Deep Network Compression based on Partial Least Squares

Artur Jordao*, Fernando Yamada, William Robson Schwartz

Smart Sense Laboratory, Computer Science Department
Universidade Federal de Minas Gerais, Brazil

Abstract

Modern visual pattern recognition methods are based on convolutional networks since they are able to learn complex patterns directly from the data. However, convolutional networks are computationally expensive in terms of floating point operations (FLOPs), energy consumption and memory requirements, which hinder their deployment on low-power and resource-constrained systems. To address this problem, many works have proposed pruning strategies, which remove neurons (i.e., filters) in convolutional networks to reduce their computational cost. Despite achieving remarkable results, existing pruning approaches are ineffective since the accuracy of the network is degraded. This loss in accuracy is an effect of the criterion used to remove filters, as it may result in the removal of the filters with high influence to the classification ability of the network. Motivated by this, we propose an approach that eliminates filters based on the relationship of their outputs with the class label, on a low-dimensional space. This relationship is captured using Partial Least Squares (PLS), a discriminative feature projection method. Due to the nature of PLS, our method focuses on keeping discriminative filters. As a consequence, we are able to remove up to 60% of FLOPs while improving network accuracy. We show that our criterion is superior to existing pruning criteria, which include state-of-the-art feature selection techniques and handcrafted approaches. Compared to state-of-the-art pruning

*Corresponding author
Email address: arturjordao@dcc.ufmg.br (Artur Jordao)

strategies, our method achieves the best tradeoff between drop/improvement in accuracy and FLOPs reduction.

Keywords: Pruning Convolutional Networks, Convolutional Networks Compression, Partial Least Squares

1. Introduction

Convolutional networks have been an active research topic in Computer Vision mostly because they have achieved state-of-the-art results in numerous tasks [1, 2, 3]. It has been demonstrated that deeper architectures achieve better results, but they are computationally expensive, present a large number of parameters and consume a considerable amount of memory. To handle these problems, there exists many optimization strategies such as depth-wise separable convolution [4], neural architecture search [5, 6, 7], binarization of weights [8, 9], dynamic inference [10, 11] and pruning approaches [12, 13, 14]. Among these strategies, pruning approaches have been widely explored due to their simplicity and notable results.

According to Han et al. [15], a pruning method can be defined as a three-step iterative process: locate potential neurons to be removed, rebuild the network without them and perform fine-tuning at the end of each iteration. In the step of finding neurons to eliminate, different criteria have been proposed, such as removing neurons with small L1-norm [15, 16] or zero activation [17], as well as learning a combination of neurons to be discarded [18]. In the rebuilding step, a new network is built (without the discarded neurons) and the weights of the kept neurons are transfered to it. Finally, in the fine-tuning step, the new network is fine-tuned to compensate for the neurons that have been removed.

Despite being simple and presenting considerable results, modern pruning approaches either require human effort or demand a high computational cost. For instance, the method proposed by Li et al. [16] employs the L1-norm to locate candidate filters to be eliminated. However, it requires considerable human effort to evaluate different tradeoffs between network performance and pruning

rate (percentage of filters removed). Based on this limitation, Huang et al. [18] proposed a pruning approach that removes unnecessary filters by learning pruning agents. These agents take the filter weights from a layer as input and output binary decisions indicating whether a filter will be kept or removed. Even though Huang et al. [18] achieved a performance superior to the hand-crafted pruning criterion by [16], their method demands a higher computational cost because each agent is modeled as a neural network. In addition, when a higher number of filters is eliminated the network accuracy decreases considerably.

Motivated by the limitations in current pruning methods [17, 16, 18], we propose a novel approach to efficiently eliminate filters in convolutional networks. Our method relies on the hypothesis that estimating the filter importance based on its relationship with the class label, on a low-dimensional space, is an adequate strategy to locate potential filters to be eliminated. For this purpose, our method works as follows. First of all, we interpret the output of each convolutional filter as a feature vector (or a set of features), as illustrated in Figure 1. After this stage, we create a high dimensional feature space, representing all convolutional filters of the network at once. Then, we project this high dimensional

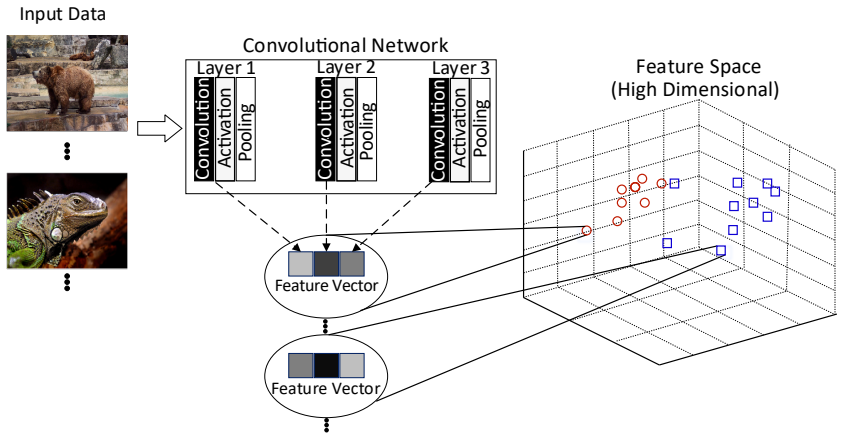


Figure 1: Representation of feature maps (i.e., filters) as feature vectors. For simplicity, the network shows only one filter (one dimension of the feature space) in each layer. Red and blue points denote positive and negative samples, respectively (best viewed in color).

space onto a latent space using Partial Least Squares (PLS), a discriminative feature projection method [19, 20, 21]. Next, we employ Variable Importance in Projection (VIP) to estimate the contribution of each feature in generating the latent space, enabling PLS to operate as a feature selection method. The idea behind this process is that, since the output of a filter is represented as a feature, we are estimating the filter importance based on its relationship with the class label on the latent space (PLS criterion). Finally, we eliminate filters with low importance (least discriminative) and repeat this process until a specific number of iterations is reached.

We show that our criterion is superior to existing pruning criteria, which include state-of-the-art feature selection techniques and hand-crafted approaches, since it focuses on keeping discriminative filters. Compared to state-of-the-art pruning strategies, our method achieves the highest FLOPs reduction and the smallest drop in accuracy.

We evaluate our method by pruning the VGG16 [22] and ResNet [23] architectures on the ImageNet [24] and CIFAR-10 [25] datasets, where we are able to reduce up to 60% of the floating point operations (FLOPs) while improving network accuracy.

