Discriminative Layer Pruning for Convolutional Neural Networks

Artur Jordao, Maiko Lie, William Robson Schwartz

***Abstract*—The predictive ability of convolutional neural networks (CNNs) can be improved by increasing their depth. However, increasing depth also increases computational cost significantly, in terms of both floating point operations and memory consumption, hindering applicability on resource-constrained systems such as mobile and internet of things (IoT) devices. Fortunately, most networks have spare capacity, that is, they require fewer parameters than they actually have to perform accurately. This motivates network compression methods, which remove or quantize parameters to improve resource-efficiency. In this work, we consider a straightforward strategy for removing entire convolutional layers to reduce network depth. Since it focuses on depth, this approach not only reduces memory usage, but also reduces prediction time significantly by mitigating the serialization overhead incurred by forwarding through consecutive layers. We show that a simple subspace projection approach can be employed to estimate the importance of network layers, enabling the pruning of CNNs to a resource-efficient depth within a given network size constraint. We estimate importance on a subspace computed using *Partial Least Squares*, a feature projection approach that preserves discriminative information. Consequently, this importance estimation is correlated to the contribution of the layer to the classification ability of the model. We show that cascading discriminative layer pruning with filter-oriented pruning improves the resource-efficiency of the resulting network compared to using any of them alone, and that it outperforms state-of-the-art methods. Moreover, we show that discriminative layer pruning alone, without cascading, achieves competitive resource-efficiency compared to methods that prune filters from all layers.**

***Index Terms*—Network compression, network pruning, convolutional neural networks.**

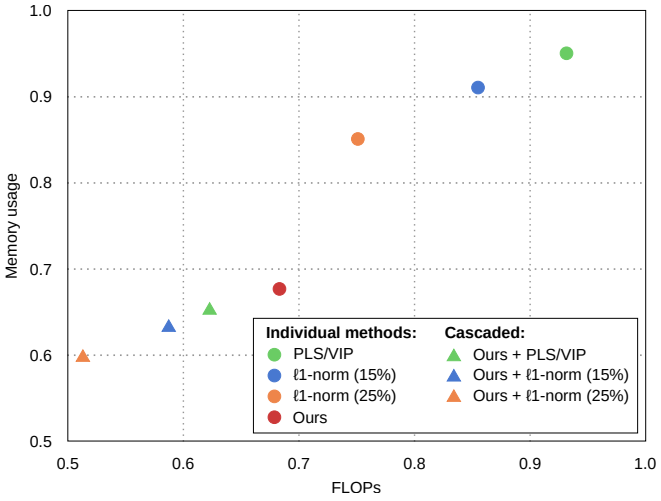

Fig. 1. Resource-efficiency of a ResNet110 model compressed by different pruning methods – the closer to the bottom left, the better. *The accuracy of all models shown are within one percent of the original, unpruned, network.* Discriminative layer pruning (labeled as "Ours"), can be cascaded with filter-oriented pruning methods to substantially improve their resource-efficiency. In fact, it is capable of achieving competitive performance even by itself, without cascading. The original network was pre-trained on the CIFAR-10 dataset. The axes indicate the memory and floating point operations (FLOPs) used during inference as fractions of the usage by the original network.

I. Introduction

CURRENT visual pattern recognition models are predominantly based on convolutional neural networks. This approach was proposed in the nineties by LeCun et al. [1], and entered the computer vision mainstream with the work by Krizhevsky et al. [2], which surpassed the accuracy of previous approaches by a large margin on the *ILSVRC 2012* challenge [3] using a deep convolutional neural network (CNN). The typical structure of early CNNs was straightforward — stacks of convolutional layers (some of them normalized and spatially pooled), followed by fully-connected layers. Since then, many works have suggested improvements such as normalization approaches [4] and more sophisticated convolutional blocks [5]. However, the most influential principle resulting from such improvements might be that increasing network depth is an effective approach to improve prediction ability [6][7][5], a strategy that is adopted by most modern CNN architectures. In fact, several of the most notable contributions in deep learning for visual tasks in recent years have involved strategies for training deeper convolutional neural networks [8][9][10].

While deeper networks are capable of learning to recognize more complex patterns, the resulting number of parameters incurs large computational cost in terms of both floating point operations (FLOPs) and memory consumption. The latter is a problem not only due to static storage size but, more importantly, due to run-time memory access, which can be a bottleneck due to the large number of feature maps computed during inference and accounts for most of the energy consumption by the network [11]. These issues lead to slow inference and can hinder, or even prevent, its deployment on resource-constrained systems such as mobile and internet of things (IoT) devices. This work addresses these issues by exploring network redundancy in depth (i.e., layer removal), which can improve memory usage by reducing the amount of feature maps computed (Fig. 1), as well as prediction time by reducing the amount of serialization overhead incurred by forwarding through consecutive layers.

There are two main strategies to leverage the large generalization ability of deep neural networks while mitigating

The authors are with the Smart Sense Laboratory, Department of Computer Science, Federal University of Minas Gerais, Belo Horizonte MG 31270-901, Brazil. (A. Jordao and M. Lie contributed equally to this work.)

their high computational cost — *neural architecture search* and *network compression*. The former is a synthesis approach and can be seen as a search for high-performance sub-graphs in a larger graph [12], typically with genetic algorithms [13][14] or more sophisticated controller mechanisms [15]. Since each candidate architecture in this search incurs an additional training cycle, this process is computationally intensive — some of them take several days to process even with hundreds of GPUs [12]. The latter, on the other hand, is a transformation approach, which starts from an existing network and leverages the redundancy in its parameters to achieve better resource-efficiency. The most popular approaches for compressing neural networks are parameter quantization [11][16], encoding [11][17] and pruning [18][19].

In this work, we focus on network compression by pruning, which consists in removing unimportant parameters from the network to reduce resource usage while preserving as much accuracy as possible. Pruning is an attractive approach because it can be applied to off-the-shelf networks to fit a given set of resource constraints and it has achieved impressive trade-offs between resource usage and accuracy. In some cases, pruning is capable of compressing network size while also improving accuracy, demonstrating its ability to reduce overfitting [20][21].

We introduce *layer pruning* using a discriminative importance estimation criterion based on *Partial Least Squares* (PLS) projection [22]. Since PLS projection preserves discriminative information, this criterion is correlated to the contribution of the parameters to the classification ability of the network. Differently from most previous approaches, instead of removing parameters from all layers, or even on a layer-by-layer basis, we remove *entire layers* (or *blocks*, depending on the architecture), starting from the end of the network and stopping according to an importance criterion, therefore decreasing depth. The remaining layers can then be pruned using other approaches, such as filter-oriented methods [18].

