Soft Person Re-identification Network Pruning via Block-wise Adjacent Filter Decaying

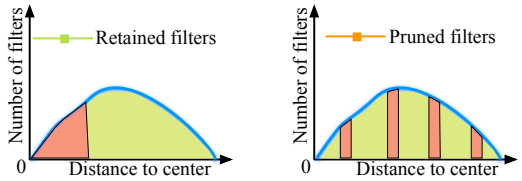
Xiaodong Wang, Zhedong Zheng, Yang He, Fei Yan, Zhiqiang Zeng, and Yi Yang, *Senior Member, IEEE*

***Abstract*—Deep learning has shown significant successes in person re-identification (re-id) tasks. However, most existing works focus on discriminative feature learning and impose complex neural networks, suffering from low inference efficiency. In fact, feature extraction time is also crucial for real-world applications and light-weight models are needed. Prevailing pruning methods usually pay attention to compact classification models. However, these methods are sub-optimal for compacting re-id models, which usually produce continuous features and are sensitive to network pruning. The key point of pruning re-id models is how to retain the original filter distribution in continuous features as much as possible. In this work, we propose a block-wise adjacent filter decaying method to fill this gap. Specifically, given a trained model, we first evaluate the redundancy of filters based on the adjacency relationships to preserve the original filter distribution. Second, previous layer-wise pruning methods ignore that discriminative information is enhanced block-by-block. Therefore, we propose a block-wise filter pruning strategy to better utilize the block relations in the pre-trained model. Third, we propose a novel filter decaying policy to progressively reduce the scale of redundant filters. Different from conventional soft filter pruning that directly sets the filter values as zeros, the proposed filter decaying can keep the pre-trained knowledge as much as possible. We evaluate our method on three popular person re-identification datasets, *i.e.*, Market-1501, DukeMTMC-reID, and MSMT17_V1. The proposed method shows superior performance to existing state-of-the-art pruning methods. After pruning over 91.9% parameters on DukeMTMC-reID, the $Rank$-1 accuracy only drop 3.7%, demonstrating its effectiveness for compacting person re-identification.**

***Index Terms*—Deep learning, representation learning, network pruning, person re-identification.**

I. Introduction

Deeply-learned features generated by the Convolutional Neural Network (CNN) have been successfully applied to person re-identification in recent years [1], [2]. However, training and testing the CNN models need expensive computational resources [3]–[6], making them hard to deploy on devices with limited resources, such as embedded devices and autonomous vehicles. For example, given a $256 \times 128 \times 3$ input image, it takes nearly 0.2 seconds to extract the feature embedding using a ResNet-50 network [7] on an 1.3GHz Dual-Core I5 CPU, which is not ideal for most real-time applications.



(a) Center-based methods (b) Our method

Fig. 1: Filter distribution (distance to the geometric center) for different pruning methods. The red shadow and the green shadow denote the pruned filters and retained filters, respectively. **(a)** For center-based methods, only the filters far away from the geometric center (green) are kept. The original filter distribution will be damaged when a large proportion of filters (red) is entirely removed. **(b)** In contrast, our method prunes filters according to their adjacent relationships, which can adequately maintain the original filter distribution.

To accelerate the CNN model inference, researchers resort to compressing model without significantly hurting the accuracy [8]. Network pruning is one of the most popular compression techniques due to its compatibility with the traditional training procedure and efficiency in shortening the inference time [9], [10]. Most existing pruning methods focus on the classification problem, which is entirely different from the person re-identification problem. Generally, the classification model usually focuses on categorical logic output. However, in many person re-identification applications, one usually deploys the deep network as the feature extractor and combines it with some distance metrics for final evaluation. In this case, changes in these continuous features could lead to a large bias on final person matching or ranking results [11]. Therefore, compared with the classification model, the person re-identification model could be more sensitive to visual representation. Such a difference poses three challenges for existing pruning methods for person re-identification tasks.

First, **how to effectively retain the original filter distribution as much as possible?** Basically, the previous network pruning methods [12]–[14] typically consider the center-based criteria. These criteria are rooted in that redundant filters are close to the reference center, such as small-norm values (*e.g.*,

Work done during the visiting at University of Technology Sydney. This paper was supported by China Scholarship Council (No.201908350025), National Natural Science Foundation of China (Grant Nos.61871464, U1805264), National Natural Science Foundation of Fujian Province (Grant Nos.2020J01266, 2021J011186), the "Climbing" Program of XMUT (Grant No.XPDKT20031), Program of XMUT for high-Level talents introduction plan (Grant No.YKJ19003R).

Xiaodong Wang is with the college of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China (e-mail:xdwangjsj@xmut.edu.cn)

Zhedong Zheng is with NExT++, School of Computing, National University of Singapore, Singapore. (e-mail: zdzheng@nus.edu.sg)

Yang He, and Yi Yang are with the Australian Artificial Intelligence Institute, University of Technology Sydney, NSW 2007, Australia. Yang He is also with Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore 138632.

Fei Yan and Zhiqiang Zeng are with the college of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China.

l_1-norm [12] and l_2-norm [13]) or geometry median [14]. One disadvantage of these criteria, as shown in Figure 1, is that the filters close to the reference center can be removed entirely. Hence, these center-based criteria might cause an enormous change in original filter distribution and hurt the performance of person re-identification.

Second, **how to effectively utilize the block relations in the pre-trained model?** Recent CNN models have achieved great successes in imposing a block-wise network framework [7], [15], [16], which wraps a sequence of coherent layers (*e.g.*, Conv-BN-ReLU-Conv-BN-ReLU) as a block. Some studies also have made deep investigations into the function of intermediate feature layers and block relations, demonstrating that the discriminative information in CNN is enhanced block by block [17], [18]. This observation indicates that different blocks exhibit diverse levels of redundancy [19]. However, recent layer-wise works usually focus on retaining the layer-topology in the same block, neglecting the redundancy diversity of blocks [12]–[14].

Third, **how to leverage the pre-trained knowledge in original models while pruning?** Pre-trained models are widely adopted in previous person re-identification studies [20], [21]. However, most of the existing network pruning methods remove redundant filters in a hard-pruning manner, where the pruned filters are directly deleted and have no chance to be recovered. As shown in Figure 2, cursorily discarding the redundant filters can prevent pre-trained knowledge utilization and reduce the model capacity. To deal with this problem, some recent soft-pruning methods propose to relieve performance degradation by postponing the deletion of pruned filters until the end of the fine-tuning phase [13], [14]. However, when the pruning rate increases, the parameter gaps between the pruned network and the pre-trained network will become larger, resulting in degraded pre-trained knowledge transfer.

To solve the above-mentioned problems, we propose a novel Block-wise Adjacent Filter Decaying (BAFD) method to accelerate person re-identification. To be specific, before pruning, a block-wise adjacent filter ranking strategy is imposed to determine the redundancy of each block. This ranking strategy explores the block redundancy and simultaneously retains the topology of the pre-trained blocks. Concretely, the layers in the same block share a similar pruning strategy, while those in different blocks are pruned with diverse pruning rates. Then, we progressively reduce the scale of redundant filters (*i.e.*, filter decaying) and fine-tune the model subsequently. This "decaying and fine-tuning" process continues until the pruned model is converged. Finally, some filters with small scales (*i.e.*, zero) will be selected and safely deleted. This progressive block-wise adjacent filter decaying enables the compact network to recover the missing knowledge and retain its original model capacity. Therefore, it is easy to improve the model compactness with limited performance dropped.

The main contributions of this paper are listed as follows.

- We propose a block-wise adjacent filter pruning manner to accelerate person re-identification. It explores the block relations in the pre-trained model and retains the original filter distribution as well.
- We propose to progressively reduce the scales of redundant filters, giving the pruned model more space to optimize itself and retaining pre-trained knowledge.
- Extensive experiments on three widely-used person re-identification datasets have demonstrated the effectiveness of the proposed method. After reducing more than 76.8% parameters, the compact model increases by nearly 0.35% on mAP over the original model on Market-1501. Furthermore, we reduce 91.9% parameters only with an increase of 3.7% $rank$-1 error on DukeMTMC-reID.

II. Related works

The proposed method is close to the network pruning and person re-identification. In this section, we briefly introduce several related studies on these two research domains.

A. Person Re-identification

Person re-identification, which originates from pedestrian tracking [22], [23], has gained increasing research attention over the past decade [24]. Recently, CNN-based image representation has shown extraordinary performance on person re-identification [24]–[33]. To explore the discriminative features at different locations and scales, Qian *et al.* [27] propose a multi-scale deep learning framework. Considering the importance of diversity in feature maps, Zheng *et al.* [34] introduce a pedestrian alignment network to extract discriminative representation without additional annotations. Shen *et al.* [30] impose a similarity constraint to both low-level and high-level convolutional layers, which is superior to learn local and global representation simultaneously. Jiang *et al.* [35] design a self-attention learning method to capture multi-level information among different levels of convolutional layers. Some studies pay attention to impose different loss functions to extract person representation, such as metric learning [25], [36], classification losses [24], [37], or the combination of different losses [21], [38]. Inspired by the recent development of the Generative Adversarial Network (GAN) [39], some studies propose to improve the re-identification embeddings by leveraging the augmented data [21], [40]–[42]. To increase the interpretability of person re-identification networks, Yuan *et al.* [43] propose a Gabor convolution module, which can boost the representation learning ability of existing networks. The abovementioned studies have achieved lots of successes in person re-identification. Unfortunately, they usually consume expensive computation resources to extract CNN descriptors, hampering their practicability for real-world applications.

B. Network Pruning

Most previous pruning methods can be roughly classified into two groups: weight pruning and filter pruning. Weight pruning mainly focuses on pruning the fine-grained weight of filters and results in unstructured models, making it hard for hardware deployment [44]–[46]. In contrast, filter pruning reduces the model size by removing the whole redundant filters and can achieve a structured model, which is more hardware friendly [47]–[49].

