Neural Networks 169 (2024) 713–732

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

Filter pruning for convolutional neural networks in semantic image segmentation

Clara I. López-González [a,*], Esther Gascó [a], Fredy Barrientos-Espillco [b], Eva Besada-Portas [b], Gonzalo Pajares [c]

[a] Department of Software Engineering and Artificial Intelligence, Complutense University of Madrid, Madrid, 28040, Spain
[b] Department of Computer Architecture and Automation, Complutense University of Madrid, Madrid, 28040, Spain
[c] Institute for Knowledge Technology, Complutense University of Madrid, Madrid, 28040, Spain

ARTICLE INFO

Keywords:
Convolutional Neural Networks (CNNs)
Explainable Artificial Intelligence (xAI)
Filter pruning
Image segmentation
Model compression
Principal Component Analysis (PCA)

ABSTRACT

The remarkable performance of Convolutional Neural Networks (CNNs) has increased their use in real-time systems and devices with limited resources. Hence, compacting these networks while preserving accuracy has become necessary, leading to multiple compression methods. However, the majority require intensive iterative procedures and do not delve into the influence of the used data. To overcome these issues, this paper presents several contributions, framed in the context of explainable Artificial Intelligence (xAI): (a) two filter pruning methods for CNNs, which remove the less significant convolutional kernels; (b) a fine-tuning strategy to recover generalization; (c) a layer pruning approach for U-Net; and (d) an explanation of the relationship between performance and the used data. Filter and feature maps information are used in the pruning process: Principal Component Analysis (PCA) is combined with a next-convolution influence-metric, while the latter and the mean standard deviation are used in an importance score distribution-based method. The developed strategies are generic, and therefore applicable to different models. Experiments demonstrating their effectiveness are conducted over distinct CNNs and datasets, focusing mainly on semantic segmentation (using U-Net, DeepLabv3+, SegNet, and VGG-16 as highly representative models). Pruned U-Net on agricultural benchmarks achieves 98.7% parameters and 97.5% FLOPs drop, with a 0.35% gain in accuracy. DeepLabv3+ and SegNet on CamVid reach 46.5% and 72.4% parameters reduction and a 51.9% and 83.6% FLOPs drop respectively, with almost no decrease in accuracy. VGG-16 on CIFAR-10 obtains up to 86.5% parameter and 82.2% FLOPs decrease with a 0.78% accuracy gain.

1. Introduction

In the last decades, the progress in computational capacity has led to an increasing interest in deep neural networks, which have proven to be incredibly successful in many different tasks. Convolutional neural networks (CNNs) have achieved state-of-the-art performance in image classification (He et al., 2016; Simonyan & Zisserman, 2015), object detection (Law & Deng, 2020), semantic segmentation (Lou & Loew, 2021; Ronneberger et al., 2015), among others (Hinton et al., 2012; Zhang et al., 2018). Besides, they go beyond traditional methods (Arthur & Vassilvitskii, 2006; Hinton et al., 2011, 2006; Krizhevsky et al., 2012), which perform well on certain types of images, but are not able to reach the same generalization capabilities.

Due to the growth of computer power, there is a trend to enlarge convolutional neural networks in width and depth, resulting in an overall rise in the number of their parameters and operations (He et al., 2016; Simonyan & Zisserman, 2015). This makes extremely difficult to use these networks on small devices with limited resource supply, such as on-board systems for autonomous vehicles or mobile devices, where real time computation is essential for navigation and detection. It is necessary to start from well-tested models and then apply compression techniques to achieve their reduction without loss of efficiency, justifying the need for pruning.

Furthermore, training or transfer learning of well known CNN models often yield over-parametrized networks with large redundancy in the number of feature maps, unnecessary computation, and excessive memory usage, as well as affecting the overall performance (Denil et al., 2013).

In order to solve these problems, many model compression techniques have been developed in the last few years. Results show that

* Corresponding author.
E-mail addresses: claraisl@ucm.es (C.I. López-González), egasco@ucm.es (E. Gascó), fredybar@ucm.es (F. Barrientos-Espillco), ebesada@ucm.es (E. Besada-Portas), pajares@ucm.es (G. Pajares).

https://doi.org/10.1016/j.neunet.2023.11.010
Received 22 June 2023; Received in revised form 1 October 2023; Accepted 5 November 2023
Available online 7 November 2023

pruning can be performed with negligible accuracy loss (Hassibi & Stork, 1992; LeCun et al., 1989). When it comes to CNNs they focus on convolutional layers, since most of the computations are performed there. Methods that remove single unimportant weights result in non-structured sparse models, which require specific software and hardware (Han et al., 2016, 2015). On the contrary, filter pruning approaches delete the whole convolutional kernel, with a direct decrease in computation, memory, and number of parameters (He et al., 2019; Li et al., 2017, 2020; Shao et al., 2021). Moreover, since the structure remains the same, sparse libraries are not needed.

The main objective of this paper is to design filter pruning strategies for complex CNN models. It is oriented towards, but not limited to, CNN-based semantic segmentation in images, where each pixel is classified as belonging to a specific category. Compression has not been widely evaluated for this task, nor has a detailed analysis been carried out in combination with classification as presented in this paper, adding value to our work. We hypothesize that pruning can increase computational performance and decrease redundancy without reducing accuracy. Indeed, the proposed methods exhibit efficiency not only with semantic segmentation models, such as U-Net (of which there is nearly no pruning analysis), DeepLab, and SegNet, but also with classification networks as VGG-16, proving their generalization ability.

We exploit the criterion of calculating filters' importance scores and pruning those with lower values, using in our case statistics of its own and next layer, and combining feature maps and filter weights knowledge. Two methods are used to get the number of channels to be deleted without several iterations: Principal Component Analysis (PCA) and a new proposal, based on the distribution of importance scores. These methods, combined with the introduced importance metrics, give rise to the novel presented strategies. Moreover, the methodology developed allows extending them to define a U-Net layer pruning, and to our knowledge the first of its kind. Besides, a fine-tuning procedure, to recover the generalization power of the original CNN, is presented, adding another dimension to our contributions. Finally, it is worth noting that, although PCA has already been used in the first step of the low effort technique developed by Garg et al. (2020), unlike us they consider all filters equally important and do not retain previously learned weights, in addition to not being a suitable method for complex convolutional networks.

Although compression methods have been widely studied, less attention has been paid to the dependency of their performance on the dataset and to how they help to understand networks learning. Recent strategies such as Liu et al. (2021) and He et al. (2022), based on measuring redundancy through discriminative-aware losses and a receptive field criterion respectively, perform a visualization analysis to corroborate their proposal but do not study how they are influenced by the type of used data. Moreover, no evaluation on semantic segmentation models is performed. This also motivates this paper, which explores that influence, obtaining information about the process of decision making. By doing this, model compression connects with explainable artificial intelligence (xAI, Tjoa & Guan, 2021). The use of different datasets allows us to test the validity of the proposed global strategy as well. The obtained results support the previous hypotheses.

The major contributions and novelties of this article are the following:

1. The design of new effective CNNs filter pruning strategies (supported by known but efficient techniques) and of a fine-tuning strategy, for both segmentation and classification models.
2. The improvement of our approaches with the inclusion of a novel depth pruning method for U-Net, a relevant semantic segmentation model with a encoder–decoder architecture with cross layer connections from which the new extension benefits (Ronneberger et al., 2015).
3. A detailed analysis of the pruning processes and the compressed networks performance, in terms of evaluation metrics, pruned parameters and reduction of floating point operations (FLOPs[1]). The latter are compared with those of other existing methods in the literature. To the best of our knowledge, this is the first time an exhaustive pruning survey is done at the same time on encoder–decoder, with special interest in U-Net, and classification models.
4. An in-depth study on the dependency of pruned models performance on different datasets. We show that pruning can help, not only to reduce computational cost, but also to understand learning models and their behavior on different datasets (connecting our approach to xAI).

The paper is organized as follows. Section 2 presents a revision of compression-based methods, describing their pros and cons. Section 3 focuses on the first two main contributions of this paper and presents our filter global pruning methodology. Section 4 addresses the third contribution, analyzing and discussing the pruning results. Section 5 delves into the performance of the approaches, taking care of the fourth contribution. Finally, Section 6 draws the conclusions and introduces some future work.

2. Related work

The development of compression methods to accelerate CNNs is focused mainly on classification models. It began by the mid-1990s, with works by LeCun et al. (1989), and Hassibi and Stork (1992), which remove unimportant weights based on second derivatives of the objective function. A more recent and simple pruning technique by Han et al. (2015) deletes all neuron connections whose weight values are below a threshold. However, pruning individual weights results in sparse networks that require special hardware and software, as Han et al. (2016) shows.

The structure of CNNs allows pruning the entire convolutional kernel, and hence all its connections, avoiding sparsity. This is mainly done by removing irrelevant kernels, the method on which this paper is based. The importance metric used to decide which kernel to prune can be defined by the magnitude of the filters. A simple approach, given by Li et al. (2017), deletes the filters whose weights have a low l_1-norm magnitude. The techniques by He et al. (2019) and Ayinde et al. (2019) consider redundancy by using respectively the geometric median estimator (to obtain the common information of all the filters within a layer) and the relative cosine distance. However, unlike us, previous approaches do not take into consideration importance scores distribution. It is also possible to identify relevant channels during training, as Liu et al. (2017) do, where the importance is defined through scaling factors, which are trained with networks weights adding a sparsity regularization to the objective function. The main drawback of this last approach is that new hyperparameters arise, increasing the training time. Although important, filter weight information may not be sufficient for intensive tasks such as segmentation, where contextual information is essential (He, Wu, Liang, & Lam, 2021).

Data dependent strategies consider the information gather in feature maps. The approach by Shao et al. (2021) uses information entropy of convolutional output feature maps to measure the importance of the filters. Li et al. (2020) propose a combination between the information diversity of each feature map and the similarity between them. Instead of using knowledge from the feature maps exclusively, we extract information from the filter weights and the feature maps and combine them. Combination, as Li et al. (2020) suggests, is an interesting concept that we exploit, and the use of both sources of information combined results in a novelty of interest.

[1] We only consider FLOPs of convolutional operations, including multiplication and addition, as it is commonly used for comparison.

Among the data dependent methods, reconstruction based ones turn filter selection into an optimization problem by pruning those filters that minimize the reconstruction error of feature maps. Luo et al. (2017) obtain this error from the feature maps of next layer. However, reconstructed maps may have redundant information and require iterative processes, in contrast with our strategy. The proposed next convolution influence-based importance metric also considers the following convolutional layer. In this way, the pruning of the current layer is not affected by the harmful pruning of previous layers. Besides, a novel depth pruning method can be proposed in U-Net architectures, as will be shown in Section 3.6.2.

Pruning rates are in general specified by the user (He et al., 2019; Li et al., 2017). Alternatively, Singh et al. (2019) suggest a min–max game, where compression rates are chosen adaptively and the tolerance in accuracy drop guides the pruning. Although it minimizes the number of filters, it maximizes the accuracy of the pruned network, and its flexibility gain requires an increment of iterations. This is a widespread problem in pruning techniques, which require time intensive study or iterative processes. To overcome this, Garg et al. (2020) present a low effort approach that performs PCA over the feature maps of each convolutional layer to obtain its "optimal" dimension, although is not applicable to networks with shortcut connections. After this process, a new network architecture is defined, reducing the number of filters without iterative retraining procedures. Since the obtained model is not necessarily the optimal one, this compression method may be followed by others such as the l_1-norm criterion as introduced by Zhang and Wang (2022), or the geometric median criterion as proposed by Ahmed et al. (2022). Our approach, in contrast to the previous ones, is not a concatenation of methods but a single method that benefits from PCA and combines it with a next convolution influence-based metric. In this way, it simultaneously uses the information gathered in feature maps and filter weights, and takes advantage of the previously learned weights. In addition, complex convolutional neural networks can also benefit from this pruning.

Recently, some pruning strategies for semantic segmentation networks have emerged, reducing the gap with respect to classification models. The work of He, Wu, Liang, and Lam (2021), based on contextual information, is expanded in He, Wu, and Lam (2021) to different vision tasks through an adaptive correlation-driven sparsity learning (ACSL), where channel correlation guides the pruning. Nevertheless, both strategies focus on batch normalization layers, needing to be adapted if these do not exist. A multi-task approach (MTP) is performed by Chen et al. (2022), adding classification and segmentation terms to the loss function. Another pruning approach during training, but without the drawbacks of additional loss functions such as the previous one, is presented by Ditschuneit and Otterbach (2022). They perform visualization analyses but, like those mentioned above, do not study the dependencies on the dataset as we do. Although the literature is rather sparse, they seem to agree on the fact that, due to the pixel-level complexity of semantic segmentation tasks, previous compression strategies cannot be straightforwardly applied. This would yield to performance degradation, as shown by He, Wu, Liang, and Lam (2021). Our work demonstrates that simple, but no less effective, techniques provide great pruning results in both segmentation and classification, presenting an exhaustive analysis. Moreover, despite the fact that U-Net is a large model that would benefit from compression, it is hardly taken into account in these types of studies (Dinsdale et al., 2022). The thorough analysis presented in this paper is therefore considered to be of great interest.

Table 1 classifies and summarizes the previously described pruning strategies, distinguishing among classification and segmentation approaches. A description is shown with comments pointing out their limitations and the improvements (in italics) included in our methods with regard to the others. Datasets are identified (if possible) because we take special account of the dependency of our models on them, which is a further improvement.

Next section contains a description of our pruning strategies, based on the information provided by the filters and the feature maps.

3. Filter pruning methodology

As mentioned before, in this section two pruning methods for convolutional neural networks are presented. As many other pruning strategies, they have three phases, which are sketched in Fig. 1: (1) starting with a pre-trained convolutional network, (2) pruning the convolutional layers according to the desired method, and (3) fine-tuning the resulting network in order to recover the performance of the original model. The overall approach shown in Fig. 1 is explained in Section 3.5, once all the parts have been described.

3.1. General framework (phase 2, Fig. 1)

Considering the advantages already mentioned, we focus on filter pruning. The basis of this strategy is to determine which filters of a convolutional layer are important, which filters are not, and to delete the latter ones.

Note that there is a one-to-one correspondence between the filters (convolutional kernels) of a convolutional layer and its output channels (as Fig. 2 schematizes), in the sense that each filter gives rise to an output channel. Hence, when the filter/channel is removed so is the corresponding output channel/filter, and the corresponding connections to the next layer.

Thus, given a convolutional layer, two operations are needed: determining the relevance of each channel and deciding the number of channels to be removed. Once they are done, we just need to delete as many channels as we have calculated, starting from the one with the lowest relevance score.

Next sections present a detailed description of the proposed approach based on the criteria expressed above. Our approach omits manipulating nonconvolutional layers, since their connections are one-to-one and removing a channel from the output of a convolution implies removing the same channel from all the following layers until we reach the next convolution.

3.2. Computing the channels relevance (box A, Fig. 1)

Let C_i be the ith convolutional layer. Let $x^i \in \mathbb{R}^{h_i \times w_i \times n_i}$ be its output, which contains h_i rows, w_i columns and n_i channels. Hence, the weight tensor and the bias are given by $W^i \in \mathbb{R}^{k_i \times k_i \times n_{i-1} \times n_i}$ and $b^i \in \mathbb{R}^{n_i}$ respectively, where $k_i \times k_i$ is the kernel dimension, n_{i-1} the number of input channels, and n_i the number of filters or output channels.

In the following, two criteria are to be defined: next convolution influence and next convolution influence-variance.