The main contributions of this work are: (i) the introduction of a discriminative layer ranking criterion and experiments showing its efficacy for layer pruning in CNNs, (ii) the demonstration that cascading layer pruning with filter pruning not only improves memory usage and FLOP count compared to using any of them alone, but also improves prediction time even at similar FLOP count due to the reduction of serialization in the network, which is proportional to depth.

We show that cascading layer pruning with other pruning methods improves the resource-efficiency of the resulting network compared to using any of them alone (Fig. 1). Since the computational cost of several pruning methods is dominated by fine-tuning cycles for each layer [19][23][24][25], pruning efficiency can also be improved by this cascaded approach. Another interesting result is that layer pruning alone, without cascading, while seemingly simplistic, still achieves competitive resource-efficiency compared to filter-oriented methods. To demonstrate the effectiveness of discriminative layer pruning, we conduct experiments with residual networks [9] on the CIFAR-10 and ImageNet datasets. We focus on residual networks since they are not only the state-of-the-art in computer vision, but are also deeper than other models often used in pruning experiments (e.g., AlexNet and VGG) and have fewer parameters in their fully-connected layers, thus allowing a more challenging assessment [18].

II. Related Work

Since deep neural networks are typically over-parameterized, removing unimportant parameters — *pruning*, is a reasonable strategy to improve their resource-efficiency and is often the first choice for network compression. There are technical and theoretical motivations for compression by pruning. Technically, pruning is attractive because it can be applied to off-the-shelf networks and often leads to higher compression rates than quantization [11]. Moreover, it is more efficient than neural architecture search, since starting from a pre-trained network entails search on a much smaller parameter space. Theoretically, it has been argued that pruning over-parameterized networks leads to higher accuracy than training small networks from scratch because the combination of weights and connections from a pre-trained network leads to a more effective region of the parameter space, which would be harder to reach otherwise [26]. This fact has also been suggested by several experimental works [18][23].

Pruning consists in parameter removal and re-training (i.e., fine-tuning), possibly over several iterations, to readjust the network to the architecture change. Thus, the central problem in pruning is to choose which parameters should be removed from the network. Since re-training the network to evaluate all possible cases is intractable, efforts have been devoted in the design of importance estimation functions that perform *parameter ranking*, that is, assignment of scores used to estimate the relative importance of different sets of parameters. While these sets can be selected from arbitrary locations in the network, a structured approach is often preferred. In other words, instead of individual parameters, coarser elements of the network (e.g., filters, layers) are removed. This is motivated mainly by computational efficiency since removing individual parameters leads to fine-grained sparsity, which can result in poor locality of reference and is detrimental to hardware optimization [27]. It can also be argued that coarser elements are more interpretable than individual parameters since, for instance, filters often describe meaningful visual patterns and the position of layers suggest their relative abstraction level. Moreover, there is evidence suggesting that coarse-grained (i.e., structured) sparsity acts as regularization once it constrains the position of the parameters [28].

A reasonable heuristic to follow when designing pruning criteria is evaluating the effect of parameters on activations. For instance, among the simplest criteria, the ℓ_1-norm is arguably the most popular. Pruning based on this criterion consists basically in removing filters that have small ℓ_1-norm (i.e., average magnitude) and it is motivated by the fact that filters with small weights tend to output weak activations [18]. A more data-driven strategy is to apply the network to a dataset and measure the average number of zero activations associated with each parameter during prediction [23] — since zero activations are assumed to have little, if any, impact on the output of subsequent layers, the parameters that originate them

can be good candidates for pruning. An alternative is to avoid assuming the impact of parameters on activations altogether and simply minimize the reconstruction error associated with them [19][24], possibly considering consecutive layers [29].

The aforementioned approaches operate locally, that is, the information they consider for importance estimation is constrained to the neighborhood of the parameter (e.g. its surrounding filter or layer). A recent work [30] has argued that parameters assumed to be unimportant in an early layer can actually contribute significantly to the response of important parameters in later layers and that, consequently, global importance estimation might be more adequate. To address this, Yu et al. [30] proposed a strategy that performs feature ranking on the final response layer of the network (i.e., the one immediately before the classifier) and propagates it backwards through the network as a neuron importance measure. Considering that the response of that layer is what is effectively input to the classifier, they argue that minimizing its reconstruction error is an effective goal for pruning. While feature ranking on the final response layer can provide useful information regarding the effect of parameters on the classification ability, it is still restricted to what is input to the classifier. Zhuang et al. [31] argued that a more direct approach is to simply assess the output of the classifier, for instance, by optimizing a discrimination-aware loss during fine-tuning. They proposed and demonstrated the effectiveness of computing several discrimination-aware losses distributed across the network. In our experiments, we assess discrimination using a different strategy — we project the output of different layers of the network on subspaces using Partial Least Squares (PLS) [22], a feature projection approach that maximizes the covariance between the transformed features and the class labels. PLS has shown remarkable effectiveness for dimensionality reduction of visual features [32] and has recently been employed for filter-oriented pruning by Jordao et al. [33]. In contrast to the latter, instead of projecting a single subspace for all filters, we project one subspace for each layer, therefore significantly reducing memory requirements during pruning. Moreover, we focus on reducing depth instead of removing filters across the entire network.

Depth reduction has been mostly neglected in network compression, which is reasonable since the trend in neural network architecture design has been the exact opposite — increasing depth to improve prediction ability [7][8][9][10]. However, in resource-constrained systems, it is disadvantageous to employ very deep networks in a one-size-fits-all manner. While, to our knowledge, this has not been explored in pruning, recent work on adaptive neural networks [34][35] have emphasized the possibility of truncating prediction when early layers are sufficient to classify with high confidence [36][37][38]. The motivation is that only hard samples need to reach very deep layers for accurate prediction, while the majority of samples are actually easy to predict and can stop early [39]. This suggests a trade-off that can be leveraged to increase efficiency while preserving accuracy. Depth reduction by pruning relies on similar assumptions, but it does not insert additional parameters or decision logic to the network. A significant advantage is that, by not making any drastic modification to the architecture, other pruning methods can be cascaded to increase resource-efficiency without any additional implementation effort.

III. Discriminative Layer Pruning

A. Problem Definition

Let the input model M be a residual network composed by a set of residual blocks B_i. The pruning method must assign an importance score S_i to each block and employ a policy to remove those with low scores. After fine-tuning, the pruned model M_p can then be employed for prediction as-is or pruned further using, for instance, filter-oriented methods.

Note that we adopt notation and definitions in terms of residual blocks only for convenience, since our experiments are based on residual networks. The approach is identically applicable to the output of ordinary convolutional layers or more elaborate blocks (e.g., inception blocks [7]).

