the number of channels, respectively. The CODE-Block is processed as follows:

1) Dynamic branch extracts fine-grained features $\{F_i^D\}_{i=1}^N$ for capturing domain-invariant or domain-specific features through several parallel embedding

networks, where N is the number of embedding networks. Then discriminative domain-adaptive features $F^D \in \mathbb{R}^{H\times W\times C}$ are aggregated with the guidance of the meta-weights $W \in \mathbb{R}^N$ generated by domain-aware adapter; 2) Static branch extracts the robust domain-invariant features $F^S \in \mathbb{R}^{H\times W\times C}$ from the input feature F; 3) Fusion module integrates the F^S from static branch and the F^D from dynamic branch into the final output feature $\tilde{F} \in \mathbb{R}^{H\times W\times C}$.

Dynamic Branch. Dynamic Branch is the core design of the CODE-Block, which consists of several parallel *embedding networks* to model a common feature space for all domains and a *domain-aware adapter* to guide the combination of the discriminative domain-adaptive features for each domain.

1) Embedding Networks. Due to the fact that different domains significantly differ in hue, carrying objects, resolution, and many other principles, it is hard to capture good features for the target domains with a single network, thus we design a series of parallel networks to capture both domain-invariant and domain-specific features in an implicit way, each of which aims at capturing domain-invariant features or domain-specific features for each domain.

Specifically, we build N parallel embedding networks (i.e., bottleneck block in ResNet-IBN but with different initialization). For a light computation cost, we reduce the number of channels for intermediate features to $1/N$. The fine-grained features $\{F_i^D\}_{i=1}^N$ from i-th embedding network are obtained with:

$$F_i^D = Embed_i(F),\ s.t.\ i \in [1 \cdots N], \tag{1}$$

where $Embed_i$ indicates the i-th embedding network. Note that all the fine-grained features from different embedding networks are embedded into the same space, which is the so-called common feature space.

Parallel embedding networks could capture the fine-grained domain-invariant and domain-specific features as the bases for the following combination. In contrast to previous methods [2, 56, 59] that model the features for each domain using domain-level individual experts, our strategy simultaneously learns domain-invariant and domain-specific features in multiple parallel embedding networks, thus reducing the risk of cross-domain interference and redundant modeling.

2) Domain-Aware Adapter. To adaptively capture the features for each domain, as shown in Fig. 3, the Domain-Aware Adapter produces meta-weights with a weight generator for aggregating the fine-grained features extracted from all embedding networks. The weight generator takes the original features as input and produces the meta-weights. To further enhance the sample-adaptive capability, we propose to generate the parameters of Domain-Aware Adapter with a meta-parameter generator, which is formulated with:

$$W_g = \delta(W_2\delta(W_1 pool(F))), \tag{2}$$

which consists of a global average pooling layer followed by two FC layers that are parameterized by $W_1 \in \mathbb{R}^{\frac{C}{r}\times C}$ and $W_2 \in \mathbb{R}^{NC\times\frac{C}{r}}$ with a reduction ratio r that is set to 16, and the $\delta(\cdot)$ indicates the ReLU activation function.

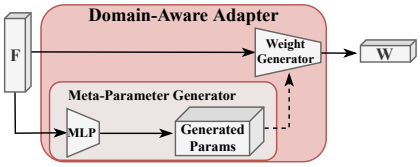

Fig. 3. Architecture of the domain-aware adapter.

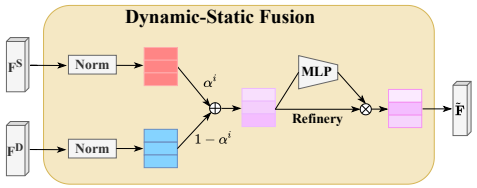

Fig. 4. Architecture of the fusion module. Best viewed in color.

To this end, the meta-weights are obtained with $W = \sigma(W_g pool(F))$, where W has the dimension of $\mathbb{R}^N$ and $\sigma(\cdot)$ denotes the Softmax function. Through the domain-aware combination of the cross-domain features, we integrate the final domain-adaptive feature with $F^D = \sum_{i=1}^{N} W_i F_i^D$.

Static Branch. Apart from the dynamic branch that dynamically extracts discriminative domain-adaptive features for each domain, we additionally maintain a static branch to extract the domain-invariant features for robust lower-bound performance. Specifically, the Static Branch has the same architecture as a conventional bottleneck block in ResNet, but we replace the batch normalization (BN) layers in the bottleneck block with the instance normalization (IN) layers. IN can be regarded as a style normalization [53], which filters the specific features in different domains to reduce the discrepancy of samples, has been widely used in many previous methods [50, 37]. Therefore, we could statically extract the robust domain-invariant features for stabilizing and complementing the adaptive features extracted from the dynamic branch.

