Continuous Pruning of Deep Convolutional Networks Using Selective Weight Decay

Hugo Tessier  
Groupe PSA

Vincent Gripon  
IMT Atlantique

Mathieu Léonardon  
IMT Atlantique

Matthieu Arzel  
IMT Atlantique

Thomas Hannagan  
Groupe PSA

David Bertrand  
Groupe PSA

Abstract

During the last decade, deep convolutional networks have become the reference for many machine learning tasks, especially in computer vision. However, large computational needs make them hard to deploy on resource-constrained hardware. Pruning has emerged as a standard way to compress such large networks. Yet, the severe perturbation caused by most pruning approaches is thought to hinder their efficacy. Drawing inspiration from Lagrangian Smoothing, we introduce a new technique, Selective Weight Decay (SWD), which achieves continuous pruning throughout training. Our approach deviates significantly from most methods of the literature as it relies on a principle that can be applied in many different ways, for any problem, network or pruning structure. We show that SWD compares favorably to other approaches in terms of performance/parameters ratio on the CIFAR-10 and ImageNet ILSVRC2012 datasets. On CIFAR-10 and unstructured pruning, with a parameters target of 0.1%, SWD attains a Top-1 accuracy of 81.32% while the reference method only reaches 27.78%. On CIFAR-10 and structured pruning, with a parameters target of 2.5%, the reference technique drops at 10% (random guess) while SWD maintains the Top-1 accuracy at 93.22%. On the ImageNet ILSVRC2012 dataset with unstructured pruning, for a parameters target of 2.5%, SWD attains 84.6% Top-5 accuracy instead of the 77.07% reached by the reference.

1. Introduction

Long after the introduction of convolution layers by LeCun et al. [36], deep convolutional networks became the state of the art for image recognition during the last decade [31]. Since then, many architectural innovations have improved their performance and efficiency [22, 28, 38, 52, 55, 58]. However, for a same type of architecture, the number of parameters and performance tend to be correlated, resulting in the best performing networks having prohibitive requirements, in term of memory footprint, computation power and energy consumption [5].

This is a crucial issue for multiple reasons. Indeed, many applications, such as autonomous vehicles, require networks that can provide real-time adequate responses on energy-efficient hardware: for such tasks, one cannot afford to have either an accurate network that is too slow to run or one that performs quickly but crudely. Also, research on deep learning relies heavily on running experiments that require a lot of computation time and power: lightening networks would help speeding up the whole process.

Many approaches have been proposed to tackle this issue. These include techniques such as distillation [25, 33], quantization [8, 49], factorization [10] or pruning [18]; most of them can be combined [17]. The whole field tends to indicate that there may exist a Pareto optimum, between performance, memory occupation and computation power, that compression could help to attain. However, progress in the field shows that this optimum is yet to be reached.

Our work focuses on pruning. The basis of most pruning methods is to train a network and, according to a certain criterion, to identify which parts of it contribute the least to its performance [18, 39]. This process can be iterative.

Multiple problems prevent this method from being optimal. Indeed, the philosophy of deep learning is to avoid as much as possible manual interventions on the learning process: every handcrafted aspect is likely to be greatly suboptimal [34]. Hence, this discontinuous way of removing parts of the network could hardly be the best possible one. Also, pruning networks at discrete, defined steps only operates on particular states of its parameters and can hardly take into account the intricate dependencies between them.

Our contribution aims at tackling these very issues. Our method, Selective Weight Decay (SWD), is a continuous pruning method for deep convolutional networks, which is
inspired by Lagrangian Smoothing. It consists in a continuous regularization which, at each moment during the training process, penalizes the weights that, at a given moment, would be pruned according to a given criterion. The penalization grows throughout the training until the magnitude of the targeted parameters is so close to 0 that pruning them induces no drop in performance. This method presents many desired properties. Indeed, it spreads the principle of pruning all throughout the training process, which prevents any discontinuity. The removal is itself learned, which reduces the manual aspects of the process. Also, since the penalized weights are not completely removed before the very end of training, the subset of the target parameters can freely change during training, depending on the little weights that grows and those big that shrink. This allows to take better into account the dependencies between weights.

Our experiments show that SWD works well for both lightweight and large-scale datasets and networks, with various pruning structures. Our method especially shines for aggressive pruning rates (low parameters target) and manages to achieve great results at targets for which classic methods reduce the network to mere random guess.

We have the following claims:

- we introduce SWD, a method which deep convolutional networks continuously during training;
- using standardized benchmark datasets, we proved that SWD performs greatly better, on aggressive pruning targets, than standard methods;
- SWD needs fewer hyperparameters, introduces no discontinuity, needs no fine-tuning and can be applied to any pruning structure with any pruning criterion.

In the next sections, we will present in details the field of network pruning, describe our method, show our experiments and discuss our results.

2. Problem Statement and Related Work

We now review the main pruning methods, and attempt to categorize them in sub-families.

