Feather: An Elegant Solution to Effective DNN Sparsification

Athanasios Glentis Georgoulakis
athglentis@gmail.com

George Retsinas
gretsinas@central.ntua.gr

Petros Maragos
maragos@cs.ntua.gr

School of Electrical and Computer
Engineering
National Technical University of Athens
15773 Athens, Greece

Abstract

Neural Network pruning is an increasingly popular way for producing compact and efficient models, suitable for resource-limited environments, while preserving high performance. While the pruning can be performed using a multi-cycle training and fine-tuning process, the recent trend is to encompass the sparsification process during the standard course of training. To this end, we introduce Feather, an efficient sparse training module utilizing the powerful Straight Through Estimator as its core, coupled with a new thresholding operator and a gradient scaling technique, enabling robust, out-of-the-box sparsification performance. Feather's effectiveness and adaptability is demonstrated using various architectures on the CIFAR dataset, while on ImageNet it achieves state-of-the-art Top-1 validation accuracy using the ResNet-50 architecture, surpassing existing methods, including more complex and computationally heavy ones, by a considerable margin.

1 Introduction

Machine learning applications have benefited tremendously in the past decade from the use of Deep Neural Networks (DNNs), especially in the field of computer vision [22]. While DNNs are more than capable at achieving state-of-the-art (SoA) results, they rely upon having very large numbers of trained parameters and thus require huge amounts of computational and memory resources [14, 20, 34]. Those requirements often prohibit the use of powerful DNNs in mobile and resource-limited devices, resulting in a gap between the achieved SoA results and the ones coming from the deployed technology. Recognizing that DNNs are heavily over-parametrized [32], network pruning (*i.e.* the process of removing network parameters) has been studied extensively in the recent years as a way of drastically reducing the model's size and computational footprint [3, 5, 15, 24].

Broadly the pruning methodology can be categorized in terms of (a) the granularity of the sparsified elements, (b) the pruning criterion and (c) the timeframe the sparsity is induced. Based on granularity methods are divided between structured or unstructured. Structured methods remove groups of parameters such as kernels and filters [23, 28], attaining sparsity that is more easily utilized by commodity hardware for computational gains, while

unstructured freely remove individual parameters, generally leading to more accurate sparse models [11, 12]. As pruning criterion, a large number of methods associate parameter importance with large magnitude values and prune elements with magnitudes smaller than some threshold value [11, 12, 40]. This simple criterion, judging a weight's importance based on its magnitude, is efficient to compute and is found to be effective in the pruning literature, achieving high sparsity ratios with minimal performance loss [9, 12]. An alternative way involves estimating how pruning an element will affect the training loss, however requires calculating Hessian information which is computationally expensive [1, 7, 13]. Finally, considering at what stage of the training process the pruning takes place, the different algorithms are usually divided into dense-to-sparse and sparse-to-sparse training [21]. The former, starting with a dense network produce the requested sparse version at the end of the pruning procedure [12, 21, 40]. Some methods of this type require multiple pruning and retraining cycles [29] and thus prolong the required training time. Sparse-to-sparse methodologies aim to reduce training times even further by starting training an already sparse network[6, 26, 27] although the achieved performances generally fall behind of those using dense-to-sparse training methods [21].

More recently, a number of works [18, 30, 53, 55] which are built around the concept of the Straight Through Estimator (STE) [2] have demonstrated that SoA results can be achieved by pruning DNNs along the normal course of training. Sparse training with the STE is performed by computing the forward pass using the thresholded (pruned) version of the weights while updating the dense weights during the backward pass, treating the thresholding function (that performs the pruning) as the identity.

