Supplementary Materials: Reliability-Aware Prediction via Uncertainty Learning for Person Image Retrieval

Zhaopeng Dou[1], Zhongdao Wang[1], Weihua Chen[2], Yali Li[1] and Shengjin Wang[1]✉

[1] Department of Electronic Engineering and BNRist, Tsinghua University, China
[2] Machine Intelligence Technonlogy Lab, Alibaba Group
dcp19@mails.tsinghua.edu.cn, wgsgj@tsinghua.edu.cn

Concents

1 Quality-Degrading Transformations

As described in Section 4.3 in the main text, to simulate the complex scenes, we transform the query images via different quality-degrading transformations, including: (1) adding Gaussian noise to mimic camera quality differences; (2) cropping the image to simulate pedestrians are partially out of the camera's field of view; (3) adding motion blur to mimic fast-moving pedestrians; (4) adding fog to imitate complex weather. Here, we detail these transformations.

Adding Gaussian noise. For an image tensor, *i.e.*, $\boldsymbol{x}$, we generate a noise tensor $\boldsymbol{\epsilon}$ that is sampled from Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, *i.e.*, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then, we pollute a query sample by $\boldsymbol{x} \leftarrow \boldsymbol{x} + \eta\boldsymbol{\epsilon}$ with a probability of p. **In the experiments in the main text, we set** $p = 0.5$ **and** $\eta = 1.0$**.** Fig. 1 shows some images with Gaussian noise added. It can be seen that with the increase of noise strength η, the quality of the image decreases.

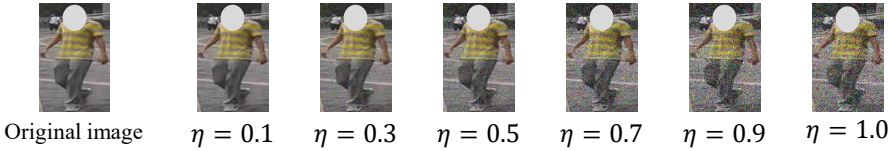

Fig. 1: Images with Gaussian noise added. When we increase the noise strength η, the quality of the image decreases.

Cropping image. For an image tensor $\boldsymbol{x}$, suppose the spatial size of it is (h, w), where h and w are the height and width, respectively. Then, with a probability of p, we crop $\boldsymbol{x}$ along the height (from top to bottom) to generate the new image tensor $\boldsymbol{x}'$ whose size is $(\xi * h, w)$, where $\xi \in [0, 1]$. Finally, $\boldsymbol{x}'$ will be resized to (h, w). **In the experiments of the main text, we set** $p = 0.8$ **and** $\xi = 0.5$**.** Fig. 2 shows some cropped images. It can be seen that with the decrease of the crop ration ξ, the discriminative information contained by the image decreases.

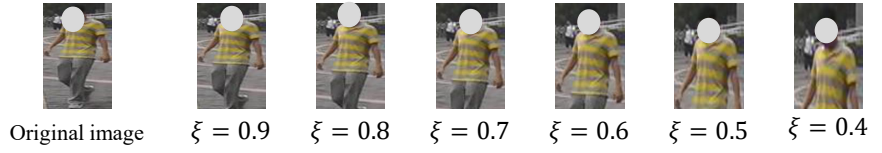

Fig. 2: Cropped images. With the decrease of the crop ratio ξ, the discriminative information contained by the image decreases.

Adding motion blur. We generate the kernel matrix of the motion blur to pollute the image with a probability of p. **In the experiments of the main**

text, we set $p = 0.5$. The corresponding python code is shown in Listing 1.1. There are two main parameters when adding motion blur to the image, *i.e.*, "degree" and "angle". The "degree" and "angle" control the strength and the direction of the motion, respectively. **In the experiments of the main text, we set them as 30 and 10, respectively.** Fig. 3 shows some motion-blurred images when varying the parameter "degree". It can be seen that with the increase of the strength of the motion blur, the quality of the image decreases.

