OPT2021: 13th Annual Workshop on Optimization for Machine Learning

Deep Neural Networks pruning via the Structured Perspective Regularization

Matteo Cacciola MATTEO.CACCIOLA@POLYMTL.CA
CERC, Polytechnique Montreal, Montreal, Canada

Antonio Frangioni FRANGIO@DI.UNIPI.IT
University of Pisa, Pisa, Italy

Xinlin Li XINLIN.LI1@HUAWEI.COM
Huawei Montreal Research Centre,Montreal, Canada

Andrea Lodi ANDREA.LODI@POLYMTL.CA
CERC, Polytechnique Montreal, Montreal, Canada

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, etc.) 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 Operations 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.

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 [14, 19, 23]. While one may just resort to smaller models, the fact that a large mode trained even for a few epochs performs better than smaller ones trained for much longer lends credence to the claim [21] 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.

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 [13] that obtained unexpectedly good results from a trivial magnitude-based approach. The same magnitude-based approach was extended in [12] with a re-training phase where the non-pruned weight are re-initialised to their starting values.

From the structured pruning side, a 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. In [24] 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.

MIP techniques have been successfully used in the ANN context, but mostly in applications unrelated to pruning like construction of adversarial examples (with fixed weights) [8]. In [3], the approach is extended to a larger class of activation functions and stronger formulations are defined. An exception is [6], where a score function is defined to asses 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.

The link between Perspective Reformulation techniques and sparsification has been recently recognized [4, 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 where the set of parameters $W = \{ w_j \,|\, j \in I \}$ is partitioned in N subsets, that we call prunable entities, such that $W = \cup_{i=1}^{N} E_i$ 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 is possible only if $w_j = 0$ for all $j \in E_i$. We should consider a three-objective optimization problem where we: i) minimize the loss, ii) minimize some standard regularization term aiming at 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 a scaling of the three objective functions by means of hyperparameters whose optimal values are found by standard grid-search techniques.

Starting from the usual ML framework with ℓ_2 regularization, we consider the following MIP

$$\min\ L(X,W) + \lambda[\alpha \textstyle\sum_{j\in I} w_j^2 + (1-\alpha)\sum_{i=1}^{N} y_i] \tag{1}$$

$$-My_i \le w_j \le My_i \qquad \forall w_j \in E_i \quad i = 1,..,N \tag{2}$$

$$y_i \in \{0,1\} \qquad i = 1,..,N \tag{3}$$

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 (2) ensure that if $y_i = 0$ then $0 \le w_j \le 0$ 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). In other words, the term "$\sum_{i=1}^{N} y_i$" represents the ℓ_0 norm of the structured set of weights, i.e., the cardinailty of the set $\{ i \in \{1,\dots,N\} : \exists j \in E_i \text{ s.t. } w_j \neq 0\}$. In general, solving (1)–(3) directly is not computationally efficient, and therefore a standard strategy is to consider its *continuous relaxation* whereby (3) is relaxed to $y_i \in [0,1]$. If $L(\cdot)$ is convex, as in our case, this is easily solvable but its solution in both the y and w variables, which is what one could use to perform the pruning, can be rather different from the optimal solution of (1)–(3), therefore leading to inefficient prunings. To improve on that, we consider the Perspective Reformulation [11]

$$\min \Big\{ L(X,W) + \lambda[\alpha \textstyle\sum_{i=1}^{N} \sum_{j\in E_i} w_j^2/y_i + (1-\alpha)\sum_{i=1}^{N} y_i] \ :\ (2)\ ,\ (3) \Big\} \tag{4}$$

of problem (1)–(3). It can be shown that (4) has the same integer optimal solution as the original problem, but its continuous relaxation (the Perspective Relaxation) is "better" in a well-defined mathematical sense: its optimal objective value is (much) closer to the true optimal value of (1)–(3), which typically implies that its optimal solution is more similar to the true optimal solution.

