APPENDIX

A. MSN in the Lovrić embedding

We confirm MSN is inherently adopted in the Gaussian matrix $\boldsymbol{P} \in Sym^+_{d+1}$ of Lovrić *et al.* [29] as follows:

Let $\boldsymbol{G} \in Sym^+_{d+1}$ be an SPD matrix excluding the scaling term of $\boldsymbol{P}$ and let $\boldsymbol{G}(i,j)$ be its (i,j)-th block, *i.e.,* $\boldsymbol{G} = \begin{bmatrix} \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{G}(1,1) & \boldsymbol{G}(1,2) \\ \boldsymbol{G}(2,1) & \boldsymbol{G}(2,2) \end{bmatrix}$ and $\boldsymbol{P} = |\boldsymbol{\Sigma}|^{-\frac{1}{d+1}}\boldsymbol{G}$. The property of the submatrix[9] indicates that

$$\begin{aligned} |\boldsymbol{G}| &= |\boldsymbol{G}(2,2)||\boldsymbol{G}(1,1) - \boldsymbol{G}(1,2)\boldsymbol{G}^{-1}(2,2)\boldsymbol{G}(2,1)| \\ &= |1||\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T - \boldsymbol{\mu}(1^{-1})\boldsymbol{\mu}^T| \\ &= |\boldsymbol{\Sigma}|. \end{aligned} \tag{1}$$

Consequently, we have $\boldsymbol{P} = |\boldsymbol{G}|^{-\frac{1}{d+1}}\boldsymbol{G} = \eta(\boldsymbol{G})$.

B. Justification of the bias removal

One motivation of HGDs is the importance of the mean information of pixel features. The Gaussian and ZmG embeddings include the mean information *within each patch/region of an image*. In the norm normalization step in §4.2, the E-L2 and I-L2 normalizations remove the biased components of the training set, which are the mean information *among different images*. One question arises: Do the bias removal steps in the norm normalization contradict with the importance of the mean information? In this appendix, we explain that this contradiction occurs in neither of the cases of the E-L2/I-L2 normalizations.

Bias removal in the E-L2 normalization.
Let $\{\boldsymbol{z}_i\}_{i=1}^{N_T}$ be a set of GOG (or ZOZ) descriptors, $\bar{\boldsymbol{z}} = \frac{1}{N_T}\sum_{i=1}^{N_T}\boldsymbol{z}_i$ be their bias component and $\{\boldsymbol{z}'_i = \boldsymbol{z}_i - \bar{\boldsymbol{z}}\}_{i=1}^{N_T}$ be the descriptors after the bias removal. For any (i,j) pair of descriptors in the set, we see that:

$$\boldsymbol{z}'_i - \boldsymbol{z}'_j = \boldsymbol{z}_i - \bar{\boldsymbol{z}} - (\boldsymbol{z}_j - \bar{\boldsymbol{z}}) = \boldsymbol{z}_i - \boldsymbol{z}_j. \tag{2}$$

Because the descriptors $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ include both the mean and covariance information of the respective images, the bias removal step retains their difference information.

Bias removal in the I-L2 normalization.
Let $\{\boldsymbol{A}_i\}_{i=1}^{N_T}$ be a set of the region Gaussian matrices of GOG or ZOZ. In the I-L2 normalization, the half-vectorized representation is given by $\boldsymbol{z}' = \mathrm{vec}\left(\log\left(\boldsymbol{M}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{M}^{-\frac{1}{2}}\right)\right)$. From Lemma 1 in Ref. [82], $\log(\boldsymbol{M}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{M}^{-\frac{1}{2}})$ can be approximated by $\boldsymbol{M}^{-\frac{1}{2}}\log(\boldsymbol{A})\boldsymbol{M}^{-\frac{1}{2}}$. Thus, we have

