electronics

MDPI

Article

Pruning Convolutional Neural Networks with an Attention Mechanism for Remote Sensing Image Classification

Shuo Zhang [1], Gengshen Wu [1], Junhua Gu [2,*] and Jungong Han [3]

[1] School of Computing and Communications, Lancaster University, Lancaster LA1 4WA, UK; s.zhang28@lancaster.ac.uk (S.Z.); gengshen.wu@lancaster.ac.uk (G.W.)
[2] School of Artificial Intelligence, Key Laboratory of Big Data Computing, Hebei University of Technology, Tianjin 300401, China; jhgu@hebut.edu.cn
[3] Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3FL, UK; jungonghan77@gmail.com
* Correspondence: jhgu@hebut.edu.cn

Received: 8 June 2020; Accepted: 24 July 2020; Published: 27 July 2020

Abstract: Despite the great success of Convolutional Neural Networks (CNNs) in various visual recognition tasks, the high computational and storage costs of such deep networks impede their deployments in real-time remote sensing tasks. To this end, considerable attention has been given to the filter pruning techniques, which enable slimming deep networks with acceptable performance drops and thus implementing them on the remote sensing devices. In this paper, we propose a new scheme, termed Pruning Filter with Attention Mechanism (PFAM), to compress and accelerate traditional CNNs. In particular, a novel correlation-based filter pruning criterion, which explores the long-range dependencies among filters via an attention module, is employed to select the to-be-pruned filters. Distinct from previous methods, the less correlated filters are first pruned after the pruning stage in the current training epoch, and they are reconstructed and updated during the next training epoch. Doing so allows manipulating input data with the maximum information preserved when executing the original training strategy such that the compressed network model can be obtained without the need for the pretrained model. The proposed method is evaluated on three public remote sensing image datasets, and the experimental results demonstrate its superiority, compared to state-of-the-art baselines. Specifically, PFAM achieves a 0.67% accuracy improvement with a 40% model-size reduction on the Aerial Image Dataset (AID) dataset, which is impressive.

Keywords: deep feature learning; filter pruning; remote sensing imagery; self-attention

1. Introduction

With mass applications of remote sensing equipment, how to perform efficient remote sensing image scene classification is becoming a significant, yet challenging research problem. In recent years, Convolutional Neural Networks (CNNs) have shown appealing performance on various computer vision tasks, which have been applied broadly in remote sensing image scene classification [1–6]. However, large amounts of computational resources from the high-performance GPUs are required to run the complicated CNNs. Moreover, traditional CNNs usually contain many network parameters, even millions, which indicates that remote sensing equipment suffers from high demands for large memory storage space. Most of the remote sensing devices prefer to conduct real-time data collection and analysis on an aircraft, rather than making a decision in a workshop. In this context, the computing power in existing remote sensing devices is quite limited due to the embedded low-level CPUs and GPUs, thus making it infeasible to deploy the deep learning techniques on those machines directly.

Hence, the above research issues motivated us to build a lightweight CNN model for remote sensing image classification, which significantly reduces the required hardware resources, e.g., memory costs and FLOPs (Floating-Point Operations), such that remote sensing images can be processed in the smart sensor without the need to send those images back to the data center for further processing.

Many approaches have been proposed for deep network compression and acceleration, where three mainstream categories are discussed: network binarization, compact block, and filter pruning. In network binarization, the deep networks [7,8] are compressed by using the binarized weights and activations to reduce the memory space while achieving relatively good performance, which refuted the conclusion of the previous work [9] that a model that is highly binarized might achieve very bad performance at an early time. After that, BinaryConnect [10] constrained the full precision weights of the neural network filters to the discrete values (+1 or -1) during propagations. As an extension work of BinaryConnect, BinaryNet [11] binarized both weights and activations. However, the arbitrary binarization significantly weakens the feature representation ability of the deep networks, thus leading to less favorable performance. To reduce the quantization errors caused by the binarization, XNOR-Net [12] made use of a single scaling factor and binary filters, while Modulated Convolutional Networks (MCN) [13] combined a real-valued matrix with binary filters to reconstruct unbinarized filters. Both methods make the models compressed, but still maintain high accuracy when conducting image classification tasks. Beyond this, many works are devoted to employing compact blocks (e.g., convolutional filter with small receptive fields) in the deep network structures to reduce computational costs while avoiding large quantization errors that damage the original network's expressiveness. For instance, Network in Network [14] used 1×1 convolution kernels to reduce the network parameters. ResNet [15] reduced a large number of network parameters by involving the residue modules. Moreover, ShuffleNet [16] proposed pointwise group convolution and channel shuffle to construct an efficient network structure that can run on mobile devices with limited computing resources. MobileNet [17] used depth-wise convolution and point-wise convolution instead of normal convolution to build light, deep neural networks. However, a small receptive field focused on the local details excessively without considering the global information, thus compromising the classification performance.

Apart from these above methods, some prior works adopted filter pruning (i.e., channel pruning or network slimming), which is also the focus of this paper, to compress the deep network. The core idea behind filter pruning is that the small-valued (i.e., unimportant) activation and connection can be pruned during the iterative training processes to obtain more compact and efficient models [18–24]. For example, in [24], they compressed the deep neural networks by simply discarding unnecessary connections that were less than the default threshold. However, it was still required to retrain the sparse network model to compensate for the accuracy decline caused by the pruning. Instead of merely discarding the network parameters, recent works [25–31] compressed the complicated deep models via pruning the less important filters, thus reducing the computation costs dramatically due to fewer feature maps involved in the subsequent calculations. Subsequently, the work in [29] proposed a filter pruning method to remove the filters with the smallest absolute values and their corresponding feature maps in the next convolutional process for a compact network model. Nevertheless, most of the previous filter pruning works compressed the deep CNNs based on the pretrained models. They removed those filters permanently and then fine-tuned the pruned models to recover the huge accuracy drop, which was computationally expensive and inefficient. He et al. [32] adopted a soft pruning method to dynamically remove redundant filters, where the significance of a filter was evaluated by calculating the norm value of each filter and making a comparison among them. However, they only focused on the importance of individual filters without taking the correlation/dependency among filters into consideration, which made the filter selection less discriminative. Moreover, they evaluated the filter significance based on shallow strategies like thresholding, absolute, or norm values, which affected the classification performance considerably because of mistaken pruning.

In this paper, we propose a novel filter pruning method, termed Pruning Filter with Attention Mechanism (PFAM), which integrates the attention module with softly pruning the less correlated filters to obtain compact deep network model for the remote sensing image classification. To be specific, by deploying the attention module, the target filters that have smaller correlation values that others are pruned in the current pruning stage. In the next training epoch, the pruned filters in the previous pruning stage are recovered and further updated to avoid the accuracy loss because of the pruning process. By conducting such iterative training steps, the compact deep network model can be finally obtained with satisfactory performance. The contributions of our work are illustrated in the following three aspects:

(1) A novel deep network compression method termed Pruning Filter with Attention Mechanism (PFAM) is proposed for efficient remote sensing image classification. The compact network model is obtained by integrating the attention-based filter pruning strategy into a unified end-to-end training process.
(2) A novel correlation-based filter selection criterion is proposed in the filter pruning, where the correlation value of each filter is calculated through the attention module, and then, the less correlated filters are pruned to reduce the network complexity. By using the proposed attention module, it models the correlation among filters efficiently via exploring their long-range dependencies, which is more likely to make wise decisions in selecting the to-be-pruned filters without compromising the network performance.
(3) Extensive experiments on three remote sensing image datasets demonstrate that our proposed PFAM outperforms the state-of-the-art algorithms significantly.

The rest of this paper is organized as follows. Section 2 discusses previous representative approaches in the research field of network compression. Then, our proposed filter pruning method is detailed in Section 3. Section 4 shows the experimental results on multiple remote sensing datasets. Finally, the discussion and conclusions of this research are drawn in Sections 5 and 6, respectively.

2. Related Works

In network pruning, the unimportant activations and connections could be removed to obtain more compact models. For instance, Han et al. [23] discarded the unnecessary connections that were less than a default threshold value, and then, they retrained such a sparse model to improve its accuracy. Deep Compression [33] achieved great energy and memory savings by extending their previous work [23]. In particular, they combined the connection pruning with the quantization techniques, where several remaining important connections could share the same weights. Then, Huffman encoding technology was utilized to make further compression. Although Deep Compression obtained a high compression ratio on the CNN models by removing unimportant parameters, the parameter importance varied dramatically if changing the network structure. This implies that this sort of hard pruning method will suffer if the important connections are removed incorrectly during long-time training.

