Deep Neural Networks pruning via the Structured Perspective Regularization*

Matteo Cacciola†, Antonio Frangioni ‡, Xinlin Li§, and Andrea Lodi ¶

Abstract. In Machine Learning, Artificial Neural Networks (ANNs) are a very powerful tool, broadly used in many applications. Often, the selected (deep) architectures include many layers, and therefore a large amount of parameters, which makes training, storage and inference expensive. This motivated a stream of research about compressing the original networks into smaller ones without excessively sacrificing performances. Among the many proposed compression approaches, one of the most popular is *pruning*, whereby entire elements of the ANN (links, nodes, channels, . . .) and the corresponding weights are deleted. Since the nature of the problem is inherently combinatorial (what elements to prune and what not), we propose a new pruning method based on Operational Research tools. We start from a natural Mixed-Integer-Programming model for the problem, and we use the Perspective Reformulation technique to strengthen its continuous relaxation. Projecting away the indicator variables from this reformulation yields a new regularization term, which we call the Structured Perspective Regularization, that leads to structured pruning of the initial architecture. We test our method on some ResNet architectures applied to CIFAR-10, CIFAR-100 and ImageNet datasets, obtaining competitive performances w.r.t. the state of the art for structured pruning.

Key words. Compression, Artificial Neural Networks, Optimization

MSC codes. 68T07, 90C10, 90C25

1. Introduction.

The striking practical success of Artificial Neural Networks (ANN) has been initially driven by the ability of adding more and more parameters to the models, which has led to vastly increased accuracy. This brute-force approach, however, has numerous drawbacks: besides the ever-present risk of overfitting, massive models are costly to store and run. This clashes with the ever increasing push towards edge computing of ANN, whereby neural models have to be run on low power devices such as smart phones, smart watches, and wireless base stations [29, 52, 43]. While one may just resort to smaller models, the fact that a large model trained even for a few epochs performs better than smaller ones trained for much longer lends credence to the claim [34] that the best strategy is to initially train large and over-parameterized models and then shrink them through techniques such as pruning and low-bit quantization.

Loosely speaking, pruning requires finding the best compromise between removing some of the elements of the ANN (weights, channels, filters, layers, blocks, . . .) and the decrease in accuracy that this could bring [35, 30, 26]. Pruning can be performed while training or after

*Submitted to the editors DATE.

Funding: This work has been supported by the NSERC Alliance grant 544900- 19 in collaboration with Huawei-Canada

†CERC, Polytechnique Montréal,
Montréal, QC, Canada (matteo.cacciola@polymtl.ca).

‡University of Pisa, Pisa, PI, Italy (frangio@di.unipi.it).

§Huawei Montreal Research Centre, Montreal, QC, Canada (xinlin.li1@huawei.com).

¶CERC, Polytechnique Montréal, Montréal, QC, Canada, and Jacobs Technion-Cornell Institute, Cornell Tech and Technion - IIT, New York, NY, USA (andrea.lodi@cornell.edu).

training. The advantage of the latter is the ability of using standard training techniques unmodified, which may lead to better peformances. On the other hand, pruning while training automatically adapts the values of the weights to the new architecture, dispensing with the need to re-train the pruned ANN.

A relevant aspect of the process is the choice of the elements to be pruned. Owing to the fact that both ANN training and inference is nowadays mostly GPU-based, pruning an individual weight may yield little to no benefit in case other weights in the same "computational block" are retained, as the vector processing nature of GPUs may not be able to exploit un-structured forms of sparsity. Therefore, in order to be effective pruning has to be achieved simultaneously on all the weights of a given element, like a channel or a filter, so that the element can be deleted entirely. The choice of the elements to be pruned therefore depends on the target ANN architecture, an issue that has not been very clearly discussed in the literature so far. This motivates a specific feature of our development whereby we allow to arbitrarily partition the weight vector and measure the sparsity in terms of the number of partitions that are eliminated, as opposed to just the number of weights.

In this work, we develop a novel method to perform structured pruning during training through the introduction of a Structured Perspective Regularization (SPR) term. More specifically, we start from a natural exact Mixed-Integer Programming (MIP) model of the sparsity-aware training problem where we consider, in addition to the loss and ℓ_2 regularization, also the ℓ_0 norm of the structured set of weights. A novel application of the Perspective Reformulation technique leads to a tighter continuous relaxation of the original MIP model and ultimately to the definition of the SPR term. Our approach is therefore principled, being grounded on an exact model rather than based on heuristic score functions to decide what entities to prune as prevalent in the literature so far. It is also flexible as it can be adapted to any kind of structured pruning, provided that the prunable entities are known before the training starts, and the final expected amount of pruning is controlled by the hyper-parameter providing the weight of the ℓ_0 term in the original MIP model. While our approach currently only solves a relaxation of the integer problem, it would clearly be possible to exploit established Operations Research techniques to improve on the quality of the solution, and therefore of the pruning. Yet, the experimental results show that our approach is already competitive with, and often significantly better than, the state of the art. Furthermore, since we perform pruning during training by just changing the regularization term, our approach can use standard training techniques and its cost is not significantly higher than the usual training without sparsification.

2. Related works.

The field of pruning is experiencing a growing interest in the Machine Learning (ML) community, starting from the seminal work [28] that obtained unexpectedly good results from a trivial magnitude-based approach. The same magnitude-based approach was extended in [22] with a re-training phase where the non-pruned weights are re-initialized to their starting values. Moreover, in [51] the authors claim that, for most pruning methods, the most important result is the final structure of the pruned ANN, while the final values of the weights or their original initialization are not crucial.

A multitude of pruning approaches has been developed over the years, including but not limited to Bayesian methods [56, 7, 53, 79], regularization methods [72, 48], and combinations of pruning with other compression techniques [3, 55, 23]. Part of the literature [6, 27, 10, 76] focuses on pruning without modifying the model outputs or at least trying to minimize the output change. This approach can be effective when the model is highly over-parameterized or when very few parameters need to be pruned, but it is sub-optimal otherwise.

Another possibility is adding to the network parameters a scaling factor for each prunable entity, multiplying all the corresponding parameters; then, sparsity is enforced by adding the ℓ_1 norm of the scaling factors vector, as done for example in [50]. In [61] a pruning mask is defined, i.e., a differentiable approximation of a thresholding function that pushes the scaling factors to 0 when they are lower than a fixed threshold, avoiding numerical issues. Other methods that use a similar approach are [54, 71, 49].

Most recently-published state-of-the-art pruning methods either use a magnitude-based approach to identify prunable parameters [70, 45, 12, 78, 25, 40, 55, 11], or try to estimate the impact of a parameter removal [13, 41, 63, 75, 60, 24, 15, 42, 57, 58, 74]. In both cases, they rely on heuristic rules to compute the *importance* of an element of the ANN, mostly based just on its l_2 norm. This is arguably sub-optimal in general, and we aim at improving on this by using a principled approach. The need for a more theoretically grounded approach has been clearly been felt already, as proven by the proposals [77, 9, 54, 56] that, like ours, start from an exact theoretical model of the pruning problem formulated through the l_0 norm. A significant difference, that has a profound impact on the developed technique, is that all these previous proposals do not focus on structured pruning.

Elsewhere, MIP techniques have been successfully used in the ANN context, but mostly in applications unrelated to pruning, such as the construction of adversarial examples (with fixed weights) [18]. In [4], the approach is extended to a larger class of activation functions and stronger formulations are defined. An exception is [16], where a score function is defined to assess the importance of a neuron and then a MIP is used to minimize the number of neurons that need to be kept at each layer to avoid large accuracy drops. In [62] a MIP is used first to derive bounds on the output of each neuron, which is then used in another MIP model of the entire network to find equivalent networks, local approximations, and global linear approximations with fewer neurons of the original network. Since MIPs are $\mathcal{NP}$-hard, these techniques may have difficulties scaling to large ANNs. Indeed, the pruning method developed in [1, 2] rather solves a simpler convex program for each layer to identify prunable entities in such a way that the inputs and outputs of the layer are still consistent with the original one. This layer-wise approach does not take into account the whole network at once as our own does.

The link between Perspective Reformulation techniques and sparsification has been previously recognized [14, 5], but typically in the context of regression problems that are much simpler than ANNs. In particular, all the above papers count (the equivalent of) each weight individually, and therefore they do not consider structured pruning of sets of related weights as it is required for ANNs. Furthermore, the sparsification approach is applied to input variables selection in settings that typically have orders of magnitude fewer elements to be sparsified than the present one.

3. Mathematical model. We are given a dataset X, an ANN model architecture whose set of parameters $W = \{ w_j \mid j \in I \}$ includes *prunable entities*, that is, disjoint subsets $\{ W_i =$

$[\, w_j \,]_{j \in E_i} \,\}_{i \in N}$ for disjoint subsets of indices $\{\, E_i \,\}_{i \in N}$ s.t. $I \supseteq \cup_{i \in N} E_i$, and a loss function $L(\cdot)$. If the value of a parameter w_j is zero it could be eliminated from the model (pruned) but, for the reasons discussed above, we are only interested in pruning the entities E_i, which corresponds to $w_j = 0$ for all $j \in E_i$. We therefore face a three-objective optimization problem which aims at: i) minimize the loss, ii) minimize some standard regularization term aiming a improving the model's generalization capabilities, and iii) maximize the number of pruned entities E_i. As customary in this setting, we approach this by scaling the three objective functions by means of hyperparameters whose optimal values are found by standard grid-search techniques. Employing the usual ℓ_2 regularization, the problem can be cast as the MIP

$$\min \, L(X, W) + \lambda [\, \alpha \| \, W \, \|_2^2 + (1 - \alpha) \textstyle\sum_{i \in N} y_i \,] \tag{3.1}$$

$$- M y_i \leq w_j \leq M y_i \qquad w_j \in E_i \quad i \in N \tag{3.2}$$

$$y_i \in \{0, 1\} \qquad\qquad i \in N \tag{3.3}$$

where $\alpha \in [\, 0 \,,\, 1 \,]$ and $\lambda > 0$ are scalar hyper-parameters while M is an upper bound on the absolute value of the parameters. The binary variable y_i is 0 if the corresponding prunable entity is pruned, 1 if it is not. The standard "big-M" constraints (3.2) ensure that if $y_i = 0$ then $0 \leq w_j \leq 0$ for all parameters in the entity E_i, while if $y_i = 1$ the parameters can take any possible useful value (since M is an upper bound). Hence, the term "$\sum_{i \in N} y_i$" in the objective (3.1) represents the ℓ_0 norm of the structured set of weights. In the unstructured case, i.e., when each E_i is a singleton, the standard sparsification approach is to substitute the ℓ_0 norm with the ℓ_1 one; this allows to do away with the y_i variables entirely, replacing the corresponding term in the objective with $\| \, W \, \|_1$. This *elastic net regularization* [81] combines the properties of the ridge/Tikhonov (ℓ_2) and Lasso (ℓ_1) regularizations; it has also been extended to different forms, like the *Huber regularization* [33, 59] where the ℓ_2 and ℓ_1 norms, rather than being summed, are applied to different subsets of the space. The choice of the ℓ_1 norm is motivated by it being the best possible convex approximation of the nonconvex (and not even continuous) ℓ_0 one. However, these arguments do not readily carry over to the structured case.

