International Journal of Innovative
Computing, Information and Control
Volume **19**, Number **6**, December **2023**

ICIC International ©2023 ISSN 1349-4198
pp. **1697–1715**

FILTER PRUNING BASED ON LOCAL GRADIENT ACTIVATION MAPPING IN CONVOLUTIONAL NEURAL NETWORKS

Monthon Intraraprasit and Orachat Chitsobhuk*

School of Engineering
King Mongkut's Institute of Technology Ladkrabang
Chalongkrung Road, Ladkrabang, Bangkok 10520, Thailand
61601175@kmitl.ac.th; *Corresponding author: orachat.ch@kmitl.ac.th

Received January 2023; revised April 2023

Abstract. *Convolutional Neural Network (CNN) is a well-known Deep learning model utilized extensively in the field of computer vision. The structure of convolutional neural networks is quite complicated and necessitates a substantial amount of computational time and storage resources. As a result, it is difficult to adopt a CNN model on a resource-constraint device. Model pruning can help to reduce computation time and storage requirements. In this research, we propose a filter pruning technique based on Localized Gradient Activation heatmaP (LGAP) for the purpose of pruning CNNs. Analyzing a filter based on statistical criterion of single neuron can lead to a loss in spatial relations within the filter activation itself, the relationship to target prediction, as well as the relationship among filters in that specific layer. To minimize the limitations, we evaluate the significance of a filter through the spatial information of local gradient activation related to the target prediction in terms of the layer-wise loss of the investigated filter. The effect of loss of an investigated filter demonstrates the significance or insignificance of the filter. Our pruning criteria ensure that these significant filters are preserved, while maintaining the model accuracy. The performance of our pruning method was validated using VGG-16 and ResNet-50. With pruning ratio of 50%, VGG-16 tends to decrease 1.66% of its accuracy, 3.6× of FLOP and 3.9× of storage reduction. For ResNet-50, with 50% pruning ratio, the results show that Top-1 and Top-5 of our pruning techniques outperform all the baseline techniques with a reduction of top-1 accuracy by 3.56%, top-5 accuracy by 1.89%, Floating Point Operation by 2.3×, and storage by 2.05×.*
Keywords: Filter pruning, Convolutional neural networks, Deep learning, Model compression

1. **Introduction.** Convolutional Neural Networks (CNNs) are being used to solve human problems in both industry and academia. In computer vision, they are particularly helpful for tasks like image classification [1, 2], object detection [3, 4, 5, 6, 7, 8], object recognition [9, 10, 11, 12], and semantic segmentation [13, 14]. The performance of CNNs has been demonstrated to be exceptional in recent years. Their structures are complex and made up of numerous interconnected layers. This structure enables in capturing data patterns through filters in each layer. Normally, CNNs have filters for detecting edges and colors in their first layer. Deeper layers can recognize more complex patterns (grids or stripes) by using data from earlier layers. This results in the layer's filter having a large number of parameters and a large amount of storage space. For example, ResNet-50 [15] contains 25.6 million parameters and requires 98.2 megabytes (MB) of storage space. It influences the total amount of time necessary for the learning and prediction procedures.

DOI: 10.24507/ijicic.19.06.1697

The cost of the Graphic Processing Unit (GPU) to run the model is considerable. Moreover, it is difficult to execute CNNs model on hardware with limited resources [16], such as a microcontroller or smart phone. Limited resources refer to limited battery, computational unit, storage, etc. Model compression is a key strategy for running CNNs models effectively in constrained environments and resources.

Model compression is a technique for reducing the size of CNN's model, which can be classified into 3 categories [17, 18]. 1) Precision reduction is a technique for maintaining CNN performance while reducing the number of bits necessary to represent the CNN weights [17]. Although this technique can minimize the amount of storage and power requirement [19], it greatly depends on the criteria for selecting the number of bits, which could affect the model's accuracy. 2) Network pruning, a compression technique, is used to evaluate weights with less impact on the model's accuracy. The chosen weights are then set to zero or eliminated from CNN, and retraining is performed after that. The weight removal procedure is challenging, and the outcomes heavily depend on weight analysis. Inefficient weight analysis might result in decreased accuracy and ineffective compression issues [17]. 3) Compact network architecture is a technique for reducing the number of weights in CNN by altering the kernel design. Presently, a new structure in a sequence of convolutional layers with reduced kernel sizes has replaced the underlying structure. For instance, one of the most well-known architectures of the compact CNN is SqueezeNet [20]. The three strategies are implemented as follows: 1) The kernel size of the filter has been reduced from 3×3 to 1×1; 2) There is a reduction in the number of input channels that are transmitted to the 3×3 filters; 3) Subsequent layers reduce the feature map to a more manageable size. The compact nature of SqueezNet results in a significant reduction in the number of parameters, which in turn minimizes the amount of processing time required for CNN training and operation. A significant amount of information has been lost due to the squeeze convolutional layer's reduction of the channel number. Despite the fact that the model makes an effort during the expand process to recover the information loss, it is impossible to restore such a large amount of information that has been pruned, particularly when non-linear layers are being replaced.

Network pruning is the most intriguing technique among all CNN compression techniques. According to network training observation, this strategy can remove a variety of irrelevant weights with less impact on the pruned model's accuracy. The authors of [21] made use of the second derivative principle to enable tradeoffs between training errors and network complexity. Even though it is useful for real-world applications, the structure of a deep model necessitates a significant amount of storage space and processing capacity.

It was recently proposed to use the basic pruning technique to reduce weights when all connections were below a certain threshold. The model was then adjusted to restore accuracy [22]. After numerous iterations, the model became extremely sparse and non-structured, making it unsupportable by standard libraries. As a result, the model needed specialized hardware and software in order to collaborate effectively due to the unpredictable nature of network connections. The issue of non-structured random connectivity had an impact on memory accessibility and cache utilization [23]. Random connectivity caused poor cache locality, erratic memory access, and potential effects on practical acceleration. This restriction can occasionally cause a slowdown in acceleration.

Filter level pruning is one way to overcome non-structured limitations [24]. After redundant filters were removed, the pruned model was not sparse and did not cause a non-structure issue. This made it possible for the model to be more effective than non-structured pruning in the following ways. 1) This results in valid network connectivity, where implementation can be assisted by deep learning libraries, since the model structure is not harmed after pruning. As a result, the model can be further compressed using

other techniques, such as quantization of the parameters [25]. 2) As a result of model parameters being pruned, storage can be significantly reduced.

From previous research, it can be seen that the pruning criteria used for filter selection can have a great effect on the performance of the pruned models. The statistical based criteria are mostly adopted since the estimation is simple and can be used to represent the overall significance level of a neuron. Global representation might lack ability to deal with localizing the objects or Region-of-Interest (ROI), whereas single neuron analysis may lose information regarding the relationships within and between layers respected to the final prediction.

We, therefore, proposed a filter pruning method based on the Localized Gradient Activation heatmaP (LGAP), which considers the spatial correlation between the investigated filter and the target prediction. The LGAP was adopted as our criteria to prune filters in the CNN layers by analyzing the significance of a filter through the similarity of each CNN layer's behavior before and after pruning to estimate the filter's impact. The layer-wise loss of the investigated filter was determined in order to analyze the gradient difference between a complete layer and a layer without an investigated filter. The effect of loss of an investigated filter demonstrates the significance or insignificance of the filter. Our pruning criteria ensure that these critical filters are preserved, while maintaining the model accuracy. The model is then fine-tuned to recover prediction performance after pruning. The effectiveness of the proposed pruning method will be compared to other pruning techniques. From the experimental results, filter pruning based on our LGAP strategy can identify appropriate filters to be pruned, thus preserving the superior performance over other pruning techniques. The pruned model contributes to a reduction in the total amount of computational time, the number of parameters, and the amount of storage space required, depending on the selected pruning ratio.

