IDM: An Intermediate Domain Module for Domain Adaptive Person Re-ID

Yongxing Dai 1 Jun Liu 2 Yifan Sun 3 Zekun Tong 4 Chi Zhang 3 Ling-Yu Duan 1, 5*  

1 Institute of Digital Media (IDM), Peking University, Beijing, China  
2 Singapore University of Technology and Design, Singapore  
3 Megvii Technology  
4 National University of Singapore, Singapore  
5 Peng Cheng Laboratory, Shenzhen, China  
{yongxingdai, lingyu}@pku.edu.cn, jun.liu@sutd.edu.sg, sunyf15@tsinghua.org.cn

Abstract

Unsupervised domain adaptive person re-identification (UDA re-ID) aims at transferring the labeled source domain’s knowledge to improve the model’s discriminability on the unlabeled target domain. From a novel perspective, we argue that the bridging between the source and target domains can be utilized to tackle the UDA re-ID task, and we focus on explicitly modeling appropriate intermediate domains to characterize this bridging. Specifically, we propose an Intermediate Domain Module (IDM) to generate intermediate domains’ representations on-the-fly by mixing the source and target domains’ hidden representations using two domain factors. Based on the “shortest geodesic path” definition, i.e., the intermediate domains along the shortest geodesic path between the two extreme domains can play a better bridging role, we propose two properties that these intermediate domains should satisfy. To ensure these two properties to better characterize appropriate intermediate domains, we enforce the bridge losses on intermediate domains’ prediction space and feature space, and enforce a diversity loss on the two domain factors. The bridge losses aim at guiding the distribution of appropriate intermediate domains to keep the right distance to the source and target domains. The diversity loss serves as a regularization to prevent the generated intermediate domains from being over-fitting to either of the source and target domains. Our proposed method outperforms the state-of-the-arts by a large margin in all the common UDA re-ID tasks, and the mAP gain is up to 7.7% on the challenging MSMT17 benchmark. Code is available at https://github.com/SikaStar/IDM.

1. Introduction

Person re-identification (re-ID) [1, 8, 53] aims to identify the person across the non-overlapped cameras. Though fully supervised re-ID methods [41, 45, 3, 60] have achieved great progress in recent years, it takes much time and effort to label the person images and thus can be difficult to be well applied to some practical scenarios. To overcome this problem, researchers have been focusing on studying unsupervised domain adaptive (UDA) re-ID [48, 13, 38, 2, 9], whose goal is to transfer the knowledge from the labeled source domain to improve the model’s discriminability on the unlabeled target domain. UDA re-ID is a challenging problem because the source and target domains can have two extreme distributions, and there can be no overlap between the two domains’ label space. In this paper, we are devoted to studying the problem of UDA re-ID.

Though many existing methods [48, 13, 16] have made great progress in UDA re-ID, they have not explicitly considered the bridging between the two extreme domains, i.e., what information about the similarity/dissimilarity of the two domains can be utilized to tackle the UDA re-ID task. For example, GAN transferring methods [48, 10] use GANs to translate images’ style across domains and then focus on a single domain’s supervised feature learning. Fine-tuning methods [38, 13, 9] only use the source domain to obtain the pretrained model and then focus on the feature learning in the target domain. Joint training methods [70, 16] just combine the source and target data together for training. From
a novel perspective, we explicitly and comprehensively investi-
gate into the bridging between the source and target do-
main, and argue that using this bridging can help better adapt between two extreme domains in UDA re-ID.

In a spirit of the golden mean, we propose to use the deep model to learn intermediate domains which can bridge the source and target domains and thus ease the UDA re-ID task. The source domain’s knowledge is hard to be directly transferred to the target domain because there can be huge shift between the two extreme domains’ distributions. As shown in Figure 1, assuming the source and target domains are located in a manifold, an appropriate “path” may exist to bridge the two extreme domains along which the source domain’s knowledge can be smoothly transferred to guide the learning of the target domain. Specifically, the appropriate intermediate domains should be located along with this “path” to help the gradual adaptation between the two extreme domains. For example, if an intermediate domain is closer to the source domain, the source reliable labels can be more leveraged. On the contrary, the target domain’s intrinsic distribution can be more exploited.