In our work, we focus on improving sparse training with the STE by addressing a number of overlooked shortcomings of the method, eventually introducing a novel pruning module. We propose (i) a new thresholding function used for magnitude pruning and (ii) a straightforward way to control gradient flow of the pruned weights. More specifically, instead of using hard or soft thresholding [8], which previous STE based methods mostly use, we propose a family of thresholding functions that lie in between the aforementioned two and combine their advantages, namely reduced bias between the thresholded weights and their dense counterparts and a smooth transition region near the threshold. Complementary to the proposed thresholding approach, we suggest scaling the gradients attributed to pruned parameters by a parameter $\theta \in (0, 1)$, aiming on improving the stability of the pruning mask, a factor that we find to be crucial when targeting very high sparsity ratios.

We demonstrate the effectiveness of our sparse training approach when applied to magnitude based unstructured pruning frameworks, which reach a user-specified sparsity ratio by incrementally pruning the network during training. In more detail, we perform extensive experiments on both CIFAR [19] and ImageNet [4] datasets using a very simplistic global thresholding pruning procedure as a backbone and additionally with a recently proposed layer-wise pruning framework [30]. Our sparse training approach surpasses the (generally more computationally expensive) current SoA unstructured pruning algorithms, significantly improving the previously achieved generalization accuracies of the resulting sparse models.

Overall, the contributions of our paper can be summarized as follows:

- We introduce Feather, a versatile sparse training module that can be used to efficiently and effectively prune neural networks up to extreme sparsity levels. The proposed module was evaluated on different magnitude pruning backbones and achieved consistent improvements over the prior state-of-the-art.

- We highlight the importance of a well-crafted thresholding function that finds a fine

balance between the two standard ones, namely hard and soft operators.

- We highlight the correlation between scaling the gradients of the pruned weights and the targeted sparsity ratio: high sparsity targets should be accompanied with lower scaling values to provide high performing pruned models.

2 Proposed Method

In this work, we developed a novel pruning module, dubbed as Feather. The name symbolizes its lightweight nature, the elegance with which it achieves sparsity, as well as the lightness of the resulting pruned networks. It can be utilized in various magnitude pruning schemes, including strategies where pruning can be performed globally or in a layer-wise fashion. The setting of interest is magnitude pruning, where a weight is kept only if its magnitude surpasses a threshold value T, and a sparse training process is followed (i.e., perform the pruning procedure along the standard course of training), as done in most of the SoA systems. In essence, we capitalize on the effectiveness of modern STE-based approaches, carefully addressing potential issues. Implementation-wise, the proposed module is applied at each layer, replacing a typical pruning operation, affecting both the forward and the backward step.

In what follows, we first provide the necessary prerequisites for sparse training with STE and then we describe the proposed module, emphasizing on the proposed modifications on both the forward and the backward steps and highlighting the underlying motivations.

2.1 Preliminaries: Sparse Training

In this analysis we examine how the pruning and the subsequent weight update is performed (during a training iteration) on a single layer under the STE framework.

First, let us consider a thresholding operator, the core of the magnitude pruning approaches, as a function $\mathcal{P}_{(T)}(x)$ that performs the pruning given a threshold value T. As a general rule this function takes zero values when $|x| \leq T$ and non-zero values otherwise. Typical instantiations of this function are the hard and the soft thresholding operators.

A first approach is to directly back-propagate through the thresholding operator. By doing so, the gradients belonging to the pruned weights will be zeroed thus excluding them from the update step. Regrettably, due to the resulting sparse gradient, this approach may lead to unwanted decay of immaturely trained weights and a slow exploration of the possible sparsity patterns [18, 53].

Lately, to circumvent the aforementioned issues, a number of pruning methods [18, 50, 53, 55] have achieved SoA results relying on the concept of a Straight Through Estimator [2] training approach. In a nutshell, with the STE formulation, the weight update and thresholding equations are now decoupled into:

$$\tilde{\boldsymbol{w}}_k = \mathcal{P}_{(T)}(\boldsymbol{w}_k) \tag{1}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta \cdot \nabla \mathcal{L}(\tilde{\boldsymbol{w}}_k), \tag{2}$$