Fusion Module. Once the dynamic domain-adaptive feature F^D and static domain-invariant feature F^S are extracted, we further develop a fusion module called Dynamic-Static Fusion (DSF) to integrate these two features for complementation. As shown in Fig. 4, DSF first respectively normalizes the two input features, then sums up them with an optimized weight vector $\alpha \in \mathbb{R}^C$, finally refines the fused feature using channel attention [17], which is formulated as:

$$\begin{aligned} \tilde{F} &= \alpha * Norm_S(F^S) + (1-\alpha) * Norm_D(F^D), \\ \tilde{F} &= \tilde{F} * Sigmoid(W_2^r \delta(W_1^r pool(\tilde{F}))), \end{aligned} \tag{3}$$

where $*$ notes the elementwise multiplication, $\delta(\cdot)$ denotes the ReLU function, W_1^r and W_2^r are FC layers parameterized with $\mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbb{R}^{C \times \frac{C}{r}}$, respectively.

3.3 Objective Function

To train the ACL framework, four loss functions are applied: 1) Domain-Aware Cluster Loss $\mathcal{L}_{cluster}$. To make the generated meta-weights being aware of different domains, we adopt a domain-level cluster regularization to increase the inter-domain variances, while narrowing down the intra-domain variances. Given the

predicted meta-weights of all these modules $W \in \mathbb{R}^{M*N}$, where M is the number of CODE-Block and N is the number of embedding networks in a CODE-Block, the intra-domain loss $\mathcal{L}_{intra}$ and the inter-domain loss $\mathcal{L}_{inter}$ are calculated as follows,

$$\begin{aligned} \mathcal{L}_{intra} &= \frac{1}{K \times L} \sum_{i=1}^{K} \sum_{j=1}^{L} [||W_{i,j} - \overline{W_{i,:}}||_2 - m_1]_+^2, \\ \mathcal{L}_{inter} &= \frac{1}{K(K-1)} \sum_{i=1}^{K} \sum_{j \neq i}^{K} [m_2 - ||\overline{W_{i,:}} - \overline{W_{j,:}}||_2]_+^2, \end{aligned} \tag{4}$$

where K is the number of domains, m_1 and m_2 are the margins, L is the number of instances for the corresponding domain, $W_{i,j}$ means the j-th instance of the domain i, and $\overline{W_{i,:}}$ means the cluster center of the domain i. As a result, the final cluster loss $\mathcal{L}_{cluster}$ can be aggregated as follows,

$$\mathcal{L}_{cluster} = \mathcal{L}_{intra} + \mathcal{L}_{inter}. \tag{5}$$

2) Cross-Entropy Loss $\mathcal{L}_{ce}$, Triplet Loss $\mathcal{L}_{tri}$, and Center Loss $\mathcal{L}_{cntr}$. These three losses follow the standard formulation in previous works [30]. Notably, to further reduce the conflicts between different domains, we calculate the cross-entropy loss for each domain, respectively.

3.4 Optimization Process

To improve the generalization capability on the unseen target domains, we adopt a meta-learning algorithm to simulate the unseen domain scenarios following [37]. As shown in Algorithm 1, the training process of our ACL framework is divided into two stages, a basic training process to train the whole network, and a meta-learning process for optimizing the domain-aware adapter module.

For the first basic training stage, the objective function $\mathcal{L}_{basic}$ consists of cross-entropy loss, triplet loss, center loss, and our cluster loss,

$$\mathcal{L}_{basic}(\mathcal{X}; \theta; \phi) = \mathcal{L}_{ce} + \mathcal{L}_{tri} + \mathcal{L}_{cntr} + \mathcal{L}_{cluster}, \tag{6}$$

where all the parameters in the whole network (i.e., θ for the domain-aware adapter and ϕ for others) will be updated by optimizing these loss functions. For the second meta-learning stage, we follow the process and objective functions of [37], which adds two cluster losses $\mathcal{L}_{scat}$ and $\mathcal{L}_{shuf}$ for a more discriminative feature representation. Specifically, $\mathcal{L}_{scat}$ is to make the feature distribution more compact in each domain and $\mathcal{L}_{shuf}$ is to avoid the model collapse for each domain. We split the K seen domains into $K-1$ meta-train domains and a meta-test domain, respectively. Then we calculate the objective function $\mathcal{L}_{mtr}$ for the meta-training stage to calculate the temporary optimized parameters θ' for domain-aware adapter as follows,

$$\mathcal{L}_{mtr}(\mathcal{X}; \theta; \phi) = \mathcal{L}_{tri} + \mathcal{L}_{cluster} + \mathcal{L}_{scat} + \mathcal{L}_{shuf}. \tag{7}$$

Algorithm 1: Training for ACL framework

Input: Source domains $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K\}$; Learning rate γ; MaxIters; Meta-learning Frequency $\mathcal{F}$.