To model the characteristics of appropriate intermediate domains, we propose a plug-and-play intermediate domain module (IDM). The IDM module can be plugged at any hidden stage of a network, which will generate intermediate domains’ representations on-the-fly to gradually bridge the source and target domains in a joint training framework. Specifically, the IDM module takes some hidden stage’s representations of the source and target domains as the input, and obtains two domain relevance variables that we call domain factors. We use these two domain factors to mix the hidden stage’s representations of both source and target domains, where the mixed representations can represent the characteristics of intermediate domains. All these representations from the source, target and intermediate domains are fed into the network’s next stage on-the-fly.

However, there may be infinite intermediate domains when mixing the source and target domains, and only a part of these domains may be more helpful to smoothly adapt between the two extreme domains (the source and target domains). Inspired by the traditional works [17, 19] that construct intermediate domains using kernel-based methods in a manifold, we first propose the “shortest geodesic path” definition that the appropriate intermediate domains should lie on the shortest geodesic path connecting the source and target domains. From this definition, we further propose two properties where appropriate intermediate domains should satisfy: (1) Keep the right distance to the source and target domains. (2) Be diverse enough to balance the source and target domains’ learning and avoid being over-fitting to either of them. For the first property above, we enforce the bridge losses on both feature and prediction spaces of intermediate domains. For the second property, we propose a diversity loss by maximizing the standard deviation of the domain factors generated by the IDM module. By enforcing these losses on the learning of the IDM module, we can model the characteristics of appropriate intermediate domains to better transfer the source knowledge to improve the model’s discriminability on the target domain.

Our contributions can be summarized as follows. (1) To the best of our knowledge, we are the first to explicitly consider how to utilize the characteristics of intermediate domains as the bridge to better transfer the source knowledge to the target domain in UDA re-ID. Specifically, we propose a plug-and-paly IDM module to generate intermediate domains on-the-fly, which will smoothly bridge the source and target domains to better adapt between the two extremes to ease the UDA re-ID task. (2) To make the IDM module learn more appropriate intermediate domains, we propose two properties of these domains, and design the bridge losses and diversity loss to satisfy these properties. (3) Our method outperforms state-of-the-arts by a large margin on all the common UDA re-ID tasks.

2. Related Work

Unsupervised Domain Adaptation. A line of works convert the UDA into an adversarial learning task [14, 42, 31, 37]. Another line of works use various metrics to measure and minimize the domain discrepancy, such as MMD [30] or other metrics [39, 72, 32, 24]. Another line of traditional works [17, 7, 19] try to bridge the source and target domains based on intermediate domains. In traditional methods [17, 19], they embed the source and target data into a Grassmann manifold and learn a specific geodesic path between the two domains to bridge the source and target domains, but they are not easily applied to the deep models. In deep methods [18, 6], they either use GANs to generate a domain flow by reconstructing input images on pixel level [18] or learn better domain-invariant features by bridging the learning of the generator and discriminator [6]. However, reconstructing images may not guarantee the high-level domain characteristics in the feature space, and learning domain-invariant features may be not suitable for two extreme domains in UDA re-ID. Different from the above methods, we propose a specific IDM module to model the intermediate domains, which can be easily inserted into the existing deep networks. Instead of hard training for GANs or reconstructing images, our IDM module can be learned in an efficient joint training scheme.

Unsupervised Domain Adapative Person Re-ID. In recent years, many UDA re-ID methods have been proposed and they can be mainly categorized into three types based on their training schemes, i.e., GAN transferring [48, 10, 22, 73], fine-tuning [38, 13, 15, 9, 4, 55, 23, 28, 54], and joint training [69, 70, 44, 16, 11]. GAN transferring methods use GANs to transfer images’ style across domains
Figure 2. Illustration of our method. (a) In a joint training way, our proposed IDM module can be plugged after any stage (e.g., stage-1) of ResNet-50, where the source and target domains’ hidden representations (i.e., $G^s$, $G^t$) are mixed with IDM to generate intermediate domains’ representations $G^{\text{inter}}$. GAP means the global average pooling layer. For simplicity, we omit the classifier. The common ReID loss $L_{\text{ReID}}$ (including the classification loss $L_{\text{cls}}$ and triplet loss $L_{\text{tri}}$) are enforced on the source and target domains. To generate more appropriate intermediate domains, we enforce bridge losses ($L_{\text{f bridge}}$ and $L_{\phi bridge}$) on intermediate domains’ $f^{\text{inter}}$ and $\phi^{\text{inter}}$ respectively, and enforce a diversity loss ($L_{\text{div}}$) to regularize the domain factors $[a^s, a^t]$ obtained from the IDM module. (b) The detailed structure of the IDM module. “Avg” and “Max” mean average-pooling and max-pooling operations respectively. (c) Assuming different domains as different points in a manifold. An appropriate intermediate domain $P^{\lambda}_{\text{inter}}$ should be located along with the shortest geodesic path between the source and target domains (i.e., $P_s$ and $P_t$), making $P^{\lambda}_{\text{inter}}$ keep the right distance to $P_s$ and $P_t$.