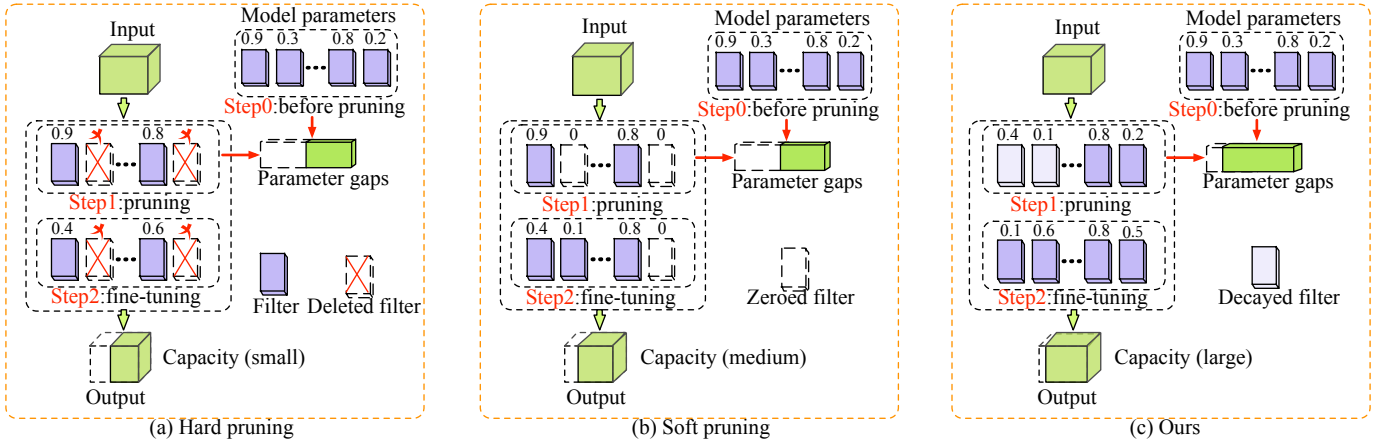


Fig. 2: Filter pruning interpretation of different methods. **(a)** Given a pre-trained model, the hard-pruning method directly removes the redundant filters to compress model, which disuses the pruned filters during the fine-tuning procedure, leading to a model capacity reduction. **(b)** Soft-pruning method zeros the redundant filters, which are allowed to join the subsequent model optimization, to improve the model capacity. However, the large parameter gap between the pruned model and the pre-trained model increases when a large number of filters is pruned, resulting in performance degradation. **(c)** Our method deploys the filter decaying policy to adjust the scale of pruned filters (light purple squares). It iteratively re-samples candidate filters and fine-tunes the model to give the wrongly-selected filters chances to be rectified, until the weight of redundant filters gradually converges to zeros (that can be safely removed). Therefore, our method could give pruned models more chances to recover the missing knowledge.

Weight pruning: One of the earliest works on weight pruning imposes the second-order Taylor expansion to evaluate the importance of the weight [44]. Then, it resets the redundant weight to zero to reduce the pruning impacts on the network. Similar to [44], a "surgical recovery" technology is proposed to remove redundant weights while allowing the other weight values for further restoration. However, the above methods require calculating the second-order Hessian matrix, which needs a heavy computing burden [45]. Later, several advancements on weight magnitude, such as direct threshold [46], l_1-norm [12], have been proposed to overcome this problem. To reduce the negative impacts on model accuracy, a dynamic network technology solution is proposed in [50]. It allows the zeroed weights to participate in the network training, giving them a certain opportunity to restore the weight value. Some recent works also remove weights and reduce their quantitative compression (the weight bits) simultaneously [51]. Nevertheless, weight pruning uses an unstructured way to delete the weight and increase the sparsity of network parameters. Therefore, to accelerate the network, it usually depends on special software libraries or hardware, limiting its application.

Filter pruning: Similar to the weight pruning, the magnitude criteria have also been successfully applied to filter pruning, such as l_1-norm [12] and l_2-norm [13]. To improve the robustness of network pruning, some studies prune filters according to the network training status [52], adjacent-layer relations [53], batch normalization [54], ReLU output [55], and the contributions to network loss [47], [56]. Nevertheless, all of the above methods are "hard" pruning methods. In these methods, the redundant filters are directly deleted from the original network, followed by fine-tuning to restore model accuracy. However, due to the early cut of network connections, the pruned network is difficult to utilize the pre-trained knowledge, resulting in the network capacity reduction. To tackle this issue, some studies propose to keep the redundant filters while fine-tuning, giving the pruned filters more chances to recover themselves. Soft Filter Pruning (SFP) is one of the early attempts to embed pruning into the training procedure [13]. However, SFP depends on the norm-based criterion, which fails when the "smaller-norm-less-important" requirement is not satisfied. To solve this problem, a Filter Pruning method via Geometric Median (FPGM) is proposed to determine the network redundancy according to the geometric relations of filters. However, these soft-pruning methods need to search for a center reference to evaluate the importance of filters. Such criteria may lead to filters in dense areas are deleted and damage the original filter distribution.

III. The Proposed Method

A. Preliminary

Let a deep CNN network be parameterized by $W = \{W^l = [w_1^l, w_2^l, \cdots, w_{C_{out}^l}^l] \in R^{C_{out}^l \times C_{in}^l \times K^l \times K^l}, 1 \leq l \leq L\}$, where W^l denotes a matrix of connection weights in the l-th layer, L denotes the number of layers, C_{out}^l and C_{in}^l represent the numbers of output channels and input channels for the l-th convolutional layer, respectively, K^l is the kernel size. We denote $\mathcal{F} = \{\mathcal{F}^1, \mathcal{F}^2, \cdots, \mathcal{F}^L\}$ as a set of filters for the whole network, where $\mathcal{F}^l$ is the set of filters for the l-th convolutional layer, *i.e.*, $\mathcal{F}^l = \{w_1^l, w_2^l, \cdots, w_{C_{out}^l}^l\}$.

B. Formulation

From the analysis of the related work, evaluating the importance of filters in a soft-pruning manner is helpful to improve the pruning performance. An acceptable reason may lie in that the soft-pruning method can give the pruned filters more chances to adjust themselves to the new structure in the pruned network. However, the way the soft-pruning

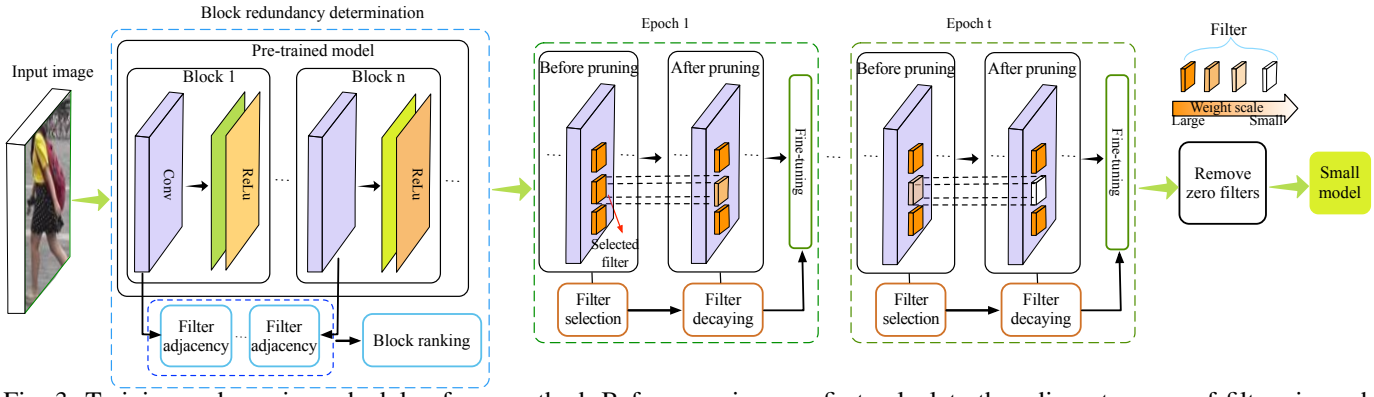


Fig. 3: Training and pruning schedule of our method. Before pruning, we first calculate the adjacent power of filters in each block, and sort these filters to determine the pruning rate of each block. Then, we individually select redundant filters in each convolutional layer according to the block-wise pruning rate. After that, we prune the redundant filters by progressively reducing their scales, followed by one epoch fine-tuning. We iteratively perform adjacent filter selection, filter decaying, and fine-tuning several epochs till the redundant filters converge to zeros. Finally, we could obtain a slim model by removing the zero-weight filters.

evaluates the redundant filters easily damages the original filter distributions. Intuitively, if some filters are redundant, they can be safely replaced by their adjacency. Inspired by this, it may be beneficial for network pruning if the adjacent relationships is considered under the soft-pruning scenarios. Formally, let $\mathcal{F}_{pr}$ be the filter set in pre-trained model, and $f(\mathcal{F})$ be the distribution function (the l_2-norm of the filter in this paper) of $\mathcal{F}$. Our soft-pruning method can be formulated as:

$$\begin{aligned} &\min_{\mathcal{F}} \ell(\mathcal{F};(X,Y)) \\ &\text{s.t. } \|f(\mathcal{F})-f(\mathcal{F}_{pr})\|<\epsilon, \mathcal{G}_0(\mathcal{F})\leq\kappa, |\mathcal{F}|=|\mathcal{F}_{pr}| \end{aligned} \tag{1}$$

where $\ell(\cdot)$ is the loss function (*e.g.*, cross-entropy loss and triplet loss); ϵ is a small constant; $\mathcal{G}_0(\mathcal{F})$ refers to the number of non-zeros filters in $\mathcal{F}$; κ denotes the sparse level of the network (*i.e.*, the number of non-zeros filters); X and Y refer to the input data and the target label, respectively; $|\cdot|$ represents the cardinalities of the filter set.

One critical shortcoming in Eq.(1) is that it neglects the block relations of the network. Under this framework, existing soft-pruning methods usually pruning filters in a layer-wise manner, which manually sets the same pruning rate for each layer. Nevertheless, as discussed in Section I, recent CNN networks benefit from block-wise structure, where discriminative information is enhanced block by block. In other words, different blocks may exhibit diverse filter redundancy and deserve different pruning rates. Concretely, given an l-th layer in b-th block with pruning rate P_b^l, we want the layers in the same block to share the same pruning rate. Formally, we want

$$P_b^l = P_b^k, l,k \in \mathcal{B}(b), 1\leq b\leq B, \forall l,k,b, \tag{2}$$

where $\mathcal{B}(b)$ is the index set of filters in the b-th block.

