Can Unstructured Pruning Reduce the Depth in Deep Neural Networks?

Zhu Liao   Victor Quétu   Van-Tam Nguyen   Enzo Tartaglione
LTCI, Télécom Paris, Institut Polytechnique de Paris, France
{name.surname}@telecom-paris.fr

Abstract

Pruning is a widely used technique for reducing the size of deep neural networks while maintaining their performance. However, such a technique, despite being able to massively compress deep models, is hardly able to remove entire layers from a model (even when structured): is this an addressable task? In this study, we introduce EGP, an innovative Entropy Guided Pruning algorithm aimed at reducing the size of deep neural networks while preserving their performance. The key focus of EGP is to prioritize pruning connections in layers with low entropy, ultimately leading to their complete removal. Through extensive experiments conducted on popular models like ResNet-18 and Swin-T, our findings demonstrate that EGP effectively compresses deep neural networks while maintaining competitive performance levels. Our results not only shed light on the underlying mechanism behind the advantages of unstructured pruning, but also pave the way for further investigations into the intricate relationship between entropy, pruning techniques, and deep learning performance. The EGP algorithm and its insights hold great promise for advancing the field of network compression and optimization.

1. Introduction

Deep neural networks have become a cornerstone of artificial intelligence and machine learning, achieving remarkable success in a wide range of applications, from video recognition [1] and natural language processing [25] to robotics [21] and autonomous driving [2]. However, the large size of these models poses significant challenges in terms of storage, memory, and computational requirements. To address these issues, researchers have developed various techniques to reduce the size of neural networks without sacrificing their performance, including quantization [9, 26], compression [6, 24], and pruning [10, 11, 15, 16, 23].

Pruning, in particular, has emerged as a popular and effective method for reducing the number of parameters in a neural network [3, 15, 20]. The idea behind pruning is to selectively remove weights, neurons, or entire layers
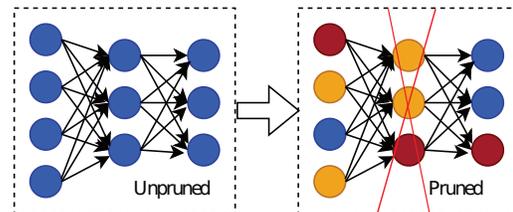

Figure 1: Effect of unstructured pruning in ReLU-activated models: some neurons are always OFF (yellow) and others are consistently in the linear region (red). These can be removed with no impact on the performance.

based on their magnitude or other criteria [7, 17, 27]. The pruned model can then be fine-tuned to recover its original performance, resulting in a smaller and more efficient network. Pruning is particularly effective for deep neural networks, which tend to be over-parameterized and therefore have many redundant or irrelevant weights.

Despite the widespread use of pruning, its underlying mechanism and the factors that contribute to its benefits are not well understood. Previous work has investigated the effects of pruning on the sparsity and connectivity patterns of neural networks [4, 5, 8], as well as its impact on the generalization performance of the pruned models [10]. Nowadays, indeed, massive study around this subject is being conducted [14, 22]. Although some works used entropy to drive pruning mechanisms [13], to the best of our knowledge, no previous work has explored the relationship between pruning and the entropy of activations in the network.

In this work, we investigate the impact of unstructured pruning on the entropy of activations in the network. In the context of neural networks, entropy can provide insights into the complexity and diversity of the representations learned by the model. Leveraging this measure, we propose EGP, a strategy to drive the unstructured pruning mechanism to prune layers exhibiting low entropy. Our experiments show that unstructured pruning reduces the entropy of activations in the network (Fig. 1). This reduction in entropy suggests that pruning results in a more structured activation pattern, which may as well improve the ability

of the network to generalize to new data. Interestingly, this approach overcomes the limits of structured pruning, which removes neurons persistently OFF.

Our work provides a new perspective on the mechanism behind the benefits of unstructured pruning and sheds light on the relationship between entropy, pruning, and deep learning performance. These findings open up avenues for further exploration. We have also developed an entropy-based iterative pruning technology that shows promise in compressing neural networks more effectively.

Despite other works proposing entropy-based approaches to drive pruning [13, 18, 19], to the best of our knowledge, EGP is the very first entropy-driven approach that can effectively reduce the depth of a deep neural network model without harming its performance. Different from previous entropy-based pruning methods which tried to remove a certain amount of parameters or neurons inside layers, EGP tries to remove whole layers. EGP is tested on popular models for computer vision, like ResNet-18 and Swin-T, and opens the roads to a new class of pruning algorithms.

