Convolutional Feature Transfer via Camera-Specific Discriminative Pooling for Person Re-Identification

Tetsu Matsukawa, Einoshin Suzuki
Faculty of Information Science and Electrical Engineering (ISEE), Kyushu University, Japan
Email: {matsukawa, suzuki}@inf.kyushu-u.ac.jp

***Abstract*—Modern Convolutional Neural Networks (CNNs) have been improving the accuracy of person re-identification (re-id) using a large number of training samples. Such a re-id system suffers from a lack of training samples for deployment to practical security applications. To address this problem, we focus on the approach that transfers features of a CNN pre-trained on a large-scale person re-id dataset to a small-scale dataset. Most of the existing CNN feature transfer methods use the features of fully connected layers that entangle locally pooled features of different spatial locations on an image. Unfortunately, due to the difference of view angles and the bias of walking directions of the persons, each camera view in a dataset has a unique spatial property in the person image, which reduces the generality of the local pooling for different cameras/datasets. To account for the camera- and dataset-specific spatial bias, we propose a method to learn camera and dataset-specific position weight maps for discriminative local pooling of convolutional features. Our experiments on four public datasets confirm the effectiveness of the proposed feature transfer with a small number of training samples in the target datasets.**

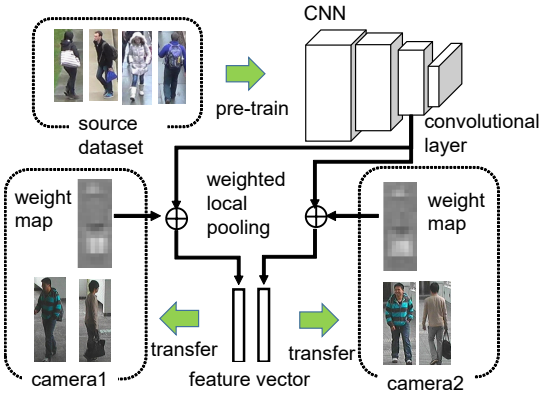


Fig. 1. **Camera-specific Discriminative Pooling of Convolutional Features (C-DPCF)** transfers convolutional features of a backbone model trained on a large training dataset to each of the camera views of a target dataset. We conduct weighted local pooling of convolutional features with camera-specific weight maps.

I. Introduction

Person re-identification (re-id) is a surveillance task that matches a person image captured in a camera view from a large number of gallery images in disjoint camera views [1]. The sensitivity of imaging conditions, *e.g.,* variations in illumination, viewpoint, pose changes makes the task challenging. Also, similar persons in appearance add difficulty.

Traditionally, person re-id is considered as a matching task of extracted features from person images. Supervised distance metric learning trains robust and discriminative distance metric of the features from labeled training samples, which improve the accuracy of person re-id [2], [3], [4], [5].

Recently, Convolutional Neural Networks (CNNs) have shown remarkably high performance on large-scale datasets [6], [7], [8], [9], [10]. CNNs can be regarded as a feature extractor which is trained with the supervision of labeled training samples. Deep CNNs rely on millions of parameters, and they inevitably require a considerable amount of training samples compared with the metric learning approach. Typically, CNN-based methods on person re-id use more than ten thousand training images, which include more than a half thousand person IDs.

For deploying a person re-id system in practical security applications where users frequently install camera systems to new locations, the requirement of a training dataset which contains plenty of training person IDs is a severe bottleneck. For instance, one may install a camera system temporally in a limited period for a special event in the countryside where the traffic of people would be heavy on the event day, whereas light during the preparation period. In the preparation period, only a limited number of different person IDs would be observed in the camera system. When the frame rates of cameras are low, the number of captured images from a person is also limited.

Several recent works focused on learning domain generalizable person re-id models such that users can directly apply CNN models to different datasets without any model update [11], [10]. The unnecessity of the training samples in a target dataset is a merit of these works. Paradoxically, they do not consider the characteristics of the target dataset, which are still available from a small number of training samples.

A natural choice to exploit a small number of training samples in a target dataset would be to use them for conducting fine-tuning of a CNN model pre-trained on a large-scale dataset. Nevertheless, with the limited number of training samples in the target dataset, the model fine-tuning has a potential risk of overfitting. Moreover, the fine-tuning of a deep CNN model demands a high-end GPU with large memory, which increases the deployment cost.

CNN feature transfer is an approach which circumvents the end-to-end training on a small number of training samples in the target dataset. The approach re-uses the extracted features from the trained CNN on a large-scale dataset [12], [13]. In person re-id, metric learning is applied to the penultimate

layer of CNNs [14], which is typically a fully connected layer that entangles locally pooled convolutional features of different local parts.

Unfortunately, different person re-id datasets have spatial bias caused by the change of camera viewing angles and cropping methods of person bounding boxes. Also, captured images in each camera view in the same dataset have a biased trend due to variations in camera angle to persons' walking directions. For example, a camera which captures an underground passageway from a side view from a person walking directions would tend to capture persons from side views. In contrast, a camera which captures a front view of a building-entrance likely takes persons from the front and rear views. Consequently, features of the fully connected layers are less transferrable to different datasets and camera views.

To account for the camera and dataset-specific spatial bias, we propose a method to learn Camera-specific position weight maps for Discriminative Pooling of Convolutional Features (C-DPCF), shown in Fig. 1. Because the convolution operation has translation-invariance, we presume that convolutional features are relatively generalizable for different cameras and datasets if we conduct local pooling operation appropriately by considering the spatial bias.

We summarize the contributions of this paper below:

1) Proposal for the convolutional feature transfer using discriminative spatial pooling, which requires no end-to-end re-training of a deep CNN model on the target dataset.
2) Proposal for camera-specific spatial weight maps to handle spatial bias of different camera views.
3) Development of an efficient training algorithm of the discriminative weight maps for high-dimensional convolutional features.
4) Evaluation of CNN feature transfer with a small number of training person IDs. To the best of our knowledge, this paper is one of the initial works which targets at the improvement of CNN models in such a situation.

II. Related Works

A. CNN feature transfer in person re-id

By using metric learning for feature transfer [14], [15] or directly using cosine similarity of extracted features [16], several works showed the effectiveness of CNN features pre-trained on large-scale auxiliary datasets for different person re-id datasets. These approaches typically use fully connected layers for feature extraction, which entangles the convolutional features of different spatial locations in an image. Thus, their transferrable ability to different datasets is limited.

A work on one-shot metric learning [15] focused on color bias problem in CNN feature transfer. It jointly transfers CNN features trained on gray-scale images with color histograms to target cameras using one-positive image pair and unlabeled samples. A color checker is utilized to account for the color bias problem. Because the method transferred features of a fully-connected layer, they remained the spatial bias problem in the CNN features unaddressed.

Histogram of Intensity Pattern and Histogram of Ordinal Patterns (HIPHOP) extracts features from lower convolutional layers of a pre-trained CNN with a spatial weight map [17]. Nevertheless, the weight map used in the HIPHOP was fixed and pre-determined. Besides, the used CNN for feature maps was limited to a pre-trained CNN on the ImageNet dataset, which is known ineffective for person re-id [14].

Camera-coRrelation Aware Feature AugmenTation (CRAFT) learns camera-specific feature transformation of HIPHOP by augmenting feature vectors [17]. The feature augmentation enables CRAFT to learn camera-specific feature weights, but such camera-specific weights are acquired for already pooled features. Thus, they remain the spatial bias in the convolutional feature map unaddressed. Also, augmenting feature vectors is likely to have a high risk of overfitting due to the high dimensionality of the pooled feature vector. Thus, CRAFT requires regularization to make the weights of different camera views similar; it requires an additional parameter setting on the target dataset.