3.2.1. Next convolution influence criterion

Its purpose is to remove those channels that have the least influence on the activation maps of the following convolutional layer C_{i+1}. If f is the $(i+1)$th convolutional operation, whose output is $x^{i+1} \in \mathbb{R}^{h_{i+1} \times w_{i+1} \times n_{i+1}}$, the influence of the jth channel of x^i is quantified by the norm of the corresponding partial derivatives $(D_q f_p)_{p,q}$. Here, $1 \leq p \leq h_{i+1} w_{i+1} n_{i+1}$ and D_q, with $1 \leq q \leq h_i w_i$, refers to the partial derivatives w.r.t the components of the jth channel, i.e., with respect to $x^i_q := x^i(\tilde{s}, \tilde{t}, j)$ for certain locations $(\tilde{s}, \tilde{t})$ of the jth activation map. Note that omitting nonlinearities, the convolutional operation gives rise to $x^{i+1} = x^i * W^{i+1}$, where $W^{i+1} \in \mathbb{R}^{k_{i+1} \times k_{i+1} \times n_i \times n_{i+1}}$. Hence, the component at locations s and t in channel c is obtained with

$$x^{i+1}(s,t,c) = \sum_m \sum_n \sum_{r=1}^{n_i} x^i(s+m, t+n, r) \cdot W^{i+1}(m,n,r,c) + b^{i+1}(c),$$

where m and n refer to the kernel dimension, so they belong to sets of k_{i+1} consecutive integers, depending on the concrete implementation details; and r runs over the channels of x^i. Therefore, the contribution of the jth channel of x^i ($r = j$ in the above equation) to the output feature maps x^{i+1} is given by $W^{i+1}(\cdot, \cdot, j, \cdot)$, which, from now on, will be treated as a vector in $\mathbb{R}^{k_{i+1}^2 n_{i+1}}$. Indeed, the following inequality is satisfied.

Table 1
Summary of comparisons of previous and proposed methods.

CNN type	Methods review	References	Dataset	Description	Limitations/*Improvements*
CLASSIFICATION	Remove unimportant weights	Hassibi and Stork (1992) LeCun et al. (1989)	–	• Based on second derivatives of the loss function • Measure the change caused by deleting weights	• Sparse pruned networks • Require special hardware and software
	Remove unimportant connections	Han et al. (2015)	MNIST ImageNet	• Connections with weights below a threshold	• Sparse pruned networks *Avoided by removing entire conv. kernels*
	Remove unimportant filters	Li et al. (2017)	CIFAR-10 ImageNet	• Filter weights with low l_1 norm • Prunes the lowest m filters	• Information only from current layer weights • Do not consider the importance distribution
	Remove redundant s filters	He et al. (2019) Ayinde et al. (2019)	CIFAR-10 MNIST ImageNet	• Obtain common filter's information within a layer • Uses the geometric median estimator • Or the relative cosine distance between weights	• Above limitations *Considers also feature maps information and importance distribution*
	Relevance during training	Liu et al. (2017)	CIFAR-10 MNIST ImageNet SVHN	• Filters relevance defined through scaling factors • Adds sparsity regularization to the loss function	• New hyperparameters arise • Increase of training time • Could lead to suboptimal solutions *No new trainable hyperparameters or long training*
	Adaptative compression rates	Singh et al. (2019)	CIFAR-10 ImageNet	• Min–max game to choose compression rates • Tolerance in accuracy drop guides the pruning	• Maximization of pruned networks accuracy • Increase of training iterations *No time intensive processes*
	Data dependent	Shao et al. (2021) Li et al. (2020)	CIFAR-10/100 ImageNet	• Relevance given by information entropy of features maps • Combines diversity and similarity-aware feature selection	• Do not consider filter weights information *Considers also filter weights information*
	Reconstruction based	Luo et al. (2017)	ImageNet CUB-200	• Optimization problem • Minimization of the reconstruction error	• Intensive iterations • Ignores channel discriminative power *No iterative processes* *Discriminative power is considered through importance scores*
	PCA based	Garg et al. (2020) Zhang and Wang (2022)	CIFAR10/100 ImageNet	• Compute layer dimensions via PCA of feature maps • Layer and depth pruning by defining a new architecture • Followed by a l_1 norm filter pruning	• Not suitable for networks with shortcut connections • No use of previously learned weights • Ignore channel discriminative power *Retains previously learned weights* *Suitable for complex networks*
	Discriminative based	He et al. (2022) Liu et al. (2021)	CIFAR-10/100 ImageNet LFW	• Filter's importance from a feature discriminative view • Use of discriminative-aware losses • Or receptive field criterion	• Do not delve into the influence of the dataset used • Intensive processes • No evaluations on semantic segmentation models *Analysis of the dependence on the dataset and FLOPs drop* *Evaluated also on semantic segmentation models*
SEMANTIC SEGEMENTATION	Contextual information	He, Wu, Liang, and Lam (2021) He, Wu, and Lam (2021)	CamVid Cityscapes Crowd counting CIFAR-100	• Pruning guided by contextual or channel correlation info • Context-aware guiding module and sparsification • Imposes penalty strength in loss • Adaptative sparsity-learning with a correlation-driven penalty	• Pruning focuses on batch normalization (BN) layers • Add hyperparameters *No need of adaptations if there are no BN layers* *No intensive processes and reduction of FLOPs*
	Multi-task	Chen et al. (2022)	Cityscapes Pascal Voc 2012 ADE20K	• Loss function with classification and segmentation terms • Based on l_1 norm	• Adds hyperparameters, increasing training time • Could lead to suboptimal solutions *No new trainable hyperparameters*
	Auto-compressing	Ditschuneit and Otterbach (2022)	CamVid Cityscapes Pascal Voc 2012 ADE20K	• Filter's importance reduced during training • Autocompression during training without additional loss terms	• No evaluations on classification models • Does not delve into the influence of the dataset *Evaluated also on classification models* *Analysis of the dependence on the dataset used*
	Medical image	Dinsdale et al. (2022)	–	• U-net simultaneously trained and pruned • Adaptative channelwise targeted dropout • Prunes by smallest l_2 norm of feature maps	• Only evaluated on U-Net • Intensive processes *Evaluated on other CNNs* *Presents also a layer pruning method*

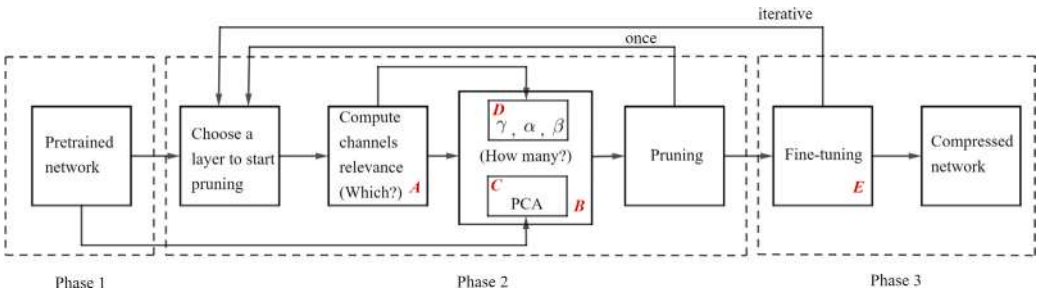

Fig. 1. Flowchart of the proposed pruning methods. On one hand, the backwards flow "once" applies when choosing the strategy of pruning at once all the convolutional layers and retraining later. On the other hand, "iterative" flow applies when using the pruning-fixing-retraining iterative strategy (see Section 3.4 for further details).

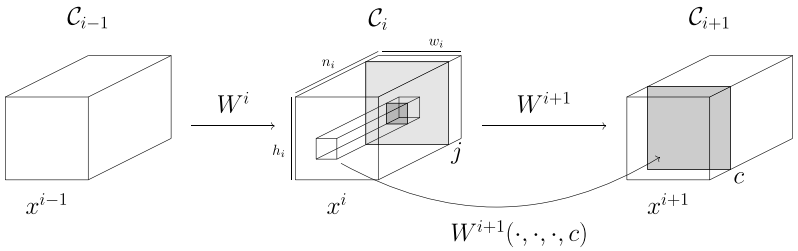

Fig. 2. Channel c of convolutional layer $\mathcal{C}_{i+1}$ is obtained by convolving the output $x^i \in \mathbb{R}^{h_i \times w_i \times n_i}$ of layer $\mathcal{C}_i$ by the c filter of layer $\mathcal{C}_{i+1}$, given by $W^{i+1}(\cdot,\cdot,\cdot,c) \in \mathbb{R}^{k_{i+1} \times k_{i+1} \times n_i}$.

Lemma 1. *Let us denote the $(i+1)$th convolutional operation by $f : \mathbb{R}^{h_i w_i n_i} \to \mathbb{R}^{h_{i+1} w_{i+1} n_{i+1}} = \mathbb{R}^L$. That is, $f(x^i) = (f_1(x^i), \dots, f_L(x^i)) = x^{i+1} = x^i * W^{i+1}$. Let us define the influence of the jth channel as the l_1-norm of the partial derivatives $\left\|(D_q f_p)_{p,q}\right\|_1$, where $1 \le p \le L$ and where q runs over the components corresponding with the jth channel of the input. Then, this influence is bounded above by the l_1-norm of $W^{i+1}(\cdot,\cdot,j,\cdot)$,*

$$\left\|\left(D_q f_p\right)_{p,q}\right\|_1 \le \left\|W^{i+1}(\cdot,\cdot,j,\cdot)\right\|_1.$$

Proof. Due to the definition of the l_1-norm[2] of a matrix, we focus on the column vectors. That is, fixing q we consider

$$\mathbf{v}_q := \left(D_q f_1(x^i), \dots, D_q f_L(x^i)\right)^\top.$$

Note that, for every p, $f_p(x^i) = x^{i+1}(s,t,c)$ for certain locations s, t and channel c, and that q corresponds with some index $(\tilde{s},\tilde{t},j)$. That is, $x^i_q := x^i(\tilde{s},\tilde{t},j)$. Then,

$$f_p(x^i) = \sum_m \sum_n \sum_{r=1}^{n_i} x^i(s+m, t+n, r) W^{i+1}(m,n,r,c) + b^{i+1}(c),$$

which makes $D_q f_p(x^i)$ vanish when $s+m \neq \tilde{s}$, $t+n \neq \tilde{t}$, or $r \neq j$. Otherwise, $D_q f_p(x^i) = W^{i+1}(\tilde{s}-s, \tilde{t}-t, j, c)$ whenever the right side makes sense. Therefore, $\left\|\mathbf{v}_q\right\|_1 \le \left\|W^{i+1}(\cdot,\cdot,j,\cdot)\right\|_1$ for any q and

$$\left\|\left(D_q f_p\right)_{p,q}\right\|_1 = \max_q \left\|\mathbf{v}_q\right\|_1 \le \left\|W^{i+1}(\cdot,\cdot,j,\cdot)\right\|_1. \quad \square$$

As a result, a weight vector with low l_1-norm implies that the jth channel of layer $\mathcal{C}_i$ has little influence on the output of $\mathcal{C}_{i+1}$.[3] For this reason, and for its simplicity, the l_1-norm of the weight vector associated to the next convolution is proposed to measure the relevance of a channel. Hence, given a convolutional layer $\mathcal{C}_i$, we assign to each channel j the importance score

$$l^i_j := \left\|W^{i+1}(\cdot,\cdot,j,\cdot)\right\|_1, \qquad 1 \le j \le n_i, \tag{1}$$

being those channels with the lowest l^i_j the least relevant. In other words, we keep those channels whose connections with layer $\mathcal{C}_{i+1}$ are more active.[4]

It is worth noting that this is not the standard norm criterion used in other compression methods. In our case, the pruning of layer i is not guided by the same layer as in Li et al. (2017) but by layer $i+1$. One of the advantages of adopting this point of view is that the pruning of layer $\mathcal{C}_i$ is not affected by that of previous layers. This means that any adverse effect on the pruning of preceding layers does not impact that of $\mathcal{C}_i$, which results in an important conceptual contribution. In addition, choosing the next convolutional layer does not alter the computational complexity w.r.t Li et al. (2017), as illustrated in Appendix A.

3.2.2. Next convolution influence-variance criterion

Hypothesizing that homogeneous channels in a convolutional layer do not collect/provide relevant information from/to the previous/next layer, we alternatively propose to add channel's variance to the calculation of their importance. The combination with the previous criterion leads to the least important channels being those that have little influence on the output of the following convolution and, moreover, have small variance.

Let consider a dataset with N images. Let $x^{n,i}(\cdot,j) \in \mathbb{R}^{h_i w_i}$ be the jth feature map of the ith convolutional layer $\mathcal{C}_i$ of the nth image, seen as a vector, with $h_i \cdot w_i$ pixels. Define the variance of this feature map as

$$(\sigma^n_{i,j})^2 = \frac{1}{h_i w_i - 1} \sum_{q=1}^{h_i w_i} \left(x^{n,i}(q,j) - \overline{x}^{n,i}_j\right)^2, \text{ with } \overline{x}^{n,i}_j = \frac{1}{h_i w_i} \sum_{q=1}^{h_i w_i} x^{n,i}(q,j).$$

We compute the mean standard deviation for all the images in the data set

$$s^i_j = \frac{1}{N} \sum_{n=1}^{N} \sigma^n_{i,j}. \tag{2}$$

Note that a small score s^i_j indicates that the standard deviation of the jth feature map is small in general. Hence, there is no significant

[2] Given a matrix A with m rows and n columns, $\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^m \left|a_{ij}\right|$.

[3] Note that this is still true if we consider nonlinearities such as ReLU or max-pooling, since their derivatives are 1 or 0.

[4] An analogous reasoning can be carried out in the case of transposed convolutional layers, included in the decoder part of some semantic segmentation models. In that case $W^{i+1} \in \mathbb{R}^{k_{i+1} \times k_{i+1} \times n_{i+1} \times n_i}$, so the desired weight vector is given by $W^{i+1}(\cdot,\cdot,\cdot,j) \in \mathbb{R}^{k^2_{i+1} n_{i+1}}$.

variance in its pixel values throughout the dataset, which we assume implies that the feature map is not storing relevant information and can be deleted.

When pruning a convolutional layer C_i, we propose to combine (1) and (2) in order to measure the relevance of each channel with the score ls^i_j defined by

$$ls^i_j := l^i_j \cdot s^i_j, \qquad 1 \le j \le n_i. \tag{3}$$

As mentioned before, this combination strategy makes another important contribution. The product is the chosen method in order to ensure that if one of the values of the previous criteria is zero the channel is removed. Besides, how much effort it needs to compute this score w.r.t the previous one is illustrated in Appendix A.

We recall that the basis of our strategy is to prune those channels with the lowest importance scores. Therefore, we are not so much interested in the values of the scores themselves, but in their (ascending) order. The scale of l^i_j and s^i_j may be unified before multiplication in (3) by dividing by the maximum. Given that it is positive in both cases, the order of both scores is maintained after scaling. However, this is not necessary, since the order of $\{ls^i_j\}_{j=1}^{n_i}$ and the order of the product of the scaled scores is the same.[5] This implies that the same answer to the question *Which channels to prune?* (box A, Fig. 1) is obtained in both cases. As for the *How many channels to prune?* (box B, Fig. 1), we will show in Section 3.3.2 that there is no need for scaling scores either. This will prove that the same pruning is achieved with and without adjusting the scale.

3.3. Determining the number of channels to prune (box B, Fig. 1)

Let C_i be the ith convolutional layer as before. In the following, we present two streamlined strategies to determine the number of channels to prune within C_i, resulting in layer-specified pruning rates based on channel importance or feature map information, which adds flexibility to our methods. Consequently, our approach itself controls the compression rates.

3.3.1. PCA (box C, Fig. 1)

Principal component analysis is a dimensionality reduction technique used to interpret, visualize and obtain the "most relevant information" of high dimensional data, determining the projection subspace that minimizes the dispersion of the original data. Moreover, dimensions are ranked based on the amount of variance of the input data. Therefore, this technique allows us to collect as much information as possible about the originally distributed data and to remove redundancy between correlated data.

As a consequence, PCA has been recently used to reduce the redundancy present in neural networks by deleting highly correlated channels (Ahmed et al., 2022; Garg et al., 2020). After one forward pass we get the feature maps $\mathrm{X}^i \in \mathbb{R}^{h_i \times w_i \times n_i \times B}$, where n_i is the number of channels and B is the mini-batch size. By flattening this tensor we get a matrix with $h_i w_i B$ rows and n_i columns. According to the width optimization strategy developed in Garg et al. (2020, Algorithm 1), in order to detect redundancy it is enough to repeat this procedure to collect data as many times as $T = \lceil 100 n_i/(h_i w_i B) \rceil$.

