Backbone Can Not be Trained at Once: Rolling Back to Pre-trained Network for Person Re-identification

Youngmin Ro¹ Jongwon Choi² Dae Ung Jo¹ Byeongho Heo¹ Jongin Lim¹ Jin Young Choi¹
{treeoflife, mardaewoon, bhheo, ljin0429, jychoi}@snu.ac.kr, jw17.choi@samsung.com
¹Department of ECE, ASRI, Seoul National University, Korea
²Samsung SDS, Korea

Abstract
In person re-identification (ReID) task, because of its shortage of trainable dataset, it is common to utilize fine-tuning method using a classification network pre-trained on a large dataset. However, it is relatively difficult to sufficiently fine-tune the low-level layers of the network due to the gradient vanishing problem. In this work, we propose a novel fine-tuning strategy that allows low-level layers to be sufficiently trained by rolling back the weights of high-level layers to their initial pre-trained weights. Our strategy alleviates the problem of gradient vanishing in low-level layers and robustly trains the low-level layers to fit the ReID dataset, thereby increasing the performance of ReID tasks. The improved performance of the proposed strategy is validated via several experiments. Furthermore, without any additions such as pose estimation or segmentation, our strategy exhibits state-of-the-art performance using only vanilla deep convolutional neural network architecture. Code is available at https://github.com/youngminPIL/rollback

Introduction
Person re-identification (ReID) refers to the tasks connecting the same person, for instance, a pedestrian, among multiple people detected in non-overlapping camera views. Different camera views capture pedestrians in various poses with different backgrounds, which interferes with the ability to correctly estimate the similarity among pedestrian candidates. These obstacles makes it difficult to recognize the identities of numerous pedestrians robustly by comparing them with a limited number of person images with known identities. Furthermore, it is infeasible to obtain large training datasets sufficient to cover the appearance variation of pedestrians, making the ReID problem difficult to be solved. When sufficient training data is not available, it is a common approach to fine-tune the network pre-trained by another large dataset (e.g., ImageNet) which contains abundant information. The fine-tuning approach results in better performance than the approaches in which networks are trained from randomly initialized parameters. This is a practical approach used in many research areas (Ren et al. 2015; Long, Shelhamer, and Darrell 2015) to avoid the problem of overfitting. Likewise, the previous ReID algorithms (Chang, Hospedales, and Xiang 2018; Si et al. 2018; Sun et al. 2017) have utilized the fine-tuning approach. Most of recent works in ReID research have attempted to utilize semantic information such as pose estimation (Zhao et al. 2017; Xu et al. 2018; Sarfraz et al. 2018), segmentation mask (Song et al. 2018), and semantic parsing (Kalayeh et al. 2018) to improve the accuracy of ReID by considering the additional pedestrian contexts.

In contrast to the previous studies, we are interested in incrementally improving the performance of ReID by enhancing the basic fine-tuning strategy applied to the pre-trained network. A few attempts have been made to improve learning methods by the ways designing a new loss function or augmenting data in a novel way (Zhang et al. 2017; Chen et al. 2017; Zhong et al. 2017b; Sun et al. 2017). However, there has been no research on improving the learning method to consider the characteristics of each layer filter.

Before suggesting our novel fine-tuning strategy for ReID, we first empirically analyze the importance of fine-tuning low-level layers for ReID problems. According to related research (Zeiler and Fergus 2014; Mahendran and Vedaldi 2015), the low-level layers concentrate on details of
appearance to discriminate between samples while the high-
level layers contain semantic information. Thus, we need
to sufficiently fine-tune the low-level layers to improve
the discriminant power for the specific class ‘person’ in ReID
because the low-level layers of the pre-trained network
include detailed information on numerous classes. However,
since the gradients delivered from high-level layers to low-
level layers are reduced through back-propagation, the low-
level layers suffer from a gradient-vanishing problem, which
causes early convergence of the entire network before the
low-level layers are trained sufficiently.