$$\begin{aligned} &\boldsymbol{z}'_i - \boldsymbol{z}'_j \\ &\approx \mathrm{vec}\left(\boldsymbol{M}^{-\frac{1}{2}}\log(\boldsymbol{A}_i)\boldsymbol{M}^{-\frac{1}{2}}\right) - \mathrm{vec}\left(\boldsymbol{M}^{-\frac{1}{2}}\log(\boldsymbol{A}_j)\boldsymbol{M}^{-\frac{1}{2}}\right) \\ &= \mathrm{vec}\left(\boldsymbol{M}^{-\frac{1}{2}}\left(\log\boldsymbol{A}_i - \log\boldsymbol{A}_j\right)\boldsymbol{M}^{-\frac{1}{2}}\right). \end{aligned} \tag{3}$$

Because the matrices $\log\boldsymbol{A}_i$ and $\log\boldsymbol{A}_j$ contain both the mean and covariance information of the respective images, $\log\boldsymbol{A}_i - \log\boldsymbol{A}_j$ completely retains their difference information. Because $\boldsymbol{M}^{-\frac{1}{2}}$ is a full-rank matrix, the transformation $\boldsymbol{M}^{\frac{1}{2}}(\cdot)\boldsymbol{M}^{\frac{1}{2}}$ in the last equation in Eq.(24) does not nullify the information. In this way, $\boldsymbol{z}'$ retains the information about the difference in both the mean and covariance of the respective images.

9. See the matrix cookbook http://www2.imm.dtu.dk/pubdb/p.php?3274

TABLE 5
Computational complexity for extracting GOG/ZOZ in one color space in each step: (i) Pixel feature extraction. (ii)/(iii) Patch Gaussian construction/flattening. (iv)/(v) Region Gaussian construction/flattening.

Step	Time Complexity		Space Complexity	
	GOG	ZOZ	GOG	ZOZ
(i)	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$
(ii)	$\mathcal{O}(Nd^2)$	$\mathcal{O}(Nd^2)$	$\mathcal{O}(Nd^2)$	$\mathcal{O}(Nd^2)$
(iii)	$\mathcal{O}(N_p\,(d+1)^3)$	$\mathcal{O}(N_p d^3)$	$\mathcal{O}(N_p\,(d+1)^2)$	$\mathcal{O}(N_p d^2)$
(iv)	$\mathcal{O}(GN_q m^2)$	$\mathcal{O}(GN_q m'^2)$	$\mathcal{O}(N_p m + Gm^2)$	$\mathcal{O}(N_p m' + Gm'^2)$
(v)	$\mathcal{O}(G\,(m+1)^3)$	$\mathcal{O}(Gm'^3)$	$\mathcal{O}(G\,(m+1)^2)$	$\mathcal{O}(Gm'^2)$

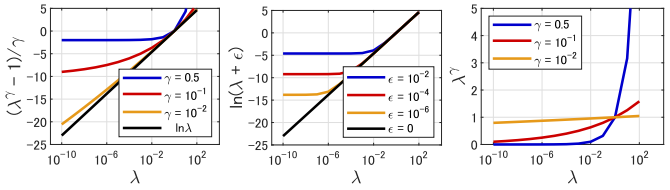
C. Computational complexity

Table 5 lists the computational complexities of each step for GOG/ZOZ in one color space. The total time/space complexities are the summation/maximum complexity of each step.