To enhance it, dynamic network surgery was proposed in [34], where the pruning and splicing methods are jointly combined to recover some important connections that were removed incorrectly before. They are recovered as significant connections again during the next training period, thus reducing the possibility of misclassification to the maximum extent. More importantly, the pruning and training processes are synchronized seamlessly such that the problems of long retraining time and incorrect pruning can be solved effectively. Alternatively, unlike conventional weight or connection pruning, Li et al. [29] proposed a filter pruning method named Pruning Filters for Efficient ConvNets (PFEC) to delete some filters and their related feature maps in the next convolutional process to reduce the computation costs. In particular, the absolute values of the filters are calculated and ordered from smallest to largest, then the filters with the smallest values are removed to obtain a highly compressed network model. Liu et al. [30] proposed a pruning method termed network slimming by enforcing

channel-level sparsity in the network. To be specific, they evaluated the scores of the input channels by calculating the corresponding weights of the batch normalization layers. These scores are compared with a pre-set threshold value, and those filters with scores lower than the threshold are removed.

Unlike previous research works that utilized hard filter pruning technology, the soft pruning method was proposed in Soft Filter Pruning (SFP) [32] and Filter Pruning via Geometric Median (FPGM) [35], where the pruned convolution filters are recovered and involved in the next training iteration, rather than being deleted once and gone permanently. By doing so, there is no need to go through the fine-tuning process on the pretrained model to compensate for the accuracy drop after the pruning. One more advantage of such a soft pruning scheme is that it saves much training time by conducting the pruning process immediately after the end of each training process. Our method generally follows the soft pruning method as [32,35]; however, it utilizes an advanced pruning filter selection strategy based on the correlation between filters from the involved attention module, thus obtaining a more robust and compact network model.

3. Methodology

3.1. Motivation

As discussed in the previous sections, earlier filter pruning works [29–31] generally compressed the deep CNNs in a hard manner. To be specific, these algorithms firstly prune the unimportant filters in every single convolutional layer from a pretrained model directly, where the significance evaluation of the single filter is often inexact because of ambiguous calculations in one certain layer. Then, the pruned model needs to go through the fine-tuning stage to compensate for the performance degradation caused by the pruning. However, once the filters are selected, these to-be-pruned filters are abandoned permanently during the pruning process and never recovered again in the following fine-tuning stage. Although the model is dramatically reduced in size due to the removal of filters, such a hard pruning method is more likely to yield unsatisfactory performance due to the shrinkage of model capacity. Besides, it is worth mentioning that these methods may have expensive computational resources costs to get the final pruned model fine-tuned on the training data.

In contrast to the hard pruning methods, Reference [32] proposed to dynamically remove the filters in a soft manner by calculating the norm value of each filter independently. Despite its advances, the underlying mechanism that focuses on the individual effect of the single filter in one convolutional layer without considering the global relationship among them is deemed suboptimal. In other words, the least important filter that is determined individually is not the least important one if a global view considering all the filters is adopted. From another perspective, it might be useful if we investigate the long-range dependency of filters and involve these dedicated dependency/relationships among filters in the to-be-pruned filter selection. In traditional CNNs, however, the long-range dependency among layers can only be obtained by repeatedly backpropagating through the stacking convolutional layers. That is to say, the current deep networks are usually inefficient at capturing such long-range dependencies, and it is difficult to operate locally when the information needs to be passed back and forth between relatively remote locations [36]. To tackle the above problems, the attention mechanism was applied in many applications to obtain better network performance. For example, Mnih et al. [37] utilized an attention module in the network training that pays more attention to the local areas with high-correlated weights from the whole target areas, which simulates the human's visual attention behavior when observing images. Following previous works [38–40], they adopted a self-attention module that calculates the response at a specific position as a weighted sum of the features at all positions [41–44]. In this case, the weights or attention vectors were calculated with low computational costs, while a good balance between the long-range dependency modeling ability and computational efficiency was achieved.

Inspired by previous works, we propose a new filter pruning method termed Filter Pruning with Attention Mechanism (PFAM), which integrates an attention mechanism into the filter pruning. In

particular, the attention module is used to count and collect the correlation value of each filter and then select the less correlated filters based on those values; hence, the overall correlations among all filters in one convolutional layer are considered to obtain the minimal accuracy loss globally. In the pruning stage, the values of those selected as less correlated are set to zero, which means these filters are removed. In the next training epoch, the values of pruned filters are recovered from zero to non-zero and updated by the upcoming forward-backward operations. By doing so, the training data can be processed by the original training strategy without compromising the performance, while the compressed network model can be obtained in the end with no need for the pretrained model. The general process of the proposed filter pruning approach is shown in Figure 1.

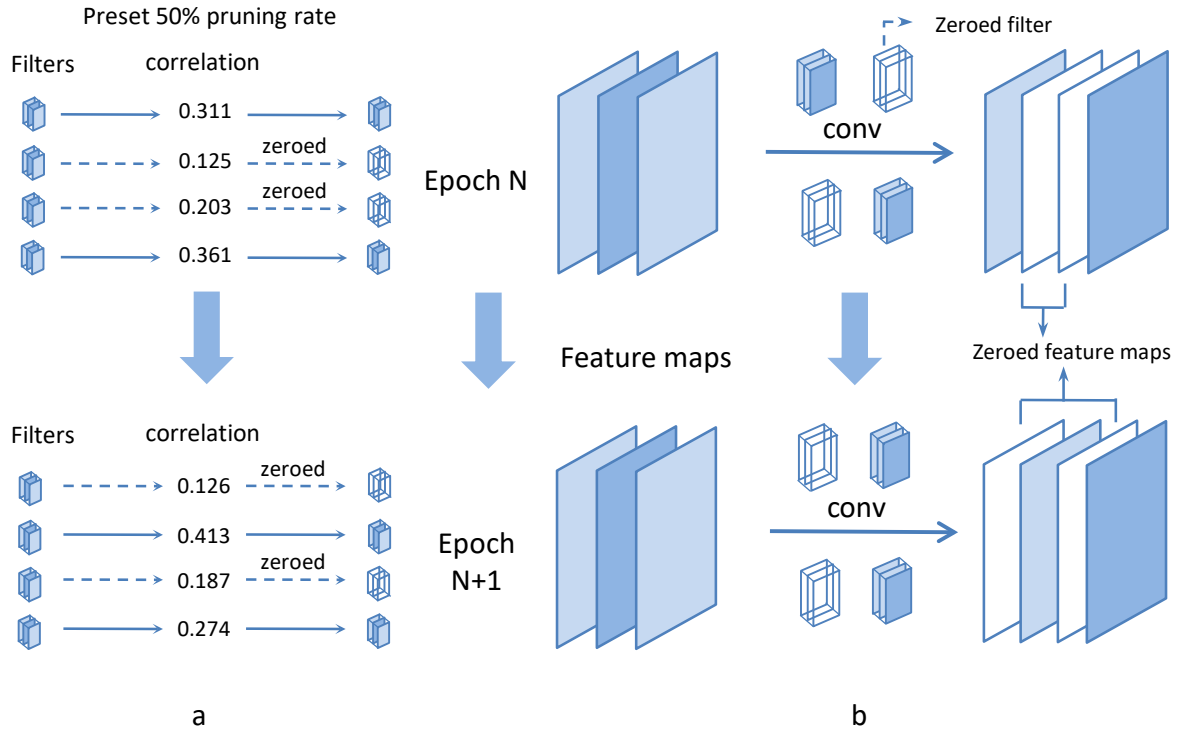

Figure 1. The overview of Filter Pruning with Attention Mechanism (PFAM). (**a**) Filter selection based on the correlation criterion. Note that the sum of the correlation values is 1 due to the output of softmax. The filters with less correlation values will be pruned based on the pre-set pruning ratio. (**b**) An example of filter pruning in convolutional layers. The pruned filters are allowed to be updated to non-zero during each training epoch prior to the next pruning stage to maintain the model capacity.