To better utilize the pre-trained knowledge, the initial parameters W_0 (*i.e.*, the weight values in $\mathcal{F}$) used to train the pruned model should be close to the parameters W_{pr} (*i.e.*, the weight values in $\mathcal{F}_{pr}$) in the pre-trained model. In other words, network pruning should minimize the parameter gaps between W_0 and W_{pr} while pruning. Such an observation motivates us to propose a new soft-pruning framework as follows:

$$\begin{aligned} &\min \ell(\mathcal{F};(X,Y)) \\ &\text{s.t. } \|f(\mathcal{F})-f(\mathcal{F}_{pr})\|<\epsilon,\ \mathcal{G}_0(\mathcal{F})\leq\kappa \\ &\quad |\mathcal{F}|=|\mathcal{F}_{pr}|, \|W_{pr}-W_0\|<\delta \\ &\quad P_b^l = P_b^k, l,k\in\mathcal{B}(b), 1\leq b\leq B, \forall l,k,b, \end{aligned} \tag{3}$$

where δ is a small constant.

To solve the problem in Eq.(3), we propose a two-step filter pruning method: the adjacent block-wise pruning and progressive filter decaying. In the adjacent block-wise pruning step, we first select the representative filters for each block across the network. Then, we rank these filters according to their adjacent power to determine the pruning rate of each block. After that, we individually select redundant filters in each convolutional layer according to the pruning rate. In the filter decaying step, we progressively reduce the scale of pruned filters for each layer, followed by fine-tuning to optimize pruned model. This filter decaying operation allows the wrongly pruned filters to recover themselves. Such a "adjacent filter selection, filter decaying and fine-tuning" process is repeated several iterations until all the redundant filters converge to zeros. Finally, we can safely remove these zeroed-weight filters to obtain a compact model. Figure 3 shows the proposed framework.

C. Block-wise Adjacent Filter Selection

(1) Block Redundancy Determination

In the adjacent block-wise pruning stage, we intend to find the redundancy for each block in the pre-trained network. To achieve this goal, we propose to rank the convolutional filters across the whole network according to their adjacent relations. Intuitively, we can thoroughly analyze the redundancy for all the convolutional layers, such as layers with 1x1, 3x3, and 5x5 convolutions. However, this strategy has the following drawbacks. On the one hand, it will consume expensive computation resources due to a large number of convolutional filters. On the other hand, the 1x1 convolution, which is usually used to change the dimensionality, appears

in the plain scale and contains less information for block redundancy evaluation. To cope with these problems, we skip the 1x1 convolutional layers and propose the following filter redundancy evaluation metric:

$$\mathcal{H}(w_i^l) = \begin{cases} \mathbb{E}\big[-\phi(w, w_i^l)\big] & \text{if } w_i^l \notin \mathbf{\Omega}_{1\times1} \\ 0, & \text{otherwise} \end{cases} \quad \text{s.t. } w \in \mathcal{U}(w_i^l), \tag{4}$$

where $\mathbf{\Omega}_{1\times1}$ is a set that contains all the 1x1 convolutional filters in the pre-trained network; $\phi(w, w_i^l)$ refers to the distance measurement function *w.r.t.* w and w_i^l, such as Euclidean distance (used in this paper) and cosine distance; $\mathcal{U}(w_i^l)$ represents the adjacent set of w_i^l, such as k-nearest neighbors of w_i^l. This measurement in Eq.(4) assesses the correlation and its neighbors of w_i^l. A larger value of $\mathcal{H}(w_i^l)$ indicates a higher level of redundancy of w_i^l.

It is worth noticing that the criterion in Eq.(4) is different from that of the previous center-based type pruning methods, which need to find a global reference for all the filters. BAFD chooses the filters themselves as the reference. For example, given four filters A=(1,1), B=(1,0.99), C=(0,0), D=(-2,-2). If we want to prune one of these filters. Center-based methods tend to remove filter C (center reference). When we take a closer look at these filers, we can easily find that filters A and B are very similar and may make similar contributions to the network. Therefore, pruning A or B seems to be more reasonable. BAFD evaluates the adjacent relations of filters and can locate the most redundant filter, *i.e.*, filter B.

We evaluate the redundancy of each filter across the whole network using Eq.(4). Suppose $\mathcal{T}_{vec}^l$ is the adjacent redundancy vector for l-th convolutional layer, *i.e.*, $\mathcal{T}_{vec}^l = \{\mathcal{H}(w_1^l), \mathcal{H}(w_2^l), \cdots, \mathcal{H}(w_{C_{out}^l}^l)\}$. We normalize it for the consistent scales of different convolutional layers. Specifically, we use the following normalization equation:

$$(\hat{\mathcal{T}}_{vec}^l)^{(k)} = \frac{(\mathcal{T}_{vec}^l)^{(k)}}{\sqrt{\sum_j ((\mathcal{T}_{vec}^l)^{(j)})^2}}. \tag{5}$$

After we get the adjacent redundancy vectors for each layer in Eq.(5), we rank the filters across layers in descending order and select the first $p * |\mathcal{F}|$ filters as the redundancy filter set $\hat{\mathcal{F}}$, where p is the global pruning rate for the network. Then, for the b-th block, we collect all the redundant filters in it and calculate its pruning rate as follows:

$$P_b = \mathop{\mathbb{E}}_{k\in\mathcal{B}(b)} \Big[\frac{\sum_{w\in\mathcal{F}^k} \zeta(w, \hat{\mathcal{F}})}{|\mathcal{F}^k|}\Big], \tag{6}$$

where $\zeta(w, \hat{\mathcal{F}}) = 1$ if $w \in \hat{\mathcal{F}}$; $\zeta(w, \hat{\mathcal{F}}) = 0$ otherwise.

(2) Iterative Adjacent Filter Selection

After determining the redundancy for each block, we need to select the redundant filters for each convolutional layer. Given an l-th layer in the b-th block, we want to select and prune $m = P_b^l \times C_{out}^l$ filters, *i.e.*, filters in $\hat{\mathcal{F}}^l$, to compress the network. Formally, we propose to optimize the following objective function:

$$\min_s \sum_{i=1}^{C_{out}^l} \Big\| \mathcal{H}(s_i w_i^l) - \mathcal{H}(w_i^l) \Big\| \quad \text{s.t. } s_i \in \{0,1\}, \sum_{i=1}^{C_{out}^l} s_i = C_{out}^l - m, \tag{7}$$

where s is a filter selection vector. If s_i equals to 1, the filter w_i^l will remain; otherwise, the filter is pruned. $\mathcal{H}(w_i^l)$ and $\mathcal{H}(s_i w_i^l)$ refer to the adjacent redundancy before and after pruning w_i^l, respectively. For each convolutional layer, we hope their adjacent redundancy to be minimized after pruning.

To optimize Eq.(7), we can simply select and prune the first m largest filters according to their redundancy. However, every time we update the filter selection s, the adjacent relationship of remained filters may also be changed. Therefore, this naive optimization may cause the dense area of filters to be deleted. To solve this problem, we propose an iterative optimization approach. Concretely, we sample one filter in $\mathcal{F}^l$ according to Eq. (7), re-evaluate the adjacent redundancy of each remaining filter using Eq.(4) without $w_i^l \notin \mathbf{\Omega}_{1\times1}$. We repeat the optimization procedure till m filters are sampled. If two or more filters reveal the same adjacent geometry power, we calculate their global distance (the global distance of filter w_i^l is the total distance between w_i^l and the other filters in the same layer) and select the filter with the minimum global distance score. The detailed algorithm is provided in Algorithm 1.

D. Progressive Pruning via Filter Decaying

After sampling the pruned filters of each layer, we can directly remove them to eliminate their impact on the network. However, as discussed in Section III-B, the hard-pruning methods tend to cut the network connections prematurely. This may prevent the knowledge transfer between the pre-trained and the pruned models. Although the soft-pruning method can ease this problem by zeroing the redundant filters, they suffer from performance degradation when the parameter gaps between W_{pr} and W_0 become larger. This problem appears to be more severe when a large percentage of filters pruned. To fix this problem, we propose to softly reduce the impacts of pruned filters by decaying their weight scale while pruning. Formally, let $\mathcal{N}^t$ be the network in the t-th pruning iteration, and X_l be the input for the l-th convolutional layer. The output of the l-th layer is defined as $z_l = \sigma_l(W_l, X_l)$, and is the input of the $(l+1)$-th layer, *i.e.*, $z_{l+1} = \sigma_l(W_{l+1}, z_l)$, where $\sigma_l(\cdot)$ is the activation function. Then, after the decay operation, the network can be formulated as:

$$\mathcal{N}^{t+1} = \sigma_L^t\Big([\dot{W}_L^t, \beta\tilde{W}_L^t], \cdots, \sigma_2^t\Big([\dot{W}_2^t, \beta\tilde{W}_2^t], \big(\sigma_1^t([\dot{W}_1^t, \beta\tilde{W}_1^t], X)\big)\Big)\Big) \quad \text{s.t. } 0 \le \beta \le 1, \tag{8}$$

where $\tilde{W}_l$ and $\dot{W}_l$ represent the pruned parameters and retained parameters of the l-th layer, respectively; β refers to

Algorithm 1 Iterative Adjacent Filter Selection

Input: The network filter set $\mathcal{F}^l$ for the l-th convolutional layer in the b-th block; the pruning rate P_b for the b-th block.

Output: The selected redundant filter subset $\hat{\mathcal{F}}^l$ with the best adjacent power preservation.

Initialize the selected subset $\hat{\mathcal{F}}^l \leftarrow \{\}$
Initialize the rest subset $\dot{\mathcal{F}}^l \leftarrow \{w_1^l, w_2^l, \cdots, w_{C_{out}^l}^l\}$
Construct an undirected graph $G = \langle V, E \rangle$ with the vertex set $V = \{w_1^l, w_2^l, \cdots, w_{C_{out}^l}^l\}$ and the edge set E are defined as: $\forall v_i, v_j \in V, 1 \leq i, j \leq |V|$,

$$E_{ij} \leftarrow \begin{cases} \mathcal{H}(v_i), & if\ v_j \in \mathcal{U}(v_i) \\ 0, & \text{Otherwise.} \end{cases}$$

while $|\hat{\mathcal{F}}^l| < P_b \times C_{out}^l$ **do**
  $i^*, j^* = \arg\min_{1 \leq i,j \leq |V|} E_{ij}$
  $k^* = \arg\min_{k \in \{i^*, j^*\}} \sum_{i=1}^{|\dot{\mathcal{F}}^l|} E_{ki}$
  $\hat{\mathcal{F}}^l \leftarrow \hat{\mathcal{F}}^l \bigcup w_{k^*}^l$, $\dot{\mathcal{F}}^l \leftarrow \dot{\mathcal{F}}^l - w_{k^*}^l$
  Delete node v_{k^*} in V and the incident edges from G
end while
return $\hat{\mathcal{F}}^l$

Algorithm 2 Progressive Filter Decaying

Input: training data X and training epoch $epoch_{max}$; the model with parameters W; the pruning rate p and the weight decay parameter β

Output: The compact model and its parameters W^*.