[48, 10] or disentangle features into id-related/unrelated features [73]. For fine-tuning methods, they first train the model with labeled source data and then fine-tune the pre-trained model on target data with pseudo labels. The key component of these methods is how to alleviate the effects of the noisy pseudo labels. However, these methods ignore the labeled source data while fine-tuning on the target data, which will hinder the domain adaptation process because of the catastrophic forgetting in networks. For joint training methods, they combine the source and target data together and train on an ImageNet-pretrained network from scratch. All these joint training methods often utilize the memory bank [51, 50] to improve target domain features’ discriminability. However, these methods just take both the source and target data as the network’s input and train jointly while neglecting the bridging between both domains, i.e., what information of the two domains’ dissimilarities/similarities can be utilized to improve features’ discriminability in UDA re-ID. Different from all the above UDA re-ID methods, we propose to consider the bridging between the source and target domains by modeling appropriate intermediate domains with a plug-and-paly module, which is helpful for gradually adapting between two extreme domains in UDA re-ID.

**Mixup and Variants.** Mixup [56] is an effective regularization technique to improve the generalization of deep networks by linearly interpolating the image and label pairs, where the interpolating weights are randomly sampled from a Dirichlet distribution. Manifold Mixup [43] extends Mixup to a more general form which can linearly interpolate data at the feature level. Recently, Mixup has been applied to many tasks like point cloud classification [5], object detection [59], and closed-set domain adaptation [52, 49, 34]. Our work differs from these Mixup variants in: (1) All the above methods take Mixup as a data/feature augmentation technique to improve models’ generalization, while we bridge two extreme domains by generating intermediate domains for UDA re-ID. (2) We design an IDM module and enforce specific losses on it to control the bridging process while all the above methods often linearly interpolate data using the random interpolation ratio without constraints.
3. Proposed Method

The purpose of our method is to generate appropriate intermediate domains to bridge the source and target domains. By training the source, target, and intermediate domains together at the same time, the network will adaptively characterize the distributions of the source and target domains. This will smoothly adapt between the two extreme domains and better transfer the source knowledge to improve the model’s performance on the target domain.

3.1. Overview

In UDA re-ID, we are often given a labeled source domain dataset \( \{ (x^s_i, y^s_i) \} \) and an unlabeled target domain dataset \( \{ x^t_i \} \). The source dataset contains \( N^s \) labeled person images, and the target dataset contains \( N^t \) unlabeled images. Each source image \( x^s_i \) is associated with a person identity \( y^s_i \) and the total number of source domain identities is \( C^s \). We use ResNet-50 [20] as the backbone network \( f(\cdot) \) and add a hybrid classifier \( \varphi(\cdot) \) after the global average pooling (GAP) layer, where the hybrid classifier is comprised of the batch normalization layer and a \( C^s + C^t \) dimensional fully connected (FC) layer followed by a softmax activation function. \( C^t \) is the cluster number when performing DBSCAN clustering on the target domain features at the beginning of every training epoch like existing clustering-based UDA re-ID methods [13, 38, 16]. After clustering, we assign the pseudo labels \( y^t_i \) for the target data \( x^t_i \).

Figure 2 (a) shows the pipeline of our method. The overall training scheme is jointly training on the source and target data. A mini-batch including \( n \) source samples and \( n \) target samples is fed to the network. Our proposed Intermediate Domain Module (IDM) can be plugged after any stage of the deep network like ResNet-50. As shown in Figure 2 (a), the IDM module is plugged between the stage-1 and stage-2, where the source and target domains’ hidden representations are mixed with the IDM module to generate new intermediate domains’ representations. Next, all the representations of the source, target, and intermediate domains are fed into the stage-2 until the end of the network. We enforce ReID loss \( L_{reid} \) (including the classification loss [65] and triplet loss [21]) on the source and target domains’ features \( (f^s, f^t) \) and predictions \( (\varphi^s, \varphi^t) \). To make the IDM module generate more appropriate intermediate domain representations, we propose two bridge losses and a diversity loss. The bridge losses \( L^t_{bridge}, L^s_{bridge} \) are respectively enforced on intermediate domains’ features \( f^{\text{inter}} \) and predictions \( \varphi^{\text{inter}} \). The diversity loss \( L_{div} \) is used to enforce on the domain factors obtained from the IDM module, aiming to prevent the module from being overfitting to either of two extreme domains. The IDM module is only used for training and will be discarded in testing.