2. Related Work

The idea behind pruning networks is to find neurons that could be removed without degrading its original performance. Several works have proposed different approaches to identify the neurons to be removed. In particular, most existing pruning approaches focus on eliminating neurons (filters) in convolutional layers, since 99% of overall FLOPs are concentrated on these layers [15].

To find unimportant filters, Hu et al. [17] proposed to observe the percentage of zeros activations (APoZ) in the resulting feature map from a filter (coupled with ReLU activation). The idea behind the method is that filters which yield feature maps with large APoZ are redundant and could be removed. In their method [17], once APoZ from all filters have been computed, the ones larger

than one standard deviation are eliminated and the resulting network is fine-tuned. Hu et al. [17] demonstrated that measuring the APoZ considering all layers, decreases network accuracy considerably and it is difficult to recover its original performance, therefore, their method is executed layer-by-layer. Similar to [17], Li et al. [16] proposed to discard filters based on its magnitude. For this purpose, the authors compute the L1-norm from each filter and remove the filters with small magnitudes. Since filters coming from different layers present different magnitude orders, their L1-norms cannot be compared directly to decide which filters to remove. Hence, the approach of Li et al. [16] prunes the network considering one layer at the time. A notable drawback in their work is the requirement of human efforts to test different tradeoffs between the pruning rate and the network performance. The work of Mittal et al. [26] suggests that different pruning criteria (e.g., APoZ, L1-norm and random pruning) can lead to similar performance due to the plasticity of deep networks. Plasticity refers to the capacity of the network to recover its accuracy after pruning. While their work provides useful insights about pruning, they consider only simple and few discriminative pruning criteria, which, de-facto, can lead to similar results. Compared to these pruning criteria as well as more elaborate criteria, we show that PLS+VIP attains superior performance since it focuses on keeping filters that preserve the classification ability of the network (high relationship with the class label).

Instead of designing hand-crafted criteria as mentioned above, another scheme for pruning is to learn which filters should be removed based on some policy. For example, Huang et al. [18] proposed to eliminate redundant filters by learning agents, which are encouraged to remove a large number of filters while preserving the network performance above a threshold. According to results presented by Huang et al. [18], for each agent (which is responsible for pruning a single layer) around 150 training iterations are necessary for convergence. In addition, after pruning a layer, some stages of fine-tuning are performed on the entire network. As a consequence, their method is unfeasible for deep networks. Similar to Huang et al. [18], He et al. [27] proposed a pruning strategy based on

agents that learn to penalize accuracy loss and encourage model compression. These agents can employ different policy search protocols: (i) latency, where resulting network has the best accuracy given an amount of hardware resources and (ii) quality, where the output network is the smallest as possible with no loss of accuracy. We show that our method is able to achieve both criteria and is also more accurate.

To address the problem of eliminating neurons layer-by-layer, Yu et al. [28] proposed to remove the filters coming from all layers at once. To this end, the authors applied the Infinity Feature Selection [29] method on the final layer output to obtain the importance of each neuron. Then, this importance is recursively propagated throughout the network using a proposed neuron score propagation algorithm. Finally, neurons with low importance are removed and the network is fine-tuned. He et al. [14] suggested removing filters from different layers, simultaneously, in the training step. Their method sets the same pruning ratio to all layers and, after each training epoch, removes the filters based on the L2-norm. He et al. [30] observed that simply removing filters based on their norms is not suitable, instead, filters with redundant information should be eliminated. For this purpose, the authors employed the geometric median to find candidate filters that could be replaced by others. To identify unimportant filters, Liu et al. [31] associated each filter to an agent named scale-factor. These scale-factors are learned jointly with the network weights and, after training, the filters associated with scale-factors near zero are removed and the network is fine-tuned. Huang et al. [32] extended the scale-factors to remove different components from a convolutional network such as filters, layers and groups. In contrast to most existing pruning approaches, due to our discriminative pruning criterion, we achieve competitive FLOPs reduction with the smallest drop in accuracy.

As presented above, fine-tuning is a predominant step in pruning approaches, even when the network is pruned on training [31]. Alternatively, Frankle et al. [33] argued that the pruned network can recover the accuracy (compared to its unpruned counterpart) by training it from scratch with the original initial-

ization (weights) of the kept filters. Liu et al. [34] demonstrated that by setting an optimal learning rate and randomly re-initializing the weights, it is possible to achieve even better results. An interesting aspect of these works [33, 34] is that network pruning can be explored as a neural architecture search, where the search space is a restricted subset of all possible architectures. We show that training the pruned network from scratch (as suggested by Liu et al. [34]) does not bring notable improvements to our method.

3. Proposed Approach

This section defines the proposed method to eliminate filters in deep convolutional networks. We start by describing the representation of convolutional filters as feature vectors. Then, we explain the Partial Least Squares and Variable Importance in Projection techniques, which project the filter representation onto a low dimensional space and measure the filter importance, respectively. Finally, we describe how to remove filters with low importance. An overview of the proposed method is presented in Figure 2.

Filter Representation. The first step in our method is to represent the output of the filters (i.e., its feature maps) that compose the network as feature vectors. For this purpose, let us consider we have m data samples (i.e., the training samples), which are forwarded on the network to obtain the feature maps provided by each convolutional filter. Since these feature maps are high dimensional, we apply a pooling operation (i.e., max-pooling) to reduce their dimension. In this work, we consider the following pooling operations: global max-pooling, global average pooling and max-pooling 2×2. Then, the output

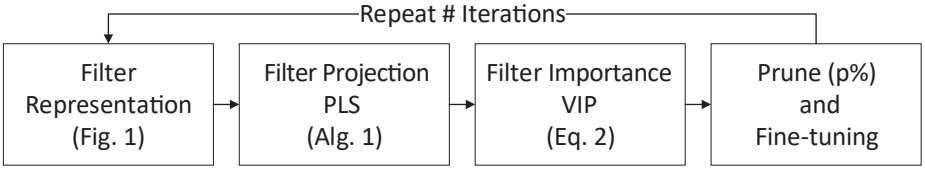


Figure 2: Steps of the proposed method to prune convolutional filters from deep networks.

of the pooling operation is interpreted directly as one feature (when using the global pooling operations) or a set of features (when using the max-pooling 2×2). Specifically, each filter is represented by its feature map followed by the pooling operation. Finally, the filter representations from different layers are concatenated to compose the final feature vector that represents all filters of the network. Figure 1 illustrates this process.