3.2. Filter Selection with Attention-Based Correlation

Figure 2 illustrates the working flow of the proposed attention module in calculating the correlation values between filters in one convolutional layer, which shares a similar structure as many computer vision tasks. In particular, the filters are treated like the feature maps in the sense that flatting one filter into a one-dimensional vector is similar to flatting one feature map into a one-dimensional vector, but with different vector length. In our paper, instead of finding the most attractive feature maps in previous works, we aim at selecting the least attractive filters from a certain number of candidates and pruning those less correlated filters to create a compact model.

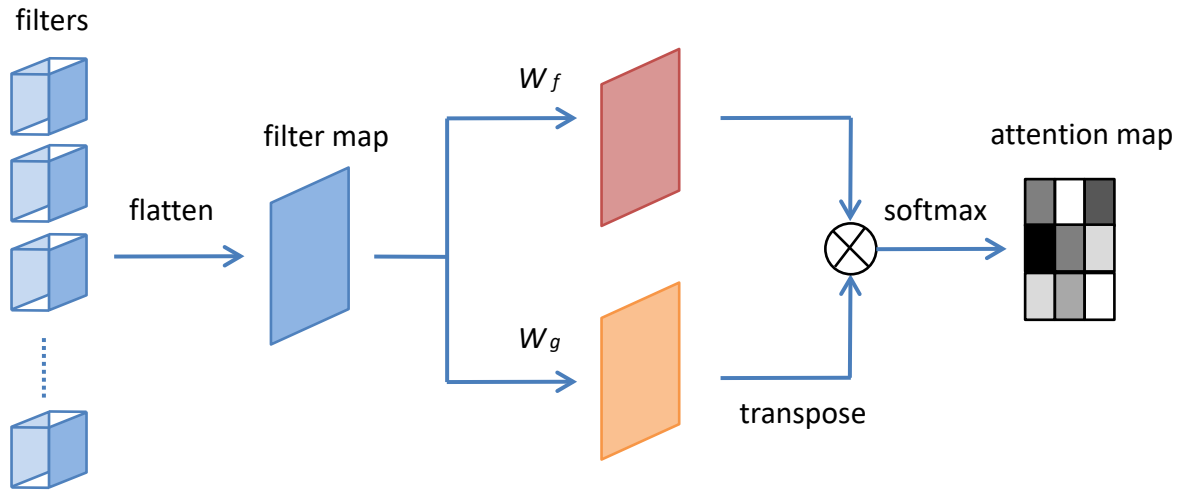

Figure 2. The proposed attention module for PFAM. ⊗ denotes the matrix multiplication. The softmax operation is performed on each row.

Without loss of generality, we first define some mathematical symbols following [32] to ease the explanation. To be specific, the dimension of filter tensor with a $k \times k$ filter size in the i-th convolutional layer is defined as $W^{(i)} \in R^{C_{i+1} \times C_i \times k \times k}$, where $1 \leq i \leq L$. C_{i+1} and C_i mean the number of output and input channels separately for the i-th convolutional layer and L is the number of layers. Then, all the filters are flattened in the i-th layer as $W^{(i)} \in R^{C_{i+1} \times V}$, where V denotes the shape of each filter in the i-th convolutional layer and equals $C_i \times k \times k$. The filters in the i-th layer are first transformed into two weight spaces $F(w) = W_f^{(i)}$ and $G(w) = W_g^{(i)}$ to calculate the attention map, which can be formulated as below:

$$\theta_{j,k} = \frac{e^{s_{jk}}}{\sum_{k=1}^{M} e^{s_{jk}}}, \quad \text{where} \quad s_{jk} = F(w_j)G(w_k)^T. \tag{1}$$

$F(w_j)$ and $G(w_k)$ represent the values of the j-th filter in $W_f^{(i)}$ and the k-th filter in $W_g^{(i)}$ weight spaces, respectively. $\theta_{j,k}$ represents the correlative extent between the j-th and k-th filter. M is the number of filters in the i-th convolutional layer. Therefore, the output of the attention value is $\partial = (\partial_1, \partial_2, \ldots \partial_k, \ldots \partial_M) \in R^{1 \times M}$, where:

$$\partial_k = \sum_{j=1}^{M} \theta_{j,k}. \tag{2}$$

As discussed above, the attention module is used to evaluate the correlation of each filter based on Equation (2) in the pruning stage. The filters with smaller correlation values can be pruned, because it turns out that they have less impact on the network performance, as opposed to other highly correlated filters.

3.3. Filter Pruning and Reconstruction

In the pruning stage, all the candidate convolution layers with the same pruning rate $P_i = P$ are pruned at the same time, which saves a large amount of computations, compared to the hard pruning methods. In particular, a pruning rate P_i is set to select a total number of $C_{i+1}P_i$ less correlated filters in the i-th convolution layer [32,35]. After pruning the filters in each convolution layer, existing methods always require extra training to converge the network [21,22]. During the training process, these selected filters are first zeroed out, which means they have no contribution to the network output in the current pruning stage.

In most filter pruning methods, however, the pruned filters and their associated feature maps are removed permanently during the pruning process, which could affect the performance significantly. To deal with this problem, these pruning methods usually are conducted based on the pretrained

model, and they also need to spend extra fine-tuning time for accuracy compensation. To get rid of the heavy dependences on the pretrained model and the time-consuming fine-tuning process, we follow the same reconstruction strategy as [27] at this stage, where the pruned filters in the previous pruning process are reconstructed during one epoch of retraining. To be specific, these pruned filter values are updated from zero to non-zero after the backpropagation [32,35]. By doing so, the pruned model still has the same capacity as the original model during the training process. More importantly, each of those filters can still contribute to the final prediction. As a result, we can train our network from either scratch or the pretrained model and obtain competitive results even without the need for the fine-tuning stage.

3.4. Compact Network Creation

Figure 3 shows the flowchart of the proposed PFAM, where the iterative training is repeatedly performed until the accuracy loss becomes converged after several training epochs. We can get a sparse model with multiple zeroed filters when the model becomes converged. These selected filters will remain unchanged because the iteration has completed. Since each filter corresponds to one feature map, these feature maps corresponding to those zeroed filters will always be zero during the inference procedure. Removal of these zeroed filters, as well as their related feature maps will not have any effect. After the iteration of the previous steps, the compact model is finally created. The whole process is briefly summarized in Algorithm 1.

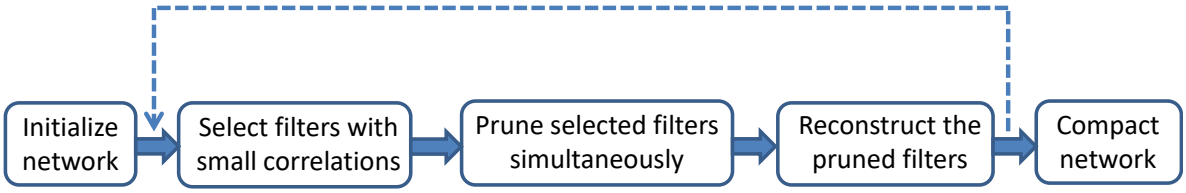

Figure 3. The flowchart of the proposed filter pruning. The dotted line means the iterative training scheme.

Algorithm 1 Algorithm description of PFAM.

Input: Training data X; Pruning rate P_i; Training epoch number $epoch$;
Output: The compact model W^* from W;
1: Randomly initialize the network parameters $W = \{W^{(i)}, 0 \leq i \leq L\}$;
2: **for** $epoch = 1; epoch \leq epoch_{max}; epoch++$ **do**
3: Update the model parameter W based on data X;
4: **for** $i = 1; i \leq L; i++$ **do**
5: Find $N_{i+1}P_i$ filters that satisfy Equation (2);
6: Zeroize selected filters;
7: **end for**
8: **end for**
9: **return** the compact model W^* from W.

4. Experiments and Analysis

In this section, we provide extensive experimental results and analysis to illustrate the system performance of our algorithm on three popular remote sensing benchmarks: Remote Sensing Image

Scene Classification, created by Northwestern Polytechnical University (NWPU-RESISC45) [45], Aerial Image Dataset (AID) [46], and RSSCN7 [47].

4.1. Benchmark Datasets

4.1.1. NWPU-RESISC45 Dataset

NWPU-RESISC45 (http://www.escience.cn/people/JunweiHan/NWPU-RESISC45.html) [45] is a popular public dataset for remote sensing image scene classification, which was extracted from Google Earth by experts at Northwestern Polytechnical University (NWPU). This dataset is made up of a total of 31,500 images, which are categorized into 45 scene classes as shown in Figure 4. Each class includes 700 images with a size of 256 × 256 pixels in the RGB color space.