3.1. The Perspective Reformulation. Basically all known strategies to solve MIPs like (3.1)–(3.3), be them exact or heuristic, start from considering its *continuous relaxation* whereby (3.3) is relaxed to $y_i \in [\, 0 \,,\, 1 \,]$. Such a problem is significantly easier than the original MIP, in the sense that a locally optimal solution $(\bar{w}, \bar{y})$ is efficiently obtainable using standard techniques for ANN training. However, it is well-known that such a solution can be rather different from the optimal solution (w^*, y^*) of (3.1)–(3.3), in both the y and w variables, due to the rather crude approximation of the nonconvex constraints (3.3) by means of their convex counterpart $y_i \in [\, 0 \,,\, 1 \,]$. This would hold even if the $(\bar{w}, \bar{y})$ were globally optimal, which happens, e.g., if $L(X, \cdot)$ is convex (not typical in the ANN context), save in the fortunate case where $\bar{w}$ happens to satisfy (3.3). Since $\bar{w}$ is typically what one could use to decide what entities to remove, this could lead to inefficient prunings. We therefore we seek a different relaxation that can provide us with higher quality solutions. In principle, an "exact" convex relaxation exists, which is obtained by constructing the *convex envelope* of the objective function (3.1) on the set of integer solutions, i.e., its best possible convex approximation (technically, the convex function with smallest epigraph containing that of the original function). However, constructing the convex envelope of a function is in general $\mathcal{NP}$-hard, even in much less demanding settings than (3.1)–(3.3). A strategy that has proved successful is to devise convex envelope formulæ of fragments of the problems with specific structure; while the combination of these is typically not equivalent to the true convex envelope, it is often a much better approximation, leading to much better continuous relaxation solutions and therefore more efficient computational approaches. We can rewrite (3.1)-(3.3) as the following unconstrained optimization problem,

$$\min \left\{ L(X,W) + \lambda \left[\textstyle\sum_{i \in N} h_i(W_i, y_i) \right] \right\},$$

where

$$h_i(W_i, y_i) = \begin{cases} 0 & \text{if } y_i = 0 \text{ and } w_j = 0 \ \ \forall j \in E_i \\ \alpha \sum_{j \in E_i} w_j^2 + (1-\alpha) & \text{if } y_i = 1 \text{ and } |w_j| \leq M \ \forall\, j \in E_i \\ +\infty & \text{otherwise.} \end{cases}$$

The (clearly, nonconvex) function $h_i(\cdot,\cdot)$ belongs to a class of functions whose convex envelope can be explicitly computed: following [20], the convex envelope of h_i can be proven to be

$$\hat{h}_i(W_i, y_i) = \begin{cases} 0 & \text{if } y_i = 0 \text{ and } w_j = 0 \ \ \forall j \in E_i \\ \alpha \sum_{j \in E_i} \frac{w_j^2}{y_i} + (1-\alpha) y_i & \text{if } |w_j| \leq y_i M \ \forall j \in E_i \text{ and } y_i \in (0,1] \\ +\infty & \text{otherwise.} \end{cases}$$

This leads to the new formulation of problem (3.1)–(3.3)

$$\min \left\{ L(X,W) + \lambda \textstyle\sum_{i \in N} \left[\alpha \sum_{j \in E_i} \frac{w_j^2}{y_i} + (1-\alpha) y_i \right] \ : \ (3.2)\ ,\ (3.3) \right\} \tag{3.4}$$

known in the literature as *Perspective Reformulation* (PR), that is easily seen to have the same integer optimal solution (w^*, y^*) as the original problem but a continuous relaxation (the *Perspective Relaxation*) that is "better" in a well-defined mathematical sense: its optimal objective value is (much) closer to the true optimal value of (3.1)–(3.3), which typically implies that its optimal solution $(\bar{w}, \bar{y})$ is more similar to the true optimal solution (w^*, y^*). Indeed, $\hat{h}_i(W_i, y_i)$ can be seen to have larger value than $h_i(W_i, y_i)$, the more so the more y_i is close to 0.5, i.e., "farther from being integer" [20], thereby discouraging highly fractional values in y^*. This has been already shown to leading to much better performances of both exact and heuristic approaches, w.r.t. using the standard continuous relaxation, for other MIPs with similar structure.

3.2. Eliminating the y variables. While one can expect that the solution $(\bar{w}, \bar{y})$ of the Perspective Relaxation can provide a better guide to the pruning procedure, the presence of the explicit variables y makes it more difficult to apply standard training techniques to obtain it. Following the lead of [21, 19], we proceed at simplifying the PR model by projecting away the y variables. This amounts to computing a closed formula $\tilde{y}(w)$ for the optimal value of the y variables in the continuous relaxation of (3.4) assuming that w are fixed: the problem then decomposes over the E_i subsets, and therefore we only need to consider each fragment

$$f_i(W_i, y_i) = \lambda\big[\,\alpha \textstyle\sum_{j\in E_i} w_j^2/y_i + (1-\alpha)y_i\,\big]$$

separately. Since f_i is convex in y_i if $y_i > 0$, we just need to find the root of the derivative

$$\frac{\partial f_i(W_i, y_i)}{\partial y_i} = \lambda\left[-\alpha \sum_{w_j\in E_i} \frac{w_j^2}{y_i^2} + (1-\alpha)\right] = 0\ \ ,$$

that is

$$y_i = \sqrt{\frac{\alpha \sum_{w_j\in E_i} w_j^2}{1-\alpha}}$$

(we are only interested in positive y), and then project it on the domain. Note that, technically, $f_i(W_i, y_i)$ is nondifferentiable for $y_i = 0$ but that value is only achieved when $W_j = 0$, in which case the choice is obviously optimal. The constraints that defines the domain of y_i can be rewritten as $y_i \geq |\,w_j\,|/M$ for all $j \in E_i$, together with $y_i \in [0, 1]$; putting everything together, we obtain

$$\text{(3.5)}\qquad \tilde{y}_i(w) = \min\Big\{\ \max\Big\{\{\,|w_j|/M\ :\ j\in E_i\}\,,\ \sqrt{\alpha \textstyle\sum_{j\in E_i} w_j^2/(1-\alpha)}\,\Big\}\,,\ 1\,\Big\}$$

where we note that we do not need to enforce positivity since all the quantities are positive.

Replacing y_i with $\tilde{y}_i(W_i)$ in the objective function of (3.4) we can rewrite the continuous relaxation of (3.4) as

$$\text{(3.6)}\qquad \min\big\{\, L(X, W) + \lambda \textstyle\sum_{i=1}^N z_i(W_i; \alpha, M)\,\big\},$$

where

$$z_i(W_i;\alpha,M) = \begin{cases} \alpha \sum_{j\in E_i} \frac{\sqrt{(1-\alpha)}w_j^2}{\sqrt{\alpha\sum_{j\in E_i} w_j^2}} + (1-\alpha)\sqrt{\frac{\alpha\sum_{j\in E_i} w_j^2}{(1-\alpha)}} & \text{if } \frac{\|W_i\|_\infty}{M} \leq \sqrt{\frac{\alpha\sum_{j\in E_i} w_j^2}{1-\alpha}} \leq 1 \\ \alpha \sum_{j\in E_i} \frac{w_j^2 M}{\|W_i\|_\infty} + (1-\alpha)\frac{\|W_i\|_\infty}{M} & \text{if } \sqrt{\frac{\alpha\sum_{j\in E_i} w_j^2}{1-\alpha}} \leq \frac{\|W_i\|_\infty}{M} \leq 1 \\ \alpha \sum_{j\in E_i} w_j^2 + (1-\alpha) & \text{otherwise,} \end{cases}$$

$$= \begin{cases} \sqrt{(1-\alpha)\alpha}||W_i||_2 + \sqrt{(1-\alpha)\alpha}||W_i||_2 & \text{if } \frac{||W_i||_\infty}{M} \leq \sqrt{\frac{\alpha}{1-\alpha}}||W_i||_2 \leq 1 \\ \frac{\alpha M}{||W_i||_\infty}||W_i||_2^2 + (1-\alpha)\frac{||W_i||_\infty}{M} & \text{if } \sqrt{\frac{\alpha}{1-\alpha}}||W_i||_2 \leq \frac{||W_i||_\infty}{M} \leq 1 \\ \alpha||W_i||_2^2 + (1-\alpha) & \text{otherwise,} \end{cases}$$

$$\text{(3.7)}\qquad = \begin{cases} 2\sqrt{(1-\alpha)\alpha}||W_i||_2 & \text{if } \frac{||W_i||_\infty}{M} \leq \sqrt{\frac{\alpha}{1-\alpha}}||W_i||_2 \leq 1 \\ \frac{\alpha M}{||W_i||_\infty}||W_i||_2^2 + (1-\alpha)\frac{||W_i||_\infty}{M} & \text{if } \sqrt{\frac{\alpha}{1-\alpha}}||W_i||_2 \leq \frac{||W_i||_\infty}{M} \leq 1 \\ \alpha||W_i||_2^2 + (1-\alpha) & \text{otherwise.} \end{cases}$$

We call $z_i(W_i; \alpha, M)$ the *Structured Perspective Regularization* (SPR) w.r.t. the structure specified by the sets E_i. It is easily seen that the SPR behaves like the ordinary ℓ_2 regularization in parts of the space but it is significantly different in others. Due to being derived from (3.4), we can expect, all other things being equal, the SPR to promote sparsity—in terms of the sets E_j—better than the ℓ_2 norm. Indeed, SPR for $i \in I$ depends on the ℓ_∞ norm of W_i. This means that it penalizes entities on the ground of their maximum non-zero component, regardless to how many w_j have the maximum value. This arguably better promotes structured sparsity, as required by our application, w.r.t., say, using the ordinary ℓ_1 norm that rather promotes sparsity on each weight individually. This intuition is substantiated in the next § 3.3 where a more detailed discussion about the properties of the SPR regularizer can be found. Yet, all of the usual algorithms for training ANNs (SGD, Adam, etc.) can be employed for the solution of (3.6), which therefore should not, in principle, be more costly than non-sparsity-inducing training or unstructured sparsity-inducing terms like the ℓ_1 norm.