Output: Trained network $F(\theta; \phi)$ with domain-aware adapter θ and other parameters ϕ.

for *iter in MaxIters* **do**
- **Basic Training Stage**:
  - Sample a mini-batch $\mathcal{X}_B$ from $\mathcal{D}$.
  - $\mathcal{L}_{basic}(\mathcal{X}_B; \theta; \phi) = \mathcal{L}_{ce} + \mathcal{L}_{tri} + \mathcal{L}_{cntr} + \mathcal{L}_{cluster}$
  - $(\theta, \phi) \leftarrow (\theta - \gamma \nabla_\theta \mathcal{L}_{basic}(\mathcal{X}_B; \theta; \phi), \phi - \gamma \nabla_\phi \mathcal{L}_{basic}(\mathcal{X}_B; \theta; \phi))$
- **Meta-Learning Stage**:
- if iter % $\mathcal{F}$ == 0:
  - Split $\mathcal{D}$ as ($\mathcal{D}_{mtr} \cap \mathcal{D}_{mte} = \varnothing$, $\mathcal{D}_{mtr} \cup \mathcal{D}_{mte} = \mathcal{D}$)
  - **Meta-Training**:
    - Sample a mini-batch $\mathcal{X}_S$ from $\mathcal{D}_{mtr}$.
    - $\mathcal{L}_{mtr}(\mathcal{X}_S; \theta; \phi) = \mathcal{L}_{tri} + \mathcal{L}_{cluster} + \mathcal{L}_{scat} + \mathcal{L}_{shuf}$
    - $\theta' = \theta - \gamma \nabla_\theta \mathcal{L}_{mtr}(\mathcal{X}_S; \theta; \phi)$
  - **Meta-Testing**:
    - Sample a mini-batch $\mathcal{X}_T$ from $\mathcal{D}_{mte}$.
    - $\mathcal{L}_{mte}(\mathcal{X}_T; \theta'; \phi) = \mathcal{L}_{tri} + \mathcal{L}_{cntr} + \mathcal{L}_{cluster}$
    - $\theta = \theta - \gamma \nabla_\theta \mathcal{L}_{mte}(\mathcal{X}_T; \theta'; \phi)$

With the temporary parameters θ', we can obtain the final optimized parameters θ by optimizing the loss for the meta-testing stage, which is formulated by

$$\mathcal{L}_{mte}(\mathcal{X}; \theta'; \phi) = \mathcal{L}_{tri} + \mathcal{L}_{cntr} + \mathcal{L}_{cluster}. \tag{8}$$

4 Experiments

4.1 Implementation Details

We use ResNet50 with IBN [51] pretrained on ImageNet as our backbone. Similar to previous methods [14, 12], we set the stride of the last layer as 1. Images are resized to 256×128 and the training batch size is set to 64, including 32 identities and two instances for each identity. For data augmentation, we use random horizontal flipping, random cropping, color jittering, and auto augmentation [26]. We optimize the model using the SGD optimizer with a momentum of 0.9 and weight decay of 5e-4 for 60 epochs, and the warmup strategy is used in the first 10 epochs. The margin for intra-domain loss m_1 and inter-domain loss m_2 are set to 0.1 and 0.3, respectively. The meta-learning frequency $\mathcal{F}$ is set to 3. The initial learning rate is set to 4e-2, which is cosine decayed to 4e-5 at the final iteration. We conduct all the experiments with PyTorch on four 1080Ti GPUs.

4.2 Datasets and Evaluation Settings

Datasets. We conduct experiments on several person re-identification benchmarks: Market1501 [23], MSMT17 [27], CUHK02 [48], CUHK03 [47], CUHK-

Table 1. Comparison with state-of-the-art methods under Protocol-1. We achieve better generalization performance even with fewer data for training (i.e., without DukeMTMC dataset). '$*$' denotes the re-implementation for the work under the new protocol, based on the author's code on Github. Best results are highlighted in **bold**.

Method	Reference	Source	Target								Average	
			PRID		GRID		VIPeR		iLIDs			
			mAP	R1	mAP	R1	mAP	R1	mAP	R1	mAP	R1
DIMN [18]	CVPR19	M+D+C2 +C3+CS	52.0	39.2	41.1	29.3	60.1	51.2	78.4	70.2	57.9	47.5
SNR [50]	CVPR20		66.5	52.1	47.7	40.2	61.3	52.9	89.9	84.1	66.4	57.3
RaMoE [59]	CVPR21		67.3	57.7	54.2	46.8	64.6	56.6	**90.2**	**85.0**	69.1	61.5
DMG-Net [56]	CVPR21		68.4	60.6	56.6	51.0	60.4	53.9	83.9	79.3	67.3	61.2
QAConv$_{50}$* [24]	ECCV20	M+C2 +C3+CS	62.2	52.3	57.4	48.6	66.3	57.0	81.9	75.0	67.0	58.2
M^{3}L* [60]	CVPR21		64.3	53.1	55.0	44.4	66.2	57.5	81.5	74.0	66.8	57.3
MetaBIN* [37]	CVPR21		70.8	61.2	57.9	50.2	64.3	55.9	82.7	74.7	68.9	60.5
Ours		M+C2 +C3+CS	**73.4**	**63.0**	**65.7**	**55.2**	**75.1**	**66.4**	86.5	81.8	**75.2**	**66.6**

SYSU [46], PRID [32], GRID [3], VIPeR [10], and iLIDs [49]. For CUHK03, we use the 'labeled' data following [66, 59]. We do not use the DukeMTMC [62] dataset since its privacy issues. The statistics of these datasets can be seen in supplementary material.