where $\boldsymbol{w}$ are the vectorized layer weights, $\tilde{\boldsymbol{w}}$ the pruned weights after applying the thresholding operation, $\mathcal{L}(\boldsymbol{w})$ the loss function and η the step size. The reported expressions correspond to k-th iteration of a Gradient Descent formulation to highlight the effect of STE. The same procedure is trivially extended to any optimizer required.
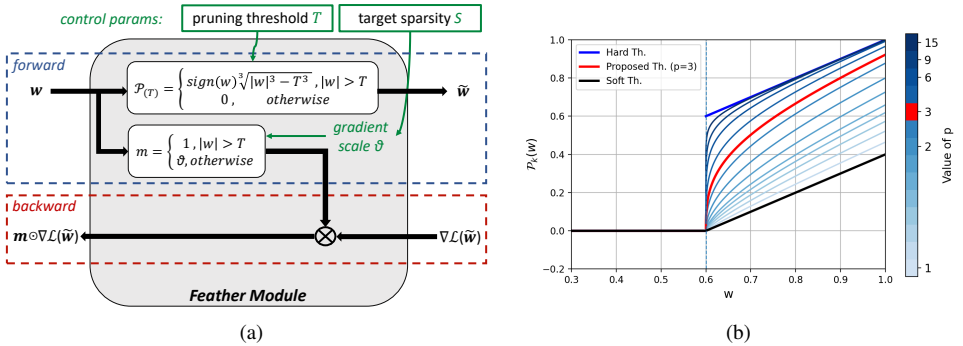
Figure 1: (a) The proposed sparse training block, utilizing the new thresholding operator and the gradient scaling mask (b) The proposed family of thresholding operators for varying values of p. We adopt $p = 3$, resulting to a fine balance between the two extremes, hard and soft thresholding respectively.

The main idea is to consider the thresholding operator as the identity function during back-propagation and update both pruned and unpruned weights based on the gradients of the loss w.r.t. the sparse set of weights. During the forward pass however, only the sparse weights are used so that the network is trained under the sparsity constrain. The key benefit of sparse training with the STE is that it allows for pruned weights to become active again, if at some point during the training they get a large enough magnitude. This process therefore promotes the exploration of different sparsity patterns and is found to result to better performing sparse networks [18, 33].

2.2   Proposed Sparse Training Module

The aforementioned STE pipeline was a stimulus for the proposed module; our primary goal was to maintain the simplicity of the original idea, and thus we concentrated on enhancing two fundamental elements: the thresholding operator during the forward step and the gradient manipulation during the backpropagation step. Overall the proposed enhancements aim to assist the training process by promoting convergence to well-performing yet highly sparse solutions. The functionality of Feather, the proposed module, is summarized in Figure 1(a), where the depicted components will be described in detail in what follows.

2.2.1   STE Thresholding Operator

Focusing on the thresholding operator used during the forward pass, the most straightforward way to define a magnitude pruning step is through the hard thresholding function, which is discontinuous at the threshold value. This discontinuity might result to training instabilities when weights pass from pruned to active states and vice versa, since the gradient revived might not justify the value after thresholding. To addressing this matter, a recent method [35] suggested that the soft thresholding operator is preferable in the context of STE, while soft thresholding is a common choice in pruning approaches in general [21]. Soft thresholding, although suppressing the discontinuity, induces a constant bias (equal to the threshold value) between the active weights and their dense counterparts, updated during the STE back-propagation phase. Note that by its nature the STE introduces an inconsistency between the

forward and the backward pass, since during the former the sparse set of weights is used whereas during the later the computed gradients are used to update the dense weights.