2. **Related Work.** Compression techniques can be used to resolve the problem of an excessive number of parameters presented by deep learning models. To accelerate CNN [21], the model coefficients were initially removed using the optimal brain damage technique. This was one of the early techniques for pruning a network. The authors attempted to select parameters for regularization of the pruning process using a second-order Taylor expansion. In contrast to conventional fine-tuning, this method demanded the computation of the Hessian matrix, which added extra memory and computational overhead. Research on a new technique that integrates group-wise pruning into the learning phase was proposed by Lebedev and Lempitsky [26]. This allowed for group sparsity regularization. Since most of their values were close to zero, some weight groups might be eliminated. The approach in [23] adopted the Structured Sparsity Learning (SSL) technique. CNN structures can be regularized in a variety of ways, including the use of filters, filter shapes, channels, and layer depth. Even though SSL can effectively prune the network, the loss of the basic network structure may result in unsupported libraries. Other filter level pruning methods were proposed such as random pruning, weight sum technique [27], Average Percentage of Zeros (APoZ) [28], mean gradient [29], and Structured Sparsity Regularization (SSR) [30]. 1) Random pruning, in which filters are removed at random. 2) The technique of weight sum in [27] eliminated filters based on the sum of their absolute weights. As the weight sum increases, the filter becomes more significant. 3) Average Percentage of Zeros (APoZ) [28] technique determined the significance score of the filter, which was computed from the percentage of zero values in the activation output. A large percentage of zero was an indication that the filter was unnecessary and should be removed. 4) Mean gradient [29] is a pruning technique that analyzed layer feature maps based on its mean gradient. 5) Structured Sparsity Regularization (SSR) [30] was a technique in which the objective

function incorporated the structured sparsity constraint as well as the correlation between the global output loss and the local filter removal. Alternative Updating with Lagrange Multipliers (AULM) was employed for adaptive filter pruning based on two different types of structure sparse regularizers to deal with the convergence challenge. The approach was reportedly capable of achieving a fast convergence rate. 6) ThiNet [24] adopted a greedy strategy for channel selection, to prune the target layer by greedily selecting the input channel that has the least increase in reconstruction error.

The majority of currently available methods tend to concentrate on applying statistics to the filter weights (weight sum [27]) or feature map activations (APoZ [28], and SSR [30]), as a statistical criterion to distinguish the importance of filter neurons. However, these methods did not carry out data-driven (data feed forward) operations across the network of the model, which led to a lack of relational data analysis inside a layer as well as consequences on the data's propagation to the final prediction. Even though the statistical criterion can be used to represent overall significant level of a neuron, aiming to understand the predictions of a model by analyzing the individual units and seeking an explanation for specific activation, it does not provide confident relationship of the neuron respect to the final prediction. ThiNet [24] evaluates channel importance based on reconstruction loss from the least-square estimation to identify filter channels with the smallest impact to the output feature map. Nevertheless, previous mentioned techniques rely on the analysis of statistical evaluation of the filters, which lack the spatial information related to the target prediction. The localized prediction map is helpful in distinguishing the critical regions of the target prediction. Moreover, there have been fewer studies on the effects of pruned filter on the layer-wise activation.

Eliminating filter based on statistical criterion of single neuron analysis can lead to a loss in spatial relations within the filter activation itself, as well as the relationship to target prediction and the relationship among filters in that specific layer. To minimize the limitations, our contribution aims to

1) Analyze the localized gradient activation to estimate the spatial relationship between an individual filter and the output prediction;

2) Estimate filter pruning criterion based on the Localized Gradient Activation heatmaP (LGAP), which considers the spatial relationships between the feature map activation of the investigated filter and the target prediction;

3) Evaluate weak filter pruning strategy based on layer-wise loss of the investigated filter.

LGAP offers the advantage of using a data-driven method in conjunction with a study of the spatial relationship among filters in each layer based on layer-wise behavior rather than a single neuron analysis to identify weak filters while preserving the model's performance. As a result, the efficacy of our pruned model outperforms the previously discussed strategies [24, 28, 30].

3. **Methods.**

In this section, we detail our filter pruning technique into 4 sections: an overview of our filter level pruning method (Section 3.1), the LGAP estimation (Section 3.2), Significant filter scoring (Section 3.3), and weak filter pruning (Section 3.4).

3.1. **Overview of our filter level pruning.**

Our filter pruning process is performed on a pre-trained CNN model. To visualize the behavior of a filter neuron, the idea of gradient-based localization [31] is adopted. The localization can be achieved by passing a single forward of a set of training data to the pre-trained model. The gradient information is then estimated according to partial backpropagating gradient respected to feature map activations from the last convolutional layer of CNN to the beginning layers since the last

convolutional layer is expected to be the best approximation between high-level semantics from the fully-connected layers and detailed spatial relations from the CNN layers. The benefit of gradient-based localization is its ability to generate visual explanations from CNN neuron behaviors without architectural modifications. The global average pooling of local-gradient information of each filter is calculated as a filter significant weight.

Then, a layer-wise heatmap of weighted channel activations is generated as a layer summary. As a result, the activation from filter that corresponds best to the top class output will be emphasized, while the activation from the filter that corresponds less to the desired class will be diminished. The layer-wise heatmap can provide an excellent visualization on the layer's behavior. However, it may not be efficient illustration of the filter's behavior. Analyzing individual filter in each layer can be performed in two ways, through analysis of a significant weighted activation heatmap corresponding to a single filter neuron, or through analysis of heatmap loss from eliminating an investigated filter activation. We discover that analyzing significant weighted activation heatmap on a single layer basis may not be an effective way to determine the significance of a filter due to large variations in activations among filters in specific layers. This can have an impact on the development of filter scoring criterion for determining significant filters to be preserved and pruning strategy.

However, analyzing the similarity of a complete layer versus a layer without an investigated filter based on a local gradient emphasized heatmap, known as loss persistence heatmap, can provide an effect of loss of an investigated filter and prove to be better distinguish significant from insignificant filters. Therefore, we would use the loss persistence heatmap as our filter scoring criterion to prune the less effective filter to the desired output class. The overview of our filter level pruning technique is illustrated in Figure 1.

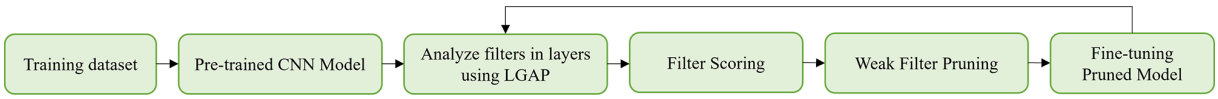


FIGURE 1. Overview of our pruning technique

3.2. **Localized Gradient Activation heatmaP (LGAP).** To estimate the LGAP, we first compute the gradient estimation of the top class prediction with respect to an output feature map activation obtained from training dataset prediction before softmax as shown in Equation (1) [31].

$$G^{l,k} = \frac{\partial y^c}{\partial A^{l,k}} \tag{1}$$

where $G^{l,k}$ is the gradient estimation of the kth filter in the layer l, y^c is the output of the top predicted class c, and $A^{l,k}$ is the output feature map activation of the kth filter neuron in the layer l.

The significant weight (α) of a neuron in terms of a global-average-pooling of the gradient $G^{l,k}$ over both its height (H) and width (W) corresponding to the output class c can be calculated in Equation (2) [31]. The results of significant weighted feature map help to emphasize the feature map of the significant filter, which can be visualized in Figure 2.

$$\alpha_{l,k}^c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} G_{i,j}^{l,k} \tag{2}$$