We show the details of each step for GOG as follows: (i) The pixel feature extraction costs $\mathcal{O}(Nd)$ floating-point operations and storages because d-dimensional features are extracted from N pixels. (ii) The complexities for constructing the patch Gaussians are dominated by the extraction of d-dimensional covariance matrices through integral images with N pixels. This costs $\mathcal{O}(Nd^2)$ floating-point operations and storages. (iii) Flattening a patch Gaussian involved in a principal matrix logarithm that requires an eigenvalue decomposition of a $(d+1)$-dimensional matrix. This costs $\mathcal{O}((d+1)^3)$ floating-point operations and $\mathcal{O}((d+1)^2)$ storages. The flattening process is carried out in each of N_p patches, thus N_p is multiplied. (iv) The complexities for constructing the region Gaussian matrices are dominated by the extraction of m-dimensional covariance matrices. Because the integral images are efficient only when the number of overlapping regions are large, we directly calculate the covariance matrix by Eq.(11). For each of G regions, it costs $\mathcal{O}(N_q m^2)$ floating-point operations where N_q is the number of patches in a region $\mathcal{G}$. In addition, it costs $\mathcal{O}(N_p m)$ storages to maintain the input of the patch Gaussian vectors and $\mathcal{O}(Gm^2)$ storages to maintain the G output matrices. (v) The region Gaussians are flattened for $(m+1)$-dimensional matrices for G regions. It costs $\mathcal{O}(G(m+1)^3)$ floating-point operations and $\mathcal{O}(Gm^2)$ storages. Note that in the case of the I-L2 normalization, $(m+1)$-dimensional matrices are multiplied by each of the region Gaussian matrices before the flattening operation. Because a multiplication between $(m+1)$-dimensional matrices costs $\mathcal{O}((m+1)^3)$ floating-point operations and $\mathcal{O}(m+1)$ storages, this step requires the same complexities as the step (v). The complexities for ZOZ are derived similarly.

In summary, the complexities of each step are of a linear order *w.r.t.* spatial parameters (N, N_p, N_q, and G) and of a square or cubic order *w.r.t.* the dimensionality of pixel features d or patch Gaussian vectors m or m'.

Compared with other handcrafted descriptors, the complexities of GOG/ZOZ are higher than LOMO [11], whereas they are lower than gBiCov [34]. The complexities of LOMO are of a linear order *w.r.t.* spatial parameters and the dimensionality of pixel features because LOMO takes maximal values of local histograms along horizontal strips. gBiCov calculates the similarities of patch covariance matrices between nearby scales of biologically inspired features (BIF) images. A similarity calculation step requires an eigendecomposition. Thus, these complexities are similar to the steps (i)-(iii) of HGDs, but are repeated for each of S BIF images. Hence, the complexities of gBiCov are about S times of the steps (i)-(iii) of HGDs. Note that the running times of the steps (iv)-(v) of HGDs are nearly the same as steps (ii)-(iii) because G is much



(a) PoEM (b) LERM (c) MPN

Fig. 13. Comparison of eigenvalue transformation: (a) PoEM; (b) LERM; (c) MPN.

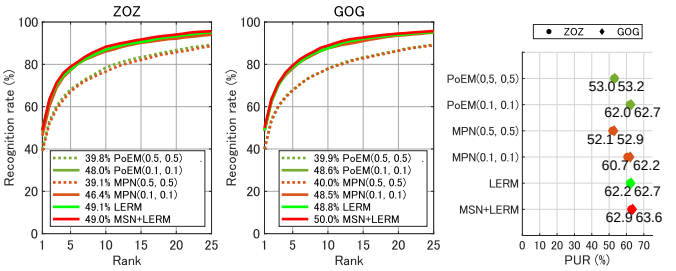


Fig. 14. Comparison of Riemannian metrics on the VIPeR dataset. The numbers of the MPN/PoEM are the parameters $(\gamma, \gamma^{\mathcal{G}})$. The numbers on the CMC curves indicate the rank-1 rates.

smaller than N_p, whereas m is much larger than d.

Running time

We implemented HGDs in MATLAB[10] with MEX functions for calculation of the covariance matrices, and ran on a PC equipped with an Intel Xeon E5-2687W v3 @3.1 GHz CPU.

The training time and memory usage for the I-L2 normalization on one training split of the VIPeR dataset were approximately 10.7 and 7.2 minutes and 4 and 3.8GB for GOG and ZOZ, respectively. On other datasets, they increase linearly against the number of training samples.