Calculate the pruning rate P_b for b-th ($1 \leq b \leq B$) block by Eq.(6)
for $epoch \leftarrow 1$ **to** $epoch_{max}$ **by** 1 **do**
  Update $W = \{W^l, 1 \leq l \leq L\}$ based on X
  Calculate the redundant filter set $\hat{\mathcal{F}} = \{\hat{\mathcal{F}}^l, 1 \leq l \leq L\}$ by Algorithm 1
  for $l \leftarrow 1$ **to** L **by** 1 **do**
    Get the pruned parameters $\hat{W}^l$ from W^l according to $\hat{\mathcal{F}}^l$
    Decrease the scale of $\hat{W}^l$ by $\hat{W}^l \leftarrow \beta \hat{W}^l$
  end for
end for
return The compact model with optimal parameters W^*

the weight decay parameter. In each pruning phrase, we reduce the scale of the pruned filters for all the layers by Eq.(8), then we fine-tune the pruned network one epoch to let the pruned filters recover themselves.

The "adjacent filter selection, filter decaying, and fine-tuning" process is repeated several times until the redundant filters converge to zeros. To this end, we can safely remove the filters with zero scales from the network and achieve the final compact model. The detailed algorithm is provided in Algorithm 2.

IV. Experiments

We prune three kinds of ResNet-type networks, *i.e.*, ResNet-18, ResNet-34, ResNet-50, on two popular person re-identification datasets, *i.e.*, Market-1501 [57] and DukeMTMC-reID [58]. One person re-identification baseline [21] and three closely related pruning methods are introduced for comparison, including two soft-pruning methods (*i.e.*, SFP [13] and FPGM [14]) and one hard-pruning method (*i.e.*, HFP [12]). Note that this paper mainly focuses on network pruning for person re-identification. In other words, we try to keep the pruned model as small as possible while with limited performance degradation.

*1) **Datasets**:* **Market-1501 [57]** contains 32,668 images, each collected in a university by at most six cameras. All images are automatically detected using the Deformable Part Model (DPM) detector [59]. Market-1501 consists of 1,501 different individuals. These individuals are divided into two groups without overlapping: 751 individuals from 12,936 images for the training set and 750 individuals from 19,732 images for the testing set. There are on average 17.2 photos per individual in the training set. In testing, 3368 images are randomly selected as queries to retrieval the matching person in the testing set.

DukeMTMC-reID [58] is a subset of the well-known DukeMTMC dataset [58] in the format of the Market-1501 dataset [57]. Following [60], we collect 36,411 pedestrian images with IDs annotated by [58]. In total, there are 1,812 identities, where 1,404 identities appear in more than two cameras and 408 identities (distractor ID) walk in front of only one camera. 702 IDs are randomly selected as the training set, and the remaining 702 IDs form the testing set. In the testing set, we pick up one query image for each individual from each camera it appears, to make up the query set. The rest of the testing set is put in the gallery. As a result, we get 16,522 training images, 2,228 query images, and 17,661 gallery images.

MSMT17_V1 [61] is one of the widely-used large-scale person re-identification datasets. This dataset is collected by a 15-camera network deployed in different locations (including indoors and outdoors) of campus, making it much complex and challenging. It contains 126,441 images, which are automatically detected by Faster RCNN [62]. Following [61], 32,621 images are randomly selected for the training set. The rest of the dataset is put in the testing set. From the testing set, 11,659 images are randomly selected as query set and the other 82,161 images are put in the gallery.

*2) **Compared Methods**:* We compare the proposed method with the competitive pruning approaches, including (1) **HFP** [12], the hard-pruning method. It uses the l_1-norm-based criterion to select redundant filters and directly deletes them to get the small model; (2) **SFP** [13], the soft-pruning method. Instead of direct filter deletion, it allows the redundant filters to join the model optimization by setting them to zeros; (3) **FPGM** [14], the soft-pruning via geometric median [14]. It imposes a geometric relation criterion, *i.e.*, the distance between filters and their geometric centers, to evaluate the redundancy of filters.

The pruned model is based on a widely-used person re-identification baseline [21].

*3) **Experimental Setting**:* Following [13], for the layer-wise pruning methods, *i.e.*, HFP [12], SFP [13], and FPGM [14], all the convolutional layers are pruned with the

TABLE I: Performance evaluation of the compared pruning methods on Market-1501 using ResNet-18, ResNet-34, and ResNet-50 backbone networks. In the third column, the letters S and D refer to "same pruning rate for different layers" and "different pruning rates for different layers", respectively.

Depth	Methods	Pruned rate	mAP(%)	Rank-1(%)	Rank-5(%)	Rank-10(%)	Parameters(M)	Parameters(%)↓
18	baseline [21]		65.71	84.65	92.90	95.16	12.08	0
	SFP [13]	S: 20%	63.12	82.19	92.25	94.98	8.89	26.41
	FPGM [14]		63.02	**82.63**	**92.87**	**95.25**		
	HFP [12]		59.96	80.97	91.66	94.74		
	Ours	D: p=0.5	**63.19**	81.29	92.19	94.95	**3.97**	**67.14**
	SFP [13]	S: 50%	58.53	78.71	89.99	93.79	5.01	58.53
	FPGM [14]		59.04	78.53	91.30	**94.27**		
	HFP [12]		52.30	73.72	88.57	92.76		
	Ours	D: p=0.6	**62.28**	**81.38**	**91.66**	94.24	**2.99**	**75.25**
	SFP [13]	S: 90%	46.04	68.38	**85.57**	89.76	1.63	86.51
	FPGM [14]		45.14	67.43	84.62	89.79		
	HFP [12]		41.45	64.34	82.63	88.42		
	Ours	D: p=0.9	**47.50**	**69.06**	85.39	**90.38**	**1.43**	**88.16**
34	baseline [21]		66.88	84.44	93.50	95.61	22.18	0
	SFP [13]	S: 20%	65.60	83.58	**93.26**	**95.48**	16.18	27.05
	FPGM [14]		64.82	82.75	92.96	95.31		
	HFP [12]		63.17	82.01	92.04	94.83		
	Ours	D: p=0.5	**68.13**	**84.38**	93.17	94.46	**6.89**	**68.94**
	SFP [13]	S: 50%	61.30	80.40	90.80	93.88	8.80	60.32
	FPGM [14]		61.66	80.29	91.36	94.77		
	HFP [12]		56.32	76.60	89.46	92.73		
	Ours	D: p=0.6	**67.23**	**84.80**	**92.87**	**95.60**	**5.14**	**76.83**
	SFP [13]	S: 90%	49.96	71.88	86.76	90.83	2.19	90.13
	FPGM [14]		50.05	72.47	86.85	90.94		
	HFP [12]		47.29	70.01	85.75	90.14		
	Ours	D: p=0.9	**56.79**	**76.63**	**89.31**	**93.02**	**1.81**	**91.84**
50	baseline [21]		70.81	86.63	93.74	96.05	27.10	0
	SFP [13]	S: 20%	69.03	84.77	93.74	95.69	20.37	24.83
	FPGM [14]		70.07	86.07	93.79	95.81		
	HFP [12]		67.93	85.51	93.97	96.14		
	Ours	D: p=0.5	**71.71**	**87.11**	**94.63**	**96.70**	**11.31**	**58.27**
	SFP [13]	S: 50%	65.85	83.05	92.96	95.52	12.64	53.36
	FPGM [14]		65.31	83.02	91.98	94.66		
	HFP [12]		57.22	77.82	90.29	93.74		
	Ours	D: p=0.6	**70.07**	**86.02**	**93.94**	**96.20**	**9.52**	**64.87**
	SFP [13]	S: 90%	48.02	71.17	86.88	90.91	6.97	74.28
	FPGM [14]		45.31	68.26	84.74	89.64		
	HFP [12]		47.24	70.57	85.09	89.85		
	Ours	D: p=0.9	**57.80**	**77.02**	**90.02**	**93.62**	**6.77**	**75.02**

same pruning rate at the same time. Besides, following [48], to keep the residual structure of the original network, we skip the downsample layer while pruning. The pruning rate is tuned from $\{20\%, 50\%, 90\%\}$. For our method, we tune the pruning rate parameter p from $\{0.1, 0.2, \cdots, 0.8, 0.9\}$ and choose three values, *i.e.*, $\{0.5, 0.6, 0.9\}$, which achieve the similar accuracy as the other layer-wise methods. Note that in our experiment, we impose a round-down strategy for pruning. For example, if the pruning rate is 20% and the number of the filter is 64, then the actual number of pruned filters is $\lfloor 64 \times 0.2 \rfloor = \lfloor 12.8 \rfloor = 12$.

For the soft-pruning methods, *i.e.*, SFP, FPGM, and our method, we embed the pruning operation into the training procedure and prune filters while fine-tuning for 100 epochs. To conduct a fair comparison, for the hard-pruning method, *i.e.*, HFP, we prune the network once and fine-tune it with the same number of epochs as the soft-pruning methods. There are three constraints controlled by κ, ϵ, and δ, in the objective function of our method in Eq.(1). For κ, we set it to $(1-p)|\mathcal{F}|$. For ϵ and δ, it is hard to directly assign precise values for them. Alternately, since ϵ and δ are used to control the progress of adjacent filter selection and redundant filter decaying, in practice, we dynamically monitor the status of pruning and perform an early stop if these constraints are met. Concretely, if the $\hat{\mathcal{F}}^l$ in Algorithm 1 and $\|\hat{W}^l\|$ in Algorithm 2 for every l-th convolutional layer are no longer changed in 5 consecutive epochs, then we assume that the constraints $\|f(\mathcal{F}) - f(\mathcal{F}_{pr})\| < \epsilon$ and $\|W_{pr} - W_0\| < \delta$ are met, respectively. Our method contains one hyper-parameter, *i.e.*, β. We choose the optimal β according to the sensitive analysis in Section IV-D3. Concretely, we set $\beta = 1 \times 10^{-2}$ when $p < 0.7$ and $\beta = 3 \times 10^{-1}$ when $p \geqslant 0.7$.