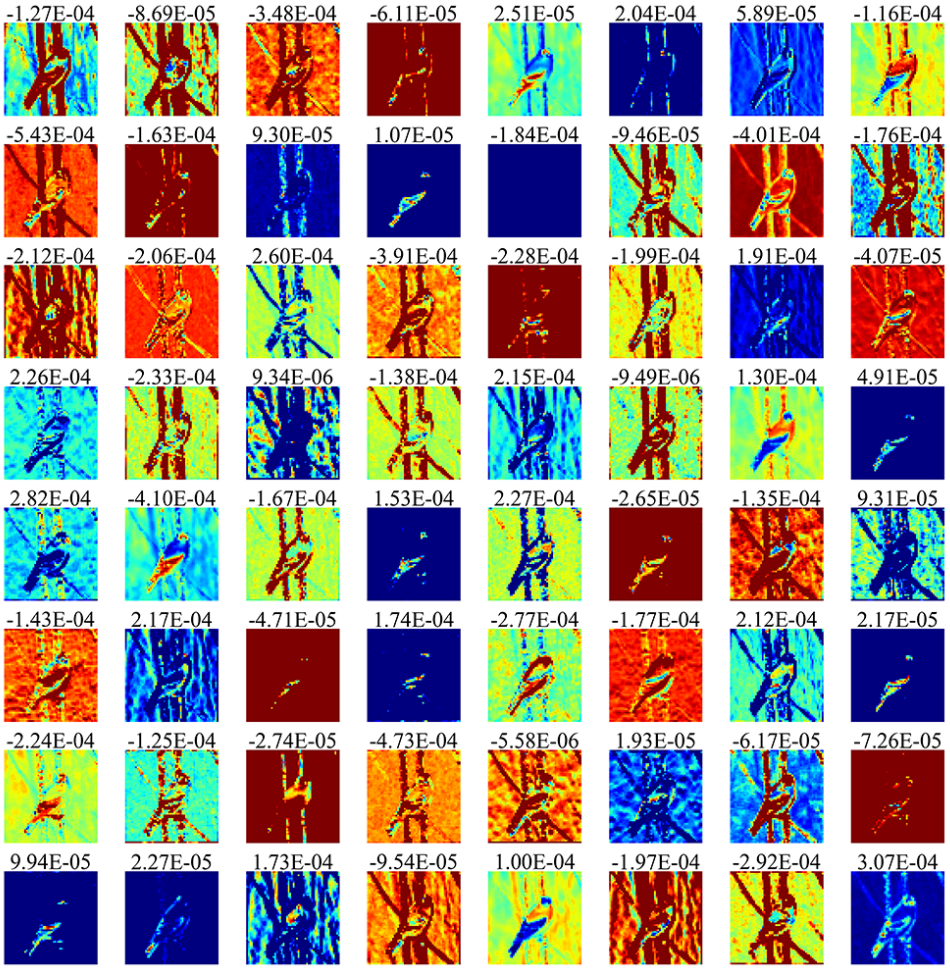


FIGURE 2. Significant weighted feature map

A layer summary can be expressed in terms of the layer-wise heatmap of weighted channel activations called Localized Gradient Activation heatmaP (LGAP), where the activations from filters best corresponding to the top-class output will be emphasized. We found that the layer-wise heatmap (H_c) can provide an excellent visualization on the layer's behavior and can be calculated in terms of a normalized sum of the significant weighted feature map activations of all neurons in the particular lth layer and we proposed those relations in Equation (3).

$$H_c = norm\left(\left|\sum_k \alpha_{l,k}^c A^{l,k}\right|\right) \tag{3}$$

where $|.|$ is absolute value and *norm* is based on the max-min normalization.

Localized gradient information estimates are typically based on regulated activation, with the assumption that negative activation may correspond to an undesired class [31]. Neglecting negative responses, on the other hand, may be beneficial for visualization, but may not be effective for filter significant analysis. To learn all filter's behaviors, both positive and negative activations are taken into account using a normalized activation heatmap visualization.

Analyzing individual filter in each layer can be performed through analysis of a significant weighted activation heatmap corresponding to a single filter neuron or analysis of the

heatmap of loss from eliminating an investigated filter activation. We discover that analyzing significant weighted activation heatmap on a single layer basis may not be an effective way to determine the significance of a filter due to large variations in activations among filters in specific layers. This can have an impact on the development of filter scoring criterion for determining significant filters to be preserved and pruning strategy. However, analyzing the similarity of a complete layer versus a layer without an investigated filter based on a local gradient emphasized heatmap, known as loss persistence heatmap, can provide an effect of loss of an investigated filter and prove to be better distinguished significantly from insignificant filters. Therefore, we would use the loss persistence heatmap ($H_{l,f}$) to calculate our filter scoring criterion to prune the less effective filter respected to the desired output class. The loss persistence heatmap can be calculated in Equation (4).

$$H_{l,f} = norm\left(\left|\left(\sum_k \alpha_{l,k}^c A^{l,k}\right) - \alpha_{l,f}^c A^{l,f}\right|\right) \tag{4}$$

where $A^{l,f}$ is the activation of fth filter kernel at layer l, which will be removed.

3.3. **Filter scoring.** This section explains our filter scoring criteria. Each filter is scored according to a similarity between the heatmap of a complete layer (H_c) and the loss persistence heatmap ($H_{l,f}$) (see in Figure 3). The similarity can be viewed as a persistence score when the particular filter f is removed from the layer. The higher the persistence score, the less the filter's removal effects on the desired output. We calculate the similarity based on cosine similarity method as shown in Equation (5). This comparison shows how the layer l is changed after the filter f is removed.

$$S_{c,f} = \frac{H_c, H_{l,f}}{||H_c||, ||H_{l,f}||} \tag{5}$$

where $S_{c,f}$ is the similarity score of the fth filter in terms of cosine similarity between H_c and $H_{l,f}$, and $||.||$ is L2 norm.

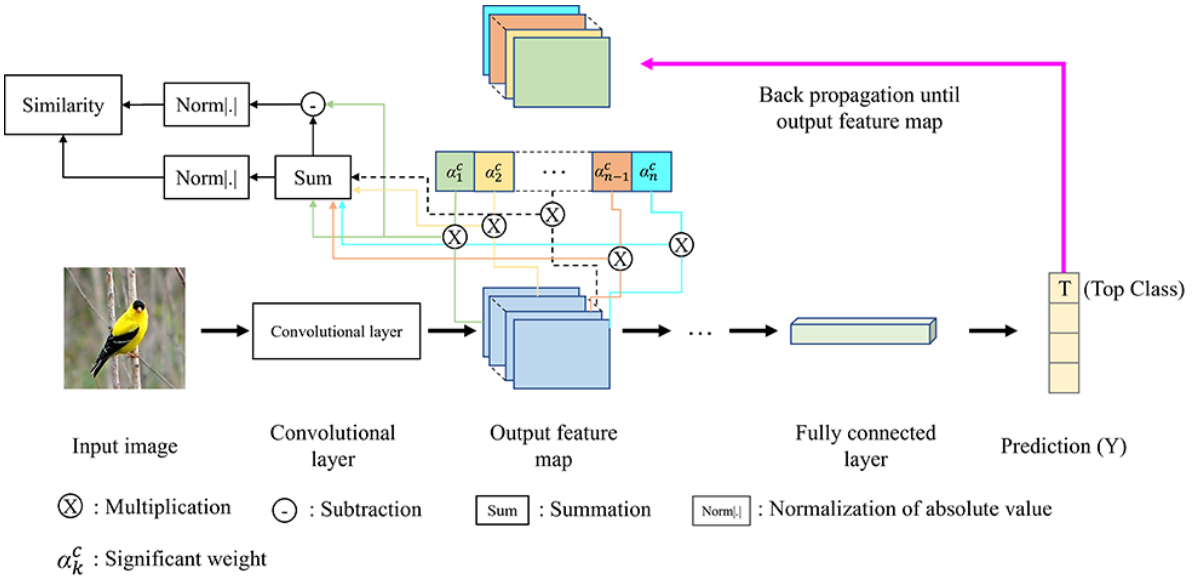


FIGURE 3. Overview of our filter scoring procedure for the lth layer

After obtaining cosine similarity of all filters, we rank the similarity results. The higher the similarity, the less effect the removed filters have on layer operation and are considered as weak filters that must be pruned from the layer.

3.4. **Weak filter pruning.** The procedure of weak filter pruning of layer l in CNN model is illustrated in Figure 4. In the first step, the output feature maps of layer l ($A^l \in \mathbb{R}^{H \times W \times K}$, where H is the height, W is the width and K is the number of channels) corresponding to the convolution filters at the lth layer ($conv^l \in \mathbb{R}^{S \times S \times K \times D}$, where $S \times S$ (3×3) is kernel size, K is the number of filter channels and D is the number of filters or

