Sauron U-Net: Simple automated redundancy elimination in medical image segmentation via filter pruning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present Sauron, a filter pruning method that eliminates redundant feature maps by discarding the corresponding filters with automatically-adjusted layer-specific thresholds. Furthermore, Sauron minimizes a regularization term that, as we show with various metrics, promotes the formation of feature maps clusters. In contrast to most filter pruning methods, Sauron is single-phase, similarly to typical neural network optimization, requiring fewer hyperparameters and design decisions. Additionally, unlike other cluster-based approaches, our method does not require pre-selecting the number of clusters, which is non-trivial to determine and varies across layers. We evaluated Sauron and three state-of-the-art filter pruning methods on three medical image segmentation tasks. This is an area where filter pruning has received little attention and where it can help building efficient models for medical grade computers that cannot use cloud services due to privacy considerations. Sauron achieved models with higher performance and pruning rate than the competing pruning methods. Additionally, since Sauron removes filters during training, its optimization accelerated over time. Finally, we show that the feature maps of a Sauron-pruned model were highly interpretable. The Sauron code is publicly available at `https://github.com/blindedrepository`.

1 Introduction

Pruning is the process of eliminating unnecessary parameters to obtain compact models and accelerate their inference. There are two main strategies for pruning convolutional neural networks (CNNs): weight pruning and filter pruning. In weight pruning, weights for unimportant connections are zeroed without consideration of the network structure, leading, in practice, to sparse weight matrices [21, 12, 10, 11, 40]. On the other hand, filter pruning methods eliminate CNNs filters directly. Thus, unlike weight-pruned models, utilizing filter-pruned networks efficiently requires no specialized hardware or software [7, 35]. Most pruning methods have been developed or evaluated exclusively for natural image classification. Other tasks, such as medical image segmentation, have received significantly less attention [32]. In medical imaging, small models can enable computationally-limited medical grade computers to segment medical images that cannot be uploaded to a cloud server due to privacy reasons. Moreover, models with a few filters can be easier to interpret than large models, which is crucial not only in clinical applications but also in research. Motivated by these possibilities, we propose a filter pruning method called Sauron that generates small CNNs. We demonstrate its

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

application to prune U-Net-like networks [36], bringing together filter pruning and medical image segmentation.

Sauron applies filter pruning during optimization in a *single phase*, while most filter pruning frameworks consist of three distinct phases: Pre-training the model, pruning its filters, and fine-tuning to compensate for the loss of accuracy (or re-training from scratch [27, 6]). Other approaches combine pruning with training [46, 48, 13, 38] or fine-tuning [29, 26], resulting in two-phase frameworks, and other methods repeat these phases multiple times [46, 29, 4]. Single-phase filter pruning methods [48], such as Sauron, are advantageous since they require fewer hyperparameters and design decisions, including the number of epochs for training and fine-tuning, pruning iterations, or whether to combine pruning with training or fine-tuning. In particular, Sauron does not insert additional parameters into the optimized architecture to identify filter candidates for pruning, such as channel importance masks [6, 29, 15, 26, 16]. This avoids potential optimization hindrance and requires less extra training time and GPU memory.

Sauron facilitates and promotes the formation of feature map clusters by optimizing a regularization term, and, unlike previous cluster-based approaches [9, 13, 4], Sauron does not enforce the number of these clusters. Since these clusters vary depending on the training data and across layers, the optimal number of feature maps per cluster is likely to differ. Thus, determining the number of clusters is not trivial and may limit the accuracy and the pruning rate.

Our specific contributions are the following:

- We introduce Sauron, a single-phase filter pruning method that resembles the typical CNN optimization, making it easier to use, and that does not add any additional parameters to the optimized architecture.
- We show that Sauron promotes the formation of feature map clusters by optimizing a regularization term.
- We compare Sauron to other methods on three medical image segmentation tasks, where Sauron resulted in more accurate and compressed models.
- We show that the feature maps generated by a model pruned with Sauron were highly interpretable.
- We publish Sauron and the code to run all our experiments at `https://github.com/blindedrepository`.

2 Previous work

Filter importance Most filter pruning approaches rely on ranking filters to eliminate the unimportant filters. The number of eliminated filters can be determined by either a fixed [3] or an adaptive threshold [38]. Filter importance can be found via particle filtering [3] or it can be computed via heuristic relying on measures such as L_p norms [23, 44, 38], entropy [28], or post-pruning accuracy [1]. Pruning methods can include extra terms in the loss function, such as group sparsity constraints, although these extra terms guarantee no sparsity in CNNs [45]. Other methods aim to learn filter importance by incorporating channel importance masks into CNNs' architectures [6, 29, 15, 26, 16]. However, these adjustments modify the architectures to be optimized, increasing the required GPU memory during training, optimization time, and potentially hindering the optimization. Alternatively, other methods consider the scaling factor of batch normalization layers as channel importance [45, 48], but in e.g. medical image segmentation, batch normalization is occasionally replaced by other normalization layers due to the small mini-batch size [18].

Difference minimization Methods that remove filters while trying to preserve characteristics such as classification accuracy [27], Taylor-expansion-approximated loss [46], and the feature maps [47, 42, 44, 30] of the original unpruned models. A disadvantage of these methods is that they require a large GPU memory to avoid loading and unloading the models in memory constantly, which would

Algorithm 1 Sauron

Input: training data: $\mathcal{D}$.