Listing 1.1: The python code of adding motion blur

```
import numpy as np
import cv2
from PIL import Image

def Addmotionblur(image, degree=30, angle=10):
    """
    image: PIL image
    degree: the strength of the motion blur
    angle: the angle of the motion blur
    """
    image = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)
    image = np.array(image)

    M = cv2.getRotationMatrix2D((degree / 2, degree / 2), angle, 1)
    motion_blur_kernel = np.diag(np.ones(degree))

    motion_blur_kernel = cv2.warpAffine(motion_blur_kernel,M,(degree, degree))
    motion_blur_kernel = motion_blur_kernel / degree

    blurred = cv2.filter2D(image, -1, motion_blur_kernel)
    cv2.normalize(blurred, blurred, 0, 255, cv2.NORM_MINMAX)

    blurred = np.array(blurred, dtype=np.uint8)
    blurred = Image.fromarray(cv2.cvtColor(blurred, cv2.COLOR_BGR2RGB))
    return blurred
```

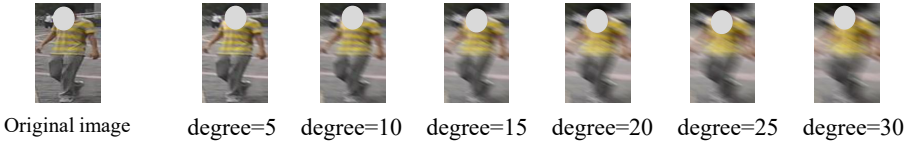

Fig. 3: Motion-blurred images when varying the strength of the motion blur, *i.e.*, "degree". The parameter "angle" is fixed to 10. Note that with the increase of the strength of the motion blur, the quality of the images decreases.

Adding fog. The corresponding python code of adding fog is shown in Listing 1.2. There are two main parameters when adding fog to the image, *i.e.*, "beta" and "fog_size". The "beta" controls the concentration of the fog. The "fog_size" controls the scope of the fog. **In the experiments of the main text, they are set as 0.3 and 10, respectively.** An image will be added fog with a probability of p. **We set $p = 0.5$ in the experiments of the main paper.** Fig. 4 shows some images with fog added when vary the parameter "beta" and fix the parameter "fog_size". It can be seen that with the increase of the concentration of the fog, the image contains more ambiguous information.

Listing 1.2: The python code of adding fog

```
import numpy as np
import cv2
from PIL import Image

def Addfog(img, beta=0.1, fog_size=8):
    """
    input: PIL image, the size will be resized as (128, 256)
    beta: the concentration of the fog
    fog_size: the scope of the fog
    """
    img = img.resize((128, 256))

    img = cv2.cvtColor(np.array(img), cv2.COLOR_RGB2BGR)
    img = np.array(img)
    img_f = img / 255.0
    (row, col, chs) = img.shape

    A = 1.0
    center = (row // 40, col // 2)
    for j in range(row):
        for l in range(col):
            d=-0.04*math.sqrt((j-center[0])**2 + (l-center[1])**2) + fog_size
            d = d if d > 0 else 0
            td = math.exp(-beta * d)
            img_f[j][l][:] = img_f[j][l][:] * td + A * (1 - td)

    img_f = np.array(img_f * 255, dtype=np.uint8)
    img_f = Image.fromarray(cv2.cvtColor(img_f, cv2.COLOR_BGR2RGB))
    return img_f
```

Fig. 4: Images with fog added. We vary the parameter "beta" that controls concentration of the fog. The parameter "fog_size" that controls the scope of the fog is fixed as 10. Note that with the increase of the concentration of the fog, the image contains more ambiguous information.

2 Extensive Results on Multi-Query Settings

2.1 Multi-Query Settings on Noisy Samples

As described in Section 4.3 in the main paper, we reconstruct the test set of Market-1501 [3] to validate the effectiveness of our method on multi-query settings. We collect the images belonging to the same personal identity and same camera identity from the query set and gallery set. Then, we randomly select half of these images to be allocated to the reorganized query set and the other half to the reorganized gallery set. Simultaneously, to simulate the complex scenes

in reality, we add Gaussian noise to images in the query set. Specifically, for an image tensor, *i.e.*, $\boldsymbol{x}$, we generate a noise tensor $\boldsymbol{\epsilon}$ that is sampled from Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Then we pollute a query sample by $\boldsymbol{x} \leftarrow \boldsymbol{x} + \eta\boldsymbol{\epsilon}$ with a probability of 0.5. Here, we vary the noise strength η to observe how the performance of our method changes.

As described in Section 4.3 in the main paper, the reliability score of the i-th query is calculated by $w_i = \frac{m_i d_i}{\sum_{j=1}^{n} m_j d_j}$. When we only use the data uncertainty to calculate the reliability score, the m_* is set to 1. When we only use the model uncertainty to calculate the reliability score, the d_* is set to 1.

Table 1: Results of the multi-query settings on noisy samples when varying the noise strength η. "***w/*** R" ("***w/o*** R") means we use (don't use) the reliability score to adjust the weights for different queries.