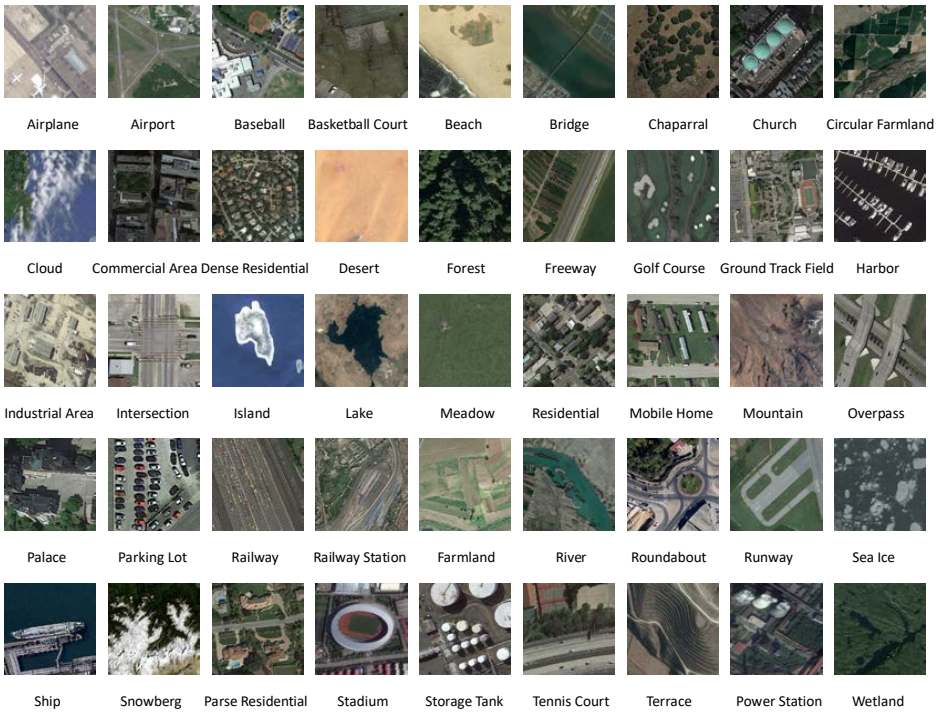

Figure 4. Example images of the Northwestern Polytechnical University (NWPU)-RESISC45 dataset.

4.1.2. RSSCN Dataset

The RSSCN7 (https://github.com/palewitout/RSSCN7) [47] dataset was released in 2015 by Wuhan University, China, which contains 2800 remote sensing images in total from seven typical scene categories: grasslands, forests, farmland, car parks, residential areas, industrial areas, and rivers and lakes, as shown in Figure 5. For each category, there are 400 images with the size 400 × 400 collected from Google Earth, and these pictures are sampled at four different scales. It is also a challenging dataset because the remote images were taken in different seasons and weather conditions with various sampling scales.

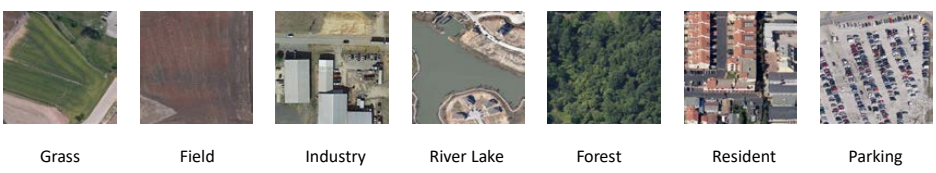

Figure 5. Example images of the RSSCN7dataset.

4.1.3. AID Dataset

AID (https://captain-whu.github.io/AID/) [46] is a large-scale aerial image dataset, which was selected from Google Earth imagery. This dataset contains in total 10,000 images with a fixed size of 600 × 600 pixels within 30 classes, as shown in Figure 6. Compared to other classic datasets, the number of images for each category is not equal, and different scene types range from 220 to 420, which makes it more challenging in the image classification. Although images in this dataset were acquired at different times with different imaging conditions, some classes are quite similar and therefore make the differences between classes smaller.

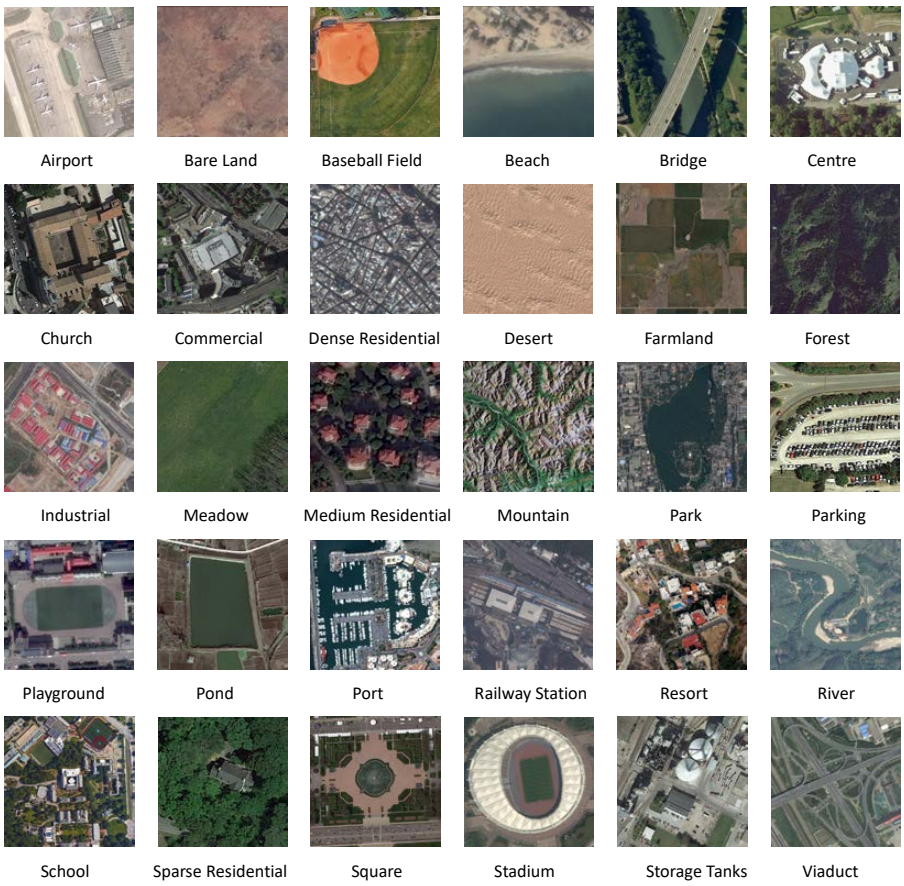

Figure 6. Example images of the AID dataset.

4.2. Experimental Settings

4.2.1. Deep Architecture

In the experiment, we chose a popular deep model, ResNet [15], as the backbone network because of the more complex and less redundant structure of ResNet compared to VGG models [48]. We also conducted experiments on the VGG models to further consolidate the superiority of our method.

4.2.2. Implementation Details

In this work, the PyTorch framework was used to implement the proposed method. In the dataset preprocessing, we followed the PyTorch guidance [49] to perform data argumentation. To be specific, all images in the AID dataset were resized to 256 × 256 pixels from the original 600 × 600 pixels, which follows the same image size in the NWPU-RESISC45 dataset. Regarding the RSSCN7 dataset, we resized all images from 400 × 400 pixels to 256 × 256 pixels for the same reason.

We applied the same training ratio (80%) to make fair comparisons of the experiments, and all three datasets were randomly divided into training and testing sets based on the pre-set ratios to calculate the overall classification accuracy. All the experiments were conducted three times to get fair and reliable results. The hardware configurations were the Linux Ubuntu 14.04 operating system with i7-5960X CPU, 64GB RAMs, and one NVIDIA GTX1080Ti GPU.

In the pruning stage, only one hyper-parameter $P_i = P$ was set to prune all the convolutional layers after finishing every training epoch. The pruning rate makes a trade-off between the compression and the accuracy throughout the pruning process [32]. Notably, the projection shortcuts do not need to be pruned for the compression because of the limited contribution to the overall costs when using ResNet [15] in the evaluation.

