How Many Does It Take to Prune a Network: Comparing One-Shot vs. Iterative Pruning Regimes

Mikołaj Janusz* **Tomasz Wojnar** **Luca Benini** **Yawei Li** **Kamil Adamczewski**
Jagiellonian University ETH Zürich IDEAS NCBR

Abstract

Pruning is one of the classical methods for network compression and model acceleration. There are two main ways pruning can be implemented: iterative pruning and one-shot pruning. In the literature, iterative pruning appears to have more proponents; however, the support for this method is often taken for granted without any substantial justification. Conversely, other approaches favour one-shot pruning. Despite the long history of pruning, there has been no thorough comparison of these two techniques. In this work, we conduct a broad and comprehensive benchmark to evaluate the regimes in both structured and unstructured pruning setting. Our findings indicate that we cannot conclusively determine which method is superior, but present scenarios where one-shot pruning is preferable for lower pruning ratios while iterative methods in higher pruning rates. This study also proposes a hybrid approach that outperforms both one-shot and traditional iterative pruning.

1 Introduction

Pruning is a crucial technique used to reduce the size of these networks without significantly compromising their performance [39, 2, 14, 20, 28, 27, 6, 31]. By cutting down the excess parts of a network, we can achieve results comparable to those obtained with fully trained networks, with a reduced size of the model, which improved efficiency and reduces the cost of inference. Pruning techniques can be broadly classified into two categories: one-shot pruning and iterative pruning. Despite the prevalence of these methods, a systematic comparison of these two approaches under various regimes is lacking in the literature. Previous investigation is centred around either the pruning criteria [39, 21] or the pruning methods [28, 3, 37] themselves. This lack of comparative analysis leaves a gap in understanding how different pruning regimes perform under varying conditions and configurations.

In this work, we thoroughly examine and compare the performance of one-shot and iterative pruning. This work organizes and synthesizes various pruning regimes found in the literature, providing a clearer framework for future research. A key contribution is the introduction of a geometric pruning ratio scheduler, which prunes a fixed percentage of remaining weights at each step, demonstrating its effectiveness in iterative pruning. The work also proposes hybrid pruning methods, highlighting their potential advantages in specific contexts. Finally, it emphasizes the importance of retraining and iterative pruning rates in pruning algorithms, advocating for a patience-based fine-tuning approach.

We summarize the key findings of this work: 1. One-shot pruning can perform better than iterative pruning for lower pruning rates, and iterative pruning is better for higher rates. 2. Iterative geometric pruning is superior to iterative pruning constant in most cases. 3. Using early stopping allows for a more suitable fine-tuning time. 4. Number of retraining iterations matter. 5. One-shot pruning in many cases requires less overall retraining time compared to iterative pruning, as it avoids the need for multiple cycles of pruning and retraining. This can be beneficial when computational resources are limited.

*Correspondence to mikolaj1.janusz@student.uj.edu.pl

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

2 Pruning regimes

In the context of neural network optimization, pruning regimes can be split into categories based on the following question:*Should we prune all the weight at once, or should we divide the pruning between different iterations that are separated by modifying the structure of the network and weight updates?*

For the purpose of the comparison of the pruning regimes, assume that W is the total number of weights in the neural network and p is the desired pruning percentage. E.g. $p = 0.8$ means a pruning of 80% of weights.

- **One-shot Pruning:** One-shot pruning involves removing a specified percentage of weights (or neurons) in a single step. The least significant weights are eliminated based on a predetermined importance criterion. The final number of weights remaining after one-shot pruning is given by:
$$(1 - p) \times W$$
- **Iterative Pruning:** Iterative pruning is a process where weights are pruned over multiple iterations, allowing the network to gradually retrain and recover some performance loss. In each iteration the ranking, pruning, and fine-tuning occurs. We define two common approaches to iterative pruning and name them differently to avoid confusion.
  - **Iterative Constant:** A constant number of parameters is pruned at each step. Let *steps* be the number of iterations, then $\frac{p \times W}{\text{steps}}$ weights are pruned at each stage. The pruned percentage per step is fixed in relation to the initial number of weights.
  - **Iterative Geometric Pruning**: A fixed percentage p of the **remaining** weights is pruned at each step, meaning that as the pruning process progresses, progressively fewer weights are pruned.
    The number of weights at step n is given by the following formula: $W \times (1 - \frac{p}{steps})^n$
- **Hybrid pruning:** We propose a new pruning regime, hybrid (few-shot) pruning, which is a combination of the idea of one-shot and geometric regimes. The majority of the weights are removed at the first one-shot like step and the model is retrained for a longer time. Then for the remaining weights, we perform a more fine-grained geometric-like pruning over several iterations.