C-DPCF inherits the formulation of feature augmentation of CRAFT to handle the camera-specific spatial bias problem. In contrast to CRAFT, C-DPCF only learns bias in spatial locations, of which training parameters are smaller than that of the feature augmentation of the pooled feature vectors. Thus, C-DPCF has a lower overfitting risk without using the view-consistency regularization.

B. Discriminative spatial pooling in person re-id

Based on the properties of person images that they tend to be well aligned in a vertical direction and include drastic changes in the horizontal direction, traditional hand-crafted descriptors summarize low-level features, such as color and textural patterns, into horizontal stripes [2], [3].

The latest CNN-based person re-id works adopt horizontal stripe pooling of convolutional features to establish strong baseline models [6], [7], [8], [9], [18]. Most of the models extract features of the penultimate layer, whereas several works report that discriminative low-level features exist on mid to lower layers [18].

For discriminative pooling of hand-crafted local features, Discriminative Accumulation of Local Features (DALF) [19] jointly learns weight maps to summarize local features into multiple local regions and distance metric of the pooled local features. Thereby DALF preserves spatial information in a vertical direction. Meanwhile, DALF generates multiple discriminative weight maps, which capture diverse discriminative spatial information for local features to improve re-id accuracy.

Nevertheless, DALF requires a considerable amount of computational cost in its joint learning process of several pairs of a weight map and a distance metric. The cost increases as feature dimensions, which is too expensive to handle high dimensional feature channels in convolutional layers. Moreover, all camera views share the same learned weight maps, limiting the adaptive ability to substantial spatial bias in different camera views. C-DPCF inherits the advantage of DALF and addresses these disadvantages and limitations.

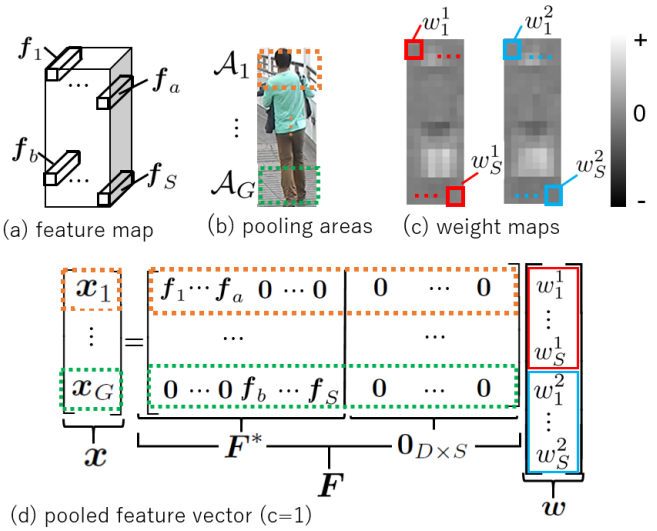


Fig. 2. Camera-specific local weighted average pooling

III. Camera-specific Discriminative Pooling of Convolutional Features

C-DPCF handles a practical scenario where users would use the largest dataset to train a deep CNN model and transfer it to a new camera location with a small number of training samples. We first conduct the CNN model training on the largest dataset that is available. We then transfer the trained CNN model to each of the camera views of the target dataset via weighted local pooling with camera-specific weight maps (see Fig.1). More specifically, we formulate the weighted pooling as a linear form of a convolutional feature and position weight maps (Sec. III-A). We then efficiently train the discriminative weight maps using labeled training samples in the target dataset (Sec. III-B). We further learn distance metrics for the pooled features to also improve the discriminative ability (Sec. III-C). Finally, we apply the weight map and distance metric learnings for several convolutional layers (Sec. III-D).

A. Camera-specific weighted local average pooling

A linear form of a weight local average pooling is a core component of our weight map learning algorithm. A convolutional layer is a feature map of the size $D' \times H \times W$, where D' is the feature channel dimension and H, W are height and width, respectively. Let $\boldsymbol{f}_s \in \mathbb{R}^{D'}$ be the feature vector of the s-th position (Fig. 2 (a)). For convenience, we define a feature matrix $\boldsymbol{F}^o \in \mathbb{R}^{D' \times S} = [\boldsymbol{f}_1, ..., \boldsymbol{f}_S]$, where $S = HW$. We summarize the features vectors within G local pooling areas $\{\mathcal{A}_g\}_{g=1}^G$. For simplicity, we use horizontal stripes that have no overlap with each other as the areas (Fig. 2 (b)). Let us assume that there are C cameras in total and denote the camera ID of the input sample as $c \in \{1, ..., C\}$, which is known at both training and test times. We define a weight map vector for camera c as $\boldsymbol{w}^c \in \mathbb{R}^S = [w_1^c, ..., w_S^c]$, where w_s^c is a position weight of the s-th position (Fig. 2 (c)).

1) Local weighted average pooling: A feature vector $\boldsymbol{x}_g \in \mathbb{R}^{D'}$ produced by a weighted local average pooling in the g-th local pooling area $\mathcal{A}_g$ is given by

$$\boldsymbol{x}_g = \sum_{s \in \mathcal{A}_g} w_s^c \boldsymbol{f}_s = \sum_{s=1}^{S} a_{g,s} (w_s^c \boldsymbol{f}_s) = \boldsymbol{F}_g^* \boldsymbol{w}^c, \tag{1}$$

where $a_{g,s}$ is a binary scalar. We set it as $a_{g,s} = 1$ if $s \in \mathcal{A}_g$ and $a_{g,s} = 0$ otherwise. The matrix $\boldsymbol{F}_g^* \in \mathbb{R}^{D' \times S}$ is given by

$$\begin{aligned} \boldsymbol{F}_g^* &= \left[\begin{array}{ccc} a_{g,1}\boldsymbol{f}_1 & \cdots & a_{g,S}\boldsymbol{f}_S \end{array} \right] \\ &= \left[\begin{array}{ccc} a_{g,1}\boldsymbol{1}_{D'} & \cdots & a_{g,S}\boldsymbol{1}_{D'} \end{array} \right] \odot \boldsymbol{F}^o \\ &= (\boldsymbol{a}_g^T \otimes \boldsymbol{1}_{D'}) \odot \boldsymbol{F}^o, \end{aligned} \tag{2}$$

where $\boldsymbol{a}_g \in \mathbb{R}^S = [a_{g,1}, ..., a_{g,S}]^T$, $\boldsymbol{1}_{D'} \in \mathbb{R}^{D'} = [1, ..., 1]^T$, $\odot$ and $\otimes$ are an element-wise and a Kronecker product of matrices, respectively.