To solve this problem, we propose a novel fine-tuning
strategy in which a part of the network is intentionally
perturbed when learning slows down. The proposed fine-
tuning strategy can recover the vanished gradients by rolling
back the weights in the high-level layers to their pre-trained
weights, which provides an opportunity for further tuning
of weights in the low-level layers. As shown in Figure 1,
the proposed fine-tuning strategy allows the network to con-
verge to a minimum in a basin with better generalization
performance than the conventional fine-tuning method. We
validate the proposed method that uses no add-on schemes
via a number of experiments, and the method outperforms
state-of-the-art ReID methods appending additional context
to the basic network architecture. Furthermore, we apply
the proposed learning strategy to the fine-grained classific-
ation problem, which validates its generality for various computer
vision tasks.

**Related Work**

Traditionally, the ReID problem has been solved by using a
metric learning method (Koestinger et al. 2012) to narrow
the distance among the images of the same person. Clo-
thing provides an important hint in the ReID task, and some
approaches (Pedagadi et al. 2013) have used color-based histograms. With the develop-
ment of deep learning, many ReID methods to learn dis-

criminative features by deep architectures appear, which dra-

matically increases the ReID performance (Sun et al. 2017;
Li, Zhu, and Gong 2018; Hermans, Beyer, and Leibe 2017).

Recently, the state-of-the-art approaches (Si et al. 2018;
Song et al. 2018; Zhong et al. 2018) have also used the
advanced deep architecture, especially pre-trained on Im-
ageNet (Deng et al. 2009), as a backbone network.

**Add-on semantic information method in ReID** To in-
crease the performance, many recent works based on the
deep architectures have tried to consider additional semantic
information such as poses of pedestrians and attention
marks. One of the most popular approaches is to use the
off-the-shelf pose estimation algorithms (Cao et al. 2017;
Insafutdinov et al. 2016) to tackle the misaligned poses of
the candidate pedestrians. In (Su et al. 2017), using the pose
information, Su et al aligned each part of a person, produc-
ing pose-normalized input to deal with the problem of the
deformable variation of the ReID object. Sarfraz et al. (Sar-
fraz et al. 2018) proposed a view predictor network that dis-

Distinguish the front, back, and sides of a person using pose
normalization. The synthesized image can be used as training data because the label is pre-
served. (Xu et al. 2018) proposed attention-aware com-
pilation network. They pointed out the conventional methods
using pose information based on rigid body regions such as
rectangular RoI. They obtained non-rigid parts through con-
nectivity information between the human joints and matched
them individually. In contrast to the previous ReID methods,
we target on improving the training method itself without
any additional semantic information or extra architecture.

**Advanced fine-tuning methods** There are other studies
to improve learning methods on pre-trained networks. Li
and Hoiem (Li and Hoiem 2017) suggested a method which can learn a new task without forgetting the existing tasks in
transfer learning. In (Kornblith, Shlens, and Le 2018), Ko-
rbliht et al. analyzed a conventional fine-tuning method,
which concluded that the state-of-the-art ImageNet archi-

tecture yields state-of-the-art results over many tasks. In the
ReID task, several methods have improved learning strategy
on pre-trained networks. The quadruplet loss was proposed
in (Chen et al. 2017). In this research, Chen et al. have de-
veloped an improved version of triplet losses, which does
not only make the inter-class close but also add a negative
sample, making the distance in the intra-class much longer.
In (Zhang et al. 2017), Zhang et al were inspired by the

distillation method (Hinton, Vinyals, and Dean 2015) be-

tween teacher and student networks and proposed a learn-
ing method based on co-student networks which can be
trained without teacher network. However, there has been no
research considering the fine-tuning characteristics for the
ReID problem. In this paper, we propose a novel fine-tuning
strategy adapted to the ReID task, which takes into account
the layer-by-layer characteristic of the network.

**Methodology**

In this section, we first analyze the conventional fine-tuning
strategy to determine which layer is insufficiently trained for
ReID problems. Based on the analysis, we propose a new
fine-tuning strategy that alleviates the vanishing gradient in
the poorly trained layers, consequently improving the gen-
eralization performance of the fine-tuned network.
Overall framework
Before describing the empirical analysis and the proposed fine-tuning strategy, we first introduce an overall framework including a network architecture with its training and testing processes. The notations defined in this section are used in the following sections.