While one can expect that the solution of the relaxation of (4) can provide a better guide to the pruning procedure, its form makes it more difficult to apply standard training techniques to solve it. Following the lead of [9, 10], we then proceed at simplifying the model by projecting away the y variables, the details about this step are reported in the appendix. All in all, we can rewrite the continuous relaxation of (4) as

$$\min \Big\{ L(X,W) + \lambda \textstyle\sum_{i=1}^{N} z_i(w_i;\alpha,M) \Big\}, \tag{5}$$

where $w^i = [\, w_j \,]_{j \in E_i}$ and

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

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 (4), we can expect (with a proper choice of the hyperparameters) 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, which means that it is zero only if all the components of w_i are zero. In other words, while using the, say, ordinary ℓ_1 would promote sparsity on each weight individually, the SPR can be expected to better promote structured sparsity as required by our application. However, the solution of (5) can be sought for with all of the usual algorithms for training ANNs (SGD, Adam, etc.), and therefore should not, in principle, be more costly then non-sparsity-inducing training.

The above mentioned solution scheme still needs some minor fixes and improvements to fully adapt to our scenario, details about these changes can be found in the appendix,

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. The code used to run the experiments was written starting from the public repository [1] and [2].

For our experiments, we used 3 very popular datasets: CIFAR-10 [17], CIFAR-100 [17] and ImageNet [18]. As architectures, we focused on resnet [15] and, in particular, we used resnet-20 and ResNet-56 for both CIFAR datasets and resnet-18 for the ImageNet dataset.

For all the experiments, we used Pytorch (1.7.1 and 1.8.1) with Cuda, the CrossEntropyLoss and the SGD optimizer with 0.9 momentum.

After the first phase, we considered as pruned all weights with absolute value lower than 1e-4 and we decided to prune all entities with more than 95% of parameters pruned.

To get the values of M_i's, we used 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 the pretrained version available from torchvision).

Further details on the hardware that were used, general settings and detailed results tables can be found in the appendix for each of the datasets.

As shown in Table 8, training ResNet-20 on CIFAR-10, we were able to prune more than 23% of the parameters by still increasing the accuracy of the original model, while we can prune almost 70% of the model by still preserving more than 90% accuracy.

Using ResNet-20 on the CIFAR-100 dataset, we can again prune more than 15% of parameters while improving the accuracy of the original model (Table 9). Due to the fact that CIFAR-100 is

more challenging than CIFAR-10, it was not possible to prune very many parameters without losing a significant amount of accuracy: we can still achieve more than 68% accuracy by pruning a few more than 30% of the parameters, but accuracy drop to less than 67% if we try to prune more.

Finally, Table 10 reports results of training the ResNet-56 architecture on CIFAR-10: once again pruning about 30% of the parameters improves accuracy and we can keep more than 93% accuracy while pruning more than a half of the network.

Results on ImageNet using ResNet-18, Table 11, show that even in a very large and difficult dataset our method is able to improve the original model results by a consistent margin, reaching almost 71% accuracy while pruning more than 6% of the parameters. Pruning more than 50% of the network causes a drop of 2.5% in the accuracy, while a way more consistent decrease happens when we prune about 80% of the parameters.

4.1. 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 [16] (denoted by SSS), [25] (denoted by EPFS), [26] (denoted by L2PF), [20] (denoted by PFFEC), [22] (denoted as HRANK) and [7] (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 [16], 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 [16], we denote the points as P1, P2, etc.). For ImageNet the top5 accuracy is not reported in [7], so we marked the corresponding field in our table with a “N/A". Finally, we report results for different settings of each method since they were present in the original papers; however, it should be remarked that not all of them are filter pruning methods, some rather being general structured pruning ones.

Regarding ResNet-20 on CIFAR-10, Table 1, our results outperform the other methods in most of the cases, meaning that we can reach equal or better accuracy while pruning a larger amount of parameters. Whenever we do not outperform the other methods we have comparable performances and we can have a better accuracy or sparsity maintaining the other metric close. This is for instance the case of L2PF, that achieves 89.9% accuracy with 73.96% sparsity, while we achieve a little higher sparsity (74.00%) loosing a little accuracy (89.47%).

On CIFAR-100 using ResNet-20, Table 2, we clearly outperform SSS, as we can achieve more than 68.3% accuracy wile pruning more than 30% of parameters while SSS could prune only 14.81% to obtain a little bit more than 67% accuracy. In Table 3 we can observe a similar situation to ResNet-20 on CIFAR-10 for ResNet-56 on the same dataset. The only result we did not outperform was the HPF 93.30 accuracy with 50% sparsit,y but we can obtain a little bit more sparsity (54.21%) with almost the same accuracy (93.13%).