The running time to extract GOG and ZOZ with the E-L2/I-L2 normalizations was approximately 0.92/1.03 and 0.77/0.85 seconds per one image, respectively. The memory usage of the feature extraction was below 70 MB, excluding approximately 500 MB required to run the MATLAB environment itself.

D. Comparison with the matrix power normalization

The use of an appropriate Riemannian metric plays a central role in HGDs. Recently, Matrix Power Normalization (MPN) [65] has shown superiority against LERM on CNN feature embeddings [64], [66]. In an interpretation of the Riemannian metric, MPN corresponds to the Power Euclidean Riemannian Metric (PoEM) [67]. This appendix employs these methods in the framework of HGDs and compares the performance with LERM and MSN.

PoEM and MPN.

In PoEM, the distance between two symmetric positive semi-definite matrices[11] $\boldsymbol{X}, \boldsymbol{Y} \in Sym_e^+$ is defined as follows:

$$d_\gamma(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{\gamma}\|\boldsymbol{X}^\gamma - \boldsymbol{Y}^\gamma\|_F, \tag{4}$$

where $\boldsymbol{X}^\gamma$ is the MPN defined as

$$\boldsymbol{X}^\gamma = \boldsymbol{U}\mathrm{Diag}(\lambda_\mathrm{i}^\gamma)\boldsymbol{U}^\mathrm{T}, \quad \gamma > 0. \tag{5}$$

Here $\boldsymbol{X} = \boldsymbol{U}\mathrm{Diag}(\ln\lambda_\mathrm{i})\boldsymbol{U}^\mathrm{T}$ is an eigendecompostion of $\boldsymbol{X}$ and γ is a parameter of a positive real number.

It is known that PoEM approximates LERM as follows [65]: $\lim_{\gamma\to 0} d_\gamma(\boldsymbol{X}, \boldsymbol{Y}) = \|\log\boldsymbol{X} - \log\boldsymbol{Y}\|_F$. This correspondence is because $d_\gamma(\boldsymbol{X}, \boldsymbol{Y}) = \|\frac{1}{\gamma}(\boldsymbol{X}^\gamma - \boldsymbol{I}) - \frac{1}{\gamma}(\boldsymbol{Y}^\gamma - \boldsymbol{I})\|_F$, where $\frac{1}{\gamma}(\boldsymbol{X}^\gamma - \boldsymbol{I}) = \boldsymbol{U}\mathrm{Diag}\left(\frac{\lambda_i^\gamma - 1}{\gamma}\right)\boldsymbol{U}^T$ and $\lim_{\gamma\to 0}\frac{\lambda_i^\gamma - 1}{\gamma} = \ln\lambda_i$. Thus, we define the associated approximate tangent space mapping of PoEM as the following equation:

$$\log\boldsymbol{X} \approx \frac{1}{\gamma}(\boldsymbol{X}^\gamma - \boldsymbol{I}). \tag{6}$$

10. Will be released at http://www.i.kyushu-u.ac.jp/~matsukawa/ReID.html.

11. PoEM allows positive and zero eigenvalues; thus it is defined for symmetric positive semi-definite matrices.

A notable property of MPN and PoEM is that they allow zero eigenvalues. Following the previous work [65], we use MPN and PoEM without regularizing the SPD matrix. Because LERM is sensitive to small eigenvalues, we use the regularizer ϵ for LERM.

Fig. 13 shows how an eigenvalue is transformed by the parameters for PoEM, LERM, and MPN. PoEM and LERM reverse the order of the original eigenvalues and magnify the effects of small eigenvalues. Fig. 13 (a) shows that as γ increases, PoEM reduces the magnifying effects of a small eigenvalue of a logarithmic function. Fig. 13 (b) shows that ϵ for LERM also limits the smallest values in a transformed eigenvalue, and this behavior is similar to the effects of γ in PoEM. In contrast to PoEM and LERM, MPN reserves the order of original eigenvalues. Fig. 13 (c) shows that a larger positive value is transformed into a larger value in MPN.