2.1. Notations

We first recall the standard optimization problem with weight-decay. Let \( N \) be a network with parameters \( w \), trained over the dataset \( D \) containing \( N \) pairs of input/groundtruth pairs \((x_i, y_i)\). The network is trained through the error function \( \mathcal{E} \), penalized by a weight-decay with a coefficient \( \mu \) \cite{32,48}. The training process then consists in minimizing the objective function \( \mathcal{L} \) defined as:

\[
\mathcal{L}(w) = \sum_{(x,y) \in D} \mathcal{E}(N(x, w), y) + \mu \|w\|_2
\]  

**Err**: error term \hspace{10cm} **WD**: weight-decay
2.2. The birth and rebirth of pruning

The idea behind pruning is that although it seems that the larger the network, the better its performance, not all of their parts seem to be useful once trained. Such parts may be removed without penalizing the performance.

At the end of the '80s and at the beginning of the '90s, the field of pruning already sprouted from a few seminal studies [20, 35, 45]. At the time, as observed by Reed [50], two major branches cohabited: 1) sensitivity calculation methods, which consisted in evaluating the contribution of each parameter to the error function and in pruning those which contributed the least, and 2) penalty-term methods, which penalized weights globally so as to encourage convergence to networks having a few big weights rather than lots of small ones. It is worth mentioning that pruning was originally intended to help the generalization of networks instead of being a compression method per se.

This field of research seems to have almost completely vanished during the ensuing decade, only to be resurrected by Han et al. [18] in 2015. From this work on, the number of pruning investigations has expanded so quickly as to make any reviewing task a challenging one [3, 9, 23] to either regrow previously pruned weights or not to completely prune parameters by still allowing them to be trained once reduced to 0 by pruning.

As the method of Han et al. [18] is nowadays the prototypical pruning technique, it is worth detailing it. Assume we want to train and prune a given model with a target pruning rate $T$. The method of Han et al. consists in first training, and then, pruning and fine-tuning the network iteratively, with each time an increasing pruning rate $t$ until $T$ is reached. In particular, pruning is achieved here by reducing to 0 a proportion of the parameters of the whole network whose absolute magnitude are the smallest. Algorithm 1 sums up the method of Han et al. [18].

**Algorithm 1:** Summary of method proposed by Han et al. [18]

**Data:** the network $\mathcal{N}$ of weights $\mathbf{w}$, the dataset $\mathcal{D}$, the target pruning rate $T$

$t \leftarrow 0$;

while the network has not fully converged and $t \leq T$
do

| $||\mathbf{w}||_0 = (1 - t) \times n(\mathbf{w})$ parameters are pruned in total; |
| train $\mathcal{N}$ with $\mathcal{D}$ according to Equation 1 until convergence; |
| increase $t$; |

end

Though it is still possible to prune and fine-tune the model only once, doing so can be viewed as a particular case of Algorithm 1 with only one iteration.

After the work of Han et al. [18], many research efforts focused on distinct aspects such as the discontinuity of pruning, the pruning criterion or the pruning granularity. Each of these topics will be tackled in the following sections.

2.3. On the discontinuity of pruning

One may object that removing weights, even those that seem least important, may damage the network in such a way that no fine-tuning could ever make it recover. Indeed, doing so introduces a severe discontinuity, during the training process, by removing parts of the network while the latter is trying to learn to solve a problem. The work of Le Cun, Bengio and Hinton [34] tends to show that the less the training process is disrupted, the better it performs.

For example, there is no guarantee that weights which seemed unimportant at first could not become crucial again in the new context of the shrunk network. That is the reason many efforts focused onto allowing weights to regrow in a way or another. Different approaches were proposed [3, 9, 23] to either regrow previously pruned weights or not to completely prune parameters by still allowing them to be trained once reduced to 0 by pruning.

The principle of regrowing weights is central to the family of methods that could be called sparse training. First introduced by Mocanu et al. [42] and then further explored within the literature [11, 13, 44], sparse training consists in training the network with a constant level of sparsity, at first spread randomly with uniform probability and then adjusted during steps which combine 1) pruning of a certain portion of the weights, according to a certain criterion, and 2) regrowing an equivalent amount of weights, depending on another criterion. In such a framework, the pruning algorithm can be described as Algorithm 2:

**Algorithm 2:** Summary of sparse training

**Data:** the network $\mathcal{N}$ of weights $\mathbf{w}$, the dataset $\mathcal{D}$, a pruning ratio $p$, a sparsity level $T$

initialize $\mathcal{N}$ with a randomly spread sparsity $T$;

while the network is not fully trained do

| train $\mathcal{N}$ with $\mathcal{D}$ according to Equation 1 until convergence; |
| prune $p \times n(\mathbf{w})$ parameters according to the pruning criterion; |
| regrow $p \times n(\mathbf{w})$ parameters according to the regrowing criterion; |

end

Such a family of methods seems to have shown a promising way to get around the problem of falsely unimportant weights, while allowing to limit the impact of an increasing sparsity all throughout the training process.