Motivated by works in the area of sparse regression, aiming on finding thresholding operators that lie in between hard and soft thresholding [10, 25], we propose the following family of operators to be used with STE:

$$\mathcal{P}_{(T)}(w) = \begin{cases} \text{sign}(w) \cdot (|w|^p - T^p)^{1/p}, & \text{if } |w| > T \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The behavior of $\mathcal{P}_{(T)}$ is depicted in Figure 1(b), for varying values of the power parameter p. Intuitively, as p grows we deviate from the soft ($p = 1$) and approach the hard thresholding operator. Under that point of view, the proposed operator can be considered a generalization of the existing two. Note that the proposed function, when avoiding extreme selections for p, tries to balance between the two aforementioned properties: continuity and bias. Based on that, we suggest a value of $p = 3$ being a reasonable compromise that adequately addresses both issues. We provide further validating experiments in the supplementary material.

In a final note, we want to emphasize that our proposed thresholding operator, apart from having an intuitive explanation for the resulting improved performance (as empirically evaluated in Section 4), due to its simplicity it practically imposes no training overhead and can be used off-the-self in pruning frameworks in conjunction with the STE.

2.2.2 STE and Gradient Scaling

The main motivation behind the STE is to allow gradient flow to pruned weights and thus enable the exploration of multiple sparsity patterns during sparse training. As stated before, this is achieved by treating the thresholding function as the identity during the back-propagation. However, in certain cases it might be beneficial to limit the variations of the mask and favor a more stable sparsity pattern. We find that a rather straightforward, yet intuitive way to control the mask's stability is to scale the gradients of the pruned weights by a constant value $\theta \in [0, 1]$, effectively modifying the update step of Eq. 2 into:

$$w_{k+1} = w_k - \eta \cdot m_k \odot \nabla \mathcal{L}(\tilde{w}_k), \tag{4}$$

where $m_k \in \{\theta, 1\}^N$ such that $m_{i,k} = 1$ if $w_{i,k} > T$ and $m_{i,k} = \theta$ otherwise, with $\odot$ denoting element-wise product.

At the two extreme points, $\theta = 0$ and $\theta = 1$, the method aligns with the non-STE and the standard STE approaches. For in-between values of θ, pruned weights continue to receive gradient updates but with scaled down magnitudes, and therefore are, to some extent, promoted to decay, resulting to a more stable (but not completely fixed) mask.

In our experiments (Section 4.1) we find that scaling the gradients becomes beneficial when targeting very high sparsity ratios (e.g. 98% and above) where, due to having very few active weights, too frequent mask variations appear to destabilize the network. For more conservative ratios, using $\theta < 1$ appears not to improve the results and even lead to a small decline in accuracy at some cases.

To this end, based on experimental evidence, essentially relying on a profiling approach, we define an "automatic" way to set θ. In more detail, we select $\theta = g(S)$ as a simple step function, where $\theta = 1$ if the final target sparsity S is under 95% and $\theta = 0.5$ when $S \geq 95\%$. The value of θ is selected at the beginning of training and remains constant throughout the training process. Interestingly, no performance gains were observed by adopting more

complex scheduling tactics (e.g. having a gradual smooth transition over θ from 1 to a lower value) during training. Note that ideally a more complex function could be defined given an exhaustive profiling procedure, but we considered such ideas out of scope for this work.

3   Application on Pruning Frameworks

Our proposed sparse training module is versatile and not restricted to a particular pruning framework. We will demonstrate its effectiveness using two distinct frameworks, both a global and a layer-wise magnitude pruning backbone, as described bellow.

Global Magnitude Pruning: Most pruning methods that utilize the STE (or some variant of it) use global thresholding, in the sense that a single threshold T is selected for all layers, computed by sorting all weights, in order to prune the network up-to a specified sparsity ratio [18, 33, 35]. Common practice is to incrementally increase the requested ratio throughout the training process, thus giving the network time to adjust to different sparsity levels [21, 40]. A popular sparsity schedule was proposed in [40], where the network is trained densely for a small number of warm up epochs followed by a cubical increase of the sparsity ratio, until it reaches the final target ratio, which is then kept constant for the rest of the training epochs. For simplicity and since the exact sparsity schedule is not the focus of this work, we reach the final target ratio at 50% of the epochs, without having a densely training warm up phase which we found not to have an effect to the final performance.