Method	Uncertainty		Noise strength η							
			$\eta = 0.3$		$\eta = 0.5$		$\eta = 0.7$		$\eta = 1.0$	
	Data	Model	Market		Market		Market		Market	
			R1	mAP	R1	mAP	R1	mAP	R1	mAP
Ours (***w/o*** R)	×	×	94.1	87.1	81.3	73.3	67.2	61.4	55.0	33.1
Ours (***w/*** R)	✓	×	**94.3**	**87.4**	83.1	76.5	75.3	69.6	71.4	66.7
	×	✓	94.1	87.3	82.9	76.4	75.3	69.6	70.7	66.5
	✓	✓	**94.3**	**87.4**	**83.9**	**77.5**	**77.7**	**72.1**	**76.7**	**71.4**

Results are shown in Table 1. From the results, we can make several observations. First, when varying the noise strength η, our method "Ours (***w/*** R)" stably outperforms the vanilla method "Ours (***w/o*** R)", which shows the proposed reliability assessment is credible and our method is effective for the multi-query settings. Second, when combing the data uncertainty and model uncertainty, the performance is higher than using data uncertainty alone or model uncertainty alone. This shows these two types of uncertainty can provide complementary information to suppress the negative influence of unreliable predictions. Third, with the increase of η, the margin between our method and the vanilla method grows. This shows that our method has greater advantages in complex scenes.

2.2 Multi-Query Settings on Partial Samples

We also verify our method on the partial samples. As described in Sec. 1, we generate the partial images through cropping the original images. We vary the crop ration ξ to observe how the performance of our changes. Results are shown in Table 2. From the results, we can draw similar conclusions to the experiments on noisy samples. Especially, with the decrease of the crop ratio ξ, the discriminative information contained by the image decreases, and the margin between our method "Ours (***w/*** R)" and the vanilla method "Ours (***w/o*** R)" grows. This

further shows the proposed reliability assessment is credible and our method is more effective in more complex scenes.

Table 2: Results of the multi-query settings on partial samples when varying the crop ratio ξ. "***w/*** R" ("***w/o*** R") means we use (don't use) the reliability score to adjust the weights for different queries.

Method	Uncertainty		Crop ratio ξ							
			$\xi = 0.9$		$\xi = 0.7$		$\xi = 0.5$		$\xi = 0.3$	
	Data	Model	Market		Market		Market		Market	
			R1	mAP	R1	mAP	R1	mAP	R1	mAP
Ours (***w/o*** R)	×	×	96.7	90.9	86.0	76.8	37.2	35.7	25.6	23.7
Ours (***w/*** R)	✓	×	96.8	90.9	86.6	77.4	43.0	41.0	35.0	31.9
	×	✓	96.8	90.8	86.2	77.3	44.1	42.1	37.3	33.6
	✓	✓	**96.9**	**91.0**	**87.0**	**77.9**	**46.3**	**44.4**	**41.4**	**37.8**

2.3 Multi-Query Settings on Motion-Blurred Samples

We further verify our method on the motion-blurred samples. The details of generating the motion-blurred images are described in Sec. 1. We vary the strength of the motion blur ("degree") to observe how the performance of our changes. Results are shown in Table 3. From the results, we can draw similar conclusions to the experiments on noisy samples. Especially, with the increase of the strength of motion blur, the quality of the image decreases, and the margin between our method "Ours (***w/*** R)" and the vanilla method "Ours (***w/o*** R)" grows. This further shows the proposed reliability assessment is credible and our method is more effective in more complex scenes.

Table 3: Results of the multi-query settings on motion-blurred samples when varying the strength of the motion blur. "***w/*** R" ("***w/o*** R") means we use (don't use) the reliability score to adjust the weights for different queries.

Method	Uncertainty		strength of motion blur: "degree"							
			"degree" = 20		"degree" = 30		"degree" = 40		"degree" = 50	
	Data	Model	Market		Market		Market		Market	
			R1	mAP	R1	mAP	R1	mAP	R1	mAP
Ours (***w/o*** R)	×	×	74.9	68.1	69.7	63.4	67.9	61.2	66.1	60.0
Ours (***w/*** R)	✓	×	75.5	69.7	73.2	67.2	71.5	66.2	70.8	65.6
	×	✓	**79.1**	73.1	76.8	70.6	75.4	69.6	75.2	69.2
	✓	✓	79.0	**73.3**	**77.4**	**71.4**	**76.4**	**70.8**	**76.1**	**70.6**

2.4 Multi-Query Settings on Samples with Fog

Finally, we verify our method on the samples with fog. The details of adding fog to the images are described in Sec. 1. We vary the strength of the concerntration of the fog ("beta") to observe how the performance of our changes. Results are shown in Table 4. From the results, we can draw similar conclusions to the experiments on noisy samples. Especially, with the increase of the concentration of the fog, the quality of the image decreases, and the margin between our method "Ours (***w/*** R)" and the vanilla method "Ours (***w/o*** R)" grows. This further shows the proposed reliability assessment is credible and our method is more effective in more complex scenes.