For simplicity, we denote Market1501, MSMT17, CUHK02, CUHK03, CUHK-SYSU as M, MS, C2, C3, and CS in the following.

Evaluation Settings. The mean Average Precision (mAP) and Cumulative Matching Characteristics (CMC) are used for evaluation. There are three evaluation protocols following [59, 2, 21].

For Protocol-1, the model is trained by both the training and testing data in M+C2+C3+CS datasets and then tested on four small datasets (i.e., PRID, GRID, VIPeR, and iLIDs), respectively. Since some of these datasets have no official split for probe and gallery sets, we randomly split the probe/gallery sets for ten times and report the averaged evaluation performance as the final result. For Protocol-2 and Protocol-3, we orderly use one of the datasets' data for testing (i.e., only the testing data) and the remaining datasets (Protocol-2 only uses training data, whereas Protocol-3 uses both training and testing data of source domains) for training the model. Notably, all ablation studies below are conducted under Protocol-2. More details can be seen in supplementary material.

4.3 Comparison with the State-of-the-Arts

Comparison with DG-ReID Methods under Protocol-1. We compare our method with previous DG-ReID methods under Protocol-1, which are tested on four datasets (i.e., PRID, GRID, VIPeR, and iLIDs). As shown in Table 1, our method could outperform previous methods by a large margin.

Table 2. Comparison with state-of-the-art methods under Protocol-2 and Protocol-3. Four large-scale datasets are involved in the leave-one-out setting.

Setting	Method	Reference	M+MS +CS→C3		M+CS +C3→MS		MS+CS +C3→M		Average	
			mAP	R1	mAP	R1	mAP	R1	mAP	R1
Protocol-2	SNR* [50]	CVPR20	8.9	8.9	6.8	19.9	34.6	62.7	16.8	30.5
	QAConv$^*_{50}$ [24]	ECCV20	25.4	24.8	16.4	45.3	63.1	83.7	35.0	51.3
	M^{3}L* [60]	CVPR21	34.2	34.4	16.7	37.5	61.5	82.3	37.5	51.4
	MetaBIN* [37]	CVPR21	28.8	28.1	17.8	40.2	57.9	80.1	34.8	49.5
	Ours		**41.2**	**41.8**	**20.4**	**45.9**	**74.3**	**89.3**	**45.3**	**59.0**
Protocol-3	SNR* [50]	CVPR20	17.5	17.1	7.7	22.0	52.4	77.8	25.9	39.0
	QAConv$^*_{50}$ [24]	ECCV20	32.9	33.3	17.6	46.6	66.5	85.0	39.0	55.0
	M^{3}L* [60]	CVPR21	35.7	36.5	17.4	38.6	62.4	82.7	38.5	52.6
	MetaBIN* [37]	CVPR21	43.0	43.1	18.8	41.2	67.2	84.5	43.0	56.3
	Ours		**49.4**	**50.1**	**21.7**	**47.3**	**76.8**	**90.6**	**49.3**	**62.7**

Comparison with DG-ReID Methods under Protocol-2 and Protocol-3. We also conduct the experiments under the leave-one-out setting with different amounts of data. As shown in Table 2, under these two protocols with large-scale datasets, our method still maintains the superiority in learning a more generalizable feature representation from multiple domains.

4.4 Ablation Study

Effectiveness of the Static/Dynamic Branches. As we have stated above, our CODE-Block consists of two branches, a static branch to extract the domain-invariant features and a dynamic branch to extract the domain-adaptive features. As shown in Table 3, we explore the effects of these two branches to demonstrate the effectiveness of our framework. 'Baseline' indicates a vanilla model without our CODE-Block module, which follows the same training setting as our basic training stage. Our training pipeline shows a great improvement, which could achieve a considerable performance even with just the dynamic branch adopted, showing the effectiveness of the adaptive feature modeling. And the performance can be further improved with the complement of the static branch.

Effectiveness of the Domain-Aware Adapter. Our domain-aware adapter adopts the *meta-parameter generator* to predict the parameter weights of the weight generator for adaptively generate *domain-aware meta-weights* of each sample. The effectiveness of these components is explored in Table 4, respectively.

We first compare our pipeline with a standard MLP-style weight generator, which also consists of two FC layers (i.e., $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $W_2 \in \mathbb{R}^{N \times \frac{C}{r}}$) but directly predicts the combination weights. Compared to this counterpart, our pipeline can be more adaptive to each sample, which better reduces the conflicts among domains. Moreover, the meta-parameter generator can be regarded as a

Table 3. Ablation study on the effectiveness of the static and dynamic branches.

