Group Sparsity: A Unified Framework for Network Pruning and Neural Architecture Search

Avraam Chatzimichailidis[1,2,5], Arber Zela[3], Shalini Shalini[1,4], Peter Labus[1,5], Janis Keuper[1,6], Frank Hutter[3,7], Yang Yang[1,5]

[1]Department of High Performance Computing, Fraunhofer ITWM
{avraam.chatzimichailidis, peter.labus, yang.yang}@itwm.fraunhofer.de
[2]Chair for Scientific Computing, TU Kaiserslautern
[3]Department of Computer Science, University of Freiburg
{zelaa, fh}@cs.uni-freiburg.de
[4]Department of Computer Science, TU Kaiserslautern
s_shalini18@cs.uni-kl.de
[5]Fraunhofer Center Machine Learning, Germany
[6]Institute for Machine Learning and Analytics, Offenburg University
janis.keuper@hs-offenburg.de
[7]Bosch Center for Artificial Intelligence, Renningen

Abstract

We demonstrate how to exploit group sparsity in order to bridge the areas of network pruning and neural architecture search (NAS). This results in a new one-shot NAS optimizer that casts the problem as a single-level optimization problem and does not suffer any performance degradation from discretizating the architecture.

1. Introduction

Network pruning is a key technique to compress neural networks and reduce their computational complexity. Among the various pruning techniques, *group sparsity* can bring structured sparsity in the form of kernel/channel pruning. A seemingly obvious but often overlooked perspective of one-shot neural achitecture search (NAS) is that it is in essence a pruning process. In this paper, we extend the group sparsity approach to tackle the one-shot NAS problem. Because group sparsity is based on the nonsmooth $L_{2,1}$-norm regularization, the resulting training problem is nonsmooth and cannot be solved by standard stochastic gradient descent (SGD). In the proposed GSparsity method, we use the recent ProxSGD algorithm [12] in combination with group sparsity to efficiently converge to a sparse solution.

Contributions We make fundamental contributions to the area of one-shot NAS:

- We unify the areas of network pruning and NAS. We achieve this by grouping the trainable parameters of kernels/filters/operations together and apply group sparsity regularization directly to those groups.
- This approach renders the architectural parameters typical of most one-shot methods superfluous, casting the NAS problem as a standard single-level optimization problem, which is much easier to solve optimally than bi-level optimization problems.
- While the optimization problem with group sparsity regularization is nonsmooth, we use the recent proximal stochastic gradient descent (ProxSGD) [12] optimizer to solve this problem efficiently. ProxSGD converges to a sparse solution, where the weights of non-important groups are exactly zero. As a result, while previous methods, such as DARTS [6] and its successors, show substantial performance degradation in their discretization step, our approach avoids any such performance degradation (verified in our experiments for operation pruning).

Related work. DARTS [6] formulates the one-shot NAS problem as a differentiable optimization problem, which can be solved by standard low-complexity stochastic gradient descent algorithms. However, several limitations

$$\widehat{\boldsymbol{w}}_k(t) = \underset{\boldsymbol{w}_k}{\arg\min} \left\{ (\boldsymbol{w}_k - \boldsymbol{w}_k(t))^T \boldsymbol{v}_k(t) + \frac{\tau_k(t)}{2} \|\boldsymbol{w}_k - \boldsymbol{w}_k(t)\|_2^2 + \mu_k \|\boldsymbol{w}_k\|_2 \right\} \tag{1a}$$

$$= \left(1 - \frac{\mu_k}{\tau_k(t) \left\| \boldsymbol{w}_k(t) - \frac{1}{\tau_k(t)} \boldsymbol{v}_k(t) \right\|_2} \right)^+ \left(\boldsymbol{w}_k(t) - \frac{1}{\tau_k(t)} \boldsymbol{v}_k(t) \right), \ k = 1, \ldots, K. \tag{1b}$$

are still prevalent. Firstly, the importance of each operation is modelled by a learnable scalar parameter. Discretizing these continuous architectural parameters after search is completed can lead to a performance degradation [13, 14, 7]. Secondly, the problem is framed as a bi-level optimization problem [4], and it is indeed complex and computationally intractable to solve exactly. Lastly, as shown in [13], the behaviour of DARTS is not robust across search spaces, often producing degenerate results with architectures composed by only parameterless operations.