  The hybrid pruning approach can be summarized as follows:
  1. One-shot pruning to a given pruning percentage p_k with a longer fine-tuning.
  2. Iterative geometric pruning from p_k to a desired final pruning percentage p.

3 Pruning regime factors

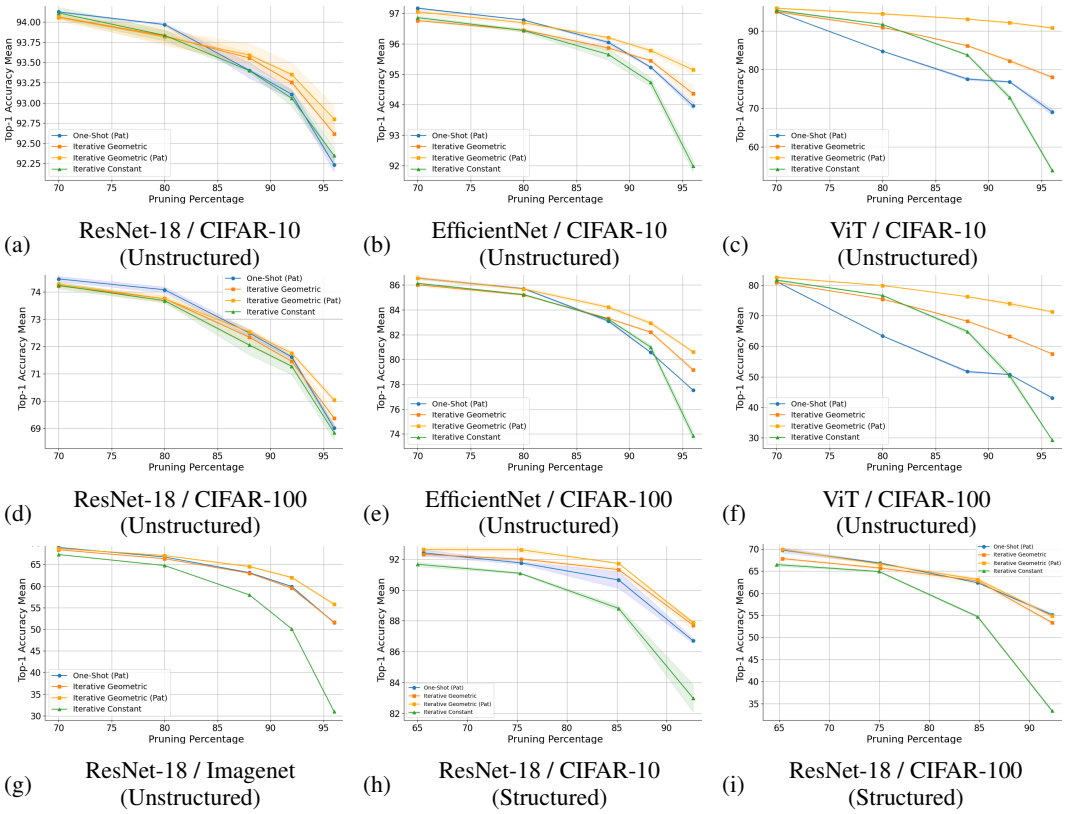
Pruning regimes, both one-shot and iterative, come with parameters that have decisive impact on their performance. Therefore, to properly evaluate the regimes, we shall take them into account. Besides, see Appendix for an analysis of these factors in pruning methods in literature.