The intuition for using the feature map as a feature is that we are able to measure its relationship with the class label on the latent space (PLS criterion). In this way, a filter associated with a feature with low relationship might be removed.

Feature Projection. After executing the previous step, we have generated a high dimensional space R^d that represents all filters of the convolutional network. The second step of our method is to project this high dimensional space onto a low dimensional space R^c ($c \ll d$), referred to as latent space. To this end, we employ Partial Least Squares (PLS), a discriminative feature projection method widely employed to model the relationship between dependent and independent variables. PLS works as follows. Let $X \subset R^{m \times d}$ and $y \subset R^{m \times k}$ be a matrix of independent and dependent variables, respectively. In our method, the matrix X is the representation of the filters we have generated (first step of the proposed method) and y is the class label matrix, where k denotes the number of categories.

Partial Least Squares estimates a projection matrix W $(w_1, w_2, ...w_c)$ that projects the high dimensional space R^d onto a low dimensional space R^c (c is a parameter) such that each component $w_i \in W$ represents the maximum covariance between the X and y, as shown in Equation 1.

$$w_i = argmax(Cov(Xw, y)), \mathtt{s.t} \|w\| = 1. \tag{1}$$

To solve Equation 1, we can use Nonlinear Iterative Partial Least Squares (NIPALS) [21] or SVD. In this work, we use the NIPALS algorithm since it is faster than SVD. In addition, it allows us to find only the first c components, while SVD finds all d components, spending more computational resources. Algo-

Algorithm 1: NIPALS Algorithm.

Input : $X \subset R^{m \times d}$, $y \subset R^{m \times k}$

Input : Number of components c

Ouput: Projection matrix $W \subset R^{d \times c}$

for $i = 1$ ***to*** c **do**

- randomly initialize $u \in \mathbb{R}^{m \times 1}$
- $w_i = \frac{X^T u}{\|X^T u\|}$, where $w_i \in W$
- $t_i = X w_i$
- $q_i = \frac{y^T t_i}{\|y^T t_i\|}$
- $u = y q_i$
- Repeat steps $3 - 6$ until convergence
- $p_i = X^T t_i$
- $X = X - t_i p_i^T$
- $y = y - t_i q_i^T$

end

rithm 1 introduces the steps of NIPALS to obtain the first c components, where the convergence step is achieved when no changes occur in w_i. In addition, we might define a finite number of steps (e.g., 2) as convergence criterion, to ensure that the method stops.

It should be noted that, in this step of our method, other feature projection methods could be employed, for instance, PCA or linear discriminant analysis (LDA). However, we believe that the idea behind PLS, which is to capture the relationship between the feature (in our context a filter) and its class label, is more suitable. In particular, when compared to LDA, PLS is robust to sample size problem (singularly) [35]. Moreover, as shown in our experiments, PLS can be learned using few samples, not requiring all the data to be available in advance. These advantages make PLS more flexible and efficient than traditional

feature projection methods, mainly for large datasets.

Filter Importance. The next step in our method is to measure the filter importance score to remove the ones with low importance. To this end, once we have found the projection matrix W using Algorithm 1, we estimate the importance of each filter based on its contribution to yield the latent space. Recall that, following the modeling performed in the first step of our method, a feature (or a set of features when using the max-pooling) corresponds to a filter.

To achieve the aforementioned goal, we employ the Variable Importance in Projection (VIP) technique [36], where the importance score of a filter, f_j, can be computed by

$$f_j = \sqrt{d \sum_{i=1}^{c} SS_i (w_{ij}/\|w_i\|^2) / \sum_{i=1}^{c} SS_i}, \tag{2}$$

where d is the number of features and SS_i is the sum of squares explained by the i-th component, which is expressed as $q_i^2 t_i' t_i$ (defined in Algorithm 1) [36]. Note that when using the max-pooling operation as filter representation, we have a set of features for each filter; therefore, the final score to a filter on this representation is the average of its f_j.

Following our modeling, the filter importance is given by the linear relationship between the filters and the class label. As argued by previous works [37, 38, 39], the linear relationship between deep features and their labels provides surprising results. This suggests that linear models (i.e., PLS) are good candidates to be employed in this context.

Prune and Fine-tuning. Given the importance of all filters that compose the network, we have generated a set of scores, $F(f_1, f_2, ..., f_j)$. Then, given a pruning ratio p (e.g., 10%), we remove p% of the filters based on its scores. The removal stage consists of creating a new network, without the discarded filters, and transferring the weights of the kept filters [16, 18]. An alternative to fine-tuning is training the pruned network from scratch. As argued by Frankle et al. [33] and Liu [34], this process leads to efficient networks. However, we show that training the pruned network from scratch (as suggested by Liu [34])

does not bring notable improvements to our method.

By executing all the above steps, we have executed one iteration of the proposed method, as illustrated in Figure 2. It is important to note that the input network to the next iteration is the pruned network of the previous iteration.

4. Experimental Results

In this section, we first introduce the experimental setup, the experiments regarding the parameters of our method and how to employ it on large datasets. Then, we show the advantages of executing the method iteratively and the importance of representing all filters of the network at once. Finally, we compare our method with existing pruning criteria and approaches, and present qualitative results.

Experimental setup. We conduct experiments using a single NVIDIA GTX 1080 on a machine with 64GB of RAM. Following previous works [16, 18], we examine some aspects and parameters of our method by considering VGG16 only on CIFAR-10 and discuss the results using drop in accuracy in percentage points (p.p.), where negative values denote improvement w.r.t the original network. We report the drop in accuracy using Top-1 and Top-5 accuracy on CIFAR-10 and ImageNet, respectively. It is important to mention that for some experiments we consider only the 32×32 version of ImageNet due to the high computational cost of fine-tuning in the original version (224×224). In addition, as suggested by Loshchilov and Hutter [40], this low-resolution version of ImageNet is more challenging than the original version, enabling us to validate the methods on a more difficult scenario. Finally, we compute FLOPs following the work of Li et al. [16] and set the pruning rate of 10% in all experiments.