Layer-wise Magnitude Pruning: To further reveal the efficacy of Feather, we adopted a considerably different pruning framework and pair it with the proposed module. We considered the Adaptive Sparsity Loss framework (ASL) [30], which tackles the magnitude pruning problem in an explicit layer-wise formulation. Specifically, ASL considers the per-layer pruning thresholds as trainable parameters which are included in an additional loss term (Sparsity Loss), constructed based on assumptions for the per-layer distributions. The thresholds are consequently learned during training by minimizing the extra loss term, resulting into a learned non-uniform per-layer sparsity. The overall sparsity of the DNN is constrained, via the loss, to a specific target sparsity. It should be noted that STE is also a core element of this approach. In this work, we used an enhanced version, referred to as ASL+, where we addressed existing shortcomings of the original ASL, including the addition of a sparsity scheduler, similar to the one defined in the global pruning counterpart, an element missing from [30], as well as a sorting-based correction step of the per-layer threshold to compensate for the approximate nature of the distribution-motivated sparsity loss. The alluring aspect of ASL is the ability to further define more complex sparsity-related goals through appropriately crafted sprasity losses, but such extension is outside the scope of this work.

4   Experimental Evaluation

The present section provides the experimental validation of the proposed method's effectiveness. First, we perform ablation studies to showcase the efficacy of each element of our approach. Then, we provide comparisons with relevant baselines and SoA unstructured prun-

ing algorithms. Specifically, we experiment with modern compact architectures ResNet-20 [14], MobileNetV1 [16] and DenseNet40-24 [17] on CIFAR-100[1] [19]. Furthermore, we provide large-scale experiments on the ImageNet [4] dataset using the ResNet50 [14] architecture in order to further verify the generalization abilities of our method and its performance edge over the current SoA. The reported results are obtained using SGD optimizer along with a Cosine Annealing scheduler, while the training hyperparameters are the typically used in literature for such settings (described in detail in the supplementary material).

4.1   Ablation Studies

The ablation studies were performed on the CIFAR-100 dataset, using the global magnitude pruning framework as the backbone to the Feather pruning module. We note that similar behaviors were observed using the Feather in conjunction with the layer-wise ASL+ framework. All figure data points represent averages of 3 runs and the corresponding standard deviations are shown as shaded regions.

Impact of Thresholding Operator: The effectiveness of the proposed thresholding function (with $p = 3$) is empirically evaluated and compared against that of the hard and soft threshold functions in Figure 2, while no gradient scaling was considered for this experiment (we used typical STE). The proposed threshold enables the training of more accurate sparse networks, especially at high pruning ratios (95% and above). Notably, the hard thresholding approach achieves considerably low results with the MobileNetV1 network, while the soft threshold with the other two. *Overall, our approach consistently leads to better trained networks, regardless the architecture and the sparsity ratio, supporting our claims of effectively combining the strengths of both soft and hard thresholding.*
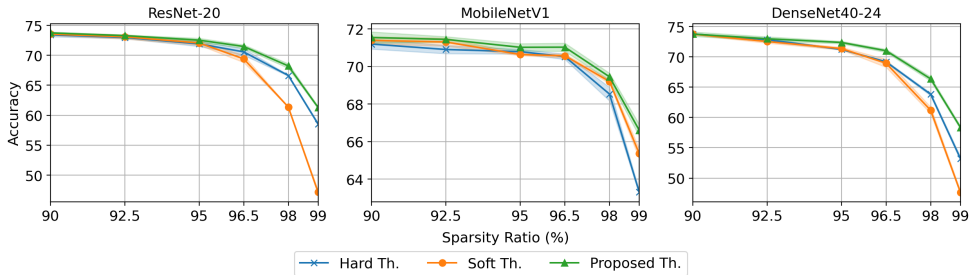

Figure 2: Study of the effect of the thresholding operator on the final sparse model accuracy. The proposed threshold steadily outperforms the hard and soft operators.