In the network training, we used Stochastic Gradient Descent (SGD) as the optimizer with the weight decay and momentum as 0.0005 and 0.9, respectively. The learning rates were set separately for two training phases: 0.01 in the first 50 epochs and 0.002 for the last 50 epochs. The training processes for all network models were terminated when the training losses converged. Moreover, the batch-size was set to 64 for the NWPU-RESISC45 dataset and 32 for the AID dataset and the RSSCN7 dataset to balance the requirements of the computer memory and the image number contained in the training and test sets.

We tested our method on VGG-16 [48] and ResNet-18, -34, -50, and -101 [15], respectively. In order to make fair comparisons, we used the pruning rate of 40% for the same model, while the pruning methods for both the scratch and the pretrained model during the same training epochs were also tested and analyzed. There is no need to conduct the fine-tuning after the scratch-wise model training in our method when compared to many previous approaches with the hard pruning manner.

4.2.3. Evaluation Metrics

Four different evaluation metrics, Accuracy (Acc.), Accuracy Drop (Acc. Drop), FLOPs, and pruning ratio (Pruning), were used in the experiments, where the italics denote the symbols in the tables. Almost all model compression methods compare the performance using the pruning ratio vs. model accuracy. We tweaked the global pruning rate (normally a parameter) of each algorithm involved in the comparison, such that all the models pruned by different methods were more or less identical, e.g., 40%, in terms of the size of the model. Then, we tested the pruned models on remote sensing datasets and calculated the top 1 accuracy, given the image classification task. In particular, Acc. measures the classification accuracy obtained by the specific method, which is expressed as the average and the standard deviation of accuracy after running the experiments three times. Acc. Drop is computed by the accuracy of the pruned model minus that of the baseline model, where a negative number implies that the pruned model achieves even higher accuracy than the baseline model. The smaller Acc. Drop is, the better performance. The pruning rate denotes the real compression ratio of the network models. The larger the pruning rate is, the more compact model. FLOPs denotes the total number of floating-point operations, which is used as a reference metric in evaluating the pruning method.

Followed by [50] (Equation (3)), the top k prediction was made up of any set of labels related to the k biggest scores and $k \in \{1, \ldots, n-1\}$. Concretely, we assumed that $Y^{(k)} \in \{1, \ldots, n\}$ was the set of k-tuples with k distinct elements in the output space. $\overline{y}$ means the set of k-tuples of the ground truth label. $s \in R^n$ and $s_{[k]}$ are defined as a vector of scores per label and the k-th biggest element of s, respectively. Next, from Equation (4), we sum up the number of $P_k(s)$ by Equation (3) and then divide the batch size to get the top k accuracy for one batch. Finally, we get the final average top k accuracy for all images by averaging them from all batches.

$$P_k(s) \in \left\{ \overline{y} \in Y^{(k)} : \forall i \in \{1, \ldots, k\}, s_{\overline{y}i} \geq s_{[k]} \right\} \tag{3}$$

$$Top\,(k)\ accuracy = Avg\left(\frac{sum\,(P_k(s))}{batch\ size}\right) \tag{4}$$

4.3. Comparisons with the State-of-the-Art

4.3.1. Results on NWPU-RESISC45

Table 1 shows the results on the NWPU-RESISC45 dataset when applying ResNet-18, -34, -50, and -101 respectively for remote sensing image classification, where our proposed method achieved superior performance compared to the other state-of-the-art filter pruning methods. In particular, PFEC [29] used the hard pruning method in pruning ResNet-18 and -34 with accuracies of 89.78% and 90.75% independently, whereas the figures from SFP [32] were 0.29% and 0.42% lower than them, but FPGM [35] obtained better results than them. Although ThiNet [51] obtained the second highest accuracy in pruning ResNet-18 and -50 with 92.47% and 92.60%, which outperformed the other two soft pruning methods, our method still obtained the highest accuracy among them, where PFAM achieved the highest accuracies (92.56% and 92.87%) with compression ratios of 42.3% and 40.6%. Moreover, PFAM achieved the lowest accuracy drops of 0.27% on ResNet-34 with the second largest compression ratio. FPGM [35] achieved the best accuracy (92.64%) among the five methods when pruning ResNet-101, which was slightly better than our PFAM (92.52%). It is noted that, in this case, the pruning ratio of our PFAM (39.6%) was larger than that of FPGM [35] (38.1%). To sum up the above experimental results, our PFAM obtained a highly compressed network model with competitive performance in most cases, given the NWPU-RESISC45 dataset.

Table 1. Comparison of pruning ResNet on the NWPU-RESISC45 dataset. Acc, Accuracy.

Model	Method	Acc. (%)	Acc. Drop (%)	FLOPs	Pruning (%)
18	Baseline	93.40 (±0.32)	0	6.05×10^8	0
	PFEC [29]	89.78 (±0.15)	3.62	3.78×10^8	37.5
	ThiNet [51]	92.47 (±0.11)	0.93	3.79×10^8	37.4
	SFP [32]	89.49 (±0.27)	3.91	3.41×10^8	43.6
	FPGM [35]	92.28 (±0.18)	1.12	3.68×10^8	39.2
	PFAM (ours)	92.56 (±0.23)	0.84	3.49×10^8	42.3
34	Baseline	93.73 (±0.11)	0	1.22×10^9	0
	PFEC [29]	90.75 (±0.24)	2.98	7.47×10^8	38.8
	ThiNet [51]	93.08 (±0.22)	0.65	7.46×10^8	38.9
	SFP [32]	90.33 (±0.17)	3.40	6.81×10^8	44.2
	FPGM [35]	93.24 (±0.08)	0.49	7.39×10^8	39.4
	PFAM (ours)	93.46 (±0.14)	0.27	7.25×10^8	40.6
50	Baseline	93.37 (±0.21)	0	1.36×10^9	0
	PFEC [29]	90.65 (±0.33)	2.72	8.50×10^8	37.5
	ThiNet [51]	92.60 (±0.17)	0.77	8.67×10^8	36.3
	SFP [32]	91.35 (±0.23)	2.02	8.49×10^8	37.6
	FPGM [35]	92.57 (±0.12)	0.80	8.69×10^8	36.1
	PFAM (ours)	92.87 (±0.25)	0.50	8.44×10^8	37.9
101	Baseline	93.17 (±0.13)	0	2.60×10^9	0
	PFEC [29]	88.49 (±0.35)	4.68	1.43×10^9	45.0
	ThiNet [51]	92.22 (±0.21)	0.95	1.59×10^9	38.8
	SFP [32]	91.90 (±0.28)	1.27	1.58×10^9	39.2
	FPGM [35]	92.64 (±0.06)	0.53	1.61×10^9	38.1
	PFAM (ours)	92.52 (±0.14)	0.65	1.57×10^9	39.6

4.3.2. ResNet on the AID Dataset

Similarly, our proposed method consistently outperformed the other state-of-the-art methods on the AID dataset, as shown in Table 2. Although SFP [32] achieved the largest pruning ratios on

ResNet-18, -34, -50, and -101, much higher accuracies were achieved by our proposed PFAM, where the gaps were 4.01%, 4.26%, 3%, and 3.17%, respectively. FPGM [35] obtained the second-best results in these experiments, and the accuracy was at least 0.19% (on ResNet-18: 85.08% vs. 85.27%) lower than that of PFAM. Although hard pruning methods such as PFEC [29] and ThiNet [51] obtained higher accuracy than SFP [32], their performance was still worse than ours. It is worth mentioning that our filter pruning method even outperformed the original model when pruning ResNet-101 in terms of accuracy, which was 84.17% and 84.10%, respectively. That indicates the powerful ability of PFAM in producing a highly compressed model while maintaining competitive performance.

Table 2. Comparison of pruning ResNet on the AID dataset.