B. Importance Estimation

The criterion we use for importance estimation is based on Partial Least Squares (PLS), which is a discriminative feature projection method that maximizes the covariance between the transformed features and the class labels [22]. PLS is often used for dimensionality reduction and can be employed as an alternative to Linear Discriminant Analysis (LDA), since it does not have some limitations of the latter, namely having issues when the number of features is larger than the number of samples and being capable of providing only as many meaningful features as there are classes. PLS can be computed using *nonlinear iterative partial least squares* (NIPALS), described in Alg. 1, where c is the dimensionality of the output features.

There are several approaches to perform feature selection based on a subspace projected via PLS. We employ the *Variable Importance in Projection* (VIP) score [40], which is

Algorithm 1: NIPALS

Input : Data matrix $X \in \mathbb{R}^{m \times n}$
Label vector $y \in \mathbb{R}^n$
Number of components c

Output: Weight matrix W

for $i \leftarrow 1$ **to** c **do**
  Repeat until w_i converges:
    Let $u \in R^{m \times 1}$ be a randomly initialized vector
    $w_i \leftarrow X^\top u / ||X^\top u||$
    $t_i \leftarrow X w_i$
    $q_i \leftarrow y^\top t_i / ||y^\top t_i||$
    $u \leftarrow y q_i$
  $p_i \leftarrow X^\top t_i$
  $X \leftarrow X - t_i p_i^\top$
  $y \leftarrow y - t_i q_i^\top$
end

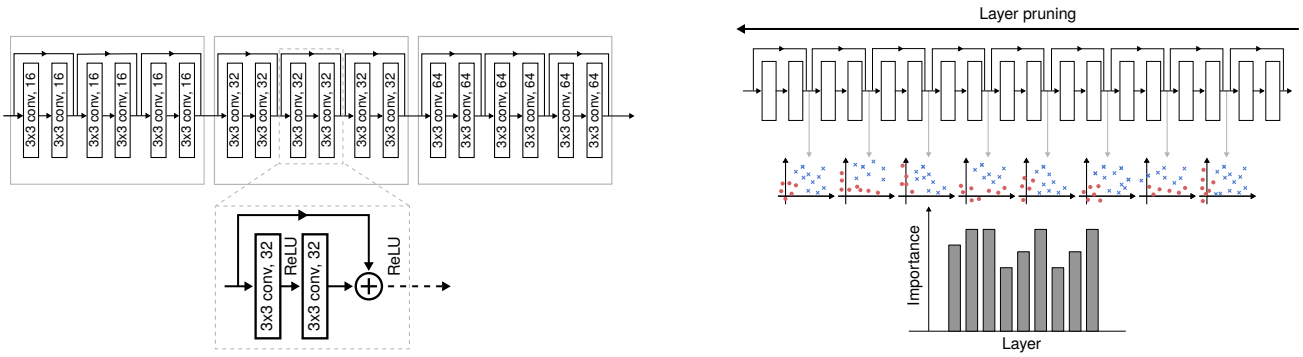

Fig. 2. Overview of discriminative layer pruning. **Left.** Convolutional layers of a residual network (ResNet20). The PLS projection is computed from the output of residual blocks (dashed lines). Rectangles in solid gray lines indicate where feature map dimensions are the same. **Right.** A PLS subspace is computed from the output of each residual block. The subspaces are represented by the colored plots, where each color is a different class, indicating that the projection is discriminative. The normalized VIP scores of each subspace are employed for importance estimation and ranking. Pruning is performed from the end towards the beginning of the network.

perhaps the most popular approach employed with PLS. VIP measures how much a feature contributes to the projection and, for each jth feature, is calculated as

$$VIP_j = \sqrt{m \sum_{k=1}^{c} SS_k (w_{kj}/\|w_k\|^2) / \sum_{k=1}^{c} SS_k}, \qquad (1)$$

where m is the number of features, SS_k is the sum of squares explained by the kth feature, and w_{jk} is the jth element of the weight vector $w_k \in W$.

We could estimate the score of the feature output by each residual block as the average VIP score of its subspace. However, since the scores of different blocks are computed from different subspaces, comparing their average values directly is not particularly meaningful. Instead, to extract more meaningful values to ranking, we compute the reciprocal of their coefficient of variation (CV), i.e., ratio of the mean to the standard deviation, as importance score.

Algorithm 2: Discriminative layer pruning

Input : Pre-trained model M
Training samples X
Training labels y
Output: Pruned model M_p

foreach $B_i \in M$ **do**
  $L_i \leftarrow \text{NIPALS}(B_i, y, c)$
  $I_i \leftarrow \text{VIP}(B_i)$
  $S_i \leftarrow 1/\text{CV}(I_i)$
end
$M_p \leftarrow M$
for $i \leftarrow |M|$ **downto** 1 **do**
  if $S_i < S_{i-1}$ **then**
    $M_p \leftarrow M_p \setminus B_i$
  end
end
Fine-tune M_p

C. Layer Pruning

The policy adopted in this work is to (i) prune from the end towards the beginning of the network and (ii) stop pruning before the feature map dimension changes in the network (i.e., Fig. 2, left — connections between rectangles in gray solid lines). The former is motivated by the fact that abstraction level increases towards the end of the network, with its beginning containing fundamental visual features (e.g., oriented edges, corners) that are more likely to compromise prediction ability if removed. The latter is because accuracy degrades significantly from that point on, so it is not computationally advantageous to further reduce depth. This is particular to ResNets and might be dispensable with other architectures. In fact, such degradation is consistent with the observation made by some authors that there are several layers in ResNets that are best left as-is since they are more sensitive to pruning and that some of these layers take place close to where the dimensions of the feature map changes [18].

With our policy in place, the procedure for discriminative layer pruning, illustrated in Fig. 2 (right) and detailed in Alg. 2, is straightforward and proceeds as follows. Given a pre-trained model M, for the output of each residual block B_i in this model, we compute a PLS projection and its respective importance scores, where the X and y input to NIPALS are the vectorized feature maps from this block and their labels, respectively. Starting from the end of the network (i.e., last residual block), we follow a greedy approach and keep removing blocks if the importance score of the next block B_{i-1} is larger than that of the current block B_i (remember that we are going backwards through the network). After these removals, the model is fine-tuned, resulting in the pruned model M_p.

IV. Experiments

A. Experimental Setup

Datasets. We evaluate the discriminative layer pruning on the CIFAR-10 dataset [41], which is composed of 32×32 RGB images (50K for training and 10K for testing), describing 10 classes of objects. Additionally, we perform comparative assessment on the ImageNet large-scale image classification

dataset [3], which is composed of 224×224 RGB images (1.2M for training and 50K for testing), describing 1,000 object classes. In the cascading experiments, we employ the 32×32 variant of ImageNet instead, to enable training several network combinations since this would be unfeasible on the original ImageNet due to the large computational cost.