On the whole, we end up with a matrix with n_i columns and $h_i w_i BT$ rows. Seeing each row as a n_i-dimensional vector, we perform PCA. Each eigenvalue obtained explains a ratio of the total variance. Starting from the largest to the smallest, the number of eigenvalues required to explain the 99,9% of the cumulative variance will be considered as the optimal dimension of the convolutional layer. This gives us the number of channels to delete.

[5] Let $M_l = \max_j l^i_j > 0$ and $M_s = \max_j s^i_j > 0$. Denote the scaled scores by $\tilde{l}^i_j = l^i_j/M_l$ and $\tilde{s}^i_j = s^i_j/M_s$. Then, $ls^i_j \le ls^i_k \Leftrightarrow l^i_j \cdot s^i_j \le l^i_k \cdot s^i_k \Leftrightarrow l^i_j \cdot s^i_j/M_l M_s \le l^i_k \cdot s^i_k/M_l M_s \Leftrightarrow \tilde{l}^i_j \cdot \tilde{s}^i_j \le \tilde{l}^i_k \cdot \tilde{s}^i_k$.

Note that although PCA is a low-effort approach, it has limitations in terms of the percentage of pruned parameters. With this method we can only vary the proportion of total variance preserved, and PCA is responsible for determining how many parameters will be pruned, which may not always lead to an adequate result for our purposes.

3.3.2. Distribution of channels importance score (box D, Fig. 1)

When the dimensions of the feature maps are small (e.g. 4×4), a change in a single pixel is significant, and we may need several channels to explain the required cumulative variance, resulting in poor pruning. However, not all those channels might be important for the final decision. For this reason, we develop a method that considers the relevance of each of them through their importance score when deciding how many to prune.

In more detail, let $\{a^i_j\}_{j=1}^{n_i}$ be the importance scores of the channels of a layer C_i, where n_i is the total number of channels. We assume w.l.o.g that the channels are sorted in ascending order according to their relevance, i.e., $a^i_j \le a^i_k$ whether $j \le k$. Note that then the kth element does not necessarily correspond to the kth channel. Let $m = \min_j a^i_j$, $M = \max_j a^i_j$, and consider the interval $[m, M]$. We map this interval to $[0, 1]$ through the function

$$g : [m, M] \to [0, 1]; \quad z \mapsto g(z) = (z - m)/(M - m),$$

which satisfies $a^i_j \le a^i_k$ if and only if $g(a^i_j) \le g(a^i_k)$ for all $j, k \in \{1, \dots, n_i\}$. Thus, the order is maintained.

We could simply choose $\gamma \in [0, 1]$ and, for all layers in the network, remove those channels j such that $g(a^i_j) \le \gamma$ (i.e., such that $a^i_j \le m + (M - m)\gamma$). Formally, this can be expressed as pruning the first d^i_γ sorted channels, where

$$d^i_\gamma := \max\{d \in \mathbb{N} \ : \ 1 \le d \le n_i, \ g(a^i_d) \le \gamma\}. \tag{4}$$

Nevertheless, by following this procedure we are not considering the distribution of $\{g(a^i_j)\}_{j=1}^{n_i}$, which is crucial in order to pick the right value of γ. On one hand, there may be some layers with low scores and numerous values concentrated around zero. In those cases, even a small γ may produce an excessive pruning. On the other hand, there could be layers with multiple high scores accumulated around one, where even a high γ value could not prune enough channels.

To solve this problem, we define two more thresholds that are related to the maximum and minimum number of channels to be pruned. Let A^i be the sum of all C_i importance scores and A^i_k be the cumulative sum up to the kth score, that is,

$$A^i := \sum_{j=1}^{n_i} g(a^i_j), \qquad A^i_k := \sum_{j=1}^{k} g(a^i_j). \tag{5}$$

Since $\{g(a^i_j)\}_{j=1}^{n_i}$ is sorted in ascending order of importance values, A^i_k starts from the least relevant channel and consists of the sum of the scores of the least important k channels.

Let $\alpha \in [0, 1]$ and define

$$k^i_\alpha := \max\{k \in \mathbb{N} : 1 \le k \le n_i, \ A^i_k \le \alpha A^i\}, \tag{6}$$

where n_i is the total number of channels in C_i. That is, k^i_α is the maximum number of channels (starting from the least relevant) such that the cumulative sum of their importance scores does not exceed the α percentage of the total layer score. We choose two thresholds $\alpha, \beta \in [0, 1]$ as the maximum and the minimum, respectively, that we are willing to afford for any layer C_i, in the sense that we want to prune no more than k^i_α channels neither less than k^i_β, where both are given by Eq. (6). Fix a value of $\gamma \in [0, 1]$ from before and obtain d^i_γ from Eq. (4). As explained above, d^i_γ is the number of sorted channels whose importance score is less than γ. Then, given C_i the convolutional layer, we proceed as follows:

- If $k^i_\beta \le d^i_\gamma \le k^i_\alpha$ we remove the first d^i_γ sorted (in ascending order according to the importance scores) channels.

- If $k^i_\beta \leq k^i_\alpha \leq d^i_\gamma$ we remove the first k^i_α sorted channels.
- If $d^i_\gamma \leq k^i_\beta \leq k^i_\alpha$ we remove the first k^i_β sorted channels.

In any case, the most relevant channels, i.e., those with the highest importance scores, are kept in the layer.

This procedure ensures that, in any convolutional layer C_i, the sum of the importance scores of the channels that we eliminate is always almost equal to or greater than the β percentage of the total score A^i, but smaller than or equal to the α percentage. Keeping this in mind, it is not difficult to choose reasonable parameters so that neither too many or too few channels are pruned. Select γ as a first approximation of the maximum value of relevance that you are willing to disregard (e.g., based on the desired compression rates via g). Since the importance scores $\{g(a^i_j)\}_{j=1}^{n_i}$ are most likely not evenly distributed, one should not be too ambitious with γ. Then, pick α and β to control over- or under-pruning: the importance scores of the removed channels do not exceed the α percentage but is at least the β percentage. Besides, the same threshold values (γ, α, β) are used to prune all the network, but give rise to different pruning rates (d^i_γ, k^i_α or k^i_β) in each layer according to the channels importance distribution. In addition, this method does not depend on the chosen importance metric. Algorithm 1 summarizes this proposal.

Algorithm 1 Distribution based-method on layer C_i with next convolution influence-variance criterion.

Input: Layer outputs $\{x^{n,i}\}_n$, weight tensor W^{i+1} and thresholds (γ, α, β)
Output: The channels Y to be removed

1: Compute $\{ls^i_j\}_{j=1}^{n_i}$ from (3)
2: Sort the channels of C_i by increasing values of ls^i_j
3: Compute A^i using sorted $\{ls^i_j\}$ and (5)
4: Obtain k^i_α, k^i_β from (6) and d^i_γ from (4)
5: $\nu \leftarrow$ the one with the middle value from $d^i_\gamma, k^i_\alpha, k^i_\beta$
6: **return** $Y = \{$first ν sorted channels$\}$

As noted in Section 3.2.2, when using the next convolution influence-variance criterion to obtain the importance scores, a scaling may be considered. We show in the following that the proposed distribution-based method provides the same number of channels with and without scaling. In Section 3.2.2 it was explained that the increasing order of $\{ls^i_j\}$ and that of the product of the scaled scores is the same. Therefore, according to Algorithm 1, it is enough to check that the values of k^i_α, k^i_β and d^i_γ are equal in both cases. Denote the scaled score by $\tilde{ls}^i_j := l^i_j/M_l \cdot s^i_j/M_s$, where $M_l > 0$ and $M_s > 0$ are the corresponding maximums. It is easy to verify that $g(ls^i_j) = g(\tilde{ls}^i_j)$. Hence, the results of Eqs. (4), (5), and therefore (6), do not change, concluding that k^i_α, k^i_β and d^i_γ are the same with and without scaling.

3.4. Fine-tuning (box E, Fig. 1)

After pruning, a retraining phase is necessary to regain accuracy and recover generalization. The most common and obvious fine-tuning consists of pruning a layer and retraining the network before moving to the next one. Although this seems to be the right process, for large datasets and complex models it can take very long. Consequently, pruning all the layers at once and retraining later is more practical. However, this second strategy could require many epochs to recover the performance of the original network.

The fine-tuning strategy proposed in this paper is an intermediate solution that consists of pruning a layer and fixing the remaining weights, i.e., those that are assumed to be relevant for decision making. Then, retrain the network and move to the next layer. When all the network has been pruned, a global retraining may be performed.

On the one hand, our approach takes less time than pruning and retraining iteratively, since some weights are not updated. Besides, the number of fixed weights increases as pruning progresses. On the other hand, it retrains the network at every pruning step, not leaving all the fine-tuning until the end as pruning at once does. An analysis of the convergence at each step is presented in Appendix B. This intermediate solution provides another important contribution, as the retention of relevant weights allows much information to be extracted from the data set, which means that less data will be needed for model convergence. In fact, the information retained by the fixed weights provides more insights than other strategies on whether or not the selection of important channels was correct, as exploited in Section 5.3.

3.5. Complete pruning strategies

The flowchart of our complete pruning strategies was already shown in Fig. 1. Given a pre-trained convolutional network, let us consider the first convolutional layer. In order to determine how many and which channels to remove, we combine PCA with the Next convolution influence criterion (calling this approach PCA-N here-after), while Next convolution influence-Variance is used to calculate the importance scores of our distribution-based method (calling it NV here-after). Regarding the scaling of the scores in the NV method, the explanations provided at the end of Sections 3.2.2 and 3.3.2 show that it is not necessary. Once the chosen channels are pruned, we retrain the resulting network and move to the next convolutional layer, or we move directly if pruning at once is performed.

In this way, both strategies leverage information from filters and feature maps. Even though data dependent, we only require a single forward pass. Besides, no layer-wise analysis is performed. It is worth mentioning that combining the next convolution influence criterion with PCA causes an increased in the computational time, since we need to add that of determining the number of channels via PCA. How much effort this implies is illustrated in Appendix A and compared with that of the NV method. Indeed, PCA-N is not always the fastest, so the raise in the computing time due to the variance in ls criterion is not necessarily reflected in the overall pruning strategy.

Note that with PCA we get the "optimal" dimension of each layer from the baseline network, knowing how many channels are going to be deleted before the pruning starts. Then, in each step, we determine which are those channels. On the contrary, in NV we find out the number of channels at each step after having obtained the resulting pruned model. This results in different pruning, not only between PCA-N and NV methods, but also depending on the latter's fine-tuning strategy (at once or iterative).

In order to identify the approaches, the term *it* is added after the method name (e.g., NV *it*) to indicate that the novel pruning-fixing-retraining strategy was used, while *once* is added to refer to the pruning at once method. Note that in Section 4 we will not present pruning-retraining approach results for the reasons stated in Section 3.4, and because it does not provide better results than the other strategies.

3.6. Extending the approach to complex neural networks

An enhanced approach is necessary when dealing with complex convolutional networks with cross layer connections. This section presents how to handle two commonly used complex architectures: U-Net (Ronneberger et al., 2015) and Residual Networks (ResNets, He et al., 2016). In the case of U-Net models, we extend our strategy with an in-depth pruning method.

3.6.1. Calculating the channels relevance

As explained in Section 3.2, to obtain channels importance scores of a convolutional layer C_i we focus on the weight tensor of the following convolution C_{i+1}. That is, the convolutional layer that has the output of C_i (x^i) as input (omitting nonlinearities). When a bifurcation appears in the network graph after C_i, there may be more than one convolutional layer whose input (or part of it) is x^i, and then more than one weight tensor must be evaluated to obtain the importance score.

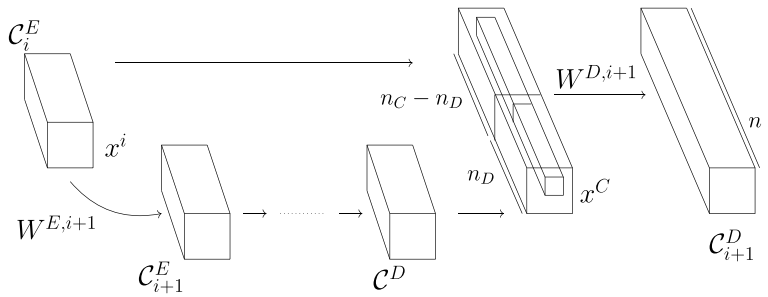

Fig. 3. Encoder–decoder (E–D) concatenation in U-Net. The output $x^C \in \mathbb{R}^{h_C \times w_C \times n_C}$ is just the concatenation of the output x^i of layer $\mathcal{C}_i^E$ and the one of layer $\mathcal{C}^D$. It has n_C channels, coming the first n_D from the decoder. These feature maps are the inputs of the next convolutional layer $\mathcal{C}_{i+1}^D$, with weight tensor $W^{D,i+1} \in \mathbb{R}^{k \times k \times n_C \times n}$.

This is the case in encoder–decoder (E–D) concatenations in U-Net. As shown in Fig. 3, the output x^i of layer $\mathcal{C}_i^E$ is part of the input of two convolutional layers, one in the downsampling path $\mathcal{C}_{i+1}^E$ and another one in the upsampling path $\mathcal{C}_{i+1}^D$. Due to the importance of these encoder–decoder connections to recover information from the downsampling path, we decide to measure the relevance of $\mathcal{C}_i^E$ channels according to their influence on the output maps of $\mathcal{C}_{i+1}^D$. Hence, following (1) and being $W^{D,i+1}$ the weight tensor of the convolutional layer $\mathcal{C}_{i+1}^D$, we compute $l_j^i = \left\| W^{D,i+1}(\cdot,\cdot,j,\cdot) \right\|_1$, where j runs through the channels received from the encoder. The rest of the weight tensor, i.e., $W^{D,i+1}(\cdot,\cdot,j',\cdot)$ where j' runs through the channels received from the decoder, is used as usual to prune layer $\mathcal{C}^D$ (according to Fig. 3).

This type of bifurcation problem also appears when dealing with residual blocks on ResNets. However, in this case it is not clear which path of the graph is the most important. Moreover, the presence of additions instead of concatenations complicates the pruning, since the two inputs need to have the same number of channels. For this reason, no pruning is performed in the layers preceding those bifurcations. Hence, inside residual blocks, we prune all but the last convolutional layer, keeping the output dimension unchanged.

3.6.2. Depth pruning on U-Net

In this section, we present a layer pruning method based on the encoder–decoder concatenations of U-Net networks. It checks which side is more relevant, whether the one received from the encoder or from the decoder. The hypothesis is that, if the former occurs, we can exclude the layers between the first and second input of the concatenation.

Let $x^C \in \mathbb{R}^{h_C \times w_C \times n_C}$ be the output of a concatenation, which has n_C feature maps. Suppose that the first n_D come from the decoder, while the rest ones come from the encoder, as shown in Fig. 3. Let $W \in \mathbb{R}^{k \times k \times n_C \times n}$ be the weight tensor of the convolutional layer that follows the concatenation. Depending on whether it acts on the portion received from the decoder or from the encoder, it can be divided into two parts, $W^D \in \mathbb{R}^{k \times k \times n_D \times n}$ and $W^E \in \mathbb{R}^{k \times k \times (n_C - n_D) \times n}$, being

$$W^D(a,b,j,l) = W(a,b,j,l), \quad W^E(a,b,j',l) = W(a,b,j',l),$$

where $1 \le a,b \le k$; $1 \le j \le n_D$; $n_D < j' \le n_C$; and $1 \le l \le n$. The magnitude of these weights determines which side is more relevant for the network. Hence, we get the l_2-norm of W^E and W^D, seen as vectors, and divide the former by the latter (i.e., we compute l_2^E / l_2^D). If we obtain a high value (trial and error proves that a threshold of 100 is enough), the layers between the inputs of the concatenation could be removed. Before doing it, we check if these layers are not contributing to decision making according to the optimal dimensions obtained from PCA. That is, we check if PCA provides low values for the convolutional layers that lie between the inputs of the concatenation. When both verifications (i.e., large norm division and consistency with PCA) hold, we proceed with the pruning. Note that these calculations can be done with the original or the pruned networks.