Given: λ, maximum threshold τ_{max}, $epochs$, percentage of pruned filters μ, patience ρ, number of steps κ.
Initialize: model's weights $\mathbf{W} \leftarrow \{\mathbf{W}^l, 1 \leq l \leq L\}$, layer-specific thresholds $\boldsymbol{\tau} \leftarrow \{\tau_l = 0, 1 \leq l \leq L\}$
for $e = 1; e \leq epochs$ **do**
    for $b = 1; b \leq N$ **do** *# Mini batches*
        Compute predictions $\hat{\boldsymbol{y}}$, loss $\mathcal{L}$, δ_{opt} (Eq. (2)), $\boldsymbol{\delta}_{prune}$ (Eq. (3)) *# Forward pass*
        Update $\boldsymbol{\theta}$ *# Backward pass*
    end for
    for $l = 1; l \leq L$ **do** *# Pruning step*
        ## Procedure 1: Increasing τ_l ##
        C1: Training loss is converging; C2: Validation loss is not improving
        C3: Less than μ% of filters pruned in $(e-1)$; C4: τ_l has not increased in last ρ epochs
        if (C1 $\wedge$ C2 $\wedge$ C3 $\wedge$ C4) $\wedge$ $(\tau_l < \tau_{max})$ **then**
            $\tau_l \leftarrow \tau_l + \tau_{max}/\kappa$
        end if
        ## Procedure 2: Pruning ##
        $\mathbf{W}^l \leftarrow \{\mathbf{W}^l : \boldsymbol{d}^l > \tau_l\}$
    end for
end for

Output: Pruned CNN.

slow down the training. Furthermore, since finding the appropriate filters for their elimination is NP-hard, certain methods resorted to selecting filters based on their importance [47, 44, 46], or via genetic [27] or greedy [30] algorithms.

Redundancy elimination Approaches, including Sauron, that identify redundant filters by computing a similarity metric among all [43, 39] or within clusters of filters/feature maps [13, 9, 4]. Previously, cluster-based approaches have considered redundant those within-cluster filters near the Euclidean center [9] and median [13], or filters with similar L_1 norm over several training epochs [4]. A disadvantage of these approaches is an extra “number of clusters" hyperparameter, which is data dependent and the same hyperparameter value might not be optimal across layers. Other methods have used Pearson's correlation between the weights [43] or between the feature maps [39] within the same layer, and feature maps' rank [25] to indicate redundancy, although, their computations are more expensive than utilizing distances as in cluster-based methods.

3 Sauron

In this section, we present our approach to filter pruning, which we call **S**imple **AU**tomated **R**edundancy eliminati**ON** (Sauron). Sauron optimizes, jointly with the loss function, a regularization term that leads to clusters of feature maps at each convolutional layer, accentuating the redundancy of CNNs. It then eliminates the filters corresponding to the redundant feature maps by using automatically-adjusted layer-specific thresholds. Sauron requires minimal changes from the typical neural network optimization since it prunes and optimizes CNNs jointly, i.e., training involves the usual forward-backward passes and a pruning step after each epoch. Moreover, Sauron does not integrate optimizable parameters, such as channel importance masks [6, 29, 15, 26, 16], into the CNN architecture. This avoids complicating the optimization task and increasing the training time and the required GPU memory. Algorithm 1 summarizes our method.

3.1 Preliminaries

Let $\mathcal{D} = \{\mathbf{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ represent the training set, where $\mathbf{x}_i$ denotes image i, $\mathbf{y}_i$ its corresponding segmentation, and N is the number of images. Let $\mathbf{W}^l \in \mathbb{R}^{s_{l+1} \times s_l \times k \times k}$ be the weights, composed by $s_{l+1}s_l$ filters of size $k \times k$ at layer l, where s_{l+1} denotes the number of output channels, s_l the number of input channels, and k is the kernel size. Given feature maps $\mathbf{O}^l \in \mathbb{R}^{s_l \times h \times w}$ of $h \times w$ image dimensions, the feature maps $\mathbf{O}^{l+1} \in \mathbb{R}^{s_{l+1} \times h \times w}$ at the next layer are computed as

$$\mathbf{O}^{l+1} = \sigma(Norm(\mathbf{W}^l * \mathbf{O}^l)), \tag{1}$$

where * is the convolution operation, $Norm$ is a normalization layer, and σ is an activation function. For simplicity, we omit the bias term in Eq. (1), and we include all CNN's parameters in $\boldsymbol{\theta} = \{\boldsymbol{W}^1, \dots, \boldsymbol{W}^L\}$, where L is the number of layers. We denote the predicted segmentation of the image $\mathbf{x}_i$ by $\hat{\boldsymbol{y}}_i$.