Impact of Gradient Scaling: In Figure 3 we investigate the relation between the value of the parameter $\theta \in [0, 1]$, that scales the gradient of the pruned weights, and the final model accuracy when requesting different sparsity ratios. Note that the under-performing case of $\theta = 0$ is equivalent to a non-STE variant. *Across all considered models, a clear trend can be seen; When targeting lower sparsity levels, best results are achieved with values of θ close to unity, whereas at more extreme sparsity ratios (such as 98% and 99%) the optimal values of θ seem to shift towards the middle of its range.*

---

[1]The channels of ResNet-20 are doubled for the experiments on CIFAR, as also done in [36, 39].

This observed dependency is the motivation behind an automatic selection of θ, as described in Section 2.2.2, where $\theta = 0.5$ for $S \geq 95\%$ and $\theta = 1$ otherwise. Note that this selection just aligns with the reported trend and is not optimal for every case reported. Despite this being a very "crude" selection, it is very effective, as the forthcoming experimental evaluations hint. Nonetheless the takeaway of this experiment is not a simple function of two modes, but bringing this relation to the spotlight. Based on this observation, we pave the way towards more complex functions or, more interestingly, towards having different scales per layer, relying on the per-layer sparsity rather than the overall sparsity. The later idea, a possible future direction of practical value, simply states that under-pruned layers can be more flexible to sparsity pattern variations than the overly pruned ones.
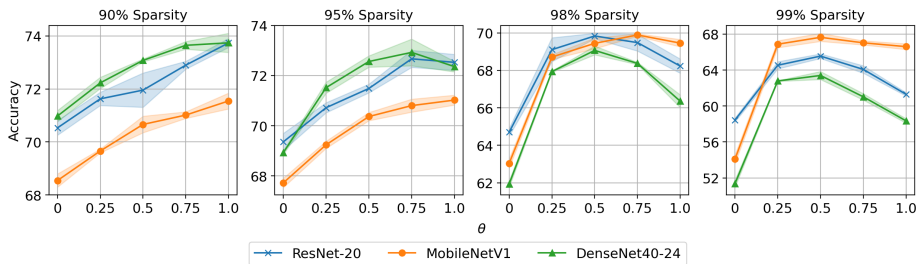

Figure 3: Study of the effect of gradient scaling. Under conservative final sparsity, θ near unity is preferable, while when targeting high sparsity, models benefit from θ near the middle of its range.

Gradient Scaling under different Thresholding Functions: Finally, we compare the impact of gradient scaling using the three different types of thresholding operators. Figure 4 shows that, regardless the choice of operator, using a scale $\theta \in (0, 1)$ is beneficial to the final accuracy at the high sparsity regimes. Nevertheless, our threshold maintains the lead in performance compared to the two standard ones.
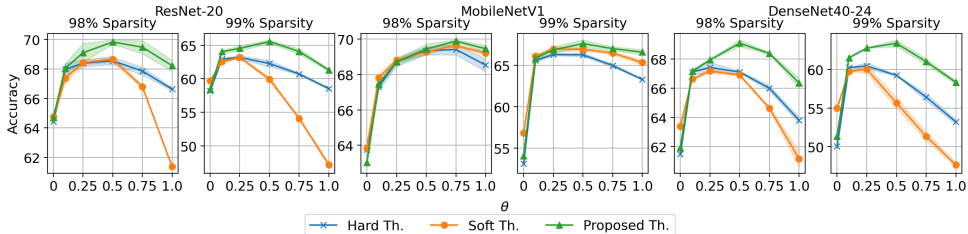

Figure 4: Gradient scaling improves the final accuracy at high sparsity, regardless the thresholding operator, while maximum performance is achieved if combined with the proposed threshold.