Last but not least, the experiments on noisy samples, partial samples, motion-blurred samples, and fogged samples all reflect the effectiveness of our method. This shows the robustness of our method in the complex scenes.

Table 4: Results of the multi-query settings on images with fog when varying the concentration of the fog. "***w/*** R" ("***w/o*** R") means we use (don't use) the reliability score to adjust the weights for different queries.

Method	Uncertainty		Concentration of the fog: "beta"							
			"beta"= 0.1		"beta"= 0.2		"beta"= 0.3		"beta"= 0.4	
	Data	Model	Market		Market		Market		Market	
			R1	mAP	R1	mAP	R1	mAP	R1	mAP
Ours (***w/o*** R)	×	×	78.9	72.2	69.4	63.7	65.1	60.3	61.9	58.2
Ours (***w/*** R)	✓	×	79.5	72.9	74.9	69.4	75.1	69.4	74.4	68.9
	×	✓	79.0	73.1	74.1	68.9	74.8	69.1	74.2	68.7
	✓	✓	**79.7**	**73.5**	**75.8**	**71.0**	**77.1**	**71.9**	**77.0**	**71.8**

2.5 Ablation Study of $\tau_{\min}$ and $\tau_{\max}$

As described in the Section 3.4 in the main paper, when applying the proposed reliability assessment to the multi-query settings, to combine the data uncertainty and model uncertainty without being affected by their numerical scale, we project the data (model) uncertainty into the interval $[\tau_{\min}, \tau_{\max}]$. The final similarity between multiple query images $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and the gallery image $\boldsymbol{y}$ is calculated by $s = \sum_{i=1}^{n} w_i s_i$, where $w_i = \frac{m_i d_i}{\sum_{j=1}^{n} m_j d_j}$, $d_i \in [\tau_{\min}, \tau_{\max}]$ ($m_j \in [\tau_{\min}, \tau_{\max}]$) is the mapped data (model) uncertainty of $\boldsymbol{x}_i$ and s_i is the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{y}$. As the reliability score of each query image, *i.e.*, w_i, is normalized, the final similarity is related to the ***relative size*** of $\tau_{\min}$ and $\tau_{\max}$. Here, we fixed the $\tau_{\max}$ at 1.0 and varying the value of $\tau_{\min}$ from 0.1 to 0.9 to see the performance of our method on the synthetic partial dataset (cropped Market-1501 [3], described in Sec. 1). Results are shown in Table 5. From the results, we can see that, for small ξ (*e.g.*, 0.3, 0.4, 0.5), our method shows higher performance when adopting small $\tau_{\min}$ (*e.g.*, 0.1). For large ξ (eg, 1.0, 0.9, 0.8), our method shows higher performance when adopting large $\tau_{\min}$ (*e.g.*, 0.9, 0.7,

Table 5: Ablation study of $\tau_{\min}$ under the multi-query settings. $\tau_{\max}$ is fixed at 1.0. Results are conducted on cropped Market-1501 [3] (partial dataset). "***w/*** R" ("***w/o*** R") means we use (don't use) the reliability score to adjust the weights for different queries.