It is perhaps useful to remark that the Lasso/elastic net regularization can be seen as the application of an analogous process in the non-structured case. Indeed, assume W is fixed in (3.1)–(3.3): the optimal value of the y variables in the continuous relaxation of the problem solves (independently for each i)

$$\min\{\, (1-\alpha) y_i \,:\, (3.2)\,,\, y_i \in [\,0\,,\,1\,]\,\}$$

where the constraints are of course equivalent to $y_i \geq |w_i|/M$: hence, $y_i^* = |w_i|/M$, which leads to the replacing of the ℓ_0 norm with the ℓ_1 one. Thus, our approach can be seen as a generalization of the standard one, but with two meaningful differences: i) it takes into account the effect of the quadratic regularization term, and ii) it applies the PR to the problem before doing the projection. Note that the first point is crucial to the second, because the PR of a linear function is easily seen to be the original function itself: in other words, the PR has no effect on linear problems. It is interesting to remark what happens to the SPR term in the context of unstructured pruning. In this case, the vector W_i in (3.7) is just a scalar, so $\|W_i\|_\infty = \|W_i\|_2 = |W_i|$ and the formula becomes much simpler. First, we only get two possible cases: if $1/M \leq \sqrt{\alpha/(1-\alpha)}$, then the second case is never possible; otherwise, it is the first case that never verifies. Moreover, both the first and the second cases of (3.7) become equal to the l_1 norm times a constant. This yield the known Berhu (reverse Huber) penalty [39], which has already been shown to be effective. However, doing this in the structured case is novel, and yields the SPR term that is significantly more complex than what was previously known, as better illustrated next.

3.3. Intuition on our new regularization term. We now provide a discussion on the shape of the SPR term, focussing on the features that could be linked to its better structured sparsification properties. We remark that, unlike what was done with the heuristic approaches in the literature, we did not develop the SPR in order to obtain such properties: instead, they were the natural results of constructing a better continuous approximation of the inherently combinatorial (and, therefore, hard) exact training-with-structured-sparsification problem (3.1)–(3.3).

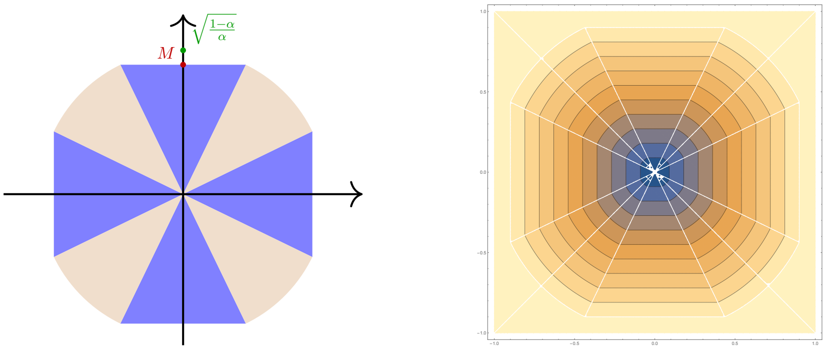
First, we notice that SPR is not differentiable in zero. Since the gradient does not vanish in points close to the origin, this is known to increase the amount of parameters that are effectively zero after training is completed; indeed, this is the effect underlying the Lasso (l_1) regularizer for unstructured sparsity. This property is likely crucial, and in fact it is common to basically all other regularization-based approaches to structured sparsification, many of which use the non-squared ℓ_2 norm (also known as l_2/l_1 norm [11, 48]). Again, this feature was not planned, but it emerged as a result of our principled approach.

Out of 0, the behaviour of the SPR is different in different zones of the space. In particular, when the norm of a prunable entity is “large” (more precisely, when at least one among $\|W_i\|_\infty \geq M$ and $\|W_i\|_2 \geq \sqrt{(1-\alpha)/\alpha}$ holds, the white region of Figure 1), then SPR is equivalent to the standard ridge/Tikhonov (ℓ_2) regularization. Intuitively, the SPR identifies the entities that are “not likely” to be pruned, and, since no structured regularization needs to be applied there, the usual regularization is used which is still needed for generalization purposes. This is similar to the (much simpler) Berhu regularization [39] (for the unstructured case) that coincides with the l_2 norm “far from 0”, while rather being the (nondifferentiable) l_1 norm “close to 0”. Again, we did not explicitly plan for this to happen, and such a behavior is not foreseen in the popular regularizers employed in the sparsification literature.

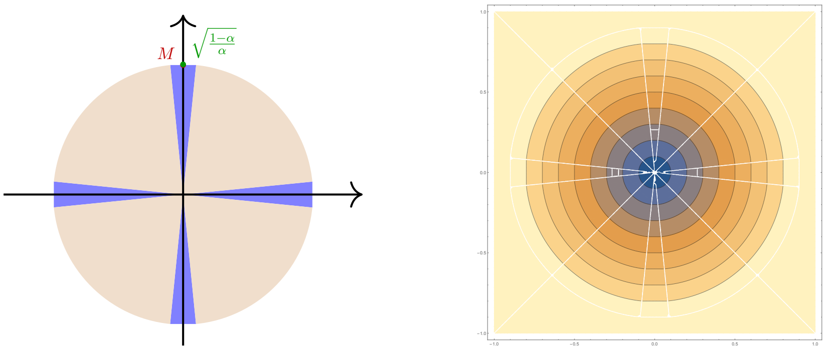
If an entity is “still within pruning range”, SPR has a complex behavior organized around two different kinds of regions of the space. The first is the one in which a few parameters of an entity have disproportional larger absolute value compared to the others in the same entity (more precisely when $\|W_i\|_\infty \geq \sqrt{\alpha/(1-\alpha)}M\|W_i\|_2$, blue region of Figure 1). There the SPR is close to the infinity norm, and therefore the learning process focuses on reducing precisely the largest entries, since the infinity norm gradient is non-zero only in the entries corresponding to the coordinates in which the norm is reached (the ones with maximum absolute value). From a structured pruning point of view, entities with *unbalanced* parameters are not ideal since they may have many “small” (even possibly 0) weights, that therefore likely provide small (or null) benefit in terms of loss reduction, and yet they can not be removed due to a “few” large weights. The SPR identifies such entities and promotes the reduction of the disproportion among the weight magnitudes, possibly leading to the final removal.

In fact, when instead an entity has parameters with similar magnitudes (more precisely when $\|W_i\|_\infty \leq \sqrt{\alpha/(1-\alpha)}M\|W_i\|_2$, grey region of Figure 1), a sparse gradient could cause convergence speed problems. In this case, the SPR is equal to the (non-squared) l_2 norm whose gradient is not sparse; thus, the SPR promotes the simultaneous reduction of all the parameters, hopefully finally leading to the pruning of the entity.

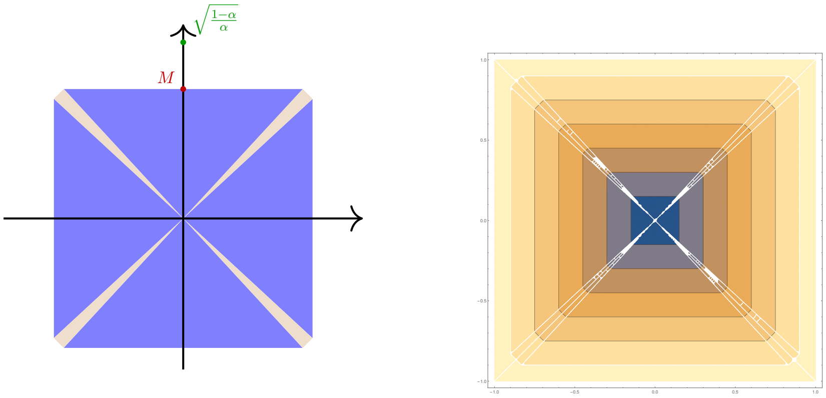
A pictorial representation of the previous discussion is provided in Figure 1 for a two-dimensional entity, with the left panels highlighting the regions where each case of the SPR occurs, while the right panels show the level sets of the SPR term that induces structured sparsity (that is, the term that multiplies $(1-\alpha)$ in (3.7)). Different plots corresponding to different choices of α (for fixed and M) are given to illustrate the complexity of the term as a function of its hyperparameters, and therefore its flexibility. A three-dimensional plot of the SPR term that induces structured sparsity is reported in Figure 2, illustrating how it transitions between different regions. Arguably, such a complex behaviour would have been rather complex to engineer; yet, it naturally emerged from our use of sophisticated mathematical optimization techniques.



(a) $M = 0.9$, $\alpha = 0.5$



(b) $M = 0.9$, $\alpha = 0.55$



(c) $M = 0.9$, $\alpha = 0.4$

Figure 1: Left, regions in which the SPR changes definition, right level sets of the structured sparsity term of the SPR

3.4. Minor improvements. Remarkably, the SPR depends on the choice of M, which is, in principle, nontrivial. Indeed, all previous attempts of using PR techniques for promoting

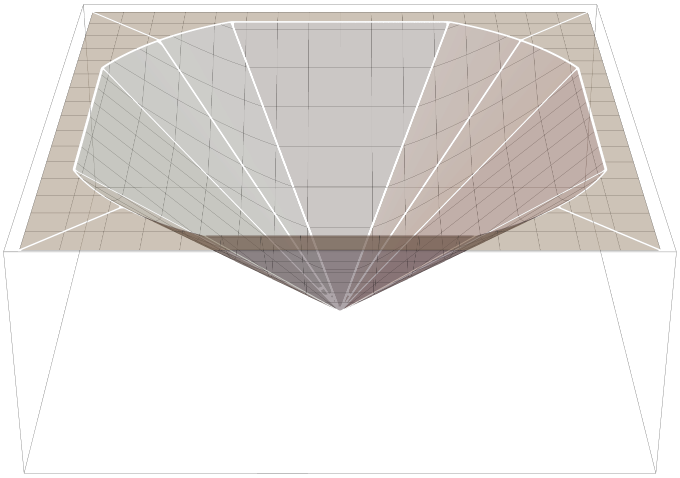


Figure 2: 3-dimensional plot of the structured sparsity term of the SPR. When the norm of the entity is big enough, the term is constant. Otherwise, it is more similar to the l_2 or l_∞ norm, based on how are distributed the weights in the entity.