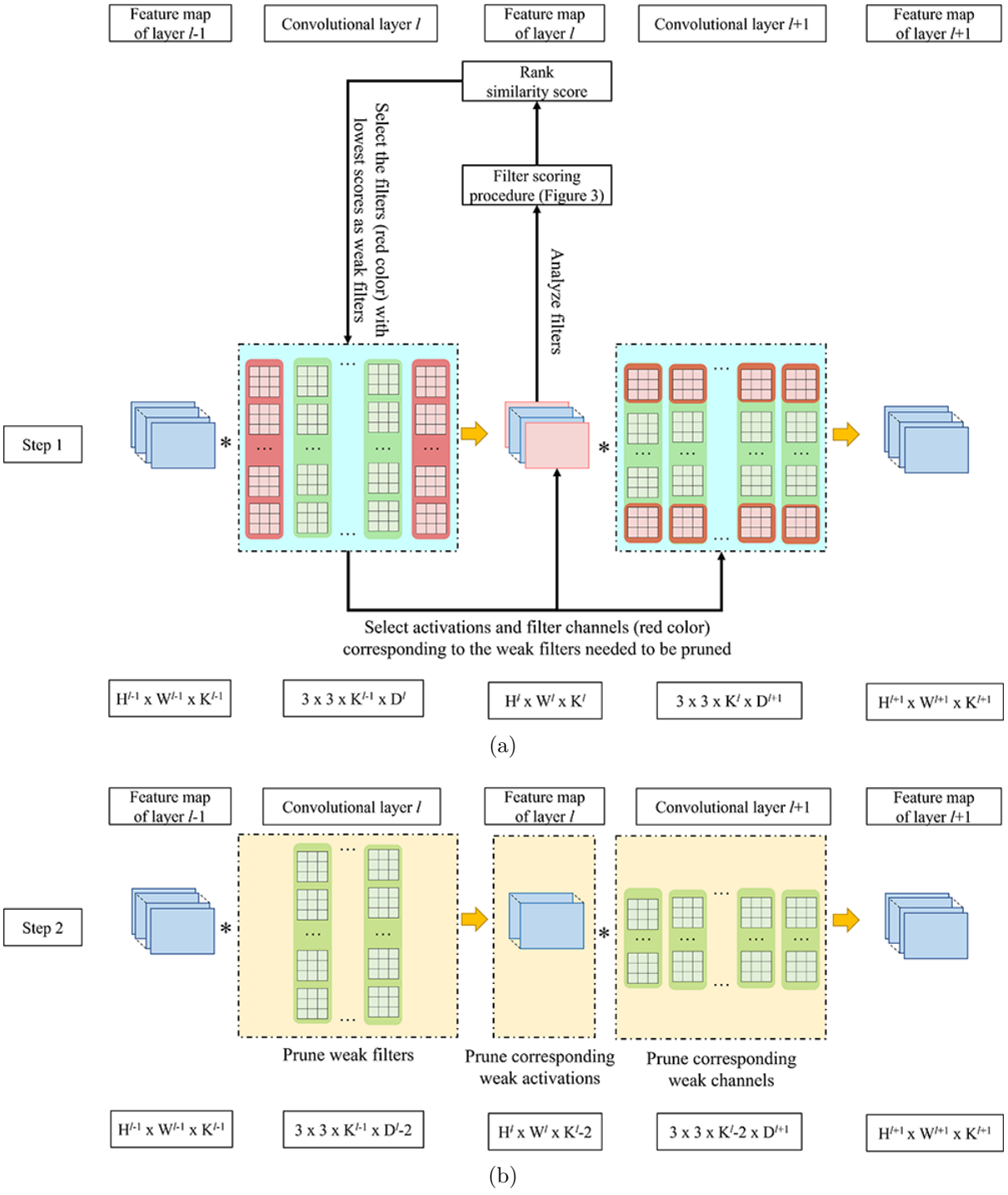


FIGURE 4. (color online) Weak filter pruning: (a) Step 1: Weak filter analysis; (b) Step 2: Prune weak filters and their corresponding weak activations and channels

number of channels of next $(l+1)$th layer) are extracted and the LGAP and filter score corresponding to the top-class output is estimated to obtain the loss persistence score of each filter. The n filters with the highest filter score are considered as weak filters since losses of those filters have less effect to the lth layer. The average filter scores of all images in the dataset for all filters in the specified layer are ranked and the filters corresponding to the highest scores are pruned in the second step. Pruning filters of layer l will affect their output feature maps of the next $(l+1)$th layer, which must also be removed as illustrated in Figure 4. After filter pruning, CNN model is fine-tuned to recover the model's prediction performance.

4. **Experiments.** In this session, we evaluate our filter level pruning method on pre-trained CNN models in Keras Applications [32]. We explain the experimental implementation in detail in Section 4.1, while, in Section 4.2, presenting a comparison of the pruning results obtained from our pruning technique and several baseline techniques.

4.1. **Implementation details.** In this section, we explain the dataset and the pre-trained models used to analyze and validate our pruning technique. To study the pruning impacts and performance losses, the models are pruned with different compression ratios.

4.1.1. *Dataset and models.* Our filter level pruning technique is evaluated on 2 different datasets.

• CIFAR-10 [33]: it is a color image dataset consisting of 60,000 images with 32×32 pixels in size. There are 10 different categories labeled from 1 to 10. The numbers of training and testing images are 50,000 and 10,000 images, respectively.

• ImageNet (ILSVRC 2012) [34]: it is a large image dataset with 1,000 different categories. There are 1.2 million images for training set and 50,000 images for validation set. All images are resized to 256×256 pixels and then cropped to 224×224 pixels based on their center.

For ImageNet, the images are resized to 256×256, and then cropped at its center area to obtain 224×224 pixels in size. An augmentation technique was used for generating transformation effects such as horizontal flip. The CIFAR-10 test set and ImageNet validation set are used to verify classification performance of the pruned model.

Two state-of-the-art classification models, VGG-16 and ResNet-50, are adopted.

• VGG-16 [35]: it is a CNN model developed from large scale of images for image classification. The base structure of model was trained with ImageNet dataset, and then modified to fit CIFAR-10 dataset [36] to illustrate the effective model performance. VGG-16 on CIFAR-10 consists of 13 convolutional layers with 3×3 kernel size. Filter pruning starts from the second convolutional layer since the first convolutional layer is necessary for creating the basis output feature map as mentioned in previous related works. The fully connected layers are modified to reduce the number of categories from 1000 ImageNet's categories down to 10 CIFAR-10's categories. The smaller size of CIFAR-10's images can greatly reduce the number of parameters from baseline model. Moreover, the model is integrated with a batch normalization layer [37] and a dropout layer [38] after each convolutional layer. The CNN structure of VGG-16 for CIFAR-10 is shown in Table 1. Floating Point Operation (FLOPs) in this paper referred to $2\times$ of Multiply-Accumulate Computations (MACs). VGG-16 model has various convolutional layers with the total of FLOPs and parameter as 6.3E+08 and 1.5E+07, respectively.

• ResNet-50 [15]: it is a famous CNN model. This model was developed using the same large-scale dataset (ImageNet) as the baseline VGG-16, but with improved model performance. Since ResNet-50 has special structure of the residual blocks (as seen in Figure 5), the model can be pruned for the first two convolutional layers and the channels

TABLE 1. CNN structure of VGG-16 on CIFAR-10

Layer name	Filter ($S \times S \times K \times D$)	FLOPs	Parameter
conv2d	$3 \times 3 \times 3 \times 64$	3.5E+06	1.8E+03
conv2d_1	$3 \times 3 \times 64 \times 64$	7.5E+07	3.7E+04
conv2d_2	$3 \times 3 \times 64 \times 128$	3.8E+07	7.4E+04
conv2d_3	$3 \times 3 \times 128 \times 128$	7.5E+07	1.5E+05
conv2d_4	$3 \times 3 \times 128 \times 256$	3.8E+07	3.0E+05
conv2d_5	$3 \times 3 \times 256 \times 256$	7.5E+07	5.9E+05
conv2d_6	$3 \times 3 \times 256 \times 256$	7.5E+07	5.9E+05
conv2d_7	$3 \times 3 \times 256 \times 512$	3.8E+07	1.2E+06
conv2d_8	$3 \times 3 \times 512 \times 512$	7.5E+07	2.4E+06
conv2d_9	$3 \times 3 \times 512 \times 512$	7.5E+07	2.4E+06
conv2d_10	$3 \times 3 \times 512 \times 512$	1.9E+07	2.4E+06
conv2d_11	$3 \times 3 \times 512 \times 512$	1.9E+07	2.4E+06
conv2d_12	$3 \times 3 \times 512 \times 512$	1.9E+07	2.4E+06

conv2d: 2-dimensional operation of convolutional layer

$S \times S \times K \times D$: $S \times S$ is kernel size, K is the number of channels and D is number of filters

FLOPs: Floating Point Operation

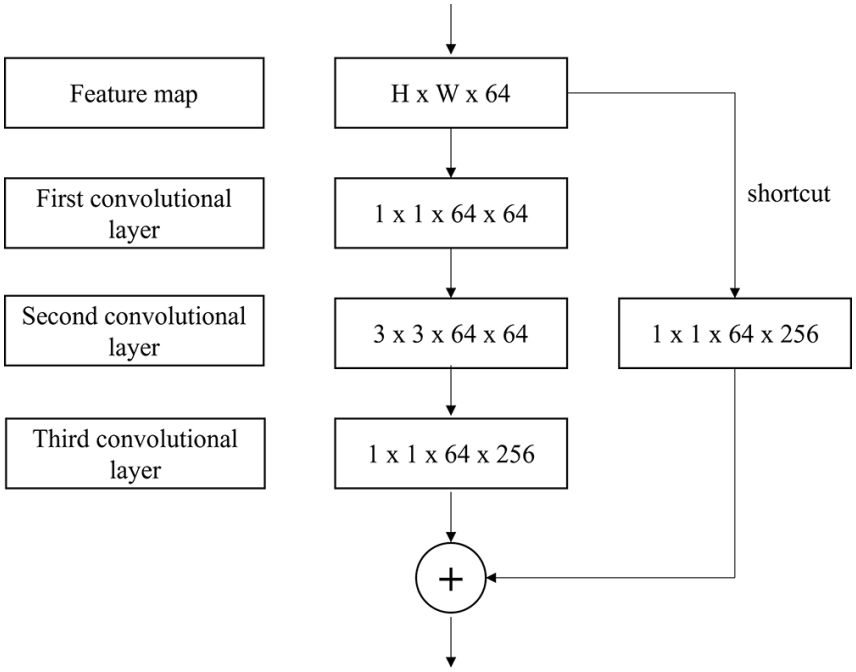


FIGURE 5. An example of the residual block of ResNet-50

of the next layers corresponding to the pruned layers must be removed. The output feature map of the third layer does not change dimension. The ResNet-50 trained on ImageNet is used to validate our filter pruning technique. Since ResNet-50 model consists of various convolutional layers and its structure is more complicated than VGG-16 as shown in Table 2, we will illustrate FLOPs and parameters of each layer in terms of a summarized base structure with the total of 7.7E+09 FLOPs and 2.3E+07 parameters, respectively.