On ImageNet using ResNet-18, Table 4, we can see that even if our method does not outperform the other ones, we are able to prune consistently more parameters without a very large accuracy drop. Likely some more 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 pruning ANNs, we proposed a new regularization term, based on the projected Perspective Reformulation, designed to promote structured sparsity.

Table 1: Results of state of the art method on CIFAR-10 using ResNet-20.

Method	Setting	Accuracy	Pruned parameters (%)	
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)
SPR	λ1.1-α0.3	89.47	199809	(74.00)
	λ0.5-α0.3	91.23	140535	(52.05)
	λ0.2-α0.3	92.69	64188	(23.77)

Table 2: Results of state of the art method on CIFAR-100 using ResNet-20.

Method	Setting	Accuracy	Pruned parameters (%)	
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	λ1.0-α0.1	66.28	123984	(45.92)
	λ1.3-α0.3	68.36	84960	(31.47)
	λ0.7-α0.3	69.20	42480	(15.73)

Table 3: Results of state of the art method on CIFAR-10 using ResNet-56

Method	Setting	Accuracy	Pruned parameters (%)	
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)
SPR	λ0.5-α0.001	92.55	633960	(74.58)
	λ1.1-α1e-4	93.13	460800	(54.21)
	λ0.8-α0.1	93.98	268128	(31.54)

The proposed method is able to prune any kind of structures and the amount of pruning is 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. The results have shown that our method is competitive.

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 starting 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.

References

[1] URL https://github.com/akamaster/pytorch_resnet_cifar10.

Table 4: Results of state-of-the-art method on ImageNet using ResNet-18

Method	Setting	Accuracy (top1)	Accuracy (top5)	Pruned parameters (%)	
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)
SPR	λ1.3-α0.3	67.26	87.71	5796625	(54.33)

[2] URL https://github.com/pytorch/examples/tree/master/imagenet.

[3] Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks, 2020.

[4] Alper Atamtürk and Andrés Gómez. Safe screening rules for ℓ_0-regression, 2020.

[5] Hongbo Dong, Kun Chen, and Jeff Linderoth. Regularization vs. relaxation: A convexification perspective of statistical variable selection, 2015.

[6] Mostafa ElAraby, Guy Wolf, and Margarida Carvalho. Identifying critical neurons in ANN architectures using mixed integer programming. *CoRR*, abs/2002.07259, 2020. URL https://arxiv.org/abs/2002.07259.

[7] Lukas Enderich, Fabian Timm, and Wolfram Burgard. Holistic filter pruning for efficient deep neural networks. *CoRR*, abs/2009.08169, 2020. URL https://arxiv.org/abs/2009.08169.

[8] Matteo Fischetti and Jason Jo. Deep neural networks as 0-1 mixed integer linear programs: A feasibility study. *CoRR*, abs/1712.06174, 2017. URL http://arxiv.org/abs/1712.06174.

[9] A. Frangioni, C. Gentile, E. Grande, and A. Pacifici. Projected perspective reformulations with applications in design problems. *Operations Research*, 59(5):1225–1232, 2011.

[10] A. Frangioni, F. Furini, and C. Gentile. Approximated perspective relaxations: a project&lift approach. *Computational Optimization and Applications*, 63(3):705–735, 2016.

[11] Antonio Frangioni and Claudio Gentile. Perspective cuts for a class of convex 0–1 mixed integer programs. *Mathematical Programming*, 106(2):225–236, 2006.

[12] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[13] Song Han and B Dally. Efficient methods and hardware for deep learning. *University Lecture*, 2017.

[14] 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. URL http://papers.nips.cc/paper/

5784-learning-both-weights-and-connections-for-efficient-neural-network.pdf.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[16] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. *CoRR*, abs/1707.01213, 2017. URL http://arxiv.org/abs/1707.01213.

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, (canadian institute for advanced research), 2009.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[19] Cong Leng, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with ADMM. *CoRR*, abs/1707.09870, 2017. URL http://arxiv.org/abs/1707.09870.

[20] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016. URL http://arxiv.org/abs/1608.08710.