3.2. Intermediate Domain Module

Our proposed IDM module can be plugged after the hidden stage in the backbone network. The module takes both source and target hidden stage’s representations as the input and generate two domain factors \( i.e., a^s \) and \( a^t \). With these two domain factors, we can mix the source and target domains’ hidden representations to generate intermediate domains’ representations on-the-fly.

Specifically, we denote backbone network as \( f(x) = f_m(g_m(x)) \), where \( g_m \) represents the part of the network mapping the input data \( x \) to the hidden representation \( g_m(x) \in \mathbb{R}^{h \times w \times c} \) (i.e., feature map) after the \( m \)-th stage and \( f_m \) represents the part of the network mapping the hidden representation \( g_m(x) \) to the 2048-dim feature after the GAP layer. As shown in Figure 2 (b), the IDM module is composed of a fully connected layer FC1, and a MLP (Multi-Layer Perception) followed by a softmax function.

In a mini-batch including \( n \) source and \( n \) target samples, we randomly combine two domains’ samples into \( n \) pairs. For each sample pair \( (x^s, x^t) \), we can obtain their feature maps at the \( m \)-th hidden stage: \( G^s, G^t \in \mathbb{R}^{h \times w \times c} \), both of which are performed with the average-pooling and max-pooling operations respectively, generating the \( 1 \times 1 \times c \) dimensional features \( i.e., (G^s_{\text{avg}}, G^s_{\text{max}}) \) for the source domain, and \( (G^t_{\text{avg}}, G^t_{\text{max}}) \) for the target domain. We concatenate the \( \text{avg} \) and \( \text{max} \) features for each domain and feed them to FC1. The output feature vectors of FC1 are then merged using element-wise summation and fed into MLP to obtain a domain factor vector \( a \in \mathbb{R}^2 \). The procedure to obtain the domain factors in Figure 2 (b) is as follows:

\[
 a = \delta(\text{MLP}(FC1((G^s_{\text{avg}}, G^s_{\text{max}})) + FC1((G^t_{\text{avg}}, G^t_{\text{max}}))))
\]

(1)

where \( \delta(\cdot) \) is the softmax function. Next, we define the operation of generating intermediate domain representations after the \( m \)-th hidden stage by mixing the source and target representations with two domain factors as follows:

\[
 G^{\text{inter}} = a^s \cdot G^s + a^t \cdot G^t,
\]

(2)

where \( G^s \) and \( G^t \) are the \( m \)-th hidden stage’s representations, and \( a^s \) and \( a^t \) are the two domain factors for the source and target domains respectively (\( [a^s, a^t] = a \)).

3.3. Modelling Appropriate Intermediate Domains

We can generate infinite intermediate domains by the IDM module in a mixing operation as Eq. (2). However, only a part of these intermediate domains can be appropriate to bridge the source and target domains [19]. Inspired by the traditional works [17, 19] that construct intermediate domains using kernel-based methods, all domains can be assumed to be located in a manifold. Only those intermediate domains located along the shortest geodesic path between the source and target domains can be helpful for the
smooth adaptation procedure [19]. Based on this “shortest geodesic path” definition, we argue that appropriate intermediate domains should satisfy two properties and further propose specifically-designed losses to ensure the two properties in the concept of UDA re-ID. We denote distributions of source, target, and intermediate domains as $P_s, P_t,$ and $P_{\text{inter}}$, and use $d(\cdot)$ to represent geodesic distance in a manifold. As shown in Figure 2 (c), different domains can be seen as different points in a manifold.

**Definition (Shortest geodesic path):** Appropriate intermediate domains should be located onto the shortest geodesic path between the source and target domains, i.e., $d(P_s, P_{\text{inter}}) + d(P_t, P_{\text{inter}}) = d(P_s, P_t)$.