Nonetheless, all these methods remain deeply discontinuous as the conditions under which the network is trained...
deeply change each time its connectivity is being altered. Thus, a sensible way not to disrupt training too much is to delegate the care of sparsity to the gradient descent.

The field of learnable sparsity counts many contributions. The most widespread idea is to find a way to learn a pruning mask during training [15, 40, 53, 57]; some of them propose to learn such a mask using variants of the quantification method of Courbariaux et al. [8]. Carreira-Perpín et al. [6] study the question of pruning in a more theoretical way and formalize the principle of considering connectivity as learnable annex parameters.

However, Gale et al. [14] compared multiple advanced pruning methods, including the work of Louizos et al. [40], to the one of Han et al. [18] and surprisingly found out that the latter got very comparable or even better results, especially on large datasets such as ImageNet ILSVRC2012 (Gale et al. did not manage to make the method of Louizos et al. work properly on it without damaging critically the performance of the trained networks). Also, Blalock et al. [4] raised the alarm about the lack of comparability between the various methods of the literature. Hence, we have no solid proof that the aforementioned methods do work better than the basic one from Algorithm 1.

It then appears that the question of continuity in pruning cannot be considered as solved yet.

2.4. The choice of a pruning criterion

Many pruning criteria have been tested [2, 26, 41, 54, 60], for example: Anwar et al. [2] try various random masks and select that which induces the least degradation; Hu et al. [26] prune on the basis of the average rate of null activation after each pruned layer. However, each of these criteria seems not to be found among the rest of the literature. Indeed, the two most widespread criteria are gradient magnitude and weight magnitude, which we will both detail.

The early branch of sensitivity calculation methods, birthed by the studies of Le Cun et al. [35] and Mozer and Smolensky [45], then studied within multiple articles [20, 21, 30, 56], led some recent studies to prune the weights of least back-propagated gradient [12, 43].

Nevertheless, the criterion that remains the most common, which is the mere magnitude of the parameters, turns out to be surprisingly effective while being intuitive. Although re-introduced by Han et al. [18], it was first introduced by Chauvin [7] and Hanson and Pratt [19], then presented under the name of clipping by Janowsky [29]. Segeer and Carter [51] observed the surprising correlation between this intuitive criterion and that of Mozer and Smolensky [45], which was yet much more theoretically grounded. Even though one might question the relevance of such old results nowadays, these studies tend to confirm that magnitude is a good proxy for the contribution of a parameter to optimization problem summed up by Equation 1.

The other main branch identified by Reed [50] revolved around enforcing sparsity using various kinds of weight-decay regularization. The commonly admitted motivation was that, if a certain parameter contributes poorly to the error term \( \text{Err} \), then the weight-decay term should outweigh it so that this very parameter would decrease toward 0.

Since weight decay is required for weight-magnitude pruning, which is the favored criterion among the best implementations, it seems that sparsity-inducing regularizations are worth exploring further.

2.5. Pruning structures

As pointed out by Anwar et al. [1] or Li et al. [37], the parameter-wise pruning of Han et al. [18] produces sparse matrices that are hardly exploitable by modern hardware and deep learning frameworks. That is why a whole field of pruning is dedicated to finding ways to eliminate parts of the networks in a structured way that can actually induce a measurable speedup. The most widespread type of structure to be considered by the field is convolutional channels (or filters) [1, 2, 23, 24, 27, 37, 39, 41, 59]. Indeed, pruning filters induces a direct shrinking of the very architecture of the network and a quadratic reduction in the parameter count as each removed filter is one less input feature map for the next convolution layers.

Other types of structures have been experimented on, such as kernels or intra-kernel strided structures [1, 2] or the reduction of convolutions to shift operations [16]. In this work we focus onto two granularity levels: parameter-wise (unstructured) and filter-wise (structured).

3. Selective Weight Decay

We now present our contribution, illustrated in Figure 1, and explain how it addresses the aforementioned issues.

3.1. Principle

Selective Weight Decay (SWD) is a regularization which induces sparsity continuously on any type of structure: at each training step, a certain penalization is applied to the parameters which, at this very step according to a certain criterion and a certain structure, would be pruned. The criterion we chose is the weight magnitude, or variants of it according to the chosen structure. The penalized optimization problem can be viewed as:

\[
\mathcal{L}(\mathbf{w}) = \sum_{(x,y) \in \mathcal{D}} \mathcal{E}(\mathcal{N}(x,\mathbf{w}), y) + \mu \|\mathbf{w}\|_2^2 + a\mu \|\mathbf{w}^*\|_2^2
\]

with \( a \) being a coefficient which determines the importance of SWD relatively to the rest of the optimization problem. \( \mathbf{w}^* \) is the subset of \( \mathbf{w} \) which, at a certain step, would be pruned. SWD is summed up as the Algorithm 3.
Allowing the network to converge before applying a strong penalization, we favored an exponential increase over a linear one so that the evolution of \( a \) is designed to be exponential and, according to two bounds \( a_{\min} \) and \( a_{\max} \) at a certain training step \( s \), is defined as such:

\[
a(s) = a_{\min} \cdot e^{s/s_{\text{final}} \cdot \ln(\frac{a_{\max}}{a_{\min}})}
\]  

(3)

with \( s_{\text{final}} \) the total number of training steps. The exponential increase in Equation 3 allows the network to converge before applying a strong penalization. We favored an exponential increase over a linear one so that \( a \) can reach large final values without penalizing the training too much all throughout the training process.