TABLE 2. CNN structure of ResNet-50 on ImageNet

Layer name	Filter ($S \times S/D$)	Number of series	Output size
conv2d_1	$7 \times 7/64$	1	112×112
conv2d_2_x	$1 \times 1/64$	3	56×56
	$3 \times 3/64$		
	$1 \times 1/256$		
conv2d_3_x	$1 \times 1/128$	4	28×28
	$3 \times 3/128$		
	$1 \times 1/512$		
conv2d_4_x	$1 \times 1/256$	6	14×14
	$3 \times 3/256$		
	$1 \times 1/1024$		
conv2d_5_x	$1 \times 1/512$	3	7×7
	$3 \times 3/512$		
	$1 \times 1/2048$		

conv2d: 2-dimensional operation of convolutional layer
$S \times S/D$: $S \times S$ is kernel size and D is number of filters

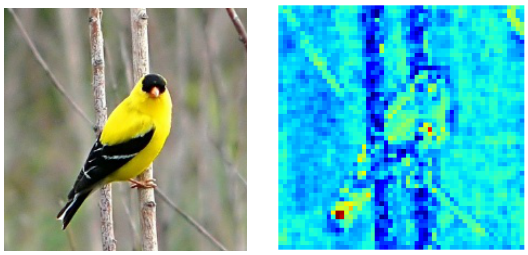
Prior to pruning the pre-trained model, we must explore parameters (learning rate, momentum, weight decay, and batch size) in order to fine-tune the original pre-trained models and produce results equivalent to the original pre-trained model. These parameters will be utilized to regain the pruning model's performance. We discovered that the VGG-16 model based on CIFAR-10 would be well fine-tuned with the following parameters: SGD optimizer, learning rate of 0.001, momentum of 0.9, weight decay of 0.0005, and batch size of 128; however, the parameters of ResNet-50 based on ImageNet would require the same SGD optimizer with a difference in batch size of 32. After pruning all layers based on a layer-by-layer basis, both models are fine-tuned to restore model performance. The number of fine-tuned epochs for VGG-16 on CIFAR-10 and ResNet-50 on ImageNet has been set to 40 and 50, respectively, with scheduled learning rate ranging from 10^{-3} to 10^{-5}.

4.1.2. *Pruning ratio.* It is a challenge to discover suitable percentage of pruning for each layer since each layer has different pruning sensibilities. For VGG-16 on CIFAR-10, our experiments are implemented on pruning ratio of 10%-90% for each layer. However, for ResNet-50 on ImageNet containing a large model size, we perform the experiments on 30%, 50% and 70% since it takes a long time to prune and fine-tune the model. Pruning can help to decrease complexity and size of the model. Large compression ratio is always preferred to reduce the complexity while maintaining the model performance.

4.1.3. *Filter scoring results.* A set of images from training dataset is randomly selected as inputs to obtain the output feature maps, which will be used to analyze filter and estimate filter pruning scores. Figure 6 shows an example of the input image (ImageNet) that is fed into the ResNet-50 to analyze first convolutional layer and obtain the corresponding layer-wise heatmap based on LGAP. Loss persistence heatmaps based on LGAP (from Equation (4)) are estimated from all filters in each layer (for example, 64 filters in the first layer ResNet-50 in Figure 7). Then, a filter pruning score (from Equation (5)) is calculated and the average score $Score_f$ (from Equation (6)) is investigated to analyze the significance of a filter.

$$Score_f = \frac{1}{N}\sum_{n=1}^{N} S_{c,f} \tag{6}$$

where N is number of input images. The filter scores are ranked in ascending order. The lower the score, the more significance the filter.



(a) (b)

FIGURE 6. (a) The input image and (b) the layer-wise heatmap based on LGAP (from Equation (3))

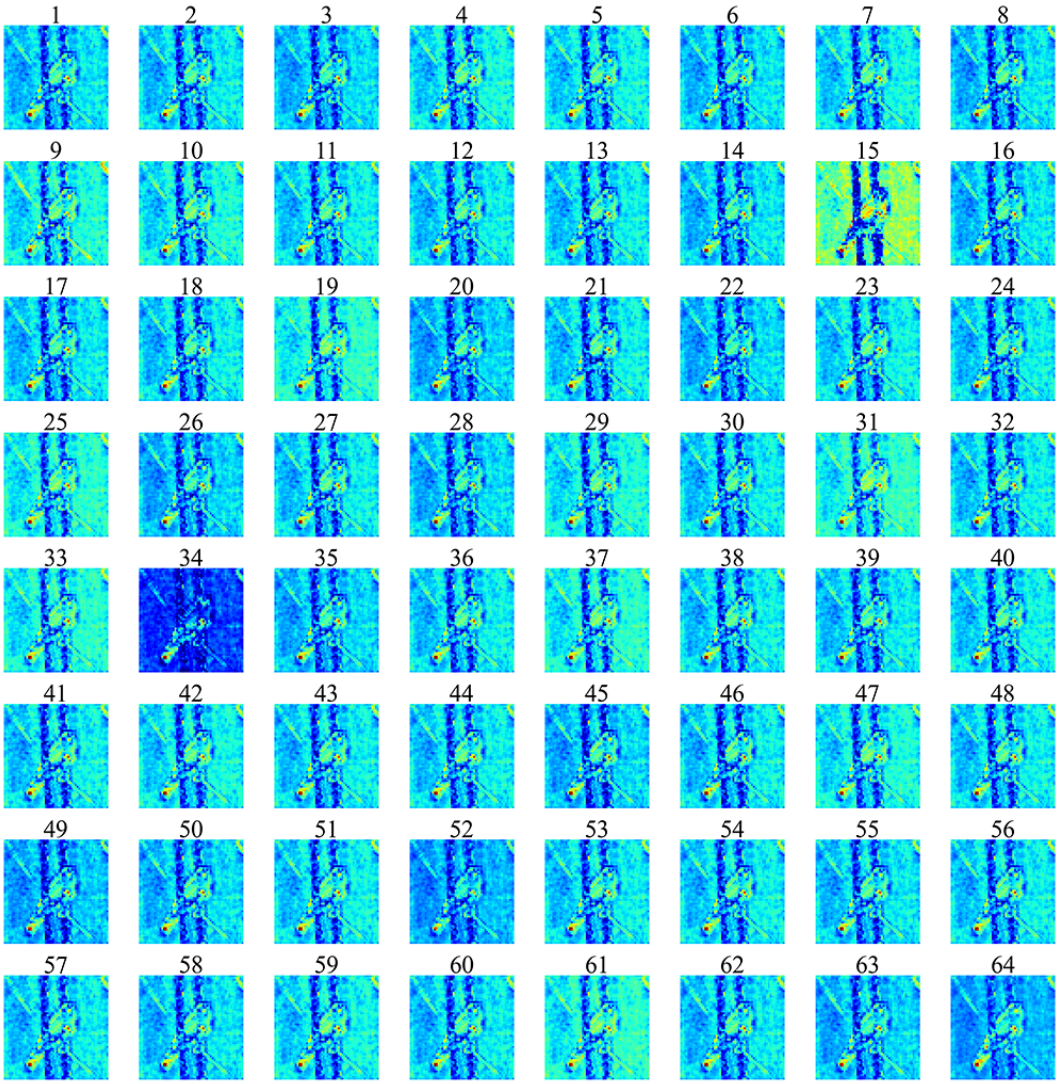


FIGURE 7. Loss persistence heatmap (Equation (4))

In our experiments, sets of 10, 100, and 200 images per category are adopted from CIFAR-10 and used to analyze our pruning technique on VGG-16. An example of filter pruning on the first convolutional layer of VGG-16 with pruning ratio 50% can achieve the accuracy of 93.24%, 93.33%, and 93.28%, respectively. It can be seen that increasing number of input images may not cause significant effects on the model accuracy. Hence, our experiment on VGG-16 will adopt a set of 10 images per class for filter analysis and scoring for both VGG-16 on CIFAR-10 and ResNet-50 on ImageNet.