3.1 Fine-tuning

Fine-tuning is a crucial step in the pruning process, serving to recover the performance of a neural network after portions of its weights have been removed. The necessity of retraining arises because the initial pruning can degrade the model's accuracy by eliminating parameters that the network had previously relied on for making predictions.

Retraining is necessary both in case of one-shot and iterative pruning but can vary in length. The larger the drop in accuracy, the longer the fine-tuning should last, and the drop in accuracy correlates with the number of parameters that are pruned and their utility. Consequently, the fine-tuning phase is longer for one-shot and shorter for each iteration in case of iterative pruning.

The length of the update in both one-shot and iterative has not, however, been properly researched, with each method selecting its own value. For example, [33] performs fine-tuning for the original training time, which may be computationally wasteful as the network is not trained from scratch. On the other hand, in [30, 4] the model is fine-tuned for one epoch after pruning in each iteration.



(a) ResNet-18 / CIFAR-10 (Unstructured) (b) EfficientNet / CIFAR-10 (Unstructured) (c) ViT / CIFAR-10 (Unstructured)

(d) ResNet-18 / CIFAR-100 (Unstructured) (e) EfficientNet / CIFAR-100 (Unstructured) (f) ViT / CIFAR-100 (Unstructured)

(g) ResNet-18 / Imagenet (Unstructured) (h) ResNet-18 / CIFAR-10 (Structured) (i) ResNet-18 / CIFAR-100 (Structured)

Figure 1: Comparison of one-shot pruning and iterative pruning across various network architectures and datasets. The iterative pruning comes in two types: constant and geometric. The plots show the benefits of patience-based pruning (Pat) (see Appendix for more details on patience)

3.2 Iteration pruning rate.

In pruning, the inherent parameter is final pruning percentage, additionally we need to set the number of iterations or the pruning rate per iteration. Given the values from literature, we look at a range of pruning rates from single to double-digit iterative pruning rates for both constant and geometric iterative pruning. In case of the constant pruning, the iterative rate should divide the final pruning rate the without a remainder. In case of the geometric pruning rate, the amount of pruned weights are each iteration decreases and the rate is determined so that the geometric sum of the pruning rates at each iteration sum up to perform a whole number of iterations.

4 Empirical evaluation

We perform experiments on several datasets and model architectures. The datasets include CIFAR-10 [17], CIFAR-100 and Imagenet1K [5]. The experiments are performed both on convolutional neural networks and transformers, in particular ResNet [12], EfficientNet [34] and Visual Transformer [7]. As recommended in [10], we use $1/10^{\text{th}}$ of the original learning rate for the fine-tuning phase.

4.1 One-shot patience, fixed iterative retraining and Iterative patience

In this experiment, we show that one-shot pruning can be effective when provided with an appropriate retraining length, and may outperform iterative pruning. In one-shot pruning, we apply patience-based retraining, which allows the pruned model to terminate when no improvements are seen over a specified number of epochs. Patience-based retraining is a more natural way than setting a fixed number of epochs, which can turn out to be too little or too much and avoids more than one retraining to find the optimal number of epochs.

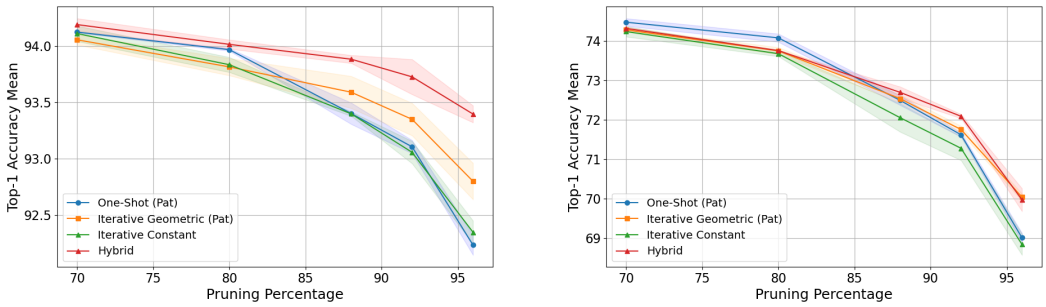
This is in contrast with iterative pruning that is often given a fixed fine-tuning phase, with sometimes as little as one fine-tuning epoch [4]. Here, we look through a set of fixed values for both iterative pruning regimes, and select the best result, which is plotted in Figure 1. The experimental results show that short fixed fine-tuning phase in case of iterative pruning underperforms patience-based pruning. The experimental results show that one-shot pruning outperforms iterative pruning, especially for pruning rates lower than 80%. This consistent result across datasets shows that when the pruning level is lower (that is, less than 80%), performing iterations is unnecessary and all the weights can be removed at one shot. The geometric iterative pruning performs better for higher pruning ratios and in case of transformers and structured pruning.