2. Entropy Guided Pruning

In this section, we will provide details on how we compute the entropy of the activations inside the neural network model and develop our entropy-based iterative pruning. Fig. 2 proposes an overview of the proposed entropy computing and pruning method. First, we introduce the entropy calculation method (Sec. 2.1), showing also how to handle critical cases (Sec. 2.2), and then we provide an overview of the entropy-based iterative pruning method (Sec. 2.3).

2.1. Derivation

Let us extract the output $\boldsymbol{y}_{l,i}^{\xi}$ of the i-th neuron in the l-th layer (where N_l is the number of neurons in the l-th layer), given as input the ξ-th sample. Let us assume the activation function $\phi_n(\cdot)$ for the l-th layer is ReLU. Toward this end, we can write

$$\boldsymbol{y}_{l,i}^{\xi} = \mathrm{ReLU}\left(\boldsymbol{z}_{l,i}^{\xi}\right), \qquad (1)$$

where $\boldsymbol{z}_{l,i}$ is the *post-synaptic potential* for the i-th neuron in the l-th layer. Please note that in convolutional layers the dimensionality for $\boldsymbol{y}_{l,i}^{\xi}$ is not necessarily unitary, but is proportional to the input's size $M_l = K_{1,l} \times K_{2,l}$, where $K_{1,l}$ and $K_{2,l}$ are the dimensions for the generated feature map. We assume, for the sake of simplicity, that the size of every ξ is the same for the whole dataset Ξ. We know that ReLU-activated neurons have essentially two working regions:

- an ON region, for $z_{l,i,j} > 0$;
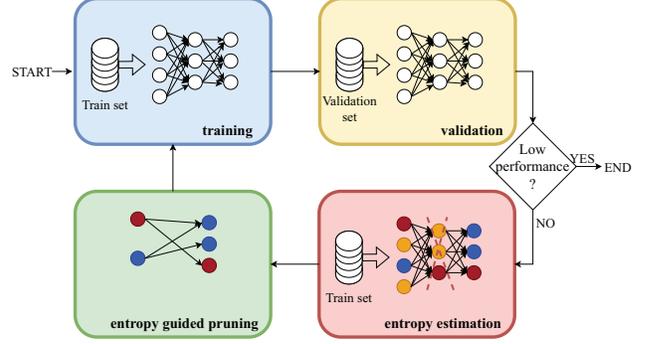- an OFF region, for $z_{l,i,j} \leq 0$.

Figure 2: Overview of the EGP approach.

These working regions will be our possible "states". To identify the state $s_{l,i,j}$ for the j-th feature extracted by the i-th neuron, we can apply the one-step function $H(\cdot)$ to the post-synaptic potential, obtaining $s_{l,i,j}^{\xi} = H(z_{l,i,j}^{\xi})$. At this point, we can obtain the frequency of the i-th neuron to be in the ON state, simply through

$$p^{\mathrm{ON}}(l,i|\Xi) = \frac{1}{\|\Xi\|_0 \cdot M_l} \sum_{\xi \in \Xi} \sum_j (s_{l,i,j}^{\xi}), \qquad (2)$$

where $\|\Xi\|_0$ is the cardinality of the dataset. Evidently, we know that in this case $p^{\mathrm{OFF}}(l,i|\Xi) = 1 - p^{\mathrm{ON}}(l,i|\Xi)$. At this point, we can use the definition in (2) to write the entropy $\mathcal{H}(l,i|\Xi)$:

$$\mathcal{H}(l,i|\Xi) = -\frac{1}{\log(2)} \left\{ p^{\mathrm{ON}}(l,i|\Xi) \log\left[p^{\mathrm{ON}}(l,i|\Xi)\right] + \right.$$
$$\left. + p^{\mathrm{OFF}}(l,i|\Xi) \log\left[p^{\mathrm{OFF}}(l,i|\Xi)\right] \right\}. \qquad (3)$$

2.2. When the entropy goes to zero

The evaluation of the entropy proposed in (3) is not intended as a measure of information flowing through the given layer, but it gives us an important indication of the effective use of the ReLU non-linearity. More specifically, for the i-th neuron in the l-th layer, we will have $\mathcal{H}(l,i|\Xi) = 0$ in two possible cases.