We evaluate the performance of GOG and ZOZ when we use MPN and PoEM instead of LERM. We normalize the features by the E-L2 normalization without PN and evaluate the performance with the XQDA metric. We examine the best parameters for patch and region Gaussians $(\gamma, \gamma^{\mathcal{G}})$ from all combinations of $\gamma, \gamma^{\mathcal{G}} \in \{0.5, 0.25, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. Based on these results, we use $(\gamma, \gamma^{\mathcal{G}}) = (0.1, 0.1)$. Additionally, following the previous work [65], we also use $(\gamma, \gamma^{\mathcal{G}}) = (0.5, 0.5)$. For reference, we show the performance of LERM and MSN which use $(\epsilon, \epsilon^{\mathcal{G}}) = (10^{-4}, 10^{-2})$. Fig. 14 shows the results on the VIPeR dataset.

The results indicate that: (1) PoEM shows comparable performance to LERM. This result is probably because the transformed eigenvalues are similar between PoEM ($\gamma = 0.1$) and LERM ($\epsilon = 10^{-4}$) (Fig. 13 (a) and (b)). In addition, LERM with MSN outperforms PoEM. This result is because PoEM only approximates LERM and has no ability to adjust the diagonal matrix elements as in MSN. (2) The performance of MPN is slightly inferior to that of PoEM and LERM. The difference between the superiority of MPN on the embedding of CNN features [64], [66] is probably due to the difference of the input features. CNN captures the high-level concept of images, and the higher eigenvalues of feature distribution might be significant. In contrast, HGDs use low-level color and gradient features, and thus a subtle difference in feature distribution might be significant to distinguish different persons.

E. Comparison with other Gaussian embeddings

This appendix employs other Gaussian embeddings to the base embedding of HGDs and compares their performances.

The Lovrić embedding [29] includes a mean vector and a covariance matrix in one SPD matrix. To examine the effectiveness of the joint embedding, we carry out the comparison with a simple concatenation of the mean vector and the covariance matrix.

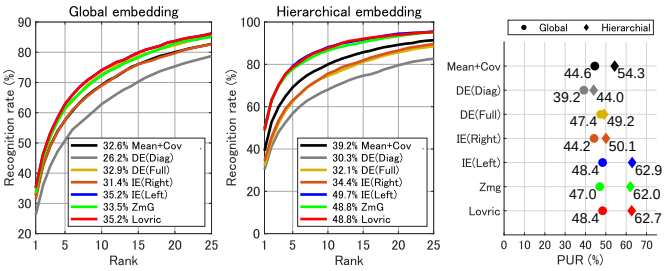


Fig. 15. Comparison of Gaussian embeddings on the VIPeR dataset: The numbers on the CMC curves indicate the rank-1 rates.

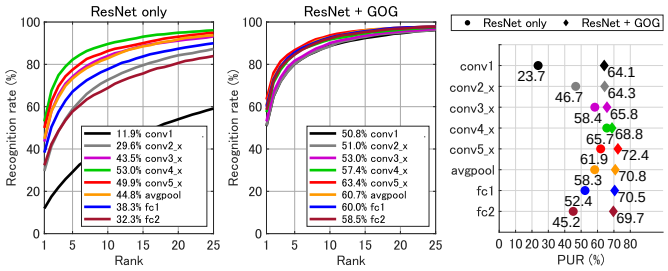


Fig. 16. Analysis of feature extraction layers of ResNet on the VIPeR dataset: The numbers on the CMC curves indicate the rank-1 rates.