Metrics. We assess resource-efficiency in terms of classification accuracy, FLOPs and memory consumption. Additionally, we assess the ability of different importance functions to correctly assign relative importance to different layers (i.e., rank). We assume that a layer is important if the removal of all layers after it, followed by fine-tuning, results in a network with high accuracy (i.e., pruning stops at this layer). Thus, we compute these classification accuracies and use them as ground-truth for measuring the importance ranking ability of a given importance function. This is done by computing the *pairwise ranking accuracy*

$$R = \frac{|\mathcal{P}| - \sum_{(p,q)\in\mathcal{P}} f(S, A, p, q)}{|\mathcal{P}|}, \tag{2}$$

where S contains the scores assigned to each layer by the importance function being assessed, A contains the classification accuracies of the network when pruning stops at each layer, $\mathcal{P}$ is the set of all ordered pairs of layers, and f is a binary function describing whether the importance scores of a pair of layers preserve the same rank as their classification accuracy

$$f(S, A, p, q) = \begin{cases} 1, & \text{if } S_p \geq S_q \text{ and } A_p < A_q, \\ 1, & \text{if } S_p \leq S_q \text{ and } A_p > A_q, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Implementation details. In our experiments, the training is performed using SGD for 200 epochs, with a batch size of 128. We follow a learning rate schedule similar to He et al. [9], starting with a learning rate of 0.01 and dividing it by 10 once training reaches 100 epochs and again when it reaches 150 epochs. The data augmentation procedure consists in padding the image with four pixels in each direction and sampling a random 32×32 crop from this padded image, or its horizontal flip with a 50% chance. All PLS projections are computed for two components (i.e., c = 2 in Alg. 1, see next section). The experiments were conducted on a computer system with an Intel Xeon Silver 4116 CPU, with an NVIDIA GTX 1080 GPU.

B. Number of Components

The dimensionality of the PLS model is defined by the number of components onto which it projects the input features. We assessed the ranking accuracy of the PLS/VIP criterion for different number of components the CIFAR-10 and ImageNet datasets. The results are presented in Figure 3.

The number of components considered in our assessment is rather small (i.e., ≤10). This is usual for PLS models since they often require few latent variables [42][43][44] — for instance, features with dimensions as high as 170K have been shown to require only 20 components for optimal

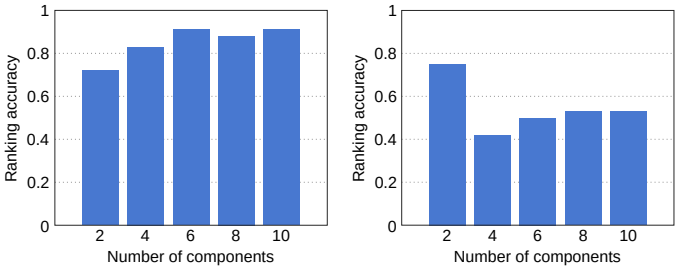

Fig. 3. Ranking accuracy according to number of PLS components. **Left:** ResNet20 on CIFAR-10. **Right:** ResNet50 on ImageNet.

TABLE I
PAIRWISE RANKING ACCURACY. RANKING WAS COMPUTED FOR RESIDUAL BLOCKS OF A RESNET20 NETWORK ON THE CIFAR-10 DATASET. PLS WAS COMPUTED WITH $c = 2$ COMPONENTS.

Method	Pairwise ranking accuracy
IFS [45]	0.64
ILFS [46]	0.50
PLS/VIP	**0.72**

representation in terms of discriminability [32]. We employed c = 2 in our experiments considering computational efficiency, model size, and its adequate ranking accuracy on ImageNet and CIFAR-10.

C. Ranking Layer Importance

Since importance estimation functions that operate on filters and channels do not translate directly to layers, we compare the layer ranking ability of the PLS/VIP criterion with other two state-of-the-art feature ranking methods employed in vision tasks, namely *infinite feature selection* (IFS) [45] and *infinite latent feature selection* (ILFS) [46] — both are based on the representation of feature subsets as paths on an affinity graph. The former was employed in the importance propagation pruning approach by Yu et al. [30] as alternative to a magnitude-based criterion.

The pairwise ranking accuracy of the compared methods is presented in Table I, considering the residual blocks of a ResNet20 model on the CIFAR-10 dataset. We limited this assessment to this setting because of the high computational cost required to compute the ground-truth accuracies A in Eq. 2 (i.e., one accuracy per residual block) and because it is unfeasible to compute IFS and ILFS on ImageNet. The PLS/VIP criterion presented the highest ranking accuracy, demonstrating its effectiveness for ranking layer importance and supporting its use for discriminative layer pruning. Note that we compute the PLS model using the entire training set, but this does not need to be the case since NIPALS is resilient to missing data [47]. For instance, Jordao et al. [33] have shown that, when computing PLS for CNN filter pruning, the difference between employing the full training set and 10% of it was smaller than one percentage point in validation accuracy on the CIFAR-10 dataset.

The resource-efficiency of the compared methods is presented in Table II for the CIFAR-10 dataset — since all pruned models have accuracy within one percent of the original

TABLE II
RESOURCE-EFFICIENCY OF DISCRIMINATIVE LAYER PRUNING ON THE CIFAR-10 DATASET. FLOP REDUCTION AND MEMORY USAGE ARE INDICATED AS PERCENTAGE OF THE VALUES COMPARED TO THE ORIGINAL NETWORK. ACCURACY REDUCTION IS INDICATED IN PERCENTAGE POINTS, IN WHICH NEGATIVE VALUES DENOTE IMPROVEMENT WITH RESPECT TO THE ORIGINAL NETWORK. CASCADING DISCRIMINATIVE LAYER PRUNING IMPROVES RESOURCE-EFFICIENCY IN ALL CASES COMPARED TO USING ANY SINGLE METHOD ALONE, WITH ACCURACY DROP WITHIN ONE PERCENT.