4.2. **Pruning results.** In this section, we illustrate result from our pruning technique via VGG-16 and ResNet-50 compared to other related pruning techniques.

4.2.1. *Results from pruning VGG-16 on CIFAR-10.* First, we illustrate the results of our filter pruning for each convolutional layer with different pruning ratios in Figure 8. The capability of the model to endure a larger pruning ratio can be determined by the severity of the decline in accuracy caused by the pruned layer (sensibility). As the pruning ratio increases, the accuracy of the lower layers (conv2d_1 to conv2d_6) tends to be affected more than the accuracy of the deeper levels (conv2d_7 to conv2d_12).

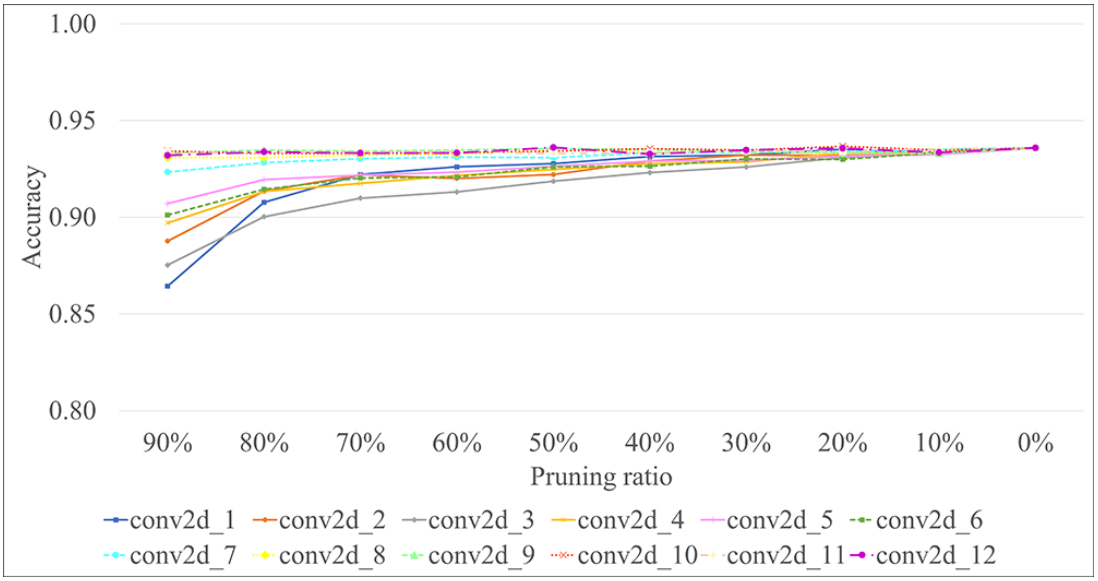


FIGURE 8. VGG-16 model accuracy after filter pruning each convolutional layer with different pruning ratios

We perform layer-wise pruning of all convolutional layers (conv2d_1 to conv2d_12) with the pruning ratio in range of 10%-90% with the same pruning ratio for all layers (see in Table 3). In Table 3, we calculate FLOPs from conv2d_1 layer to conv2d_12 layer. For test set of CIFAR-10 (10,000 images), the pruned model at 30% pruning ratio provides a reduction in FLOPs, parameters, and storage by 2×. Prediction time can be reduced about 1.7× while accuracy is decreased about 0.71%. The most reduction in FLOPs, parameters, storage and prediction time are 44.1×, 84.4×, 69.5× and 2.2×, respectively, with the accuracy reduction about 17.07% at 90% pruning ratio.

4.2.2. *A performance comparison for VGG-16 on CIFAR-10 with different filter pruning techniques.* In order to evaluate our filter pruning method, a performance comparison is conducted with several baseline techniques on N images, described as follows.

1) Random: filters are randomly pruned from layers. The result with the random seed is selected.

TABLE 3. The model performance after filter pruning in all layers on VGG-16

Pruning ratio	Error (%)	FLOPs	Parameters	Storage size (MB)	Prediction times (Second)
0%	6.41	6.3E+08	1.5E+07	57.4	2.25
10%	6.32	5.1E+08	1.2E+07	46.2	2.21
20%	6.64	4.1E+08	9.6E+06	36.6	2.02
30%	7.12	3.2E+08	7.4E+06	28.2	1.70
40%	7.28	2.4E+08	5.4E+06	20.9	1.31
50%	8.07	1.7E+08	3.8E+06	14.6	1.10
60%	8.53	1.2E+08	2.4E+06	9.4	1.10
70%	10.34	7.2E+07	1.4E+06	5.5	1.07
80%	13.88	3.7E+07	6.4E+05	2.6	1.08
90%	23.48	1.4E+07	1.8E+05	0.8	1.04

MB: Megabyte

2) Weighted sum [27]: a sum of absolute filter kernel weight values is calculated as a filter score. Weak filters with low weighted sum scores are pruned.

$$Score_i^{\text{weighted sum}} = \sum |W(:,:,:,i)|$$

where W is the weights of each filter i.

3) APoZ [28]: a percentage of zero activation after ReLU function from each filter is calculated as a filter score. The higher APoZ score, the weaker the filter.

$$Score_i^{\text{APoZ}} = \frac{1}{N \times H \times W} \sum_{n=1}^{N} (f(O_k(:,:,i) = 0))$$

where $H \times W$ is the dimension of output feature map $O_k \in \mathbb{R}^{H \times W \times D}$, $f(.) = 1$ if (.) is true, and $f(.) = 0$ if (.) is false.

4) Mean gradient [29]: a mean of gradient from each filter is calculated as filter score. Weak filters with low mean of gradient are pruned.

$$Score_i^{\text{Mean gradient}} = \frac{1}{N} \sum_{n=1}^{N} |mean(G(:,:,i))|$$

where G is the gradient calculated from each filter i.

5) Weighted Activation (WA): a normalized heatmap of absolute significant weighted feature map is calculated as a filter score below. Weak filters with low scores are pruned.

$$H_i^{WA} = norm\left(\left|\alpha_{l,i}^c A^{l,i}\right|\right)$$

$$Score_i^{WA} = \frac{1}{N} \sum_{n=1}^{N} \frac{H_c, H_i^{WA}}{||H_c||||H_i^{WA}||}$$

where H_i^{WA} is the normalized heatmap of absolute significant weighted feature map.

A performance comparison of our proposed filter pruning criteria and previous mentioned techniques is shown in Figure 9. The pruning ratios used in the experiments range from 10% to 90%. It can be shown that the accuracy of all strategies is comparable within the pruning ratio range of 10%-60%. However, as pruning ratio exceeds 60%, we start to see that the accuracy of APoZ, mean gradient, and Weight Activation begins to decline. Especially when the pruning ratio exceeds 90%, our filter pruning score based on loss