By concatenating the feature vectors of G areas, we have D $(= D'G)$ dimensional feature vector,

$$\boldsymbol{x} = \left[\begin{array}{c} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_G \end{array} \right] = \left[\begin{array}{c} \boldsymbol{F}_1^* \boldsymbol{w}^c \\ \vdots \\ \boldsymbol{F}_G^* \boldsymbol{w}^c \end{array} \right] = \boldsymbol{F}^* \boldsymbol{w}^c, \tag{3}$$

where $\boldsymbol{F}^* \in \mathbb{R}^{D \times S}$ is a feature matrix given by

$$\begin{aligned} \boldsymbol{F}^* &= \left[\begin{array}{c} \boldsymbol{F}_1^* \\ \vdots \\ \boldsymbol{F}_G^* \end{array} \right] = \left[\begin{array}{ccc} a_{1,1}\boldsymbol{f}_1 & \cdots & a_{1,S}\boldsymbol{f}_S \\ \vdots & \ddots & \vdots \\ a_{G,1}\boldsymbol{f}_1 & \cdots & a_{G,S}\boldsymbol{f}_S \end{array} \right] \\ &= \left[\begin{array}{ccc} a_{1,1}\boldsymbol{1}_{D'} & \cdots & a_{1,S}\boldsymbol{1}_{D'} \\ \vdots & \ddots & \vdots \\ a_{G,1}\boldsymbol{1}_{D'} & \cdots & a_{G,S}\boldsymbol{1}_{D'} \end{array} \right] \odot \left[\begin{array}{c} \boldsymbol{F}^o \\ \vdots \\ \boldsymbol{F}^o \end{array} \right] \\ &= (\boldsymbol{A} \otimes \boldsymbol{1}_{D'}) \odot (\boldsymbol{1}_G \otimes \boldsymbol{F}^o), \end{aligned} \tag{4}$$

where $\boldsymbol{A} \in \mathbb{R}^{G \times S} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_G]^T$, and $\boldsymbol{1}_G \in \mathbb{R}^G$ is a vector whose elements are all 1.

2) Camera-specific pooling: We further use the following linear relationship of a concatenated vector of C weight maps $\boldsymbol{w} \in \mathbb{R}^{CS} = [\boldsymbol{w}^{1^T}, ..., \boldsymbol{w}^{C^T}]^T$ and feature matrix:

$$\boldsymbol{x} = \boldsymbol{F}^* \boldsymbol{w}^c = \sum_{i=1}^{C} b_i \boldsymbol{F}^* \boldsymbol{w}_i^c = \boldsymbol{F}\boldsymbol{w}, \tag{5}$$

where b_i is a binary scalar that shows the camera ID. We set it as $b_i = 1$ if $i = c$ and $b_i = 0$ otherwise. The augmented feature matrix $\boldsymbol{F} \in \mathbb{R}^{D \times CS}$ is given by

$$\boldsymbol{F} = [b_1 \boldsymbol{F}_1^*, ..., b_C \boldsymbol{F}^*] = \boldsymbol{b}^T \otimes \boldsymbol{F}^*, \tag{6}$$

where $\boldsymbol{b} \in \mathbb{R}^C = [b_1, ..., b_C]^T$.

For example, when $C = 2$, the augmented feature matrix is given by $\boldsymbol{F} = [\boldsymbol{F}^* \quad \boldsymbol{0}_{D \times S}]$ for camera $c = 1$ and $\boldsymbol{F} = [\boldsymbol{0}_{D \times S} \quad \boldsymbol{F}^*]$ for camera $c = 2$, where $\boldsymbol{0}_{D \times S} \in \mathbb{R}^{D \times S}$ is the matrix whose elements are all zero (see Fig. 2 (d)). Applying this framework in the case for $C > 2$ is straightforward from Eq.(6).

B. Discriminative weight map learning

For each of the concerned convolutional layers of the trained CNN model, we form a training dataset $\{\boldsymbol{F}_i, p_i, c_i\}_{i=1}^N$ on the target dataset where $\boldsymbol{F}_i \in \mathbb{R}^{D\times CS}$, $p_i \in \{1, ..., P\}$, and $c_i \in \{1, ..., C\}$ are a feature matrix, person ID and camera ID of the i-th image, respectively. P and C are the total number of the IDs.

The weighted pooling compresses CS-dimensional spatial information into 1-dimension. This process inevitably reduces information in the original feature map. We thus presume the existence of K discriminative weight maps $\boldsymbol{W} \in \mathbb{R}^{CS\times K} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_K]$ and represent the set of pooled features vectors as $\{\{\boldsymbol{x}_{k,i} = \boldsymbol{F}_i\boldsymbol{w}_k\}_{k=1}^K\}_{i=1}^N$.

1) Objective function: For the objective function, we adopt *Maximum Margin*-like criterion with an orthogonal constraint [20], [21] to learn different weight maps in closed-form.

We consider sample pairs of different camera views by reflecting the person re-id task. Let $\mathcal{S} = \{(i,j)|p_i = p_j, c_i \neq c_j\}$ and $\mathcal{D} = \{(i,j)|p_i \neq p_j, c_i \neq c_j\}$ be sample index sets of the same and different persons' pairs on the training dataset, respectively. $N_\mathcal{S}$ and $N_\mathcal{D}$ are their sample number.

Discriminative weight maps should make pooled feature distances in the set $\mathcal{S}$ small, and those of the set $\mathcal{D}$ large. We define discriminative criterion by

$$J(\boldsymbol{W}) = \frac{1}{N_\mathcal{D}} \sum_{(i,j)\in\mathcal{D}} \delta^2_{\boldsymbol{W}}(i,j) - \frac{1}{N_\mathcal{S}} \sum_{(i,j)\in\mathcal{S}} \delta^2_{\boldsymbol{W}}(i,j), \quad (7)$$

where $\delta^2_{\boldsymbol{W}}(i,j) = \sum_{k=1}^K \delta^2_{\boldsymbol{w}_k}(i,j)$ is the sum of distances of samples $\{i,j\}$ over K weight maps and $\delta^2_{\boldsymbol{w}_k}(i,j)$ is a distance between features vectors $\boldsymbol{x}_{k,i}$ and $\boldsymbol{x}_{k,j}$ produced by the k-th weight map $\boldsymbol{w}_k$.

2) Approximate distance: The feature channel dimension D' of convolutional layers often exceeds one thousand, and the dimension of $\boldsymbol{x}$ is a product of D' and the number of areas G. If we use Euclidean distance directly, $\delta^2_{\boldsymbol{w}_k}(i,j)$ requires a high computational cost.

To reduce the computational cost without any data-dependent process, we propose to use random projections [22]. A random projection matrix $\boldsymbol{R} \in \mathbb{R}^{D\times E}, E \ll D$ which has properties of *Johnson-Lindenstrauss (JL) Lemma* is known to preserve the geometry of Euclidean distance. We use a random orthonormal projection matrix $\boldsymbol{R}$ which satisfies JL Lemma [23]. Then the approximate squared Euclidean distance of the sample $\{i,j\}$ is given as follows:

$$\begin{aligned}\delta^2_{\boldsymbol{w}_k}(i,j) &= \|\boldsymbol{R}^T\boldsymbol{x}_{k,i} - \boldsymbol{R}^T\boldsymbol{x}_{k,j}\|_2^2 \quad (8)\\ &= \|\boldsymbol{R}^T\boldsymbol{F}_i\boldsymbol{w}_k - \boldsymbol{R}^T\boldsymbol{F}_j\boldsymbol{w}_k\|_2^2 = \|\boldsymbol{Q}_i\boldsymbol{w}_k - \boldsymbol{Q}_j\boldsymbol{w}_k\|_2^2.\end{aligned}$$

Here we denote $\boldsymbol{Q} \in \mathbb{R}^{E\times CS} = \boldsymbol{R}^T\boldsymbol{F}$.

Note that we use the random projection only for the weight map learning process to prevent loss of discriminative feature channel information of pooled features. Meanwhile, we experimentally confirmed that compressing feature channel information gives a positive impact on weight map learning.