Subsequently, we compare one-shot pruning with iterative pruning when both regimes utilize the patience-based retraining. We proposed patience-based iterative pruning as a more effective alternative of iterative pruning fine-tuning. Since at every iteration a smaller fraction of weights is pruned, we also set the patience relatively smaller compared to the one-shot to make complete retraining length manageable. As seen in Figure 1, patience-based iterative pruning leverages iterative pruning to higher accuracies, in particular for high-pruning ratios. In Appendix we also take into account the retraining budget in addition to the pruning rate and pruning accuracy

4.2 Hybrid (Few-Shot)

The observations from this benchmark study have led us to conclude that for lower pruning ratios, performing pruning in one-shot is more beneficial than iterative pruning. Inspired by this finding, we propose a hybrid few-shot regime. Hybrid pruning combines the two modes of training to prune a large part of the network in a one-shot-like manner, and then follows with a more tuned approach that used geometric pruning. This combination leads to results that outperform all the benchmarks regimes and provides insights into a better way of pruning.

The results are presented in Figure 2 and shows that the hybrid approach is the preferable choice across almost all pruning rates. Hybrid pruning carries the benefits of both one-shot and iterative pruning. It removes the majority of weights in the first iteration, avoiding redundant iteration in the early stage of the pruning process. At the same time, the hybrid approach retains the sensibility of geometric iterations, where at the higher pruning rates the remaining weights carry more importance and require more fine-grained consideration.



(a) ResNet-18 / CIFAR-10 (b) ResNet-18 / CIFAR-100

Figure 2: Hybrid approach in comparison with one-shot and iterative pruning.

5 Conclusions

One-shot pruning can perform better than iterative pruning for lower pruning rates, and iterative pruning is better for higher rates, especially considering the fine-tuning length and computational cost. Iterative geometric pruning is superior to iterative pruning constant in most cases. Using patience allows for higher accuracy recovery than small fixed-epoch iterative pruning and is more efficient than setting up a very high number of epochs, it is also a more natural and easier way to set the length of fine-tuning. One-shot pruning in many cases requires less overall retraining time compared to iterative pruning, as it avoids the need for multiple cycles of pruning and retraining.

6 Acknowledgements

The research was funded by the program Excellence Initiative - Research University at the Jagiellonian University in Kraków. We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2024/017173

Appendix

A Pruning regimes in literature

Pruning is widely regarded as one of the oldest and most established techniques for compressing neural networks. Historical research has shown that training large networks and subsequently removing weights can significantly enhance computational and parameter size efficiency [18, 11]. Despite this, the literature does not present a unified procedure for network pruning, with two main approaches identified: one-shot pruning and iterative pruning. These methods typically focus more on the criterion for pruning, such as magnitude or derivative-based methods, and less on the specifics of how pruning is implemented. Often, descriptions of pruning procedures do not include details like the exact percentage of weights removed in each iteration. However, as argued in Section 4, the pruning percentage is a critical aspect that influences the number of pruning epochs and ultimately affects performance.

We provide a summary of common pruning methods and their training protocols in Table 1

One-shot pruning. One-shot pruning requires one cycle of pruning and retraining and is appreciated in literature due to its computational efficiency. One-shot pruning requires computing node ranking only once, leading to the final shape of the network in one step.