### 3.2. SWD as a Lagrangian smoothing of pruning

Since penalizing weights until they reach 0 appears to be a viable method to relax the hard constraint that is pruning, we designed SWD so that it can be viewed as a Lagrangian smoothing with the coefficient \( a \) of the SWD term of Equation 2 being a Lagrangian multiplier.

As pointed out by the work of Walter and Kien-Ming [46], Lagrangian smoothing allows convergence relatively to both the error term and the constraint. If \( a \) can be mostly negligible at the start of the training, it becomes preponderant at the end and forces the target weights to be pruned, so as not to hinder the convergence of the network during training while allowing for pruning.

The fact that SWD only penalizes weights selectively during training allows them to recover as soon as the pruning criterion does not target them anymore, which combines both the pruning and regrowing criteria of sparse training. Therefore, SWD is a non-greedy method that allows weights to recover when needed.

### 3.3. On the adaptability of SWD to structures

The exact definition of \( w^* \) depends on the chosen type of structure to prune. As SWD induces no constraint on such consideration, it can be applied to any type of structures.

Unstructured pruning is defined by removing all the weights of least magnitude in the whole network so that the proportion of pruned parameters matches as closely as possible the pruning target. Formally stated:

\[
w^* = \{ |w| \leq \delta, w \in W \},
\]

with \( \delta \) so that \( n(w^*) = T n(w) \)

(4)

with \( T \) being the pruning target and \( n(w) \) the number of elements of the parameters \( w \).

We based the structured version of our SWD on the method of Liu et al. [39] to solve a problem induced by residual connections in modern convolutional networks: to ensure that the exact output dimensions of the feature maps after each residual connection match the desired target, one must prune exactly the same channels among the connection and the last convolution before them. To the best of our knowledge, this problem has not been tackled, and approaches that use some norms of the filters as a criterion could not be adjusted to tackle this important problem without altering them drastically.

But since the method of Liu et al. [39] prunes multiplicative coefficients of batchnorm layers, it is easy for it to solve the residual connection issue as soon as a batchnorm layer is inserted after each residual connection (which does not change the overall performance of the network).

Liu et al. [39] considered the magnitudes of multiplicative coefficients in a batchnorm layer as an estimator of the importance of their corresponding filters. Those batchnorm layers were then penalized during training by a smooth-\( \ell_1 \) norm. In their work, a global threshold was applied on all the batchnorm layers in order to prune globally a target percentage of all the smallest batchnorm coefficients.

However, in order to have a better control over the exact number of parameters at the end of the pruning process, we instead prune all the smallest batchnorm layers until a portion of the overall network (once subtracted the parameters of the corresponding convolutional filters) is removed. Therefore, we get:

\[
\mathbf{b} \in W, \mathbf{b}^* = \{ \mathbf{b}_i^*, \mathbf{b}_i \in \mathbf{b} \},
\]

\[
\mathbf{b}_i^* = \{ |w| \leq \delta, w \in \mathbf{b}_i \},
\]

(5)

with \( \delta \) so that \( n(w(b^*)) = T n(w) \) the parameters count obtained when the filters that correspond to the pruned batchnorm coefficient are removed.

### 4. Experiments

Our code is available at https://github.com/HugoTessier-lab/SWD.
Experiments on ImageNet ILSVRC2012

| Model       | Pruning type | Method        | Parameters | Accuracy |
|-------------|--------------|---------------|------------|----------|
|             |              |               | % (target) | Top-1    | Top-5    |
| ResNet-50   | -            | Baseline      | 100        | 72.55    | 91.0     |
| Unstructured| Han 2015 [18]| SWD           | 2.5        | 52.37    | 77.07    |
|             | Liu 2017 [39]| SWD           | 20         | 32.0     | 58.1     |

Table 1: Results with ResNet-50 on ImageNet ILSVRC2012, with unstructured and structured pruning. SWD outperform the reference method in both cases.