3.2 Forward pass

Sauron minimizes a loss $\mathcal{L}$ consisting of Cross Entropy $\mathcal{L}_{CE}$, Dice loss $\mathcal{L}_{Dice}$ [31], and a novel channel distance regularization term δ_{opt}: $\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{Dice} + \lambda\delta_{opt}$, where

$$\delta_{opt} = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{s_{l+1}}\sum_{r=2}^{s_{l+1}}||\phi(\boldsymbol{O}_1^l;\omega) - \phi(\boldsymbol{O}_r^l;\omega)||_2, \tag{2}$$

λ is a hyperparameter that balances the contribution of δ_{opt}, and ϕ denotes average pooling with window size and strides ω. Before computing δ_{opt}, feature maps $\boldsymbol{O}_1^l$ and $\boldsymbol{O}_{-1}^l$ (all channels except the first) are normalized to the range $[0, 1]$ via min-max normalization, as we experimentally found this normalization strategy to be the best (see Appendix A). For pruning, Sauron computes distances between a randomly-chosen feature map $\pi \in \{1, \dots, s_{l+1}\}$ and all the others: $\boldsymbol{\delta}_{prune} = \{d_r^l / \max_r d_r^l : l = 1, \dots, L, r = 1, \dots, \pi - 1, \pi + 1, \dots, s_{l+1}\}$, where

$$d_r^l = ||\phi(\boldsymbol{O}_\pi^l;\omega) - \phi(\boldsymbol{O}_r^l;\omega)||_2. \tag{3}$$

Importantly, π is different in every layer and epoch, enabling Sauron to prune different feature map clusters. Moreover, since finding an appropriate pruning threshold requires the distances to lie within a known range, Sauron normalizes d_r^l such that their maximum is 1, i.e., $d_r^l \leftarrow d_r^l / \max_r(d_r^l)$.

3.3 Backward pass: δ_{opt} regularization

Optimized CNNs have been shown to have redundant weights and to produce redundant feature maps [13, 43] (Appendix E). By minimizing the extra regularization term δ_{opt}, CNNs further promote the formation of clusters, facilitating their subsequent pruning. δ_{opt} regularization makes those feature maps near the feature map in the first channel $\boldsymbol{O}_1^l$ (i.e., within the same cluster) even closer. At the same time, those feature maps that are dissimilar to $\boldsymbol{O}_1^l$ (i.e., in other clusters) become more similar to other feature maps from the same cluster, as it holds that $||\phi(\boldsymbol{O}_i^l;\omega) - \phi(\boldsymbol{O}_j^l;\omega)||_2 \leq ||\phi(\boldsymbol{O}_1^l;\omega) - \phi(\boldsymbol{O}_i^l;\omega)||_2 + ||\phi(\boldsymbol{O}_1^l;\omega) - \phi(\boldsymbol{O}_j^l;\omega)||_2$ for $i \neq j$, i.e., the right hand side—minimized via δ_{opt} regularization—is an upper bound of the left hand side. We demonstrate this clustering effect in Section 4.2. Furthermore, for pruning, we focus on the feature maps rather than on the weights since different non-redundant weights can lead to similar feature maps. Thus, eliminating redundant weights guarantees no reduction in feature maps redundancy.

3.4 Pruning step

Sauron employs layer-specific thresholds $\boldsymbol{\tau} = [\tau_1, \dots, \tau_L]$, where all τ_l are initialized to zero and increase independently (usually at a different pace) until reaching τ_{max}. This versatility is important as the ideal pruning rate differs across layers due to their different purpose (i.e., extraction of low- and high-level features) and their varied number of filters. Additionally, this setup permits utilizing high thresholds without removing too many filters at the beginning of the optimization, as feature maps may initially lie close to each other due to the random initialization. In consequence, pruning is embedded into the training and remains *always active*, portraying Sauron as a single-phase filter pruning method.

Procedure 1: Increasing τ_l Pruning with adaptively increasing layer-specific thresholds raises two important questions: how and when to increase the thresholds? Sauron increases the thresholds linearly in κ steps until reaching τ_{max}. Then, thresholds are updated once the model has stopped improving (C1 and C2 in Algorithm 1) and it has pruned only a few filters (C3). An additional "patience" hyperparameter ensures that the thresholds are not updated consecutively (C4). Conditions C1, . . . ,C4 are easy to implement and interpret, and they rely on heuristics commonly employed for detecting convergence.

Procedure 2: Pruning Sauron considers nearby feature maps to be redundant since they likely belong to the same cluster. In consequence, Sauron removes all input filters $\mathbf{W}^{l}_{:,s_l}$ whose corresponding feature map distances $\boldsymbol{\delta}_{prune}$ are lower than threshold τ_l. In contrast to other filter pruning methods, Sauron needs to store no additional information, such as channel indices, and the pruned models become more efficient *and* smaller. Additionally, since pruning occurs during training, Sauron accelerates the optimization of CNNs. After training, pruned models can be easily loaded by specifying the new post-pruning number of input and output filters in the convolutional layers.

3.5 Implementation

Sauron's simple design permits its incorporation into existing CNN optimization frameworks easily. As an example, in our implementation, convolutional blocks are wrapped into a class that computes δ_{opt} and $\boldsymbol{\delta}_{prune}$ effortlessly in the forward pass, and the pruning step is a callback function triggered after each epoch. This implementation, together with the code for running our experiments and processing the datasets, was written in Pytorch [33] and is publicly available at `https://github.com/blindedrepository`. In our experiments, we utilized an Nvidia GeForce GTX 1080 Ti (11GB), and a server with eight Nvidia A100 (40GB).