4. Experiments and performance results

In this section, we aim to validate the proposed strategies in as many distinct experimental settings as possible to prove their generalization capabilities. For this reason, we consider both classification and segmentation settings. With regard to the latter, and in order to cover as many variants as possible, we consider: (a) data with a few classes spread over all the pixels of the image, (b) images with few well-defined classes, and (c) datasets with many classes and great detail. In particular, for (a) and (b) we select outdoor images from two agricultural benchmarks (Di Cicco et al., 2017; Vidovic et al., 2016), for which U-Net was used due to highest performance among other segmentation models. As for (c), we pick urban scenes[6] from CamVid dataset (Brostow et al., 2008), for which DeepLabv3+ (Chen et al., 2018) and SegNet (Badrinarayanan et al., 2017) were chosen instead, in view of better results. For the classification setting, we consider VGG-16 (Simonyan & Zisserman, 2015) on CIFAR-10 (Krizhevsky, 2009) as a traditional convolutional architecture. Comparisons with state-of-the-art methods are reported in Tables 10 and 15. Since the best of our knowledge, this is the first time an exhaustive pruning study is performed on both encoder–decoder and classification models, contributing towards computer vision and deep learning.

Finally, it is worth noting that all the experiments are conducted within Matlab and executed on a i9-10900X CPU 3.70 GHz with 46 GB of RAM and NVIDIA GeForce RTX 3080 GPU.

4.1. Experimental settings

Dataset settings. The Crop Row Benchmark Dataset (CRBD) consists of 281 color crop row images of different types, like maize, potato or onion, and ground truth data. They have been taken at moderately varying yaw, pitch, and roll angles and resized to 240 × 320 pixels as required by the input layer. They are segmented in two classes, soil and greenness. Pixel-level annotations are created following a semi-automatic process. Firstly, greenness is identified in each pixel checking that the G (green) component dominates over R (red) and B (blue). Secondly, we manually retouch the annotations until the desired enhancement at the human expert's discretion is achieved.

Before training, we pre-process the images by adding 5 pseudo-bands obtained from the RGB bands according to different vegetation indices, widely used in agricultural image segmentation. Namely, the excess green index (ExG, Ribeiro et al., 2005; Woebbecke et al., 1995), the excess green minus excess red index (ExGR, Camargo Neto, 2004), the color index of vegetation (CIVE, Kataoka et al., 2003), the green leaf index (GLI, Louhaichi et al., 2001), and the combination index (COM). The latter is a combination of the indexes presented above together

[6] CamVid being a very common dataset, and in view of the comparable preliminary results obtained with other such datasets (e.g., cityscapes), to demonstrate generalization priority is given to data and network diversity versus experiments with similar data.

with the excess red index (ExR, Meyer et al., 1999), the normalized difference vegetation index (NDI, Meyer & Neto, 2008), the red green blue vegetation index (VI, Bendig et al., 2015), the modified green red vegetation index (MGRV, Bendig et al., 2015), and the vegetative index (VEG, Hague et al., 2006).

Continuing with outdoor agriculture images, we use the dataset publicly available in Di Cicco et al. (2017), consisting of 1252 realistic 360×480 synthetic images of sugar beet instances and random weeds, together with RGB channels pixel-wise annotations for three classes: soil, crop, and weed. This dataset will be denoted as SBD here-after.

Alternatively, the Cambridge-driving labeled Video dataset (CamVid) provides a collection of 701 street-level images of size 720×960, together with pixel-level labels. Following MathWorks (2022a), we group the original 32 classes into 11: sky, building, pole, road, pavement, tree, sign symbol, fence, car, pedestrian, and bicyclist. We also use random horizontal reflection and x/y translation of ± 10 pixels for data augmentation.

For these three datasets, we randomly split image and pixel label data into 60% for training, 20% for validation, and the remaining 20% for testing.

Finally, CIFAR-10 is a widely used dataset with 32×32 images, 50000 training images and 10000 test images, with 10 classes. It is divided into five training batches (each one containing 5000 images from each class) and one test batch (containing 1000 randomly selected images from each class). From the training set we randomly pick 10000 images for validation. Random horizontal and vertical reflections are used for data augmentation.

Training settings. We use the default U-Net architecture provided by MathWorks (2022b). It has an encoder depth equal to 3 and a total of 46 layers. The bridge between the encoder and the decoder contains two dropout layers with 50% drop probability. In the upsampling path each transpose convolution is followed by a ReLU. To improve training, classes are balanced by using class weighting and by replacing the pixel classification layer.

We also use DeepLabv3+, which is an encoder–decoder architecture, based on ResNet-18, that uses DeepLabv3 as encoder. We exclude the global average pooling layer in the atrous-spatial pyramid pooling (ASPP) as in MathWorks (2022a), obtaining a total of 100 layers. In general, each convolutional layer is followed by a batch normalization layer and a ReLU.

SegNet, also taken from Mathworks predefined architectures, has a VGG-16 based encoder with connections to the decoder. Thanks to them, the max pooling locations are used during upsampling. Batch normalization and ReLU layers follow each convolution.

Experiments are also carried out with the VGG-16 modified version given by Li et al. (2017). It consists of 13 convolutional layers and 2 fully connected layers. This model includes batch normalization and ReLU layers after each convolution, without dropout.

Finally, it is worth noting that each model is trained from scratch, except SegNet that uses the VGG-16 encoder pre-trained on ImageNet (Simonyan & Zisserman, 2015). Regarding training options, after trial and error, we select the following values. For the U-Net architectures we choose Adam optimizer with a squared gradient decay factor of 0.9, learning rate 10^{-3}, weight decay of 5×10^{-3}, mini-batch size of 8. We also set the validation patience to 4. CRBD is trained for 30 epochs, while 20 epochs are employed for SBD. In this case the learning rate drop period is also equal to 3, with factor 0.3. Besides, DeepLabv3+, SegNet and VGG-16 are trained with sgdm optimizer and a momentum of 0.9. In DeepLabv3+ we use a fixed learning rate of 10^{-3}, weight decay of 5×10^{-3}, 30 epochs, mini-batch size of 8, and we set the validation patience to 4. SegNet has the same weight decay but a learning rate of 10^{-2}, dropping every 100 epochs by a factor of 0.7. It uses 200 epochs and a mini-batch of 8. In VGG-16, the learning rate is equal to 10^{-2} and drops every 25 epochs by a factor of 3×10^{-3}. Moreover, weight decay takes value 10^{-3}, epochs are 80, and mini-batch size is 128.

Pruning settings. For retraining, the previous hyperparameters, named here-after as original, are used, except for the number of epochs. As explained in Section 3.5, we only present the results of pruning at once and pruning-fixing-retraining strategies, since those of pruning-retraining are not better.

When following the pruning-fixing-retraining proposal, the original number of epochs is reduced by 1/3. The impact of each fine-tuning on network convergence is analyzed in Appendix B for our two strategies (PCA-N *it* and NV *it*), and all pairs of CNNs and datasets, by comparing the validation loss before and after each retraining. Results can be found in Fig. B.12, B.13, B.14, and B.15. The consistency of the validation loss after fine-tuning through the layers, and its proximity to the loss of the unpruned network, allow us to conclude that each fine-tuning gets the converged result. Even though we only retrain for a few epochs (1/3 of the original number), a competitive loss is obtained in each layer. Thus, convergence is not only due to the fine-tuning, but also to proper pruning, including fixing the appropriate channels. Given that in some cases the validation loss in the last layer is slightly higher than expected, a final global retraining is proposed. The number of epochs used is shown in parentheses next to the method name (e.g., PCA-N *it* (10)).

For the pruning at once method, either the original number of epochs or 1/3 of this value is considered. In the case of U-Net, we conduct in-depth pruning analysis in all compressed networks and proceed with the (extra) pruning if verifications hold. After removing the layers, the resulting networks are retrained. As before, the number of epochs used when pruning at once and in-depth is shown in parentheses next to the method name. It is worth noting that even though more retraining epochs provides better performance, as can be seen in Tables 3, 5, 7, 9, and 12, excellent results are obtained with poor fine-tuning.

Finally, a first choice of parameters (γ, α, β) is made based on the pruning results of previous studies (Ayinde et al., 2019; Chen et al., 2022), where some layers achieve a 75% drop in parameters, while others only reach the 25%. Hence, following the recommendations of Section 3.3.2, we set $\gamma = 0.25$ and $\alpha = 0.75$. Since it is reasonable that the importance scores of the removed channels reach at least the 10% of the total score, $\beta = 0.1$. If necessary to obtain better results, parameters can be coherently modified. For example, if performance worsens, reduce α and/or γ. If not enough pruning is obtained, increase β and/or γ. This naive search was enough for U-Net, whereas an exhaustive one was needed for VGG-16 and DeepLabv3+, as reported in Sections 4.4 and 5.2. An ablation study can be found in Appendix C.

4.2. Results for U-Net on CRBD

Both pruning methods proposed in our strategy prune all convolutional layers except the last one. Concerning the next convolution influence-variance criterion, the entire dataset is used to calculate the mean standard deviation (2), whereas the first choice of parameters $\gamma = 0.25$, $\alpha = 0.75$ and $\beta = 0.1$ was enough to obtain great results, as reported below.

Fig. 4 shows the percentage of channels removed in each convolutional layer, with respect to the total number of channels, by each of our methods. The name of the layers appears at the bottom. Note that the distinction between fine-tuning strategies (*it* or *once*) is only needed in NV, since PCA-N prunes the same number of channels in both cases. Observe that the optimal dimensions obtained by PCA are small in the first layer and in the intermediate ones, a fact that will be used to prune in depth. A notable pruning of the first layer also occurs in the NV method. In this case, when using the proposed iterative fine-tuning procedure, more channels are removed.

We summarize the performance in Tables 2 and 3. The analyzed metrics, presented in Table 2, are: global accuracy (Acc.), mean accuracy (Mean acc.), mean IoU, weighted IoU (W. IoU), mean boundary F1 score (Mean BF), validation loss (Val. loss), and validation accuracy

Table 2
Pruning results of U-Net on CRBD, where different metrics are shown. The numbers in parentheses are the epochs used during fine-tuning.

Model	Acc.	Mean acc.	Mean IoU	W. IoU	Mean BF	Val. loss	Val. acc.(%)
Original	0.9521	0.9611	0.8663	0.9135	0.9626	0.6283	94.87
PCA-N *it*	0.9528	0.9587	0.8781	0.9135	0.9547	0.6108	**96.19**
PCA-N *it* (10)	0.9564	**0.9670**	0.8771	0.9208	0.9654	**0.5581**	95.68
PCA-N *once* (10)	0.9440	0.9585	0.8484	0.9007	0.9507	0.6704	94.75
PCA-N *once* (30)	**0.9588**	0.9605	**0.8813**	**0.9243**	**0.9698**	0.7842	96.16
NV *it*	0.9518	0.9596	0.8653	0.9127	0.9675	0.7860	95.45
NV *it* (10)	0.9554	0.9630	0.8741	0.9190	0.9573	0.6400	95.48
NV *once* (10)	0.9474	0.9597	0.8558	0.9060	0.9574	0.6985	95.30
NV *once* (30)	0.9521	0.9610	0.8662	0.9135	0.9546	0.6648	95.16
Layers1 (30)	**0.9599**	**0.9695**	**0.8855**	**0.9266**	**0.9775**	**0.5626**	**96.46**
Layers2 (10)	0.9534	0.9651	0.8700	0.9159	0.9640	0.6438	95.78

Table 3
Comparison of pruning results for the distinct strategies for U-Net on CRBD. Acc. is short of accuracy and Param. of parameters. ↓ indicates the drop percent between the pruned models and the baseline model (included on the top).

Original: Acc = 95.21%, Param = 7.70M, FLOPs = 8.28×10^{10}, time = 0.0243 s					
Model	Acc. ↓(%)	Param.	Param. ↓(%)	FLOPs ↓(%)	time (s)
PCA-N it	−0.07	0.56M	92.7	64.7	0.0115
PCA-N *it* (10)	−0.45				0.0141
PCA-N *once* (10)	0.85				0.0114
PCA-N *once* (30)	**−0.70**				0.0118
NV it	0.03	**0.10M**	**98.7**	**97.5**	**0.0103**
NV *it* (10)	−0.35				0.0122
NV *once* (10)	0.49	0.91M	88.1	74.4	0.0116
NV *once* (30)	0.004				0.0114
Layers1 (30)	**−0.81**	0.56M	92.7	64.7	0.0142
Layers2 (10)	−0.14	**0.22M**	**97.1**	**80.9**	**0.0090**

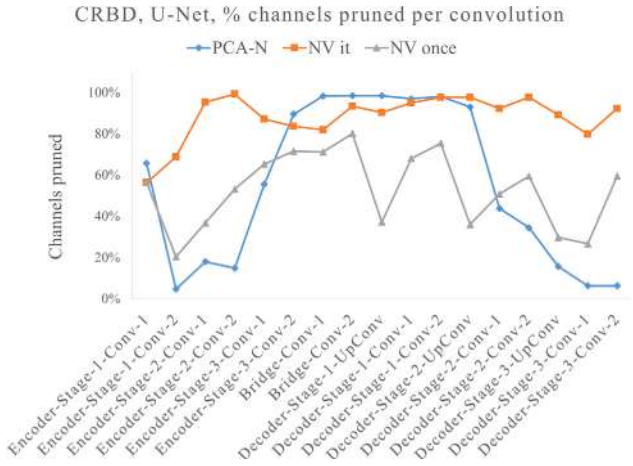

Fig. 4. Percentage of channels pruned in each convolutional layer for U-Net on CRBD.

(Val. acc.). Original refers to the baseline U-Net model described in Section 4.1. Table 3 compares the pruning results of the proposed strategies in terms of the percentage by which the accuracy, parameters, and FLOPs have decreased w.r.t the baseline model, whose values are also included on top of the table for better comprehension. The last column corresponds to the average time it takes to segment an image, which has been obtained with the entire test set. The same schema will be followed in the different datasets.

With the information of both tables, we observe that the metrics of the pruned networks are as good as those of the original one, without a significant increase in the loss. Besides, a drop of more than 90% of the parameters and of more than 64% in FLOPs (and in some cases of nearly 99%), is reached without almost reduction in accuracy (at most 0.85% for PCA-N *once* (10)) or with some improvements. This reduction in the number of parameters and FLOPs is reflected on the average segmentation time of the pruned and re-trained models, by a decrease of at least one half. The success of the iterative fine-tuning strategy is proven, performing better than pruning at once (NV *it*). However, even the fastest of the latter, consisting of retraining with just 1/3 of the original epochs, shows great performance, with a drop in accuracy of only 0.5% for the NV strategy (NV *once* (10)). If 30 epochs are used, we could even improve the accuracy in 0.7% (PCA-N *once* (30)).

Next, we prune in depth the original neural network model and all the compressed ones by proceeding as explained in Section 3.6.2. Hence, the weight tensors of the convolutional layers after each concatenation are considered. We compute the l_2-norm of the portion coming from the encoder and from the decoder, and divide the former by the later l_2^E/l_2^D. Two high values are obtained. Namely, for the PCA-N *it* model, $l_2^E/l_2^D = 202.6$ in the deepest concatenation, between layers Encoder-Stage-3-ReLU2 and Decoder-Stage-1-Conv1 (Fig. 4). For the NV *once* (30) model, $l_2^E/l_2^D = 125.25$ in the middle concatenation, between layers Encoder-Stage-2-ReLU2 and Decoder-Stage-2-Conv1. Fig. 4 shows that almost all the convolutional layers in between have low optimal PCA dimension. Thus, we prune and remove the corresponding layers. We end up with two extra-pruned models, Layers1 and Layers2 respectively in Tables 2 and 3. While the former improves accuracy, by a 0.81% gain, neither their parameters or FLOPs are significantly reduced further. A general improvement is observed in Layers2, achieving more than 90% reduction in parameters and 80% reduction in FLOPs, while recovering accuracy.