ResNet-20 on CIFAR-10

| Params target (%) | Unstructured pruning | Structured pruning |
|-------------------|----------------------|--------------------|
|                   | Han et al. [18] (%)  | SWD (%)            | Liu et al. [39] (%)| Operations (%) | SWD (%)| Operations (%) |
| 90                | 95.45                | 94.83              | 84.13              | 95.42          | 86.47  |
| 80                | 95.47                | 94.88              | 70.41              | 95.25          | 80.76  |
| 70                | 95.43                | 94.88              | 58.20              | 95.56          | 70.77  |
| 60                | 95.48                | 94.91              | 48.06              | 95.18          | 58.29  |
| 50                | 95.44                | 94.92              | 40.25              | 95.13          | 48.50  |
| 40                | 95.32                | 94.29              | 33.88              | 95.27          | 39.52  |
| 30                | 95.3                | 93.24              | 26.01              | 94.96          | 31.59  |
| 25                | 95.32               | 92.08              | 21.36              | 94.77          | 26.29  |
| 20                | 95.05               | 94.99              | 12.07              | 94.65          | 18.52  |
| 15                | 94.77               | 94.90              | 8.04               | 93.93          | 13.25  |
| 10                | 94.48               | 94.58              | 5.87               | 93.22          | 9.89   |
| 7.5               | 94.03               | 94.4               | 4.01               | 71.29          | 8.09   |
| 5                 | 93.38               | 94.14              | 3.39               | 46.65          | 7.36   |
| 4                 | 93.95               | 93.76              | 2.84               | 26.05          | 5.34   |
| 3                 | 91.43               | 93.52              | 2.26               | 20.90          | 5.41   |
| 2.5               | 90.58               | 93.49              | 1.80               | 32.76          | 4.40   |
| 2                 | 87.44               | 93.0               | 1.37               | 10.86          | 3.61   |
| 1.5               | 83.42               | 92.5               | 0.99               | 10.0           | 2.28   |
| 1                 | 66.9                | 91.05              | 0.57               | 10.0           | 1.58   |
| 0.5               | 48.52               | 86.81              | 0.26               | 10.0           | 0.35   |
| 0.2               | 27.78               | 81.32              | 0.12               | 10.0           | 0.26   |

Table 2: Top-1 accuracy of ResNet-20 on CIFAR-10 for various parameters target, with different methods. For structured pruning, the corresponding estimated number of operations is given. SWD has a better performance/parameters tradeoff, especially on very high pruning targets (i.e. very low parameters targets).

4.1. General training conditions

In order to eliminate all unwanted variables, each network was trained from scratch under the same conditions, except when explicitly stated, with the same initialization and same seed for the random number generators of the various used libraries. We used no pre-trained network and we trained ours in a very standard way. Therefore, even though our baseline models may not match the absolute state of the art, the performance they reach are satisfying enough to draw conclusions from the experiments.

The standard conditions are as follows: all our networks were trained using the Pytorch framework (Paske et al. [47]); the optimizer we used was SGD, with a base learning rate of 1e−1 for the first third of the training,
then $1e^{-2}$ for the second and finally $1e^{-3}$ for the last one; the momentum of the SGD is set to 0.9; the standard weight-decay (parameter $\mu$) is set to $5e^{-4}$ for CIFAR-10 and $1e^{-4}$ for ImageNet ILSVRC2012. The models on ImageNet ILSVRC2012 were trained during 30 epochs; those on CIFAR-10 during 300 epochs.

### 4.2. Specificities of each method

**Han et al. [18]**. The networks trained with this method were pruned and fine-tuned for 5 iterations. At each step, the pruning target is a fraction of the final one: for example, when first pruned, a fifth of the final pruning target is actually removed. Each fine-tuning lasts for 15 epochs on CIFAR-10 and 5 epochs on ImageNet ILSVRC2012 (except for the last iteration which lasts 15 epochs).

**Liu et al. [39]**. The network is only pruned once and fine-tuned over 50 epochs on CIFAR-10 or CIFAR-100 or 45 epochs on ImageNet ILSVRC2012. In accordance to the paper, the smooth-$L_1$ norm applied to every prunable batch-norm layer has a coefficient that is $\lambda = 10^{-4}$ on CIFAR-10.

**SWD**. Whether on unstructured or structured pruning, when trained with SWD the network is not fine-tuned at all and only pruned once. The bounds of $a$ were found by trial and error and are defined as $a_{\text{min}} = 10^{-1}$ and $a_{\text{max}} = 10^{5}$.

### 4.3. Experiments on ImageNet ILSVRC2012

The results of the experiments on ImageNet ILSVRC2012 are shown in Table 1, which presents the top-1 and top-5 accuracy of ResNet-50, from He et al. [22] on ImageNet ILSVRC2012, under the conditions described in Sections 4.1 and 4.2. The “Baseline” method is a regular, non-pruned network, that serves as a reference.

Since unstructured pruning is hardly exploitable on GPUs and cannot be taken benefit from without a very specific hardware implementation which we did not want to speculate on, we found it best not to count the operations.

### 4.4. Impact of SWD on the pruning/accuracy trade-off

Even though obtaining a better accuracy for a given pruning target is not without interest, it makes more sense to know, for one accuracy target, which is the maximal compression rate that SWD would allow. Figure 2 shows the influence of SWD on the pruning/accuracy trade-off for ResNet-20 on CIFAR-10. We used that lighter dataset, instead of ImageNet ILSVRC2012 because of the expensive cost of computing so many points. The precise results used for Figure 2 are reported in Table 2.