3) Optimization problem: By maximizing the discriminative criterion with an orthogonal constraint, the optimization problem becomes as follows[1]:

$$\begin{aligned}\max_{\boldsymbol{W}} \quad & J(\boldsymbol{W}) = \mathrm{Tr}\left[\boldsymbol{W}^T\boldsymbol{\Sigma}_D\boldsymbol{W}\right] - \mathrm{Tr}\left[\boldsymbol{W}^T\boldsymbol{\Sigma}_S\boldsymbol{W}\right], \quad (9)\\ s.t. \quad & \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_K,\end{aligned}$$

where $\boldsymbol{\Sigma}_\mathcal{D} = \frac{1}{N_\mathcal{D}}\sum_{(i,j)\in\mathcal{D}}(\boldsymbol{Q}_i - \boldsymbol{Q}_j)^T(\boldsymbol{Q}_i - \boldsymbol{Q}_j)$ and $\boldsymbol{\Sigma}_\mathcal{S} = \frac{1}{N_\mathcal{S}}\sum_{(i,j)\in\mathcal{S}}(\boldsymbol{Q}_i - \boldsymbol{Q}_j)^T(\boldsymbol{Q}_i - \boldsymbol{Q}_j)$ are covariance matrices of matrix $\boldsymbol{Q}$ for sets $\mathcal{S}$ and $\mathcal{D}$, respectively, and $\boldsymbol{I}_K$ is a K-dimensional identity matrix.

By using the Lagrange multiplier method, the optimization problem reduces to an eigenproblem of $\boldsymbol{\Sigma}_\mathcal{D} - \boldsymbol{\Sigma}_\mathcal{S}$, where the optimal weight maps $\boldsymbol{W}$ are obtained as the eigenvectors corresponding to the K largest eigenvalues.

C. Distance metric learning for pooled features

After we learned the weight maps, we obtain feature vectors using each of the weight maps. Because the CNN model is trained on a different dataset from the target dataset, the pooled features would not be suitable for the target dataset due to nonoptimal feature channel information.

We thus learn a distance metric $\boldsymbol{M}_k \in \mathbb{R}^{D\times D}$ for each k-th weighted feature. Note that $\boldsymbol{M}_k$ is effective if constrained with a low-rank positive semi-definite matrix and in such a case, there is a decomposition of $\boldsymbol{M}_k = \boldsymbol{V}_k\boldsymbol{V}_k^T$, where $\boldsymbol{V}_k \in \mathbb{R}^{D\times U}$ and $U \leq D$. Namely, the squared Mahalanobis distance for the k-th feature map of sample pair (i,j) becomes as follows:

$$\begin{aligned}\delta^2_{\boldsymbol{M}_k,\boldsymbol{w}_k}(i,j) &= (\boldsymbol{x}_{k,i} - \boldsymbol{x}_{k,j})^T\boldsymbol{M}_k(\boldsymbol{x}_{k,i} - \boldsymbol{x}_{k,j}) \quad (10)\\ &= \|\boldsymbol{V}_k^T\boldsymbol{x}_{k,i} - \boldsymbol{V}_k^T\boldsymbol{x}_{k,j}\|_2^2 = \|\boldsymbol{z}_{k,i} - \boldsymbol{z}_{k,j}\|_2^2.\end{aligned}$$

Thus, calculation of the distance is efficient without random projections for the pooled features, because the vector $\boldsymbol{z}_k \in \mathbb{R}^U = \boldsymbol{V}_k^T\boldsymbol{x}_k$ is low-dimensional.

Because the weight map and distance metric learning are disjoint, we can use an arbitrary metric learning method. In this paper, we use XQDA metric [2], [5]. Note that XQDA estimates optimal rank U automatically, and we use the default regularization parameter after we normalize each feature vector $\boldsymbol{x}$ with mean removal and L2 norm normalization.

D. Application to multiple convolutional layers

Because different CNN layers include features of different semantic-levels and sizes, the use of multiple convolutional layers would help to distinguish different persons. We thus transfer convolutional features from L convolutional layers.

For simplicity, we independently treat each convolutional layer and repeat the training procedure to obtain L sets of metric and weight map pairs $\Omega = \{\{\boldsymbol{w}_k^l, \boldsymbol{M}_k^l\}_{k=1}^K\}_{l=1}^L$.

In the test phase, given the probe image $\hat{i}$ and gallery image $\hat{j}$, we use the following distance to perform person re-id:

$$\delta^2_\Omega(\hat{i},\hat{j}) = \sum_{l=1}^L\sum_{k=1}^K \delta^2_{\boldsymbol{M}_k^l,\boldsymbol{w}_k^l}(\hat{i},\hat{j}). \quad (11)$$

[1] See Appendix A for derivation.

IV. Experiments

A. Implementation details

We use a stable baseline model for person re-id, Part-based Convolutional Baseline (PCB) [6] with ResNet-50 architecture as the CNN model. We exploit the MSMT17 dataset [24] as the source dataset, containing 126,441 images of 4101 persons captured from 15 camera views, which is currently the largest dataset to the best of our knowledge. We use the public PyTorch code[2] with the default parameters. We run the training for 60 epochs on a machine equipped with two NVIDIA TITAN XP GPUs.

We implement C-DPCF in Matlab and run it on a computer equipped with an Intel Core-i9 10940X CPU. We extract features from the output of four convolutional layers, *i.e.,* $L = 4$; these are conv2_x, conv3_x, conv4_x, and conv5_x layers of the ResNet-50 architecture [25]. To reduce the computational cost, we reduce the spatial size of the feature map by local average pooling to 24×8. Thus, the spatial dimension is common for all layers and that is $S = 192 (= 24 \times 8)$. The size of the feature channel dimension D' is $256, 512, 1024$, and 2048, for the aforementioned layers, respectively. We set the parameters of C-DPCF as follows: the number K of weight maps to 10, the dimension E of the random projection to 64, and the number G of horizontal stripes to 6. We evaluate our method in the case of two camera views, *i.e.,* $C = 2$.

B. Datasets and settings

We evaluate the proposed method on four small-scale target datasets: VIPeR [26], GRID [27], PRID450S [28] and CUHK01 [29]. The VIPeR dataset contains 1264 images, where every two images are captured for each identity from two camera views. The GRID dataset contains 1,275 images with 250 annotated persons and an additional 775 gallery images of persons except those included as annotated persons. We regard the gallery and probe sets as different camera views. The PRID450S dataset contains 900 images of 450 persons captured by two different surveillance cameras. The CUHK01 dataset contains 3,884 images of 971 persons, and each camera view in the dataset contains two images of each person.

We use the standard *single-shot* protocols on the testing datasets for comparison with prior works. We report the average performance of 10 random person splits for the training/test sets in which each set includes half of all person identities. We calculate the CMC curve, which gives an expectation of finding the correct person in the top ranks. To evaluate the whole CMC curve, we also report the PUR, which assesses the uncertain reduction from the random ranking [4].

C. Compared methods

We use the following baselines for comparison:

PCB (source): The PCB model trained on the source dataset. We use cosine distance of the PCB-g layer [6], which is the local averaged pooled features of the last convolutional layer (conv5_x) on 6 horizontal stripes. Note that with the cosine distance, lower layers perform poorly, and the use of multiple layers could not improve the performance (see Sec.IV.F).

PCB (target): The fine-tuned PCB model on the target dataset. We conduct the training on the training set of a target dataset for 60 epochs with the default hyper parameter of the code by using PCB (source) as initialization.

AvgP+XQDA: The unweighted average pooling and transfer learning with XQDA. We use the same horizontal stripes as C-DPCF on the L-layers of PCB (source). We use the sum of the distances of L-layers with the XQDA distance metric in the same way as C-DPCF.

AvgP+pad.+XQDA: A naive camera-specific feature augmentation applied to unweighted AvgP. We use zero-padding for each camera view [17] and the same setup as AvgP+XQDA.

DPCF: The proposed method without the camera-specific feature matrix augmentation.