4.3. Results for U-Net on the synthetic sugar beet dataset

As before, both strategies prune all convolutional layers except the last one. We randomly select 50% of the dataset to calculate the mean standard deviation (2) for the next convolution influence-variance method. Regarding the remaining parameters, the previous value of α (0.75) leads to an excessive drop in accuracy. However, a second choice of values for $\gamma = 0.25$, $\alpha = 0.5$, and $\beta = 0.1$ was enough to obtain great results.

Table 4
Pruning results of U-Net on the SBD, where different metrics are shown.

Model	Acc.	Mean acc.	Mean IoU	W. IoU	Mean BF	Val. loss	Val. acc.(%)
Original	0.9905	**0.9883**	0.7831	0.9857	0.8607	**1.4314**	98.95
PCA-N *it*	0.9901	0.9872	0.7768	0.9844	0.8583	1.6723	98.90
PCA-N *it* (7)	0.9877	0.9867	0.7552	0.9803	0.8237	1.9016	98.65
PCA-N *once* (7)	0.9901	0.9832	0.7731	0.9845	0.8427	2.0757	98.89
PCA-N *once* (20)	**0.9911**	0.9838	**0.7855**	**0.9858**	**0.8649**	2.0805	**99.00**
NV *it*	0.9901	0.9810	0.7787	0.9838	0.8485	3.6285	98.88
NV *it* (7)	0.9884	0.9868	0.7590	0.9821	0.8301	1.7015	98.71
NV *once* (7)	0.9864	0.9785	0.7372	0.9801	0.7668	2.6978	98.49
NV *once* (20)	0.9882	0.9781	0.7519	0.9823	0.7967	2.7763	98.68

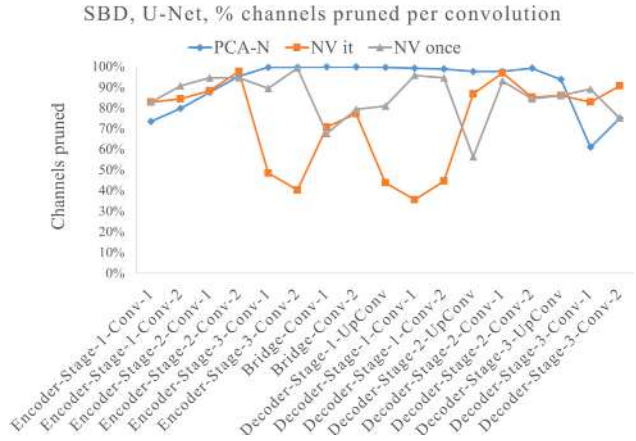

Fig. 5. Percentage of channels pruned in each convolutional layer for U-Net on SBD.

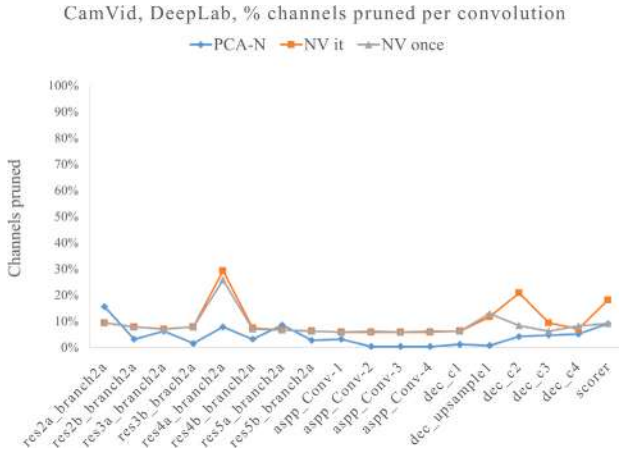

Fig. 6. Percentage of channels pruned in each convolutional layer for DeepLabv3+ on CamVid dataset.

Fig. 5 shows the graphs of the percentage of pruned channels in each convolutional layer. In this case, fewer channels are discarded in the NV method, but always surpassing the 30% threshold and obtaining further parameter reduction when pruning at once. Even though the same U-Net model is involved, Figs. 4 and 5 exhibit that the channels removed by different methods vary greatly. Note that our two approaches rely on the activation maps of the layers: PCA-N via PCA and NV through the mean standard deviation (2). As explained in Section 4, CRBD and SBD contain images with dissimilar characteristics. Thus, it is not unusual to assume that the network has learned differently how to segment them, so the information encoded in the maps differ. Consequently, the number of channels provided by PCA-N and NV changes from one dataset to another. Despite this, the same (less evident) inverted U-shape appears in PCA as in Fig. 4, with the smallest optimal dimension in the intermediate layers.

The effectiveness of our proposals is summarized in Tables 4 and 5. In this case, no depth pruning is performed since verifications did not hold (we did not find high norm quotient values for l_2^E/l_2^D). A better performance of the PCA-N method compared to NV is observed in Table 4. While the PCA-N shows no significant reduction in metric values or increase in validation loss, those are more pronounced in NV. It is also worth noting that whereas a final global retraining in the iterative case reduces the difference in loss (NV *it* (7)), it does not recover metrics performance.

Regarding PCA-N, by simply retraining for 7 epochs after pruning all layers at once, we can reduce the parameters in more than a 99% and the FLOPs in almost a 95%, with a negligible accuracy drop of 0.04%. The last column of Table 5 shows the benefits on segmentation time. As before, more epochs provide less drop in accuracy, with even an improvement of 0.06% for PCA-N *once* (20). However, a final global retraining in the iterative fine-tuning case (PCA-N *it* (7) and NV *it* (7)) worsens the performance. Although the NV strategy prunes less with more accuracy reduction, it also achieves satisfactory results, fulfilling an improvement in FLOPs of 98.5% with only a 0.23% accuracy drop.

4.4. Results for DeepLabv3+/SegNet on CamVid

As explained in Section 3.6.1, in the case of DeepLabv3+, within residual blocks we only prune the first convolution, while in the rest of the network all but the first and last convolutional layers are pruned. We obtain the mean standard deviation (2) for the NV method by randomly picking a 10% of the dataset. After trial and error tests, to reduce the drop in accuracy γ, α, and β are respectively adjusted to 0.1, 0.5, and 0.05, w.r.t. previous experiments.

As Fig. 6 shows, the percentage of channels pruned in each convolutional layer is low. In particular, in PCA-N strategy almost no pruning is performed in the ASPP module. NV procedure discards 10% of the channels in each layer, except for two of them where the percentage is higher, causing a greater pruning than the previous method.

We report the performance results in Tables 6 and 7. Although at most a 10.7% pruning is achieved with a drop of 13.4% in FLOPs for the NV method, the validity of both strategies and of our iterative procedure is full well proven. Metrics surpass the original ones with no rise in validation loss, though a final global retraining may be needed (e.g. PCA-N *it* (10) and NV *it* (10)). Moreover, NV exceeds accuracy by nearly a 2% when fine-tuning with our strategy. Even pruned at once models show excellent performance, with an accuracy improvement of 0.46% for only 1/3 of the original epochs (NV *once* (10)). Yet, no decrease in the average segmentation time is observed, confirming the low drop in the number of parameters and FLOPs.

This poor pruning may be explained thanks to DeepLab's architecture, which is characterized by residual and ASPP modules. In the former, information flows through two paths: (1) directly via the skip connection and (2) processed by the convolutional branch, being both necessary to obtain generalization. If excessive pruning were carried out on the convolution, the residual block would work nearly as the identity, worsening the results. ASPP modules, designed in this case at different scales, are in charge of obtaining contextual information at

Table 5
Comparison of pruning results for the distinct strategies for U-Net on the SBD.

Original: Acc = 99.05%, Param = 7.70M, FLOPs = 1.85×10^{11}, time = 0.0344 s					
Model	Acc. ↓(%)	Param.	Param. ↓(%)	FLOPs ↓(%)	time (s)
PCA-N it	0.04	**0.01M**	**99.86**	94.8	0.0107
PCA-N *it* (7)	0.28				0.0106
PCA-N *once* (7)	0.04				0.0104
PCA-N *once* (20)	**−0.06**				0.0106
NV it	0.04	1.29M	83.23	88.3	0.0129
NV *it* (10)	0.21				0.0127
NV *once* (7)	0.41	0.21M	97.33	**98.5**	0.0107
NV *once* (20)	0.23				**0.0105**

Table 6
Pruning results of DeepLabv3+ on CamVid, where different metrics are shown.

Model	Acc.	Mean acc.	Mean IoU	W. IoU	Mean BF	Val. loss	Val. acc.(%)
Original	0.9094	0.8773	0.7021	0.8504	0.7420	5.2527	92.33
PCA-N *it*	0.9146	0.8790	**0.7363**	0.8526	0.7760	7.1384	92.92
PCA-N *it* (10)	0.9211	0.8837	0.7216	0.8659	0.7807	5.2969	93.31
PCA-N *once* (10)	0.9145	0.8816	0.7118	0.8574	0.7604	5.4590	92.73
PCA-N *once* (30)	0.9181	0.8744	0.7188	0.8613	0.7701	5.5579	93.31
NV *it*	**0.9272**	0.8077	0.6892	0.8729	0.7854	8.8171	93.88
NV *it* (10)	0.9265	**0.8874**	0.7359	**0.8731**	**0.7914**	5.2267	**93.91**
NV *once* (10)	0.9136	0.8811	0.7024	0.8559	0.7545	5.1903	92.71
NV *once* (30)	0.9161	0.8811	0.7075	0.8592	0.7597	**5.0893**	92.93

Table 7
Comparison of pruning results for distinct strategies for DeepLabv3+ on CamVid dataset. An exhaustive search leads to a **51.9%** FLOPs, **46.5%** parameters, and **0.18%** accuracy drop for NV *once* (30) and $(\gamma, \alpha, \beta) = (0.1, 0.6, 0.2)$.

Original: Acc = 90.94%, Param = 20.6M, FLOPs = 2.17×10^{11}, time = 0.0411 s					
Model	Acc. ↓(%)	Param.	Param. ↓(%)	FLOPs ↓(%)	time (s)
PCA-N it	−0.57	19.8M	3.9	6.0	0.0416
PCA-N *it* (10)	−1.29				**0.0400**
PCA-N *once* (10)	−0.56				0.0419
PCA-N *once* (30)	−0.96				0.0417
NV it	**−1.96**	**18.4M**	**10.7**	**13.4**	0.0416
NV *it* (10)	−1.88				0.0414
NV *once* (10)	−0.46				**0.0400**
NV *once* (30)	−0.73				0.0421

different resolutions. Due to the characteristics of CamVid images (with many different textures at distinct resolutions and variability of pixel-level information), these layers are essential to capture the contextual information, and provide good performance and generalization. Thus, we can infer that ASPP are of great importance, hence there is not much pruning. It is worth noting that one of the objectives of these modules is to reduce the computational cost, so poor pruning need not be a problem. Note that none of these modules exist in SegNet. Indeed, the same problem arises in the classification setting when pruning ResNets, characterized by residual blocks (He et al., 2022; Li et al., 2020; Shao et al., 2021). As indicated in Luo et al. (2017) and Ayinde et al. (2019), residual networks are compact with poor redundancy, which makes pruning more challenging.

Focusing on the NV *once* (30) method, if we conduct an exhaustive search of parameters prioritizing pruning (see Appendix C), a 51.9% FLOPs and 46.5% parameters reduction is reached with only a 0.18% drop in accuracy. The values required for γ, α, and β are 0.1, 0.6, and 0.2 respectively. Although the reduced number of segmentation pruning experiments in the literature does not let us make further direct comparisons, it is worth noting that the results obtained are similar to those of Chen et al. (2022): 44.26% FLOPs, 43.31% parameters, 1.6% performance drop. In this work, DeepLabv3 (with ResNet-101 as encoder) is trained on cityscapes, a dataset with urban outdoor images similar to those of CamVid. In our case, the base network ResNet-18 has far fewer layers, giving even more validity to the pruning achieved.

In the case of SegNet on CamVid, all convolutional layers except the last one are pruned. This model has connections between the pooling and unpooling layers of the encoder and decoder respectively. Hence, those layers should have the same number of channels when fine-tuning is performed. For this reason, we only apply the PCA-N at once strategy, with the 99,5% or 99% of the cumulative variance explained. The number of channels removed from each convolution preceding a pooling layer is used to prune the convolution before the connected unpooling layer, and all the previous convolutional layers until we reach another unpooling. This can be seen in Fig. 7, as well as a decrease in PCA optimal dimensions as we go deeper into the encoder.

Results are presented in Tables 8 and 9. There is no significant reduction in metric values if sufficient retraining is done, with only a small increase in the validation loss in all cases. We reach a 60% parameters drop and 75.4% FLOPs reduction with a 0.26% gain in accuracy. If PCA retains less information (PCA-N 99%), we achieve a decrease of 72.4% in parameters and of 83.6% in FLOPs, with reasonable metrics and a slight 0.07% improvement in accuracy. This decrease is reflected in the average segmentation time.

In Table 10 we quantitatively compare the compressed models with other comparable results extracted from the state-of-the-art, which are, to the best of our knowledge, the ones identified. As a performance measure, the drop in mean IoU is considered instead of the accuracy. Similarly to the previous cases, the negative values imply that the metric has improved with respect to the original network. Despite the lack of complexity of our strategy, we achieve the greatest reduction in FLOPs and a competitive percentage of pruned parameters, while improving the mean IoU over most methods.

Table 8
Pruning results of SegNet on CamVid dataset, where different metrics are shown.

Model	Acc.	Mean acc.	Mean IoU	Weight. IoU	Mean BFS	Val. loss	Val. acc.(%)
Original	0.9559	0.7847	0.7109	**0.9256**	**0.7492**	**0.0932**	96.67
PCA-N 99.5% *once* (60)	0.9534	0.7571	0.6933	0.9138	0.7392	0.1123	96.57
PCA-N 99.5% *once* (200)	**0.9584**	**0.7946**	**0.7223**	0.9227	0.7405	0.1050	**97.03**
PCA-N 99% *once* (60)	0.9487	0.7308	0.6635	0.9057	0.7148	0.1288	95.96
PCA-N 99% *once* (200)	0.9566	0.7829	0.7173	0.9160	0.7280	0.1193	96.71

Table 9
Comparison of pruning results for distinct strategies for SegNet on CamVid.

Original: Acc = 95.59%, Param = 29.4M, FLOPS = 1.10×10^{11}, time = 0.0639 s					
Model	Acc.↓(%)	Param.	Param.↓(%)	FLOPS↓(%)	time (s)
PCA-N 99.5% *once* (60)	0.26	11.7M	60.0	75.4	0.0416
PCA-N 99.5% *once* (200)	**−0.26**				0.0376
PCA-N 99% *once* (60)	0.75	**8.1M**	**72.4**	**83.6**	**0.0365**
PCA-N 99% *once* (200)	−0.07				0.0373

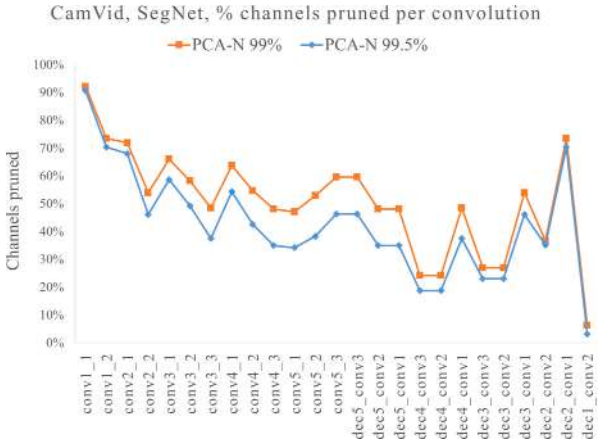

Fig. 7. Percentage of channels pruned in each convolutional layer for SegNet on CamVid.