4.2   Comparison to SoA

CIFAR-100: For these experiments, we directly compare our results with the ones of ST-3 [35] and Spartan [33], being the two most recent and best performing sparse training ap-

Ratio	90%	95%	98%	99%
ResNet-20 (1.096M Params): 73.59 ±0.44				
ST-3 [55]	72.81 ±0.13	71.72 ±0.20	67.53 ±0.53	58.32 ±0.17
Spartan [53]	72.56 ±0.35	71.60 ±0.40	67.27 ±0.31	61.70 ±0.21
Feather-Global	**73.74** ±0.17	**72.53** ±0.32	**69.83** ±0.14	**65.55** ±0.25
Feather-ASL+	72.86 ±0.10	72.42 ±0.17	69.76 ±0.09	64.95 ±0.47
MobileNetV1 (3.315M Params): 71.15 ±0.17				
ST-3 [55]	70.94 ±0.25	70.44 ±0.23	69.40 ±0.06	66.63 ±0.15
Spartan [53]	70.52 ±0.51	69.01 ±0.11	65.52 ±0.24	60.65 ±0.22
Feather-Global	**71.55** ±0.30	71.03 ±0.20	**69.44** ±0.29	67.64 ±0.45
Feather-ASL+	71.10 ±0.31	**71.26** ±0.10	69.42 ±0.12	**67.86** ±0.03
DenseNet40-24 (0.714M Params): 74.70 ±0.51				
ST-3 [55]	72.56 ±0.31	71.21 ±0.35	65.48 ±0.18	56.18 ±0.60
Spartan [53]	73.13 ±0.25	71.61 ±0.04	65.94 ±0.07	58.64 ±0.18
Feather-Global	73.75 ±0.36	72.36 ±0.21	69.06 ±0.23	**63.40** ±0.44
Feather-ASL+	**73.92** ±0.19	**72.47** ±0.12	**69.08** ±0.19	62.94 ±0.14

Ratio	90%	95%	98%	99%
ResNet-50 (25.6M Params): 77.10				
GMP [40]	73.91	70.59	57.90	44.78
DNW [37]	74.00	68.30	58.20	-
STR [21]	74.31	70.40	61.46	51.82
ProbMask [39]	74.68	71.50	66.83	61.07
OptG [58]	74.28	72.45	67.20	62.10
ST-3 [55]	76.03	74.46	70.46	63.88
Spartan [53]	76.17	74.68	-	63.87
Feather-Global	**76.93**	**75.27**	**72.92**	**68.85**

(a) CIFAR-100 (b) ImageNet

Table 1: Comparison of Top-1 accuracy on CIFAR-100 (a) and ImageNet (b).

proaches. The methods are both dense-to-sparse, global magnitude based pruning algorithms, which gradually introduce sparsity during training, utilizing variants of the STE. Specifically, ST-3 adopts soft thresholding and a weight rescaling technique similar to the one used by dropout [31], while Spartan computes a soft top-k mask by solving a regularized Optimal Transport problem, therefore is more computationally expensive than our approach.

In particular, Table 1 (a) provides the results obtained on CIFAR-100 using the afore-mentioned methods over three different architectures and four different levels of sparsity. The reported results correspond to averages of 3 runs with the corresponding standard deviations. The results demonstrate that sparse training with Feather yields steadily more accurate models compared to both SoA methods, either using the simple global pruning approach or combined with ASL+. Notably, the gap in accuracy between our approach and that of the next best performing baseline grows up to 4% when considering the 99% sparse ResNet-20 and DenseNet40-24 models. Another interesting remark is that Feather can result to 90% sparse ResNet-20 and MobileNetV1 models with slightly better generalization accuracies that those of their dense counterparts, trained using the same number of epochs. The last remark hints that a well designed sparse training method can even be beneficial, not only for producing compact models, but also for improving the generalization performances, when considering relatively conservative sparsity ratios. Note that similar SoA results are observed using both tested backbones, a point that further validates Feather's versatility.