D. Performance comparison on the standard protocol

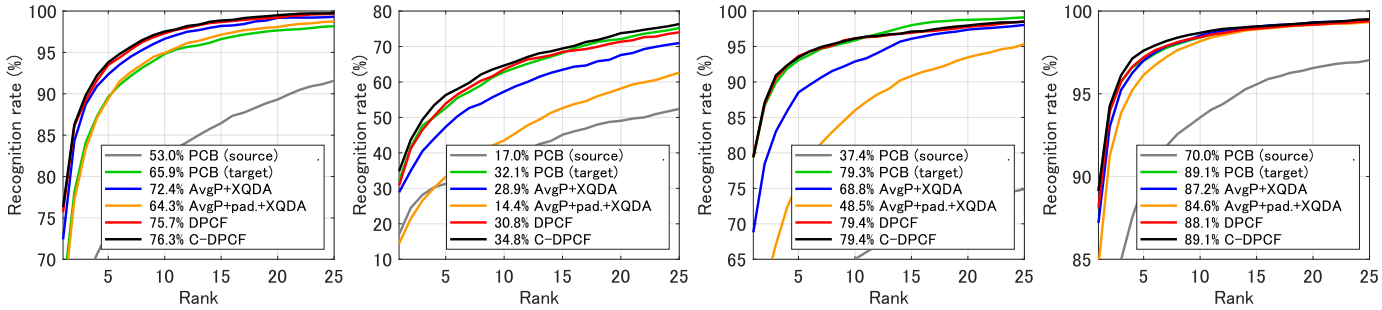
1) Baselines: Fig. 3 shows the comparison with baselines. It is evident that discriminative weight maps improve the re-id accuracies by seeing the fact that DPCF moderately outperforms AvgP+XQDA. Also, AvgP+XQDA outperforms PCB (source), which confirm the effectiveness of metric learning for features transfer compared with the direct use of the pre-trained model.

Slightly higher scores of C-DPCF compared with DPCF reveal the impact of camera-specific weight maps. Meanwhile, lower scores of AvgP+pad.+XQDA compared with AvgP+XQDA ensure the higher overfitting risk of feature vector augmentation. These results are probably because the zero-padding on vector spaces produces large dimensions for training XQDA. Such performance decreases by the naïve feature augmentation agree with the previous work [17]. In contrast, the feature matrix augmentation in spatial dimension keeps the size of the pooled features vector. Thus, it has a smaller overfitting risk and could improve performance.

Lastly, we compare C-DPCF with the fine-tuned model on the respective target dataset. Overall, the C-DPCF performs slightly higher or competitive to PCB (target). On the VIPeR dataset, PCB (target) performs much lower than C-DPCF. The reason would be the high diversity of persons' clothing, race, and poses. With the limited number of training images, the fine-tuned convolutional features could not generalize to its test dataset. Note that the use of multiple layers of C-DPCF potentially achieves higher performance than PCB (target), which uses only the last convolutional layer (see Sec.IV.F). On the PRID450S dataset, illumination colors are consistently different from other datasets in most images. Thus, fine-tuned convolutional features could easily learn such a color bias with a small number of training images, which would lead to higher performance gains of PCB (target). High-performance of PCB (target) on the CUHK01 dataset is not surprising because it contains a larger number of training images and smaller person variations compared with the VIPeR dataset.

Note that C-DPCF/DPCF required shorter training times than PCB (target). On the VIPeR dataset, PCB (target) re-

[2]https://github.com/syfafterzy/PCB_RPP_for_reID



(a) VIPeR (b) GRID (c) PRID450S (d) CUHK01

Fig. 3. CMC curve comparison on the standard evaluation protocol. The number shows the rank-1 rates.

quired 312 seconds with two GPUs, whereas the total training times of DPCF and C-DPCF (excluding the time needed for extracting the feature map $\boldsymbol{F}^o$ of training data) were 39.7 and 42.5 seconds on a single CPU, respectively.

2) State-of-the-arts : Table I lists the state-of-the-art results on the target datasets with various approaches; metric learning on hand-crafted descriptors [19], [30], [3], Domain Adaptation (DA) [31], CNN feature transfers [15], [17] and Domain Generalization (DG) [11], [10]. Most of the DA methods [32], [33] are unsupervised, which uses a large number of unlabeled target samples. Synthesis [31] translates a synthetic person image dataset to the target domain, and the translated dataset is combined with real datasets on other source domains for the training of a CNN model. Thus, Synthesis uses a Multi-Source (MS-a)[3] dataset for training. One-shot metric learning transferred a trained CNN on another Multi-Source (MS-b)[4] dataset to a target dataset with one-positive pair and unlabeled samples [15]. We see that C-DPCF outperforms these methods, yet the used CNNs and trained datasets are different.

To properly evaluate the advantage of C-DPCF to a recent DG model, we assessed the trained MobileNet_IFN[5] model on the MS-c dataset[6] by DualNorm (DN) [10]. We then applied the C-DPCF to the last four convolutional blocks of the MobileNet_IFN. Because the CUHK02 and CUHK03 datasets include the CUHK01 dataset, we could not apply DN to the CUHK01 dataset. The bottom block of Table I shows the results. We see that C-DPCF could significantly improve the performance of the DG re-id model by using the dataset-specific properties on the target dataset.

E. Weight map analysis

1) Typical example: Fig. 4 visualizes the learned weight maps on the CUHK01 dataset. We observe that most stripes have the weights in the same sign. Note that the difference of the sign among stripes would have no significant influence on the matching because the pooled features are a weighted sum within each pooling area. Therefore, a point that has a higher absolute value is more discriminate.

[3]MS-a: CUHK03 + DukeMTMC4ReID + SyRI [31].
[4]MS-b: CUHK03+CUHK01+VIPeR+iLIDS+PRID (excluding target) [15].
[5]https://github.com/BJTUJia/person_reID_DUalNorm
[6]MS-c: Market-1501 + Duke + CUHK02 + CUHK03 + PersonSearch [10].

TABLE I
STATE-OF-THE-ARTS OF THE TARGET DATASETS (RANK-1 RATES) .

Method	Type	Source	VIPeR	GRID	PRID450S	CUHK01
DALF [19]	S	-	35.4	18.1	-	-
CMDL [30]	S	-	66.4	30.9	52.0	78.2
HGD [3]	S	-	52.8	28.2	-	-
Synthesis [31]	U	MS-a[3]	43.0	-	-	54.9
One-shot [15]	U+S	MS-b[4]	34.3	-	-	45.6
CRAFT [17]	S	ImageNet	50.3	22.4	-	-
DPCF/C-DPCF	S	MSMT17	75.7/76.3	30.8/34.8	79.4/79.4	88.1/89.1
DIMN [11]	DG	MS-c[6]	51.2	29.3	-	-
DN [10]	DG	MS-c[6]	58.8*	39.7*	73.6*	-
DN [10] + ours	DG+S	MS-c[6]	73.6/73.9	40.2/42.3	83.6/84.1	-

U/S: Unsupervised/Supervised on a target dataset, DG: Domain Generalization, MS: Multi-Source dataset, *: experimented by us.

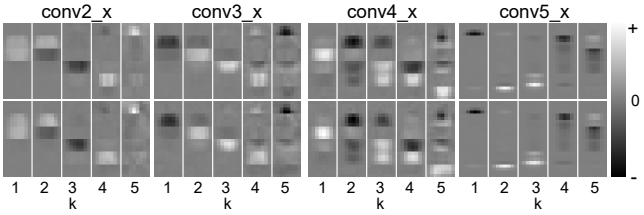


Fig. 4. 1st-5th weight maps of each layer learned on the CUHK01 dataset. The upper and lower rows correspond to camera 1 and 2, respectively.