Architecture In this paper, we use a classification-based network (Zheng, Yang, and Hauptmann 2016) that determines the entire identity label as a class. We assume that the deep convolutional neural network consists of two components: a feature extractor and a classifier. The feature extractor is composed of multiple convolutional layers and the classifier consists of several fully-connected (FC) layers. As the feature extractor, we utilize convolutional layers of pre-trained ResNet (He et al. 2016), which are widely used in many ReID algorithms (Sun et al. 2017; Zhong et al. 2018; Qian et al. 2018). The three structures ResNet-34, ResNet-50, and ResNet-101 are used for the feature extractor to show the generality of the proposed fine-tuning strategy. According to the resolution of the convolutional layers, the feature extractor is composed of multiple convolutional layers and the first convolutional layer. The resulting feature vector is a 2048-D vector for ResNet-34, ResNet-50, and ResNet-101 are used for the feature extractor to show the generality of the proposed fine-tuning strategy. Following feature extraction, a feature vector is obtained by feeding the feature vector obtained from the feature extractor into the classifier. The classifier is newly defined by minimizing Eq. 2.

In the training process, a stochastic gradient descent method is used to train θ by minimizing Eq. 2.

Testing process The identities given to the testing set are completely different than the identities in the training set. Thus, the classifier trained in the training process cannot be used for the testing process. To find correspondence between pedestrian candidates without using the classifier, we estimate the similarity of two pedestrians based on the distance between the feature vectors of each pedestrian extracted from the trained feature extractor. To evaluate the performance, the testing set is divided into a query set and a gallery set with Mq and Mg samples, respectively. The samples of the query and gallery sets are denoted by \{xq,i\}_i=1^{Mq} and \{xg,j\}_j=1^{Mg}, respectively. Each sample in the query set is a person of interest, which should be matched to the candidate samples in the gallery set.

The distance between xq,i and xg,j is calculated by L-2 norm as follows:

\[ q(i) = \mathcal{F}(xq,i|\theta_1,...,\theta_N) \]

\[ g(j) = \mathcal{F}(xg,j|\theta_1,...,\theta_N) \]

\[ s_{i,j} = ||q(i) - g(j)||^2_2. \]

The identity of the gallery sample with the lowest distance \( s_{i,j} \) is determined as the identity of the \( i \)-th query sample.
Analysis of fine-tuning method

This section determines which layer converges insufficiently by conventional fine-tuning. Figure 3 shows the convergence, supporting the key ideas of the proposed fine-tuning strategy. ‘baseline’ denotes the conventional fine-tuning, while ‘Block i’ indicates the refine-tuning wherein every block except ‘Block i’ is rolled back after the ‘baseline’ fine-tuning. Table 1 shows the generalization performance of each scheme. A meaningful discovery is that a rolling-back scheme with remaining low-level blocks (Block1, Block2, Block3) shows slower convergence than applying the rolling-back scheme to the remaining high-level blocks (Block3, Block4). However, as shown in Table 1 the scheme that maintains the low-level blocks gives better generalization performance than the scheme preserving the high-level blocks. This indicates that the ‘baseline’ fine-tuning causes the low-level layers to be converged at a premature. This gives us an insight that rolling back of the high-level layers except the low-level layers might give the low-level layers an opportunity to learn further. As additional consideration, all the weights cannot be given in pre-trained states. This is because the output layer of a deep network for a new task is usually different from the backbone network. Hence, the FC layers must be initialized in a random manner. Rolling back the FC layers to random states does not provide any benefit. Thus, in our rolling-back scheme, FC layers are excluded from rolling back, although it is a high-level layer, to keep a consistent learning of the low-level layers.