4 Experiments

In this section, we compare Sauron with other state-of-the-art filter pruning methods and conduct an ablation study to show the impact on pruning and performance of δ_{opt} regularization. We empirically demonstrate that the proposed δ_{opt} regularization increases feature map clusterability, and we visualize the feature maps of a Sauron-pruned model.

Datasets We employed three 3D medical image segmentation datasets: Rats, ACDC, and KiTS. *Rats* comprised 160 3D T2-weighted magnetic resonance images of rat brains with lesions [41], and the segmentation task was separating lesion from non-lesion voxels. We divided Rats dataset into 0.8:0.2 train-test splits, and the training set was further divided into a 0.9:0.1 train-validation split, resulting in 115, 13, and 32 images for training, validation, and test, respectively. *ACDC* included the Automated Cardiac Diagnosis Challenge 2017 training set [5] (CC BY-NC-SA 4.0), comprised by 200 3D magnetic resonance images of 100 individuals. The segmentation classes were background, right ventricle (RV), myocardium (M), and left ventricle (LV). We divided ACDC dataset similarly to Rats dataset, resulting in 144, 16, and 40 images for training, validation, and test, respectively. We only utilized ACDC's competition training set due to the limitation to only four submissions to the online platform of ACDC challenge. Finally, *KiTS* was composed by 210 3D images from Kidney Tumor Challenge 2019 training set, segmented into background, kidney and kidney tumor [14] (MIT). KiTS training set was divided into a 0.9:0.1 train-validation split, resulting in 183 and 21 images for training and validation. We report the results on the KiTS's competition test set (90 3D images). All 3D images were standardized to zero mean and unit variance. The train-validation-test divisions and computation of the evaluation criteria was at the subject level, ensuring that the data from a single subject was completely in the train set or in the test set, never dividing subject's data between train and test sets. See Appendix C for preprocessing details.

Model and optimization Sauron and the compared filter pruning methods optimized nnUNet [18] via deep supervision [22] with Adam [20] starting with a learning rate of 10^{-3}, polynomial learning

Table 1: Performance on Rats dataset.

Method	Lesion	
	Dice	HD95
nnUNet	0.94 ± 0.03	1.1 ± 0.3
Sauron	**0.94 ± 0.03**	1.1 ± 0.3
Sauron ($\lambda = 0$)	0.93 ± 0.03	1.2 ± 0.5
cSGD ($r = 0.5$)	0.86 ± 0.13	9.6 ± 16.8
FPGM ($r = 0.5$)	0.93 ± 0.04	**0.5 ± 0.5**
Autopruner	0.91 ± 0.04	0.8 ± 1.2

Table 2: Performance on ACDC dataset. **Bold**: best performance among pruning methods.

LV		M		RV	
Dice	HD95	Dice	HD95	Dice	HD95
0.91 ± 0.05	4.4 ± 3.0	0.90 ± 0.02	3.4 ± 5.8	0.95 ± 0.03	2.5 ± 1.8
0.90 ± 0.06	**4.7 ± 3.2**	**0.90 ± 0.02**	3.6 ± 8.0	**0.95 ± 0.03**	**2.7 ± 2.0**
0.89 ± 0.08	5.3 ± 4.4	**0.90 ± 0.02**	**2.4 ± 1.7**	**0.95 ± 0.03**	3.1 ± 3.0
0.10 ± 0.15	72.6 ± 74.1	0.54 ± 0.19	19.5 ± 35.6	0.64 ± 0.20	13.9 ± 8.2
0.57 ± 0.13	37.8 ± 7.3	0.89 ± 0.03	2.2 ± 1.6	0.00 ± 0.00	194.1 ± 23.5
0.88 ± 0.07	5.9 ± 4.6	0.88 ± 0.03	2.5 ± 1.7	**0.95 ± 0.03**	3.1 ± 3.0

Table 3: Performance on KiTS datasets.

Method	Kidney	Tumor
	Dice	Dice
nnUNet [17]	0.9595	0.7657
Sauron	**0.9564**	**0.7482**
Sauron ($\lambda = 0$)	0.9556	0.7352
cSGD [9] ($r = 0.5$)	0.9047	0.5207
FPGM [13] ($r = 0.5$)	0.9509	0.6830
Autopruner [29]	0.9167	0.5854

Table 4: Decrease in FLOPs with respect to the baseline nnUNet. **Bold**: highest decrease.

Method	Rats	ACDC	KiTS
nnUNet [17]	0.00%	0.00%	0.00%
Sauron	96.45%	**92.41%**	**93.02%**
Sauron ($\lambda = 0$)	**96.62%**	89.04%	85.82%
cSGD [9] ($r = 0.5$)	50.03%	49.80%	49.81%
FPGM [13] ($r = 0.5$)	50.00%	50.0%	49.98%
Autopruner [29]	83.61%	88.52%	82.00%

rate decay, and weight decay of 10^{-5}. During training, images were augmented with TorchIO [34] (see Appendix C). nnUNet is a self-configurable U-Net and the dataset optimized nnUNet architectures slightly differed on the number of filters, encoder-decoder levels, normalization layer, batch size, and number of epochs (see Appendix C).