Number of components. As explained in Section 3, the only parameter of PLS is the number of components, c. To estimate its best value, we create a set of possible c by varying it from 1 to 15 (step of 1) and evaluate the network accuracy for each value after executing one iteration of our method. To measure the network accuracy, we use a validation set with 10% of the training data. By

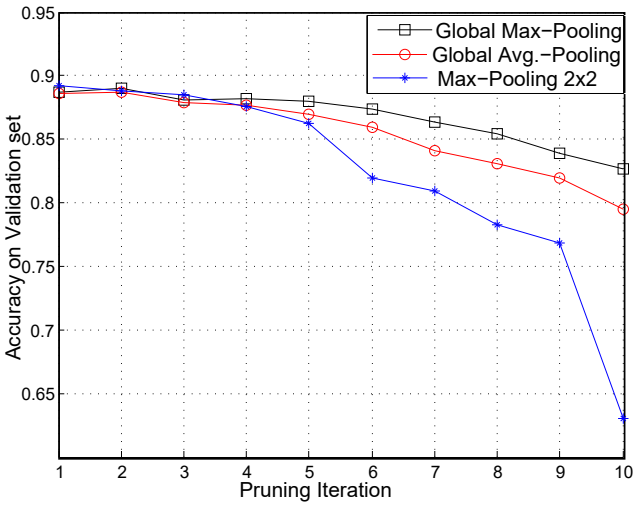
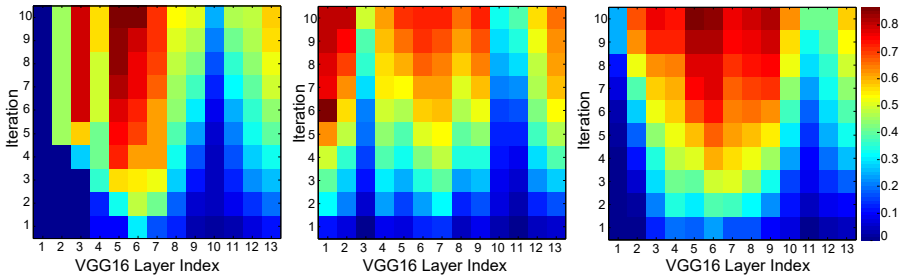


Figure 3: Accuracy obtained by pruning VGG16 on CIFAR-10 dataset (validation set) using different filter representations.

using this process, we found that the value that resulted in the smallest drop in accuracy was $c = 2$, thus it was used in the remaining experiments (including the pruning on ResNet).

Influence of the filter representation. One of the most important issues in our method is the pooling operation, referred to as filter representation, employed on the feature map provided by a filter. This experiment aims at validating this issue. For this purpose, we execute ten pruning iterations using different pooling operations. As illustrated in Figure 3, accuracy decreases slower when global max-pooling is employed. On the contrary, by using the max-pooling 2×2 accuracy drops faster, where at the 10th iteration the method drops 26 p.p. compared to the unpruned network. In addition, this representation has the drawback of consuming additional memory compared to the global operations (global and average), which reduce the feature map to one dimension.

Besides playing an important role in the pruning performance, the filter representation has an interesting aspect regarding scores assigned by VIP. Note that, to select a filter to be removed means that VIP assigned a low score to it, indicating that it is unimportant to explain the class label. In particular, by



(a) Max-pooling 2×2. (b) Global Avg. pooling. (c) Global Max pooling.

Figure 4: Heat map of the relation between the number of filters removed, by layer, and the iteration of the proposed method using different filters representations. Warmer regions indicate that more filters were removed (best viewed in color).

modifying the filter representation, we drastically alter the selection of filters to be removed. Figure 4 illustrates this idea, where we show the relation between the pruning iteration and the number of removed filters per layer. According to Figure 4, the max-pooling representation eliminates a larger number of filters from layers 3 to 7, while the global average pooling has a similar distribution of the VIP scores, since it removes filters from all layers uniformly (except for layers 3, 10 and 11). On the other hand, the global max-pooling representation removes a larger number of filters from layers 3 to 9 and keeps the filters from layers 1 and 2. Finally, VIP assigns high scores for the filters from the layers 10 and 11. Based on the results, we used the global max-pooling as filter representation in the remaining experiments.

According to Li et al. [16], filters from the first layer should not be pruned since their removal degrades network performance significantly. According to Figure 4, our method is able to identify this because either it does not remove or it removes few filters from this layer, which indicates the suitability of PLS to identify the relevant filters. It is important to mention that in the work of Li et al. [16], the conclusion that the filters from the first layer are important was done by a human analyzing the accuracy drop when removing these filters. However, this is performed automatically in our work.

Another interesting aspect concerning VIP distributions is that the linear projection of PLS explains well the relationship between filters and the class label. This is because if there were a strong non-linear relationship, VIP would always assign a low score to some filters since they would not be explained by PLS. Thereby, filters from specific layers would always be removed. As shown in Figure 4, however, this behavior did not occur, since filters from different layers were removed.

Number of samples to learn the PLS. A basic requirement in deep learning approaches is a large number of training samples to avoid overfitting and provide a good generalization. Based on this statement, large datasets have been proposed, e.g., ImageNet [24] and VGGFaces [41] with 1.2 and 3.31 million images, respectively. On these datasets, our method could be impracticable due to memory constraints since NIPALS requires the training samples (X in Algorithm 1) be in memory. However, an advantage of PLS is that it can be learned with a very small number of samples. Therefore, we can subsample X before executing Algorithm 1, enabling our method operate on large datasets.

In this experiment, we intend to demonstrate that the proposed method is robust when fewer samples are used to learn the PLS projection. To this end, we vary the percentage of training samples (using a uniform subsampling) used to compose X in Algorithm 1. Table 1 shows the results obtained after one pruning iteration, where it is possible to observe that the network accuracy is slightly changed as a function of the number of samples used to learn the PLS. In particular, sometimes, the accuracy is the same as employing 100% of the samples (e.g., using 20% or 60%). In addition, the difference between using 100% and 10% of the samples is only 0.1 p.p.. Therefore, to conduct the

Table 1: Accuracy by pruning VGG16 on CIFAR-10 (validation set), using different number of samples to learn the PLS.

% Training Samples	10	20	40	60	80	100
Accuracy after pruning	89.7	89.8	89.7	89.8	89.6	89.8

Table 2: Drop in accuracy when executing our method with few iterations and a low pruning ratio (Iterative Pruning), and when executing a single iteration with a high pruning ratio (Single Pruning). Results on CIFAR-10 (test set). Negative values denote improvement regarding the original network.

Removed (%) Filters	Iterative Pruning Drop in Acc.↓	Single Pruning Drop in Acc.↓
10	−0.89 (it=1)	−0.89
27	−1.08 (it=3)	−0.03
40	−0.69 (it=5)	1.76
65	1.56 (it=10)	20.21

experiments on ImageNet, we used only 10% of the samples.