Refine-tuning with rolling back

The aforementioned analysis shows that a premature convergence degrades performance and rolling back high-level layers can be a beneficial strategy to mitigate the premature convergence problem in the low-level layers. For further tuning of the low-level layers, we designed a rolling-back refine-tuning scheme that trains the low-level layers incrementally from the front layer along with rolling back the remaining high-level layers. The detailed rolling back scheme is described in the following.

1. In the first refine-tuning period \((p = 1)\), the weights, \(\theta_1, \ldots, \theta_N\), are initialized with the pre-trained weights, \(\hat{\theta}_i^{(0)}\).

\[
\theta_1^{(1)} = \hat{\theta}_1^{(0)}, \forall i = 1, \ldots, N
\]

\(\Rightarrow\) Initialize weights to pre-trained one

2. From the refine-tuning period with rolling back \((p \geq 2)\), we roll-back the high-level layers as in the following procedure. First, Block1 \((\theta_1)\) is maintained in the state of previous period and all the remaining blocks \((\theta_2, \ldots, \theta_N)\) are rolled back to their pre-trained states \(\hat{\theta}_i^{(0)}\). In other words, Block1 continues the learning, and the other blocks restart the learning from the beginning with the pre-trained initial weights. In the incremental manner, the next low-level block is added one-by-one to the set of blocks continuing the learning, while the remaining ones are rolled back. The rolling-back refine-tuning is repeated until all layers are included in the set of blocks continuing the learning. In summary, in the \(p\)-th refine-tuning period, the weights of the network are rolled back as

\[
\theta_i^{(p)} = \begin{cases} 
\hat{\theta}_i^{(p-1)} & i < p \\
\hat{\theta}_i^{(0)} & i \geq p
\end{cases}
\]

where \(\hat{\theta}_i^{(p-1)}, i = 1, \ldots, N\) are the updated weights in the \((p - 1)\)-th refine-tuning period. During the refine-tuning process, the \(\theta_{FC}\) is not rolled back as mentioned above.

\[
\theta_i^{(p)} = \hat{\theta}_{FC}^{(p-1)}
\]

The detailed procedure of the refine-tuning scheme with rolling-back is summarized in Algorithm 1.

**Algorithm 1** Re-fine learning

| Parameter: | N: Number of total block, M: Number of lower block |
| Parameter: | \(\theta_1^{(0)}, \ldots, \theta_N^{(0)}\): weights of pre-trained network |
| Input: | \(\theta_1, \ldots, \theta_N, \theta_{FC}, X, Y\) (dataset) |

1. \(\theta_1^{(1)} = \hat{\theta}_1^{(0)}, \forall i = 1, \ldots, N\) \(\Rightarrow\) First fine-tune on ReID dataset \(X,Y\)

2. \(\theta_1^{(1)} \leftarrow \text{random initialization}\) \(\Rightarrow\) Remain certain layers and roll back others

3. \(\hat{\theta}_1^{(1)}, \ldots, \hat{\theta}_N^{(1)}, \hat{\theta}_{FC}^{(1)} \leftarrow \text{FINE-TUNE}(X,Y,\theta_1^{(1)}, \ldots, \theta_N^{(1)}, \theta_{FC}^{(1)})\) \(\Rightarrow\) Do not roll back FC layers

4. \(\theta_i^{(p)} = \begin{cases} 
\hat{\theta}_i^{(p-1)} & i < p \\
\hat{\theta}_i^{(0)} & i \geq p
\end{cases}\) \(\Rightarrow\) Refine-tune on ReID dataset \(X,Y\)

**Dataset**

**Market-1501** Market-1501 (Zheng et al. 2015) is widely used dataset in person ReID. Market-1501 contains 32,668
Table 2: Results of our rolling-back scheme on different ReID dataset

| Period | continuously tuned blocks | Market-1501 mAP | DukeMTMC mAP | CUHK03-L mAP | CUHK03-D mAP |
|--------|--------------------------|----------------|--------------|--------------|--------------|
|        |                          | rank-1         | rank-1       | rank-1       | rank-1       |
| 1      | none                     | 73.16          | 89.43        | 63.26        | 80.83        |
| 2      | B1+ FC                   | 75.65          | 90.95        | 66.09        | 81.96        |
| 3      | B1+B2+FC                 | 76.54          | 91.12        | **66.57**    | **82.41**    |
| 4      | B1+B2+B3+FC              | **77.01**      | **91.24**    | **66.39**    | **82.32**    |