We also see that each weight map is sparse in terms of the horizontal stripes. The reason for the sparse weights is that the selection of the most discriminative horizontal stripes can maximize the discriminative criterion. Besides, the orthogonal constraint enables us to generate several weight maps. Thereby, C-DPCF maintains discriminative spatial properties in diverse spatial locations.

2) Dataset-wise analysis: Fig. 5 compares the sum of the absolute values of the 1st-5th weight maps of each dataset.

On the VIPeR dataset, cameras 1 and 2 respectively capture most persons from near-front views and the side or rear views. We ensure that the average images of cameras 1 and 2 appear to be front view and back view, respectively. The upper body parts of the learned weight maps of two camera views disagree.

On the PRID450S dataset, person images in both camera views tend to be a side view, whereas the background color tends to differ. The learned weight maps have no significant differences because there is no significant spatial difference between the two camera views.

On the CUHK01 dataset, the person images of camera 1 tend to be front or rear views, and camera 2 tends to be a side view. We see that the weight maps on cameras 1 and 2 have

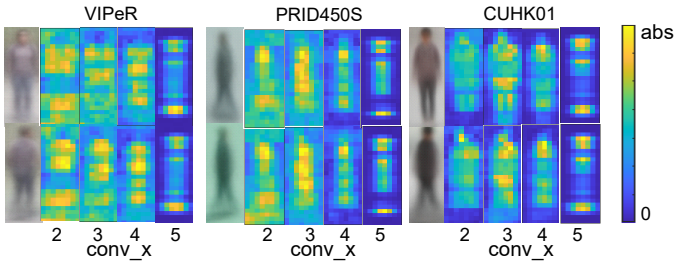


Fig. 5. Sum of the absolute values of the 1st-5th weight maps. The leftmost image of each dataset shows the average image on each camera view.

shallower and broader lower body regions, respectively. These weight maps reflect the spatial bias of each camera view well.

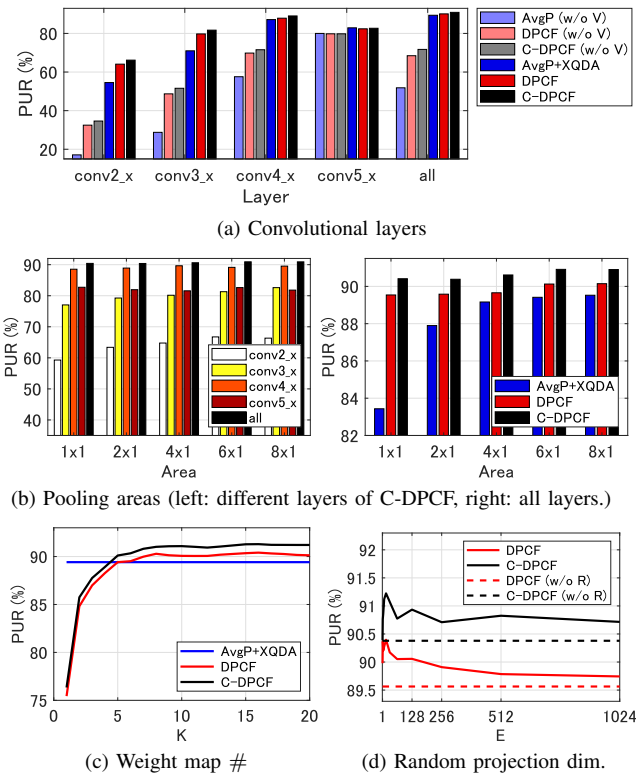
F. Parameter analysis

1) Convolutional layers: Fig. 6 (a) compares the performance of each layer. Without XQDA (denoted by w/o $\boldsymbol{V}$), the upper layers perform better than the lower layers, and the conv5_x layer produces the best score. XQDA significantly improves the performance of lower layers and promotes the conv4_x layer to the best single layer. Also, the use of all four layers (denoted by all) shows the best results in the case with XQDA. These results confirm the effectiveness of the use of multilevel semantic information and feature sizes after the feature channels are optimized in the target dataset.

We also ensure that the effective layers of DPCF and C-DPCF are consistent with AvgP+XQDA. Meanwhile, the performance gain from AvgP+XQDA is more significant in the lower layers. The reason is probably due to the large receptive field of upper layers. Namely, due to the hierarchical processing of convolution and local pooling, features in upper layers correspond to large regions in the input image. Hence, omitting indiscriminative spatial information is harder for upper layers compared with lower layers.

2) Pooling areas: Fig. 6 (b) compares the different pooling areas. We see that narrower horizontal stripes tend to show a higher score. Compared with AvgP+XQDA, narrower stripes show moderate improvements for DPCF/C-DPCF, and their score saturates at 6×1 stripes. The reason is that even with the global (1×1) area, each of the weight maps corresponds to local parts, as shown in Fig. 4, and the use of multiple weight maps causes somewhat a similar effect to the use of narrower stripes. Nevertheless, we still see an improvement by the use of shallower stripes, which validates the impact of localized pooling areas [19].

3) Number of weight maps: Fig. 6 (c) compares different numbers K of weight maps. We see that the performance increases as the number increases. Note that when the $K = 1$, DPCF/C-DPCF performs no better than AvgP+XQDA. As shown in Fig. 4, these results are due to the fact that the focus of discriminative points is limited within a sub-region of an image, and the important information would be lost. When $K \geq 5$, we can ensure that the systematic generation of multiple weighted pooled features outperforms the simple average pooling.



(a) Convolutional layers

(b) Pooling areas (left: different layers of C-DPCF, right: all layers.)

(c) Weight map $\#$

(d) Random projection dim.

Fig. 6. Detailed analysis on the CUHK01 dataset.

4) Random projection dimension: Fig 6 (d) compares the dimension E of the random projection. We see that lower E shows a higher score and outperforms the case without dimension compression (denoted by w/o $\boldsymbol{R}$). These results are probably because the covariance matrices for the weight map learning are equivalent to the sum of the covariance matrices of each feature channel (see Appendix B). If the common feature channel is not activated between different images, it may fail to learn discriminative spatial information. In contrast, the random projections aggregate several feature channel dimensions; thus, it would be robust to such a situation. We also see that too few random projection dimension reduces the performance, which confirms that keeping the feature channel dimension is still useful for learning weight maps.

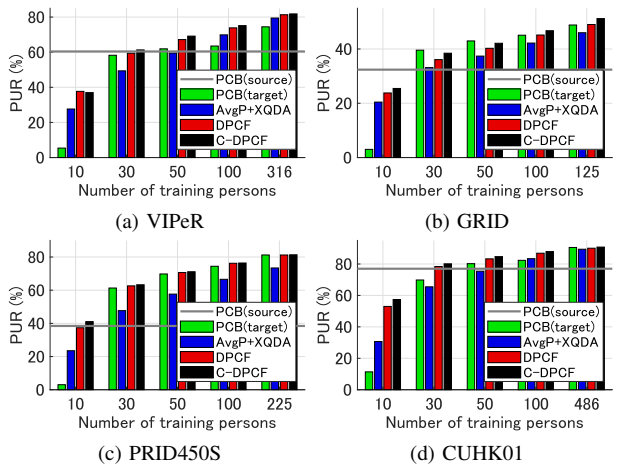
Table II shows computational time of the weight map learning with $E = 64$. We see that the random projection enables about 25-30$\times$ faster training.

G. Evaluation on a smaller number of training person IDs

To evaluate our model on a smaller training person IDs, we restrict training samples on a target dataset by randomly selecting $10/30/50/100$ persons from the original training split. Test datasets are the same as the original settings for ease of comparison. As a remedy of the person sampling bias, we repeat the experiments 5 times on each of the 10 random splits of the standard settings and report the average results.