Iterative pruning vs. single pruning. In this experiment, we show that it is more appropriate to execute our method iteratively, as illustrated in Figure 2, with a low pruning ratio (i.e., 10%) instead of using a single pruning iteration with a high pruning ratio. In other words, if we intend to remove i.e. 40% of filters, it is better to execute some iterations of our method with a low pruning ratio instead of setting a pruning ratio of 40% and execute only a single iteration. To this end, we first execute five iterations of the proposed method with a pruning ratio of 10%. Then, after each iteration, we compute the percentage of removed filters, p_i. Finally, we use each p_i as the pruning ratio to execute a single iteration of the method. According to the results shown in Table 2, performing our method iteratively with a low pruning ratio is more effective than using it with a large pruning ratio, which led to a higher drop in accuracy. For instance, by executing five iterations of the method with a pruning ratio of 10%, we are able to remove 40% of filters while improving the network accuracy (indicated by negative values in Table 2). On the other hand, by applying a single iteration with a pruning ratio of 40%, the accuracy decreased 1.76 p.p..

Multiple projections vs. single projection. This experiment shows the performance of our method when using the filters layer-by-layer and all filters at once to learn PLS. While the former has a PLS model associated with each

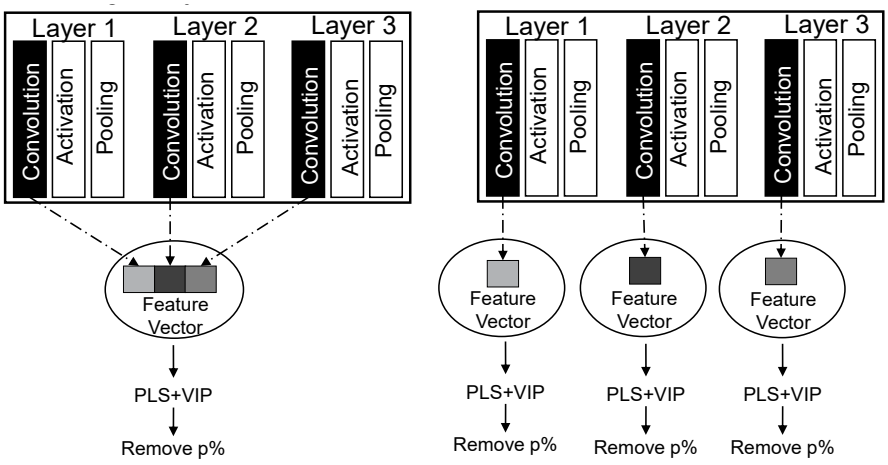


Figure 5: (a) Single projection scheme. In this strategy, a single PLS model is learned considering all filters that compose the network at once. (b) Multiple projections scheme. In this strategy, one PLS model is learned considering filters layer-by-layer.

layer, the latter has only one PLS model, as shows Figure 5.

Table 3 (last two rows) presents the results of the proposed method when using single and multiple projections, called PLS(Single)+VIP and PLS(Multi)+VIP, respectively. On the CIFAR-10 dataset, PLS(Multi)+VIP and PLS(Single)+VIP achieved similar performance, where PLS(Multi)+VIP obtained a drop in accuracy 0.08 p.p. better than PLS(Single)+VIP. On the ImageNet dataset, on the other hand, PLS(Multi)+VIP attained a drop in accuracy 1.90 p.p. worse than PLS(Single)+VIP.

These results indicate that learning PLS projection considering all filters, at once, is more suitable for pruning. According to Li et al. [16], filters coming from different layers degrade network performance in different ways. Therefore, removing $p\%$ of filters of each layer (as is done in PLS(Multi)+VIP) is more critical than removing $p\%$ of all filters[1]. This remark reinforces the usage of PLS(Single)+VIP.

[1]By removing $p\%$ considering all filters is different from removing $p\%$ of filters per layer.

Table 3: Drop in accuracy using different criteria for determining the filter importance. PLS(Multi)+VIP indicates our method projecting the filters layer-by-layer. PLS(Single)+VIP indicates our method projecting all the filters that compose the network, at once. Negative values denote improvement regarding the original network.

Filter Importance Criterion	CIFAR-10 Drop in Acc.↓	ImageNet (32x32) Drop in Acc.↓
L1Norm	−0.69	8.52
APoZ	−0.70	8.06
Infinity FS [29]	−0.69	8.05
Infinity Latent FS [42]	−0.65	7.80
PLS(Multi)+VIP (Ours)	**−0.97**	**8.27**
PLS(Single)+VIP (Ours)	**−0.89**	**6.36**

Based on this result, we used PLS(Single)+VIP (hereafter referred to as PLS+VIP) in the remaining experiments.

Fine-tuning vs. training from scratch. As we mentioned in Section 2, an increasing number of works have proposed training the pruned architecture from scratch instead of performing fine-tuning [33, 34]. In this experiment, we demonstrate the effect of these strategies on the proposed method. To this end, we follow the training procedures scratch-E and scratch-B suggested by Liu et al. [34]. Scratch-E consists of training the pruned network on the same number of epochs as the fine-tuning. Scratch-B consists of training the pruned network using more epochs than fine-tuning. The rationale behind scratch-B is that since the pruned network requires less computational cost than the original model, we can increase the number of training epochs while maintaining, nearly, the same computational budget.

The ResNet56 architecture pruned using our method with one iteration achieves a drop in accuracy of 0.07 and −0.32, on scratch-E and scratch-B, respectively. By using fine-tuning, on the other hand, the pruned network achieves a drop in accuracy of −0.60 p.p.. These results indicate that, to the proposed method, it is more suitable to perform the traditional fine-tuning than training

from scratch. We also considered pruning using more iterations and found the same trends.

Based on these results and in the interest of fairness to most existing pruning approaches, we used the fine-tuning in the remaining experiments.

Comparison with other pruning criteria. In this experiment, we demonstrate that the criterion employed by our method is more effective to eliminate filters than existing pruning criteria as well as state-of-the-art feature selection techniques. To this end, we use one iteration of pruning and follow the process suggested by Yu et al. [28], which consists of setting the same pruning ratio (10%) and modifying only the criterion for locating the filters to be removed.

By employing one pruning iteration, we are able to show the robustness of the methods on a single stage of fine-tuning. It is important to remember that, for each iteration of our method, we execute a single stage of fine-tuning, thereby, the number of iterations defines the number of fine-tuning stages. Finally, as input to the methods of feature selection (Infinity FS [29] and Infinity Latent FS [42]), we use the global max-pooling[2] filter representation.