[21] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *CoRR*, abs/2002.11794, 2020. URL https://arxiv.org/abs/2002.11794.

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

[23] Andrea Lodi, Mario Toma, and Roberto 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, 01 2002. doi: 10.1109/ICASSP.2002.5745512.

[24] Ramchalam Kinattinkara Ramakrishnan, Eyyüb Sari, and Vahid Partovi Nia. Differentiable mask pruning for neural networks. *arXiv preprint arXiv:1909.04567*, 2019.

[25] Xu Sheng, Chen Hanlin, Gong Xuan, Liu Kexin, Lü Jinhu, and Zhang Baochang. Efficient structured pruning based on deep feature stabilization. *Neural Computing and Applications*, 1433-3058, 2021. URL https://doi.org/10.1007/s00521-021-05828-8.

[26] Manoj Rohit Vemparala, Nael Fasfous, Alexander Frickenstein, Mhd Ali Moraly, Aquib Jamal, Lukas Frickenstein, Christian Unger, Naveen Shankar Nagaraja, and Walter Stechele. L2PF - learning to prune faster. *CoRR*, abs/2101.02663, 2021. URL https://arxiv.org/abs/2101.02663.

Appendix A.

A.1. Eliminating the y variables

To remove these variables we compute a closed formula $\tilde{y}(w)$ for the optimal value of the y variables in the continuous relaxation of (4) assuming that w are the optimal weights. When w is fixed, the problem decomposes over the subsets, and therefore we only need to consider each fragment

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

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

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

(we are only interested in positive y) and then project it on the domain. Note that, technically, $f_i(W, y_i)$ is nondifferentiable for $y_i = 0$ but that value is only achieved when $w_j = 0$ for all $j \in E_i$, 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

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

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

A.2. Final improvements of the algorithm

Remarkably, the SPR depends on the choice of M, which is, in principle, nontrivial. Indeed, all previous attempts of using PR techniques for promoting sparsity [4, 5] have been using the "abstract" nonlinear form $(1 - y_i)w_i = 0$ of (2) (assuming $E_i = \{ i \}$ as in those treatments). 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 decided to compute a different constant for each layer of the network and then use it for all entities belonging to that layer (see below).

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 by scaling the term corresponding to entity E_i by $u_i / \sum_{j=1}^{N} u_j$, u_i being 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 perform the pruning (eliminating all the entities where $||w^i||_\infty$ is smaller than a given tolerance (see below) and then we re-train the pruned 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.

Appendix B.

B.1. Time complexity study

Our method is composed of two steps, in the first one the computation of an additional regularization term is required, while the second one is a fine-tuning phase that is common to many other pruning methods and is just a normal backpropagation. So, we can do a comparison between the cost of an epoch in the first step and in the second one to have a "worst case scenario" of a comparison with other methods that use a regularization term to prune. From Table 5, we can notice that for easier data sets (small input size) the cost of one epoch with our regularization term is roughly the double of a normal one, while for the hardest data set, so in the contest of real applications, this gap decreases and the two costs are almost the same.

Table 5: 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

B.2. Detail on grid search and ablation study

Table 6: Ablation study on hyperparameter (ResNet-56 on CIFAR-10)

λ	α	Accuracy	Pruned parameters
1.7	1e-4	90.12	89.04
1.4	1e-4	88.47	87.09
1.7	0.3	90.59	85.76
0.5	0.3	94.04	6.03

As we stated in the first paragraph of Section 3, α and λ hyperparameters are found through grid search. We performed a different grid search for each of the data set / architecture pair we used: for ResNet20 on CIFAR10, we had $\lambda \in [0.2, 0.5, 0.8, 1.1, 1.4, 1.7]$ and $\alpha \in [1e-4, 1e-3, 5e-3, 1e-2, 0.1, 0.3]$; for Resnet56 on CIFAR-10, we had $\lambda \in [0.5, 0.8, 1.1, 1.4, 1.7]$; for Resnet20 on CIFAR-100, we had $\lambda \in [0.1, 0.4, 0.7, 1.0, 1.3, 1.6]$ and $\alpha \in [1e-3, 1e-2, 0.1, 0.3]$; finally for Resnet18 on Imagenet, we had $\lambda \in [0.5, 0.8, 1.0, 1.3]$ and $\alpha \in [1e-3, 1e-2, 0.1, 0.3]$. The λ parameter is just the usual regularization parameter, the α parameter instead tunes how much of the regularization is computed in a "structured" way. From Table 6, we can see that to obtain the maximum level of sparsity λ needs to be high, while α needs to be close to 0, while for lower level of sparsity can be sufficient and sometimes more effective to just increase α. Finally, for very low level of pruning, also λ needs to be decreased.