Table 3: Results of our rolling-back scheme for different network types.

| Period | continuously tuned blocks | ResNet-34 | ResNet-101 |
|--------|--------------------------|-----------|------------|
|        |                          | Market-1501 mAP | DukeMTMC mAP | Market-1501 mAP | DukeMTMC mAP |
|        |                          | rank-1     | rank-1      | rank-1     | rank-1      |
| 1      | none                     | 70.65      | 86.93       | 60.06      | 78.69       |
| 2      | B1+ FC                   | 73.63      | 89.13       | 63.45      | 81.10       |
| 3      | B1+B2+FC                 | 74.85      | 90.02       | 65.16      | 82.18       |
| 4      | B1+B2+B3+FC              | **74.97**  | **90.05**   | **65.44**  | **83.08**   |

DukeMTMC-ReID (Zheng, Zheng, and Yang 2017) has been specially designed for person ReID. DukeMTMC contains 36,411 images of 1,402 identities which are divided into a training set and a testing set of 702 and 702 identities, respectively.

CUHK03-np (Zhong et al. 2017a) is a modified version of the original CUHK03 dataset. The hand-labeled (CUHK03-L) and DPM-detected (CUHK03-D) bounding boxes are offered. CUHK03-np contains 14,096 images of 1,467 identities. The new version is split into two balanced sets containing 767 and 700 identities for training and testing, respectively.

Implementation detail

Our method was implemented using PyTorch (Paszke et al. 2017) library. All inputs are resized to 288×144 and the batch size was set to 32. No other augmentation is used except horizontal flip in our training process. The initial learning rate was set to 0.01 and 0.1 for the feature extractor and the classifier, respectively. The learning rates were multiplied by 0.1 at every 20 epoch and we trained for 40 epochs as one refine-tuning period. In our experiment, the proposed refine-tuning strategy has been rolled back three times and four epochs have been trained for four refine-tuning periods, and so a total of 160 epochs have been repeated for all fine-tuning. The learning rates of rolling back blocks are restored to 0.01 at the beginning of every period. In contrast, the blocks that do not roll back begin with the low learning rate of 0.001 since a high learning rate of the sufficiently trained blocks might yield sudden exploding. The optimizer used in this study was stochastic gradient descent (SGD) with nesterov momentum (Nesterov 1983). For the optimizer, the momentum rate and the weight decay were set to 0.9 and 5×10−4, respectively. In every rolling back, the momentum of gradient was reset to 0. In the test process, the additional feature vector was used to add the feature vector of the horizontal flipped input pairwise. We report rank-1 accuracy of Cumulative Matching Characteristics (CMC) curve and the mean Average Precision (mAP) for performance evaluation.

Ablation tests

The network trained with the proposed strategy was verified via ablation tests on Market-1501, DukeMTMC, CUHK03-L and CUHK03-D. The proposed refine-tuning strategy is applied to a network over four periods. As the refine-tuning periods progress, the continuously tuned blocks are cumulative (e.g., B1+B2+FC in the third period). The other blocks are rolled back to their original pre-trained states. As shown in Table 2, the performance increases as the refine-tuning periods progress with the exception of DukeMTMC in the fourth period. However, even in this case, the gap was negligible. The improvement is most prominent in the second refine-tuning period during which the first rolling back is performed. To verify the generality of our refine-tuning scheme, we conducted additional experiments with other networks including ResNet-34 and ResNet-101 (He et al. 2016) under the same settings. Table 3 shows the performance of each network in Market-1501 and DukeMTMC. The proposed refine-tuning scheme also showed a consistent improvement in ResNet-34 and ResNet-101. The ablation test results demonstrate that the proposed refine-tuning scheme has a significant advantage as a general method to enhance the generalization performance in the ReID problem in which only a limited amount of data is available.
Effect of rolling back as a perturbation