Method	$\tau_{\min}$	Crop Ratio ξ															
		$\xi = 1.0$		$\xi = 0.9$		$\xi = 0.8$		$\xi = 0.7$		$\xi = 0.6$		$\xi = 0.5$		$\xi = 0.4$		$\xi = 0.3$	
		R1	mAP	R1	mAP	R1	mAP	R1	mAP	R1	mAP	R1	mAP	R1	mAP	R1	mAP
Ours (*w/o* R)	–	97.0	**92.2**	96.7	**91.0**	93.8	86.8	86.0	76.8	63.0	56.4	37.2	35.7	29.3	27.6	25.6	23.7
Ours(*w/* R)	0.1	96.0	90.3	95.4	89.0	93.0	85.2	84.5	75.8	64.6	59.5	**47.5**	**46.1**	**45.6**	**42.6**	**44.3**	**40.9**
	0.3	96.5	91.5	96.3	90.3	93.7	86.3	85.8	76.9	65.0	**59.8**	47.1	45.6	44.8	41.8	43.0	39.9
	0.5	**97.2**	92.1	**96.9**	**91.0**	94.0	87.0	**86.7**	77.5	**65.5**	59.6	46.3	44.4	43.1	40.0	41.4	37.8
	0.7	97.0	92.2	96.8	**91.0**	94.0	**87.2**	86.3	**77.6**	65.2	59.0	43.9	41.9	39.2	36.3	36.8	33.4
	0.9	97.1	**92.3**	96.5	**91.0**	**94.1**	**87.2**	86.4	77.2	63.9	57.5	40.0	38.1	32.6	30.7	29.1	27.0

0.5). This is because that, when ξ is small, a certain percentage of query images are severely cropped and they may provide ambiguous information to degrade the final ranking. Our method will generate large data uncertainty and model uncertainty to these severely cropped images. At this time, a small $\tau_{\min}$ means that these severely cropped images are assigned small weights, which effectively suppresses the impact of them and make the final similarity more dependent on the relatively complete query images, thus shows better performance. However, when ξ is large, a certain percentage of query images are slightly cropped and they can also provide discriminative information for ranking. If we adopt a too small $\tau_{\min}$, these images will be assigned small weights, which may waste the useful information provided by them and obtain fallen performance. At this time, a large $\tau_{\min}$ can maintain the discriminative information from these slightly cropped images and results in better performance. In conclusion, for unconstrained complex where the query images have large quality and visibility variation, a small $\tau_{\min}$ is more suitable. For constrained simple scenes where the query samples have small quality and visibility variation, a large $\tau_{\min}$ is more appropriate.

3 Visualization

3.1 Visualization of the Quality-Degraded Images and Uncertainty

Here, we visualize the quality-degraded images and their corresponding data uncertainty and model uncertainty. Results are shown in Fig. 5, Fig. 6, Fig. 7 and Fig. 8. We can see that, for different quality-degrading transformations, with the quality of the images degrades, the corresponding data uncertainty and model uncertainty grow. That is, when the discriminative information provided by an image decreases, our method will assign a lower reliability score to the prediction of this sample.

Noise strength	$\eta = 0.0$	$\eta = 0.3$	$\eta = 0.5$	$\eta = 0.7$	$\eta = 1.0$
Images					
Model uncertainty ($\times 10^{-6}$)	3.3890	4.0746	5.1533	6.9717	9.2480
Data uncertainty	0.8100	0.8131	0.8290	0.8573	0.8693

Fig. 5: Visualization of images polluted by Gaussian noise when varying the noise strength η, as well as corresponding model uncertainty and data uncertainty.

Crop ratio	$\xi = 1.0$	$\xi = 0.9$	$\xi = 0.7$	$\xi = 0.5$	$\xi = 0.4$
Images					
Model uncertainty ($\times 10^{-6}$)	3.3890	3.6961	5.2148	5.6303	6.1325
Data uncertainty	0.8100	0.8120	0.8209	0.8214	0.8216

Fig. 6: Visualization of cropped images when varying the crop ration ξ, as well as corresponding model uncertainty and data uncertainty.

"degree" of motion blur	degree = 0	degree = 10	degree = 15	degree = 20	degree = 25
Images					
Model uncertainty ($\times 10^{-6}$)	3.3890	5.7457	8.2438	9.6785	10.2341
Data uncertainty	0.8100	0.8153	0.8239	0.8285	0.8322

Fig. 7: Visualization of cropped motion-blurred images when varying the "degree" of the motion blur, as well as corresponding model uncertainty and data uncertainty.

"beta" of fog	beta = 0	beta = 0.1	beta = 0.2	beta = 0.3	beta = 0.4
Images					
Model uncertainty ($\times 10^{-6}$)	3.3890	4.3682	5.1344	6.1897	6.6359
Data uncertainty	0.8100	0.8127	0.8289	0.8553	0.8661

Fig. 8: Visualization of samples with fog when varying the concentration of the fog ("beta"), as well as corresponding model uncertainty and data uncertainty.