Method	M+MS +CS→C3		M+CS +C3→MS		MS+CS +C3→M		Average	
	mAP	R1	mAP	R1	mAP	R1	mAP	R1
Baseline	34.1	34.4	17.3	42.7	69.2	87.0	40.2	54.7
Static branch	35.4	35.1	17.2	41.7	69.4	87.8	40.7	54.9
Dynamic branch	37.5	37.7	18.7	43.4	72.2	88.2	42.8	56.4
Static+Dynamic (ACL)	**41.2**	**41.8**	**20.4**	**45.9**	**74.3**	**89.3**	**45.3**	**59.0**

Table 4. Ablation study on the effectiveness of the meta-parameter generator and the generated domain-aware meta-weights of domain-aware adapter (DAA).

Method	M+MS +CS→C3		M+CS +C3→MS		MS+CS +C3→M		Average	
	mAP	R1	mAP	R1	mAP	R1	mAP	R1
Standard MLP	38.9	37.8	19.3	44.6	72.6	88.7	43.6	57.0
Meta-Parameter (ACL)	**41.2**	**41.8**	**20.4**	**45.9**	**74.3**	**89.3**	**45.3**	**59.0**
Identity weight	37.9	38.2	19.4	45.3	72.9	88.7	43.4	57.4
Softmax w/o DAA	40.0	39.4	19.6	45.2	73.0	89.0	44.2	57.9
Softmax w/ DAA (ACL)	**41.2**	**41.8**	**20.4**	**45.9**	**74.3**	**89.3**	**45.3**	**59.0**

meta-learning process [9], which further enhances the generalization capability of our model. Therefore, our pipeline surpasses the standard MLP by a considerable margin (i.e., 1.7% mAP and 2.0% rank-1 accuracy).

As shown in Table 4, we demonstrate the effectiveness of our domain-aware meta-weights by comparing it with the naive ensemble model (i.e., the sub mapping features are average summed). Compared with this variant, our domain-aware meta-weights could fully utilize the fine-grained modeling of cross-domain embedding without conflicts among domains. Besides, we also demonstrate the effectiveness of our domain-aware cluster loss for the meta-weights. As shown in Table 4, with the domain-level cluster loss, we could explicitly model the relationship between weights and the domain, and improve the averaged performance by 1.1% mAP and 1.1% rank-1 accuracy.

Effectiveness of the Fusion Module. As shown in Table 5, we demonstrate the effectiveness of the three components in the DSF module, respectively. Without the fusion module, fusing the two branches features using direct summation helps but improves marginally. With the proposed normalization layer, weighted summation layer, and the final refinery channel attention layer, these two features can be fused better and achieve a better generalization performance. As a result, with the fusion module, the averaged performance is improved by 1.8% mAP and 2.3% rank-1 accuracy, respectively.

Table 5. Ablation study on the effectiveness of the fusion module.

Method	M+MS +CS→C3		M+CS +C3→MS		MS+CS +C3→M		Average	
	mAP	R1	mAP	R1	mAP	R1	mAP	R1
W/O DSF	38.4	37.7	19.4	44.2	72.6	88.2	43.5	56.7
+Norm	39.3	38.6	20.0	45.3	72.8	89.0	44.0	57.6
+Weighted sum	40.0	40.4	20.1	45.6	74.1	89.1	44.7	58.4
+Refinery (ACL)	**41.2**	**41.8**	**20.4**	**45.9**	**74.3**	**89.3**	**45.3**	**59.0**

Table 6. Ablation study on the effects of different embedding stages for the CODE-Block. 'Stage-1' notes that only embed it in the first stage of the backbone network.

Method	M+MS +CS→C3		M+CS +C3→MS		MS+CS +C3→M		Average	
	mAP	R1	mAP	R1	mAP	R1	mAP	R1
Stage-1	37.1	35.9	18.2	42.4	72.7	88.2	42.7	55.5
Stage-2	37.4	37.1	19.0	43.6	72.8	87.9	43.1	56.2
Stage-3	37.8	36.8	19.0	43.8	73.0	88.1	43.3	56.2
Stage-4	38.1	38.2	19.2	44.2	73.5	88.3	43.6	56.9
Stages-all (ACL)	**41.2**	**41.8**	**20.4**	**45.9**	**74.3**	**89.3**	**45.3**	**59.0**

Effects of Different Stages for Embedding CODE-Block. As an embedding module, CODE-Block can be plugged into multiple stages of the backbone network. In this part, we explore the effects of different embedding stages. Specifically, we compare our framework with four variants embedding CODE-Block in different stages. As shown in Table 6, embedding in higher semantic level turns to have a better performance. Moreover, the combination of multiple CODE-Blocks from different semantic levels could better model the features across domains and obtain a better generalization performance.

The Number of Embedding Networks for CODE-Block. As shown in Fig. 5, we analyze the effects of the number N of parallel embedding networks in a CODE-Block. In general, the more embedding networks embedded, the more diverse and fine-grained features can be modeled, but there will be more optimization challenges at the same time. As a result, we choose to use four embedding networks in each CODE-Block for better generalization performance.