Model	Method	Acc. (%)	Acc. Drop (%)	FLOPs	Pruning (%)
18	Baseline	85.56 (±0.12)	0	6.05×10^8	0
	PFEC [29]	81.95 (±0.23)	3.61	3.79×10^8	37.4
	ThiNet [51]	83.76 (±0.15)	1.80	3.79×10^8	37.4
	SFP [32]	81.26 (±0.27)	4.30	3.38×10^8	44.1
	FPGM [35]	85.08 (±0.13)	0.48	3.68×10^8	39.2
	PFAM (ours)	85.27 (±0.11)	0.29	3.60×10^8	40.5
34	Baseline	85.83 (±0.11)	0	1.22×10^9	0
	PFEC [29]	81.63 (±0.22)	4.20	7.47×10^8	38.8
	ThiNet [51]	83.34 (±0.15)	2.49	7.47×10^8	38.8
	SFP [32]	79.96 (±0.25)	5.87	6.78×10^8	44.4
	FPGM [35]	83.55 (±0.07)	2.28	7.39×10^8	39.4
	PFAM (ours)	84.22 (±0.16)	1.61	7.32×10^8	40.0
50	Baseline	84.86 (±0.27)	0	1.36×10^9	0
	PFEC [29]	81.60 (±0.19)	3.26	8.50×10^8	37.5
	ThiNet [51]	82.34 (±0.21)	2.52	8.67×10^8	36.3
	SFP [32]	81.43 (±0.33)	3.43	8.44×10^8	37.9
	FPGM [35]	83.76 (±0.15)	1.10	8.69×10^8	36.1
	PFAM (ours)	84.43 (±0.22)	0.43	8.54×10^8	37.2
101	Baseline	84.10 (±0.19)	0	2.60×10^9	0
	PFEC [29]	83.27 (±0.11)	0.83	1.69×10^9	35.0
	ThiNet [51]	81.25 (±0.18)	2.85	1.59×10^9	38.8
	SFP [32]	81.00 (±0.32)	3.10	1.57×10^9	39.6
	FPGM [35]	83.87 (±0.07)	0.23	1.61×10^9	38.1
	PFAM (ours)	84.17 (±0.13)	−0.07	1.59×10^9	38.8

To provide more comprehensive performance verification of our method, we also tested our method on pruning VGG-16 with training from scratch (Table 3) and ResNet based on the pretrained model (Table 4) on the AID dataset. The results in Table 3 describe the effectiveness of the soft pruning methods compared to the hard ones on the VGG-16 model, where PFAM still achieved the best accuracy (86.03%) under the same compression ratio among the four pruning methods. The performance degradation was inevitable for the hard filter pruning methods like PFEC [29], while the soft filter pruning methods enabled maintaining the strong network expressive ability after the reconstruction stage on the pruned filters to obtain better performance. In Table 4, the proposed PFAM achieves the best accuracies in pruning ResNet-18 (89.86%), -34 (89.77%), and -101 (90.57%), which demonstrate the effectiveness of our method in pruning ResNet with the pretrained model.

Table 3. Comparison of pruning VGG-16 on the AID dataset.

Method	Acc. (%)	Acc. Drop (%)	FLOPs	Pruning (%)
Baseline	86.55 (±0.11)	0	5.12×10^9	0
PFEC [29]	82.31 (±0.35)	4.24	3.26×10^9	36.3
SFP [32]	85.58 (±0.23)	0.97	3.08×10^9	39.8
FPGM [35]	85.88 (±0.11)	0.67	3.08×10^9	39.8
PFAM (ours)	86.03 (±0.14)	0.52	3.08×10^9	39.8

Table 4. Accuracies (%) of pruning ResNet based on the pretrained model on the AID dataset.

Method	ResNet-18 (%)	ResNet-34 (%)	ResNet-50 (%)	ResNet-101 (%)
Baseline	90.07 (±0.11)	90.53 (±0.28))	90.73 (±0.33)	90.84 (±0.27)
PFEC [29]	88.10 (±0.35)	88.34 (±0.25)	88.49 (±0.26)	89.76 (±0.15)
SFP [32]	89.03 (±0.23)	89.15 (±0.32)	89.57 (±0.44)	90.13 (±0.35)
FPGM [35]	89.44 (±0.11)	89.71 (±0.19)	90.53 (±0.21)	90.55 (±0.17)
PFAM (ours)	89.86 (±0.14)	89.77 (±0.23)	90.32 (±0.28)	90.57 (±0.29)

4.3.3. Results on RSSCN7

For the RSSCN7 dataset, we tested our PFAM on ResNet-18, -34, -50, and -101 and VGG-16 with a 40% pruning ratio to provide comprehensive insights into the performance. In Table 5, PFAM obtains the best performance compared to the other three methods on VGG-16, which is the same as Table 3. Although SFP [32], FPGM [35], and PFAM achieved the same compression ratio (39.8%), the accuracy of our method was much higher than their's. It can be seen that our proposed method also showed a high effect on selecting and pruning redundant filters on VGGNet.

Moreover, In Table 6, unlike previous results on the NWPU-RESISC45 and AID datasets, SFP [32] obtained the worst performances except on ResNet-101 in the experiments, even though it obtained the highest compression ratios. However, FPGM [35] and our proposed method generally achieved better experimental results than the norm-criterion methods like PFEC [29] and SFP [32]. The reason for the worse performance is that the norm-criterion methods only focus on pruning each individual filter without considering the global correlation among all filters. Therefore, this leads to the suboptimal performance. Compared to the methods selecting filters based on the norm-based criterion and the geometric median, the proposed attention module is utilized in PFAM to find the correlation between filters globally, which enables yielding superior performance because of the advanced pruning strategy.

Table 5. Comparison of pruning VGG-16 on the RSSCN7 dataset.

Method	Acc. (%)	Acc. Drop (%)	FLOPs	Pruning (%)
Baseline	90.00 (±0.17)	0	5.12×10^9	0
PFEC [29]	85.18 (±0.33)	4.82	3.26×10^9	36.3
SFP [32]	85.18 (±0.13)	4.82	3.08×10^9	39.8
FPGM [35]	87.68 (±0.21)	2.32	3.08×10^9	39.8
PFAM (ours)	88.04 (±0.22)	1.96	3.08×10^9	39.8

Table 6. Comparison of pruning ResNet on the RSSCN7 dataset.

Model	Method	Acc. (%)	Acc. Drop (%)	FLOPs	Pruning (%)
18	Baseline	86.43 (±0.23)	0	6.05×10^8	0
	PFEC [29]	83.57 (±0.13)	2.68	3.79×10^8	37.4
	ThiNet [51]	85.18 (±0.11)	1.25	3.77×10^8	37.7
	SFP [32]	82.86 (±0.16)	3.57	3.68×10^8	44.6
	FPGM [35]	85.00 (±0.22)	1.43	6.05×10^8	39.2
	PFAM (ours)	85.51 (±0.17)	0.92	3.63×10^8	40.0
34	Baseline	86.07 (±0.14)	0	1.22×10^9	0
	PFEC [29]	83.39 (±0.28)	2.68	7.47×10^8	38.8
	ThiNet [51]	85.63 (±0.23)	0.44	7.47×10^8	38.8
	SFP [32]	81.83 (±0.22)	4.24	6.66×10^8	45.4
	FPGM [35]	85.37 (±0.18)	0.70	7.39×10^8	39.4
	PFAM (ours)	85.71 (±0.25)	0.36	7.36×10^8	39.7
50	Baseline	85.28 (±0.21)	0	1.36×10^9	0
	PFEC [29]	80.71 (±0.17)	4.57	8.50×10^8	37.5
	ThiNet [51]	83.39 (±0.27)	1.89	8.66×10^8	36.3
	SFP [32]	80.36 (±0.31)	4.92	8.42×10^8	38.1
	FPGM [35]	84.64 (±0.25)	0.64	8.69×10^8	36.1
	PFAM (ours)	83.93 (±0.28)	1.35	8.66×10^8	36.3
101	Baseline	85.35 (±0.15)	0	2.60×10^9	0
	PFEC [29]	83.39 (±0.24)	1.96	1.69×10^9	35.0
	ThiNet [51]	82.53 (±0.33)	2.82	1.59×10^9	38.8
	SFP [32]	82.93 (±0.41)	2.42	1.56×10^9	40.0
	FPGM [35]	83.92 (±0.22)	1.43	1.61×10^9	38.1
	PFAM (ours)	83.93 (±0.26)	1.42	1.61×10^9	38.1

5. Discussion

In view of the comparison results of PEFC [29], ThiNet [51], SFP [32], and FPGM [35] on three test datasets, it is clear that our method generally yields competitive performance throughout extensive evaluations. We achieve the best performance in the vast majority of neural network architectures, which can show that the way we select the to-be-pruned filters is better than the other methods. The main advantage of the proposed method is that the overall correlation between filters in one convolutional layer is considered in the pruning, rather than only caring about the importance of each individual filter. In order to further investigate the performance difference of our method and the current leading methods under different pruning rates, we conducted extensive experiments regarding the performance variations by applying different pruning ratios on ResNet-18 and -34. The corresponding results are shown in Figure 7a,b. Since our method belongs to the soft pruning methods, we verified two other soft pruning methods for a fair comparison. Hence, three soft filter pruning methods, SFP [32], FPGM [35], and PFAM (ours), were tested on the NWPU-RESISC45 and RSSCN7 datasets, respectively. It is worth mentioning that the performance of SFP [32] became far worse than the other two methods with the increasing pruning ratio on both the RSSCN7 dataset and the NWPU-RESISC45 dataset.