To evaluate the effect of our rolling-back scheme, it is compared with 'base.cy' method that does roll back none of the block but merely adjusts the learning rate with the same timing as ours for a perturbation driving to other local basins. The 'base.cy' is similar to other studies (Loshchilov and Hutter 2016; Smith 2017) that perturb only the learning rate. Figure 4 shows the change in training loss and mAP of the whole processes of the proposed refine-tuning and the base.cy fine-tuning. After the first rolling-back at 40 epochs, the training loss from the rolling-back scheme converges to a value that is better than the value of the base.cy in the 70-80 epochs. After the second and third rolling-backs, the training loss of the base.cy converges to a lower value than that of the proposed method, but the base.cy shows a worse generalization performance (mAP) than the proposed method.

Comparison to FC warm-up training

In this section, we discuss the difference between our method and FC warm-up training (He et al. 2016). As mentioned previously, the new FC layers start randomly from scratch. FC warm-up is a way to freeze the pre-trained weights in all hidden layers except for the FC layers and train the FC layers before starting the main fine-tuning. In the comparison experiment, the baseline was warmed up for 20 epochs. In our proposed method, period 1 (see Table 2) is similar to FC warm-up where FC layers start from random scratch. However, the proposed method does not freeze the pre-trained weights in period 1. The training loss and mAP for FC warm-up and our methods are depicted in Figure 5. FC warm-up and our methods start fine/refine-tuning after training the FC layers. The FC warm-up converges to a lower training loss than the proposed method, but the proposed method shows better performance in terms of generalization.
### Table 4: Comparison with State-of-the-art methods on Market-1501, DukeMTMC and CUHK03-L/D

| Method     | Backbone | Market-1501 mAP | DukeMTMC mAP | CUHK03-L mAP | CUHK03-D mAP | Add-on   |
|------------|----------|----------------|--------------|--------------|--------------|----------|
| PT-GAN (2018) | ResNet-50 | 58.0               | 48.1         | 30.5         | 28.2         | pose+GAN |
| SVDNet (2017) | ResNet-50 | 62.1               | 56.8         | 37.8         | 37.3         | -        |
| PDC (2017)   | Inception | 63.4               | -            | -            | -            | -        |
| AACN (2018)  | GoogleNet | 66.9               | 59.3         | -            | -            | -        |
| HAP2S_P (2018) | ResNet-50 | 69.4               | 60.6         | 75.9         | -            | -        |
| PSE (2018)   | ResNet   | 69.0               | 62.0         | 79.8         | -            | -        |
| CamStyle (2018) | ResNet-50 | 71.6               | 57.6         | 41.0         | 43.8         | -        |
| PN-GAN (2018) | ResNet-50 | 72.6               | 53.2         | 73.6         | -            | -        |
| MGCAM (2018) | MSCAN    | 74.3               | -            | 50.2         | 46.7         | mask     |
| MLFN (2018)  | Original | 74.3               | 62.8         | 49.2         | 47.8         | 52.8     |
| HA-CNN (2018) | Inception | 75.7               | 63.8         | 41.0         | 38.6         | -        |
| DuATM (2018) | DenseNet-121 | 76.6             | 64.6         | 81.8         | -            | -        |
| Ours ResNet-34 |           | 75.0               | 65.4         | 48.6         | 53.0         | 45.6     |
| Ours ResNet-50 |           | 77.0               | 66.6         | 50.7         | 55.6         | 47.4     |
| Ours ResNet-101 |          | 79.9              | 70.2         | 55.7         | 59.8         | 50.5     |

### Table 5: Comparison with State-of-the-art methods using same backbone network ResNet-50

| Method     | Market-1501 mAP | DukeMTMC mAP | Add-on   |
|------------|----------------|--------------|----------|
| PT-GAN     | 58.0           | 48.1         | GAN      |
| SVDNet     | 62.1           | 56.8         | -        |
| HAP2S_P    | 69.4           | 60.6         | -        |
| CamStyle   | 71.6           | 57.6         | GAN      |
| PN-GAN     | 72.6           | 53.2         | GAN      |
| Ours       | 77.0           | 66.6         | 82.4     |

### Attention performance of our refine-tuning method

To learn discriminative features for the ReID task, it is important to distinguish the foreground from the background. Figure 6 shows that our method can generate a more distinguishable feature map in the last convolutional layer than the baseline of the conventional fine-tuning method.