One-shot pruning is often applied in methods for which the cost of pruning is high. [13] removes filters near the geometric median, claiming that they can be represented by the remaining filters. The geometric median is a non-trivial problem in computational geometry and is acknowledged to be computationally expensive. On the other hand, Dirichlet pruning [1] requires a phase of training to compute the parameters of Dirichlet distribution, which are importance weights. Then CURL [29] removes all unimportant filters of all layers at once via KL-divergence based criterion.

Once the pruning phase is done, the retraining follows. The length of retraining has not been properly research before, either. The methods often fix the number of retraining epochs to an arbitrary number. For example, the above-mentioned CURL fine-tunes the model for 100 epochs.

Iterative pruning. Iterative pruning has emerged as a method for optimizing neural network architectures by gradually removing less important parameters or structures and fine-tuning the network to maintain performance. Unlike one-shot pruning, which removes a significant portion of weights or neurons in a single step, iterative pruning involves multiple cycles of pruning and retraining.

For large networks and most of the methods, iterative pruning is prohibitive for unstructured pruning due to the large number of single parameters. For small networks, the concept of iterative pruning dates back to early works such as Optimal Brain Damage (OBD) [11] and Optimal Brain Surgeon (OBS) [11], where unimportant connections are identified and removed based on second-order derivatives of the loss function with respect to the weights. Due to computational costs of computing pruning criterion that can be high and when the forward passes are required, either only one [36] or a few [24] batches (iterations) are used to assess the parameter sensitivity.

In iterative pruning, many works select a constant pruning rate, that is a fixed percentage, that is applied at each iteration. The pruning rate, however, varies across different methods. [31] uses first and second-order Taylor expansions to approximate a filter's contribution. The method removes 10 neurons every 30 mini-batches until the predefined number of pruned neurons is reached. Then fine-tuning is done for a total of 25 epochs. In [33] the authors prune 20% lowest-magnitude weights globally and then retrain the network using learning rate rewinding followed by retraining for the original training time. [21] searches for optimal sub-architecture randomly. The method removes 10% of the weights, which is followed by fine-tuning phase.

The number of pruned elements can also be selected based on various criteria that are hyperparameters of the method. In their work, [10] selects the number of pruned elements based on a custom threshold. All the elements whose magnitude is smaller than the threshold are removed. For [26] in each pruning iteration, the algorithm identifies and removes the least important channel(s) based on the calculated Fisher Information scores. The number of channels can vary based on the importance scores calculated. Then [24] describes that the pruning ratio follows a hyperharmonic sequence,

Method	structure	regime	step
HRank [25]	structured	iterative (custom)	
ThiNet [30]	structured	iterative (custom)	
SSS [16]	structured	iterative (unspecified)	
Revisiting Random Pruning (RRCP) [21]	structured	iterative (constant)	10%
Fisher information [36]	structured	iterative	
Learning rate rewinding [33]	both	iterative (constant)	20%
Optimal brain damage [18]	unstructured	iterative	
Optimal brain surgeon [11]	unstructured	iterative	
Learning weights and connections [10]	unstructured	iterative	
Taylor Expansion [31]	structured	iterative (constant)	2%
Group Fisher Information [22]	structured	iterative (custom)	
Empirical Sensitivity Analysis [24]	structured	iterative (custom)	
CURL (KL-divergence metric) [29]	structured	one-shot	
Dirichlet Pruning [1]	structured	one-shot	
Geometric Median [13]	structured	one-shot	

Table 1: A list of selected pruning methods and their corresponding training regimes

where the i-th pruning ratio is determined by $1 - \frac{1}{(i+1)^{\alpha}}$. [25] prunes the predefined number of channels but does it layer-wise, fine-tuning the network after pruning each layer. Moreover, [30] use the statistics from the next layer to prune the current layer. The network is pruned by layer with different compression rates. The method fine-tunes one or two epochs after the pruning of one layer.