Intuitively, for an intermediate domain $P'_{\text{inter}}$ not on the shortest geodesic path, the triangle inequality ensures that $d(P_s, P_{\text{inter}}') + d(P_t, P_{\text{inter}}') > d(P_s, P_t)$. As in UDA re-ID, our goal is to reduce the distribution shift $d(P_s, P_t)$ between the source and target domains and the inappropirate intermediate domain $P_{\text{inter}}$ can bring about “extra domain shifts” (i.e., the above inequality) that will degenerate the adaptation procedure.

**Property 1 (Distance should be proportional):** We use a variable $\lambda \in [0, 1]$ to control the distance between the source domain to any intermediate domain $P_{\text{inter}}(\lambda)$ on the shortest geodesic path, i.e., $d(P_s, P_{\text{inter}}(\lambda)) = \lambda \cdot d(P_s, P_t)$. The distances satisfy the proportional relationship:

$$
\frac{d(P_s, P_{\text{inter}}(\lambda))}{d(P_t, P_{\text{inter}}(\lambda))} = \frac{\lambda}{1 - \lambda}.
$$

**Sketch of Proof:** From the shortest geodesic path definition, we can derive $d(P_s, P'_{\text{inter}}) + d(P_t, P'_{\text{inter}}) = d(P_s, P_t)$. Then from $d(P_s, P_{\text{inter}}(\lambda)) = \lambda \cdot d(P_s, P_t)$, we can get the proportional relationship as Eq. (3).

As shown in Eq. (3), if $\lambda = 0$ (or $\lambda=1$), the intermediate domain $P_{\text{inter}}(\lambda)$ is identical to the source (or target) domain $P_s$ (or $P_t$). In our method, we use the domain factors $a$ to model the proportional relationship. As shown in Eq. (2), the generated intermediate domains’ representations depend on two domain factors: $a_s$ and $a_t$, where $a_s, a_t \in [0, 1]$ and $a_s + a_t = 1$ as we use the softmax function $\delta$ in Eq. (1). The domain factors $a$ can be seen as the relevance of the intermediate domain to the other two extreme domains. Thus, the distance relationship (contrary to the relevance relationship) between intermediate domains and other two domains can be formulated as follows:

$$
\frac{d(P_s, P_{\text{inter}}(a))}{d(P_t, P_{\text{inter}}(a))} = \frac{a^t}{a^s} = \frac{1 - a^s}{a^s} = \frac{a^t}{1 - a^s},
$$

where $P_{\text{inter}}(a)$ is the intermediate domain controlled by the domain factors $a$. When $a^s$ gets close to 1, the relevance between $P_{\text{inter}}(a)$ and $P_s$ becomes larger while the distance $d(P_s, P_{\text{inter}}(a))$ becomes smaller.

Based on the above analysis, generating appropriate intermediate domains with domain factors $a$ becomes finding the points located along the shortest path that should be closest to both $P_s$ and $P_t$, and satisfy Eq. (4). Thus, the problem can be converted into minimizing the loss as:

$$
L_{\text{bridge}} = a^s \cdot d(P_s, P_{\text{inter}}(a)) + a^t \cdot d(P_t, P_{\text{inter}}(a)).
$$

If $a^s > a^t$, $L_{\text{bridge}}$ will penalize more on $d(P_s, P_{\text{inter}}(a))$, otherwise it will penalize more on $d(P_t, P_{\text{inter}}(a))$. Intermediate domains can utilize domain factors $a$ to adaptively balance the minimization of distribution shifts between the source and target domains, resulting in a smooth domain adaptation procedure.

In a deep model for UDA re-ID, we consider to enforce the bridge loss (Eq. (5)) on both the prediction and feature spaces of intermediate domains. For the prediction space, we use the cross-entropy to measure the distribution gap between intermediate domains’ prediction logits and other two extreme domains’ (pseudo) labels (Eq. (6)). For the feature space, we use the L2-norm to measure features’ distance among domains (Eq. (7)). Our proposed two bridge losses are formulated as follows:

$$
L_{\text{bridge}} = \frac{1}{n} \sum_{i=1}^{n} \sum_{k \in \{s, t\}} a_i^k \cdot \left[ \frac{1}{y_i} \log (\phi(G_{\text{inter}}^k)) \right],
$$

$$
L_{\text{bridge}}^i = \frac{1}{n} \sum_{i=1}^{n} \sum_{k \in \{s, t\}} a_i^k \cdot \left\| f_m(G_{\text{inter}}^k) - f_m(G_{\text{inter}}^i) \right\|_2.
$$

In Eq. (6) (7), we use $k$ to indicate the domain (source or target) and use $i$ to index the data in a mini-batch. $G_{\text{inter}}^i$ is the $k$ domain’s representation at the $m$-th stage. $G_{\text{inter}}^i$ is the intermediate domain’s representation at the $m$-th hidden stage by mixing $G_{\text{inter}}^i$ and $G_s$ as in Eq. (2). The $f_m(\cdot)$ is the mapping from the $m$-th stage to the features after GAP layer and $\phi(\cdot)$ is the classifier.