(non-structured, i.e., $E_i = \{ i \}$) sparsity [14, 5] have been using the "abstract" nonlinear form $(1 - y_i)w_i = 0$ of (3.2). This still yields the same Perspective Reformulation, but it is not conducive to projecting away the y variables as required by our approach. While M could in principle be treated as another hyperparameter, in a (deep) ANN, different layers can have rather different optimal upper bounds on the weights; hence, using a single constant M for all the prunable entities is sub-optimal. The ideal choice would be to compute one constant M_i for each entity E_i; however, entities in the same layer are often similar to each other, so we only computed a different constant for each layer of the network, as detailed in §4.1, and used it for all entities belonging to that layer.

Furthermore, all the development so far has assumed that all prunable entities E_i are equally important. However, this may not be true, since different entities can have different number of parameters and therefore impact differently on the overall memory and computational cost. To take this feature into account, we modify our regularization terms as

$$\lambda \sum_{i \in N} \frac{u_i}{\sum_{i \in N} u_i} z_i(W_i; \alpha, M),$$

where u_i is the number of parameters belonging to entity E_i.

Finally, we perform a fine-tuning phase. After the ANN has been trained with the SPR, we prune all the entities W_i where 99.5% of the weights are smaller than the tolerance which is found using Algorithm 3.1. The threshold value 99.5% has been obtained through simple preliminary experiments. Though it could be treated as an hyperparameter and tuned accordingly, we did not deem this necessary since the experiments have shown that it plays a limited role in the final performances. We re-train the compressed network with the standard ℓ_2 regularization, starting from the value of the weights (for the non-pruned entities) obtained at the end of the previous phase rather than re-initializing them.

Algorithm 3.1 performs a binary search in a given interval to find the highest possible pruning threshold that does not heavily affect the accuracy of the model. At each iteration, the candidate threshold is set to the medium point of the current interval, the ANN is pruned with such threshold and the new training accuracy is computed. If there was a drop in the accuracy larger than a given tolerance, the threshold is discarded and the first half of the interval becomes the interval for the next iteration. Otherwise, the threshold is accepted and the new interval is the second half of the current one. In our experiments we used $N = 10$, $a = 0$, $b = 1\text{e-}1$ and $\delta = 5\text{e-}2$.

Algorithm 3.1 Given a trained ANN with ρ^* training accuracy, the algorithm searches for the highest threshold in the interval $[a, b]$ such that the ANN compressed with such threshold does not lose more than δ accuracy.

Require: N, ρ^*, δ and $[a, b]$
  $\epsilon^* \leftarrow a$
  for $i = 1, \dots, N$ **do**
    $\epsilon \leftarrow (a + b)/2$
    compress the network with the threshold ϵ and compute the current training accuracy ρ
    if $\rho \geq \rho^* - \delta$ **then** $a \leftarrow \epsilon^* \leftarrow \epsilon$
    else $b \leftarrow \epsilon$ **end if**
  end for
  return ϵ^*

4. Experiments.

We tested our method on the task of filter pruning in Deep Convolutional Neural Networks; that is, the prunable entities are the filters of the convolutional layers. More specifically, the weights in a convolutional layer with n_{inp} input channels, n_{out} output channels and $k \times k$ kernels is a tensor with four dimensions (n_{inp}, n_{out}, k, k): our prunable entities correspond to the sub-tensors with the second coordinate fixed, and therefore have $n_{inp} \times k \times k$ parameters. Following [11], we include in the each prunable entity the corresponding bias and weight parameter belonging to the following batch normalization layer.

The code used to run the experiments was written starting from the public repository https://github.com/akamaster/pytorch_resnet_cifar10 and https://github.com/pytorch/examples/tree/master/imagenet.

4.1. Datasets, architectures and general setup.

For our experiments, we used 3 very popular datasets: CIFAR-10, CIFAR-100 [36] and ImageNet [37]. As architectures, we focused on ResNet [31] and Vgg [65]; in particular, we used ResNet-18, ResNet-20, Resnet50, ResNet-56 and Vgg-16 for the CIFAR 10 dataset, ResNet-20 for the Cifar-100 dataset and ResNet-18 for the ImageNet dataset. We chose these dataset-architecture pairs since they were among the most common in the literature.

For all the experiments, we used Pytorch (1.12.1) with Cuda, the CrossEntropyLoss and the SGD optimizer with 0.9 momentum. The M_i values were set as the maximum absolute values of the weights for each layer of a network with the same architecture but trained without our regularization term (for ResNet-20 and ResNet-56 we trained it, for ResNet-18 we used

Table 1: Results of our algorithm on CIFAR-10 using ResNet-20

L-rate	λ	α	Acc.	Pruned pars	(%)	FLOPs	(%)
0.1	1.9	0.5	85.63	242424	(89.88)	12.70M	(31.31)
0.1	1.6	0.5	86.81	232409	(86.17)	13.77M	(33.96)
0.1	1.3	0.5	88.00	228094	(84.57)	16.62M	(40.99)
0.1	1.3	0.1	89.46	213958	(79.33)	15.01M	(37.02)
0.1	1.3	1e-4	90.03	203154	(75.32)	18.09M	(44.62)
0.1	0.8	0.1	91.22	172658	(64.01)	24.86M	(61.30)
0.1	0.5	1e-3	92.23	115620	(42.87)	29.69M	(73.23)
Original model			92.03	0	(0.00)	40.56M	(100.00)

Table 2: Results of our algorithm on CIFAR-100 using ResNet-20

L-rate	λ	α	Acc.	Pruned pars	(%)	FLOPs	(%)
0.10	0.50	0.50	65.64	160394	(58.20)	23.89M	(58.90)
0.01	1.30	0.50	67.53	102944	(37.36)	33.98M	(83.78)
0.10	0.30	0.60	68.22	79720	(28.93)	29.94M	(73.83)
0.10	0.30	0.15	68.57	61515	(22.32)	29.60M	(72.98)
0.01	1.25	0.15	69.13	42009	(15.24)	37.88M	(93.39)
Original model			68.55	0	(0.00)	40.56M	(100.00)

the pretrained version available from torchvision).

Additional details are provided in the appendix.

4.2. Results on CIFAR-10 and CIFAR-100. These experiments were performed on a single GPU, either a TESLA V100 32GB or NVIDIA Ampere A100 40GB. The model was trained for 300 epochs and then fine tuned for 200 ones. The dataset was normalized, then we performed data augmentation through random crop and horizontal flip. Mini batches of size 128 (64 for CIFAR-100) were used for training. The learning rate was initialized to either 0.1 or 0.01 and then it was divided by 10 at epochs 100 (200 for CIFAR-100), 250, 350, 400 and 450. We performed grid search on the crucial hyperparameters λ and α as detailed in §A.3.

Since the learning-with-structured-pruning problem is a multi-objective one, there is no overall best solution: rather, we report a representative selection of the non-dominated solutions on the efficient frontier (the best pruning corresponding to any achieved level of accuracy), together with the hyperparameters achieving it. An example of the pareto curve obtained through our experiments is reported in §A.4. We also report the number of floating point operations (FLOPs) necessary to perform inference for each model.

Table 1 shows the results of training ResNet-20 on CIFAR-10: we were able to prune more than 42% of the parameters by still increasing the accuracy of the original model, while we

Table 3: Results of our algorithm on CIFAR-10 using ResNet-56

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.1	1.9	0.01	90.62	762869	(89.43)	30.10M	(23.99)
0.1	1.0	5e-3	91.85	726717	(85.19)	38.94M	(31.03)
0.1	0.7	0.01	92.42	677433	(79.42)	42.65M	(33.98)
0.1	0.4	0.10	92.76	612038	(71.75)	44.08M	(35.13)
0.1	0.4	0.50	93.48	553821	(64.92)	50.90M	(40.57)
0.1	0.2	0.50	93.96	395478	(46.36)	83.58M	(66.60)
Original model			93.35	0	(0.00)	125.48M	(100.00)

could prune more than 75% of the model by still preserving more than 90% accuracy. With the same architecture on the more challenging CIFAR-100 dataset (Table 2) we could prune more than 15% of parameters while improving the accuracy of the original model, but pruning many parameters resulted in a significant accuracy loss: we could still achieve more than 67% accuracy by pruning a few less than 40% of the parameters, but accuracy dropped to less than 66% if pruning more.

Table 3 reports results on training the ResNet-56 architecture on CIFAR-10: once again pruning about 65% of the parameters improved accuracy and we could keep more than 92% accuracy while pruning almost 80% of the network.

Finally, Tables 4, 5 and 6 report results on the CIFAR-10 dataset of models Resnet-18, ResNet-50, and Vgg-16 (respectively), which have a much larger number of parameters than the previous ones: in these cases we were able to prune the vast majority of the parameters (from 89% to more than 90%) without really affecting the accuracy of the ANN, sometimes even increasing it.

4.3. Results on ImageNet. These experiments were performed on single TITAN V 8GB GPU. The model was trained for 150 epochs and fine tuned for 50 ones. The preprocessing was the same as for the CIFAR datasets. We used mini batches of 256 and 0.1 learning rate that was divided by 10 every 35 epochs, and the grid search detailed in §A.3. As usual for datasets with so many classes, we report also the top5 accuracy, i.e., the percentage of samples where the correct label was on the 5 higher scored classes by the model.