4.5 Visualization

To further demonstrate the effectiveness of our domain-aware adapter, we visualize the distribution of generated meta-weights from different domains. As shown in Fig. 6, the meta-weights can be well clustered into their corresponding domains while also having good diversity in each domain cluster.

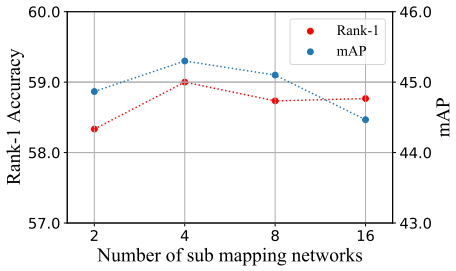

Fig. 5. The averaged performance under Protocol-2 with the different number of parallel embedding networks for the CODE-Block. Best viewed in color.

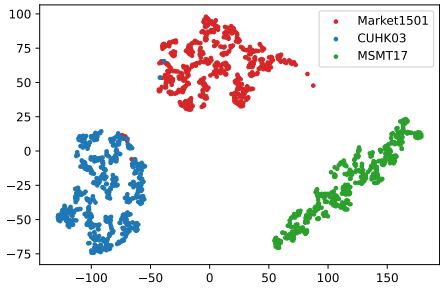

Fig. 6. The t-SNE visualization of meta-weights generated by the domain-aware adapter for target data from different domains. Best viewed in color.

5 Discussion

Since the ReID system may be abused and violate people's privacy, governments and officials must create regulations and legislation for governing the use of the ReID system. However, person ReID is an established computer vision problem with known benchmarks. The research for ReID technology under authorized datasets should not be forbidden for the ethical implications.

6 Conclusion

In this paper, we have proposed a generalizable framework, called Adaptive Cross-Domain Learning (ACL) for tackling the problem of domain generalizable person re-identification (DG-ReID). Our framework is equipped with a novel embedding CrOss-Domain Embedding Block (CODE-Block) that enables the model to adaptively adjust the architecture and capture the adaptive features with both the domain-invariant and domain-specific features in a common feature space, which could further integrate the cross-domain relevance. Specifically, two branches have been designed to extract the domain-adaptive and domain-invariant features, then we introduced a fusion module to integrate these two features for complementation. Extensive experiments have shown the effectiveness of our framework, which achieves state-of-the-art generalization performance.

Acknowledgement

This work is supported in part by Zhejiang Provincial Natural Science Foundation of China under Grant LR19F020004, National Key Research and Development Program of China under Grant 2020AAA0107400, National Natural Science Foundation of China under Grant U20A20222, Key R&D Program of Zhejiang Province, China (2021C01119), the National Natural Science Foundation of China under Grant (62002320, U19B2043).

References

1. Aditya, K., Tinghui, Z., Tomasz, M., A, E.A., Antonio, T.: Undoing the damage of dataset bias. In: ECCV. pp. 158–171 (2012)
2. Boqiang, X., Jian, L., Lingxiao, H., Zhenan, S.: Meta: Mimicking embedding via others' aggregation for generalizable person re-identification. arXiv preprint arXiv:2112.08684 (2021)
3. Change, L.C., Tao, X., Shaogang, G.: Time-delayed correlation analysis for multi-camera activity understanding. IJCV **90**(1), 106–129 (2010)
4. Chen, P., Dai, P., Liu, J., Zheng, F., Xu, M., Tian, Q., Ji, R.: Dual distribution alignment network for generalizable person re-identification. In: AAAI. pp. 1054–1062 (2021)
5. Chuanchen, L., Chunfeng, S., Zhaoxiang, Z.: Generalizing person re-identification by camera-aware invariance learning and cross-domain mixup. In: ECCV. pp. 224–241 (2020)
6. Ci-Siang, L., Yuan-Chia, C., Frank, W.Y.C.: Domain generalized person re-identification via cross-domain episodic learning. In: ICPR. pp. 6758–6763 (2021)
7. Da, L., Yongxin, Y., Yi-Zhe, S., M, H.T.: Learning to generalize: Meta-learning for domain generalization. In: AAAI. pp. 3490–3497 (2018)
8. David, H., Andrew, D., V, L.Q.: Hypernetworks. arXiv preprint arXiv:1609.09106 (2016)
9. Dominic, Z., von Oswald Johannes, Seijin, K., João, S., F, G.B.: Meta-learning via hypernetworks. In: NeurIPS (2020)
10. Douglas, G., Hai, T.: Viewpoint invariant pedestrian recognition with an ensemble of localized features. In: ECCV. pp. 262–275 (2008)
11. Feihu, Z., W, W.B.: Supplementary meta-learning: Towards a dynamic model for deep neural networks. In: ICCV. pp. 4344–4353 (2017)
12. Fengliang, Q., Bo, Y., Leilei, C., Hongbin, W.: Stronger baseline for person re-identification. arXiv preprint arXiv:2112.01059 (2021)
13. Gao, H., Shichen, L., der Maaten Laurens, V., Q, W.K.: Condensenet: An efficient densenet using learned group convolutions. In: CVPR. pp. 2752–2761 (2018)
14. Hao, L., Wei, J., Youzhi, G., Fuxu, L., Xingyu, L., Shenqi, L., Jianyang, G.: A strong baseline and batch normalization neck for deep person re-identification. TMM **22**(10), 2597–2609 (2019)
15. Haoliang, L., Jialin, P.S., Shiqi, W., C, K.A.: Domain generalization with adversarial feature learning. In: CVPR. pp. 5400–5409 (2018)
16. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. In: NeurIPS. pp. 4107–4115 (2016)
17. Jie, H., Li, S., Gang, S.: Squeeze-and-excitation networks. In: CVPR. pp. 7132–7141 (2018)
18. Jifei, S., Yongxin, Y., Yi-Zhe, S., Tao, X., M, H.T.: Generalizable person re-identification by domain-invariant mapping network. In: CVPR. pp. 719–728 (2019)
19. Kaiwen, Y., Xinmei, T.: Domain-class correlation decomposition for generalizable person re-identification. arXiv preprint arXiv:2106.15206 (2021)
20. Kaiyang, Z., Ziwei, L., Yu, Q., Tao, X., Change, L.C.: Domain generalization: A survey. arXiv preprint arXiv:2103.02503 (2021)
21. Kecheng, Z., Jiawei, L., Wei, W., Liang, L., Zheng-jun, Z.: Calibrated feature decomposition for generalizable person re-identification. arXiv preprint arXiv:2111.13945 (2021)