Following [21], we deploy ResNet-18, ResNet-34, and ResNet-50 that are pre-trained on ImageNet [63] as the backbone networks for our person re-identification baseline, and replace the last average pooling layer and fully-connected layer with an adaptive max-pooling layer. We employ the widely used triplet loss function [64] and SGD to train the network with an initial learning rate $l_0 = 0.001$, momentum 0.9, margin 0.3, pool size 128, and batch size 32.

TABLE II: Performance comparison of different pruning methods on DukeMTMC-reID using ResNet-18, ResNet-34, and ResNet-50 backbone networks. In the third column, the letters S and D refer to "same pruning rate for different layers" and "different pruning rates for different layers", respectively. "-" means the models fail to converge while training.

Depth	Methods	Pruned rate	mAP(%)	Rank-1(%)	Rank-5(%)	Rank-10(%)	Parameters(M)	Parameters(%)↓
18	baseline [21]		55.87	75.36	86.45	89.90	12.05	0
	SFP [13]	S: 20%	53.37	74.33	**85.95**	**89.63**	8.87	26.39
	FPGM [14]		53.65	73.43	85.14	88.69		
	HFP [12]		49.78	70.74	83.98	87.43		
	Ours	D: p=0.5	**54.07**	**74.37**	85.91	89.18	**3.94**	**67.30**
	SFP [13]	S: 50%	49.46	70.29	82.90	86.94	4.99	58.59
	FPGM [14]		47.80	69.03	82.72	86.62		
	HFP [12]		43.07	64.14	78.82	83.44		
	Ours	D: p=0.6	**52.36**	**72.53**	**85.01**	**88.33**	**2.96**	**75.44**
	SFP [13]	S: 90%	-	-	-	-	1.61	86.64
	FPGM [14]		-	-	-	-		
	HFP [12]		34.23	55.66	73.07	78.95		
	Ours	D: p=0.9	**40.96**	**62.66**	**79.00**	**83.75**	**1.41**	**88.30**
34	baseline [21]		56.38	74.69	86.31	89.50	22.16	0
	SFP [13]	S: 20%	55.22	74.60	**85.50**	**89.23**	16.16	27.08
	FPGM [14]		54.42	73.29	84.61	88.73		
	HFP [12]		52.85	73.15	85.10	88.29		
	Ours	D: p=0.5	**58.74**	**76.80**	87.34	90.40	**6.86**	**69.04**
	SFP [13]	S: 50%	51.18	72.08	83.62	87.34	**8.78**	60.38
	FPGM [14]		52.05	71.72	83.75	87.52		
	HFP [12]		47.26	68.04	81.73	86.22		
	Ours	D: p=0.6	**57.83**	**76.12**	**88.11**	**90.84**	**5.11**	**76.94**
	SFP [13]	S: 90%	41.85	64.59	79.49	84.43	2.17	90.21
	FPGM [14]		40.46	62.75	78.82	84.02		
	HFP [12]		37.94	59.42	76.62	81.96		
	Ours	D: p=0.9	**50.35**	**70.96**	**83.30**	**87.61**	**1.79**	**91.92**
50	baseline [21]		59.63	77.56	87.84	91.02	27.00	0
	SFP [13]	S: 20%	59.55	77.24	87.57	90.98	20.27	24.93
	FPGM [14]		60.00	78.86	88.47	90.89		
	HFP [12]		58.67	78.37	88.64	91.16		
	Ours	D: p=0.5	**62.36**	**80.39**	**89.36**	**92.10**	**11.21**	**58.48**
	SFP [13]	S: 50%	55.57	75.13	86.13	89.59	12.54	53.56
	FPGM [14]		55.94	74.60	86.22	89.90		
	HFP [12]		46.79	68.31	81.24	86.18		
	Ours	D: p=0.6	**60.60**	**78.68**	**87.97**	**91.38**	**9.42**	**65.11**
	SFP [13]	S: 90%	-	-	-	-	6.87	74.56
	FPGM [14]		-	-	-	-		
	HFP [12]		36.05	57.59	75.18	80.97		
	Ours	D: p=0.9	**50.37**	**70.02**	**83.66**	**87.66**	**6.67**	**75.30**

A. Results on the Market-1501 Dataset

Table I shows the comparison results on the Market-1501 dataset. We can conclude that:

- In most cases, the soft-pruning methods, *i.e.*, SFP, FPGM, and ours, outperform the hard-pruning method, *i.e.*, HFP. This indicates that by keeping the connections of pruned filters while fine-tuning, the soft-pruning methods can give the potentially important filters more chances to recover themselves. Thus, it can boost the performance of network pruning.
- Our method consistently achieves the best mAP over other comparison methods with all the selected backbone networks, indicating the advantage of our method. For example, on ResNet-34, our method surpasses the second best result over 6% in terms of mAP while removing over 91.8% parameters, *i.e.*, with only 1.81M left.
- With the increase of parameter gaps (*i.e.*, the rise of the pruning rate) between the pruned network and the pre-trained network, the advantages of our method over the compared soft-pruning methods are increasing. For example, on ResNet-50, our method outperforms the compared soft-pruning methods by over 1.6% on mAP when the parameter gap is 20% (p = 0.5 for BAFD). If we increase the parameter gap to 90% (p = 0.9 for BAFD), the advantages of our method over the other soft-pruning methods reach 9.7% on mAP. This demonstrates that our method is suitable for pruning a large proportion of filters. It should be noticed that SFP and FPGM are hard to converge while training on ResNet-50 when the abovementioned parameter gaps become extremely large, *i.e.*, 90% filters are pruned, which is similar to the results on DukeMTMC-reID. We show detailed reason in Section IV-B. Here we repeatedly optimize SFP and FPGM till the convergent models appear and report the best results.

B. Results on the DukeMTMC-reID Dataset

We also testify all the pruning methods on DukeMTMC-reID. Table II shows the comparison results, where the "-" refers to "fail to convergence" while running the experiment. We have the following observations:

TABLE III: Performance comparison of different pruning methods on MSMT17_V1 using ResNet-18, ResNet-34, and ResNet-50 backbone networks. In the third column, the letters S and D refer to "same pruning rate for different layers" and "different pruning rates for different layers", respectively. "-" means the models fail to converge while training.

Depth	Methods	Pruned rate	mAP(%)	Rank-1(%)	Rank-5(%)	Rank-10(%)	Parameters(M)	Parameters(%)↓
18	baseline [21]		23.31	49.11	66.85	73.21	12.22	0
	SFP [13]	S: 20%	17.70	31.48	45.17	51.40	9.04	26.02
	FPGM [14]		16.83	30.72	44.11	50.26		
	HFP [12]		15.99	29.66	43.14	49.02		
	Ours	D: p=0.5	**18.09**	**31.56**	**45.56**	**52.80**	**4.12**	**66.29**
	SFP [13]	S: 50%	14.99	27.64	41.53	47.76	5.16	57.77
	FPGM [14]		14.94	27.22	41.89	48.22		
	HFP [12]		9.81	19.62	32.59	38.86		
	Ours	D: p=0.6	**17.93**	**31.07**	**45.96**	**52.36**	**3.14**	**74.30**
	SFP [13]	S: 90%	-	-	-	-	-	-
	FPGM [14]		-	-	-	-		
	HFP [12]		-	-	-	-		
	Ours	D: p=0.9	**-**	**-**	**-**	**-**	-	-
34	baseline [21]		24.80	50.42	67.44	74.39	22.33	0
	SFP [13]	S: 20%	19.07	33.17	47.22	52.96	16.33	26.87
	FPGM [14]		18.42	32.31	46.11	51.82		
	HFP [12]		18.06	32.31	46.08	52.19		
	Ours	D: p=0.5	**22.81**	**37.78**	**51.43**	**57.34**	**7.04**	**68.47**
	SFP [13]	S: 50%	15.98	28.87	41.93	47.97	**8.95**	59.92
	FPGM [14]		16.13	29.09	43.01	48.86		
	HFP [12]		10.55	20.76	34.13	40.12		
	Ours	D: p=0.6	**21.74**	**36.62**	**50.46**	**56.34**	**5.29**	**76.31**
	SFP [13]	S: 90%	-	-	-	-	-	-
	FPGM [14]		-	-	-	-		
	HFP [12]		-	-	-	-		
	Ours	D: p=0.9	**14.33**	**26.13**	**40.94**	**47.03**	**1.96**	**91.22**
50	baseline [21]		27.63	54.93	71.52	77.27	27.69	0
	SFP [13]	S: 20%	21.67	36.32	50.61	56.48	20.96	24.30
	FPGM [14]		21.85	36.00	49.73	55.42		
	HFP [12]		21.43	36.13	49.65	55.30		
	Ours	D: p=0.5	**26.46**	**42.44**	**56.06**	**61.57**	**11.91**	**56.99**
	SFP [13]	S: 50%	15.98	28.87	41.93	47.97	13.23	52.22
	FPGM [14]		19.06	32.97	46.76	52.79		
	HFP [12]		10.16	19.79	32.68	38.95		
	Ours	D: p=0.6	**23.94**	**39.57**	**53.27**	**58.63**	**10.12**	**65.11**
	SFP [13]	S: 90%	-	-	-	-	-	-
	FPGM [14]		-	-	-	-		
	HFP [12]		-	-	-	-		
	Ours	D: p=0.9	**13.66**	**25.41**	**39.67**	**46.31**	**7.36**	**73.42**

- Similar to the results on Market-1501, our method consistently achieves the best performance in terms of mAP and $Rank$-1, especially when a large number of filters is pruned. For example, our method outperform HFP 7.7% when removing 88.3% parameters (1.41M left) on ResNet-18.
- In some cases, the soft-pruning methods, *i.e.*, SFP and FPGM, fail to converge when a large proportion of filters is pruned, *i.e.*, 90% filters on ResNet-18 and ResNet-50. The reason may lie in that the soft-pruning methods keep the pruned filters while training, which maintains the same volume of parameters as the pre-trained model. When a large number of filters is set to zeros, the initial parameters of the pruned network have a large gap to that of the pre-trained network. Such a parameter gap may be insufficient for the pruned network to keep the pre-trained knowledge in a large parameter space. In contrast, the hard-pruning method, *i.e.*, HFP, directly removes the redundant filters and generates the compact model, which optimizes the model in small parameter space (with fewer parameters) and converges easily even with part of the pre-trained parameters.
- Our method introduces a filter decaying strategy, giving the pruned model more chances to utilize the pre-trained knowledge. Therefore, it can get stable results than the others. For example, we reduce 91.9% parameters (1.79M left) only with an increase of 3.7% $rank$-1 error on ResNet-34.