Table 3 shows the results obtained by different pruning criteria on the CIFAR-10 and ImageNet datasets. According to the results, our criterion for defining the filter importance is more suitable than L1-norm and APoZ, where we achieve the lower drop in accuracy. In addition, PLS+VIP achieved superior performance when compared to methods designed specifically for feature selection [29, 42]. While the results on CIFAR-10 are similar, our method achieves a notable difference on ImageNet, where we obtained a drop in accuracy of 1.43 p.p. better than the best method [42]. The reason for these results is that PLS preserves filters with high relationship with the class label, which are the most important to the classification ability of the network.

Comparison with existing pruning approaches. This experiment compares the proposed method with state-of-the-art pruning approaches. For this

[2]This representation was the one where the methods achieved the best results.

Table 4: Comparison of existing pruning methods on CIFAR-10. Results reported by the original papers. Drop in Acc.↓ denotes drop in accuracy (in percentage points), where negative values denote improvement regarding the original, unpruned, network. FLOPs ↓ denotes the percentage of FLOPs reduced (the higher the better) w.r.t the original network.

	Method	FLOPs↓	Drop in Acc.↓
VGG16	Hu et al. [17]	28.29	−0.66
	Li et al. [16]	34.00	−0.10
	Liu et al. [31]	50.94	−0.13
	Huang et al. [18]	64.70	1.90
	Liu et al. [31](it=5)	95.70	3.35
	Ours (it=1)	23.13	−0.89
	Ours (it=5)	67.25	−0.63
	Ours (it=10)	90.66	1.50
ResNet56	Li (A) [28]	10.40	−0.06
	Li (B) [28]	27.60	−0.02
	Yu et al. [28]	43.61	0.03
	He et al. [27]	50.00	0.90
	He et al. [30]	52.60	0.10
	Ours(it=1)	7.09	−0.60
	Ours(it=5)	35.23	−0.90
	Ours(it=9)	57.06	−0.68
ResNet110	Li (A) [28]	15.90	0.02
	Li (B) [28]	38.60	0.23
	He et al. [14]	40.80	0.30
	Yu et al.[28]	43.78	0.18
	He et al. [30]	52.30	−0.17
	Ours(it=1)	6.85	−0.59
	Ours(it=5)	33.16	−1.51
	Ours(it=10)	60.17	−0.93

purpose, we report the results using one and five iterations of our method as well as the iteration where it achieved the closest drop in accuracy compared to the best method. We highlight that the results of previous methods were taken from their original papers, except for ImageNet 32×32, where we re-implemented the methods. Tables 4, 5 and 6 summarize the results on the CIFAR-10, ImageNet 224×224 and ImageNet 32×32 datasets, respectively.

On the CIFAR-10 dataset, Table 4, our method achieved the best tradeoff between the drop/improvement in accuracy and FLOPs reduction. When compared to Hu et al. [17] and Li et al.[16], we achieved around $2\times$ more FLOPs reduction with superior improvement in accuracy on both networks. Compared to Huang et al. [18], our method decreased $1.5\times$ more FLOPs with a smaller drop in accuracy on VGG16. Compared to the iterative version of the pruning approach by Liu et al. [31], our method removed $0.94\times$ fewer FLOPs, however, we obtained a lower drop in accuracy. In addition, by pruning ResNet56 and ResNet110, our method achieved a higher FLOPs reduction than the most recent pruning approaches [28, 14, 30]. Specifically, when compared to the best pruning approach [30], our method removed 4.46 p.p. and 7.78 p.p. more FLOPs while achieving a greater improvement in network accuracy. We also compared the proposed method with Li et al. [16] following the procedure suggested by [28], which consists of employing a pruning ratio of 15% and 25% (Li et al. (A)-(B)) on each layer of the ResNet architectures. By performing five iterations of the proposed method, we outperformed Li (A) and (B) on both FLOPs reduction and accuracy improvement.

An interesting aspect of these results is that our method achieves superior FLOPs reduction than Huang et al. [18] and He et al. [27], which are cost-aware approaches. The reason for this result is that our method has a smaller drop in accuracy, thus, we can further prune the network. We highlight that the approaches by He et al. [14, 30] also achieve a higher FLOPs reduction compared to cost-aware approaches due to the same trends.

By pruning VGG16 on the ImageNet (224×224 version) dataset, Table 5, with only three iterations of the proposed method, we were able to achieve

the smallest drop in accuracy and 1.80× more FLOPs reduction than all other methods. Furthermore, with two additional iterations, we decreased by 3× more FLOPs than all other methods. On the ResNet50 architecture, compared to most existing pruning approaches our method obtained the best tradeoff between accuracy drop and FLOPs reduction. In particular, by pruning ResNet50, the work of He et al. outperformed the proposed approach. The reason for this result is that our approach, when pruning this architecture on ImageNet, removed few filters from 3×3 convolutions layers within the bottleneck building block, which are the ones with the higher number of FLOPs.

Similarly to the original ImageNet (224×224) dataset, on the 32×32 version

Table 5: Comparison of existing pruning methods on ImageNet 224×224. Results reported by the original papers. Drop in Acc.↓ denotes drop in accuracy (in percentage points), where negative values denote improvement regarding the original, unpruned, network. FLOPs ↓ denotes the percentage of FLOPs reduced (the higher the better) w.r.t the original network.

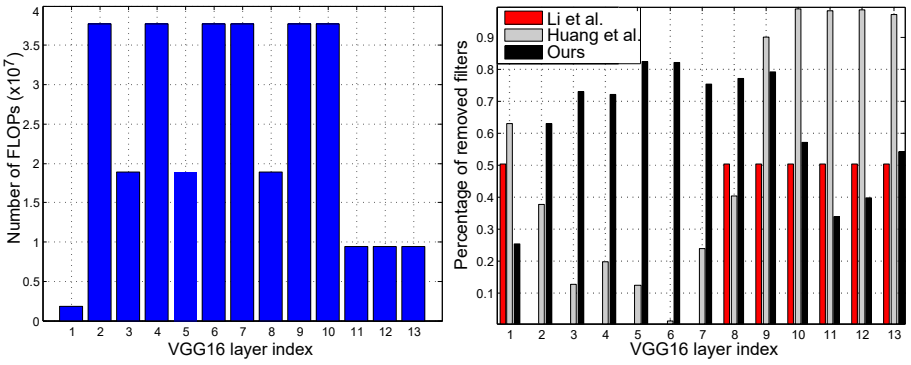
	Method	FLOPs↓	Drop in Acc.↓
VGG16	Li et al. [16]	20.00	14.60
	Hu et al. [17]	19.69	0.84
	He et al. [12]	20.00	1.7
	Wang et al. [43]	20.00	2.00
	He et al. [27]	20.00	1.40
	Ours(it=1)	9.31	−0.98
	Ours(it=3)	36.03	1.06
	Ours(it=5)	59.27	2.21
ResNet50	Luo et al. [13]	36.70	0.47
	Liu et al. [34]	36.70	1.01
	He et al. [14]	41.80	8.27
	He et al. [30]	53.50	0.55
	Ours(it=1)	6.13	−1.92
	Ours(it=5)	27.45	−0.31
	Ours(it=10)	44.50	1.01