Pruning Sauron decreased feature maps dimensionality via average pooling with window size and stride of $\omega = 2$, and utilized $\lambda = 0.5$ in the loss function, maximum pruning threshold $\tau_{max} = 0.3$, pruning steps $\kappa = 15$, and patience $\rho = 5$ (C4 in Algorithm 1). Additionally, we employed simple conditions to detect convergence for increasing the layer-specific thresholds $\boldsymbol{\tau}$. Convergence in the training loss (C1) was detected once the most recent training loss lay between the maximum and minimum values obtained during the training. We considered that the validation loss stopped improving (C2) once its most recent value increased with respect to all previous values. Finally, the remaining condition (C3) held true if the layer-specific threshold pruned less than 2% of the filters pruned in the previous epoch, i.e., $\mu = 2$.

4.1 Benchmark on three segmentation tasks

We optimized and pruned nnUNet [18] with Sauron, and we compared its performance with cSGD[1] [9], FPGM[2] [13], and Autopruner[3] [29] using a pruning rate similar to the one achieved by Sauron. Since cSGD and FPGM severely underperformed in this setting, we re-run them with their pruning rate set to only 50% ($r = 0.5$). Additionally, to understand the influence of the proposed regularization term δ_{opt} on the performance and pruning rate, we conducted ablation experiments with $\lambda = 0$. We computed the Dice coefficient [8] and 95% Hausdorff distance (HD95) [37] on Rats and ACDC test sets (see Tables 1 and 2). In KiTS dataset, only the average Dice coefficient was provided by the online platform that evaluated the test set (see Table 3). In addition to Dice and HD95, we computed the relative decrease in the number of floating point operations (FLOPs) in all convolutions: $FLOPs = HW(C_{in}C_{out})K^2$, where H, W is the height and width of the feature maps, C_{in}, C_{out} is the number of input and output channels, and K is the kernel size. For the 3D CNNs (KiTS dataset), an extra D (depth) and K are multiplied to compute the FLOPs.

Sauron obtained the highest Dice coefficients and competitive HD95s across all datasets and segmentation classes (Tables 1 to 3). Sauron also achieved the highest reduction in FLOPs, although, every

[1] https://github.com/DingXiaoH/Centripetal-SGD
[2] https://github.com/he-y/filter-pruning-geometric-median
[3] https://github.com/Roll920/AutoPruner

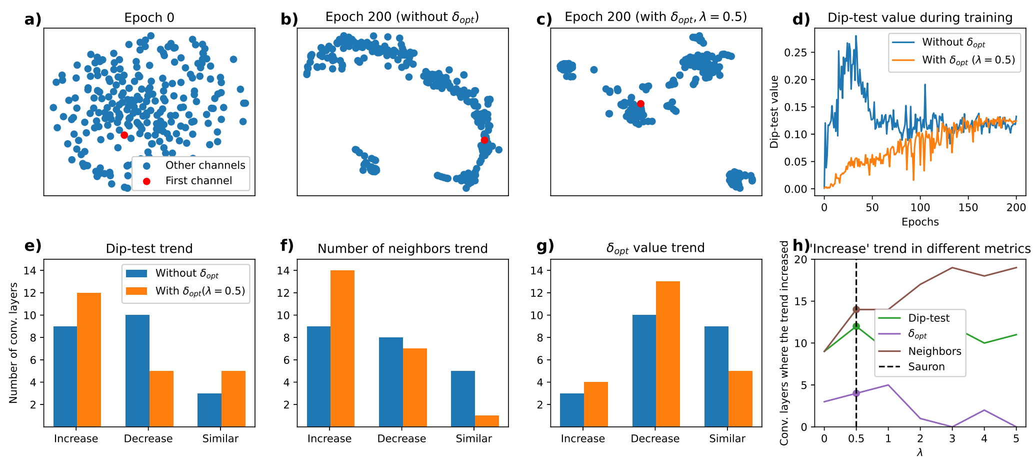

Figure 1: a-c) tSNE plot of "*dec_block_1*" feature maps at initialization (epoch 0), and after optimizing with and without δ_{opt}. d) Corresponding dip-test values during the optimization. e-g) Summary of the trends across the three clusterability measures in all convolutional layers. h) Number of layers with an increasing trend in the three clusterability measures with higher values of λ (dashed line: Sauron's default configuration).

method, including Sauron, can further reduce the FLOPs at the risk of worsening the performance (Table 4). cSGD and FPGM could not yield models with high pruning rates possibly because they aim at reducing only s_{l+1} and not s_l from $\mathbf{W}^l \in \mathbb{R}^{s_{l+1} \times s_l \times k \times k}$. Thus, very high pruning rates cause a great imbalance between the number of input and output filters in every layer that may hinder the training. Note also that cSGD and FPGM were not tested with pruning rates higher than 60% [9, 13]. In contrast, Sauron and Autopruner that achieved working models with higher pruning rate reduced both input filters s_l and output filters s_{l+1}.