Many follow-up works have been proposed to improve DARTS from the above aspects [10, 13, 1, 2, 7]. Closest to our work is the HAPG algorithm recently proposed in [9], where an additional group sparsity regularization is applied to the architectural parameters in order to reduce the discretization gap after the search procedure. The core difference between our proposed algorithm and HAPG is that we apply the group sparsity constraints directly to the one-shot model weights. Therefore, the architectural parameters are unnecessary in our framework and the problem reduces to a single-level optimization. [8] propose an algorithm to learn the connectivity pattern in neural networks by imposing a constraint on the maximal number of edges that the target network has to contain. In contrast, our algorithm determines automatically what the connectivity pattern is based on the group sparsity constraints over the parameters. [7] revisit the discretization step in many one-shot NAS algorithms and propose a new architecture selection method by evaluating the one-shot model after pruning some operations following a predefined post-hoc heuristic. This is done implicitly with our method during the search routine without the need to resolve to such heuristics.

2. Method

The vector $\boldsymbol{w}$ of trainable parameters of a neural network is decomposed into subvectors $\boldsymbol{w} = (\boldsymbol{w}_k)_{k=1}^K$, such that $\boldsymbol{w}_k$ represents a group of weights; we denote the number of (non-overlapping) groups as K. Depending on the specific task at hand, a group could be a kernel or filter. In this work, we notice that group sparsity also allows grouping of *operations* in a NAS search space. Thus, operations that are not important will be pruned away by the algorithm by setting their parameter values to zero. Therefore, there is no need to use architectural parameters and our new method, dubbed *GSparsity*, can directly address differentiable NAS as a single-level optimization problem. Benefits of this approach include that (1) we can use the whole dataset for updating the operation's weights and (2) we do not need to use expensive second order optimization in order to approximate the bi-level optimization as done in [6]; (3) we can directly prove convergence; and (4) there is no performance degradation from discretizing the architecture.

We formulate network training as the following optimization problem, which aims at minimizing the loss function f augmented by group-sparsity regularization on $\boldsymbol{w}$:

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \frac{1}{|\mathbb{X}|} \sum_{\boldsymbol{x} \in \mathbb{X}} f(\boldsymbol{w}, \boldsymbol{x}) + \sum_{k=1}^{K} \mu_k \|\boldsymbol{w}_k\|_2, \tag{2}$$

where $\boldsymbol{x}$ is a training example from the training dataset $\mathbb{X}$ (with $|\mathbb{X}|$ denoting the number of training examples). Besides, the L_2 norm is defined as $\|\boldsymbol{w}\|_2 \triangleq \sqrt{\boldsymbol{w}^T \boldsymbol{w}}$ and it is a nonsmooth convex function. The regularization $\sum_{k=1}^K \mu_k \|\boldsymbol{w}_k\|_2$ in (2) is usually referred to as the $L_{2,1}$ norm (or the *mixed* norm), and it can promote a group-sparse neural network, in the sense that most groups will be zero and hence can be removed from the neural network without incurring any performance loss.

We tackle this problem with the recent ProxSGD algorithm [12], where the weights are updated iteratively as follows:

1. Compute the instantaneous gradient $\boldsymbol{g}(t)$ based on the minibatch $\mathbb{M}(t)$:

$$\boldsymbol{g}(t) = \frac{1}{|\mathbb{M}(t)|} \sum_{\boldsymbol{x} \in \mathbb{M}(t)} \nabla_{\boldsymbol{w}} f(\boldsymbol{w}(t), \boldsymbol{x}).$$

2. Update the momentum:

$$\boldsymbol{v}(t) = (1 - \rho(t)) \boldsymbol{v}(t-1) + \rho(t) \boldsymbol{g}(t).$$

3. Compute $\widehat{\boldsymbol{w}}(t)$ by solving the approximation subproblem specified in (1) at the top of this page.

4. Update the weight:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \epsilon(t)(\widehat{\boldsymbol{w}}(t) - \boldsymbol{w}(t)).$$

Regularization gain μ	Accuracy before pruning	Accuracy after pruning		
		pruning threshold		
		1e-6	1e-3	0.5
0.0001	96.50	96.50	96.50	92.08
0.0005	96.36	96.36	96.36	13.34
0.004	96.48	96.48	96.48	10

Table 1: Operation pruning: The accuracy before pruning and after pruning (without retraining). In contrast to Figure 1, here we report accuracies obtained only through inference, *without* retraining the pruned network again.