Since each point is the result of only one experiment, there may be some fluctuations due to low statistical power. However, since the same random seed and model initialization were used each time, these may not prevent us to draw some conclusion about the behavior of each method.

As we stated in Section 2.2, pruning originally served as a method to improve generalization. This suggests that the relation between performance and pruning ratio may be subtle enough to lead to some local optima that may not always be possible to predict.

### 4.5. Grid search on MNIST and CIFAR-10

To show the influence of the values of $a_{\text{min}}$ and $a_{\text{max}}$ on the performance of networks right before and right after the final pruning step, we did a grid search using LeNet-5 on MNIST. The models were trained during 10 epochs, with a learning rate of 0.1 and no weight-decay (even though $\mu$ is set to $5e^{-4}$ for SWD). The pruning target we used is 90% (hence, a parameters target of 10%). The momentum is set to 0. The results of the grid search are reported in Table 3.

A similar grid search has been performed with ResNet-20 on CIFAR-10. The results are reported in Table 4.
Top-1 accuracy before/after final removal step (%) of LeNet-5 on MNIST

| $a_{\text{min}}$ | 1e−1 | 1e−2 | 1e−3 | 1e−4 | 1e−5 |
|------------------|------|------|------|------|------|
| 1e1              | (98.96 / 97.29) | (98.89 / 96.59) | (99.03 / 94.15) | (98.97 / 94.17) | (98.9 / 95.42) |
| 1e2              | (97.62 / 89.83) | (98.38 / 96.6) | (98.8 / 96.83) | (97.56 / 87.99) | (98.45 / 95.6) |
| $a_{\text{max}}$|              |      |      |      |      |
| 1e3              | (95.39 / 96.83) | (96.58 / 97.87) | (98.03 / 98.3) | (98.15 / 98.21) | (93.87 / 87.57) |
| 1e4              | (98.31 / 98.52) | (98.52 / 98.56) | (97.69 / 98.04) | (96.86 / 97.72) | (98.01 / 98.13) |

Table 3: Top-1 accuracy of LeNet-5 on MNIST, before and after the final removal step. Various combinations of values for $a_{\text{min}}$ and $a_{\text{max}}$ are compared. The parameters target is 10% and the pruning is unstructured. The baseline reaches an accuracy of 99%. $\mu$ is set to $5e^{-4}$. For a value of $a_{\text{max}}$ above $1e^3$, the removal actually improves the performance of the network.

Top-1 accuracy before/after final removal step (%) of ResNet-20 on CIFAR-10 (unstructured)

| $a_{\text{min}}$ | 1e−1 | 1e−2 | 1e−3 | 1e−4 | 1e−5 |
|------------------|------|------|------|------|------|
| 1e1              | (94.94 / 94.73) | (95.24 / 92.69) | (95.58 / 78.28) | (95.33 / 83.08) | (95.05 / 63.66) |
| 1e2              | (94.96 / 94.88) | (95.01 / 94.33) | (95.01 / 92.29) | (94.85 / 85.33) | (94.91 / 29.18) |
| $a_{\text{max}}$|              |      |      |      |      |
| 1e3              | (95.29 / 95.33) | (95.32 / 95.34) | (95.19 / 93.55) | (95.12 / 92.64) | (95.02 / 86.89) |
| 1e4              | (95.18 / 95.17) | (95.06 / 95.08) | (95.3 / 95.28) | (95.17 / 94.88) | (95.22 / 95.23) |
| 1e5              | (95.19 / 95.19) | (95.18 / 95.18) | (95.2 / 95.21) | (95.09 / 95.11) | (95.4 / 95.43) |

Table 4: Top-1 accuracy of ResNet-20 on CIFAR-10, before and after the final removal step. Various combinations of values for $a_{\text{min}}$ and $a_{\text{max}}$ are compared. The parameters target is 10% and the pruning is unstructured. The baseline is 95.45%.

5. Discussion

The experiments on CIFAR-10 have shown SWD to perform on par with standard methods for low pruning targets and greatly outdo them on high ones. Especially for structured pruning, our method allows for much higher targets on the same accuracy. SWD even manages to attain only slightly reduced performances at pruning targets for which the method of Liu et al. [39] reduces performance to random guess. We think that the multiple desirable properties brought by SWD over standard pruning are responsible for a much more efficient identification and removal of the unnecessary parts of networks. Indeed, dramatic degradations of performance which could come from removing by error a necessary parameter or filter are limited by both the continuity of SWD, which lets the other parameters compensate progressively for the loss, and its ability to adapt his targeted parameters, so that it can penalize at a later time the weights that are more relevant to remove.