$s_{l,i,j}^{\xi} = 0, \forall j, \xi$. This case is achieved in three possible ways: either all the parameters $\boldsymbol{w}_{l,i}$ of the i-th neuron are zero, or the input $\boldsymbol{y}_{l-1}^{\xi}$ is zero $\forall \xi$, or again $z_{l,i,j}^{\xi} \leq 0, \forall j, \xi$. In these cases, the neuron's output will always be zero, and the entire neuron can be removed/pruned from the model, with no performance loss.

$s_{l,i,j}^{\xi} = 1, \forall j, \xi$. This case is achieved if $z_{l,i,j}^{\xi} > 0, \forall j, \xi$. In this state, the i-th neuron in the l-th layer uses only the linear region, becoming a linear neuron: its contribution can be "absorbed" by the next layer (as the neuron and the next layer are a linear combination), evidencing a collapse in the neural network's depth.

In this work we will be looking for both: while the first is expected, and known to rise in high pruning regimes [3, 23], the second one is more surprising, but can potentially lead to similar gains as those obtained with pruning. It is possible to extend the proposed formulation to other activations, like the sigmoid and GeLU (we will test on Swin-T in Sec. 3).

2.3. Entropy-based iterative pruning

As our target is to reduce the depth of deep neural networks by removing zero-entropy layers, we implement iterative pruning based on the entropy of different layers to get more layers with entropy zero.

Let us set the percentage of parameters to be pruned at each pruning iteration as ζ, and the total weight parameters of the considered L layers in the model as $\|\theta\|_0$. Then, in each pruning iteration, the number of weight parameters to be pruned $\|\theta\|_0^{\text{pruned}}$ is given by

$$\|\theta\|_0^{\text{pruned}} = \zeta \cdot \|\theta\|_0. \quad (4)$$

After calculating the entropy of each neuron belonging to the l-th layer, we can calculate an average entropy for it, simply through

$$\hat{\mathcal{H}}(l|\Xi) = \frac{1}{N_l} \sum_i \mathcal{H}(l, i|\Xi). \quad (5)$$

We would like to route the magnitude pruning algorithm towards removing more parameters in layers where the entropy is low. This is because manifesting a low entropy is a symptom of not necessarily requiring a non-linear activation in these layers, and for such reason, these layers are the most promising to remove. However, we know that $\mathcal{H}(l, i|\Xi) \geq 0 \forall l, i, \Xi$: this means that, when having $\hat{\mathcal{H}}(l|\Xi) = 0 \Leftrightarrow \mathcal{H}(l, i|\Xi) = 0 \forall i$. When this happens, we know that we are able to completely remove such a layer, and we no longer need to prune it.

In order to increase the number of layers with zero entropy, more pruning should be applied to layers with lower entropy (more likely to reach zero entropy). Simultaneously, to minimize the impact on model performance, more pruning should be done on smaller magnitude weights. To achieve both goals, we propose a *pruning irrelevance* meter $\mathcal{I}_l$ for the l-th layer

$$\mathcal{I}_l = \hat{\mathcal{H}}(l|\Xi) \cdot \frac{1}{\|\theta_l\|_0} \sum_i |\theta_{l,i}|, \quad (6)$$

where $\|\theta_l\|_0$ is the cardinality of the non-zero weights in the l-th layer. Effectively, the larger this value is, the least we are interested in removing parameters from it. However, we are very interested in removing parameters from layers having very low pruning irrelevance: for such reason, we
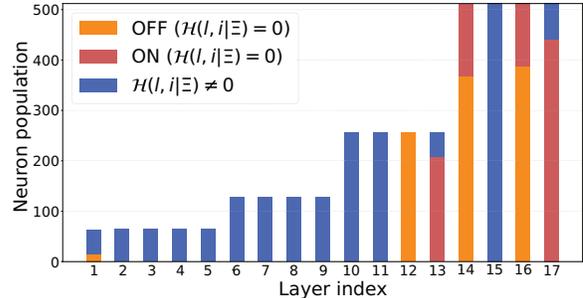

Figure 3: Distributions of neuron states per layer for ResNet-18 trained on CIFAR-10 with 98.4% of pruned parameters for EGP. In blue neurons having non-zero entropy, in orange always OFF, and in red always ON.

define the complementary measure of *pruning relevance*

$$\mathcal{R}_l = \begin{cases} \frac{\sum_j \mathcal{I}_j}{\mathcal{I}_l} & \mathcal{I}_l \neq 0 \\ 0 & \mathcal{I}_l = 0. \end{cases} \quad (7)$$

Finally, in order to assess the exact amount of parameters to be removed at the l-th layer we resort to a softmax smoothening, according to

$$\|\theta_l\|_0^{\text{pruned}} = \|\theta\|_0^{\text{pruned}} \cdot \frac{\exp[\mathcal{R}_l - \max_k(\mathcal{R}_k)]}{\sum_j \exp[\mathcal{R}_j - \max_k(\mathcal{R}_k)]}. \quad (8)$$

If the number of parameters pruned assigned to a specific layer exceeds its remaining parameter count, then prune all parameters of that layer. Afterward, distribute the remaining number of pruning parameters to other layers following the methods outlined in (6) and (8).