For the RSSCN7 dataset, from Table 7, we can see that our method achieves the best performance at most pruning ratios (20%, 40%, and 60%) in pruning ResNet-18, while the accuracy of FPGM [35] is the highest at the extreme compression ratio of 80%, and a similar situation happened in the experiment results of pruning ResNet-34. However, on the relatively high pruning rate 80%, our approach does not perform as well as it did at the other pruning rate because too many filters are pruned in each convolution layer. Given a small or moderate pruning rate, such as 20% or 40%, our proposed method can accurately find the most redundant filters by calculating the correlation between the filters. That is why we consistently outperform the other algorithms across different datasets. However, with the increase of the pruning rate, our pruning method does not perform the

best, because too many filters are removed from each convolutional layer. When the pruning rate is as high as 80%, which means nearly 80% of filters in each convolution layer need to be pruned, this greatly damages the final prediction.

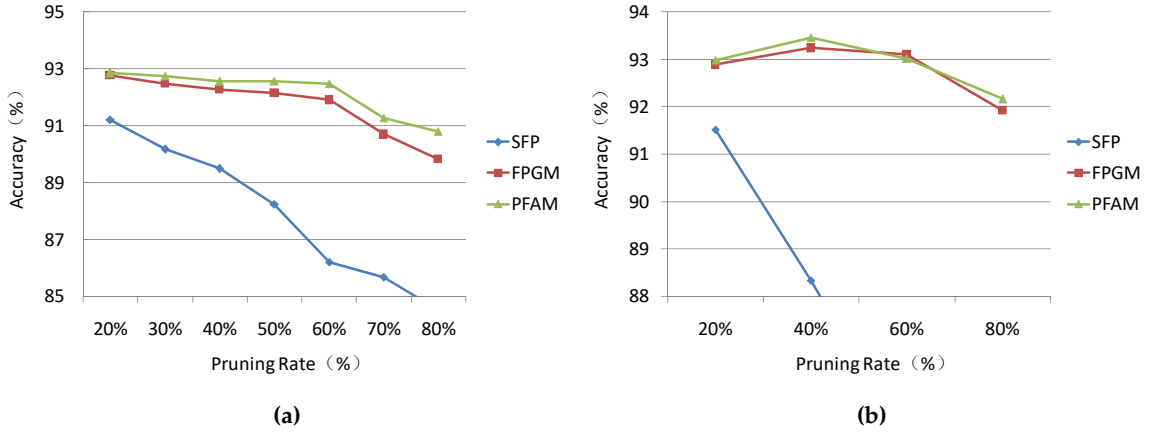

Figure 7. The comparisons among the three methods on pruning ResNet-18 (**a**) and -34 (**b**) with various pruning ratios.

Table 7. The comparisons of SFP, FPGM, and PFAM for pruning ResNet on the RSSCN7 dataset with pruning ratios of 0.2, 0.4, 0.6, and 0.8.

Compression Rate	Method	ResNet-18 (%)	ResNet-34 (%)
20%	SFP [32]	85.00 (±0.14)	83.03 (±0.22)
	FPGM [35]	85.79 (±0.21)	85.58 (±0.16)
	PFAM (ours)	86.00 (±0.15)	85.76 (±0.13)
40%	SFP [32]	82.86 (±0.16)	81.83 (±0.22)
	FPGM [35]	85.00 (±0.22)	85.37 (±0.18)
	PFAM (ours)	85.51 (±0.17)	85.71 (±0.25)
60%	SFP [32]	80.89 (±0.33)	73.39 (±0.30)
	FPGM [35]	84.29 (±0.21)	84.29 (±0.09)
	PFAM(ours)	84.82 (±0.14)	85.17 (±0.13)
80%	SFP [32]	74.46 (±0.27)	65.89 (±0.32)
	FPGM [35]	83.11 (±0.05)	83.92 (±0.25)
	PFAM (ours)	82.77 (±0.09)	82.80 (±0.23)

On the NWPU-RESISC45 dataset, SFP [32] continues its poor performance with the increasing pruning ratio, as shown in Figure 7a,b. In Figure 7a, PFAM (green line) outperforms the other two methods for all the pruning ratios, while in Figure 7b, PFAM achieves the best performance under all settings on ResNet-34, except for FPGM [35] at the pruning rate of 60%. It seems that our method occasionally performs worse than FPGM on one specific dataset. The reason might be that there are so many filters with similar correlation values, which happened to confuse our selection mechanism. In other words, our selection mechanism is not good enough, compared to FPGM, in this particular case. Although our approach does not perform very well at a pruning rate of 60% because of the filter selection, we still achieve the best performance compared to the three other settings, which can show the way that we chose the filters is better than the other comparison methods in most cases.

Although the experimental results on the three remote sensing datasets illustrate the effectiveness of the proposed filter pruning method, there are still some limitations. For example, the filter selection mechanism, given large compression ratios, does not seem optimal, though it works perfectly at relatively small compression ratios. Besides, our method treats each convolution layer as having equal significance during the pruning stage, which may have a huge negative impact on overall

performance if many filters need to be pruned from an important layer. Likewise, the neural network architecture still has high redundancy if only a few filters are pruned from some layers that are not significant. Therefore, how to prune filters in each convolution layer dynamically remains as a future research point.

6. Conclusions and Future Work

In this paper, we present a novel method termed Pruning Filter with Attention Mechanism (PFAM) for lightweight remote sensing image classification. In particular, a correlation-based filter pruning criterion is applied, where the correlation between filters is determined by the attention mechanism. Different from previous pruning methods, we prune filters with the least correlation, which has a small negative impact on the overall correlation among filters in one layer. These less correlated filters are firstly pruned after the pruning stage in the current training epoch, then they are recovered and updated during the next training epoch. In this way, the training data are processed by the original model during the training process, while the compressed network model can be obtained in the end without the need for the extra fine-tuning stage. The proposed method is extensively evaluated on three public remote sensing image datasets, and the experimental results show that our method achieves superior performance compared to the state-of-the-art methods. Notably, PFAM achieves the best performance under all types of ResNet architectures and VGG-16 on the AID dataset, especially for ResNet-34 and -50, which obtains 0.67% higher accuracy than the state-of-the-art at similar compression ratios. However, we still treat each layer equally during the pruning stage, which may have a huge negative impact on overall performance if it happens to prune many filters from an important layer. Hence, we will design a sort of dynamic pruning strategy, which could determine the number of to-be-pruned filters at certain layers based on their importance levels. Doing so will help reduce the model complexity dramatically while affecting the accuracy little. Additionally, in future work, we will use our technique to compress deep CNN architectures for various applications, such as visual tracking [52–54], video analysis [55,56], and large-scale visual search [57,58].

Author Contributions: S.Z., J.G., and J.H. were responsible for the overall design of the study. S.Z. performed the experiments and drafted the manuscript. S.Z., G.W., and J.H. revised the manuscript. All authors read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was supported in part by Hebei Provincal Innovation Capability Enhancement Project (199676146H).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, W.; Tang, P.; Zhao, L. Remote sensing image scene classification using CNN-CapsNet. *Remote Sens.* **2019**, *11*, 494.
2. Uss, M.; Vozel, B.; Lukin, V.; Chehdi, K. Efficient Discrimination and Localization of Multimodal Remote Sensing Images Using CNN-Based Prediction of Localization Uncertainty. *Remote Sens.* **2020**, *12*, 703.
3. de Bem, P.P.; de Carvalho Junior, O.A.; Fontes Guimarães, R.; Trancoso Gomes, R.A. Change Detection of Deforestation in the Brazilian Amazon Using Landsat Data and Convolutional Neural Networks. *Remote Sens.* **2020**, *12*, 901.
4. Nezami, S.; Khoramshahi, E.; Nevalainen, O.; Pölönen, I.; Honkavaara, E. Tree Species Classification of Drone Hyperspectral and RGB Imagery with Deep Learning Convolutional Neural Networks. *Remote Sens.* **2020**, *12*, 1070.
5. Maggiori, E.; Tarabalka, Y.; Charpiat, G.; Alliez, P. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Trans. Geosci. Remote Sens.* **2016**, *55*, 645–657.
6. Castelluccio, M.; Poggi, G.; Sansone, C.; Verdoliva, L. Land use classification in remote sensing images by convolutional neural networks. *arXiv* **2015**, arXiv:1508.00092.