Additionally, we use two types of Lie group embeddings [55], both types are based on a subgroup of $(d+1)$-dimensional upper triangular matrices $A^+(d+1)$. The first type, Direct Embedding (DE), maps $A^+(d+1)$ into a linear space via matrix logarithm. Within DE, we evaluate the following two embeddings, DE using Full-covariance matrices (DE (Full)) and its analytic simplification for Diagonal-covariance matrices (DE (Diag)). The second type, Indirect Embedding (IE), first maps $A^+(d+1)$ into Sym^+_{d+1} via coset and polar decomposition and then into a linear space by LERM. For IE, we use two embeddings, Left coset (IE (Left)) and Right coset (IE (Right)) embeddings. The representations of these compared embeddings in Euclidean space are defined as follows:

- Mean + Cov $\coloneqq \left[\boldsymbol{\mu}^T \;\; \mathrm{vec}(\log\boldsymbol{\Sigma})^T\right]^T$,
- DE (Diag) $\coloneqq \left[\ln\sigma_1, \ldots, \ln\sigma_d, \frac{\mu_1 \ln\sigma_1}{\sigma_1 - 1}, \ldots, \frac{\mu_d \ln\sigma_d}{\sigma_d - 1}\right]^T$,
- DE (Full) $\coloneqq \log \begin{bmatrix} \boldsymbol{L}^{-T} & \boldsymbol{\mu} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$,
- IE (Right) $\coloneqq \log \begin{bmatrix} \boldsymbol{L}^{-1}\boldsymbol{L}^{-T} & \boldsymbol{L}^{-1}\boldsymbol{\mu} \\ \boldsymbol{\mu}^T\boldsymbol{L}^{-T} & \boldsymbol{\mu}^T\boldsymbol{\mu}+1 \end{bmatrix}^{\frac{1}{2}}$,
- IE (Left) $\coloneqq \log \boldsymbol{G}^{\frac{1}{2}} = \log \begin{bmatrix} \boldsymbol{\Sigma}+\boldsymbol{\mu}\boldsymbol{\mu}^T & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix}^{\frac{1}{2}}$.

Here σ_i is a square-root of the i-th diagonal element of $\boldsymbol{\Sigma}$ and $\boldsymbol{L}$ is the Cholesky factor of $\boldsymbol{\Sigma}^{-1}$. We normalize the descriptors by the E-L2 normalization without MSN and PN, and compare the performance with XQDA metric. Fig. 15 shows the results on the VIPeR dataset.

The results indicate that: (1) The joint Gaussian embeddings (IE (Left), ZmG, Lovric) outperform Mean + Cov. These results confirm that considering the geometry of Gaussian distribution is more significant than separately considering the geometry of the mean vectors and covariance matrices. (2) The diagonal Gaussian embedding (DE (Diag)) shows the lowest performance. These results show that the diagonal assumption of covariance matrices overly simplifies the distribution of pixel features for HGDs. (3) Among the Lie group embeddings, IE (Left) shows the best performances. When applied to the global embedding, DE (Full) and IE (Right) show slightly lower performance than IE (Left), whereas DE (Full) showed slightly higher performance than IE (Right). These trends are consistent with the results of Li *et al.* [55]. (4) Lovrić and IE (Left) produce mostly the same results. These results are because the matrix logarithm on square-rooted matrix results in only the rescaled version of the matrix logarithm, *i.e.*, $\log\boldsymbol{G}^{\frac{1}{2}} = \frac{1}{2}\log\boldsymbol{G}$. Because the E-L2 normalization cancels the effects of $\frac{1}{2}$, the results of LE (Left) and Lovrić are the same in the case of global embedding. In the hierarchical embedding, the E-L2 normalization cancels the effects of $\frac{1}{2}$ on region Gaussians, whereas the effects on patch Gaussians causes slight changes in performance.

F. Details of ResNet features

We trained a 50-layered ResNet using a triplet loss function. For the implementation, we used the default setting of open-reid[12] which is a PyTorch re-implementation of the paper [78]. Our focus is to investigate if HGDs work complementary with CNN features. Because several datasets lack a sufficient amount of training samples for deep learning, we adopt the metric learning approaches [14], [45], which use CNN features trained on another large-scale dataset. We trained the network on the training split of the Market-1501 dataset.