Table 4: Results of our algorithm on CIFAR-10 using ResNet-18

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.1	2.8	0.5	94.38	10691286	(95.29)	101.53M	(18.28)
0.1	2.5	0.5	94.50	10555810	(94.08)	118.51M	(21.33)
0.1	1.9	0.5	94.81	10451461	(93.15)	143.45M	(25.82)
0.1	1.3	0.5	95.34	10059742	(89.66)	192.39M	(34.64)
Original model			95.15	0	(0.00)	555.47M	(100.00)

Table 5: Results of our algorithm on CIFAR-10 using ResNet-50

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.1	1.6	1e-4	93.49	23197453	(97.86)	62.43M	(4.81)
0.1	1.6	1e-3	93.80	22977664	(96.93)	103.97M	(8.01)
0.1	1.3	1e-4	94.36	22931931	(96.74)	166.30M	(12.81)
0.1	1.0	1e-4	94.51	22745124	(95.95)	175.85M	(13.55)
0.1	1.0	0.5	94.96	21800173	(91.96)	258.10M	(19.88)
Original model			94.83	0	(0.00)	1.30B	(100.00)

Results using ResNet-18 are reported in Table 7, and show that even in a very large and

Table 6: Results of our algorithm on CIFAR-10 using Vgg-16

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.1	1.6	1e-4	93.44	14266694	(96.87)	82.00M	(26.18)
0.1	1.6	0.1	93.56	14179500	(96.27)	89.61M	(28.61)
0.1	1.0	0.5	93.93	13647661	(92.66)	126.74M	(40.47)
0.1	0.1	0.5	94.31	12044579	(81.78)	186.67M	(59.60)
Original model			94.12	0	(0.00)	313.20M	(100.00)

Table 7: Results of our algorithm on ImageNet using ResNet-18

L-rate	λ	α	top1	top5	Pruned pars (%)		FLOPs (%)	
0.1	0.75	0.1	70.26	89.66	1992131	(17.04)	2.20B	(92.84)
0.1	1.0	0.1	69.27	89.06	3811382	(32.61)	2.07B	(87.58)
0.1	1.1	0.1	68.87	88.72	4481715	(38.34)	2.03B	(85.57)
0.1	1.0	0.3	66.20	87.15	7406308	(63.36)	1.83B	(77.31)
Original model			69.76	89.08	0	(0.00)	2.37B	(100.00)

difficult dataset our method was able to improve the original model results while pruning more than 17% of the parameters, and basically tie with it while pruning 30% of the parameters. Pruning almost 40% of the network caused a drop of only 0.5% in the accuracy, while a more consistent decrease resulted when we pruned about 60% of the parameters.

4.4. Comparison with state-of-the-art methods. In this section, we compare our results (denoted as SPR) with some of the state-of-the-art algorithms for structured pruning. We report results from [32] (denoted by SSS), [64] (denoted by EPFS), [68] (denoted by L2PF), [44] (denoted by PFFEC), [73] (denoted as RSNI), [47] (denoted as HRANK), [69] (denoted as PFC), [66] (denoted by CHIP), [38] (denoted as DNR), [11] (denoted as OTO), [45] (denoted as DHP), [74] (denoted as NISP), [80] (denoted as DCP), [67] (denoted as SCOP) , [46] (denoted as PFPE) and [17] (denoted by HFP).

Since not all the above papers reported the results for all our metrics (for example, some works only reported the percentage of parameters pruned), in some cases we had to do some conversions that naturally came with some mild approximation. Moreover, in [32], only plots were presented, so we had to approximately deduce the data from some points of the figures (Figure 2(a) and Figure 2(c) of [32], we denote the points as P1, P2, etc.). For ImageNet the top5 accuracy is not reported in [17], so we marked the corresponding field in our table with a "N/A". Finally, we report results for different settings of each method as they were given in the original papers; however, it should be remarked that not all of them are structured pruning methods as our own (in particular, pruning at the filter level), hence the results may not be completely equivalent, although in general they should be comparable.

Regarding ResNet-20 on CIFAR-10, our approach (shown in Table 8) outperforms all the other methods, meaning that we could reach equal or better accuracy while pruning a larger amount of parameters. For instance, L2PF achieved 89.9% accuracy with 73.96% sparsity, while we achieved higher sparsity (79.33%) and a little more accuracy (90.03%)

On CIFAR-100 using ResNet-20, the results in Table 9 clearly show that we outperform SSS, as we could achieve more than 68.5% accuracy while pruning more than 22% of parameters while SSS could prune only 14.81% to obtain a little bit more than 67% accuracy. In Table 10, we can observe a similar situation to ResNet-20 on CIFAR-10 for ResNet-56 on the same dataset. One of the few results we did not outperform was the CHIP 94.16 accuracy with 42.8% sparsity but we could obtain a little bit more sparsity (46.36%) with a comparable accuracy (93.96%).

The results reported in Tables 11 and 12 show that our approach is very competitive with respect to the very recent state-of-the-art methods such as OTO and DNR, sometimes

Table 8: Results of state of the art method on CIFAR-10 using ResNet-20

Method	Setting	Acc.	Pruned pars	(%)
SSS	P1	90.80	120000	(44.44)
	P2	91.60	40000	(14.81)
	P3	92.00	10000	(3.70)
	P4	92.50	0	(0.00)
EPFS	B-0.6	91.91	70000	(24.60)
	B-0.8	91.50	100000	(36.90)
	F-0,05	90.83	130000	(51.10)
	C-0.6-0.05	90.98	150000	(56.00)
L2PF	LW	89.90	199687	(73.96)
PFC	P1	90.55	135000	(50.00)
DHP	50	91.54	118327	(43.87)
SCOP	P1	90.75	151853	(56.30)
PFPE	P1	90.91	169035	(62.67)
RSNI	model A	90.9	104708	(38.82)
	model B	88.8	190800	(70.74)
SPR	λ 1.3 - α 0.1	90.03	213958	(79.33)
	λ 0.8 - α 0.1	91.22	172658	(64.01)
	λ 0.5 - α 1e-3	92.23	115620	(42.87)

being able to improve them significantly. For example, DNR can only prune less than 82% of ResNet-18 achieving 94.64% accuracy, while our method reach more than 95% accuracy pruning more than 89% of the network. The only result that is somehow stronger than SPR is that obtained by the Adaptive version of DCP, see the corresponding entries in Tables 10 and 13. However, the difference in performance is not large in all cases, which confirms that SPR is at least competitive with all the alternative approaches we could compare it to.

Similarly, when training Vgg-16 on Cifar-10, our method beats all the state-of-the-art ones but the Adaptive DCP. For example, CHIP can never prune more than 88% of the ANN but our algorithm prunes consistently more than 92% achieving similar or better accuracy (Table 13).

On ImageNet using ResNet-18, the results in Table 14 show that even if our method does not outperform all the other ones, we were able to achieve very competitive results. Likely some additional parameter tuning could lead us to even more competitive results.

5. Conclusions and future directions. Based on an exact MIP model for the problem of training-with-structured-pruning of ANNs, we proposed a new regularization term, based on the projected Perspective Reformulation, designed to promote structured sparsity. The

Table 9: Results of state of the art method on CIFAR-100 using ResNet-20

Method	Setting	Acc.	Pruned pars	(%)
SSS	P1	65.50	120000	(44.44)
	P2	67.10	40000	(14.81)
	P3	68.10	10000	(3.70)
	P4	69.20	0	(0.00)
SPR	λ0.5-α0.5	65.64	160394	(58.20)
	λ 0.3 - α 0.15	68.57	61515	(22.32)
	λ 1.25 - α 0.15	69.13	42009	(15.24)

proposed method is able to prune any kind of structures, and the amount of pruning can be tuned by appropriate hyper-parameters. We tested our method on some classical datasets and architectures and we compared the results with some of the state-of-the-art structured pruning methods, proving that our method is competitive, and often outperforms existing ones.

These results are even more promising in view of the fact that further improvements should be possible. Indeed, we are currently solving the continuous relaxation of our proposed exact model, albeit a "tight" one due to the use of the Perspective Reformulation technique. By a tighter integration with other well-established MIP techniques, further improvements are foreseeable.

Appendix A. Appendix.

A.1. SPR regularity. In the following, we prove that the SPR term defined in (3.7) is continuous, differentiable almost everywhere, and non-convex but quasi-convex. Continuity of the SPR could be established by proving equality of the limits of the distinct segments defined within (3.7) at the points where the function undergoes a change in its definition, but a more concise argument uses the fact that the definition (3.7) is equivalent to the composition of (3.4) with the optimal solution formula for the optimal w variables (3.5), which is easily seen to be a continuous function of w. Furthermore, while (3.4) would seem not to be continuous in zero, it is easy to see that

$$\lim_{y_i \to 0} \sum_{j \in E_i} \frac{w_j^2}{y_i} \leq \lim_{y_i \to 0} \sum_{j \in E_i} \frac{y_i^2 M^2}{y_i} = 0,$$

on feasible solutions (y_i, w_i), i.e., when (3.2) are satified. Thus, (3.4) can be continuously extended at zero, and therefore (3.7) is a composition of continuous functions and hence continuous itself.