Pruning criteria. In this work, we perform all the experiments using magnitude-derived criteria that include weight magnitude for unstructured pruning and L1 and L2-norms for structured pruning. It should be noted that although magnitude-based pruning is not the only pruning technique, the magnitude-based criteria are the most widespread pruning criteria applied across a variety of methods [9, 10, 33]. Moreover, these approaches are the most frequently used and have consistently proven to be reliable and effective, as highlighted in the results presented by [15]. Finally, due to the low cost of computing parameter importance, magnitude-based criteria are essential for benchmarking as they allow us to perform a wide range of experiments that would not be possible for other criteria.

Pruning at initialization and during training. While this work focuses on the post-train pruning that uses the train-prune-retrain scheme, we acknowledge other types of pruning. Pruning at initialization [8, 19, 35] does not prune a trained model, instead it looks to extracting a smaller model and training it from scratch to achieve similar performance to the larger model.

Another group of methods performs pruning during training by introducing regularization that pushes some parameters to zero [38, 32]. For example, [38] introduces scaling factors that scale the outputs of certain structures in the CNNs. Sparse regularizations are applied to these scaling factors to phase them out during training.

A.1 Unstructured and structured pruning

Pruning networks can be done for individual weights or for structures within the network. Each of them is important in its own regard and broadly researched. Hence, this benchmark includes tests where both unstructured and structured pruning are considered.

Unstructured pruning. Unstructured pruning involves selectively removing individual weights from the neural network based on certain criteria, such as the magnitude of the weights or their impact on the loss function [9, 8]. This method creates sparse weight matrices, where many elements are zero. Although it can significantly reduce the number of parameters, it often requires specialized hardware or software optimizations to achieve practical computational benefits because the remaining weights are distributed irregularly throughout the network.

Structured pruning. Structured pruning, on the other hand, removes entire structures within the neural network, such as filters, channels, neurons, or even layers [14, 39, 28, 22, 23]. This method results in a more compact and regular network architecture that retains its original dense matrix

structure, making it easier to implement and optimize on standard hardware. Structured pruning can lead to significant reductions in both the model size and computational requirements, while maintaining a more organized and efficient network. Applying structured pruning at high sparsity ratios poses greater challenges compared to unstructured methods, as it involves eliminating entire rows and columns instead of just individual elements within a weight matrix.

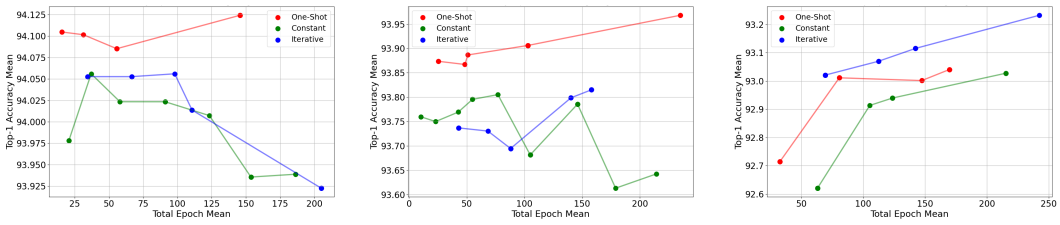
The presented regimes in the next section can apply to both types of pruning, unstructured and structured. However, the implementation may differ as a result of constraints imposed by the structure of the pruned entity.

B Retraining budget

In this section, we take into account the retraining budget in addition to the pruning rate and pruning accuracy. We ask ourselves the question: *For a given pruning rate and computational budget, what is the method that provides us with the best performance?* We present three plots for three different pruning rates and compare the budgets used by one-shot and iterative pruning to achieve a given accuracy. The presented results are for ResNet architecture trained on CIFAR-10. Please see Appendix for more plots.

The budget is given in the number of retraining epochs. In the case of one-shot pruning, this is a single sequence of epochs. In the case of iterative pruning, we sum the epochs trained over all the iterations.

As Figure 3 shows, one-shot pruning is the most efficient regime for pruning rate, up to 80 percent across all computational budgets. In both, it achieves higher accuracy across the range of the total epochs. However, for higher pruning rates the iterative pruning performs improves and would be the method of choice.



(a) Pruning rate 70% (b) Pruning rate 80% (c) Pruning rate 92%

Figure 3: The performance of training regimes for fixed computational budget, given in terms of total number of epochs. One-shot is more efficient for pruning rates below 80% while iterative geometric for higher pruning rates.