We examine suitable layers for feature extraction. A 50-layered ResNet consists of one convolutional layer (conv1), four bottleneck building blocks[13] (conv2_x, conv3_x, conv4_x, and conv5_x), one average pooling layer (avgpool), and one fully connected layer [77]. In open-reid, input images are resized to 256×128 pixels and one fully connected layer (fc1) is added before the last fully connected layer (fc2).

We modified the code to access the output of these layers/blocks. The convolutional layer and the bottleneck building blocks output feature maps. The sizes (height × width × feature dimension) of them are 128×64×64 (conv1), 64×32×256 (conv2_x), 32×16×512 (conv3_x), 16×8×1,024 (conv4_x), and 8×4×2,048 (conv5_x), respectively. To form feature vectors from the feature maps, we summarize features along seven horizontal strips determined in a similar way as HGDs by average pooling. For the implementation, we use the avgpool2d function of PyTorch, in which the height and width of the pooling kernel correspond to the one-quarter of the feature map height, and the feature map width, respectively. We set the strides in height and width directions of the avegpool2d function to the half of the kernel height, and the kernel width such that each horizontal pooling area overlaps with another area by half. We reshape the outputs of the avgpool2d function (7×1× feature dimension) into vectors. In this way, we obtain 448-dimensional feature vectors for conv1 layer, and 1,792, 3,584, 7,168, and 14,336-dimensional feature vectors for conv2_x, conv3_x, conv4_x, and conv5_x blocks, respectively. Other layers, avgpool, fc1, and fc2 directly output 2,048, 1,024, and 128-dimensional feature vectors, respectively.

Subsequently, we normalize the L2 norm of the feature vectors after removing the mean values on the training set of each applied dataset. We evaluate individual performances of each layer with

12. https://github.com/Cysu/open-reid

13. A bottleneck building block consists of several convolutional layers and a skip connection [77]. We use the output of each block.

XQDA metric. Additionally, we compare the performance when combined with the GOG descriptor. Fig. 16 shows the results on the VIPeR dataset.

The results indicate that: (1) When using the ResNet features alone, conv4_x features perform the best. (2) When combined with the GOG descriptor, the features of the upper layers outperform the conv4_x features. We also combined CNN features with the ZOZ descriptor, and the trends were similar. These results show that HGDs are highly complementary with high-level features of CNNs. Features of fully connected layers may overfit the trained dataset of CNN. Experimental results showed the conv5_x features performed the best when combined with the GOG descriptor. On the basis of these observations, we used the conv5_x features for the baseline CNN features of ResNet.

G. Parameter of XQDA

XQDA contains a regularization parameter of a covariance matrix to learn the Mahalanobis metric [11]. It is suggested that when the norm of features is normalized, the regularization parameter 10^{-3} is effective. In the comparison in §5.3, we evaluate the case when the norm is not normalized. In addition, without the bias removal methods, a large bias that exists in HGDs results in the sample distances of L2 normalized features becoming overly small. Additionally, PN changes the distribution of the sample distances. We confirmed the default setting in the public code of XQDA causes low performance for these normalizations.

For each normalization, we investigated the best parameters that generally work well among different datasets from the range $\{10^{-5}, 10^{-4}, \ldots, 10^{1}\}$. The best parameters for {None, L2, E-L2, E*-L2, I-L2} were $\{10^{-1}, 10^{-4}, 10^{-3}, 10^{-3}, 10^{-3}\}$, respectively, without PN, and $\{10^{0}, 10^{-4}, 10^{-3}, 10^{-3}, 10^{-3}\}$, respectively, with PN. We used these values in the evaluations.

Acknowledgements

We thank the anonymous reviewers and the associate editor for their valuable comments to improve this paper. This work was supported by the "R&D Program for Implementation of Anti-Crime and Anti-Terrorism Technologies for a Safe and Secure Society," under the fund for the integrated promotion of social system reform and research and development of MEXT Japan, JSPS KAKENHI JP15K16028, and JP17K20008.