3.2 Visualization of Retrieval Results

In this part, we visualize some retrieval results under the multi-query settings. Results are shown in Fig. 9, Fig. 10, Fig. 11 and Fig. 12. In Fig. 9, some query samples are polluted by the Gaussian noise. In Fig. 10, some query images are cropped to imitate the partial ReID scenarios. In Fig. 11, some query samples are motion-blurred. In Fig. 12, some query samples are added with fog. Each query set contains the images from the same personal identity and same camera identity. We display the top-8 ranking results. We report the results of "Ours (***w/*** R)" against the vanilla method "Ours (***w/o*** R)". We can see that "Ours (***w/*** R)" shows better performance, which stems from the effective reliability assessment. Specifically, our method suppresses the negative impact of noisy query images, severely partial query images, motion-blurred images and fogged images by assigning them large data uncertainty and model uncertainty (low reliability score), and simultaneously emphasizes the influence of high-quality and relatively complete query images.

Fig. 9: Visualization of the retrieved results under the multi-query settings. Some query images are polluted by **Gaussian noise**. Comparing "Ours (***w/o*** R)" and "Ours (***w/*** R)", it can be seen that when we use the reliability to adjust the weights of different queries, the results are more accurate and reliable.

Fig. 10: Visualization of retrieval results under the multi-query settings. Some query images are **partial**. Comparing "Ours (***w/o*** R)" and "Ours (***w/*** R)", it can be seen that when we use the reliability to adjust the weights of different queries, the results are more accurate and reliable.

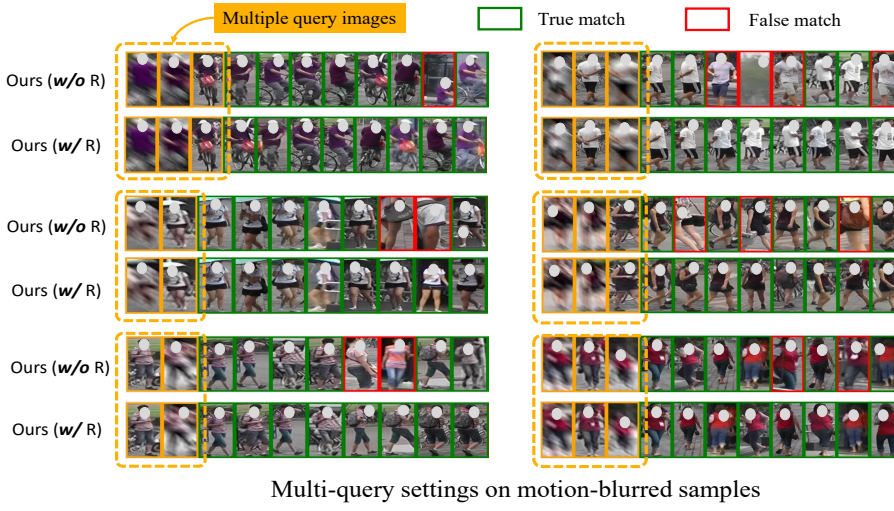

Fig. 11: Visualization of retrieval results under the multi-query settings. Some query images are **motion-blurred**. Comparing "Ours (***w/o*** R)" and "Ours (***w/*** R)", it can be seen that when we use the reliability to adjust the weights of different queries, the results are more accurate and reliable.

Fig. 12: Visualization of retrieval results under the multi-query settings. Some query images are added **fog**. Comparing "Ours (***w/o*** R)" and "Ours (***w/*** R)", it can be seen that when we use the reliability to adjust the weights of different queries, the results are more accurate and reliable.

4 Architectures of Modules ϕ, φ_μ and φ_σ

As described in the Section 3.3 in the main paper, modules ϕ, φ_μ and φ_σ are all light weighted. Here, we show their architecture in Tabel 6. "Conv(1×1, 2048, 1024)" represents a convolutional layer whose kernel size is 1×1, input channel is 2048 and output channel is 1024. "BN" represents the batch normalization layer. "ReLU" represents the rectified linear unit function. "Bayesian Conv" means that the parameters of the convolutional layer follow a Bernoulli distribution, *i.e.*, $\boldsymbol{\theta}_{ij} = \boldsymbol{\pi}_{ij} * \boldsymbol{z}_{ij}$, where $\boldsymbol{z}_{ij} \sim \text{Bernoulli}(\rho)$. In our experiments, we empirically set ρ as 0.7.