7. Soudry, D.; Hubara, I.; Meir, R. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 963–971.
8. Esser, S.K.; Appuswamy, R.; Merolla, P.; Arthur, J.V.; Modha, D.S. Backpropagation for energy-efficient neuromorphic computing. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 1117–1125.
9. Courbariaux, M.; Bengio, Y.; David, J.P. Training deep neural networks with low precision multiplications. *arXiv* **2014**, arXiv:1412.7024.
10. Courbariaux, M.; Bengio, Y.; David, J.P. Binaryconnect: Training deep neural networks with binary weights during propagations. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 3123–3131.
11. Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to + 1 or −1. *arXiv* **2016**, arXiv:1602.02830.
12. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*; Springer: Amsterdam, 2016; pp. 525–542.
13. Wang, X.; Zhang, B.; Li, C.; Ji, R.; Han, J.; Cao, X.; Liu, J. Modulated convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 840–848.
14. Lin, M.; Chen, Q.; Yan, S. Network in network. *arXiv* **2013**, arXiv:1312.4400.
15. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
16. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
17. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
18. Hanson, S.J.; Pratt, L.Y. Comparing biases for minimal network construction with back-propagation. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 27–30 November 1989; pp. 177–185.
19. LeCun, Y.; Denker, J.S.; Solla, S.A. Optimal brain damage. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 26–29 November 1990; pp. 598–605.
20. Hassibi, B.; Stork, D.G. Second order derivatives for network pruning: Optimal brain surgeon. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 1-4 January, 1993; pp. 164–171.
21. Hassibi, B.; Stork, D.G.; Wolff, G.J. Optimal brain surgeon and general network pruning. In Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, 28 March–1 April 1993; pp. 293–299.
22. Van Nguyen, H.; Zhou, K.; Vemulapalli, R. Cross-domain synthesis of medical images using efficient location-sensitive deep network. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*; Springer; Munich, Germany, 2015; pp. 677–684.
23. Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both weights and connections for efficient neural network. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 1135–1143.
24. Tang, S.; Han, J. A pruning based method to learn both weights and connections for LSTM. In Proceedings of the Advances in Neural Information Processing Systems, NIPS, Montreal, QC, Canada, 7–12 December 2015.
25. He, Y.; Zhang, X.; Sun, J. Channel pruning for accelerating very deep neural networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 1389–1397.
26. Li, J.; Qi, Q.; Wang, J.; Ge, C.; Li, Y.; Yue, Z.; Sun, H. OICSR: Out-In-Channel Sparsity Regularization for Compact Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 7046–7055.

27. Hu, Y.; Sun, S.; Li, J.; Wang, X.; Gu, Q. A novel channel pruning method for deep neural network compression. *arXiv* **2018**, arXiv:1805.11394.
28. Ye, J.; Lu, X.; Lin, Z.; Wang, J.Z. Rethinking the smaller-norm-less informative assumption in channel pruning of convolution layers. *arXiv* **2018**, arXiv:1802.00124.
29. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning filters for efficient convnets. In Proceedings of the ICLR 2017, Toulon, France, 24–26 April 2017.
30. Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; Zhang, C. Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2736–2744.
31. Luo, J.H.; Wu, J.; Lin, W. Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5058–5066.
32. He, Y.; Kang, G.; Dong, X.; Fu, Y.; Yang, Y. Soft filter pruning for accelerating deep convolutional neural networks. In Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018.
33. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.
34. Guo, Y.; Yao, A.; Chen, Y. Dynamic network surgery for efficient dnns. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 1379–1387.
35. He, Y.; Liu, P.; Wang, Z.; Hu, Z.; Yang, Y. Filter pruning via geometric median for deep convolutional neural networks acceleration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 4340–4349.
36. Cheng, J.; Dong, L.; Lapata, M. Long short-term memory-networks for machine reading. *arXiv* **2016**, arXiv:1601.06733.
37. Mnih, V.; Heess, N.; Graves, A.; kavukcuoglu, K. Recurrent models of visual attention. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014, pp. 2204–2212.
38. Ba, J.; Mnih, V.; Kavukcuoglu, K. Multiple object recognition with visual attention. *arXiv* **2014**, arXiv:1412.7755.
39. Jaderberg, M.; Simonyan, K.; Zisserman, A.; kavukcuoglu, K. Spatial Transformer Networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2017–2025.
40. Wang, F.; Jiang, M.; Qian, C.; Yang, S.; Li, C.; Zhang, H.; Wang, X.; Tang, X. Residual Attention Network for Image Classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3156–3164.
41. Parikh, A.P.; Täckström, O.; Das, D.; Uszkoreit, J. A Decomposable Attention Model for Natural Language Inference. *arXiv* **2016**, arXiv:1606.01933.
42. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved techniques for training gans. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 2234–2242.
43. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7794–7803.
44. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. *arXiv* **2018**, arXiv:1805.08318.
45. Cheng, G.; Han, J.; Lu, X. Remote sensing image scene classification: Benchmark and state of the art. *Proc. IEEE* **2017**, *105*, 1865–1883.
46. Xia, G.; Hu, J.; Hu, F.; Shi, B.; Bai, X.; Zhong, Y.; Zhang, L.; Lu, X. AID: A Benchmark Data Set for Performance Evaluation of Aerial Scene Classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3965–3981.
47. Zou, Q.; Ni, L.; Zhang, T.; Wang, Q. Deep Learning Based Feature Selection for Remote Sensing Scene Classification. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 2321–2325.
48. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.

49. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in pytorch. In Proceedings of the NIPS 2017, Long Beach, CA, USA, 4–9 December 2017.
50. Berrada, L.; Zisserman, A.; Kumar, M.P. Smooth loss functions for deep top-k classification. *arXiv* **2018**, arXiv:1802.07595.
51. Luo, J.H.; Zhang, H.; Zhou, H.Y.; Xie, C.W.; Wu, J.; Lin, W. Thinet: Pruning cnn filters for a thinner net. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 2525–2538.
52. Ding, G.; Chen, W.; Zhao, S.; Han, J.; Liu, Q. Real-time scalable visual tracking via quadrangle kernelized correlation filters. *IEEE Trans. Intell. Transp. Syst.* **2017**, *19*, 140–150.
53. Zhang, B.; Luan, S.; Chen, C.; Han, J.; Wang, W.; Perina, A.; Shao, L. Latent constrained correlation filter. *IEEE Trans. Image Process.* **2017**, *27*, 1038–1048.
54. Han, J.; Pauwels, E.J.; de Zeeuw, P.M.; de With, P.H. Employing a RGB-D sensor for real-time tracking of humans across multiple re-entries in a smart environment. *IEEE Trans. Consum. Electron.* **2012**, *58*, 255–263.
55. Zhang, B.; Yang, Y.; Chen, C.; Yang, L.; Han, J.; Shao, L. Action recognition using 3D histograms of texture and a multi-class boosting classifier. *IEEE Trans. Image Process.* **2017**, *26*, 4648–4660.
56. Han, J.; Farin, D.; de With, P.H. Broadcast court-net sports video analysis using fast 3-D camera modeling. *IEEE Trans. Circuits Syst. Video Technol.* **2008**, *18*, 1628–1638.
57. Wu, G.; Han, J.; Lin, Z.; Ding, G.; Zhang, B.; Ni, Q. Joint image-text hashing for fast large-scale cross-media retrieval using self-supervised deep learning. *IEEE Trans. Ind. Electron.* **2018**, *66*, 9868–9877.
58. Wu, G.; Han, J.; Guo, Y.; Liu, L.; Ding, G.; Ni, Q.; Shao, L. Unsupervised deep video hashing via balanced code for large-scale video retrieval. *IEEE Trans. Image Process.* **2018**, *28*, 1993–2007.