3. Experiments

In this section, we present our empirical results obtained on three different very common setups. We have performed our experiments on an NVIDIA GeForce RTX 2080 GPU and developed the code using PyTorch 1.13.1.[1]

Setup. We present results for ResNet-18 and Swin-T trained on CIFAR-10 and Tiny-ImageNet (Tint-INet). While for ResNet-18 ReLU is the non-linearity for the layers, in Swin-T the non-linear activation adopted is GELU. We adopt the same baseline training strategy as in [12]. Specifically for Swin-T, we prune just the GELU-activated layers (which are, for this model, 12). For all the sparse configurations, we set $\zeta = 0.5$.[2]

Results. The main results are reported in Table 1. Here, we compare a vanilla iterative magnitude pruning strategy

Table 1: Performance (Top-1) and number of layers removed (Lay. rem.) for different architecture/datasets.

Model	Dataset	Sparsity	EGP	Lay. rem.	Top-1
Resnet-18	CIFAR-10	0.0		0/17	92.10
		50.0		0/17	92.56
			✓	1/17	92.36
		75.0		0/17	93.00
			✓	1/17	92.81
		93.8		0/17	93.03
			✓	**3/17**	92.93
		98.4		0/17	92.73
			✓	**3/17**	**93.12**
	Tiny-INet	0.0		0/17	**41.88**
		50.0		0/17	41.70
			✓	2/17	38.96
		75.0		0/17	41.24
			✓	4/17	39.50
		93.8		0/17	41.86
			✓	4/17	39.80
		98.4		0/17	36.50
			✓	**5/17**	37.44
Swin-T	CIFAR-10	0.0		0/12	**92.11**
		50.0		0/12	91.64
			✓	1/12	92.08
		75.0		0/12	89.04
			✓	4/12	**92.11**
		93.8		0/12	84.02
			✓	5/12	91.01
		98.4		0/12	90.42
			✓	**7/12**	90.10
	Tiny-INet	0.0		0/12	**75.38**
		50.0		0/12	74.06
			✓	1/12	71.48
		75.0		0/12	72.02
			✓	2/12	70.28
		93.8		0/12	67.58
			✓	3/12	66.58
		98.4		0/12	63.46
			✓	**6/12**	62.30

Table 2: Performance of entropy-based pruned model and reinitialized model.

Model	Dataset	Lay. rem.	Method	Top-1
Resnet-18	CIFAR-10	5/17	from scratch	91.10
			EGP	**92.18**
	Tiny-ImageNet	4/17	from scratch	37.72
			EGP	**39.80**
Swin-T	CIFAR-10	5/12	from scratch	61.00
			EGP	**90.41**
	Tiny-ImageNet	1/12	from scratch	0.50
			EGP	**71.48**

also neurons in the ON state, would be overlooked: this shows the effectiveness of EGP and potentially opens the road to a new class of pruning algorithms.

Ablation study. We propose here an ablation study on the effectiveness of EGP. Table 2 compares training from scratch a model with some layers removed to the same following EGP: we observe that in all cases EGP exhibits better Top-1 performance. Especially for Swin-T trained on Tiny-ImageNet, this architecture/dataset combination does not provide good results as the smaller model should be also pre-trained on a large dataset like the vanilla one. This phenomenon suggests that EGP enhances accessibility to rare, compressed states of the deep model.

4. Conclusion

In this work, we have investigated the impact of unstructured pruning on the entropy of activations in DNNs. Our preliminary experiments on ReLU and GELU-activated models show that our pruning strategy, EGP, drives a consistent part of the neurons in the model to either ON or OFF regions. Unstructured pruning can provide a structured activation pattern and yes, it can reduce the model's depth! Our results provide new insights into the mechanisms behind the benefits of pruning and open avenues for further exploration into the relationship between entropy, pruning, and deep learning performance.