5 Derivation Details of Data Uncertainty Loss (Eq.(5))

In this part, we show the derivation details of Equation (5) in the main paper. For a sample $\boldsymbol{x}$, we project it into a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I})$ in the latent space. Assuming a sample $\boldsymbol{x}$ is from the i-th class, the posterior of $\boldsymbol{x}$ belonging to i-th class is formulated by,

$$p(\boldsymbol{x}|y=i) \propto \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \exp\left(-\frac{\|\boldsymbol{\mu} - \boldsymbol{w}_i\|^2}{2\sigma^2}\right) \tag{1}$$

Table 6: Architectures of ϕ, φ_μ and φ_σ on different backbones. Conv: convolutional layer. BN: batch normalization layer. ReLU: rectified linear unit function. Bayesian Conv: convolutional layer whose the parameters follow Bernoulli distribution.

Backbone	Module	Architecture	Backbone	Module	Architecture
ResNet50 [1]	ϕ	Bayesian Conv(1×1, 2048, 1024) $\rightarrow$ BN()$\rightarrow$ ReLU() $\rightarrow$ Bayesian Conv(1×1, 1024, 512) $\rightarrow$ BN() $\rightarrow$ ReLU() $\rightarrow$ Bayesian Conv(1×1, 512, 2048)	HRNet-W32 [2]	ϕ	Bayesian Conv(1×1, 1024, 512) $\rightarrow$ BN()$\rightarrow$ ReLU() $\rightarrow$ Bayesian Conv(1×1, 512, 512) $\rightarrow$ BN() $\rightarrow$ ReLU() $\rightarrow$ Bayesian Conv(1×1, 512, 1024)
	φ_μ	Conv(1×1, 2048, 1024) $\rightarrow$ BN() $\rightarrow$ ReLU() $\rightarrow$ Conv(1×1, 1024, 512) $\rightarrow$ BN() $\rightarrow$ ReLU() $\rightarrow$ Conv(1×1, 512, 2048)		φ_μ	Conv(1×1, 1024, 512) $\rightarrow$ BN() $\rightarrow$ ReLU() $\rightarrow$ Conv(1×1, 512, 512) $\rightarrow$ BN() $\rightarrow$ ReLU() $\rightarrow$ Conv(1×1, 512, 1024)
	φ_σ	Conv(1×1, 2048, 1024) $\rightarrow$ BN() $\rightarrow$ ReLU() $\rightarrow$ Conv(1×1, 1024, 2048)		φ_σ	Conv(1×1, 1024, 256) $\rightarrow$ BN() $\rightarrow$ ReLU() $\rightarrow$ Conv(1×1, 256, 1024)

where $\boldsymbol{w}_i$ is the weight vector of i-th class in the classifier and d is the feature dimension. Assuming each class has the equal prior probability, the posterior of $\boldsymbol{x}$ belonging to the class i is,

$$p(y=i|\boldsymbol{x}) = \frac{\exp\left(-\frac{\|\boldsymbol{\mu}-\boldsymbol{w}_i\|^2}{2\sigma^2}\right)}{\sum_j \exp(-\frac{\|\boldsymbol{\mu}-\boldsymbol{w}_j\|^2}{2\sigma^2})} = \frac{\exp\left(\frac{1}{\sigma^2}\boldsymbol{w}_i^T\boldsymbol{\mu}\right)}{\sum_j \exp\left(\frac{1}{\sigma^2}\boldsymbol{w}_j^T\boldsymbol{\mu}\right)} \tag{2}$$

Where $\boldsymbol{\mu}$ and $\boldsymbol{w}_*$ are l_2-normalized. $p(y|\boldsymbol{x})$ can be regarded as a Boltzmann distribution. The magnitude of σ^2 controls the entropy of this distribution. The larger the σ^2, the larger the entropy. Thus σ^2 can be regarded as the data uncertainty of sample $\boldsymbol{x}$. Assuming the class label of the sample $\boldsymbol{x}$ is i, the loss function is formulated by,