	Method	FLOPs↓ (%)	Memory usage↓ (%)	Accuracy↓ (top-1, p.p.)
ResNet20	Ours	11.56	10.95	−0.03
	ℓ_1-norm (15%)	14.27	8.57	**−0.15**
	Ours + ℓ_1-norm (15%)	**24.02**	**17.58**	0.00
	ℓ_1-norm (25%)	24.56	14.29	0.20
	Ours + ℓ_1-norm (25%)	**33.23**	**22.51**	**−0.35**
	PLS/VIP	8.00	5.71	**−0.07**
	Ours + PLS/VIP	**16.75**	**14.40**	0.13
ResNet56	Ours	30.01	30.16	0.98
	ℓ_1-norm (15%)	14.48	8.91	**−0.18**
	Ours + ℓ_1-norm (15%)	**39.81**	**34.45**	0.95
	ℓ_1-norm (25%)	24.86	14.85	**−0.31**
	Ours + ℓ_1-norm (25%)	**47.37**	**38.24**	0.82
	PLS/VIP	7.09	4.95	**−0.60**
	Ours + PLS/VIP	**36.70**	**32.76**	0.45
ResNet110	Ours	31.68	32.32	0.18
	ℓ_1-norm (15%)	14.53	8.96	**−0.39**
	Ours + ℓ_1-norm (15%)	**41.26**	**36.69**	0.27
	ℓ_1-norm (25%)	24.93	14.93	**−0.35**
	Ours + ℓ_1-norm (25%)	**48.69**	**40.39**	0.25
	PLS/VIP	6.85	4.98	**−0.59**
	Ours + PLS/VIP	**37.73**	**34.77**	0.06

network, we can assume that the decrease in computational cost in this dataset comes at practically no cost in prediction ability. The percentage accompanying the ℓ_1-norm indicates the pruning rate employed. Results show that, when cascaded, discriminative layer pruning improves over the result of any filter pruning method used individually. As the network gets deeper, cascaded layer pruning tends to account for more reduction in resource usage. Surprisingly, it is often competitive even without cascading, for instance, its resource-efficiency is superior to all individual pruning methods on ResNet56.

Considering the simplicity of the method, which simply discards entire residual blocks sequentially, the results in Table II are interesting For instance, while He et al. [19] suggested more aggressive pruning on early layers since deeper layers are more challenging, we find that, not just pruning, but *removing* precisely the deeper layers is surprisingly effective in terms of resource-efficiency of the resulting network. This might be explained by the recent hypothesis that the effectiveness of pruning is not so much a result of the importance criteria used for parameter selection as it is of the plasticity of neural networks [48]. In other words, while it is known that fine-tuning is essential to pruning [49], it has a greater importance in its effectiveness than is often assumed.

D. Cascading Discriminative Layer Pruning

We assess the resource-efficiency of our approach both individually and cascaded with filter pruning methods. Two filter pruning criteria were considered, one magnitude-based and one discriminative — the ℓ_1-norm approach by Li et al. [18] and the PLS/VIP approach by Jordao et al. [33], respectively. We use our own implementation of both. Note that, in their experiments, Li et al. [18] adopt several empirically defined pruning ratios for different parts of the network, which vary for different ResNet sizes. Since we are only interested in the filter importance criterion and not in optimizing results, we assess this criterion for two pruning ratios that we assume reasonable (15% and 25%), and adopt a single pruning ratio for the entire pruning in each case. However, exclusively when using the criterion by Li et al. [18], we follow their approach and prune only the first layer of residual blocks. As for the approach by Jordao et al. [33], we employ the settings indicated in their paper for one-shot pruning.

We also assess our approach for large-scale image classification. The results on the ImageNet dataset are presented in Table III. This dataset is noticeably more challenging — in general, the decrease in accuracy is not negligible. In terms of reducing resource usage, the cascaded approach remains advantageous with respect to filter pruning methods, except when pruning ResNet56 using the ℓ_1-norm (25%). In this case, the latter achieves an additional 4.69 p.p. FLOP reduction, however, its memory usage reduction is 1.55% inferior, while its accuracy decreases an additional 1.59 p.p. (percentage points) with respect to the cascaded approach. The only case where cascading leads to as decrease in accuracy compared to employing only filter pruning is for the ℓ_1-norm (25%) on ResNet20, where cascading leads to an additional 2.16 p.p. drop in accuracy. This seems to be a reflection of the high variance in accuracy when pruning ResNet20 on this dataset since using the ℓ_1-norm (15%) actually leads to an increase of 3.55 p.p., the largest accuracy improvement on this dataset. A possible cause for this high variance is that ResNet20 has

TABLE III
RESOURCE-EFFICIENCY OF DISCRIMINATIVE LAYER PRUNING ON THE IMAGENET-32 DATASET. FLOP REDUCTION AND MEMORY USAGE ARE INDICATED AS PERCENTAGE OF THE VALUES FOR THE ORIGINAL NETWORK. ACCURACY REDUCTION IS INDICATED IN PERCENTAGE POINTS, IN WHICH NEGATIVE VALUES DENOTE IMPROVEMENT WITH RESPECT TO THE ORIGINAL NETWORK.

	Method	FLOPs↓ (%)	Memory usage↓ (%)	Accuracy↓ (top-5, p.p.)
ResNet20	Ours	23.08	20.48	2.09
	ℓ_1-norm (15%)	14.24	7.68	0.35
	Ours + ℓ_1-norm (15%)	**33.72**	**25.47**	**−3.55**
	ℓ_1-norm (25%)	24.52	13.09	**2.44**
	Ours + ℓ_1-norm (25%)	**41.84**	**29.28**	4.60
	PLS/VIP	6.40	4.33	3.20
	Ours + PLS/VIP	**28.18**	**23.12**	**2.92**
ResNet56	Ours	11.25	11.05	−2.10
	ℓ_1-norm (15%)	14.47	8.54	3.80
	Ours + ℓ_1-norm (15%)	**20.15**	**16.09**	**3.39**
	ℓ_1-norm (25%)	**24.84**	14.54	4.48
	Ours + ℓ_1-norm (25%)	20.15	**16.09**	**2.89**
	PLS/VIP	9.27	5.73	3.82
	Ours + PLS/VIP	**20.15**	**16.09**	**2.78**
ResNet110	Ours	31.67	31.93	1.60
	ℓ_1-norm (15%)	14.52	8.78	**5.29**
	Ours + ℓ_1-norm (15%)	**41.25**	**36.54**	6.28
	ℓ_1-norm (25%)	24.92	14.94	**5.88**
	Ours + ℓ_1-norm (25%)	**48.68**	**40.18**	6.66
	PLS/VIP	11.77	6.85	**5.43**
	Ours + PLS/VIP	**38.62**	**35.26**	6.27

the smallest capacity among the compared networks, leading to a higher sensitivity to pruning.

Aside from the aforementioned cases, cascading is always advantageous. When the cascaded approach presents larger decrease in accuracy, it is within one percent of the decrease when using only filter pruning, while still providing substantially larger reductions in computational cost. More specifically, the relative increase in resource-efficiency of cascading ranges from 8.67% to 30.88% for FLOP reduction and from 8.22% to 29.79% for memory usage reduction.