**Property 2 (Diversity):** Intermediate domains should be as diverse as possible.

**Sketch of Proof:** Given any intermediate domain $P_{\text{inter}}(\lambda)$ along the shortest geodesic path from Property 1, the intermediate domain will be mainly dominated by the source (or target) domain if $\lambda \in [0, 0.5]$ (or $\lambda \in (0.5, 1]$). Neither conditions can well bridge the source and target domains, i.e., the shortest geodesic path will be cut off.

To ensure the above property, we propose the diversity loss to enforce on the domain factors $a$ by maximizing their differences within a mini-batch. This loss is as follows:

$$
L_{\text{div}} = -\sigma(a_{i,1}^{t} - a_{i}^{t}) - \sigma(a_{i,1}^{s} - a_{i}^{s}),
$$

where $\sigma(\cdot)$ means calculating the standard deviation of the values in a mini-batch. By minimizing $L_{\text{div}}$, we can enforce intermediate domains to be as diverse as enough to model the characteristics of the “shortest geodesic path”, which can better bridge the source and target domains.
In conclusion, we use \( L \) and \( L \) to enforce generated intermediate domains to be located along the “shortest geodesic path” and use \( L \) to characterize more intermediate domains, which will effectively build a bridge between the two extreme domains. This bridge can adaptively balance the adaptation between the source and target domains when training our network over both the source and target data, i.e., the model can gradually minimize the discrepancy between the two extreme distributions.

### 3.4. Overall Training

The ResNet-50 backbone contains five stages, where stage-0 is comprised of the first Conv, BN and Max Pooling layer and stage-1/2/3/4 correspond to the other four convolutional blocks. We plug our proposed IDM module after the \( m \)-th stage of ResNet-50 (i.e., stage-0/1/2/3/4 as shown in Figure 2). The overall training loss is as follows:

\[
L = L_{\text{ReID}} + \mu_1 \cdot L_{\text{bridge}}^e + \mu_2 \cdot L_{\text{bridge}}^f + \mu_3 \cdot L_{\text{div}},
\]

where \( L_{\text{ReID}} = (1 - \mu_1) \cdot L_{\text{cls}} + L_{\text{tri}}, \) and \( \mu_1, \mu_2, \mu_3 \) are the weights to balance losses. The training and testing procedure is shown in Algorithm 1 in the supplementary material.

### 4. Experiments

Five datasets including Market-1501 [64], DukeMTMC-reID [36, 66], MSMT17 [48], PersonX [40], and Unreal [57] are used in our experiments. We use mean average precision (mAP) and Rank-1/5/10 (R1/5/10) of CMC to evaluate performances. In training, we do not use any additional information like temporal consistency in JVTC+ [25]. In testing, there are no post-processing techniques like re-ranking [67] or multi-query fusion [64]. Implementation details can be seen in the supplementary material.

#### 4.1. Ablation Study

In this section, we will evaluate the effectiveness of different components of our method. The “Oracle” method in Table 1 means we use the target domain’s ground-truth to train together with the source domain, which can be seen as the upper bound of the performance for UDA re-ID in a joint training pipeline.

**Effectiveness of our proposed IDM module on different baseline methods.** We propose two baseline methods including Naive Baseline (Baseline1) and Strong Baseline (Baseline2). Naive Baseline means only using \( L_{\text{ReID}} \) to train the source and target data in joint manner as shown in Figure 2. Compared with Naive Baseline, Strong Baseline uses XBM [47] to mine more hard negatives for the triplet loss, which is a variant of the memory bank [51, 50] and easy to implement. Similar to those UDA re-ID methods [70, 16] using the memory bank, we set the memory bank size as the number of all the training data. As shown in Table 1, no matter which baseline we use, our methods can obviously outperform the baseline methods by a large margin. Take Market→Duke as an example, mAP/R1 of Baseline2+our IDM (full) is 4.7%/3.5% higher than Baseline2, and mAP/R1 of Baseline1+our IDM (full) is 5.2%/3.9% higher than Baseline1. Because of many state-of-the-arts methods [16, 62, 25, 63] use the memory bank to improve the performance on the target domain, we use Strong Baseline to implement our IDM for fairly comparing with them.