The fact that the SPR term is differentiable almost everywhere comes from the differentiability (almost everywhere) of the functions that define (3.7) and from the fact that the set where the SPR changes definition has zero mass. However, the previous pictures clearly show that the function can indeed be nondifferentiable there. In particular, since both the

Table 10: Results of state of the art method on CIFAR-10 using ResNet-56

Method	Setting	Acc.	Pruned pars	(%)
PFFEC	A	93.10	80000	(9.40)
	B	93.06	120000	(13.70)
EPFS	B-0.6	92.89	240000	(27.70)
	B-0.8	92.34	500000	(58.60)
	F-0.01	92.96	170000	(20.00)
	F-0.05	92.09	510000	(60.10)
	C-0.6-0.05	92.53	570000	(67.10)
HFP	0.5	93.30	425000	(50.00)
	0.7	92.31	608430	(71.58)
HRank	P1	90.72	580000	(68.10)
	P2	93.17	360000	(42.40)
	P3	93.52	140000	(16.80)
PFC	P1	93.05	425000	(50.00)
DHP	50	93.58	354685	(41.58)
	38	92.94	510958	(59.90)
SCOP	P1	93.64	480249	(56.30)
PFPE	P1	92.67	759015	(88.98)
CHIP	P1	92.05	600000	(71.80)
	P2	94.16	360000	(42.80)
NISP	P1	93.32	363386	(42.6)
DCP	P1	93.49	420014	(49.24)
	Adapt	93.81	599897	(70.33)
SPR	λ 0.7 - α 0.01	92.42	677433	(79.42)
	λ 0.4 - α 0.1	92.76	612038	(71.75)
	λ 0.4 - α 0.5	93.48	553821	(64.92)
	λ 0.2 - α 0.5	93.96	395478	(46.36)

l_1 norm and the l_∞ norm are not differentiable in zero, the SPR is not differentiable in zero, as expected from a sparsity-inducing regularization term. It is easy to see by drawing a few examples that the SPR is in general not convex. For an algebraic proof consider $\alpha = 0.65$, $M = 0.4$, $W_1 = (0.3, 0, .., 0)$ and $W_2 = (0.5, 0, .., 0)$; then $z(W_1) = 0.3405$, $z(W_2) = 0.5125$, $z(\frac{1}{2}W_1 + \frac{1}{2}W_2) = 0.4540$, $\frac{1}{2}z(W_1) + \frac{1}{2}z(W_2) = 0.4265$, where $z(\cdot)$ is defined in (3.7). We have just shown that $z(\frac{1}{2}W_1 + \frac{1}{2}W_2) > \frac{1}{2}z(W_1) + \frac{1}{2}z(W_2)$, i.e., that the SPR is not convex. Yet, the function defined in (3.5) is clearly quasi-convex and (3.4) is non-decreasing in the y

Table 11: Results of state of the art method on CIFAR-10 using ResNet-18

Method	Setting	Acc.	Pruned pars	(%)
DNR	P1	94.64	9233284	(82.36)
SPR	λ 1.3 - α 0.5	95.34	10059742	(89.66)
	λ 1.9 - α 0.5	94.81	10451461	(93.15)

Table 12: Results of state of the art method on CIFAR-10 using ResNet-50

Method	Setting	Acc.	Pruned pars	(%)
OTO	P1	94.40	21570653	(91.20)
SPR	λ 1.0 -α 0.5	94.96	21800173	(91.96)
	λ 1.3 -α 1e-4	94.36	22931931	(96.74)

Table 13: Results of state of the art method on CIFAR-10 using Vgg-16

Method	Setting	Acc.	Pruned pars	(%)
PFC	P1	93.63	7357792	(50.00)
EPSF	F-0.005	94.67	10305584	(69.10)
	F-0.001	93.61	8225584	(56.70)
PFEEC	P1	93.40	9315584	(64.00)
HRANK	P1	93.43	12205584	(82.90)
	P2	92.34	12075584	(82.10)
	P3	91.23	12935584	(92.00)
CHIP	P1	93.86	11955584	(81.60)
	P2	93.72	12215584	(83.30)
	P3	93.18	12815584	(87.30)
DNR	P1	92.00	13560314	(92.07)
PFPE	P1	92.39	13891701	(94.32)
OTO	P1	93.30	13918211	(94.50)
DCP	P1	94.16	7057294	(47.92)
	Adapt	94.57	13782934	(93.58)
SPR	λ 1.6 - α 1e-4	93.44	14266694	(96.87)
	λ 1.6 - α 0.1	93.56	14179500	(96.27)
	λ 1.0 - α 0.5	93.93	13647661	(92.66)
	λ 0.1 - α 0.5	94.31	12044579	(81.78)

variable, so the SPR is quasi-convex.

A.2. Time complexity study. During the first step of our method, in which the SPR term and its (sub)gradient have to be computed, an extra computational cost is incurred w.r.t. the standard "simple" regularizations; note that this does not happen during the fine-tuning phase, where the standard ridge/Tikhonov (ℓ_2) regularization is used instead. The impact of the SPR term is shown Table 15, which compares the cost per epoch with and without the SPR regularization. For easier data sets (small input size), our regularization term roughly

Table 14: Results of state-of-the-art method on ImageNet using ResNet-18

Method	Setting	top1	top5	Pruned pars (%)	
EPFS	F-0.05	67.81	88.37	3690000	(34.60)
HFP	0.20	69.15	N/A	2354869	(22.07)
	0.35	68.53	N/A	3976709	(37.27)
SCOP	A	69.18	88.89	4593978	(39.30)
	B	68.62	88.45	5084938	(43.50)
SPR	λ 0.75 - α 0.1	70.26	89.66	1992131	(17.04)
	λ 1.0 - α 0.1	69.27	89.06	3811382	(32.61)
	λ 1.1 - α 0.1	68.87	88.72	4481715	(38.34)

doubles the cost per epoch, while for the hardest data set (more relevant to real applications) the two costs are almost the same, which proves that our approach is, generally speaking, computationally viable.

Table 15: Average computation times (seconds) for one epoch with and without the SPR term

Architecture and data set	time SPR	time without SPR
ResNet-20 on CIFAR-10	13.05	6.51
ResNet-56 on CIFAR-10	36.58	16.99
ResNet-20 on CIFAR-100	22.99	11.26
ResNet-18 on ImageNet	2,433.14	2,401.05

A.3. Detail on grid search. As we stated in the first paragraph of Section 3, α and λ hyperparameters are found through adaptive grid search. We tested 36 pairs with $\lambda \in [0.1, 3.0]$ and $\alpha \in [1\text{e-}4, 0.6]$ for all the experiments with the Cifar-10 dataset. For the Cifar-100 experiments, the intervals for λ and α were kept the same and 70 pairs were tested. Finally, we used 12 pairs with $\lambda \in [0.5, 1.2]$ and $\alpha \in [1\text{e-}1, 0.6]$ for the experiments with the Imagenet dataset.

Finally, we report an observation on the importance of the fine-tuning phase. From Table 16, we can see that this step is crucial when the pruning caused a significant accuracy drop, while is less relevant (as one could expect) when the accuracy remains high despite the pruning.

A.4. Pareto curve. In Figure 3 we plot all the accuracy-sparsity pairs obtained with our experiments using the ResNet-20 model on the Cifar-10 dataset. Although the curve is not fully complete, it gives a good insight on how pruning affect the accuracy of the model.

A.5. Observation on the structure of the pruned network. From the experiments, we noticed that our algorithm heavily prunes the last layers of the network. This is due to the

Table 16: Accuracy before and after the fine-tuning phase (ResNet-18 on CIFAR.10)

λ	α	Accuracy before	Accuracy after
1.1	0.01	82.40	85.56
1.7	0.30	85.28	87.33
1.1	0.30	88.22	89.47
0.5	0.30	90.62	91.23
0.2	0.30	92.46	92.69
Original model		92.03	-

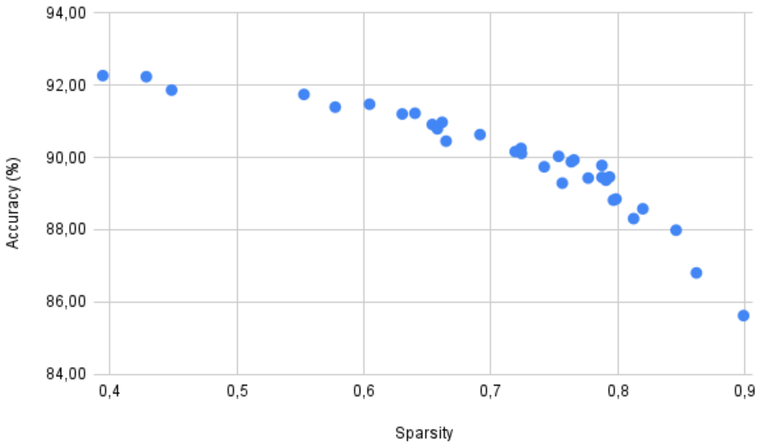


Figure 3: Pareto curve for ResNet-20 on Cifar-10. Different points correspond to different values of α and λ.

fact that the gain in sparsity is larger for these last layers, since their filters contain way more parameters than those belonging to the earliest layers. When the hyperparameters favor heavy pruning even at the cost of a consistent accuracy drop, or when the model is so over-parametrized that even pruning many of parameters only slightly affects the accuracy, basically all final layers are fully pruned. When, instead, less parameters are pruned then the final layers that are not fully pruned tend to be always the same for different configurations of the hyperparameters: for example, for ResNet-18 on ImageNet, the layer with the last residual connection is almost never pruned. This indicates that our pruning approach is successful in identifying the essential structures of the model that need be retained.

A.6. Results in the unstructured setting. As mentioned in the main body of this work, to effectively reduce the computational endeavor of GPU computations through pruning, it is necessary to remove entire structures of the network. However, we acknowledge that unstructured pruning retains its relevance in certain contexts and enables cleaner comparisons with other methods. Consequently, we have chosen to include results within the unstructured pruning setting to provide a comprehensive perspective, although it is important to note that the primary emphasis of this study lies in the structured pruning scenario.

When the prunable entities E_i described in (3.7) consist of singletons, the SPR term exhibits a strong resemblance to the Berhu regularization. While the Berhu regularization has found successful application in robust regression [39], its performance in the context of pruning remains unexplored. In the following, we present numerical results pertaining to unstructured pruning scenarios involving ResNet-32 and ResNet-56, on the Cifar10 dataset.

We compare our results with two baseline methods that use regularization to prune Neural Networks and with one relevant literature method. The first baseline method is the simple ℓ_1 regularization, known to produce sparser networks compared to the conventional ℓ_2 squared regularization. The second one is the well-known Elastic Net [81], which uses a linear combination of ℓ_1 and ℓ_2 squared regularizations. Formally, the utilization of ℓ_1 regularization yields the following optimization problem:

$$\min L(X, W) + \lambda \|W\|_1.$$

While the Elastic Net problem is defined by

$$\min L(X, W) + \lambda[\alpha \|W\|_2^2 + (1 - \alpha)\|W\|_1].$$

As the ℓ_1 regularization can be regarded as a limit case of the Elastic Net with the specific parameter α set to 0, we have aggregated their outcomes in the next section for the sake of conciseness and clarity.

Moreover, we performed a comparative evaluation alongside a more complex state-of-the-art technique developed in [9]. This method, although originating from an optimization problem akin to (3.1)-(3.3), subsequently integrates alternating learning and compression phases to systematically achieve pruning in the Neural Network.

We directly report the results from [9], while for all the other methods under comparison, we conducted a systematic grid search, following a similar configuration as detailed in Section 4.1. In Tables 17 and 18, we report only the most relevant non-dominated results of the grid search.