Sauron without the proposed regularization term δ_{opt} (Sauron ($\lambda = 0$)) achieved similar or less compressed models and worse Dice coefficients than when minimizing δ_{opt}. Overall, the results from these ablation experiments indicate that 1) typical CNN optimization (without δ_{opt} regularization) yields redundant feature maps that can be pruned with Sauron, 2) pruning rate is generally higher with δ_{opt} regularization, and 3) pruning with no δ_{opt} regularization can affect performance, possibly due to the accidental elimination of non-redundant filters. In summary, the pruning rate and performance achieved in our ablation experiments demonstrate that promoting clusterability via δ_{opt} regularization is advantageous for eliminating redundant feature maps.

4.2 Minimizing δ_{opt} promotes the formation of feature maps clusters

We investigated feature map clustering tendency during nnUNet's optimization. For this, we deactivated Sauron's pruning step and optimized $\mathcal{L}$ on Rats dataset with and without δ_{opt} while storing the feature maps at each epoch (including at epoch 0, before the optimization) of every convolutional layer. Since quantifying clusterability is a hard task, we utilized three different measures: 1) We employed **dip-test** [19], as Adolfsson et al. [2] demonstrated its robustness compared to other methods for quantifying clusterability. High dip-test values signal higher clusterability. 2) We computed the average **number of neighbors** of each feature map layer-wise. Specifically, we counted the feature maps within r, where r corresponded to the 20% of the distance between the first channel and the farthest channel. Distance r is computed every time since the initial distance between feature maps is typically reduced while training. An increase in the average number of neighbors indicates that feature maps have become more clustered. 3) We calculated the **average distance** to the first feature map channel (i.e., δ_{opt}) for each layer, which illustrates the total reduction of those distances achieved during and after the optimization.

In agreement with the literature [13, 43], Figure 1 shows that optimizing nnUNet (without δ_{opt} regularization) yields clusters of feature maps. Feature maps in layer "*dec_block_1*" (see Appendix B) show no apparent structure suitable for clustering at initialization (Fig. 1, a), and, at the end of the optimization, feature maps appear more clustered (Fig. 1, b). Figure 1 (d, blue line) also illustrates this phenomenon: dip-test value is low in the beginning and higher at the end of the training. However, this increasing trend did not occur in all layers. To illustrate this, we compared, for each layer, the average dip-test value, number of neighbors, and distance δ_{opt} in the first and last third of the training. Then, we considered the trend similar if the difference between these values was smaller than 0.001 (for the dip-test values) or smaller than 5% of the average value in the first third (for the number of neighbors and distance δ_{opt}). Figure 1 (e) shows that the number of layers in which the dip-test value increased and decreased were similar when not minimizing the δ_{opt} regularization term. In contrast, the number of layers with an increasing trend was proportionally larger with δ_{opt} regularization. Figure 1 (f) shows a similar outcome regarding the average number of neighbors, i.e., δ_{opt} regularization led to proportionally more neighbors near each feature map. In the same line, the average distance between the first feature map and the rest decreased more with δ_{opt} regularization (Fig. 1, (f)). Additionally, Figure 1 (c) also illustrates that incorporating the δ_{opt} regularization term enhances the clustering of feature maps, as there are more clusters and the feature maps are more clustered than when not minimizing δ_{opt} (Fig. 1 (b)).

We observed higher clusterability in the convolutional layers with more feature maps (see Appendix D). This is likely because such convolutional layers contribute more to the value of δ_{opt} (Eq. 2). On the other hand, convolutional layers with fewer feature maps have larger feature vectors (e.g., *enc_block_1* feature vectors are $(256 \times 256) \times 32$ in Rats dataset) whose distances tend to be larger due to the curse of dimensionality. Sauron accounts, to some extent, for these differences in the convolutional layers with the adaptively-increasing layer-specific thresholds $\boldsymbol{\tau}$. Another possible way to tackle these differences is by using different layer-specific λ's to increase the contribution of the distances of certain layers. We investigated the impact on feature map clusterability with higher λ values and, as illustrated in Figure 1 (h), a higher λ tended to increase the average number of neighbors, decrease δ_{opt}, and somewhat increase the dip-test values, which, overall, signals higher clusterability.

4.3 Feature maps interpretation

Sauron produces small and efficient models that can be easier to interpret. This is due to δ_{opt} regularization that, as we showed in Section 4.2, increases feature maps clusterability. Each feature maps cluster can be thought of as a semantic operation and the cluster's feature maps as noisy outputs of such operation. To test this view, we inspected the feature maps from the second-to-last convolutional block (*dec_block_8*, see Appendix B) of a Sauron-pruned nnUNet. For comparison, we included the feature maps from the same convolutional layer of the baseline (unpruned) nnUNet in Appendix E.

The first feature map depicted in Figure 2 (top) captured the background and part of the rat head that does not contain brain tissue. The second feature map contained the rest of the rat head without brain lesion, and the third feature map mostly extracted the brain lesion. Although the third feature map seems to suffice for segmenting the brain lesion, the first feature map might have helped the model by discarding the region with no brain tissue at all. Similarly, the first and second feature maps in Figure 2 (middle) detected the background, whereas feature maps 3, 4, and 5 extracted, with different intensities, the right cavity (red), myocardium (green), and left cavity (blue) of the heart. In Figure 2 (bottom), we can also see that each feature map captured the background, kidney (red), and tumor (blue) with different intensities. This high-level interpretation facilitates understanding the role of the last convolutional block which, in the illustrated cases, could be replaced by simple binary operations. This shows the interpretability potential of feature map redundancy elimination methods such as Sauron.