ProxSGD converges almost surely to a stationary point of the nonconvex nonsmooth optimization problem (2) under standard assumptions on the loss function f, momentum $\rho(t)$ and learning rate $\epsilon(t)$. We refer to Theorem 1 of [12] for more details.

3. Experiments

We now empirically show that (1) we can group the parameters of the same operation and use group sparsity in order to prune entire operations of a neural network, and (2) using this operation pruning in our new Group Sparsity NAS (GS-NAS) method yields very competitive and robust results for various search spaces and datasets. Throughout, we follow the NAS best practice checklist [5] by using standard NAS benchmarks without any tweaks, performing multiple runs, and making our code publicly available at https://github.com/cc-hpc-itwm/GSparsity, along with the logs of all runs.

Operation Pruning We first demonstrate that we can group the parameters of the same operation and use group sparsity in order to prune entire operations of a neural network. To our best knowledge, this is the first work considering operation pruning.

As a base architecture to be pruned, we chose one of the networks found in the original DARTS paper: DARTS-V2, which has 3.3M parameters. We use ProxSGD to train this architecture on CIFAR-10, varying the regularization gain hyperparameter μ_k across runs that controls the amount of regularization (and thus sparsification). We keep μ_k at the same value for all groups ($\mu_k = \mu$ for all k) and consider values of $\mu \in \{0.0001, 0.0002, 0.0005, 0.002, 0.004\}$. After training, we prune the operations whose L_2 norm is smaller than 10^{-6}, and retrain the pruned network by SGD with momentum.

Figure 1 shows that the retrained architecture's accuracy remains nearly constant when up to 40% of weights are pruned, and even up to the most aggressive pruning of 70% of the weights performance only degrades gradually.

Table 1 shows the accuracies before and after pruning

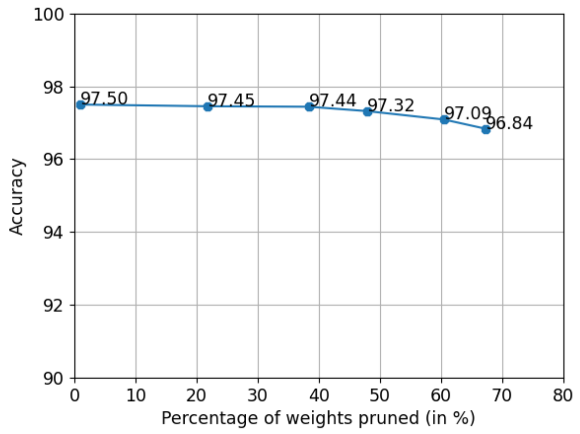

Figure 1: Trade-off achieved between percentage of pruned weights and accuracy (after retraining the pruned network) that can be achieved by operation pruning.

without retraining. We readily see that the proposed group sparsity approach incurs *no discretization error at all*, even when the pruning threshold is only modestly small (1e-3).

Architecture Space and Search Settings We now evaluate GS-NAS in a standard NAS setup. We use the DARTS setting [6], where we train on a supernet that consists of 8 cells with 16 initial channels. To apply GS-NAS, we put the same operation in different cells of the same type into the same group.

We train the supernet for 100 epochs of ProxSGD, using the full training set with 50k samples, with the following hyperparameters: (constant) learning rate 0.001, momentum 0.8, and $\tau_k(t) = 1$ for all k and t. To deal with the different number of parameters in each group, we normalize the regularization gain μ by the size of the group: $\mu_k = \mu/\sqrt{|\boldsymbol{w}_k|}$, where $\mu = 60$, $\boldsymbol{w}_k$ represents all weights in Group k and $|\boldsymbol{w}_k|$ is the size of $\boldsymbol{w}_k$. Note that μ is a hyperparameter that needs to be tuned so that the desired sparsity level is achieved. In practice, this can be done in a similar fashion as bisection, as the larger μ is, the fewer nonzero groups there are.

The *final evaluation* phase follows the original DARTS protocol: the architecture (consisting of two cells) found in the *search* phase are stacked in order to form a full network with 14 cells. The number of initial channels is increased to 36. The full network is trained for 600 epoch by SGD with momentum. The hyperparameters are also chosen as in DARTS [6].