$$\begin{aligned}
\mathcal{L}_d(\boldsymbol{\mu},\sigma^2) &= -\log p(y=i|\boldsymbol{x}) \\
&= -\frac{1}{\sigma^2}\boldsymbol{w}_i^\top\boldsymbol{\mu} + \log\sum\nolimits_j \exp(\frac{1}{\sigma^2}\boldsymbol{w}_j^\top\boldsymbol{\mu}) \\
&= -\frac{1}{\sigma^2}\boldsymbol{w}_i^\top\boldsymbol{\mu} + \frac{1}{\sigma^2}\log\sum\nolimits_j \exp\left(\boldsymbol{w}_j^\top\boldsymbol{\mu}\right) \\
&\quad -\frac{1}{\sigma^2}\log\sum\nolimits_j \exp\left(\boldsymbol{w}_j^\top\boldsymbol{\mu}\right) + \log\sum\nolimits_j \exp\left(\frac{1}{\sigma^2}\boldsymbol{w}_j^\top\boldsymbol{\mu}\right) \\
&= -\frac{1}{\sigma^2}\log\frac{\exp\left(\boldsymbol{w}_i^\top\boldsymbol{\mu}\right)}{\sum_j \exp\left(\boldsymbol{w}_j^\top\boldsymbol{\mu}\right)} + \log\frac{\sum_j \exp(\frac{1}{\sigma^2}\boldsymbol{w}_j^\top\boldsymbol{\mu})}{\left(\sum_j \exp\left(\boldsymbol{w}_j^\top\boldsymbol{\mu}\right)\right)^{\frac{1}{\sigma^2}}} \\
&\approx \frac{1}{\sigma^2}\mathcal{L}(\boldsymbol{\mu}) + \log\sigma^2
\end{aligned} \tag{3}$$

where $\mathcal{L}(\boldsymbol{\mu}) = -\log\frac{\exp\left(\boldsymbol{w}_i^\top\boldsymbol{\mu}\right)}{\sum_j \exp\left(\boldsymbol{w}_j^\top\boldsymbol{\mu}\right)}$ is the cross entropy loss. In the approximation, we assume that $\sigma^2 \rightarrow 1$, then $\sum_j \exp\left(\frac{1}{\sigma^2}\boldsymbol{w}_j^\top\boldsymbol{\mu}\right) \approx \sigma^2\left(\sum_j \exp\left(\boldsymbol{w}_j^\top\boldsymbol{\mu}\right)\right)^{\frac{1}{\sigma^2}}$. Note

that this assumption is reasonable as it prevents the model from predicting too small data uncertainty.

6 Derivation of the Bayesian Framework

In the Bayesian framework, as the true posterior $p(\boldsymbol{\theta}|\mathcal{D})$ is typically intractable, an approximate distribution $q_{\boldsymbol{\pi}}(\boldsymbol{\theta})$ parameterized by $\boldsymbol{\pi}$ is defined. $q_{\boldsymbol{\pi}}(\boldsymbol{\theta})$ is aimed to be as similar as possible to $p(\boldsymbol{\theta}|\mathcal{D})$, which is measured by the Kullback-Leibler divergence. The optimal parameters $\boldsymbol{\pi}^*$ is defined as,

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \mathrm{KL}[q_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \| p(\boldsymbol{\theta}|\mathcal{D})] \tag{4}$$

According the Bayesian theorem: $p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}$. Then $\mathrm{KL}[\mathrm{q}_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \| \mathrm{p}(\boldsymbol{\theta}|\mathcal{D})]$ can be rewritten as:

$$\begin{aligned}
\mathrm{KL}[\mathrm{q}_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \| \mathrm{p}(\boldsymbol{\theta}|\mathcal{D})] &= \int q_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \log \frac{q_{\boldsymbol{\pi}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})} d\boldsymbol{\theta} \\
&= \int q_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \log \frac{q_{\boldsymbol{\pi}}(\boldsymbol{\theta}) p(\mathcal{D})}{p(\boldsymbol{\theta}) p(\mathcal{D}|\boldsymbol{\theta})} d\boldsymbol{\theta} \\
&= \int q_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \log \frac{q_{\boldsymbol{\pi}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} d\boldsymbol{\theta} - \int q_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \log p(\mathcal{D}|\boldsymbol{\theta}) d\boldsymbol{\theta} \\
&\quad + \int q_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \log p(\mathcal{D}) d\boldsymbol{\theta} \\
&= \mathrm{KL}[\mathrm{q}_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \| \mathrm{p}(\boldsymbol{\theta})] - \mathbb{E}_{q_{\boldsymbol{\pi}}(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] + \log p(\mathcal{D})
\end{aligned} \tag{5}$$

Thus, Eq. 4 can be transferred as:

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \mathrm{KL}[q_{\boldsymbol{\pi}}(\boldsymbol{\theta}) \| p(\boldsymbol{\theta})] - \mathbb{E}_{q_{\boldsymbol{\pi}}(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \tag{6}$$

where $\log p(\mathcal{D})$ is dropped because it is independent of the parameters $\boldsymbol{\pi}$.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
2. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: CVPR. pp. 5693–5703 (2019)
3. Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., Tian, Q.: Scalable person re-identification: A benchmark. In: ICCV. pp. 1116–1124 (2015)