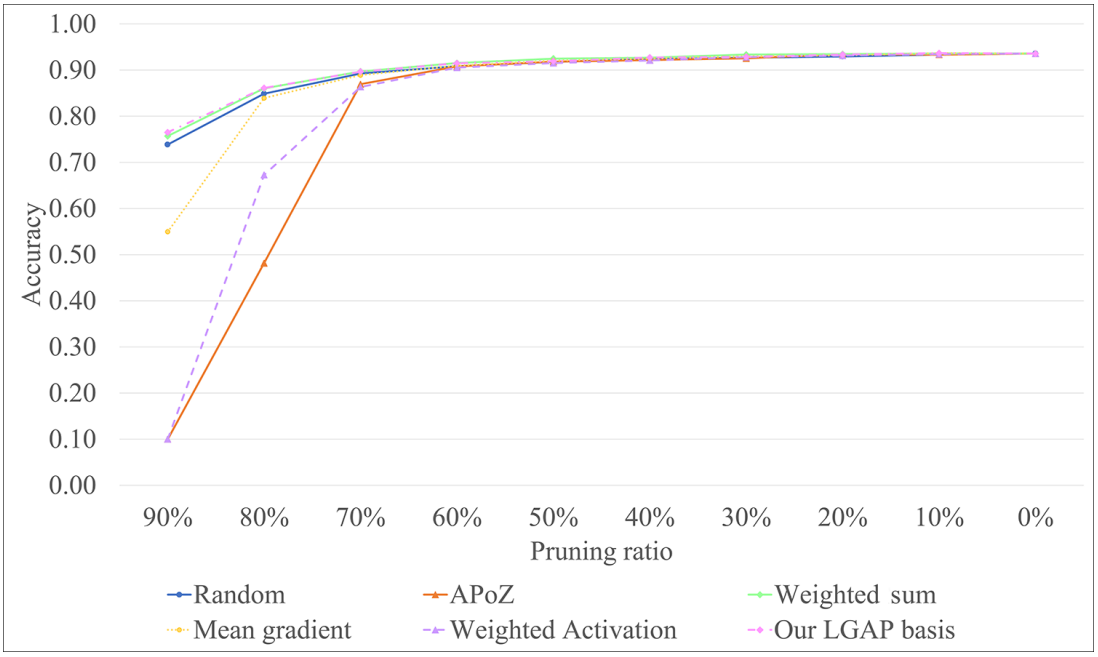


FIGURE 9. A performance comparison of our filter pruning criteria and other baseline filter selection criteria

persistence LGAP can maintain higher accuracy than the baseline techniques of random, weighted sum, APoZ, mean gradient and Weighted Activation by 2.66%, 0.83%, 66.52%, 21.57% and 66.52%, respectively. For the results of filter pruning techniques of VGG-16 on CIFAR-10, we observe that all the models can maintain their accuracies with pruning ratio less than 60%. This means that if we would like to compress the model less than 60%, any of the experimental techniques can provide comparable results. However, when the pruning ratio exceeds 60%, the decline in accuracy starts to straight down for most of the techniques while our technique can still maintain the model performance.

4.2.3. *A performance comparison for VGG-16 on CIFAR-10.* VGG-16, trained on a large-scale ImageNet dataset, provides a stronger model and tends to be well tolerant with filter pruning techniques. We observe that, at 60% pruning ratio, our filter pruning technique obtains a decrease in accuracy by 2.12%, reduction 5.4× in FLOPs and 6.2× in the number of parameters while pruning ratio at 90%, our filter pruning technique outperforms the other techniques: random, weighted sum, APoZ, mean gradient, and Weighted Activation by 2.66%, 0.83%, 66.52%, 21.57% and 66.52%, respectively.

4.2.4. *Model performance of pruned ResNet-50 on ImageNet.* The original input image is shown in Figure 6(a) while Figure 10 illustrates the LGAP estimation of the top class of prediction with respect to an output feature map activation obtained from the pruned ResNet-50 model with different filter pruning ratios compared to the original one. As the pruning ratio increases, local gradient heatmap is gradually changed, and the output activation is lessened and shifted from the gradient result of the original model. Even though it does not affect the classification result in this case, it causes errors in some of the images and the % error starts to increase with filter pruning ratio above 50%.

We illustrate the results of pruning ResNet-50 using our loss persistent LGAP pruning strategy in Table 4, which illustrates the percentage of errors from Top-1 and Top-5 results, the FLOPs, required size of parameters, storage, and prediction time from all convolutional layers in ResNet-50. Prediction times are measured from a set of ImageNet

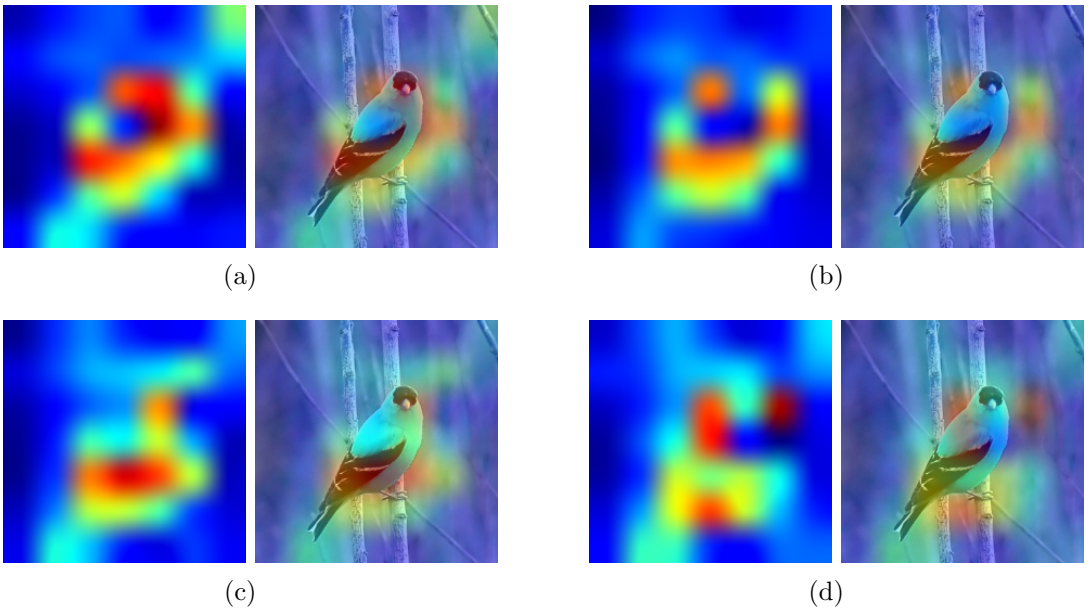


FIGURE 10. Localized Gradient Activation heatmaP (LGAP) of the original and pruned ResNet-50 with different pruning ratios: (a) Original ResNet-50, (b) 30% pruning ratio of ResNet-50, (c) 50% pruning ratio of ResNet-50 and (d) 70% pruning ratio of ResNet-50

TABLE 4. Filter pruning results on ResNet-50

Pruning ratio	Top-1 error (%)	Top-5 error (%)	FLOPs	Parameters	Storage size (MB)	Prediction time (Second)
0%	25.10	7.90	7.7E+09	2.6E+07	98.2	117.35
30%	27.55	8.94	4.8E+09	1.7E+07	65.2	108.30
50%	28.66	9.79	3.4E+09	1.2E+07	47.8	99.33
70%	31.44	11.26	2.2E+09	8.7E+06	33.6	90.21

MB: Megabyte

(50,000 images). At 30% pruning ratio, the model accuracy of the top-1 decreases by 2.45% and top-5 decreases by 1.04%, while FLOP, the number of parameters, the storage size, and the prediction time decrease about 1.6×, 1.5×, 1.5×, and 1.08×, respectively. At 50% pruning ratio, the model accuracy of top-1 and top-5 decreases by 3.56% and 1.89%, respectively. The FLOPs, the number of parameters, the storage size, and the prediction time decrease about 2.3×, 2.1×, 2.1×, and 1.18×, respectively. At 70% pruning ratio, the model accuracy of the top-1 decreases by 6.34% and top-5 decreases by 3.36%. The FLOPs decrease about 3.6×, parameters decrease about 3×, storage decreases about 2.9× and prediction times decrease about 1.3×. It can be seen that with large pruning ratio, FLOPs and parameter of ResNet-50 are greatly reduced; however, prediction times slightly decrease when compared with VGG-16. This may cause by the special structure (Residual block) inside of ResNet-50, which still requires large computation time. Even though the prediction time can be slightly reduced, the hardware resources can be significantly optimized and suitable for devices with limit resources (memory, and processing unit).

A performance comparison of filter pruning strategies for ResNet-50 on ImageNet is listed in Table 5. With 50% pruning ratio, the results show that Top-1 and Top-5 of our techniques outperform all the baseline techniques. Our technique can achieve lower

Top-1 error compared to the baseline techniques: random, SSL, ThiNet, APoZ, weighted sum, mean gradient and SSR-L2 by 1.09%, 1.02%, 0.56%, 0.69%, 0.75%, 0.34% and 0.09%, respectively. Top-5 error of our technique is lower than baseline techniques: random, SSL, ThiNet, APoZ, weighted sum, mean gradient and SSR-L2 by 0.86%, 0.79%, 0.39%, 0.52%, 0.53%, 0.05% and 0.22%, respectively. For image classification experiments in the real world, we examined the performance of the trimmed ResNet-50 on images not included in the ImageNet dataset. We found that pruned models can predict images belonging to their class with comparable accuracy and in less time than the original model.