Tables 17 and 18 present clear evidence of SPR's superiority over the baseline methods. Our approach achieves a reduction of over 90% in the number of parameters for ResNet-32 and nearly 94% for ResNet-56, while maintaining an accuracy of over 92% for both architectures. Notably, ℓ_1 regularization competes closely with Elastic Net when applied to ResNet-32 pruning, producing results that are non-dominated and reported in Table 17. Conversely, when pruning ResNet-56, Elastic Net consistently outperforms ℓ_1 regularization, occasionally achieving results that are competitive with SPR. Regarding the comparison with [9], the outcomes presented in Tables 17 and 18 highlight that, despite its relative simplicity compared to the competition, our approach remains competitive within the existing literature. Notably, when pruning ResNet-32, we successfully remove more than 90% of the parameters while achieving nearly identical accuracy compared to the state-of-the-art method that prunes exactly 90% of the network. However, our results are less favorable when pruning ResNet-56. This suggests

Table 17: Result on CIFAR-10 using ResNet-32 in the unstructured setting.

Method	Setting	Acc.	Pruned pars	(%)
Elastic Net	λ 15 - α 0.2	92.73	334791	(72.48)
	λ 20 - α 0	92.05	369338	(79.96)
	λ 25 - α 0	91.31	384277	(83.20)
	λ 35 - α 1e-2	90.35	413674	(89.56)
[9]	P-15	92.68	392601	(85.00)
	P-10	92.12	415694	(90.00)
	P-5	90.74	438788	(95.00)
	P-3	89.26	448025	(97.00)
SPR	λ 10 - α 5e-2	93.14	349601	(75.69)
	λ 25 - α 0.2	92.46	405541	(87.80)
	λ 10 - α 0.8	92.11	416428	(90.16)
	λ 35 - α 0.2	90.85	436795	(94.57)
	λ 35 - α 0.6	89.95	441673	(95.62)

Table 18: Results on CIFAR-10 using ResNet-56 in the unstructured setting.

Method	Setting	Acc.	Pruned pars	(%)
Elastic Net	λ 20 - α 0.6	93.41	604445	(70.86)
	λ 20 - α 5e-2	93.22	693854	(81.34)
	λ 25 - α 5e-2	92.77	727884	(85.33)
	λ 35 - α 5e-2	91.75	751298	(88.08)
[9]	P-15	93.08	725676	(85.00)
	P-10	93.33	768123	(90.00)
	P-5	92.49	810570	(95.00)
	P-3	91.79	827549	(97.00)
SPR	λ 30 - α 1e-2	93.90	631012	(73.97)
	λ 20 - α 5e-2	92.94	736483	(86.34)
	λ 25 - α 0.2	92.14	799059	(93.67)
	λ 25 - α 0.6	91.34	806005	(94.49)

that employing a more complex optimization algorithm may be crucial for larger architectures or that further hyper-parameter tuning is needed in such scenarios.

Appendix B. Discussion on the M hyper-parameter. In this section, we discuss the importance of the M parameter appearing in the SPR definition and some considerations surrounding its selection.

The value of M is used when projecting away the y variables in (3.4), and it conveys important information for the SPR. As partially explained in Section 3.3, the M parameter is used to assess if a weight is "large" or not: indeed, the SPR term changes its form based on the quantity $\|w\|/M$.

Ideally, the value of M could be chosen such that the weights will *naturally* stay below such value. In practice, this ideal M is not computable and we had to choose M empirically as explained in Section 4.1. It is crucial to grasp that opting for an excessively large M is detrimental. Intuitively, this is due to the previously mentioned SPR mechanism that dynamically adapts the definition of the SPR term based on the value of M. Theoretically, it is well documented in the MIP literature that, in formulations that contain constraints such as (3.2), an excessively large M value has a rather negative effect on the quality of the continuous relaxation of the MIP formulation [8]. This continuous relaxation forms the foundation of our approach and it is what we aim to strengthen when using the Perspective function in Section 3.1. The practical irrelevance of an excessively large value for M becomes evident when considering the limit where M approaches infinity. In fact, in this limit, the SPR term essentially converges to being almost identical to the ℓ_2 norm.

Acknowledgments. This work has been supported by the NSERC Alliance grant 544900-19 in collaboration with Huawei-Canada. The authors are repeatedly indebted to two anonymous referees for their careful reading and useful remarks.

REFERENCES

[1] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg, *Net-trim: Convex pruning of deep neural networks with performance guarantee*, Advances in neural information processing systems, 30 (2017).
[2] A. Aghasi, A. Abdi, and J. Romberg, *Fast convex pruning of deep neural networks*, SIAM Journal on Mathematics of Data Science, 2 (2020), pp. 158–188.
[3] J. M. Alvarez and M. Salzmann, *Compression-aware training of deep networks*, Advances in neural information processing systems, 30 (2017).
[4] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma, *Strong mixed-integer programming formulations for trained neural networks*, Mathematical Programming, 183 (2020), pp. 3–39.
[5] A. Atamtürk and A. Gómez, *Safe screening rules for l0-regression from perspective relaxations*, in Proceedings of the 37th International Conference on Machine Learning, 2020, pp. 421–430.
[6] C. Baykal, L. Liebenwein, I. Gilitschenski, D. Feldman, and D. Rus, *Sensitivity-informed provable pruning of neural networks*, SIAM Journal on Mathematics of Data Science, 4 (2022), pp. 26–45.
[7] G. Bellec, D. Kappel, W. Maass, and R. Legenstein, *Deep rewiring: Training very sparse deep networks*, in International Conference on Learning Representations, 2018, https://openreview.net/forum?id=BJ_wN01C-.
[8] P. Bonami, A. Lodi, A. Tramontani, and S. Wiese, *On mathematical programming with indicator constraints*, Mathematical programming, 151 (2015), pp. 191–223.
[9] M. A. Carreira-Perpinan and Y. Idelbayev, *"learning-compression" algorithms for neural net pruning*, in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 8532–8541, https://doi.org/10.1109/CVPR.2018.00890.
[10] J. Chee, M. Renz, A. Damle, and C. D. Sa, *Model preserving compression for neural networks*, in Advances in Neural Information Processing Systems, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds., 2022, https://openreview.net/forum?id=gt-l9Hu2ndd.
[11] T. Chen, B. Ji, T. Ding, B. Fang, G. Wang, Z. Zhu, L. Liang, Y. Shi, S. Yi, and X. Tu, *Only train once: A one-shot neural network training and pruning framework*, in NeurIPS, 2021.

[12] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, *Towards efficient model compression via learned global ranking*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 1518–1528.

[13] X. Ding, g. ding, X. Zhou, Y. Guo, J. Han, and J. Liu, *Global sparse momentum sgd for pruning very deep neural networks*, in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper/2019/file/f34185c4ca5d58e781d4f14173d41e5d-Paper.pdf.

[14] H. Dong, K. Chen, and J. Linderoth, *Regularization vs. relaxation: A convexification perspective of statistical variable selection*, 2018.

[15] X. Dong, S. Chen, and S. Pan, *Learning to prune deep neural networks via layer-wise optimal brain surgeon*, Advances in Neural Information Processing Systems, 30 (2017).

[16] M. ElAraby, G. Wolf, and M. Carvalho, *Oamip: Optimizing ann architectures using mixed-integer programming*, in Integration of Constraint Programming, Artificial Intelligence, and Operations Research, A. A. Cire, ed., Cham, 2023, Springer Nature Switzerland, pp. 219–237.

[17] L. Enderich, F. Timm, and W. Burgard, *Holistic filter pruning for efficient deep neural networks*, in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 2596–2605.

[18] M. Fischetti and J. Jo, *Deep neural networks and mixed integer linear optimization*, Constraints, 23 (2018), pp. 296–309.

[19] A. Frangioni, F. Furini, and C. Gentile, *Approximated perspective relaxations: a project&lift approach*, Computational Optimization and Applications, 63 (2016), pp. 705–735.

[20] A. Frangioni and C. Gentile, *Perspective cuts for a class of convex 0–1 mixed integer programs*, Mathematical Programming, 106 (2006), pp. 225–236.

[21] A. Frangioni, C. Gentile, E. Grande, and A. Pacifici, *Projected perspective reformulations with applications in design problems*, Operations Research, 59 (2011), pp. 1225–1232.

[22] J. Frankle and M. Carbin, *The lottery ticket hypothesis: Finding sparse, trainable neural networks*, in International Conference on Learning Representations, 2019, https://openreview.net/forum?id=rJl-b3RcF7.

[23] E. Frantar and D. Alistarh, *Optimal brain compression: A framework for accurate post-training quantization and pruning*, in Advances in Neural Information Processing Systems, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds., 2022, https://openreview.net/forum?id=ksVGCOlOEba.

[24] E. Frantar and D. Alistarh, *SPDY: Accurate pruning with speedup guarantees*, in Proceedings of the 39th International Conference on Machine Learning, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, eds., vol. 162 of Proceedings of Machine Learning Research, PMLR, 17–23 Jul 2022, pp. 6726–6743, https://proceedings.mlr.press/v162/frantar22a.html.

[25] N. Gamboa, K. Kudrolli, A. Dhoot, and A. Pedram, *Campfire: Compressible, regularization-free, structured sparse training for hardware accelerators*, arXiv preprint arXiv:2001.03253, (2020).

[26] N. S. Guang-Bin Huang, P. Saratchandran, *A Generalized Growing and Pruning RBF (GGAP-RBF) Neural Network for Function Approximation*, IEEE TRANSACTIONS ON NEURAL NETWORKS, 16 (2005), pp. 57–67.

[27] M. E. Halabi, S. Srinivas, and S. Lacoste-Julien, *Data-efficient structured pruning via submodular optimization*, in Advances in Neural Information Processing Systems, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds., 2022, https://openreview.net/forum?id=K2QGzyLwpYG.

[28] S. Han and B. Dally, *Efficient methods and hardware for deep learning*, University Lecture, (2017).

[29] S. Han, J. Pool, J. Tran, and W. Dally, *Learning both weights and connections for efficient neural network*, in Advances in Neural Information Processing Systems 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds., Curran Associates, Inc., 2015, pp. 1135–1143, http://papers.nips.cc/paper/5784-learning-both-weights-and-connections-for-efficient-neural-network.pdf.