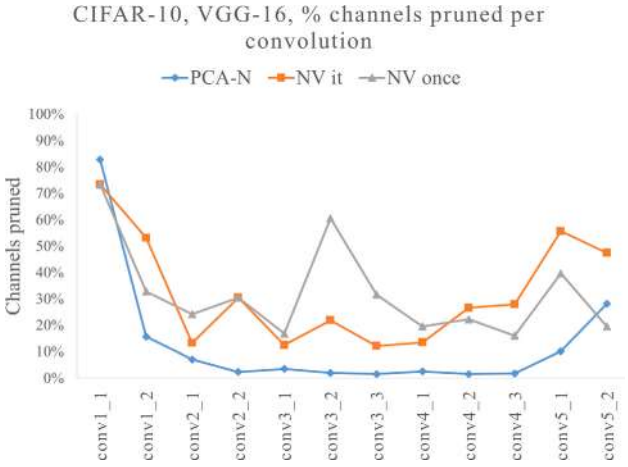

Fig. 8. Percentage of channels pruned per convolutional layer for VGG-16 on CIFAR-10.

Table 10
Results of pruned SegNet on CamVid, taken from original papers and of our experiments. The hyphen (-) shows that results are not reported in the original paper. ACoSP-R refers to the different compression ratios..

Method	Mean IoU ↓(%)	Param. ↓(%)	FLOPs ↓(%)
CAP (He, Wu, Liang, & Lam, 2021)	−2.7	79.5	71.88
ACSL (He, Wu, & Lam, 2021)	0.29	75.8	73.54
ACoSP-2 (Ditschuneit & Otterbach, 2022)	6.5	49.9	–
ACoSP-4 (Ditschuneit & Otterbach, 2022)	9.7	74.9	–
Ours - 99.5%	−1.6	60.0	75.4
Ours - 99%	−0.9	72.4	83.6

4.5. Results for VGG-16 on CIFAR-10

For this model, all convolutional layers except the last one are pruned. The mean standard deviation (2) is computed with 1000 random images from the training set. Outperformance is achieved with the parameter values of Section 4.3 (i.e., $\gamma = 0.25$, $\alpha = 0.5$, and $\beta = 0.1$), selected again after testing.

The number of channels to be removed computed by each pruning strategy is exhibited in Fig. 8. PCA obtains small optimal dimensions in the first and last layers, giving rise to a U-shape graph. The same phenomena can be observed in the NV method, where most of the channels are deleted within these layers. As expected, due to the small dimensions of the feature maps, PCA-N results in poor pruning in contrast to NV. Note that in both cases, the layer with largest pruning is the first one.

We summarize the overall results of the compressed networks in Tables 11 and 12. As it happened with SBD, PCA-N performs better than NV. Metrics are as the original ones, or even outperform them when carrying out our iterative fine-tuning or retraining for enough epochs (*once* (40)). Even though PCA-N models only reduce parameters by 14.1% and FLOPs by almost 18%, we obtain a drop in the average classification time and improve accuracy in more than 0.5%. The NV strategy achieves further compression, of about 50% in parameters and FLOPs, but exhibits worst performance, getting reasonable results in metric values and accuracy (even increasing a 0.12%) only if we retrain for enough epochs (NV *once* (40)). It should be pointed out that, after a thorough analysis in the next section, we are able to obtain results comparable to those in the literature, as Table 15 shows.

5. Deeper understanding of the approaches

In this section we analyze the relationship between the pruning performance and the type of dataset used. This helps to understand how networks learn, connecting our approaches with xAI. From the general conclusions for all datasets, which are explained below, we improve the pruning of VGG-16 on CIFAR-10, to achieve results similar to those of the state-of-the-art (Table 15). Lastly, we also compare our strategy with the pruning of random and most relevant channels.

Table 11
Pruning results of VGG-16 on CIFAR-10. Different metrics are shown as follows: accuracy, mean accuracy, mean precision, validation loss and validation accuracy.

Model	Acc.	Mean acc.	Mean prec.	Val. loss	Val. acc. (%)
Original	0.8505	0.9701	0.8511	0.4770	85.45
PCA-N *it*	**0.8558**	**0.9712**	**0.8565**	0.5505	88.66
PCA-N *it* (25)	0.8553	0.9711	0.8560	0.5293	87.84
PCA-N *once* (10)	0.8443	0.9689	0.8468	**0.3028**	90.46
PCA-N *once* (25)	0.8462	0.9692	0.8479	0.3493	89.75
PCA-N *once* (40)	0.8552	0.9710	0.8556	0.3157	**90.79**
NV *it*	0.8486	0.9697	0.8506	0.6558	87.42
NV *it* (25)	0.8383	0.9677	0.8392	0.5986	86.30
NV *once* (10)	0.8228	0.9646	0.8241	0.4437	85.22
NV *once* (25)	0.8287	0.9657	0.8310	0.5161	84.95
NV *once* (40)	0.8515	0.9703	0.8520	0.4299	87.04

Table 12
Comparison of pruning results for distinct strategies for VGG-16 on CIFAR-10.

Original: Acc = 85.05%, Param = 14.9M, FLOPs = 6.26×10^8, time = 6.8×10^{-3} s					
Model	Acc. ↓(%)	Param.	Param. ↓(%)	FLOPs ↓(%)	time ($\times10^{-3}$ s)
PCA-N it	**−0.62**				6.29
PCA-N *it* (25)	−0.56				6.29
PCA-N *once* (10)	0.73	12.8M	14.1	17.9	6.31
PCA-N *once* (25)	0.51				6.28
PCA-N *once* (40)	−0.55				6.30
NV it	0.22	**7.6M**	**49.4**	46.0	**5.27**
NV *it* (25)	1.43				5.31
NV *once* (10)	3.26				5.75
NV *once* (25)	2.56	8.8M	40.9	**50.8**	5.62
NV *once* (40)	−0.12				5.52

5.1. In-depth explanation

In general, many channels are removed from the first convolution (Fig. 4, 5, 6, and 8). It is widely known that the first layers of convolutional networks focus on features such as edge detection, colors, or textures, which are essential for decision making (Krizhevsky et al., 2012; Yu et al., 2014). Whereas the value of the filter norm obtained with (1) is not directly related to the features they detect, the mean standard deviation (2) of activation maps is. In fact, some feature maps (such as texture or color over edge detection) have greater variance than others. When pruning, if the method used is assigning high importance scores to channels with large variance, it may be prioritizing some features over others.

The hypothesis is that, if the wrong features are chosen in the first layer, essential ones will be removed, leading to a degradation in performance. On the contrary, if those features are suitable, reliable performance will be achieved. This helps to understand the predictions made. Since NV method includes the mean standard deviation of feature maps in its calculation of channel relevance, when the product ls_j^i is dominated by the variance, the latter will guide the pruning. Hence, we prioritize one feature over another, and depending on the feature this may cause an effectiveness reduction.

To verify our arguments, we show that this is exactly what is happening with VGG-16 on CIFAR-10 and U-Net on CRBD, and the reason why NV does not perform well in the former, but it does it in the latter. We do not take DeepLabv3+ on CamVid into consideration, due to its poor pruning, neither SegNet, since we have only applied to it one of our strategies.

First, we demonstrate that for CIFAR-10 the superiority in performance of PCA-N over NV is not caused by the difference in parameters drop. To achieve an in-depth explanation, another pruning is conducted. We remove the same number of channels as in the NV model but using only the weight-norm (excluding the variance) in the importance scores. Table 13 presents the results of the new pruning compared with the previous ones (extracted from Tables 11 and 12). The comparison shows how, with the same pruning ratio, these new compressed networks perform better than the NV models.

Secondly, and following our hypothesis, we carry out an analysis of the first convolutional layer of the original network (obtained as described in Section 4.1). For each activation map, the subsequent values are obtained: the associated weight norm l_j^1, mean standard deviation s_j^1 (using the entire dataset), and their combination ls_j^1. The latter defines the importance scores and guides the NV pruning. We present in Fig. 9 those feature maps with largest and lowest values, to observe that in CIFAR-10 the product is controlled by the mean standard deviation. As a result, next convolution influence-variance criterion establishes high importance scores for color features over edge detectors.

To prove that this is one of the causes of performance degradation in the CIFAR-10 NV method, we conduct the same pruning but preserving the edge detectors instead of color ones in the first layer. Note that the next convolution influence criterion associates relevant channels to edge detection in Fig. 9. Hence, it is enough to apply this criterion to the first layer. Table 14 shows how pruning all the layers at once and retraining for 10 epochs provides more than a 1% increase in accuracy. If we do not restrict ourselves to the first convolution and we prune the entire network with the next convolution influence criterion, great performance is achieved, as Table 13 shows.

These analyses reinforce the hypothesis that the NV strategy is prioritizing the wrong features in the first convolutional layer and removing essential ones, in this case edge detectors, leading to a decay in performance. On the contrary, the next convolution influence criterion is preserving these features, obtaining an effective pruned model. These conclusions are helpful in terms of the explainability of the networks (xAI). Nevertheless, it is worth noting that Tables 11 and 12 show that, even if important features are removed, performance can be recover with enough retraining epochs (*once* (40)). This, together with the fact that the second convolutional layer of VGG-16 is poorly pruned (as Fig. 8 shows), supports what was observed by Li et al. (2017): "the first layer is robust to pruning as compared to next few layers".

Moving onto U-Net, the same activation map analysis, represented in Fig. 9, shows that the next convolution influence-variance criterion is also prioritizing features with high variance over those with low values. However, as shown in Table 3, this time NV compressed models have

Table 13
Pruning results for VGG-16 on CIFAR-10 when removing the same number of channels as the NV strategy with the importance scores given only by the weight-norm. Iterative fine-tuning ("Iterative") achieves a 49.4% parameter reduction, while pruning at once and retraining ("At once") a 40.9%.

Fine-tuning	Acc.	Acc. ↓(%)	Mean acc.	Mean prec.	Val. loss	Val. acc. (%)
Iterative	**0.8540**	**−0.41**	**0.9708**	**0.8539**	0.5520	**88.76**
Iterative (25)	0.8458	0.55	0.9691	0.8486	0.5550	86.86
At once (10)	0.8385	1.41	0.9677	0.8386	**0.3855**	87.23
At once (25)	0.8403	1.20	0.9680	0.8409	0.4705	86.03
At once (40)	0.8514	−0.11	0.9703	0.8519	0.4079	88.10
NV pruning results extract from Tables 11 and 12						
Iterative	0.8486	0.22	0.9697	0.8506	0.6558	87.42
Iterative (25)	0.8383	1.43	0.9677	0.8392	0.5986	86.30
At once (10)	0.8228	3.26	0.9646	0.8241	0.4437	85.22
At once (25)	0.8287	2.56	0.9657	0.8310	0.5161	84.95
At once (40)	0.8515	−0.12	0.9703	0.8520	0.4299	87.04

Table 14
Pruning results of the experiments carried out to explain NV performance. "NV *once* (10)" is the same model as in Tables 3 and 12. "NV-1" refers to the new pruned models, where the first convolution has been treated differently.

CIFAR-10 model	Acc.	Acc. ↓(%)	Mean acc.	Mean prec.	Val. loss	Val. acc. (%)
NV *once* (10)	0.8228	3.26	0.9646	0.8242	0.4437	85.22
NV-1	**0.8322**	**2.15**	**0.9664**	**0.8323**	**0.4156**	**86.02**
CRBD model	Acc.	Acc. ↓(%)	Mean acc.	Mean IoU	Val. loss	Val. acc. (%)
NV *once* (10)	**0.9474**	**0.49**	**0.9597**	**0.8558**	0.6985	**95.30**
NV-1	0.9441	0.85	0.9592	0.8486	**0.6978**	95.02

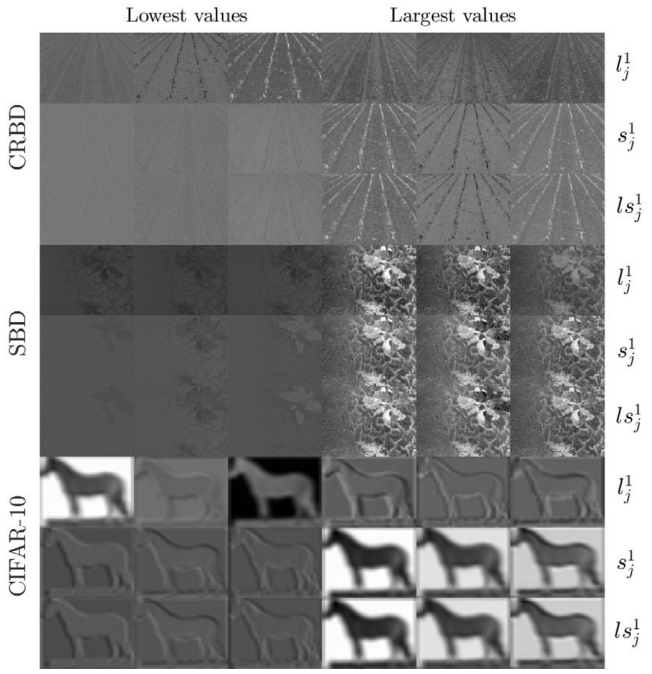

Fig. 9. Visualization of the activation maps of the first convolutional layer of U-Net on CRBD and SBD and of VGG-16 on CIFAR-10. For each pair network-dataset, the activation maps with the three lowest (first 3 columns starting from the left) and three highest (last 3) values of l_j^1 (first row), of s_j^1 (second row) and of their product ls_j^1 (third row) are represented.

an excellent performance. Following our hypothesis this means that, unlike in CIFAR-10, the preserved features are important for decision making.

To gain a deeper understanding of the process, we perform a similar experiment by pruning the same channels as in the NV strategy except in the first layer, where high importance is given to channels with small mean standard deviation. We prune at once an retrain for 10 epochs, presenting the results in Table 14. The resulting compressed network duplicates the accuracy drop of the analogue NV model, corroborating that features with large variance are relevant. Note that the next convolution influence criterion does not prioritize a particular feature detector over another as Fig. 9 shows.

On the whole, in datasets such as CIFAR-10 (where objects that almost take up all the images are classified) the NV strategy is not suitable, since it pays no attention to edge detectors despite their importance. However, it is appropriate for semantic segmentation of agriculture grainy images. More experiments over similar datasets are needed to consolidate this reasoning. We want to highlight that these arguments are not valid for the sugar beet dataset, where PCA-N performs better than NV, but both prioritize the same kind of features (as Fig. 9 shows) and have similar pruning ratios. Further analysis on the statistics of the first and following layers is needed, supported by xAI. The fact that these are synthetic images may be relevant too.

In conclusion, care must be taken when deciding on the compression method to be used in each dataset. If some features are prioritized over others, degradation in performance may occur. Note that, from pruning results, we have obtained information about how the networks are learning and which features are more relevant, entering xAI. Thus, importance scores could be defined based on some interesting characteristic and then, by pruning the network using the strategy presented in 3.3.2, results will show if that characteristic is enough for the network to learn, and will help to understand its decisions and predictions. For instance, following this approach we have found that on CIFAR-10 large variance characteristics are insufficient.

5.2. Improved pruning

By analyzing the ReLU layer after the first convolution of VGG-16 we have observed in Fig. 10 that activation maps with high standard deviation correspond to edge features. Already knowing that part of the underperformance of the NV strategy was due to the removal of too many edge detection channels, in Eq. (3) we compute this time the mean standard deviation s_j^i of the activation maps of the ReLU layer, in order to check if we can improve the performance.

After an exhaustive testing to find the optimal parameter values, with $\gamma \in (0.45, 0.5)$, $\alpha \in (0.65, 0.75)$, and $\beta \in (0.05, 0.3)$ (see Appendix C), we are able to achieve a 86.4% parameters and 82.2% FLOPs drop with a 0.78% improvement in accuracy. The values of γ, α, and β optimizing accuracy are 0.45, 0.65, and 0.05, respectively, which corresponds to

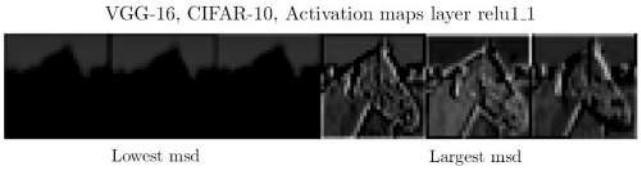

Fig. 10. Visualization of the activation maps of the first ReLU layer of VGG-16 on CIFAR-10. The ones with the three lowest and three largest mean standard deviation values (msd) are represented.