While FLOP count is a useful estimate of computational cost, it does not necessarily translate into prediction time due to factors such as locality of reference (i.e., fetching from memory) and compiler optimizations [50]. For this reason, we compare the prediction time of a model compressed using discriminative layer pruning and the previously considered filter pruning approaches. The average prediction time for each of the resulting models are presented in Fig. 4 for ResNet110 on ImageNet-32 (the same models reported in Table III). Despite achieving different and substantial FLOP reduction, the filter pruning methods lead to models with similar prediction time when used individually. The cascaded models, on the other hand, result in models with substantially smaller prediction time, which are also more consistent with their FLOP reduction. This is likely because there is a serial factor during prediction — to compute convolutions in one layer, the model must wait for the output of the previous layer. The deeper the network, the larger the effect of this factor, consequently, the cascaded approach mitigates this by reducing depth before removing filters.

Since the amount of depth reduction provides additional insight to interpret the resource-efficiency of the pruned models, we present the depth reduction of our approach in Table IV. The results indicate that up to 31% of depth was removed — in the case of CIFAR-10, with accuracy within one percentage point of the original network. Note that this is before any additional pruning, so one-third of the network is pruned away, resulting in almost no degradation in accuracy, even before searching for redundant filters. The fact that this simple approach leads to such high resource-efficiency is a strong argument for the plasticity of neural networks.

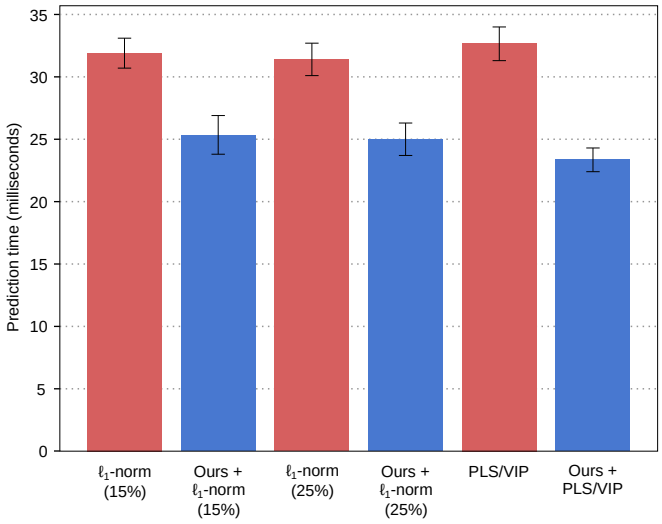

Fig. 4. Average prediction time using a ResNet110 model compressed by different pruning methods. The input is a 32×32 RGB image and each average prediction time was computed from 30 predictions. The model was pre-trained on the ImageNet-32 dataset. Error bars indicate the 95% confidence interval.

TABLE IV
DEPTH OF RESIDUAL NETWORKS COMPRESSED USING DISCRIMINATIVE LAYER PRUNING. THE NUMBERS IN PARENTHESES INDICATE THE PERCENTAGE OF CONVOLUTIONAL LAYERS REMOVED, WHILE THE ORIGINAL NUMBER OF LAYERS IN EACH MODEL IS INDICATED BY THE SUFFIX TO ITS NAME. FOR THE CIFAR-10 DATASET, THE PRUNED MODELS HAVE TOP-1 ACCURACY WITHIN ONE PERCENT OF THE ORIGINAL MODEL.

	Model		
	ResNet20	ResNet56	ResNet110
CIFAR-10	18 (↓10%)	40 (↓29%)	76 (↓31%)
ImageNet-32	16 (↓20%)	50 (↓11%)	76 (↓31%)

TABLE V
RESOURCE-EFFICIENCY OF LAYER PRUNING ON A MOBILENET TRAINED ON THE IMAGENET DATASET.

Method	FLOPs↓ (%)	Accuracy↓ (top-1, p.p.)	Parameters↓ (%)
Shallow MobileNet	53.20	5.3	30.95
Ours	11.18	5.3	47.62

E. Pruning Layers in Compact Networks

We performed an experiment with a MobileNet [51] trained on ImageNet to assess the efficacy of our layer pruning approach on a network which is already compact by design. We report results for our approach and for the *Shallow MobileNet* presented in the original paper. The latter is a variant of the MobileNet comprised by the baseline network with five $14\times14\times512$ layers removed.

Our results are presented in Table V. While our approach preserves as much accuracy as Shallow MobileNet, the latter achieves significantly more FLOP reduction since it removes layers precisely where several large feature maps would be computed. However, this approach is handcrafted by the authors of the architecture — there is no explicit criterion for this choice. Our approach, on the other hand, is fully automatic, generally applicable, and still removes a substantial amount of FLOPs. Moreover, due to our greedy policy, our approach removes significantly more parameters (i.e., an additional 16.67% — 0.7M), leading to a more compact network model.

F. Comparison with the State-of-the-Art

We compare our approach to several other state-of-the-art pruning methods. Considering that our comparison is based on the values reported by the authors of each method, the assessment is restricted to ResNet56 on the CIFAR-10 dataset since it is the most reported setting based on residual networks. Note that for this comparison, in contrast to the cascading experiments in the last section, the results for Li et al. [18] (A), (B) and Jordao et al. [33] are presented for the best configurations reported in their papers since we are now interested in the methods themselves instead of their pruning criteria. Table VI presents the resource-efficiency of the compared methods. Since the decrease in accuracy is within one percentage point of the original network for all methods, we assume that it is negligible and consider that a superior FLOP reduction implies better resource-efficiency.

Among the compared methods, our discriminative layer pruning approach cascaded with iterated PLS/VIP filter pruning is the most resource-efficient, achieving a 62.69% FLOP reduction. Note that we present results for our approach cascaded with both *one-shot* and *iterative* PLS/VIP filter pruning. The former corresponds to the setting we employed for the cascading experiments in the previous section (a single iteration), while the latter follows the results of Jordao et al. [33], which show that this criterion is more effective when performed for several iterations. On that note, the approach by Jordao et al. [33] achieved a 52% FLOP reduction, the second highest resource-efficiency in our assessment, further demonstrating the effectiveness of PLS/VIP as a pruning criterion. However, while Jordao et al. [33] achieve their optimal result with eight iterations, our cascaded approach achieves its optimal and substantially superior result with only six iterations, indicating that cascaded layer pruning is indeed advantageous. Moreover, performing iterative pruning on a shallower network is more efficient, since it reduces the computational cost of fine-tuning.

TABLE VI
COMPARISON WITH STATE-OF-THE-ART METHODS. THE ACCURACY DECREASE (INDICATED IN PERCENTAGE POINTS) IS WITHIN ONE PERCENT OF THE ORIGINAL NETWORK FOR ALL METHODS. THE RESULTS IN EACH SECTION ARE ORDERED BY FLOP REDUCTION.