Future work can explore the applicability of our findings to other activation functions and architectures as well as the design of a regularization function explicitly enforcing low entropy to favor the model's compressibility. Exploring the impact of pruning on the dynamics of the optimization process and the interpretability of the resulting models could lead to a better understanding of the mechanisms behind the benefits of pruning.

with EGP. It appears evident that plugging EGP as a scaling factor to drive the pruning process effectively removes entire layers from the model. Evidently, with a higher sparsity, a performance drop naturally occurs; however, the gap in performance between vanilla pruning and EGP (for the same sparsity) remains consistently narrow. We provide, in Fig. 3, a visualization of the activation distributions for CIFAR-10 at 98.4% sparsity for EGP. Interestingly, we observe that one layer only (#12) can be removed with traditional pruning approaches, while others (like #14), having

References

[1] Florian Beye, Hayato Itsumi, Charvi Vitthal, and Koichi Ni-hei. Recognition-aware deep video compression for remote surveillance. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1986–1990. IEEE, 2022. 1

[2] Daniel Bogdoll, Maximilian Nitsche, and J Marius Zöllner. Anomaly detection in autonomous driving: A survey. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4488–4499, 2022. 1

[3] Andrea Bragagnolo, Enzo Tartaglione, Attilio Fiandrotti, and Marco Grangetto. On the role of structured pruning for neural network compression. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3527–3531. IEEE, 2021. 1, 3

[4] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16306–16316, 2021. 1

[5] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020. 1

[6] 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. 1

[7] Shuxiang Fan, Xiaoting Liang, Wenqian Huang, Vincent Jialong Zhang, Qi Pang, Xin He, Lianjie Li, and Chi Zhang. Real-time defects detection for apple sorting using nir cameras with pruning-based yolov4 network. *Computers and Electronics in Agriculture*, 193:106715, 2022. 1

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. 1

[9] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022. 1

[10] 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, 2015. 1

[11] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5:164–171, 1993. 1

[12] Zheng He, Zeke Xie, Quanzhi Zhu, and Zengchang Qin. Sparse double descent: Where network pruning aggravates overfitting. In *International Conference on Machine Learning*, pages 8635–8659. PMLR, 2022. 3

[13] Cheonghwan Hur and Sanggil Kang. Entropy-based pruning method for convolutional neural networks. *The Journal of Supercomputing*, 75:2950–2963, 2019. 1, 2

[14] Olivier Laurent, Adrien Lafage, Enzo Tartaglione, Geoffrey Daniel, Jean marc Martinez, Andrei Bursuc, and Gianni Franchi. Packed ensembles for efficient uncertainty estimation. In *International Conference on Learning Representations*, 2023. 1

[15] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. *Advances in neural information processing systems*, 2:598–605, 1990. 1

[16] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1

[17] Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, and Luc Van Gool. Revisiting random channel pruning for neural network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 191–201, 2022. 1

[18] Jian-Hao Luo and Jianxin Wu. An entropy-based pruning method for cnn compression, 2017. 2

[19] Chuhan Min, Aosen Wang, Yiran Chen, Wenyao Xu, and Xin Chen. 2pfpce: Two-phase filter pruning based on conditional entropy, 2018. 2

[20] Dmitry Molchanov, Stephen Tyree, Tero Karras, and Timo Aila. Variational dropout and the local reparameterization trick. *International Conference on Learning Representations*, 2017. 1

[21] Riccardo Renzulli and Marco Grangetto. Towards efficient capsule networks. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2801–2805. IEEE, 2022. 1

[22] Enzo Tartaglione. The rise of the lottery heroes: why zero-shot pruning is hard. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2361–2365. IEEE, 2022. 1

[23] Enzo Tartaglione, Andrea Bragagnolo, Attilio Fiandrotti, and Marco Grangetto. Loss-based sensitivity regularization: towards deep sparse neural networks. *Neural Networks*, 146:230–237, 2022. 1, 3

[24] Enzo Tartaglione, Stéphane Lathuilière, Attilio Fiandrotti, Marco Cagnazzo, and Marco Grangetto. Hemp: high-order entropy minimization for neural network compression. *Neurocomputing*, 461:244–253, 2021. 1

[25] Huichen Yang and William Hsu. Transformer-based approach for document layout understanding. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 4043–4047. IEEE, 2022. 1

[26] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7308–7316, 2019. 1

[27] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3143–3151, 2022. 1