Table 15
Results of pruned VGG-16 on CIFAR-10 dataset, taken from original papers, and of ours experiments. Ours are pruned at once with the NV strategy and retrained for 40 epochs.

Method	Acc. ↓(%)	Param. ↓(%)	FLOPs ↓(%)
l_1-Prune (Li et al., 2017)[a]	0.67	88.5	–
FStats (Li et al., 2020)	−0.33	90.7	56.3
FPGM (He et al., 2019)	0.36	88.5	–
FM info. Shao et al. (2021)	0.39	93.3	73.2
N-Slim (Liu et al., 2017)	−0.06	88.5	51.1
Adapt-DCP (Liu et al., 2021)	−0.57	91.7	69.8
FDis (He et al., 2022)	−0.39	94.4	76.9
Ours-1	−0.78	86.5	82.2
Ours-2	2.26	91.7	89.4

[a] Results of Li et al. (2017) are obtained from Liu et al. (2017).

the Ours-1 model in Table 15. Optimizing the pruning ratio, we obtain the Ours-2 model for γ, α, and β equal to 0.5, 0.65, and 0.1. It prunes a 91.7% of parameters and 89.4% of FLOPs with 2.26% drop in accuracy. Although parameter reduction is not the best, compared with state-of-the-art results that are summarized in Table 15, we do achieve a further decrease in FLOPs and gain in accuracy. Considering that our pruning strategies are remarkably simple, the values obtained are reasonable and certify their generalization ability.

5.3. Pruning relevant and random channels

When a layer is pruned using the pruning-fixing-retraining strategy, the filters that have not been deleted, and hence have been maintained as relevant, are no longer updated. If the selection of important filters was correct, the resulting network should perform well. The more wrong filters are chosen, and therefore fixed and not updated, the worse the performance should be. To analyze this issue in detail, we compare the PCA-N approach with the elimination of random channels and of those with the highest norm l_j^i values (1). Note that SegNet is not considered for this analysis, since it is only pruned at once.

As Table 16 shows, next convolution influence criterion (lowest) outperforms pruning the largest l_j^i, indicating the relevance of channels with great weight norm l_j^i. The CamVid dataset is an exception, probably because only 3.9% of the parameters are removed. Random selection exhibits pretty good performance in some cases (such as in CRBD and CIFAR-10) whereas reduces effectiveness in others (in particular in SBD and CamVid). In theory, random selection may be a powerful criterion (Bengio et al., 2013). Still, as demonstrated in Luo et al. (2017), it is not robust and not applicable in practice. In fact, our results show that there is an important drop in some of the CamVid metrics although only a 3.9% has been pruned.

6. Conclusions

This paper proposes two filter pruning strategies for compression of deep CNN models, a layer pruning method specially design for U-Net, and a fine-tuning strategy. The approach is based on the distribution of importance scores and the combination of known (but not less effective) techniques, such as PCA. The pruning strategies are general, valid for models other than those studied, which do not require any special hardware or software support, or intensive iterative processes.

The paper also presents a detailed performance evaluation on semantic segmentation and classification settings, using U-Net, DeepLabv3+, SegNet, and VGG-16 networks. Note that these models constitute a representative set of different variants, including distinct skip connections. Significant reduction is achieved maintaining networks efficiency, showing the generalization capability of our approach. We also analyze the relationship between method performance and the dataset under test in order to understand how the networks learn (xAI). We observe that care should be taken when features with distinct variance are presented, since prioritizing one over another may lead to a drop in performance.

In the future, we would like to conduct more research on these relationships and on how pruning helps in networks explainability. Moreover, we wish to delve into the behavior of more models (complex and simpler) and larger datasets (such as cityscapes, although preliminary studies did not show significant improvements in pruning and only variations due to image size, as mentioned above) to verify and reaffirm the generalization ability of our approach, with the clear evidence shown in this work. Once the efficient methodology shown in this study has been established, and although experiments have been carried out, mainly based on trial and error tests, it would be desirable to analyze thoroughly the evolution of the percentage of pruned channels in convolutional layers and on the influence of the parameters of the methods in the results.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is publicly available.

Acknowledgments

This work has been supported by the Research Project IA-GES-BLOOM-CM (Y2020/TCS-6420) of the Synergic program of the Comunidad Autónoma de Madrid and by the Research Project AMPBAS (RTI2018-098962-B-C21) of the National Societal Challenges program of the Spanish Ministry of Science, Innovation and Universities. The first author, Clara I. López-González, is supported by a FPU Ph.D. scholarship from the Spanish Ministry of Universities. The work by Fredy Barrientos-Espillco has been funded by PRONABEC from Ministry of Education of Peru. The authors thank the anonymous referees for their very valuable comments and suggestions.

Appendix A. Computational complexity evaluation

The aim of this section is to illustrate the computational complexity of the different approaches presented.

Given a convolutional layer C_i, two criteria were proposed to compute the relevance of its channels. On the one hand, next convolution influence criterion l_j^i is based on the l_1-norm of the weight tensor $W^{i+1}(\cdot,\cdot,j,\cdot) \in \mathbb{R}^{k_{i+1}\times k_{i+1}\times n_{i+1}}$ of the next convolution C_{i+1}, as Eq. (1) shows. The computational complexity of this calculation lies mainly in the number of filters n_{i+1}, since the kernel dimension is computationally negligible. Therefore, it is given by $O(n_{i+1})$. It should be noted that this complexity is not affected by the choice of the next convolutional layer $i+1$ instead of the current one i, as Li et al. (2017) does. In the latter case, the norm of $W^i(\cdot,\cdot,j,\cdot) \in \mathbb{R}^{k_i\times k_i\times n_{i-1}}$ is computed, which leads to $O(n_{i-1})$. Indeed, convolutional filters of a network are increasing in powers of two. Hence, the difference between two consecutive layers

Table 16
Comparison of evaluation metrics of the compressed networks obtained when pruning following PCA and choosing either the channels with lowest norm weight (PCA-N), either the largest or by doing it randomly.

Model	Acc.	Mean acc.	Mean IoU	W. IoU	Mean BF	Val. loss	Val. acc. (%)
CRBD lowest	0.9528	0.9587	**0.8781**	0.9135	0.9547	**0.6108**	**96.19**
CRBD largest	0.9492	0.9011	0.8455	0.9049	0.9610	2.7530	93.40
CRBD random	**0.9533**	**0.9637**	0.8695	**0.9157**	**0.9645**	0.6948	95.35
SBD lowest	**0.9901**	**0.9872**	**0.7768**	**0.9844**	**0.8584**	**1.6723**	**98.90**
SBD largest	0.9636	0.3333	0.3212	0.9285	0.7920	34.881	95.87
SBD random	0.9812	0.9444	0.6892	0.9752	0.7188	6.1937	97.93
CamVid lowest	0.9145	**0.8790**	**0.7362**	0.8527	**0.7760**	7.1384	92.92
CamVid largest	0.9111	0.8770	0.7252	0.8483	0.7677	**6.8167**	92.73
CamVid random	**0.9260**	0.8208	0.6968	**0.8695**	0.7655	8.0607	**94.18**
Model	Acc.	Mean acc.		Mean prec.		Val. loss	Val acc. (%)
CIFAR-10 lowest	**0.8558**	**0.9711**		**0.8565**		**0.5504**	**88.66**
CIFAR-10 largest	0.8410	0.9682		0.8419		0.6845	86.79
CIFAR-10 random	0.8444	0.9689		0.8438		0.5869	87.27

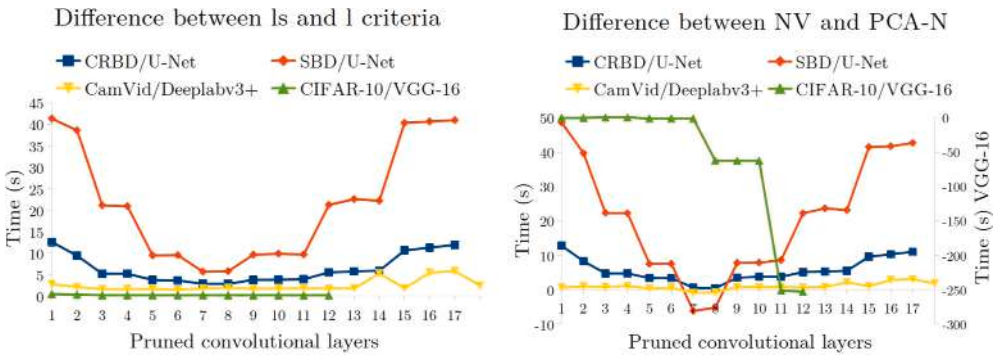

Fig. A.11. On the left, time difference between the calculation of the importance scores of all layer channels for the ls_j^i and l_j^i criteria. On the right, time difference between the calculation of the channels to prune for NV and PCA-N methods. The n number on the x-axis refers to the nth convolutional layer pruned in the corresponding network.

will be at most a multiplication by 2, i.e., $n_{i+1} = 2 \cdot n_{i-1}$ (or vice versa if they belong to the decoder). This makes the computational complexity of both calculations (Li et al., 2017 and ours) the same, since they only differ in a constant: $O(n_{i+1}) = O(2n_{i-1}) = O(n_{i-1})$. It is worth mentioning that, in general, the complexity of computing this type of weight tensor norms is $O(2^p)$, where 2^p refers to the number of filters and p increases as we go deeper into the network. Given that p does not become greater than 12, this is not a problem in computational terms.

On the other hand, for the next convolution influence-variance criterion we need to compute ls_j^i in Eq. (3). Since the feature maps of the convolutional output are required for variance calculation, there is the added complexity of needing to feed images into the CNN again. We therefore opted for an experimental instead of theoretical approach. In particular, we compare how long does it take for the network to compute each of these relevance criteria when pruning a layer. This is shown on the left of Fig. A.11 for all the datasets and CNNs studied except for SegNet, with which only PCA-N was used, so there is no possibility of comparison between methods. For each convolutional layer pruned, we display the difference in time between computing $\{l_j^i\}_{j=1}^{n_i}$ and $\{ls_j^i\}_{j=1}^{n_i}$, i.e., each of the criteria for all channels.

As expected, positive values corroborate that more effort is required for the ls_j^i criterion in all the networks, increase that is directly related to the variance calculation. Note that for VGG-16 there is almost no difference. This network's architecture is much simpler than that of the other CNNs. In addition, the dimension of the feature maps is quite small. All the above explains why computing the variance takes about the same amount of time as computing the norm. This relationship between the dimension of the feature maps and the computation time of ls_j^i (the smaller the dimension the shorter the time) can also be observed in U-Net encoder–decoder model (SBD in Fig. A.11, left). The maps' dimension is large at the beginning, it reduces as we go through the encoder to increase again in the decoder.

Nevertheless, this is not a realistic analysis of the computational complexity of our methods, since PCA-N does not only involve the calculation of the norm l_j^i but also that of the layers optimal dimension through PCA. Given that some effort is needed to obtain the latter, we performed a similar comparison but between the complete pruning strategies. In particular, for each of the convolutional layers studied, we obtain how long does it take to determine which channels to prune with both of our methods, i.e., PCA-N and NV. Then, we display the difference on the right of Fig. A.11 for all the networks and datasets employed (again excluding SegNet). Note that VGG-16 has its own y-axis on the right, due to scale discrepancies.

We observe how the difference exhibited in the previous analysis (Fig. A.11, left) reduces. In particular, for the bridge layers in U-Net (connecting the encoder with the decoder), this difference is negative. That is, PCA-N needs more effort than NV to determine the channels in these layers. This is related with the fact stated in Section 3.3.2: when the dimensions of the feature maps are small, a change in a single pixel is significant, and we may need several channels to explain the required cumulative variance, what makes it more difficult for PCA to obtain the optimal dimension. This is clearly evident in VGG-16, where the large negative values (right y-axis in Fig. A.11) implies that PCA-N takes much more time than NV. Note that the feature maps through all this network are quite small. Hence, PCA requires a lot of effort, which results in increased computational time.

All in all, although ls_j^i needs more effort than l_j^i, this may not be reflected in the complete pruning strategy. Finally, we would like to point out that many effective state-of-the-art strategies need to feed images into CNN again (Garg et al., 2020; Li et al., 2020; Shao et al., 2021).

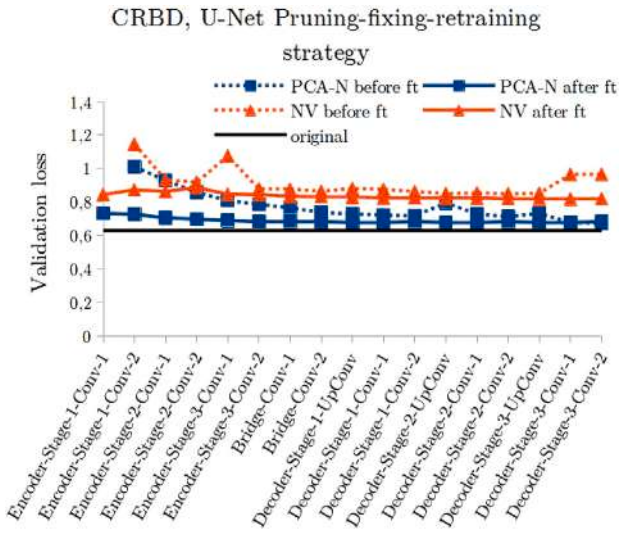

Fig. B.12. Validation loss before and after each fine-tuning (ft) step in the pruning-fixing-retraining strategy for U-Net on CRBD.

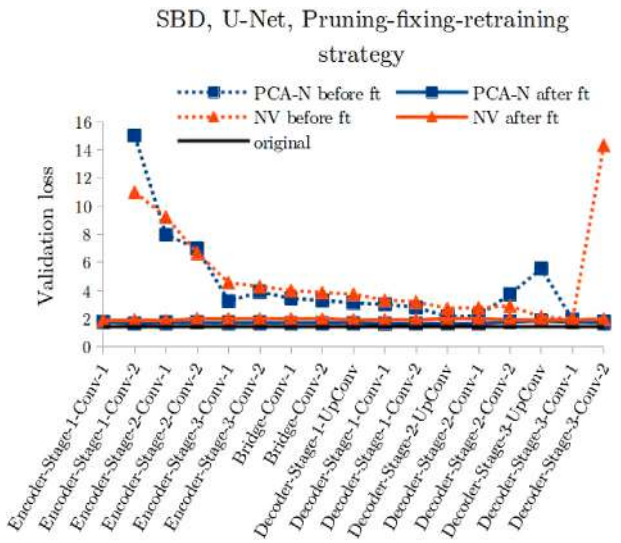

Fig. B.13. Validation loss before and after each fine-tuning (ft) step in the pruning-fixing-retraining strategy for U-Net on SBD.

Appendix B. Network convergence in the fine-tuning strategy

In this section, we show the impact on network convergence of each fine-tuning of the pruning-fixing-retraining strategy. In particular, for our two proposals PCA-N *it* and NV *it*, we examine whether each retraining gets the converged result. For this purpose, the value of the validation loss before and after fine-tuning is recorded for each layer.

Results for all pairs of CNNs and datasets are shown in Fig. B.12, B.13, B.14, and B.15. SegNet is omitted as it has not been pruned with the iterative strategy. The dotted line refers to the validation loss after pruning but before retraining, whereas the solid line corresponds to the loss after the fine-tuning. In some cases we exclude the loss of the first layer before retraining, in order to preserve the scale of the graph and properly observe the evolution of this metric through the layers. Being the first to be pruned, the value of the validation loss may be high, causing a change in the scale. However, it recovers after retraining, as shown by the solid line.