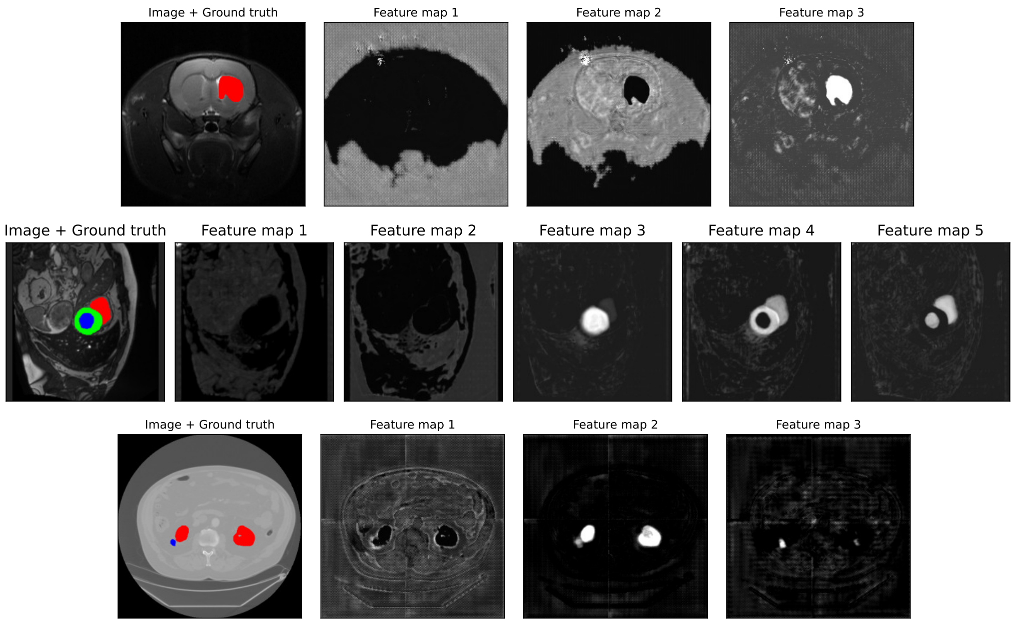

Figure 2: Image slice from Rats (top), ACDC (middle), and KiTS (bottom) datasets, its ground-truth segmentation, and all feature maps at the second-to-last convolutional block after pruning with Sauron.

5 Conclusion

We presented our single-phase filter pruning method named Sauron, and we evaluated it on three medical image segmentation tasks in which Sauron yielded pruned models that were superior to the compared methods in terms of performance and pruning rate. In agreement with the literature, our experiments indicated that CNN optimization leads to redundant feature maps that can be clustered. Additionally, we introduced Sauron's δ_{opt} regularization that, as we showed with three different clusterability metrics, increased feature maps clusterability without pre-selecting the number of clusters, unlike previous approaches. In other words, we enhanced CNN's innate capability to yield feature maps clusters via δ_{opt} regularization, and we exploited it for filter pruning. Finally, we showed that the few feature maps after pruning nnUNet with Sauron were highly interpretable.

Limitations and potential negative impact Sauron relies on feature maps for identifying which filters to prune. Thus, although Sauron is suitable for training models from scratch and fine-tuning pre-trained networks, Sauron is unable to prune CNNs without access to training data, unlike [23, 43, 24]. Furthermore, Sauron cannot enforce a specific compression rate due to its simple distance thresholding. Although we have evaluated Sauron with respect to the segmentation quality, we are not able to evaluate the potential clinical impact. It could be that even a small difference in segmentation would have large clinical impact, or vice versa, a large difference in segmentation could be clinically meaningless. Depending on the application these impacts could be either positive or negative.

References

[1] Reza Abbasi-Asl and Bin Yu. Structural compression of convolutional neural networks. *arXiv preprint arXiv:1705.07356*, 2017.

[2] Andreas Adolfsson, Margareta Ackerman, and Naomi C Brownstein. To cluster, or not to cluster: An analysis of clusterability methods. *Pattern Recognition*, 88:13–26, 2019.

[3] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13 (3):1–18, 2017.

[4] SH Basha, Mohammad Farazuddin, Viswanath Pulabaigari, Shiv Ram Dubey, and Snehasis Mukherjee. Deep model compression based on the training history. *arXiv preprint arXiv:2102.00160*, 2021.

[5] Olivier Bernard, Alain Lalande, Clement Zotti, Frederick Cervenansky, Xin Yang, Pheng-Ann Heng, Irem Cetin, Karim Lekadir, Oscar Camara, Miguel Angel Gonzalez Ballester, et al. Deep learning techniques for automatic mri cardiac multi-structures segmentation and diagnosis: Is the problem solved? *IEEE transactions on medical imaging*, 37(11):2514–2525, 2018.

[6] Jingfei Chang, Yang Lu, Ping Xue, Xing Wei, and Zhen Wei. Ucp: Uniform channel pruning for deep convolutional neural networks compression and acceleration. *arXiv preprint arXiv:2010.01251*, 2020.

[7] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[8] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3): 297–302, 1945.