22. Lanlan, L., Jia, D.: Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In: AAAI. pp. 3675–3682 (2018)
23. Liang, Z., Liyue, S., Lu, T., Shengjin, W., Jingdong, W., Qi, T.: Scalable person re-identification: A benchmark. In: CVPR. pp. 1116–1124 (2015)
24. Liao, S., Shao, L.: Interpretable and generalizable person re-identification with query-adaptive convolution and temporal lifting. In: ECCV. pp. 456–474 (2020)
25. Lingxiao, H., Wu, L., Jian, L., Kecheng, Z., Xingyu, L., Peng, C., Tao, M.: Semi-supervised domain generalizable person re-identification. arXiv preprint arXiv:2108.05045 (2021)
26. Lingxiao, H., Xingyu, L., Wu, L., Xinchen, L., Peng, C., Tao, M.: Fastreid: A pytorch toolbox for general instance re-identification. arXiv preprint arXiv:2006.02631 (2020)
27. Longhui, W., Shiliang, Z., Wen, G., Qi, T.: Person transfer gan to bridge domain gap for person re-identification. In: CVPR. pp. 79–88 (2018)
28. Lu, Y., Lingqiao, L., Yunlong, W., Peng, W., Yanning, Z.: Multi-domain joint training for person re-identification. arXiv preprint arXiv:2201.01983 (2022)
29. M, C.F., Antonio, D., Silvia, B., Barbara, C., Tatiana, T.: Domain generalization by solving jigsaw puzzles. In: CVPR. pp. 2229–2238 (2019)
30. Mang, Y., Jianbing, S., Gaojie, L., Tao, X., Ling, S., CH, H.S.: Deep learning for person re-identification: A survey and outlook. arXiv preprint arXiv:2001.04193 (2020)
31. Maniyar, U., Joseph, K.J., Deshmukh, A.A., Dogan, Ü., Balasubramanian, V.N.: Zero-shot domain generalization. arXiv preprint arXiv:2008.07443 (2020)
32. Martin, H., Csaba, B., M, R.P., Horst, B.: Person re-identification by descriptive and discriminative classification. In: SCIA. pp. 91–102 (2011)
33. Misha, D., Babak, S., Laurent, D., Marc'Aurelio, R., Nando, D.F.: Predicting parameters in deep learning. In: NeurIPS. pp. 2148–2156 (2013)
34. Muhammad, G., Bastiaan, K.W., Mengjie, Z., David, B.: Domain generalization for object recognition with multi-task autoencoders. In: ICCV. pp. 2551–2559 (2015)
35. Ningning, M., Xiangyu, Z., Jiawei, H., Jian, S.: Weightnet: Revisiting the design space of weight networks. In: ECCV. pp. 776–792 (2020)
36. Riccardo, V., Vittorio, M.: Addressing model vulnerability to distributional shifts over image transformation sets. In: ICCV. pp. 7980–7989 (2019)
37. Seokeon, C., Taekyung, K., Minki, J., Hyoungseob, P., Changick, K.: Meta batch-instance normalization for generalizable person re-identification. In: CVPR. pp. 3425–3435 (2021)
38. Shan, L., Chang-Tsun, L., C, K.A.: Multi-domain adversarial feature generalization for person re-identification. TIP **30**, 1596–1607 (2020)
39. Shankar, S., Piratla, V., Chakrabarti, S., Chaudhuri, S., Jyothi, P., Sarawagi, S.: Generalizing across domains via cross-gradient training. In: ICLR (2018)
40. Shengcai, L., Ling, S.: Graph sampling based deep metric learning for generalizable person re-identification. arXiv preprint arXiv:2104.01546 (2021)
41. Shengcai, L., Ling, S.: Transmatcher: Deep image matching through transformers for generalizable person re-identification. In: NeurIPS (2021)
42. Shijie, Y., Feng, Z., Dapeng, C., Rui, Z., Haobin, C., Shixiang, T., Jinguo, Z., Yu, Q.: Multiple domain experts collaborative learning: Multi-source domain generalization for person re-identification. arXiv preprint arXiv:2105.12355 (2021)
43. Shiyu, X., Shiliang, Z.: Intra-inter camera similarity for unsupervised person re-identification. In: CVPR. pp. 11926–11935 (2021)