Finally, we report an observation on the importance of the finetuning phase. From Table 7, we can see that this step is crucial to obtain higher accuracy when the pruning determined a big drop in this value, while is less important when the accuracy stays high despite the pruning.

Table 7: Accuracy before and after the finetuing phase (ResNet18 on CIFAR.10)

λ	α	Accuracy before	Accuracy after
1.1	0.01	82.40	85.56
1.7	0.3	85.28	87.33
1.1	0.3	88.22	89.47
0.5	0.3	90.62	91.23
0.2	0.3	92.46	92.69
Original model		92.03	-

B.3. 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 fact that the gain in the objective function is bigger for these last layers since their filters contain way more parameters than filters belonging to the first layers. When we allow our pruning algorithm to heavily prune the network at the cost of a consistent accuracy drop or when the model is so over-parametrized that even pruning a lot of parameters slightly affect the accuracy, we notice that all final layers are fully pruned. Differently, when we prune less parameters to avoid big accuracy drops the final layers that are not fully pruned tend to be the same for different configurations of the hyperparameters, i.e., likely identifying essential structures of the model. For example, pruning ResNet18 on the Imagenet data set, the layer with the last residual connection is not pruned in almost all our experiments.

Appendix C. Experiments detailed setting

C.1. CIFAR10 and CIFAR100

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 1e-1 or 1e-2 and then it is divided by 10 at epochs 120, 200, 230, 250, 350, 400 and 450.

Appendix D. 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. On the ImageNet tests, as usual for datasets with so many classes, we decided to report also the top5 accuracy, that is the percentage of samples where the correct label was on the 5 higher scored classes by the model.

Appendix E. Detailed results

Table 8: Results of our algorithm on CIFAR-10 using ResNet-20

Learning rate	λ	α	Accuracy	Pruned parameters (%)	
0.1	1.1	0.01	85.56	231597	(85.78)
0.1	1.1	1e-4	86.35	227394	(84.22)
0.1	1.7	0.3	87.33	217233	(80.46)
0.1	0.8	0.1	88.14	206694	(76.55)
0.1	1.1	0.3	89.47	199809	(74.00)
0.1	0.5	0.01	90.06	187767	(69.54)
0.1	0.5	0.3	91.23	140535	(52.05)
0.1	0.2	0.3	92.69	64188	(23.77)
Original model			92.03	0	(0.00)

Table 9: Results of our algorithm on CIFAR-100 using ResNet-20.

Learning rate	λ	α	Accuracy	Pruned parameters (%)	
0.01	1.3	1e-3	65.28	149184	(55.25)
0.01	1.0	0.1	66.28	123984	(45.92)
0.01	1.6	0.3	66.64	104256	(38.61)
0.01	1.3	0.3	68.36	84960	(31.47)
0.01	0.7	0.3	69.20	42480	(15.73)
Original model			68.55	0	(0.00)

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

Learning rate	λ	α	Accuracy	Pruned parameters (%)	
0.1	1.7	1e-4	90.12	756882	(89.04)
0.1	1.1	0.01	91.35	720135	(84.72)
0.1	0.5	0.01	92.55	633960	(74.58)
0.01	1.1	1e-4	93.13	460800	(54.21)
0.01	0.8	0.1	93.98	268128	(31.54)
Original model			93.35	0	(0.00)

Table 11: Results of our algorithm on ImageNet using ResNet-18

Learning rate	λ	α	Accuracy (top1)	Accuracy (top5)	Pruned parameters (%)	
0.1	0.5	0.3	70.89	89.79	652644	(6.11)
0.1	1.3	0.3	67.26	87.71	5796625	(54.33)
0.1	1.3	0.01	62.27	84.43	8607247	(80.67)
Original model			69.76	89.08	0	(0.00)