C. Results on the MSMT17_V1 Dataset

Table III shows the comparison results on the large-scale re-identification dataset MSMT17_V1. We have the following observations:

- The soft-pruning methods, *i.e.*, SFP, FPGM, and ours, invariably outperform the hard-pruning method, *i.e.*, HFP, on all types of networks. Note that the data collection scenarios in MSMT17_V1 are quite more complex than

that of Market-1501 and DukeMTMC-reID. This indicates that by keeping the connections of pruned filters while fine-tuning, the soft-pruning methods can retain the original model capacity as much as possible and are more adaptive to complex datasets.

- Due to the complexity of MSMT17_V1, when removing a large proportion of filters, *e.g.*, 90% filters, all the compared methods, *i.e.*, SFP, FPGM, and HFP, suffer from convergence problems on all the networks. Despite failures on ResNet18 with $p = 0.9$, our method still performs well on the other two networks, *i.e.*, ResNet34 and ResNet50. This again demonstrates the advantage of our method.

D. Ablation Study

Our method contains three main components, *i.e.*, adjacent filter selection, block-wise pruning, and filter decaying. In this section, we provide case studies to help understand these components.

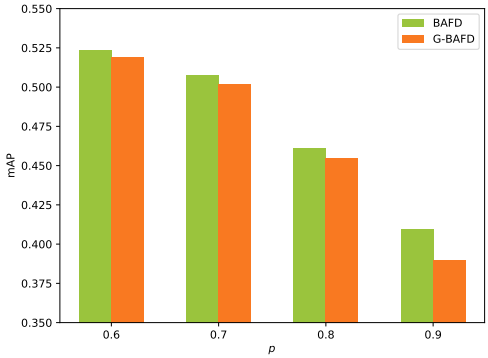


Fig. 4: Influence analysis of adjacent filter selection in the proposed method. BAFD with adjacent filter criterion is suitable for removing a large number of filters with smaller mAP dropping over G-BAFD, which selects redundant filters using the center-based criterion in FPGM [14].

*1) **Influence of Adjacent Filter Selection**:* Our method imposes the adjacent filter selection to locate the redundant filters and retains the original filter distributions in the pre-trained network. To study the effectiveness of adjacent filter selection, we replace it with the global-geometric selection in FPGM [14], namely G-BAFD. G-BAFD evaluates the importance of filters according to the distance to their geometry centers. We report the pruning performance of BAFD and G-BAFD on ResNet-18 with pruning rate tuned from $\{0.6, 0, 7, 0.8, 0.9\}$ on the DukeMTMC-reID dataset. The setting of other parameters is following Section IV-3. Figure 4 shows the comparison results. We can find that BAFD consistently outperforms G-BAFD on the selected pruning rate. Particularly, when the pruning rate is increasing, BAFD achieves much higher mAP over G-BAFD. Besides, we also report the filter distributions (the l_2-norm of filters) of different methods before and after pruning in Figure 5. In this experiment, we remove the block-wise pruning component in BAFD and obtain a layer-wise version like SFP. The pruning rate is set to $P = 50\%$ for SFP and $p = 0.5$ for BAFD. The filters are collected from the last convolutional layer of ResNet34. We can see that the center-based method, *i.e.*, SFP (green curve), tends to prune small-scale filters (close to zero), leading to a larger bias (marked by green box) compared with the original distribution (orange curve) than BAFD (purple curve). This observation demonstrates that our adjacent filter selection mechanism is more suitable for removing a large proportion of filters while retaining the original filter distribution.

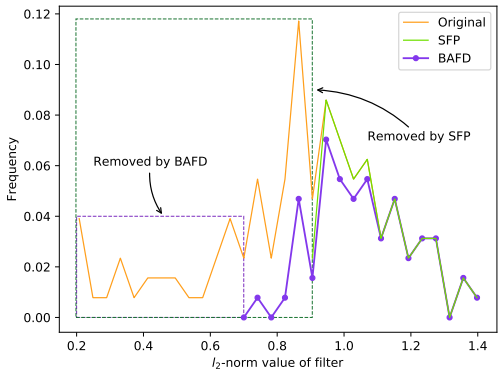


Fig. 5: Filter distributions of different methods before and after pruning. The center-based method, *i.e.*, SFP (green curve), can only keep the small-scale filters whose norm values are bigger than 0.9, resulting in a large disparity in filter distribution. In contrast, BAFD (purple curve) evaluates the redundancy of filters according to their local adjacent power, which can preserve more small-scale filters (the norm values are bigger than 0.7) and better retain the original filter distribution (orange curve).

*2) **Influence of Block-wise Pruning**:* Our method prunes filters in a block-wise manner. Concretely, the convolutional layers in the same block share the same pruning rate, while those in different blocks cover diverse pruning rates. To investigate its validity, we manually set the same pruning rate for all the blocks across the whole network, and named the new method as L-BAFD. Then, we test L-BAFD with the pruning rate tuned from $\{0.5, 0.6, \cdots, 0.9\}$ on DukeMTMC-reID using ResNet-18. Figure 6 illustrates the comparison results, where we use the size of the marker to reveal the number of parameters (a bigger marker indicates the more parameters). We can find that our method consistently outperforms L-BAFD in terms of mAP while maintaining fewer parameters. This demonstrates the efficiency of our block-wise pruning strategy.

To better explain how the redundant filters are selected, we record the number of pruned filters of the convolutional layers in each block. ResNet-50 is chosen for the backbone network on Market-1501 with pruning rate tuned from $\{0.5, 0.6, 0.9\}$. Figure 7 shows the results. Clearly, our method tends to prune filters in high-level blocks. This observation shows a consistent result with the previous studies [47], which indicates that the deep blocks are prone to encode the high-level semantic information of the input images and contain relatively high rates of redundancy compared with the lower blocks. Such observation also again demonstrates the effectiveness of our block ranking approach.

*3) **Influence of Filter Decaying**:* Our method utilizes a filter decaying strategy to prune and reduce the impact of the redundant filters on the network. Filter decaying helps

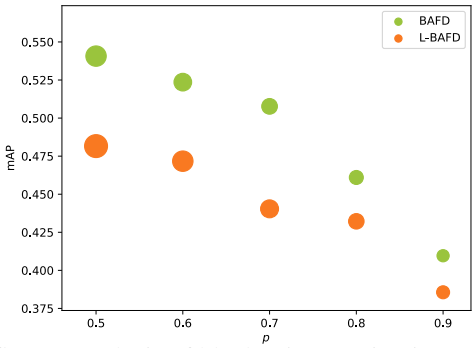


Fig. 6: Influence analysis of block-wise pruning in BAFD. The larger marker size indicates more parameters remaining. We can observe that BAFD is superior to L-BAFD, which sets the same pruning rate for all the blocks across the whole network.

the pruned network to keep the pre-trained knowledge. To understand its sensitivity, we evaluate BAFD with β tuned from $\{6\times10^{-1}, 3\times10^{-1}, 1\times10^{-1}, 6\times10^{-2}, 3\times10^{-2}, 1\times10^{-2}, 0\}$ on three validation sets using ResNet-50. Following [65], these validation sets are split out from the training set of Market-1501, DukeMTMC-reID, and MSMT17_V1, respectively. For each validation set on the selected dataset, the last 100 classes are selected for validation, and the rest classes are used for training. The pruning rate p is tuned from $\{0.4, 0.5, 0.6, \cdots, 0.8, 0.9\}$. The results are shown in Figure 8. We observe that the impact of β is slight when p is small, *e.g.*, $p < 0.7$. In contrast, when a large proportion of parameters is dropped, *e.g.*, $p \geq 0.7$, BAFD is sensitive to β. Concretely, a larger β decreases the filter scale slowly and enhances performance. This is consistent with our intuition that the filter decaying helps the model adapt to the large prune rate. Note that if the value of β is too large, *e.g.*, $\beta = 6\times10^{-1}$, the performance may degrade significantly. In this case, the network may converge very slowly, resulting in limited redundant filters are decreased to zeros. For a fair comparison, following the hard-pruning methods, all the final redundant filters are dropped to obtain the compact model. Thus, the model with $\beta = 6\times10^{-1}$ does not perform well.

To have a clear illustration of how filter decaying works, we report the pruning performance *w.r.t.* the fine-tuning epochs. In this experiment, we manually remove filter decaying (*i.e.*, $\beta = 0$) from BAFD to form a new method, namely Non-BAFD, and test it on various ResNet-type networks. Suppose $\mathcal{M}_B^i$ and $\mathcal{M}_N^i$ be the mAP for BAFD and Non-BAFD in the i-th training epoch, respectively. The results are shown in Figure 9. There are several observations:

- At the early stage of fine-tuning phase, *i.e.*, the first 10 epochs, Non-BAFD performs better than BAFD. This is because Non-BAFD directly zeros redundant filters to get a relatively stable network structure. Such a steady network structure makes Non-BAFD efficient for training and superior to BAFD.
- BAFD imposes the filter decaying strategy to smoothly reduce the filter scales, which gives the selected filters more chances to recover themselves and consumes more epochs to converge at the early training stage, *e.g.*, in the first 10 epochs on ResNet-34.
- BAFD and Non-BAFD spend a similar number of epochs to converge on the selected networks, *i.e.*, 70 for ResNet-18, 50 for ResNet-34, and 50 for ResNet-50. This demonstrates that our filter decaying strategy does not increase the total fine-tuning epochs compared with the soft-pruning strategy.
- BAFD constantly outperforms Non-BAFD after fine-tuning 60 epochs on three selected networks. Considering BAFD imposes the filter decaying to shrink the parameter gaps between W_0 and W_{pr}, this results demonstrate the effectiveness of our filter decaying, which is beneficial for pruning.

E. Pruning Efficiency Analysis

In this experiment, we study the efficiency of our method in training and inferencing on Market-1501. For training, we independently run all the pruning methods on ResNet50 100 epochs on an Intel (R) Core (TM) i9-9980XE CPU @ 3.00GHz PC with 64G memory and a GPU (Nvidia 2080Ti). The average training time for each epoch is reported, *i.e.*, 2.95 minutes, 3.13 minutes, and 3.34 minutes for SFP, FPGM, and our method, respectively. Therefore, compared with other soft-pruning methods, the increase in computational complexity of our method is limited and negligible.