Method	FLOPs↓ (%)	Accuracy↓ (top-1, p.p.)
Jordao et al. [33] (8 iter.)	52.56	−0.62
He et al. [52]	50.00	0.90
Zhuang et al. [31]	47.08	−0.01
Yu et al. [30]	43.61	0.03
Li et al. (B) [18]	27.60	−0.02
Li et al. (A) [18]	10.40	−0.06
Ours + PLS/VIP (6 iter.)	**62.69**	**0.91**
Ours + ℓ_1-norm (25%)	47.37	0.82
Ours + PLS/VIP (1 iter.)	36.70	0.45
Ours	30.00	0.98

The approach by He et al. [52] is the third most resource-efficient, achieving a 50% FLOP reduction. while discriminative layer pruning cascaded with ℓ_1-norm (25%) pruning and the approach by Zhuang et al. [31] have the fourth and fifth best results, with FLOP reductions of 47.37% and 47.08%, respectively. At this point it is important to highlight that four of the five top results are based on discriminative pruning. The exception is the approach by He et al. [52], which employs a reinforcement learning framework. The discriminative approaches are arguably simpler, since they are based on straightforward subspace projections or loss functions.

While not as close to the top-performing method as the previous three methods, the approach proposed by Yu et al. [30] still removes 43.61% of FLOPs. This demonstrates that, while not as competitive as employing a discriminative criterion, its importance propagation strategy is in fact effective. The proposed approach cascaded with one-shot PLS/VIP filter pruning removes 36.70% of FLOPs, which is significantly less than other discriminative approaches, which is expected since this criterion performs better when iterated [33], but still superior to the ℓ_1-norm approach by Li et al. [18].

Surprisingly, the proposed layer pruning approach *even without cascading* still outperforms the ℓ_1-norm approach by

Li et al. [18] in both of its configurations. Since the latter approach is not data-driven, this result is reasonable, but considering that this approach relies handcrafted adjustments, such as several empirically defined pruning ratios and layers to skip due to sensitivity to pruning, it is interesting that an approach that is essentially a truncation of the network based on a discriminative criterion can outperform it.

G. Summary and Discussion

There are several takeaways from our results. We showed that a straightforward layer pruning approach is capable of reducing between 10% and 31% of the network depth with modest decrease in accuracy, or at almost no decrease at all in the case of the CIFAR-10 dataset. The fact that a change as drastic as removing several layers from CNNs (more than 30 layers in the case of ResNet110) can come at such a modest cost in predictive power demonstrates that employing very deep networks in a one-size-fits-all manner is wasteful and inadequate for resource-constrained systems. Moreover, while we cascaded our approach with only two simple, but representative, baselines — magnitude-based and discriminative filter pruning, to enable a more extensive assessment, our results suggest that it is likely that more sophisticated state-of-the-art pruning methods would also be improved by this approach.

Even when competing methods achieve similar FLOP reduction, our assessment of prediction time showed that our approach leads to shorter prediction time. The reason is possibly the dependency relation between consecutive layers in the network — one layer must wait for the output of the preceding layer before computing its own feature map. This leads to serialization in the network, which is proportional the network depth and is consequently mitigated by layer pruning.

V. Conclusions

We introduced a pruning approach that reduces the network depth by removing layers according to a discriminative criterion based on Partial Least Squares (PLS). This criterion is used to calculate importance scores that are used in a greedy strategy that sequentially removes unimportant layers. This approach can be cascaded with other pruning approaches and it was assessed when cascaded with two filter pruning baselines: a magnitude-based approach and a discriminative approach. Our results demonstrate that the cascaded approach improves resource-efficiency substantially in practically all cases. When cascaded with iterated filter pruning using the PLS/VIP criterion, it achieves resource-efficiency superior to state-of-the-art methods. Moreover, depth pruning reduces serialization in the network, leading to improved prediction time compared to methods that prune filters across all layers.

Acknowledgments

The authors would like to thank the Brazilian National Research Council – CNPq (Grants #311053/2016-5 and #438629/2018-3), the Minas Gerais Research Foundation – FAPEMIG (Grants APQ-00567-14 and PPM-00540-17) and 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] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Adv. in Neural Inf. Process. Syst.*, 1990, pp. 396–404.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Adv. in Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. of Comput. Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[4] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. of the Int. Conf. on Mach. Learn.*, 2015, pp. 448–456.

[5] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *Proc. of the AAAI Conf. on Artif. Intell.*, 2017.

[6] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of the Int. Conf. on Learn. Representations*, 2015.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition*, 2015, pp. 1–9.

[8] Rupesh K. Srivastava, Klaus Greff, and Jürgen Schmidhuber, "Training very deep networks," in *Adv. in Neural Inf. Process. Syst.*, 2015, pp. 2377–2385.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition*, 2016, pp. 770–778.

[10] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger, "Densely connected convolutional networks," in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[11] Song Han, Huizi Mao, and William J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. of the Int. Conf. on Learn. Representations*, 2016.

[12] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean, "Efficient neural architecture search via parameters sharing," in *Proc. of the Int. Conf. on Mach. Learn.*, 2018, pp. 4095–4104.

[13] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hegde, "Designing neural networks using genetic algorithms," in *Proc. of the Int. Conf. on Genetic Algorithms*, 1989, vol. 89, pp. 379–384.

[14] Dario Floreano, Peter Dürr, and Claudio Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intell.*, vol. 1, no. 1, pp. 47–62, 2008.

[15] Barret Zoph and Quoc V. Le, "Neural architecture search with reinforcement learning," in *Proc. of the Int. Conf. on Learn. Representations*, 2017.

[16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, "Binarized neural networks," in *Adv. in Neural Inf. Process. Syst.*, 2016, pp. 4107–4115.

[17] Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein, "Training quantized nets: A deeper understanding," in *Adv. in Neural Inf. Process. Syst.*, 2017, pp. 5811–5821.

[18] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, "Pruning filters for efficient convnets," in *Proc. of the Int. Conf. on Learn. Representations*, 2017.

[19] Yihui He, Xiangyu Zhang, and Jian Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. of the IEEE Int. Conf. on Comput. Vision*, 2017, pp. 1389–1397.

[20] Yann LeCun, John S. Denker, and Sara A. Solla, "Optimal brain damage," in *Adv. in Neural Inf. Process. Syst.*, 1990, pp. 598–605.

[21] Babak Hassibi and David G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Adv. in Neural Inf. Process. Syst.*, 1993, pp. 164–171.