Because of the large granularity of filter-wise structured pruning, there is always the risk to prune all filters of a single layer and, then, to break irremediably the network. This is likely the reason why, in Table 2, the method of Liu et al. [39] makes the network drop at an accuracy of 10% at the parameters target 7.5%, while at 10% it still reaches 71.01%. Since SWD can adapt itself not to induce damage of this scale, the network does not reach random guess before a much lower target of 1%.

Our results also allow to confirm that SWD can be applied on different datasets and networks or even pruning structures and yet stay ahead of the reference method. That means that the properties of SWD are not task or network-specific and can be transposed in various contexts, which is an important issue, as shown by Gale et al. [14].

It is interesting to notice that, for almost each pruning target, the estimated operations count of the models pruned using the method of Liu et al. [39] is greater than the one of those pruned using SWD. It could seem counter-productive, it is actually a good sign. Indeed, those networks were pruned according to a parameters compression target and the goal for each was to maximize the performance. Since performance and operations count tend to be correlated, the fact that both are greater, for a same parameters compression target, means that SWD actually aimed preferably at filters that reduced the least the number of operations. So, it appears that SWD effectively targets the parameters that are the most relevant to prune.

The fact that all the results with SWD were obtained without any fine-tuning, and yet performed well, supports our hypothesis about SWD being a way to prune models continuously. Indeed, both removal and recovering are done
during the same process without any manual intervention.

However, to make sure that SWD performs as efficiently as possible, it is important to choose carefully the initial and final values of $a$. Indeed, the initial value of $a$ determines how quickly it will reach large values during training. The final value is crucial to make sure that the pruning step, once the training process is done, does not induce any significant drop in accuracy. The grid search displayed in Table 3 shows various trends, despite some local fluctuations. The difference between accuracy right before and right after pruning decreases along with the increase of $a_{\text{max}}$. For $a_{\text{min}}$, if there is an improvement from $1e-1$ to $1e-2$, other values do not show any particular trend beside not performing as well. One may observe too that, with values of $a_{\text{max}}$ from $1e3$ to above, the pruning step seems to consistently improve the accuracy instead of damaging it. Improvement of accuracy through pruning is a result that can be observed when the very smallest weights of a network are removed. That is something we see in Table 2 for the highest parameters targets with unstructured pruning. This is consistent with pruning being originally a generalization method. This improvement shows that the contribution of parameters targeted by SWD is truly nullified at the end of the training, which confirms SWD to be a continuous pruning method. Similar tendencies can be observed with ResNet-20 on CIFAR-10 in Table 4. However, the best pairs of $a_{\text{min}}/a_{\text{max}}$ in both do not match.

These specific results could hardly be generalized to any problem since they heavily depends on the training conditions and especially the number of epochs. That is why finding the right pair of values for these two hyperparameters is not an obvious task. Learning the value of $a$ during training could be a way to overcome this.

6. Conclusion

We have proposed a new approach to prune deep convolutional networks continuously during training. Our theoretically motivated method, Selective Weight Decay (SWD), shows a better performance/parameters tradeoff when compared with reference methods from the literature. We have shown that our method performs better and is lighter to deploy, as it does not need any fine-tuning process after the network is pruned. One great advantage of SWD is that it can be combined with virtually any pruning criterion on any pruning structure, which opens a field of unlimited combinations. Also, the hyperparameter $a$ and its bounds, $a_{\text{min}}$ and $a_{\text{max}}$, deserve to be studied further and leave room for future improvements.

References

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. ACM Journal on Emerging Technologies in Computing Systems, 13, 12 2015.

[2] Sajid Anwar and Wonyong Sung. Compact deep convolutional neural networks with coarse pruning. arXiv, pages arXiv–1610, 2016.

[3] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. arXiv, pages arXiv–1711, 2017.

[4] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? arXiv, pages arXiv–2003, 2020.

[5] Ghouthi Boukli Hacene. Processing and learning deep neural networks on chip. PhD thesis, Ecole nationale supérieure Mines-Télécom Atlantique Bretagne Pays de la Loire, 2019.

[6] M. A. de Carreira-Perpinan and Y. Idelbayev. ”learning-compression” algorithms for neural net pruning. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8532–8541, 2018.

[7] Yves Chauvin. A back-propagation algorithm with optimal use of hidden units. In D. S. Touretzky, editor, Advances in Neural Information Processing Systems 1, pages 519–526. Morgan-Kaufmann, 1989.

[8] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 3123–3131. Curran Associates, Inc., 2015.

[9] X. Dai, H. Yin, and N. K. Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. IEEE Transactions on Computers, 68(10):1487–1497, 2019.

[10] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 1269–1277. Curran Associates, Inc., 2014.

[11] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. arXiv, pages arXiv–1907, 2019.

[12] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 4857–4867. Curran Associates, Inc., 2017.

[13] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. arXiv, pages arXiv–1911, 2019.

[14] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. arXiv, pages arXiv–1902, 2019.