TABLE II
TRAINING TIME OF THE WEIGHT MAPS ON THE CUHK01 DATASET .

	DPCF	DPCF (w/o $\boldsymbol{R}$)	C-DPCF	C-DPCF (w/o $\boldsymbol{R}$)
Time (seconds)	43.6	1163.9	49.0	1684.4



(a) VIPeR (b) GRID

(c) PRID450S (d) CUHK01

Fig. 7. Evaluation of small number of training person IDs.

Fig. 7 shows the results. We see that DPCF consistently outperforms AvgP+XQDA. Also, DPCF outperforms PCB (source) when the number of training persons is greater than 30. These results show that our method transferred convolutional features effectively, except the case when the number of training persons is too small. Moreover, C-DPCF outperforms DPCF in most cases. These results show our camera-specific weight map training caused no serious overfitting problems on small training samples.

V. Conclusions

Deployment of a pre-trained CNN model to new camera locations demands to solve person re-id with small training samples in a target dataset. We have proposed a method to learn discriminative pooling for convolutional features of a pre-trained CNN, which requires no end-to-end re-training on the target dataset. We have also proposed camera-specific weight maps to handle spatial bias of different camera views.

Experiments showed that the discriminative pooling trained with target samples composed of a small number of person IDs improves the performance of the pre-trained CNN model. We also confirmed that the camera-specific weight maps could improve re-id accuracies when camera-bias is significant without introducing serious overfitting problems. In the future, we plan to apply the proposed method to middle-sized datasets, which include more than three camera views.

References

[1] S. Karanam, M. Gou, Z. Wu, A. Rates-Borras, O. Camps, and R. J. Radke, "A systematic evaluation and benchmark for person re-identification: feature, metrics, and datasets," *IEEE Trans. on PAMI*, vol. 41, no. 3, pp. 523–536, 2019.

[2] S. Liao, Y. Hu, X. Zhu, and S. Z. Li, "Person re-identification by local maximal occurrence representation and metric learning," in *Proc. CVPR*, 2015.

[3] T. Matsukawa, T. Okabe, E. Suzuki, and Y. Sato, "Hierarchal Gaussian descriptors with applications to person re-identification," *IEEE Trans. on PAMI*, vol. 42, no. 9, pp. 2179–2194, 2020.

[4] S. Pedagadi, J. Orwell, S. Velastin, and B. Boghossian, "Local Fisher discriminant analysis for pedestrian re-identification," in *Proc. CVPR*, 2013.

[5] T. Matsukawa and E. Suzuki, "Kernelized cross-view quadratic discriminant analysis for person re-identification," in *Proc. MVA*, 2019.

[6] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang, "Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline)," in *Proc. ECCV*, 2018.

[7] Y. Fu, Y. Wei, Y. Zhou, H. Shi, G. Huang, X. Wang, Z. Yao, and T. Huang, "Horizontal pyramid matching for person re-identification," in *Proc. AAAI*, 2019.

[8] G. Wang, Y. Yuan, X. Chen, J. Li, and X. Zhou, "Learning discriminative features with multiple granularities for person re-identification," in *Proc. ACMMM*, 2018.

[9] F. Zheng, C. Deng, X. Sun, X. Jiang, X. Guo, Z. Yu, F. Huang, and R. Ji, "Pyramidal person re-identification via multi-loss dynamic training," in *Proc. CVPR*, 2019.

[10] J. Jia, Q. Ruan, and T.M.Hospedales, "Frustratingly easy person re-identification: Generalizing person re-id in practice," in *Proc. BMVC*, 2019.

[11] J. Song, Y. Yang, Y.-Z. Song, T. Xiang, and T. M. Hospedales, "Generalizable person re-identification by domain-invariant mapping network," in *Proc. CVPR*, 2019.

[12] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. ICML*, 2014.

[13] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. CVPR Workshop*, 2014.

[14] T. Matsukawa and E. Suzuki, "Person re-identification using CNN features learned from combination of attributes," in *Proc. ICPR*, 2016.

[15] S. Bak and P. Carr, "One-shot metric learning for person re-identification," in *Proc. CVPR*, 2017.

[16] L. Zheng, Y. Yang, and A. G. Hauptmann, "Person re-identification: Past, present and future," *arXiv preprint arXiv:1610.02984*, 2016.

[17] Y.-C. Chen, X. Zhu, W.-S. Zheng, and J.-H. Lai, "Person re-identification by camera correlation aware feature augmentation," *IEEE Trans. on PAMI*, vol. 40, no. 2, pp. 392–408, 2018.

[18] N. Martinel, G. L. Foresti, and C. Micheloni, "Deep pyramidal pooling with attention for person re-identification," *IEEE Trans. on IP*, vol. 29, pp. 7306–7316, 2020.

[19] T. Matsukawa, T. Okabe, and Y. Sato, "Person re-identification via discriminative accumulation of local features," in *Proc. ICPR*, 2014.

[20] H. Li, T. Jiang, and K. Zhang, "Efficient and robust feature extraction by maximum margin criterion," in *Proc. NeurIPS*, 2003.

[21] B. Alipanahi, M. Biggs, and A. Ghodsi, "Distance metric learning vs. Fisher discriminant analysis," in *Proc. AAAI*, 2008.

[22] E. Bigham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proc. KDD*, 2004.

[23] K. Zhao, A. Alavi, A. Williem, and B. Lovell, "Efficient clustering on Riemannian manifolds: A kernelised random projection approach," *Pattern Recognition*, vol. 51, pp. 333–345, 2016.

[24] L. Wei, S. Zhang, W. Gao, and Q. Tian, "Person transfer GAN to bridge domain gap for person re-identification," in *Proc. CVPR*, 2018.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[26] D. Gray and H. Tao, "Viewpoint invariant pedestrian recognition with an ensemble of localized features," in *Proc. ECCV*, 2008.

[27] C. C. Loy, T. Xiang, and S. Gong, "Time-delayed correlation analysis for multi-camera activity understanding," *IJCV*, vol. 90, no. 1, pp. 106–129, 2010.

[28] P. M. Roth, M. Hirzer, M. Köstinger, C. Beleznai, and H. Bischof, "Mahalanobis distance learning for person re-identification," *Person Re-Identification*, 2014.

[29] W. Li, R. Zhao, and X. Wang, "Human reidentification with transferred metric learning," in *Proc. ACCV*, 2012.

[30] S. Li, M. Shao, and Y. Fu, "Person re-identification by cross-view multi-level dictionary learning," *IEEE TPAMI*, vol. 40, pp. 2963–2977, 2018.

[31] S. Bak, P. Carr, and J.-F. Lalonde, "Domain adaptation through synthesis for unsupervised person re-identification," in *Proc. ECCV*, 2018.

[32] Y. Zhai, S. Lu, Q. Ye, X. Shan, J. Chen, R. Ji, and Y. Tian, "Ad-cluster: Augmented discriminative clustering for domain adaptive person re-identification," in *Proc. CVPR*, 2020.

[33] D. Mekhazni, A. Bhuiyan, G. Eskander, and E. Granger, "Unsupervised domain adaptation in the dissimilarity space for person re-identification," in *Proc. ECCV*, 2020.