[22] H. Wold, *Partial Least Squares*, pp. 581–591, John Wiley & Sons, 2006.
[23] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.
[24] Chunhui Jiang, Guiying Li, Chao Qian, and Ke Tang, "Efficient DNN neuron pruning by minimizing layer-wise nonlinear reconstruction error," in *Proc. of the Int. Joint Conf. on Artif. Intell.* AAAI Press, 2018, pp. 2298–2304.
[25] Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann, "Learning to prune filters in convolutional neural networks," in *Proc. of the IEEE Winter Conf. on Appl. of Comput. Vision*. IEEE, 2018, pp. 709–718.
[26] Jonathan Frankle and Michael Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. of the Int. Conf. on Learn. Representations*, 2019.
[27] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung, "Structured pruning of deep convolutional neural networks," *ACM J. on Emerging Technol. in Comput. Syst.*, vol. 13, no. 3, pp. 32, 2017.
[28] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J Dally, "Exploring the granularity of sparsity in convolutional neural networks," in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition Workshops*, 2017, pp. 13–20.
[29] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, "ThiNet: A gilter level pruning method for deep neural network compression," in *Proc. of the IEEE Int. Conf. on Comp. Vision*, 2017, pp. 5058–5066.
[30] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition*, 2018, pp. 9194–9203.
[31] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Adv. in Neural Inf. Process. Syst.*, 2018, pp. 875–886.
[32] William Robson Schwartz, Aniruddha Kembhavi, David Harwood, and Larry S Davis, "Human detection using partial least squares analysis," in *Proc. of the IEEE Int. Conf. on Comp. Vision*, 2009, pp. 24–31.
[33] Artur Jordao, Ricardo Kloss, Fernando Yamada, and William Robson Schwartz, "Pruning deep neural networks using partial least squares," in *Brit. Mach. Vision Conf. Workshops: Embedded AI for Real-Time Machine Vision*, 2019.
[34] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama, "Adaptive neural networks for efficient inference," in *Proc. of the Int. Conf. on Mach. Learn.*, 2017, pp. 527–536.
[35] Dimitrios Stamoulis, Ting-Wu Rudy Chin, Anand Krishnan Prakash, Haocheng Fang, Sribhuvan Sajja, Mitchell Bognar, and Diana Marculescu, "Designing adaptive neural networks for energy-constrained image classification," in *Proc. of the Int. Conf. on Computer-Aided Des.* ACM, 2018, p. 23.
[36] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition*, 2018, pp. 8817–8826.
[37] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *Proc. of the Eur. Conf. on Comput. Vision*, 2018, pp. 409–424.
[38] Andreas Veit and Serge Belongie, "Convolutional networks with adaptive inference graphs," in *Proc. of the Eur. Conf. on Comput. Vision*, 2018, pp. 3–18.
[39] Mohammad Saeed Shafiee, Mohammad Javad Shafiee, and Alexander Wong, "Efficient inference on deep neural networks by dynamic representations and decision gates," *arXiv preprint arXiv:1811.01476*, 2018.
[40] Tahir Mehmood, Kristian Hovde Liland, Lars Snipen, and Solve Sæbø, "A review of variable selection methods in partial least squares regression," *Chemometrics and Intell. Lab. Syst.*, vol. 118, pp. 62–69, 2012.
[41] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., University of Toronto, 2009.
[42] Guodong Guo and Guowang Mu, "Simultaneous dimensionality reduction and human age estimation via kernel partial least squares regression," in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition*. IEEE, 2011, pp. 657–664.
[43] Guodong Guo and Guowang Mu, "Joint estimation of age, gender and ethnicity: CCA vs. PLS," in *Proc. of the IEEE Int. Conf. and Workshops on Autom. Face and Gesture Recognition*. IEEE, 2013, pp. 1–6.
[44] Mengyi Liu, Ruiping Wang, Zhiwu Huang, Shiguang Shan, and Xilin Chen, "Partial least squares regression on grassmannian manifold for emotion recognition," in *Proc. of the ACM Int. Conf. on Multimodal Interact.* ACM, 2013, pp. 525–530.
[45] Giorgio Roffo, Simone Melzi, and Marco Cristani, "Infinite feature selection," in *Proc. of the IEEE Int. Conf. on Comput. Vision*, 2015, pp. 4202–4210.
[46] Giorgio Roffo, Simone Melzi, Umberto Castellani, and Alessandro Vinciarelli, "Infinite latent feature selection: A probabilistic latent graph-based ranking approach," in *Proc. of the IEEE Int. Conf. on Comput. Vision*, 2017, pp. 1398–1406.
[47] Nouna Kettaneh, Anders Berglund, and Svante Wold, "PCA and PLS with very large data sets," *Comput. Statist. & Data Anal.*, vol. 48, no. 1, pp. 69–85, 2005.
[48] Deepak Mittal, Shweta Bhardwaj, Mitesh M. Khapra, and Balaraman Ravindran, "Recovering from random pruning: On the plasticity of deep convolutional neural networks," in *IEEE Winter Conf. on Appl. of Comput. Vision*. IEEE, 2018, pp. 848–857.
[49] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," in *Adv. in Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.
[50] Shuochao Yao, Yiran Zhao, Huajie Shao, ShengZhong Liu, Dongxin Liu, Lu Su, and Tarek Abdelzaher, "FastDeepIoT: Towards understanding and optimizing neural network execution time on mobile and embedded devices," in *Proc. of the ACM Conf. on Embedded Network. Sensor Syst.* ACM, 2018, pp. 278–291.
[51] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[52] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *Proc. of the Eur. Conf. on Comput. Vision*, 2018, pp. 784–800.

Artur Jordao received the B.Sc. degree in computer science from the University of Western São Paulo, Presidente Prudente, Brazil, and the M.Sc. degree in computer science from the Federal University of Minas Gerais, Belo Horizonte, Brazil, where he is currently working towards the Ph.D degree in computer science.

His research interests include machine learning and pattern recognition focused on computer vision applications.

Maiko Lie received the B.Sc. and M.Sc. degrees in computer engineering from the Federal University of Technology – Paraná, Curitiba, Brazil. He is currently working towards the Ph.D degree in computer science in the Federal University of Minas Gerais, Belo Horizonte, Brazil.

His research interests include pattern recognition, computer vision, and image processing.

William Robson Schwartz received the B.Sc. and M.Sc. degrees in computer science from Federal University of Paraná, Curitiba, Brazil, and the Ph.D. degree in computer science from the University of Maryland, College Park, MD, USA.

He is a Professor with the Department of Computer Science, Federal University of Minas Gerais, Belo Horizonte, Brazil and associate editor for the IEEE Transactions on Information Forensics and Security. His research interests include computer vision, computer forensics, biometrics, and image processing.