TABLE 5. A performance comparison of filter pruning on ResNet-50 with pruning ratio 50%

Technique	Top-1 accuracy drop (%)	Top-5 accuracy drop (%)	FLOPs	Parameters	Pruned
Random	4.65	2.75	3.4E+09	1.2E+07	50%
SSL [23]	4.58	2.68	4.2E+09	1.3E+07	$\approx 50\%$
ThiNet [24]	4.12	2.28	3.4E+09	1.2E+07	50%
APoZ [28]	4.25	2.41	3.4E+09	1.2E+07	50%
Weighted sum [27]	4.31	2.42	3.4E+09	1.2E+07	50%
Mean gradient [29]	3.90	1.94	3.4E+09	1.2E+07	50%
SSR-L2 [30]	3.65	2.11	3.4E+09	1.2E+07	50%
Our technique	**3.56**	**1.89**	**3.4E+09**	**1.2E+07**	**50%**

5. **Conclusion.** In this article, we propose a new technique for filter level pruning to prune two CNN models (VGG-16 on CIFAR-10 and ResNet-50 on ImageNet) based on the Localized Gradient Activation heatmaP (LGAP). The LGAP is a layer-wise heatmap of weighted channel activations, where the activation from filters best corresponding to the top-class output will be emphasized. It provides the spatial relationship between the investigated filter and the target prediction. The filter score is then estimated based on the similarity between the heatmap of a complete layer and the loss persistence heatmap. The similarity can be viewed as a persistence score when a filter is removed from the layer. The high filter score shows that removed filters have less effect with layer operation and are considered as weak filters that must be pruned from the layer. Even though the other related techniques adopted data-driven approach, they lacked spatial relationship among filters in each layer and relationship between filters and output prediction due to their concentration on a single-neural analysis. From the experimental results, it can be seen that the performance of the pruned model using our filter pruning strategy outperforms the other filter pruning techniques especially with high compression ratio. However, since the fine-tuning process of a large pre-trained model required a significant amount of time to recover performance after pruning, our future research will concentrate on reducing the time required for fine-tuning performance recovery.

Acknowledgment. This work was supported by King Mongkut's Institute of Technology Ladkrabang, Thailand.

REFERENCES

[1] A. Krizhevsky, I. Sutskever and G. E. Hinton, ImageNet classification with deep convolutional neural networks, *Commun. ACM*, vol.60, no.6, pp.84-90, 2017.

[2] A. F. Siregar and T. Mauritsius, Ulos fabric classification using android-based convolutional neural network, *International Journal of Innovative Computing, Information and Control*, vol.17, no.3, pp.753-766, 2021.
[3] R. Girshick, Fast R-CNN, *Proc. of 2015 IEEE International Conference on Computer Vision (IC CV)*, pp.1440-1448, 2015.
[4] J. Yuan et al., Gated CNN: Integrating multi-scale feature layers for object detection, *Pattern Recognition*, vol.105, 2020.
[5] E. Essa, D. Aldesouky, S. E. Hussein and M. Z. Rashad, Neuro-fuzzy patch-wise R-CNN for multiple sclerosis segmentation, *Medical & Biological Engineering & Computing*, vol.58, no.9, pp.2161-2175, 2020.
[6] R. Girshick, J. Donahue, T. Darrell and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, *Proc. of 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.580-587, 2014.
[7] S. Ren, K. He, R. Girshick and J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, *Proc. of the 28th International Conference on Neural Information Processing Systems*, vol.1, pp.91-99, 2015.
[8] W. Yang, G.-L. Zhou, Z.-W. Gu, X.-D. Jiang and Z.-M. Lu, Safety helmet wearing detection based on an improved YOLOv3 scheme, *International Journal of Innovative Computing, Information and Control*, vol.18, no.3, pp.973-988, 2022.
[9] C. Szegedy et al., Going deeper with convolutions, *Proc. of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1-9, 2015.
[10] F. Chollet, Xception: Deep learning with depthwise separable convolutions, *Proc. of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1800-1807, 2017.
[11] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, Aggregated residual transformations for deep neural networks, *Proc. of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.5987-5995, 2017.
[12] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, Densely connected convolutional networks, *Proc. of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.2261-2269, 2017.
[13] H. Noh, S. Hong and B. Han, Learning deconvolution network for semantic segmentation, *Proc. of 2015 IEEE International Conference on Computer Vision (ICCV)*, pp.1520-1528, 2015.
[14] Y. Wang, J. Liu, Y. Li, J. Fu, M. Xu and H. Lu, Hierarchically supervised deconvolutional network for semantic video segmentation, *Pattern Recognition*, vol.64, pp.437-445, 2017.
[15] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *Proc. of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.770-778, 2016.
[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, Rethinking the inception architecture for computer vision, *Proc. of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.2818-2826, 2016.
[17] R. Pilipović, P. Bulić and V. Risojević, Compression of convolutional neural networks: A short survey, *Proc. of the 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp.1-6, 2018.
[18] D. Ghimire, D. Kil and S.-H. Kim, A survey on efficient convolutional neural networks and hardware acceleration, *Electronics (Switzerland)*, vol.11, no.6, 2022.
[19] U. Lotrič and P. Bulić, Applicability of approximate multipliers in hardware neural networks, *Neurocomputing*, vol.96, pp.57-65, 2012.
[20] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally and K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <1MB model size, *CoRR*, vol.abs/1602.07360, 2016.
[21] Y. LeCun, J. S. Denker and S. A. Solla, Optimal brain damage, *Advances in Neural Information Processing Systems 2*, Denver, CO, USA, pp.598-605, 1989.
[22] S. Han, J. Pool, J. Tran and W. Dally, Learning both weights and connections for efficient neural network, *NIPS'15: Proc. of the 28th International Conference on Neural Information Processing Systems*, pp.1135-1143, 2015.
[23] W. Wen, C. Wu, Y. Wang, Y. Chen and H. Li, Learning structured sparsity in deep neural networks, *NIPS'16: Proc. of the 30th International Conference on Neural Information Processing Systems*, pp.2074-2082, 2016.

[24] J.-H. Luo, J. Wu and W. Lin, ThiNet: A filter level pruning method for deep neural network compression, *Proc. of 2017 IEEE International Conference on Computer Vision (ICCV)*, pp.5068-5076, 2017.
[25] J. Wu, C. Leng, Y. Wang, Q. Hu and J. Cheng, Quantized convolutional neural networks for mobile devices, *Proc. of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.4820-4828, 2016.
[26] V. Lebedev and V. Lempitsky, Fast ConvNets using group-wise brain damage, *Proc. of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.2554-2564, 2016.
[27] H. Li, A. Kadav, I. Durdanovic, H. Samet and H. P. Graf, Pruning filters for efficient ConvNets, *CoRR*, vol.abs/1608.08710, 2017.
[28] H. Hu, R. Peng, Y.-W. Tai and C.-K. Tang, Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, *CoRR*, vol.abs/1607.03250, 2016.
[29] C. Liu and H. Wu, Channel pruning based on mean gradient for accelerating convolutional neural networks, *Signal Processing*, vol.156, pp.84-91, 2019.
[30] S. Lin, R. Ji, Y. Li, C. Deng and X. Li, Toward compact ConvNets via structure-sparsity regularized filter pruning, *IEEE Transactions on Neural Networks and Learning Systems*, vol.31, no.2, pp.574-588, 2020.
[31] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, *Proc. of 2017 IEEE International Conference on Computer Vision (ICCV)*, pp.618-626, 2017.
[32] F. Chollet et al., *Keras*, https://github.com/fchollet/keras, 2015.
[33] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny Images*, Technical Report, University of Toronto, 2009.
[34] O. Russakovsky et al., ImageNet large scale visual recognition challenge, *International Journal of Computer Vision*, vol.115, no.3, pp.211-252, 2015.
[35] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *Proc. of the 3rd International Conference on Learning Representations (ICLR 2015)*, pp.1-14, 2015.
[36] S. Liu and W. Deng, Very deep convolutional neural network based image classification using small training sample size, *Proc. of the 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp.730-734, 2015.
[37] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, *Proc. of the 32nd International Conference on Machine Learning*, pp.448-456, 2015.
[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.*, vol.15, no.1, pp.1929-1958, 2014.

Author Biography

Monthon Intraraprasit received the M.E. degree in Computer Engineering from King Mongkut's Institute of Technology Ladkrabang, Thailand, in 2017. He is currently pursuing the D.Eng. degree with the King Mongkut's Institute of Technology Ladkrabang, Thailand. His research interests include machine learning, deep learning and computer vision.

Orachat Chitsobhuk received the Ph.D. degree in Electrical Engineering from University of Texas, Arlington, US, in 2001. She is currently an associate professor at King Mongkut's Institute of Technology Ladkrabang, Thailand. Her research interests include image and scene analysis, machine learning and pattern recognition.