Table 2 summarizes our results. We ran each NAS method three times and evaluated each of the resulting three architectures three times; we report means and standard deviations over the resulting nine models. In this evaluation, our GS-NAS method achieves the best performance, alongside PC-DARTS. We note that we obtained the results for this table by re-running the authors' original implementations, but that the results for most methods are somewhat worse than in the respective papers introducing them. One reason for this is that we consider the average performance based on all architectures (instead of the best architecture w.r.t. validation performance of four repetitions of the search phase, as proposed in the DARTS paper [6] and largely followed since).

Algorithm	Accuracy	Search Cost	Params (M)
DARTS (second order) [6]	96.98 +/- 0.13	1.45 days	2.11 +/- 0.33
P-DARTS [3]	97.05 +/- 0.20	0.25 day	3.98 +/- 0.19
PC-DARTS [11]	**97.13 +/- 0.16**	0.125 day	2.98 +/- 0.04
DrNAS [2]	96.95 +/- 0.08	0.84 day	4.64 +/- 0.06
GS-NAS (ours)	**97.17 +/- 0.11**	1 day	4.54 +/- 0.17

Table 2: Network Architecture Search for CIFAR-10. The performance is reproduced by using the authors' implementation.

Results on subspaces of the DARTS search space and on the Robustness of GS-NAS DARTS does not perform well on different search spaces that only allow a subset of operations from the original DARTS search space [13]. We therefore test GS-NAS on spaces **S1**, **S2** and **S4** [1] from [13].

Table 3 summarizes the performance of DARTS, ES-DARTS (the robust early stopping version of DARTS from [13]), and GS-NAS on the above search spaces. In this comparison, GS-NAS performs amongst the best for all of **S1**, **S2** and **S4**, with up to 1.53% absolute test error improvement on **S4** compared to ES-DARTS (and larger improvements compared to DARTS).

4. Conclusion

In this work, we propose to use group sparsity in order to bridge the gap between network pruning and differential NAS. We show that by reformulating NAS as a pruning problem, we are able to reach superior performance without suffering from overfitting or from performance degradation after discretizing the architecture.

[1]The search space **S3** in [13] is {3×3 SepConv, SkipConnect, Zero}. We do not consider it as we would implicitly get the ZERO operation from **S2** when none of {3×3 SepConv, SkipConnect} is selected.

Search Space	DARTS*	ES-DARTS*	GS-NAS
S1	4.66 +/- 0.71	**3.05 +/- 0.07**	**3.06 +/- 0.14**
S2	4.42 +/- 0.40	3.41 +/- 0.14	**2.60 +/- 0.11**
S4	6.95 +/- 0.18	4.17 +/- 0.21	**2.64 +/- 0.12**

Table 3: Performance (in terms of error) of DARTS, ES-DARTS and the proposed GS-NAS on CIFAR-10 (*The numbers are taken from Table 1 of [13]).

References

[1] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of ADAM-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2019. 2

[2] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. DrNAS: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021. 2, 4

[3] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019. 4

[4] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization, 2007. 2

[5] Marius Lindauer and Frank Hutter. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, 21(243):1–18, 2020. 3

[6] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 1, 2, 3, 4

[7] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable NAS. In *International Conference on Learning Representations*, 2021. 2

[8] Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019. 2

[9] Yan Wu, Aoming Liu, Zhiwu Huang, Siwei Zhang, and Luc Van Gool. Neural architecture search as sparse supernet. In *2021 AAAI Conference on Artificial Intelligence*, 2021. 2

[10] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic neural architecture search. In *International Conference on Learning Representations*, 2019. 2

[11] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: Partial channel connections for memory-efficient differentiable architecture search. In *International Conference on Learning Representations*, volume 1, pages 1–13, 2020. 4

[12] Yang Yang, Yaxiong Yuan, Avraam Chatzimichailidis, Ruud JG van Sloun, Lei Lei, and Symeon Chatzinotas. ProxSGD: Training Structured Neural Networks under Regularization and Constraints. In *Proceedings of the International Conference on Learning Representations*, 2020. 1, 2, 3

[13] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020. 2, 4

[14] Arber Zela, Julien Siems, and Frank Hutter. NAS-Bench-1Shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020. 2