[9] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4943–4953, 2019.

[10] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[11] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

[12] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.

[13] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019.

[14] Nicholas Heller, Niranjan Sathianathen, Arveen Kalapara, Edward Walczak, Keenan Moore, Heather Kaluzniak, Joel Rosenberg, Paul Blake, Zachary Rengel, Makinna Oestreich, et al. The kits19 challenge data: 300 kidney tumor cases with clinical context, ct semantic segmentations, and surgical outcomes. *arXiv preprint arXiv:1904.00445*, 2019.

[15] Saihui Hou and Zilei Wang. Weighted channel dropout for regularization of deep convolutional neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8425–8432, 2019.

[16] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018.

[17] Fabian Isensee, Paul F Jaeger, Peter M Full, Ivo Wolf, Sandy Engelhardt, and Klaus H Maier-Hein. Automatic cardiac disease assessment on cine-mri via time-series segmentation and domain specific features. In *International workshop on statistical atlases and computational models of the heart*, pages 120–129. Springer, 2017.

[18] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.

[19] Argyris Kalogeratos and Aristidis Likas. Dip-means: an incremental clustering method for estimating the number of clusters. *Advances in neural information processing systems*, 25: 2393–2401, 2012.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[22] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. PMLR, 2015.

[23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[24] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Exploiting kernel sparsity and entropy for interpretable cnn compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2019.

[25] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1529–1538, 2020.

[26] Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *IJCAI*, volume 2, page 8, 2018.

[27] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3296–3305, 2019.

[28] Jian-Hao Luo and Jianxin Wu. An entropy-based pruning method for cnn compression. *arXiv preprint arXiv:1706.05791*, 2017.

[29] Jian-Hao Luo and Jianxin Wu. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, 107:107461, 2020.

[30] Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. Thinet: pruning cnn filters for a thinner net. *IEEE transactions on pattern analysis and machine intelligence*, 41(10):2525–2538, 2018.

[31] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016.

[32] Suraj Mishra, Peixian Liang, Adam Czajka, Danny Z Chen, and X Sharon Hu. Cc-net: Image complexity guided network compression for biomedical image segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 57–60. IEEE, 2019.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[34] Fernando Pérez-García, Rachel Sparks, and Sébastien Ourselin. Torchio: a python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. *Computer Methods and Programs in Biomedicine*, page 106236, 2021. ISSN 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2021.106236. URL `https://www.sciencedirect.com/science/article/pii/S0169260721003102`.

[35] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[37] Günter Rote. Computing the minimum hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38(3):123–127, 1991.

[38] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P Namboodiri. Play and prune: Adaptive filter pruning for deep model compression. *arXiv preprint arXiv:1905.04446*, 2019.

[39] Xavier Suau, Nicholas Apostoloff, et al. Filter distillation for network compression. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3129–3138. IEEE, 2020.

[40] Frederick Tung and Greg Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7873–7882, 2018.

[41] Juan Miguel Valverde, Artem Shatillo, Riccardo De Feo, Olli Gröhn, Alejandra Sierra, and Jussi Tohka. Ratlesnetv2: a fully convolutional network for rodent brain lesion segmentation. *Frontiers in neuroscience*, 14:1333, 2020.

[42] Dong Wang, Lei Zhou, Xueni Zhang, Xiao Bai, and Jun Zhou. Exploring linear relationship in feature map subspace for convnets compression. *arXiv preprint arXiv:1803.05729*, 2018.

[43] Wenxiao Wang, Cong Fu, Jishun Guo, Deng Cai, and Xiaofei He. Cop: Customized deep model compression via regularized correlation-based filter-level pruning. *arXiv preprint arXiv:1906.10337*, 2019.

[44] Zihao Xie, Li Zhu, Lin Zhao, Bo Tao, Liman Liu, and Wenbing Tao. Localization-aware channel pruning for object detection. *Neurocomputing*, 403:400–408, 2020.

[45] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv preprint arXiv:1802.00124*, 2018.

[46] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1909.08174*, 2019.

[47] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[48] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2019.

Checklist

1. For all authors...
    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Particularly 1) Sauron being a single-phase method (Section 3.4), 2) its outperformance over other methods (Section 4.1), 3) the proposed δ_{opt} increasing feature maps clusterability (Section 4.2), and 4) Feature maps easier to interpret (Section 4.3)
    (b) Did you describe the limitations of your work? [Yes] See Section 5
    (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5
    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
    (a) Did you state the full set of assumptions of all theoretical results? [N/A]
    (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The supplementary material and `https://github.com/blindedrepository` contain the code to run all our experiments. Additionally, it includes a README file specifying how to run each experiment.
    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Most important details were described in Section 4, and we specified all data augmentation, architectural and optimization settings in Appendix C. These settings can also be seen in our code.
    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
    (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Section 3.5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
    (a) If your work uses existing assets, did you cite the creators? [Yes] Data: see Section 4, "Datasets" paragraph. Code: see Section 4.1 for the compared methods
    (b) Did you mention the license of the assets? [Yes] We mentioned the license for ACDC and KiTS datasets (Section 4, "Datasets").
    (c) Did you include any new assets either in the supplemental material or as a URL? [No]
    (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
    (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
    (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
    (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]