Table 6: Comparison of existing pruning methods on ImageNet 32×32. Results reported by the original papers. Drop in Acc.↓ denotes drop in accuracy (in percentage points), where negative values denote improvement regarding the original, unpruned, network. FLOPs ↓ denotes the percentage of FLOPs reduced (the higher the better) w.r.t the original network.

	Method	FLOPs↓	Drop in Acc.↓
VGG16	Hu et al. [17]	34.18	13.35
	Li et al. [16]	21.77	9.11
	Ours(it=1)	20.17	6.36
	Ours(it=2)	31.64	8.26
	Ours(it=3)	41.54	10.89
ResNet56	Li (A)	14.47	3.80
	Li (B)	24.84	4.48
	Ours(it=1)	9.27	3.82
	Ours(it=2)	18.18	3.97
	Ours(it=3)	27.87	3.26

(Table 6), our method attained the highest FLOPs reduction and the smallest drop in accuracy. Observe that, when evaluated on the original ImageNet dataset, the drop in accuracy of the pruning methods is small. In contrast, on the 32×32, version the drop is notable. This behavior supports the employment of this version, where we can evaluate the methods on a more difficult scenario. We emphasize that the single difference between these versions of ImageNet is the image size.

Based on the aforementioned discussion, we have shown that the proposed method achieves a superior reduction in FLOPs. This is an effect of the layers where it removes the filters. According to Figure 6 (a), the layers 2, 4, 6, 7, 9 and 10 have the higher number of FLOPs. In general, the existing methods fail to eliminate filters from these layers. For instance, the methods proposed by Li et al. [16] and Huang et al. [18] remove a large number of filters from the layers 9 to 13 (Figure 6 (b)), but they remove a small number of filters from other layers. Our method, however, eliminates a large number of filters from all layers,



(a) (b)

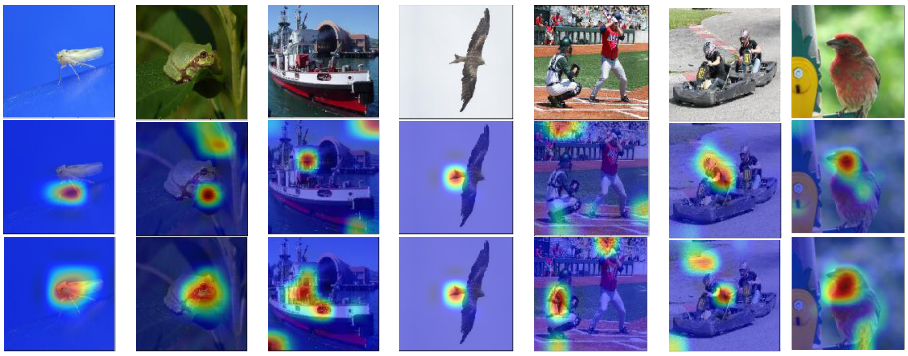
Figure 6: (a) Number of floating point operations (FLOPs) per layer of the VGG16 network. (b) Percentage of removed filters in each layer using different pruning methods (best viewed in color). Values computed from the VGG16 network using on the CIFAR-10 dataset.

as shown in Figure 6 (b). In particular, we eliminate more than 50% of filters from layers 2 to 10, which are the ones with the large number of FLOPs, and more than 25% from the other layers. Hence, we are able to achieve a higher FLOPs reduction than existing state-of-the-art methods, which are biased in eliminating filters of particular layers.

Besides removing more FLOPs, our method is more efficient in terms of the number of fine-tuning steps necessary. For instance, the methods of Hu et al. [17] and Huang et al. [18] require 16 stages of fine-tuning to prune VGG16. On the other hand, our method achieves better results with only around five fine-tuning stages. Similarly to our method, the approaches of Yu et al [28] and He et al. [27] demand few fine-tuning stages, however, we achieve a better tradeoff between FLOPs reduction and accuracy drop, as showed in Table 4.

Qualitative Results. Our last experiment shows that the regions in the image which are important to predict the class label are preserved after pruning a network with the our method.

Figure 7 shows the attention maps of the VGG16 network on images from the ImageNet dataset. It is possible to note that our method preserves the important regions (warmer regions), which are the ones where the object is located. In



(a) (b) (c) (d) (e) (f) (g)

Figure 7: Attention maps of the VGG16 network. From top to down. Input images; Attention maps from the original network; Attention maps from the pruned network.

addition, sometimes, our method locates class-discriminative regions better than the original network, e.g., Figure 7 (a)-(c). This is an effect of PLS+VIP, which focuses on keeping filters with high relationship with the class label.

5. Conclusions

This work presented an accurate pruning method to remove filters from convolutional networks. The proposed method interprets each filter as a feature vector and creates a high dimensional space using these features. Then, it projects this space onto a low-dimensional latent space, using Partial Least Squares, which captures the linear relationship between the feature (filter) and its class label. Finally, the method estimates the importance of each feature to yield the latent space and removes the ones with low importance (low relationship with the class label). We show that removing filters based on its criterion is more suitable than other pruning criteria, where we are able to remove up to 60% of FLOPs while improving network accuracy. Compared to state-of-the-art pruning methods, our method is extremely effective, as it attains the best tradeoff between drop/improvement in accuracy and FLOPs reduction.

Acknowledgments

The authors would like to thank the National Council for Scientific and Technological Development – CNPq (Grants 438629/2018-3, 309953/2019-7 and 140082/2017-4), the Minas Gerais Research Foundation – FAPEMIG (Grants APQ-00567-14 and PPM-00540-17), the Coordination for the Improvement of Higher Education Personnel – CAPES (DeepEyes Project). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Computer Vision and Pattern Recognition (CVPR), 2015 (2015).