[15] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dmns. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, Advances
in Neural Information Processing Systems 29, pages 1379–1387. Curran Associates, Inc., 2016. 4
[16] Gouthi Boukli Hacene, Carlos Lassance, Vincent Gripon, Matthieu Courbariaux, and Yoshua Bengio. Attention based pruning for shift networks. arXiv preprint arXiv:1905.12300, 2019. 4
[17] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv, pages arXiv–1510.2015. 1
[18] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 1135–1143. Curran Associates, Inc., 2015. 1, 3, 4, 6, 7
[19] Stephen Jose Hanson and Lorien Y. Pratt. Comparing biases for minimal network construction with back-propagation. In D. S. Touretzky, editor, Advances in Neural Information Processing Systems 1, pages 177–185. Morgan-Kaufmann, 1989. 4
[20] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, Advances in Neural Information Processing Systems 5, pages 164–171. Morgan-Kaufmann, 1993. 3, 4
[21] Babak Hassibi, David G. Stork, and Gregory Wolff. Optimal brain surgeon: Extensions and performance comparisons. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, Advances in Neural Information Processing Systems 6, pages 263–270. Morgan-Kaufmann, 1994. 4
[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 1, 7
[23] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, pages 2234–2240, 2018. 3, 4
[24] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 1398–1406, 2017. 4
[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. stat, 1050.9, 2015. 1
[26] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv, pages arXiv–1607, 2016. 4
[27] Q. Huang, K. Zhou, S. You, and U. Neumann. Learning to prune filters in convolutional neural networks. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 709–718, 2018. 4
[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 1
[29] Steven A. Janowsky. Pruning versus clipping in neural networks. Phys. Rev. A, 39:6600–6603, Jun 1989. 4
[30] E. D. Karnin. A simple procedure for pruning back-propagation trained neural networks. IEEE Transactions on Neural Networks, 1(2):239–242, 1990. 4
[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, 2017. 1
[32] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, Advances in Neural Information Processing Systems 4, pages 950–957. Morgan-Kaufmann, 1992. 2
[33] C. Lassance, M. Bontonou, G. B. Hacene, V. Gripon, J. Tang, and A. Ortega. Deep geometric knowledge distillation with graphs. In ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 8484–8488, 2020. 1
[34] Yann Le Cun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436–444, 2015. 1, 3
[35] Yann Le Cun, John S. Denker, and Sara A. Solla. Optimal Brain Damage, page 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. 3, 4
[36] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In Shape, contour and grouping in computer vision, pages 319–345. Springer, 1999. 1
[37] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. arXiv, pages arXiv–1608, 2016. 4
[38] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013. 1
[39] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shou meng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017. 1, 4, 5, 6, 7, 8
[40] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through L0 regularization. In International Conference on Learning Representations, 2018. 4
[41] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017. 4
[42] Decebal Mocanu, Elena Mocanu, Peter Stone, Phuong Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nature Communications, 9, 12 2018. 3
[43] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In 5th International Conference on Learning Representations, ICLR 2017-Conferece Track Proceedings, 2019. 4
[44] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. arXiv, pages arXiv–1902, 2019. 3
[45] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. S. Touretzky, editor, Advances in Neural Information Processing Systems 1, pages 107–115. Morgan-Kaufmann, 1989. 3, 4
[46] Walter Murray and Kien-Ming Ng. Algorithm for nonlinear optimization problems with binary variables. Computational Optimization and Applications, 47:257–288, 10 2010. 5
[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alch´e Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8026–8037. Curran Associates, Inc., 2019. 6
[48] David C Plaut et al. Experiments on learning by back propagation. 1986. 2
[49] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks, 2016. 1
[50] R. Reed. Pruning algorithms-a survey. IEEE Transactions on Neural Networks, 4(5):740–747, 1993. 3, 4
[51] B. E. Segee and M. J. Carter. Fault tolerance of pruned multilayer networks. In IJCNN-91-Seattle International Joint Conference on Neural Networks, volume ii, pages 447–452 vol.2, 1991. 4
[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 1
[53] Suraj Srinivas, Akshayvarun Subramanya, and R. Venkatesh Babu. Training sparse neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, July 2017. 4
[54] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. arXiv, pages arXiv–1507, 2015. 4
[55] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015. 1
[56] Volker Tresp, Ralph Neuneier, and Hans-Georg Zimmermann. Early brain damage. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems 9, pages 669–675. MIT Press, 1997. 4
[57] Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. AutoPrune: Automatic network pruning by regularizing auxiliary parameters. In Advances in Neural Information Processing Systems, 2019. 4
[58] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1492–1500, 2017. 1
[59] Kohei Yamamoto and Kurato Maeno. Pcas: Pruning channels with attention statistics for deep network compression, 2019. 4
[60] R. Yu, A. Li, C. Chen, J. Lai, V. I. Morariu, X. Han, M. Gao, C. Lin, and L. S. Davis. Nisp: Pruning networks using neuron importance score propagation. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9194–9203, 2018. 4