44. Shujun, W., Lequan, Y., Caizi, L., Chi-Wing, F., Pheng-Ann, H.: Learning from extrinsic and intrinsic supervisions for domain generalization. In: ECCV. pp. 159–176 (2020)
45. Tong, X., Hongsheng, L., Wanli, O., Xiaogang, W.: Learning deep feature representations with domain guided dropout for person re-identification. In: CVPR. pp. 1249–1258 (2016)
46. Tong, X., Shuang, L., Bochao, W., Liang, L., Xiaogang, W.: End-to-end deep learning for person search. arXiv preprint arXiv:1604.01850 (2016)
47. Wei, L., Rui, Z., Tong, X., Xiaogang, W.: Deepreid: Deep filter pairing neural network for person re-identification. In: CVPR. pp. 152–159 (2014)
48. Wei, L., Xiaogang, W.: Locally aligned feature transforms across views. In: CVPR. pp. 3594–3601 (2013)
49. Wei-Shi, Z., Shaogang, G., Tao, X.: Associating groups of people. In: BMVC. pp. 1–11 (2009)
50. Xin, J., Cuiling, L., Wenjun, Z., Zhibo, C., Li, Z.: Style normalization and restitution for generalizable person re-identification. In: CVPR. pp. 3143–3152 (2020)
51. Xingang, P., Ping, L., Jianping, S., Xiaoou, T.: Two at once: Enhancing learning and generalization capacities via ibn-net. In: ECCV. pp. 464–479 (2018)
52. Xu, J., Bert, D.B., Tinne, T., V, G.L.: Dynamic filter networks pp. 667–675 (2016)
53. Xun, H., Serge, B.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE international conference on computer vision. pp. 1501–1510 (2017)
54. Ya, L., Mingming, G., Xinmei, T., Tongliang, L., Dacheng, T.: Domain generalization via conditional invariant representations. In: AAAI. pp. 3579–3587 (2018)
55. Ya, L., Xinmei, T., Mingming, G., Yajing, L., Tongliang, L., Kun, Z., Dacheng, T.: Deep domain generalization via conditional invariant adversarial networks. In: ECCV. pp. 624–639 (2018)
56. Yan, B., Jile, J., Wang, C., Jun, L., Yihang, L., Xuetao, F., Ling-Yu, D.: Person30k: A dual-meta generalization network for person re-identification. In: CVPR. pp. 2123–2132 (2021)
57. Yi-Fan, Z., Hanlin, Z., Zhang, Z., Da, L., Zhen, J., Liang, W., Tieniu, T.: Learning domain invariant representations for generalizable person re-identification. arXiv preprint arXiv:2103.15890 (2021)
58. Yizeng, H., Gao, H., Shiji, S., Le, Y., Honghui, W., Yulin, W.: Dynamic neural networks: A survey. TPAMI (2021)
59. Yongxing, D., Xiaotong, L., Jun, L., Zekun, T., Ling-Yu, D.: Generalizable person re-identification with relevance-aware mixture of experts. In: CVPR. pp. 16145–16154 (2021)
60. Yuyang, Z., Zhun, Z., Fengxiang, Y., Zhiming, L., Yaojin, L., Shaozi, L., Nicu, S.: Learning to generalize unseen domains via memory-based multi-source meta-learning for person re-identification. In: CVPR. pp. 6277–6286 (2021)
61. Zhang, E., Jiang, X., Cheng, H., Wu, A., Yu, F., Li, K., Guo, X., Zheng, F., Zheng, W., Sun, X.: One for more: Selecting generalizable samples for generalizable reid model. In: AAAI. pp. 3324–3332 (2021)
62. Zhedong, Z., Liang, Z., Yi, Y.: Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In: ICCV. pp. 3754–3762 (2017)
63. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain adaptive ensemble learning. TIP **30**, 8008–8018 (2021)
64. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain generalization with mixstyle. In: ICLR (2021)

65. Zhuang, L., Jianguo, L., Zhiqiang, S., Gao, H., Shoumeng, Y., Changshui, Z.: Learning efficient convolutional networks through network slimming. In: ICCV. pp. 2736–2744 (2017)
66. Zhun, Z., Liang, Z., Donglin, C., Shaozi, L.: Re-ranking person re-identification with k-reciprocal encoding. In: CVPR. pp. 1318–1327 (2017)