Previous works have theoretically analyzed the speedup of conventional layer-wise pruning methods [48]. However, as different layers in BAFD may have different pruning rates, it is hard to compute the theoretical speedup ratio. To test the efficiency of pruned models, we record the inference time of various kinds of networks with 90% filters (p is set to 0.9 for BAFD) are pruned. For the soft-pruning methods, *i.e.*, SFP, FPGM, and ours, they keep the redundant filters while pruning. We remove the zeroed filters directly to obtain the slim models after they converge. All the slim models are run 10 times on an Intel Dual-Core i5 @ 1.3GHz PC with 4G memory, and the average inference time is recorded. As can be seen in Table IV, our method achieves superior performance and reduces more parameters over the other methods. For example, our method surpasses the second best result over 4% on $Rank$-1 by reducing 91.84% parameters using RestNet-34. For model inference, our method slightly spends more inference time than FPGM, *e.g.*, 8.66ms on ResNet-34. As discussed in Section IV-D2, our method tends to prune filters in the high-level convolutional layers, which is adept at reducing parameters rather than computation cost.

F. Feature Maps Visualization

To further understand the mechanism of BAFD, we apply it to ResNet-50 and visualize the feature maps of the first convolutional layer of the pruned network. In this experiment, we set the pruning rate $p = 0.5$ and $\beta = 1\times10^{-2}$. As shown in Figure 10, we provide visualization results, where the pruned feature maps are marked with the red box. Clearly, we can see that the strong similarity yields up over the pruned feature maps. For example, feature maps (2,20,41,44) show close relations, which demonstrates the existence of redundancy in the ResNet-50 network and the necessity of network pruning.

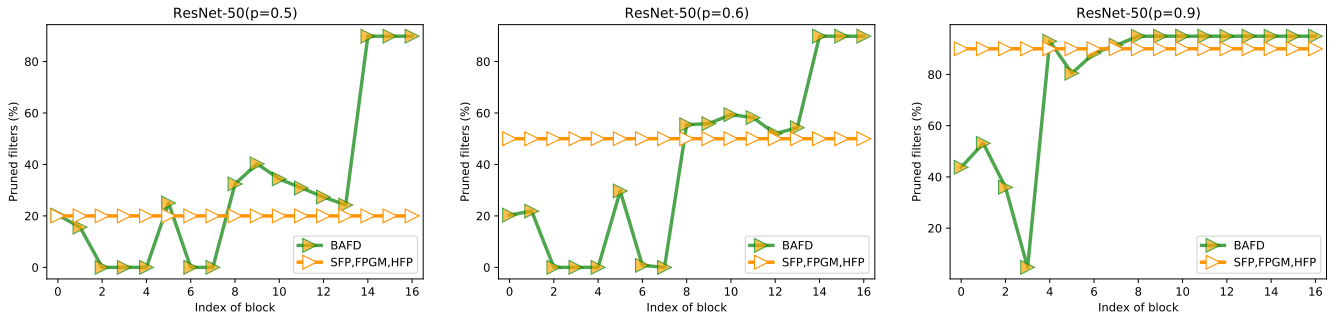


Fig. 7: The pruned filter statistics of our method on ResNet-50 with pruning rate p tuned from $\{0.5, 0.6, 0.9\}$. We observe that our method tends to prune redundant filters in high-level blocks.

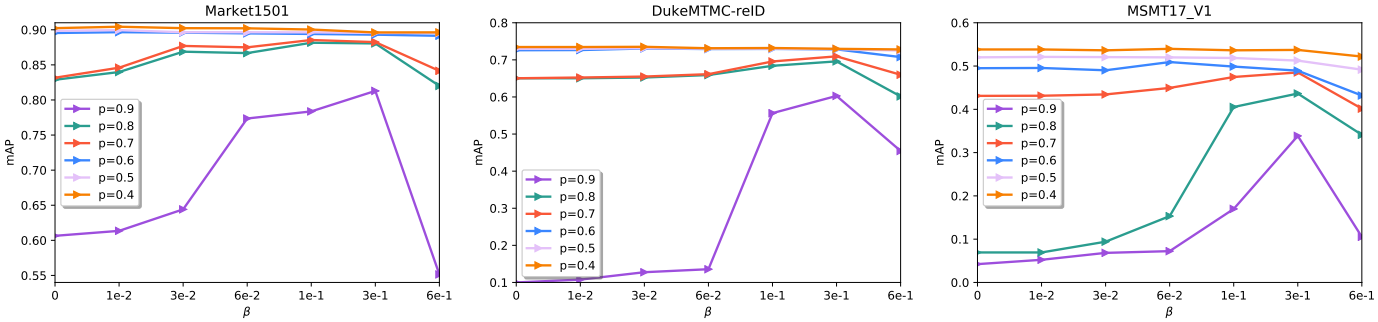


Fig. 8: Filter decaying analysis with different β. The impact of β is slight when the pruning rate is small, *i.e.*, $p < 0.7$. In contrast, larger β may be better if $p \geq 0.7$.

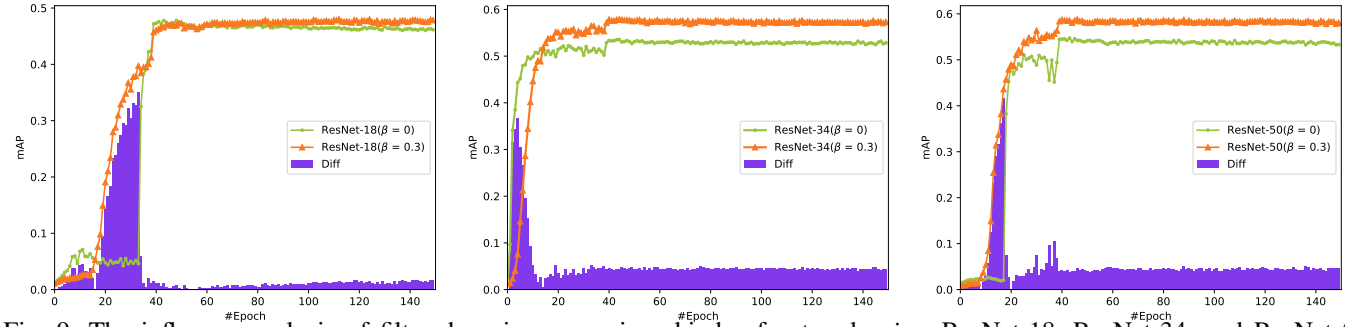


Fig. 9: The influence analysis of filter decaying on various kinds of networks, *i.e.*, ResNet-18, ResNet-34, and ResNet-50, from left to right with 90% filters pruned. "Diff" refers to the absolute difference in mAP between our pruning methods with and without filter decaying. Our filter decaying strategy could boost the pruning performance, especially when large amounts of filters are pruned on ResNet-34 and ResNet-50.

What is more, the pruned feature maps can be replaced by the retained filters. For example, feature maps (2,11,39) can be replaced by feature maps (8,40,36), respectively. This clearly shows that our method can slim the CNN network while maintaining the original filter distribution.

V. Conclusion

In this paper, we propose a novel block-wise adjacent filter pruning method, namely BAFD, to address the problem of pruning re-id models. In particular, BAFD combines adjacent filter selection, block ranking, and filter decaying into a joint soft-pruning framework. Different from existing works, BAFD is able to retain the filter distribution and block relations of the original model at the same time. More importantly, unlike existing methods that usually delete or zero redundant filters directly, BAFD is much "softer" by progressively reducing the scale of redundant filters. It outperforms other prevailing pruning methods on three popular person re-ID benchmarks and shows potential advantages when a large proportion of filters is pruned.

References

[1] X. Sun and L. Zheng, "Dissecting person re-identification from the viewpoint of viewpoint," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 608–617.

[2] D. Cheng, Z. Li, Y. Gong, and D. Zhang, "Fusion of multiple person re-id methods with model and data-aware abilities," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 561–571, 2020.

[3] K. Lin, J. Lu, C. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1183–1192.

[4] L. Zhang, T. Xiang, and S. Gong, "Learning a deep embedding model for zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2021–2030.

[5] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1017–1027, 2017.

TABLE IV: Efficiency comparison of different pruning methods on Market-1501 using three kinds of ResNet-type networks with 90% filters pruned. "FLOPs" refers to the floating point operations. "I-t" and "paras" refer to the "inference time per image" and the number of parameters, respectively. Since our approach tends to prune the high-level redundant filters instead of the low-level filters, the pruned model may cost more inference time in the low-level layers. We note that the time cost is still much less than that of the original baseline model. Despite spending a little more inference time than other pruning methods, our pruned model saves the model storage cost (fewer parameters) and achieves much better retrieval performance.

Methods	type	ResNet-18				ResNet-34				ResNet-50			
		Rank-1	paras↓	FLOPs↓	I-t	Rank-1	paras↓	FLOPs↓	I-t	Rank-1	paras↓	FLOPs↓	I-t
Baseline [21]	-	84.65%	-	-	75ms	84.44%	-	-	134ms	86.63%	-	-	186ms
HFP [12]	hard	64.34%				70.01%				70.57%			
SFP [13]	soft	68.38%	86.51%	**87.39%**	**24ms**	71.88%	90.13%	**90.83%**	**34ms**	71.17%	74.28%	**88.85%**	**46ms**
FPGM [14]	soft	67.43%				72.47%				68.26%			
Ours	soft	**69.06%**	**88.16%**	75.63%	29ms	**76.63%**	**91.84%**	80.42%	42ms	**77.02%**	**75.02%**	79.55%	67ms

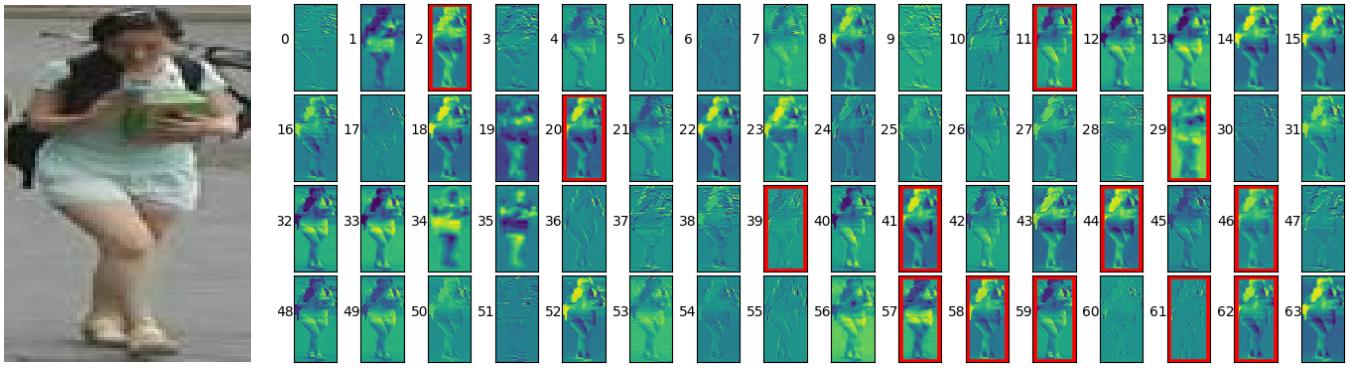


Fig. 10: The input image (**left**) and the visualization of the first convolutional layer feature maps (indexed from 0 to 63) (**right**). The maps with red boxes are selected by our method. For example, feature maps (2,11) share similar patterns with feature maps (8,40), respectively, and can be safely pruned.