APPENDIX

A. Derivation of the optimization problem of Eq.(9)

The first term of the objective function $J(\boldsymbol{W}) = \frac{1}{N_D}\sum_{(i,j)\in D}\delta^2_{\boldsymbol{W}}(i,j) - \frac{1}{N_S}\sum_{(i,j)\in S}\delta^2_{\boldsymbol{W}}(i,j)$ can be written as

$$
\begin{aligned}
&\frac{1}{N_S}\sum_{(i,j)\in S}\delta^2_{\boldsymbol{W}}(i,j) = \frac{1}{N_S}\sum_{(i,j)\in S}\sum_{k=1}^{K}\delta^2_{\boldsymbol{w}_k}(i,j) \\
&= \frac{1}{N_S}\sum_{k=1}^{K}\sum_{(i,j)\in S}\delta^2_{\boldsymbol{w}_k}(i,j) = \frac{1}{N_S}\sum_{k=1}^{K}\sum_{(i,j)\in S}\mathrm{Tr}\left[\delta^2_{\boldsymbol{w}_k}(i,j)\right] \\
&= \frac{1}{N_S}\sum_{(i,j)\in S}\sum_{k=1}^{K}\mathrm{Tr}[(\boldsymbol{R}^T\boldsymbol{x}_{i,k} - \boldsymbol{R}^T\boldsymbol{x}_{j,k})^T \\
&\qquad\qquad (\boldsymbol{R}^T\boldsymbol{x}_{i,k} - \boldsymbol{R}^T\boldsymbol{x}_{j,k})] \\
&= \frac{1}{N_S}\sum_{(i,j)\in S}\sum_{k=1}^{K}\mathrm{Tr}[(\boldsymbol{R}^T\boldsymbol{F}_i\boldsymbol{w}_k - \boldsymbol{R}^T\boldsymbol{F}_j\boldsymbol{w}_k)^T \\
&\qquad\qquad (\boldsymbol{R}^T\boldsymbol{F}_i\boldsymbol{w}_k - \boldsymbol{R}^T\boldsymbol{F}_j\boldsymbol{w}_k)] \\
&= \frac{1}{N_S}\sum_{(i,j)\in S}\sum_{k=1}^{K}\mathrm{Tr}\left[(\boldsymbol{Q}_i\boldsymbol{w}_k - \boldsymbol{Q}_j\boldsymbol{w}_k)^T(\boldsymbol{Q}_i\boldsymbol{w}_k - \boldsymbol{Q}_j\boldsymbol{w}_k)\right] \\
&= \frac{1}{N_S}\sum_{(i,j)\in S}\sum_{k=1}^{K}\mathrm{Tr}\left[\boldsymbol{w}_k^T(\boldsymbol{Q}_i - \boldsymbol{Q}_j)^T(\boldsymbol{Q}_i - \boldsymbol{Q}_j)\boldsymbol{w}_k\right] \\
&= \sum_{k=1}^{K}\mathrm{Tr}\left[\frac{1}{N_S}\sum_{(i,j)\in S}\boldsymbol{w}_k^T(\boldsymbol{Q}_i - \boldsymbol{Q}_j)^T(\boldsymbol{Q}_i - \boldsymbol{Q}_j)\boldsymbol{w}_k\right] \\
&= \sum_{k=1}^{K}\mathrm{Tr}\left[\boldsymbol{w}_k^T\boldsymbol{\Sigma}_S\boldsymbol{w}_K\right] = \mathrm{Tr}\left[\boldsymbol{W}^T\boldsymbol{\Sigma}_S\boldsymbol{W}\right]. \qquad (12)
\end{aligned}
$$

The last equality holds true because the trace of the squared matrix corresponds to the sum of its diagonal elements. Similary, the equation $\frac{1}{N_D}\sum_{(i,j)\in D}\delta^2_{\boldsymbol{W}}(i,j) = \mathrm{Tr}\left[\boldsymbol{W}^T\boldsymbol{\Sigma}_{\mathcal{D}}\boldsymbol{W}\right]$ holds true.

B. Covariance matrices of two-dimensional data

Proposition.1 The covariance matrices $\boldsymbol{\Sigma}_{\mathcal{S}}$, $\boldsymbol{\Sigma}_{\mathcal{D}}$ are respectively equivalent to the sum of covariance matrices of similar and dissimilar pairs on row vectors of $\boldsymbol{Q}$.

Proof. Let us denote $\boldsymbol{Q}_i^T = \boldsymbol{P}_i = [\boldsymbol{p}_{1,i}, ..., \boldsymbol{p}_{E,i}]$. We see that

$$
\begin{aligned}
\boldsymbol{\Sigma}_{\mathcal{S}} &= \frac{1}{N_{\mathcal{S}}}\sum_{(i,j)\in\mathcal{S}}(\boldsymbol{P}_i - \boldsymbol{P}_j)(\boldsymbol{P}_i - \boldsymbol{P}_j)^T \\
&= \frac{1}{N_{\mathcal{S}}}\sum_{(i,j)\in\mathcal{S}}\left[\boldsymbol{p}_{1,i} - \boldsymbol{p}_{1,j}, ..., \boldsymbol{p}_{E,i} - \boldsymbol{p}_{E,j}\right] \\
&\qquad\qquad \left[\boldsymbol{p}_{1,i} - \boldsymbol{p}_{1,j}, ..., \boldsymbol{p}_{E,i} - \boldsymbol{p}_{E,j}\right]^T \\
&= \frac{1}{N_{\mathcal{S}}}\sum_{(i,j)\in\mathcal{S}}((\boldsymbol{p}_{1,i} - \boldsymbol{p}_{1,j})(\boldsymbol{p}_{1,i} - \boldsymbol{p}_{1,j})^T + ... \\
&\qquad\qquad ... + (\boldsymbol{p}_{E,i} - \boldsymbol{p}_{E,j})(\boldsymbol{p}_{E,i} - \boldsymbol{p}_{E,j})^T) \\
&= \frac{1}{N_{\mathcal{S}}}\sum_{(i,j)\in\mathcal{S}}(\boldsymbol{p}_{1,i} - \boldsymbol{p}_{1,j})(\boldsymbol{p}_{1,i} - \boldsymbol{p}_{1,j})^T + ... \\
&\qquad\qquad ... + \frac{1}{N_{\mathcal{S}}}\sum_{(i,j)\in\mathcal{S}}(\boldsymbol{p}_{E,i} - \boldsymbol{p}_{E,j})(\boldsymbol{p}_{E,i} - \boldsymbol{p}_{E,j})^T \\
&= \sum_{e=1}^{E}\frac{1}{N_{\mathcal{S}}}\sum_{(i,j)\in\mathcal{S}}(\boldsymbol{p}_{e,i} - \boldsymbol{p}_{e,j})(\boldsymbol{p}_{e,i} - \boldsymbol{p}_{e,j})^T = \sum_{e=1}^{E}\boldsymbol{\Sigma}_{\mathcal{S},e}
\end{aligned}
$$

where $\boldsymbol{\Sigma}_{\mathcal{S},e} = \frac{1}{N_{\mathcal{S}}}\sum_{(i,j)\in\mathcal{S}}(\boldsymbol{p}_{e,i} - \boldsymbol{p}_{e,j})(\boldsymbol{p}_{e,i} - \boldsymbol{p}_{e,j})^T$ is the covariance matrix on the e-th row vector of $\boldsymbol{Q}$. Similarly, $\boldsymbol{\Sigma}_{\mathcal{D}}$ is equivalent to the sum of the covariance matrix of $\boldsymbol{\Sigma}_{\mathcal{D},e}$. □

From this proportion, we can calculate the covariance matrices of the matrix data as the sum of the covariance matrix of the vector space. We use an efficient implementation of cross-view constrained covariance matrices [5].

ACKNOWLEDGMENT

A part of this work was supported by JSPS KAKENHI Grant Number JP17K20008 and JP20K11890. One GPU was donated by the NVIDIA Corporation.