[30] B. Hassibi and D. G. Stork, *Second order derivatives for network pruning: Optimal brain surgeon*, in Advances in Neural Information Processing Systems 5, S. J. Hanson, J. D. Cowan, and C. L. Giles, eds., Morgan-Kaufmann, 1993, pp. 164–171, http://papers.nips.cc/paper/647-second-order-derivatives-for-network-pruning-optimal-brain-surgeon.pdf.

[31] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of

the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[32] Z. Huang and N. Wang, *Data-driven sparse structure selection for deep neural networks*, in Computer Vision–ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI 15, Springer, 2018, pp. 317–334.
[33] P. J. Huber, *Robust estimation of a location parameter*, Annals of Mathematical Statistics, 35 (1964), pp. 73–101, https://doi.org/10.1214/aoms/1177703732.
[34] E. Karnin, *A simple procedure for pruning back-propagation trained neural networks*, IEEE Transactions on Neural Networks, 1 (1990), pp. 239–242, https://doi.org/10.1109/72.80236.
[35] E. Karnin, *Simple procedure for pruning back-propagation trained neural networks*, Neural Networks, IEEE Transactions on, 1 (1990), pp. 239 – 242, https://doi.org/10.1109/72.80236.
[36] A. Krizhevsky, *Learning multiple layers of features from tiny images*, tech. report, (canadian institute for advanced research), 2009.
[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Advances in Neural Information Processing Systems, 2012.
[38] S. Kundu, M. Nazemi, P. A. Beerel, and M. Pedram, *Dnr: A tunable robust pruning framework through dynamic network rewiring of dnns*, in Proceedings of the 26th Asia and South Pacific Design Automation Conference, ASPDAC '21, New York, NY, USA, 2021, Association for Computing Machinery, p. 344–350, https://doi.org/10.1145/3394885.3431542, https://doi.org/10.1145/3394885.3431542.
[39] S. Lambert-Lacroix and L. Zwald, *The adaptive berhu penalty in robust regression*, Journal of Nonparametric Statistics, 28 (2016), pp. 487–514.
[40] V. Lebedev and V. Lempitsky, *Fast convnets using group-wise brain damage*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2554–2564.
[41] E. Lee and C.-Y. Lee, *Neuralscale: Efficient scaling of neurons for resource-constrained deep neural networks*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1478–1487.
[42] N. Lee, T. Ajanthan, and P. Torr, *SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY*, in International Conference on Learning Representations, 2019, https://openreview.net/forum?id=B1VZqjAcYX.
[43] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, *Extremely low bit neural network: Squeeze the last bit out with admm*, Proceedings of the AAAI Conference on Artificial Intelligence, 32 (2018), https://doi.org/10.1609/aaai.v32i1.11713, https://ojs.aaai.org/index.php/AAAI/article/view/11713.
[44] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, *Pruning filters for efficient convnets*, in International Conference on Learning Representations, 2017, https://openreview.net/forum?id=rJqFGTslg.
[45] Y. Li, S. Gu, K. Zhang, L. V. Gool, and R. Timofte, *Dhp: Differentiable meta pruning via hypernetworks*, in European Conference on Computer Vision, Springer, 2020, pp. 608–624.
[46] L. Liebenwein, C. Baykal, H. Lang, D. Feldman, and D. Rus, *Provable filter pruning for efficient neural networks*, in International Conference on Learning Representations, 2020, https://openreview.net/forum?id=BJxkOlSYDH.
[47] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, *Hrank: Filter pruning using high-rank feature map*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1529–1538.
[48] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, *Toward compact convnets via structure-sparsity regularized filter pruning*, IEEE transactions on neural networks and learning systems, 31 (2019), pp. 574–588.
[49] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi, *Dynamic model pruning with feedback*, in International Conference on Learning Representations, 2020, https://openreview.net/forum?id=SJem8lSFwB.
[50] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, *Learning efficient convolutional networks through network slimming*, in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2736–2744.
[51] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, *Rethinking the value of network pruning*, in International Conference on Learning Representations, 2019, https://openreview.net/forum?id=rJlnB3C5Ym.

[52] A. Lodi, M. Toma, and R. Guerrieri, *Very low complexity prompted speaker verification system based on hmm-modeling*, Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on, 4 (2002), https://doi.org/10.1109/ICASSP.2002.5745512.
[53] C. Louizos, K. Ullrich, and M. Welling, *Bayesian compression for deep learning*, Advances in neural information processing systems, 30 (2017).
[54] C. Louizos, M. Welling, and D. P. Kingma, *Learning sparse neural networks through l_0 regularization*, in International Conference on Learning Representations, 2018, https://openreview.net/forum?id=H1Y8hhg0b.
[55] Y. Ma, R. Chen, W. Li, F. Shang, W. Yu, M. Cho, and B. Yu, *A unified approximation framework for compressing and accelerating deep neural networks*, in 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2019, pp. 376–383.
[56] D. Molchanov, D. Vetrov, and A. Ashukha, *Variational dropout sparsifies deep neural networks*, in 34th International Conference on Machine Learning, ICML 2017, 2017, pp. 3854–3863.
[57] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, *Importance estimation for neural network pruning*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11264–11272.
[58] 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-Conference Track Proceedings, 2019.
[59] A. B. Owen, *A robust hybrid of lasso and ridge regression*, Contemporary Mathematics, (2007), http://finmath.stanford.edu/~owen/reports/hhu.pdf.
[60] J. Rachwan, D. Zügner, B. Charpentier, S. Geisler, M. Ayle, and S. Günnemann, *Winning the lottery ahead of time: Efficient early network pruning*, in International Conference on Machine Learning, PMLR, 2022, pp. 18293–18309.
[61] R. K. Ramakrishnan, E. Sari, and V. P. Nia, *Differentiable mask for pruning convolutional and recurrent networks*, in 2020 17th Conference on Computer and Robot Vision (CRV), IEEE, 2020, pp. 222–229.
[62] T. Serra, A. Kumar, and S. Ramalingam, *Lossless compression of deep neural networks*, in Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21–24, 2020, Proceedings, Springer, 2020, pp. 417–430.
[63] M. Shen, H. Yin, P. Molchanov, L. Mao, J. Liu, and J. M. Alvarez, *Structural pruning via latency-saliency knapsack*, in Advances in Neural Information Processing Systems, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds., 2022, https://openreview.net/forum?id=cUOR-_VsavA.
[64] X. Sheng, C. Hanlin, G. Xuan, L. Kexin, L. Jinhu, and Z. Baochang, *Efficient structured pruning based on deep feature stabilization*, Neural Computing and Applications, 1433-3058 (2021), https://doi.org/10.1007/s00521-021-05828-8.
[65] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, in International Conference on Learning Representations, 2015.
[66] Y. Sui, M. Yin, Y. Xie, H. Phan, S. Aliari Zonouz, and B. Yuan, *Chip: Channel independence-based pruning for compact neural networks*, Advances in Neural Information Processing Systems, 34 (2021), pp. 24604–24616.
[67] Y. Tang, Y. Wang, Y. Xu, D. Tao, C. XU, C. Xu, and C. Xu, *Scop: Scientific control for reliable neural network pruning*, in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., vol. 33, Curran Associates, Inc., 2020, pp. 10936–10947, https://proceedings.neurips.cc/paper/2020/file/7bcdf75ad237b8e02e301f4091fb6bc8-Paper.pdf.
[68] M.-R. Vemparala, N. Fasfous, A. Frickenstein, M. A. Moraly, A. Jamal, L. Frickenstein, C. Unger, N.-S. Nagaraja, and W. Stechele, *L2pf-learning to prune faster*, in Computer Vision and Image Processing: 5th International Conference, CVIP 2020, Prayagraj, India, December 4-6, 2020, Revised Selected Papers, Part III, Springer, 2021, pp. 249–261.
[69] Y. Wang, X. Zhang, L. Xie, J. Zhou, H. Su, B. Zhang, and X. Hu, *Pruning from scratch*, Proceedings of the AAAI Conference on Artificial Intelligence, 34 (2020), pp. 12273–12280, https://doi.org/10.1609/aaai.v34i07.6910, https://ojs.aaai.org/index.php/AAAI/article/view/6910.
[70] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, *Learning structured sparsity in deep neural networks*,

Advances in neural information processing systems, 29 (2016).

[71] X. XIAO, Z. Wang, and S. Rajasekaran, *Autoprune: Automatic network pruning by regularizing auxiliary parameters*, in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper/2019/file/4efc9e02abdab6b6166251918570a307-Paper.pdf.

[72] H. Yang, W. Wen, and H. Li, *Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures*, in International Conference on Learning Representations, 2020.

[73] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, *Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers*, in 6th International Conference on Learning Representations, ICLR 2018, 2018.

[74] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, *Nisp: Pruning networks using neuron importance score propagation*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 9194–9203.

[75] S. Yu, A. Mazaheri, and A. Jannesari, *Topology-aware network pruning using multi-stage graph embedding and reinforcement learning*, in International Conference on Machine Learning, PMLR, 2022, pp. 25656–25667.

[76] X. Yu, T. Serra, S. Ramalingam, and S. Zhe, *The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks*, in Proceedings of the 39th International Conference on Machine Learning, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, eds., vol. 162 of Proceedings of Machine Learning Research, PMLR, 17–23 Jul 2022, pp. 25668–25683, https://proceedings.mlr.press/v162/yu22f.html.

[77] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, *A systematic dnn weight pruning framework using alternating direction method of multipliers*, in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 184–199.

[78] Y. Zhang, Y. Yao, P. Ram, P. Zhao, T. Chen, M. Hong, Y. Wang, and S. Liu, *Advancing model pruning via bi-level optimization*, in Advances in Neural Information Processing Systems, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds., 2022, https://openreview.net/forum?id=t6O08FxvtBY.

[79] Y. Zhou, Y. Zhang, Y. Wang, and Q. Tian, *Accelerate cnn via recursive bayesian pruning*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3306–3315.

[80] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, *Discrimination-aware channel pruning for deep neural networks*, Advances in neural information processing systems, 31 (2018).

[81] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, Journal of the Royal Statistical Society. Series B, 67 (2005), pp. 301–320, https://doi.org/10.1111/j.1467-9868.2005.00503.x.