Although we only retrain for a few epoch (1/3 of the original number), the CNNs are able to recover from the pruning, which is evidenced by the consistency of the validation loss and its proximity to the loss of the original network (dark line). Therefore, converged results are obtained. In addition, the low number of epochs used let us conclude that convergence is not only due to the fine-tuning, but also to proper pruning, including fixing the appropriate channels.

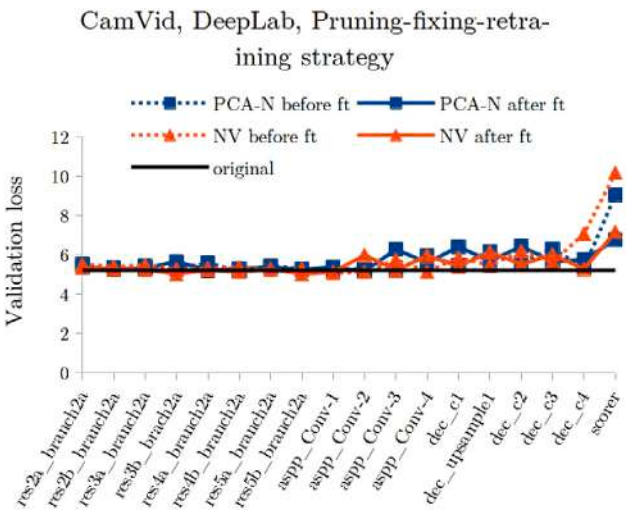

Fig. B.14. Validation loss before and after each fine-tuning (ft) step in the pruning-fixing-retraining strategy for DeepLabv3+ on CamVid.

Appendix C. Ablation study on (γ, α, β)

A description of the search performed over the parameters (γ, α, β) is presented in this section. Throughout Section 4, the proposed distribution-based method (Section 3.3.2) has proven to be highly effective in increasing pruning with respect to PCA (Section 3.3.1). How the former depends on the parameters (γ, α, β), how they can be interpreted, and recommendations for choosing their value were given in Section 3.3.2. Meanwhile, our choice for the experiments carried out was explained in Section 4.1, where we declare that exhaustive search is needed in the case of VGG-16 and DeepLabv3+. In those cases, we proceed as follows.

Set a lower bound for the percentage of pruned parameters (e.g., we pick 86% for VGG-16) and an upper bound for the validation loss and the drop in accuracy. Then, for chosen intervals of γ, α, and β values, we obtain the pruned model by fixing two of them and varying the other. Prior to conducting the final retraining and evaluation, we check that

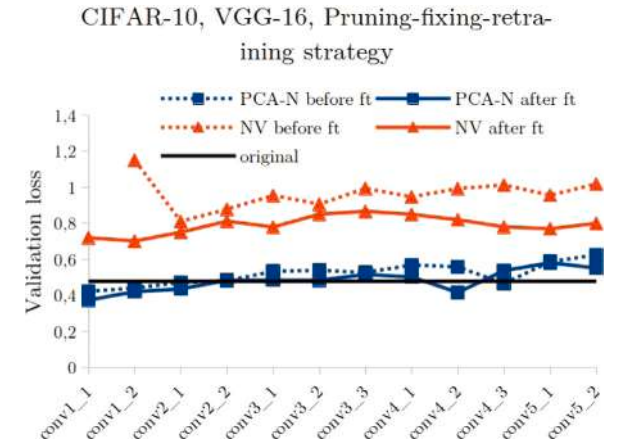

Fig. B.15. Validation loss before and after each fine-tuning (ft) step in the pruning-fixing-retraining strategy for VGG-16 on CIFAR-10.

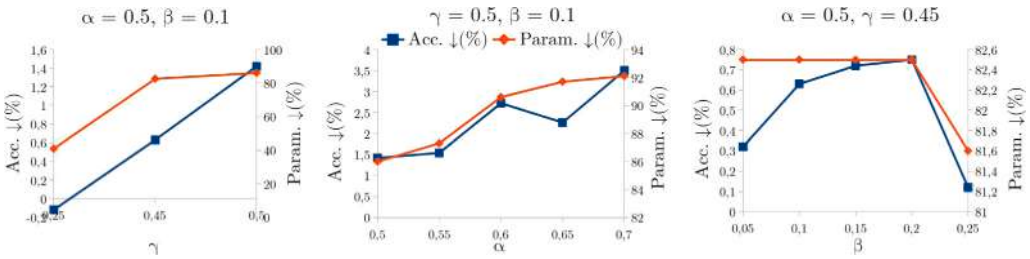

Fig. C.16. The effect of γ, α and β on the NV *once* strategy for VGG-16 on CIFAR-10.

the bounds established are satisfied. The resultant networks are compared and the best according to our priorities is selected (e.g., pruning vs. performance).

The outcome, in the case of VGG-16 on CIFAR-10, for some of these parameter combinations is displayed in Fig. C.16. The NV *once* (40) method was employed. Two fixed and one variable parameter allow us to observe the effect of each of them in terms of accuracy and pruning. On the one hand, the first two graphs in Fig. C.16 show that increasing γ and α gives more pruning, but also greater drop in accuracy. In addition, they produce the largest changes in the percentage of network parameters removed. On the other hand, the third graph exposes that modifying β is not as effective for pruning, as the biggest difference is 0.1%. This is in line with the purpose of β: control under-pruning. When there is proper pruning, small changes in β should not affect. Variations in accuracy with β may be due to small changes in the location of the pruned parameters along the network.

References

Ahmed, W., Ansari, S., Hanif, M., & Khalil, A. (2022). PCA driven mixed filter pruning for efficient convNets. *PLoS One*, *17*(1), Article e0262386. http://dx.doi.org/10.1371/journal.pone.0262386.

Arthur, D., & Vassilvitskii, S. (2006). *k-means++: The advantages of careful seeding*: *Technical report*, Stanford, Stanford InfoLab, URL http://ilpubs.stanford.edu:8090/778/.

Ayinde, B. O., Inanc, T., & Zurada, J. M. (2019). Redundant feature pruning for accelerated inference in deep neural networks. *Neural Networks*, *118*, 148–158. http://dx.doi.org/10.1016/j.neunet.2019.04.021.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(12), 2481–2495. http://dx.doi.org/10.1109/tpami.2016.2644615.

Bendig, J., Yu, K., Aasen, H., Bolten, A., Bennertz, S., Broscheit, J., Gnyp, M. L., & Bareth, G. (2015). Combining UAV-based plant height from crop surface models, visible, and near infrared vegetation indices for biomass monitoring in barley. *International Journal of Applied Earth Observation and Geoinformation*, *39*, 79–87. http://dx.doi.org/10.1016/j.jag.2015.02.012.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. http://dx.doi.org/10.1109/tpami.2013.50.

Brostow, G. J., Shotton, J., Fauqueur, J., & Cipolla, R. (2008). Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision ECCV* (pp. 44–57).

Camargo Neto, J. (2004). A combined statistical-soft computing approach for classification and mapping weed species in minimum-tillage systems. *ETD collection for University of Nebraska - Lincoln, NE*.

Chen, X., Zhang, Y., & Wang, Y. (2022). MTP: Multi-task pruning for efficient semantic segmentation networks. In *2022 IEEE international conference on multimedia and expo* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/icme52920.2022.9859583.

Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Computer vision – ECCV 2018* (pp. 833–851). Cham: Springer International Publishing.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., & de Freitas, N. (2013). Predicting parameters in deep learning. In *Proceedings of the conference on advances in neural information processing systems*: *vol. 26*, (pp. 2148–2156). Curran Associates, Inc., http://dx.doi.org/10.5555/2999792.2999852.

Di Cicco, M., Potena, C., Grisetti, G., & Pretto, A. (2017). Automatic model based dataset generation for fast and accurate crop and weeds detection. In *Proc. of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 5188–5195). http://dx.doi.org/10.1109/IROS.2017.8206408.

Dinsdale, N. K., Jenkinson, M., & Namburete, A. I. (2022). STAMP: Simultaneous training and model pruning for low data regimes in medical image segmentation. *Medical Image Analysis*, *81*, Article 102583. http://dx.doi.org/10.1016/j.media.2022.102583.

Ditschuneit, K., & Otterbach, J. S. (2022). Auto-compressing subset pruning for semantic image segmentation. In *Lecture notes in computer science* (pp. 20–35). Springer International Publishing, http://dx.doi.org/10.1007/978-3-031-16788-1_2.

Garg, I., Panda, P., & Roy, K. (2020). A low effort approach to structured CNN design using PCA. *IEEE Access*, *8*, 1347–1360. http://dx.doi.org/10.1109/ACCESS.2019.2961960.

Hague, T., Tillett, N., & Wheeler, H. (2006). Automated crop and weed monitoring in widely spaced cereals. *Precision Agriculture*, *7*, 21–32. http://dx.doi.org/10.1007/s11119-005-6787-1.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M., & Dally, W. (2016). EIE: Efficient inference engine on compressed deep neural network. In *ISCA* (pp. 243–254). http://dx.doi.org/10.1109/ISCA.2016.30.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. In *Proceedings of the conference on advances in neural information processing systems*: *vol. 28*, (pp. 1135–1143). Curran Associates, Inc..

Hassibi, B., & Stork, D. (1992). Second order derivatives for network pruning: Optimal brain surgeon. In *Proceedings of the conference on advances in neural information processing systems*: *vol. 5*, (pp. 164–171). Morgan-Kaufmann.

He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Y. (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration. In *2019 IEEE/CVF conference on computer vision and pattern recognition* (pp. 4335–4344). IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2019.00447.

He, Z., Qian, Y., Wang, Y., Wang, B., Guan, X., Gu, Z., Ling, X., Zeng, S., Wang, H., & Zhou, W. (2022). Filter pruning via feature discrimination in deep neural networks. In *Lecture notes in computer science* (pp. 245–261). Springer Nature Switzerland, http://dx.doi.org/10.1007/978-3-031-19803-8_15.

He, W., Wu, M., & Lam, S.-K. (2021). ACSL: Adaptive correlation-driven sparsity learning for deep neural network compression. *Neural Networks*, *144*, 465–477. http://dx.doi.org/10.1016/j.neunet.2021.09.012.

He, W., Wu, M., Liang, M., & Lam, S.-K. (2021). CAP: Context-aware pruning for semantic segmentation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 960–969).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition* (pp. 770–778). http://dx.doi.org/10.1109/CVPR.2016.90.

Hinton, G., Deng, l., Yu, D., Dahl, G., Mohamed, A. r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*, 82–97. http://dx.doi.org/10.1109/MSP.2012.2205597.

Hinton, G. E., Krizhevsky, A., & Wang, S. D. (2011). Transforming auto-encoders. In *Artificial neural networks and machine learning – ICANN 2011* (pp. 44–51). Berlin, Heidelberg: Springer Berlin Heidelberg.

Hinton, G., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554. http://dx.doi.org/10.1162/neco.2006.18.7.1527.

Kataoka, T., Kaneko, T., Okamoto, H., & Hata, S. (2003). Crop growth estimation system using machine vision. In *Proceedings 2003 IEEE/ASME international conference on advanced intelligent mechatronics, Vol. 2* (pp. b1079–b1083). http://dx.doi.org/10.1109/AIM.2003.1225492.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Proceedings of the conference on advances in neural information processing systems*: *vol. 25*, (pp. 84–90). Curran Associates, Inc..

Law, H., & Deng, J. (2020). CornerNet: Detecting objects as paired keypoints. *International Journal of Computer Vision*, *128*, 642–656. http://dx.doi.org/10.1007/s11263-019-01204-1.

LeCun, Y., Denker, J., & Solla, S. (1989). Optimal brain damage. In *Proceedings of the conference on advances in neural information processing systems*: *vol. 2*, (pp. 598–605). Morgan-Kaufmann.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017). Pruning filters for efficient ConvNets. http://dx.doi.org/10.48550/arXiv.1608.08710, arXiv:1608.08710.

Li, H., Ma, C., Xu, W., & Liu, X. (2020). Feature statistics guided efficient filter pruning. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence* IJCAI-20, (pp. 2619–2625). International Joint Conferences on Artificial Intelligence Organization, http://dx.doi.org/10.24963/ijcai.2020/363.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *2017 IEEE international conference on computer vision* ICCV, (pp. 2755–2763). http://dx.doi.org/10.1109/ICCV.2017.298.

Liu, J., Zhuang, B., Zhuang, Z., Guo, Y., Huang, J., Zhu, J., & Tan, M. (2021). Discrimination-aware network pruning for deep model compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1. http://dx.doi.org/10.1109/tpami.2021.3066410.

Lou, A., & Loew, M. (2021). CFPNet: Channel-wise feature pyramid for real-time semantic segmentation. http://dx.doi.org/10.48550/ARXIV.2103.12212.

Louhaichi, M., Borman, M., & Johnson, D. (2001). Spatially located platform and aerial photography for documentation of grazing impacts on wheat. *Geocarto International*, *16*, http://dx.doi.org/10.1080/10106040108542184.

Luo, J. H., Wu, J., & Lin, W. (2017). ThiNet: A filter level pruning method for deep neural network compression. In *2017 IEEE international conference on computer vision* (pp. 5068–5076). IEEE Computer Society, http://dx.doi.org/10.1109/iccv.2017.541.

MathWorks (2022a). Semantic segmentation using deep learning. URL https://es.mathworks.com/help/vision/ref/deeplabv3pluslayers.html.

MathWorks (2022b). U-Net layers. URL https://es.mathworks.com/help/vision/ref/unetlayers.html.

Meyer, G. E., Hindman, T. W., & Laksmi, K. (1999). Machine vision detection parameters for plant species identification. *vol. 3543*, In *Precision agriculture and biological quality* (pp. 327–335). SPIE, International Society for Optics and Photonics, http://dx.doi.org/10.1117/12.336896.

Meyer, G. E., & Neto, J. C. (2008). Verification of color vegetation indices for automated crop imaging applications. *Computers and Electronics in Agriculture*, *63*(2), 282–293. http://dx.doi.org/10.1016/j.compag.2008.03.009.

Ribeiro, A., Fernandez-Quintanilla, C., Barroso, J., & García-Alegre, M. (2005). Development of an image analysis system for estimation of weed pressure. *Precision Agriculture 2005, ECPA 2005*, 169–174.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention* (pp. 234–241). Cham: Springer International Publishing.

Shao, L., Zuo, H., Zhang, J., Xu, Z., Yao, J., Wang, Z., & Li, H. (2021). Filter pruning via measuring feature map information. *Sensors*, *21*(19), 6601. http://dx.doi.org/10.3390/s21196601.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International conference on learning representations*.

Singh, P., Kumar, V. V., Rai, P., & Namboodiri, V. P. (2019). Play and prune: Adaptive filter pruning for deep model compression. In *IJCAI international joint conference on artificial intelligence*, *Proceedings of the 28th international joint conference on artificial intelligence* (pp. 3460–3466). International Joint Conference on Artificial Intelligence, http://dx.doi.org/10.48550/arXiv.1905.04446.

Tjoa, E., & Guan, C. (2021). A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(11), 4793–4813. http://dx.doi.org/10.1109/TNNLS.2020.3027314.

Vidovic, I., Cupec, R., & Hocenski, Ž. (2016). Crop row detection by global energy minimization. *Pattern Recognition*, *55*, 68–86.

Woebbecke, D. M., Meyer, G. E., Bargen, K. V., & Mortensen, D. A. (1995). Shape features for identifying young weeds using image analysis. *Transactions on American Society of Agricultural Engineering*, *38*, 271–281.

Yu, W., Yang, K., Bai, Y., Yao, H., & Rui, Y. (2014). Visualizing and comparing convolutional neural networks. http://dx.doi.org/10.48550/ARXIV.1412.6631.

Zhang, Z., Liu, Q., & Wang, Y. (2018). Road extraction by deep residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, *15*(5), 749–753. http://dx.doi.org/10.1109/LGRS.2018.2802944.

Zhang, W., & Wang, Z. (2022). PCA-pruner: Filter pruning by principal component analysis. *Journal of Intelligent & Fuzzy Systems*, *Pre-press*, 1–11. http://dx.doi.org/10.3233/jifs-211555.