### Comparisons with state-of-the-art methods

We also compared the proposed method with state-of-the-art methods. Table 4 shows the comparison results when using ResNet-50. The proposed rolling-back refine-tuning scheme shows the best performance even though our method does not use any add-on scheme. Furthermore, compared to other methods without add-on scheme (SVDNet, HAP2S_P), our method outperforms them by more than 7% mAP improvement for Market-1501. Table 4 summarizes the results compared with the state-of-the-art methods on Market-1501, DukeMTMC, and CUHK03-L/D. According to the results, the rolling-back refine-tuning scheme makes a meaningful contribution to the enhancement of any backbone networks so that it outperforms state-of-the-art algorithms utilizing add-on schemes.

### Conclusion

In this paper, we proposed a refine tuning method with a rolling-back scheme which further enhances the backbone network. The key idea of the rolling-back scheme is to restore the weights in a part of the backbone network to the pre-trained weights when the fine-tuning converges at a premature state. To escape from the premature state, we adopt an incremental refine tuning strategy by applying the fine tuning repeatedly, along with the rolling-back. According to the experimental results, the rolling-back scheme makes a meaningful contribution to enhancement of the backbone network where it derives the convergence to a local basin of a good generalization performance. As a result, our method without any add-on scheme could outperform the state-of-the-arts with help of add-on scheme.

### Acknowledgement

This work was supported by Next-Generation ICD Program through NRF funded by Ministry of S&ICT [2017M3C4A7077582], ICT R&D program MSIP/IITP [2017-0-00306, Outdoor Surveillance Robots].

### References

[Bossard, Guillaumin, and Van Gool ] Bossard, L.; Guillaumin, M.; and Van Gool, L. Food-101–mining discriminative components with random forests. In *ECCV*, pages=446–461, year=2014, organization=Springer.

[Cao et al. 2017] Cao, Z.; Simon, T.; Wei, S.-E.; and Sheikh, Y. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*.

[Chang, Hospedales, and Xiang 2018] Chang, X.; Hospedales, T. M.; and Xiang, T. 2018. Multi-level factorisation net for person re-identification. In *CVPR*.

[Chen et al. 2017] Chen, W.; Chen, X.; Zhang, J.; and
[Zhang et al. 2017] Zhang, Y.; Xiang, T.; Hospedales, T. M.; and Lu, H. 2017. Deep mutual learning. arXiv preprint arXiv:1706.00384.

[Zhao et al. 2017] Zhao, L.; Li, X.; Zhuang, Y.; and Wang, J. 2017. Deeply-learned part-aligned representations for person re-identification. In ICCV.

[Zheng et al. 2015] Zheng, L.; Shen, L.; Tian, L.; Wang, S.; Wang, J.; and Tian, Q. 2015. Scalable person re-identification: A benchmark. In Computer Vision, IEEE International Conference on.

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

[Zheng, Zheng, and Yang 2017] Zheng, Z.; Zheng, L.; and Yang, Y. 2017. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In ICCV.

[Zhong et al. 2017a] Zhong, Z.; Zheng, L.; Cao, D.; and Li, S. 2017a. Re-ranking person re-identification with k-reciprocal encoding. In CVPR.

[Zhong et al. 2017b] Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; and Yang, Y. 2017b. Random erasing data augmentation. arXiv preprint arXiv:1708.04896.

[Zhong et al. 2018] Zhong, Z.; Zheng, L.; Zheng, Z.; Li, S.; and Yang, Y. 2018. Camera style adaptation for person re-identification. In CVPR.