**Effectiveness of the bridge losses for the prediction and feature spaces.** The bridge losses are enforced on both the prediction and feature spaces of intermediate domains. With both losses, we can enforce the generated intermediate domains to be located onto the geodesic path between the source and target domains in a manifold, which are more appropriate to bridge domains to perform more smooth domain adaptation. Take Market→Duke as an example, mAP/R1 of Baseline2+IDM w/o \( L \) is 3.2%/1.8% lower than Baseline2+IDM, and mAP/R1 of Baseline2+IDM w/o \( L \) is 1.5%/1.6% lower than Baseline2+IDM. From Table 1, the method w/o \( L \) is superior to that w/o \( L \) for Baseline2, while the conclusion is just opposite for Baseline1. The reason may be that Baseline2 uses XBM to mine harder negatives in the feature space, which will affect the effectiveness of \( L \).

**Effectiveness of the diversity loss.** We use the diversity loss \( L \) to prevent the IDM module from being over-fitting to either of the source and target domains when generating intermediate domains. Baseline2+IDM w/o \( L \) degenerates much when comparing with the full method. Take Market→Duke as an example (Table 1), mAP of Strong Baseline+IDM w/o \( L \) is 2.7% lower than the full method.

**Effectiveness on which stage to plug the IDM module.** Our IDM module is a plug-and-play module which can be plugged after any stage of the backbone network. In our experiments, we use ResNet-50 as the backbone, which has five stages: stage-0 is comprised of the first Conv, BN...
Table 2. Study about which stage of ResNet-50 to plug the IDM module.

<table>
<thead>
<tr>
<th>Method</th>
<th>Duke → Market</th>
<th>Market → Duke</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mAP</td>
<td>R1</td>
</tr>
<tr>
<td>Strong Baseline</td>
<td>79.1</td>
<td>91.2</td>
</tr>
<tr>
<td>IDM after stage-0</td>
<td>82.8</td>
<td>93.2</td>
</tr>
<tr>
<td>IDM after stage-1</td>
<td>82.3</td>
<td>92.2</td>
</tr>
<tr>
<td>IDM after stage-2</td>
<td>81.9</td>
<td>92.0</td>
</tr>
<tr>
<td>IDM after stage-3</td>
<td>81.3</td>
<td>91.8</td>
</tr>
<tr>
<td>IDM after stage-4</td>
<td>80.2</td>
<td>91.3</td>
</tr>
</tbody>
</table>

Table 3. Comparison with different backbones in our method.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>w/ ResNet-50</th>
<th>w/ IBN-ResNet-50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>mAP</td>
<td>R1</td>
</tr>
<tr>
<td>Duke</td>
<td>Market</td>
<td>82.8</td>
<td>93.2</td>
</tr>
<tr>
<td>Market</td>
<td>Duke</td>
<td>70.5</td>
<td>83.6</td>
</tr>
<tr>
<td>Duke</td>
<td>MSMT</td>
<td>35.4</td>
<td>63.6</td>
</tr>
<tr>
<td>Market</td>
<td>MSMT</td>
<td>33.5</td>
<td>61.3</td>
</tr>
</tbody>
</table>

and and Max Pooling layer and stage-1/2/3/4 correspond to the other four convolutional blocks. We plug our IDM module after different stages to study how different stages will affect the performance of the IDM module. No matter after which stage, our method can outperform the strong baseline. As shown in Table 2, the performance of the IDM module is gradually declining with the deepening of the network. This phenomenon shows that the domain gap becomes larger and the transferable ability becomes weaker at higher/deeper layers of the network, which satisfies the theory of the domain adaptation [29]. Based on the above analyses, we plug the IDM module after stage-0 of the backbone in all our experiments if not specified.

Ours vs. traditional mixup methods. We compare our method with the traditional Mixup [56] and Manifold Mixup (M-Mixup) [43] methods in Figure 3. We use Mixup to randomly mix the source and target domains at the image-level, and use M-Mixup to randomly mix the two domains at the feature-level. The interpolation ratio in Mixup and M-Mixup is randomly sampled from a beta distribution Beta(α, α). Specifically, Mixup and M-Mixup are only the image/feature augmentation technology and we use them to randomly mix the source and target domains. Different from them, our method can adaptively bridge the source and target domains to ensure the two properties that appropriate intermediate domains should satisfy. Thus, our method is superior to these mixup methods in UDA re-ID.