ImageNet: For the ImageNet experiments we include results from literature from an extended number of pruning methods achieved using the same number of epochs (100) and data augmentation. In particular, results from the relevant methods GMP [40], DNW [37], STR [21], ProbMask [39], OptG [58], ST-3 [55] and Spartan [53] are presented in Table 1 (b). We adopt the global pruning scheme combined with Feather module, being conceptually the simplest approach, in order to highlight our method's effectiveness compared to more sophisticated baselines. As we can see, it provides considerably better results than those from previous SoA, especially at very challenging pruning ratios over 98%. We want to emphasize that the improved performance with Feather does not come at the cost of higher training overheads or the need for complicated hyperparameter settings, in contrast to certain baselines (*e.g.* [21, 39]). Instead, the resulting accuracy gains can be attributed to the simple, yet carefully formulated modifications of the proposed module that fully utilizes the STE-based sparse training approach.

5 Conclusions

This paper proposes Feather, an effective and efficient sparse training module that can be easily applied to pruning frameworks. In particular, as demonstrated by extensive experiments on CIFAR and ImageNet datasets, using both a global and a layer-wise approach, it results to improving the previous SoA results, especially at high pruning ratios. Furthermore, ours method's success indicates the large potential of properly understanding and consequently improving the sparse training dynamics using an STE based approach, that despite its simplicity is shown to be highly effective.

Acknowledgements

References

[1] Alireza Aghasi, Afshin Abdi, and Justin Romberg. Fast convex pruning of deep neural networks. *SIAM Journal on Mathematics of Data Science*, 2(1):158–188, 2020.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[3] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.

[6] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

[7] Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in Neural Information Processing Systems*, 30:4860–4874, 2017.

[8] David L Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

[9] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[10] Katsuyuki Hagiwara. Bridging between soft and hard thresholding by scaling. *IEICE Transactions on Information and Systems*, 105(9):1529–1536, 2022.

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

[12] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems*, 28:1135–1143, 2015.

[13] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 293–299, 1993.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[15] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research*, 22(1):10882–11005, 2021.

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[18] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33:20744–20754, 2020.

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[21] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *Proceedings of the International Conference on Machine Learning*, pages 5544–5555, 2020.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

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

[24] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461: 370–403, 2021.

[25] Haoyang Liu and Rina Foygel Barber. Between hard and soft thresholding: optimal iterative thresholding algorithms. *Information and Inference: A Journal of the IMA*, 9 (4):899–933, 2020.

[26] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.

[27] Decebal Constantin Mocanu, Elena Mocanu, Tiago Pinto, Selima Curci, Phuong H Nguyen, Madeleine Gibescu, Damien Ernst, and Zita Vale. Sparse training theory for scalable and efficient agents. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2021.

[28] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[29] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

[30] George Retsinas, Athena Elafrou, Georgios Goumas, and Petros Maragos. Online weight pruning via adaptive sparsity loss. In *Proceedings of the IEEE International Conference on Image Processing*, pages 3517–3521, 2021.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[32] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12): 2295–2329, 2017.

[33] Kai Sheng Tai, Taipeng Tian, and Ser Nam Lim. Spartan: Differentiable sparsity via regularized transportation. *Advances in Neural Information Processing Systems*, 35: 4189–4202, 2022.

[34] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[35] Antoine Vanderschueren and Christophe De Vleeschouwer. Are straight-through gradients and soft-thresholding all you need for sparse training? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3808–3817, 2023.

[36] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *Proceedings of the International Conference on Learning Representations*, 2020.

[37] Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. *Advances in Neural Information Processing Systems*, 32:2684–2694, 2019.

[38] Yuxin Zhang, Mingbao Lin, Mengzhao Chen, Zihan Xu, Fei Chao, Yunhan Shen, Ke Li, Yongjian Wu, and Rongrong Ji. Optimizing gradient-driven criteria in network sparsity: Gradient is all you need. *arXiv preprint arXiv:2201.12826*, 2022.

[39] Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3599–3608, 2021.

[40] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.