C Early-stopping

In this work, we advocate the use of patience or early-stopping to select the number of fine-tuning epochs. We apply patience both for one-shot pruning and iterative pruning. Patience allows gauging the number of required fine-tuning epochs. During fine-tuning, we keep the best checkpoint and keep the training until the given criterion does not improve. Then we revert to the best checkpoint. The advantage of patience over fixed number of epochs is that it is hard to predict the required number of fine-tuning epochs to achieve a good result, patience allows gauging the length of the fine-tuning. Instead of specifying the number of fine-tuning epochs as it is common in literature, we define the patience, the number of epochs the fine-tuning continues before the given criteria does not improve. The details of the proposed patience algorithm are given in Algorithm 1.

Algorithm 1 Early Stopping Check

Note: This code assumes that a lower metric value indicates better performance (e.g., loss). Otherwise, if a higher metric value is better (e.g., accuracy) the code is run with a reversed comparison.

procedure EARLYSTOP($metric\_value : float$)
    if $metric\_value < self.best\_metric\_value$ **then**
        $self.best\_metric\_value \leftarrow metric\_value$
        $self.counter \leftarrow 0$
    else if $metric\_value > (self.best\_metric\_value + self.min\_delta)$ **then**
        $self.counter \leftarrow self.counter + 1$
    end if
    if $self.counter \geq self.patience$ **then**
        return True
    end if
    return False
end procedure

References

[1] Kamil Adamczewski and Mijung Park. Dirichlet pruning for neural network compression. *The 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

[2] Changan Chen, Frederick Tung, Naveen Vedula, and Greg Mori. Constraint-aware deep neural network compression. In *Proceeding of the European Conference on Computer Vision*, pages 400–415, 2018.

[3] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations. *arXiv preprint arXiv:2308.06767*, 2023.

[4] Elliot J Crowley, Jack Turner, Amos Storkey, and Michael O'Boyle. A closer look at structured pruning for neural network compression. *arXiv preprint arXiv:1810.04622*, 2018.

[5] 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. IEEE, 2009.

[6] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal SGD for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4943–4953, 2019.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

[9] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *Proceedings of International Conference on Learning Representations*, 2015.

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

[11] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, pages 164–171, 1993.

[12] 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.

[13] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019.

[14] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for model compression and acceleration on mobile devices. In *Proceeding of the European Conference on Computer Vision*, pages 784–800, 2018.

[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] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceeding of the European Conference on Computer Vision*, pages 304–320, 2018.

[17] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[18] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605, 1990.

[19] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. SNIP: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[20] Jiashi Li, Qi Qi, Jingyu Wang, Ce Ge, Yujian Li, Zhangzhang Yue, and Haifeng Sun. OICSR: Out-in-channel sparsity regularization for compact deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7046–7055, 2019.

[21] 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 International Conference on Computer Vision*, 2022.

[22] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[23] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. DHP: Differentiable meta pruning via hypernetworks. In *Proceeding of the European Conference on Computer Vision*, pages 608–624. Springer, 2020.

[24] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. *arXiv preprint arXiv:1911.07412*, 2019.

[25] 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.

[26] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*, pages 7021–7032. PMLR, 2021.

[27] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. MetaPruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[28] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *Proceedings of International Conference on Learning Representations*, 2019.

[29] Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1458–1467, 2020.

[30] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[31] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019.

[32] Changyong Oh, Kamil Adamczewski, and Mijung Park. Radial and directional posteriors for bayesian deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

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

[34] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[35] Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[36] Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. *arXiv preprint arXiv:1801.05787*, 2018.

[37] Huan Wang, Can Qin, Yue Bai, Yulun Zhang, and Yun Fu. Recent advances on neural network pruning at initialization. *arXiv preprint arXiv:2103.06460*, 2021.

[38] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *Proceedings of the IEEE/CVF Conference on CVPR*, pages 14913–14922, 2021.

[39] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *Proceedings of International Conference on Learning Representations*, 2018.