Scalability for different backbones. Our proposed IDM module can be easily scalable to other backbones, such as IBN-ResNet-50 [35]. As shown in Table 3, the performance gain is especially obvious on the largest and challenging MSMT dataset. In addition to IBN [35], our method can also be easily implemented with other normalization techniques like AdaBN [26] and CBN [71].

4.2. Comparison with the State-of-the-arts

The existing state-of-the-arts (SOTA) UDA re-ID works commonly evaluate the performance on four real → real tasks [13, 70]. Recently, more challenging synthetic → real tasks [16, 57] are proposed, where they use the synthetic dataset PersonX [40] or Unreal [57] as the source domain and test on other three real re-ID datasets. Using ResNet-50 as the backbone, all the results in Table 4, 5 show that our method can outperform the SOTAs significantly.

Comparisons on real → real UDA re-ID tasks. The existing UDA re-ID methods evaluated on real → real UDA tasks can be mainly divided into three categories based on their training schemes. (1) GAN transferring methods include PTGAN [48], SPGAN+LMP [10], [68], and PDA-Net [27]. (2) Fine-tuning methods include PUL [12], PCB-PAST [58], SSG [13], AD-Cluster [54], MMT [15], NRMT [61], MEB-Net [55], Dual-Refinement [9], UNRN [62], and GLT [63]. (3) Joint training methods commonly use the memory bank [51], including ECN [69], MMCL [44], ECN-GPP [70], JVTC+ [25], and SpCL [16]. However, all these methods neglect the significance of intermediate domains, which can smoothly bridge the domain adaptation between the source and target domains to better transfer the source knowledge to the target domain. Instead, we propose an IDM module to generate the appropriate intermediate domains to better improve the performance of UDA re-ID. As shown in Table 4, our method can outperform the second best UDA re-ID methods by a large margin on all these benchmarks.

Comparisons on synthetic → real UDA re-ID tasks. Compared with the real → real UDA re-ID tasks, the synthetic → real UDA tasks are more challenging because the domain gap between the synthetic and real images are often larger than that between the real and real images. As shown in Table 5, our method can outperform the SOTA methods by a large margin where mAP of our method is higher than SpCL [16] by 7.5%, 1.3%, and 7.6% when testing on Market-1501, DukeMTMC-reID, and MSMT17 respectively. All these significant results of our method have shown the superiority of our proposed IDM module that can generate the appropriate intermediate domains to better improve the performance on UDA re-ID.
5. Conclusion

This paper proposes a plug-and-play Intermediate Domain Module (IDM) to tackle the problem of UDA re-ID. From a novel perspective that intermediate domains can bridge the source and target domains, our proposed IDM module can generate appropriate intermediate domain representations to better transfer the source knowledge to improve the model’s discriminability on the target domain. The intermediate domains’ distribution is controlled by the two domain factors generated by the IDM module. Specifically, we propose the bridge losses to enforce the intermediate domains to be located onto the appropriate path between the source and target domains in a manifold. Besides, we also propose a diversity loss to constrain the domain factors to prevent the intermediate domains from being over-fitting to either of the source and target domains. Extensive experiments have shown the effectiveness of our method. 

Acknowledgements: This work was supported by the National Natural Science Foundation of China under Grant 62088102, and in part by the PKU-NTU Joint Research Institute (JRI) sponsored by a donation from the Ng Teng Fong Charitable Foundation, and was partially supported by SUTD Project PIE-SGP-AL2020-02.
References


[38] Liangchen Song, Cheng Wang, Lefei Zhang, Bo Du, Qian Zhang, Chang Huang, and Xinggang Wang. Unsupervised domain adaptive re-identification: Theory and practice. Pattern Recognition, 102:107173, 2020. 1, 2, 4


[55] Yunpeng Zhai, Qixiang Ye, Shijian Lu, Mengxi Jia, Rongrong Ji, and Yonghong Tian. Multiple expert brainstorming for domain adaptive person re-identification. ECCV, 2020. 2, 7, 8


[58] Xinyu Zhang, Jiawei Cao, Chunhua Shen, and Mingyu You. Self-training with progressive augmentation for unsupervised cross-domain person re-identification. In ICCV, pages 8222–8231, 2019. 7, 8