[2] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Computer Vision and Pattern Recognition (CVPR), 2015 (2015).

[3] T. Kong, A. Yao, Y. Chen, F. Sun, Hypernet: Towards accurate region proposal generation and joint object detection, in: Computer Vision and Pattern Recognition (CVPR), 2016 (2016).

[4] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, L. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2018 (2018).

[5] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T. Yang, E. Choi, Morphnet: Fast & simple resource-constrained structure learning of deep networks, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2018 (2018).

[6] H. Cai, L. Zhu, S. Han, Proxylessnas: Direct neural architecture search on target task and hardware, in: ICLR, 2019 (2019).

[7] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q. V. Le, Mnasnet: Platform-aware neural architecture search for mobile, in: CVPR, 2019 (2019).

[8] J. Lin, T. Xing, R. Zhao, Z. Zhang, M. B. Srivastava, Z. Tu, R. K. Gupta, Binarized convolutional neural networks with separable filters for efficient hardware acceleration, in: Computer Vision and Pattern Recognition (CVPR) Workshops, 2017 (2017).

[9] A. Bulat, G. Tzimiropoulos, Xnor-net++: Improved binary neural networks, in: British Machine Vision Conference (BMVC), 2019 (2019).

[10] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, R. S. Feris, Blockdrop: Dynamic inference paths in residual networks, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2018 (2018).

[11] X. Wang, F. Yu, Z. Dou, T. Darrell, J. E. Gonzalez, Skipnet: Learning dynamic routing in convolutional networks, in: European Conference on Computer Vision (ECCV), 2018 (2018).

[12] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in: International Conference on Computer Vision (ICCV), 2017 (2017).

[13] J. Luo, J. Wu, W. Lin, Thinet: A filter level pruning method for deep neural network compression, in: International Conference on Computer Vision, (ICCV), 2017 (2017).

[14] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, in: International Joint Conference on Artificial Intelligence (IJCAI), 2018 (2018).

[15] S. Han, J. Pool, J. Tran, W. J. Dally, Learning both weights and connections for efficient neural networks, in: Neural Information Processing Systems (NIPS), 2015 (2015).

[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, International Conference for Learning Representations(ICLR) (2017).

[17] H. Hu, R. Peng, Y. Tai, C. Tang, Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, In CoRR (2016). arXiv:1607.03250.

[18] Q. Huang, S. K. Zhou, S. You, U. Neumann, Learning to prune filters in convolutional neural networks, in: Winter Conference on Applications of Computer Vision (WACV), 2018 (2018).

[19] H. Wold, Partial Least Squares, in: Encyclopedia of Statistical Sciences, Wiley, 1985, pp. 581–591 (1985).

[20] W. R. Schwartz, A. Kembhavi, D. Harwood, L. S. Davis, Human detection using partial least squares analysis., in: International Conference on Computer Vision (ICCV), 2009 (2009).

[21] H. Abdi, Partial least squares regression and projection on latent structure regression (pls regression), Wiley Interdisciplinary Reviews: Computational Statistics (2010).

[22] S. Liu, W. Deng, Very deep convolutional neural network based image classification using small training sample size, Asian Conference on Pattern Recognition (ACPR) (2015).

[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Computer Vision and Pattern Recognition (CVPR), 2016 (2016).

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in: Computer Vision and Pattern Recognition (CVPR), 2009 (2009).

[25] A. Krizhevsky, V. Nair, G. Hinton, CIFAR-10 (Canadian Institute for Advanced Research) (2009).

[26] D. Mittal, S. Bhardwaj, M. M. Khapra, B. Ravindran, Recovering from random pruning: On the plasticity of deep convolutional neural networks, in: Winter Conference on Applications of Computer Vision (WACV), 2018 (2018).

[27] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, S. Han, Amc: Automl for model compression and acceleration on mobile devices, in: European Conference on Computer Vision (ECCV), 2018 (2018).

[28] R. Yu, A. Li, C. Chen, J. Lai, V. I. Morariu, X. Han, M. Gao, C. Lin, L. S. Davis, NISP: pruning networks using neuron importance score propagation, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2018 (2018).

[29] G. Roffo, S. Melzi, M. Cristani, Infinite feature selection, in: International Conference on Computer Vision (ICCV), 2015 (Dec 2015).

[30] Y. He, P. Liu, Z. Wang, Z. Hu, Y. Yang, Filter pruning via geometric median for deep convolutional neural networks acceleration, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2019 (2019).

[31] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: International Conference on Computer Vision, ICCV, 2017 (2017).

[32] Z. Huang, N. Wang, Data-driven sparse structure selection for deep neural networks, in: European Conference on Computer Vision (ECCV), 2018 (2018).

[33] J. Frankle, M. Carbin, The lottery ticket hypothesis: Finding sparse, trainable neural networks, in: International Conference on Learning Representations (ICLR), 2019 (2019).

[34] Z. Liu, M. Sun, T. Zhou, G. Huang, T. Darrell, Rethinking the value of network pruning, in: International Conference on Learning Representations (ICLR), 2019 (2019).

[35] A. M. Martínez, A. C. Kak, PCA versus LDA, Pattern Analysis and Machine Intelligence (PAMI) (2001).

[36] T. Mehmood, K. H. Liland, L. Snipen, S. Sæbø, A review of variable selection methods in partial least squares regression, Chemometrics and Intelligent Laboratory Systems (2012).

[37] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: A deep convolutional activation feature for generic visual recognition, in: International Conference on Machine Learning (ICML), 2014 (2014).

[38] A. S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN features off-the-shelf: An astounding baseline for recognition, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2014 (2014).

[39] W. Brendel, M. Bethge, Approximating cnns with bag-of-local-features models works surprisingly well on imagenet, in: International Conference on Learning Representations (ICLR), 2019 (2019).

[40] I. Loshchilov, F. Hutter, SGDR: stochastic gradient descent with warm restarts, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017 (2017).

[41] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, A. Zisserman, Vggface2: A dataset for recognising faces across pose and age, in: International Conference on Automatic Face & Gesture Recognition (FG), 2018 (2018).

[42] G. Roffo, S. Melzi, U. Castellani, A. Vinciarelli, Infinite latent feature selection: A probabilistic latent graph-based ranking approach, in: International Conference on Computer Vision (ICCV), 2017 (2017).

[43] H. Wang, Q. Zhang, Y. Wang, H. Hu, Structured probabilistic pruning for convolutional neural network acceleration, in: British Machine Vision Conference (BMVC), 2018 (2018).