[6] L. Zeng and X. Tian, "Accelerating convolutional neural networks by removing interspatial and interkernel redundancies," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 452–464, 2020.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016.

[8] R. Yu, A. Li, C. Chen, J. Lai, V. I. Morariu, X. Han, M. Gao, C. Lin, and L. S. Davis, "NISP: pruning networks using neuron importance score propagation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9194–9203.

[9] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1398–1406.

[10] J. Park, S. R. Li, W. Wen, P. T. P. Tang, H. Li, Y. Chen, and P. Dubey, "Faster cnns with direct sparse convolutions and guided pruning," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[11] Z. Liu, J. Qin, A. Li, Y. Wang, and L. Van Gool, "Adversarial binary coding for efficient person re-identification," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019, pp. 700–705.

[12] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[13] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018.

[14] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[15] C. Deng, E. Yang, T. Liu, J. Li, W. Liu, and D. Tao, "Unsupervised semantic-preserving adversarial hashing for image search," *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 4032–4044, 2019.

[16] C. Deng, E. Yang, T. Liu, and D. Tao, "Two-stream deep hashing with class-specific centers for supervised image search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2189–2201, 2020.

[17] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2014, pp. 818–833.

[18] E. Belilovsky, M. Eickenberg, and E. Oyallon, "Greedy layerwise learning can scale to imagenet," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 583–593.

[19] W. Wang, S. Zhao, M. Chen, J. Hu, D. Cai, and H. Liu, "Dbp: Discrimination based block-level pruning for deep model acceleration," *arXiv preprint arXiv:1912.10178*, 2019.

[20] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[21] Z. Zheng, X. Yang, Z. Yu, L. Zheng, Y. Yang, and J. Kautz, "Joint discriminative and generative learning for person re-identification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[22] X. Li, M. Chen, F. Nie, and Q. Wang, "A multiview-based parameter free framework for group detection," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 4147–4153.

[23] Y. Xu, X. Liu, L. Qin, and S. Zhu, "Cross-view people tracking by scene-centered spatio-temporal parsing," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 4299–4305.

[24] L. Zheng, Y. Yang, and A. G. Hauptmann, "Person re-identification: Past, present and future," *CoRR*, vol. abs/1610.02984, 2016.

[25] Z. Zheng, L. Zheng, and Y. Yang, "A discriminatively learned CNN embedding for person reidentification," *ACM Trans. Multim. Comput. Commun. Appl.*, vol. 14, no. 1, pp. 13:1–13:20, 2018, doi:10.1145/3159171.

[26] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1318–1327.

[27] X. Qian, Y. Fu, Y.-G. Jiang, T. Xiang, and X. Xue, "Multi-scale deep learning architectures for person re-identification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5399–5408.

[28] X. Qian, Y. Fu, T. Xiang, W. Wang, J. Qiu, Y. Wu, Y.-G. Jiang, and X. Xue, "Pose-normalized image generation for person re-identification," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 650–667.

[29] F. Yang, Z. Zhong, Z. Luo, S. Lian, and S. Li, "Leveraging virtual and real person for unsupervised person re-identification," *IEEE Trans. Multimedia*, pp. 1–1, 2019.

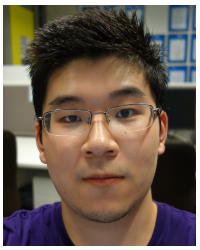
[30] C. Shen, Z. Jin, W. Chu, R. Jiang, Y. Chen, G. Qi, and X. Hua, "Multi-level similarity perception network for person re-identification," *ACM Trans. Multim. Comput. Commun. Appl.*, vol. 15, no. 2, pp. 32:1–32:19, 2019.

[31] X. Qian, Y. Fu, T. Xiang, Y.-G. Jiang, and X. Xue, "Leader-based multi-scale attention deep architecture for person re-identification," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

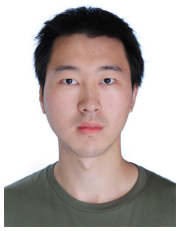
[32] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing

data augmentation." in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 13 001–13 008.
[33] Y. Ding, H. Fan, M. Xu, and Y. Yang, "Adaptive exploration for unsupervised person re-identification," *ACM Trans. Multim. Comput. Commun. Appl.*, vol. 16, no. 1, pp. 3:1–3:19, 2020.
[34] Z. Zheng, L. Zheng, and Y. Yang, "Pedestrian alignment network for large-scale person re-identification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 10, pp. 3037–3045, 2018, doi:10.1109/TCSVT.2018.2873599.
[35] M. Jiang, Y. Yuan, and Q. Wang, "Self-attention learning for person re-identification," in *BMVC*, 2018.
[36] Y. Feng, Y. Yuan, and X. Lu, "Person reidentification via unsupervised cross-view metric learning," *IEEE Trans. Cybern.*, pp. 1–11, 2019.
[37] K. Lin, J. Lu, C. Chen, J. Zhou, and M. Sun, "Unsupervised deep learning of compact binary descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 6, pp. 1501–1514, 2019.
[38] 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. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8514–8522.
[39] H. Tang, S. Bai, L. Zhang, P. H. Torr, and N. Sebe, "Xinggan for person image generation," *arXiv preprint arXiv:2007.09278*, 2020.
[40] Y. Ge, Z. Li, H. Zhao, G. Yin, S. Yi, X. Wang, and H. Li, "FD-GAN: pose-guided feature distilling GAN for robust person re-identification," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 1230–1241.
[41] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang, "Camera style adaptation for person re-identification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5157–5166.
[42] 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. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9021–9030.
[43] Y. Yuan, J. Zhang, and Q. Wang, "Deep gabor convolution network for person re-identification," *Neurocomputing*, vol. 378, pp. 387–398, 2020.
[44] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605.
[45] S. Srinivas and R. V. Babu, "Data-free Parameter Pruning for Deep Neural Networks," in *Proc. Brit. Mach. Vis. Conf.*, 2015, pp. 31.1–31.12.
[46] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *Proc. Int. Conf. Learn. Represent.*, 2016.
[47] M. Pavlo, T. Stephen, K. Tero, A. Timo, and K. Jan, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Represent.*, 2017.
[48] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, and Y. Yang, "Asymptotic soft filter pruning for deep convolutional neural networks," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3594–3604, 2020.
[49] X. Wang, Z. Zheng, Y. He, F. Yan, Z. Zeng, and Y. Yang, "Progressive local filter pruning for image retrieval acceleration," *arXiv preprint arXiv:2001.08878*, 2020.
[50] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1379–1387.
[51] F. Tung and G. Mori, "Deep neural network compression by in-parallel pruning-quantization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 568–579, 2020.
[52] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, and Y. Yang, "Learning filter pruning criteria for deep convolutional neural networks acceleration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2009–2018.
[53] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct 2017.
[54] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct 2017.
[55] X. Dong, J. Huang, Y. Yang, and S. Yan, "More is less: A more complicated network with less inference complexity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1895–1903.
[56] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 883–894.
[57] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015.
[58] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 17–35.
[59] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.
[60] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3774–3782.
[61] L. Wei, S. Zhang, W. Gao, and Q. Tian, "Person transfer gan to bridge domain gap for person re-identification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 79–88.
[62] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, p. 91–99.
[63] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009.
[64] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: CNN architecture for weakly supervised place recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1437–1451, 2018.
[65] Z. Zheng and Y. Yang, "Parameter-efficient person re-identification in the 3d space," *arXiv preprint arXiv:2006.04569*, 2020.

Xiaodong Wang received his B.E and M.E. degree from the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China in 2007 and 2010, respectively. He received his Ph.D. degree in Informatics from Chaoyang University of Technology, Taiwan, in 2019. He is currently an associate professor with the College of Computer and Information Engineering, Xiamen University of Technology. His research interests cover pattern recognition, image processing, and deep learning.



Zhedong Zheng received the Ph.D. degree from the University of Technology Sydney, Australia, in 2021 and the B.S. degree from Fudan University, China, in 2016. He is currently a postdoctoral research fellow at NExT++, School of Computing, National University of Singapore. He was an intern at Nvidia Research (2018) and Baidu Research (2020). His research interests include robust learning for image retrieval, generative learning for data augmentation, and unsupervised domain adaptation.



Yang He received the B.S. degree and MSc from the University of Science and Technology of China, Hefei, China, in 2014 and 2017, respectively. He is expected to obtain Ph.D. degree at the University of Technology Sydney in 2021. He is currently a scientist at the Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR). His research interests include deep learning, computer vision, and filter pruning.

Fei Yan received her B.E. degree from the College of Technology from Hunan Normal University, Changsha, China, in 2007 and received her M.E. degree from the College of Computer Science and Electronic Engineering from Hunan University, China, in 2010. She is currently a lab master in the College of Computer and Information Engineering, Xiamen University of Technology. Her research interests include pattern recognition and data hiding.

Zhiqiang Zeng received the M.Sc. and Ph.D. degrees in Computer Science from the Xi'an Jiaotong University and Zhejiang University, China, in 2004 and 2007, respectively. He holds a Bachelor's degree in Automation from Sichuan University, Chengdu, China, in 1994. In 2008, he joined the Computer Science Department of the Xiamen University of Technology as a research associate. His interests include pattern recognition and machine learning, in particular, support vector machines and general kernel methods.

Yi Yang received the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2010. He is currently a professor with University of Technology Sydney, Australia. He was a Post-Doctoral Research with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. His current research interest includes machine learning and its applications to multimedia content analysis and computer vision, such as multimedia indexing and retrieval, video analysis, and video semantics understanding.