Evaluating the Emergence of Winning Tickets by Structured Pruning of Convolutional Networks

Whendell F. Magalhães*, Jeferson Ferreira*, Herman M. Gomes*, Leandro B. Marinho* and Plínio Silveira†
*Universidade Federal de Campina Grande, Department of Systems and Computing
Av. Aprígio Veloso 882, 58429-900 Campina Grande-PB, Brazil
†Hewlett Packard Enterprise, Brazil
Email: whendell@copin.ufcg.edu.br

***Abstract*—The recently introduced Lottery Ticket Hypothesis has created a new investigation front in neural network pruning. The hypothesis states that it is possible to find subnetworks with high generalization capabilities (winning tickets) from an over-parameterized neural network. One step of the algorithm implementing the hypothesis requires resetting the weights of the pruned network to their initial random values. More recent variations of this step may involve: (i) resetting the weights to the values they had at an early epoch of the unpruned network training, or (ii) keeping the final training weights and resetting only the learning rate schedule. Despite some studies have investigated the above variations, mostly with unstructured pruning, we do not know of existing evaluations focusing on structured pruning regarding local and global pruning variations. In this context, this paper presents novel empirical evidence that it is possible to obtain winning tickets when performing structured pruning of convolutional neural networks. We setup an experiment using the VGG-16 network trained on the CIFAR-10 dataset and compared networks (pruned at different compression levels) got by weight rewinding and learning rate rewinding methods, under local and global pruning regimes. We use the unpruned network as baseline and also compare the resulting pruned networks with their versions trained with randomly initialized weights. Overall, local pruning failed to find winning tickets for both rewinding methods. When using global pruning, weight rewinding produced a few winning tickets (limited to low pruning levels only) and performed nearly the same or worse compared to random initialization. Learning rate rewinding, under global pruning, produced the best results, since it has found winning tickets at most pruning levels and outperformed the baseline.**

***Index Terms*—neural network compression, structured pruning, winning tickets, weight rewinding, learning rate rewinding**

I. Introduction

Solutions based on Deep Learning, more specifically Deep Neural Networks (DNNs), are the state-of-the-art for a wide range of AI-based complex tasks, such as computer vision, natural language processing, neural machine translation, speech recognition and several others [1]. However, such tasks usually rely on complex and continuously evolving DNN architectures, which are often over-parameterized [2]–[4]. By over-parametrized, we mean DNNs in which the number of parameters is vast and sometimes even surpasses the number of input features or the number of training instances. As a result, such DNNs have high computational costs for both training and inference. For comparison, LeNet-5, one of the first convolutional neural networks, introduced by LeCun *et al.* (1998) [5], has approximately 60 thousand parameters and was designed for the MNIST dataset which has 60 thousand training 28x28 grayscale images. Deeper neural networks easily reach hundreds of millions of parameters, such as the VGG-16 architecture introduced in [6], which has approximately 138 million parameters and was designed for the ILSVRC-2014 [7], which has 1 million training color images with an average resolution of 482x415.

It is worth notice that over-parametrization is well known for helping local search algorithms reach low training error since it favors the creation of a large manifold of globally optimal solutions during training [8]. Moreover, over-parametrization, combined with regularization, may help achieve better generalization. Given the considerable search space involved in building tailored DNN architectures for specific tasks, it is common practice to select off-the-shelf architectures well known for working well in various complex tasks. If the selected model is more complex than needed, one can apply techniques to reduce model complexity (e.g., regularization) while retaining (or even improving in some cases) the original model's generalization power, as explained in more detail in the sequel.

Due to the resource-hungry nature of DNNs, running them on computers with limited resources, such as edge devices based on commodity hardware, is often unfeasible. The execution of such DNNs is usually associated with specialized hardware such as general purpose graphic processing units (GPUs) or the more specialized tensor processing units (TPUs). Nevertheless, many works have appeared in recent years proposing novel techniques to enable the training and inference of such complex DNNs on commodity hardware. This is an emerging research area that is commonly referred to as neural network compression and acceleration [9].

Among the existing compression techniques, Neural Network Pruning is a well-established family of techniques aiming at reducing storage consumption, memory footprint, and computational resources usage without significantly harming the DNN accuracy, even for aggressive pruning levels [10]. Such techniques consist of removing unnecessary components (and corresponding weights) from a neural network according to a given criterion. A wide range of pruning techniques are structured according to these three core steps:

(1) Train the neural network until convergence.

(2) Prune the neural network to a specific sparsity level according to some heuristic.
(3) Retrain the pruned network to compensate for any eventual loss of accuracy and generalization power.

In most of the existing techniques [11]–[14], the standard procedure used for retraining a pruned network is fine-tuning. Starting with the weights inherited from the original neural network, fine-tuning comprises further training the pruned network using a small fraction of the training epochs, used in the original model, while using the same learning rate from the final epochs of the original neural network's training.

According to Liu *et al.* (2019) [15], the above three steps have support on two common beliefs. The first one is that over-parameterized DNNs train better than those with fewer parameters, because of the high generalization capacity of larger models [16]. After training, a set of redundant weights could then be pruned from the over-parameterized network without degrading its accuracy. Previous work, such as [13], [17], has reported this procedure to be better than training a smaller network from scratch. The second belief, as discussed by Han *et al.* (2015) [11], is that not only the resulting architecture got by pruning but also its associated weights are essential for producing the final compressed model. From this, it is easy to see why a great deal of pruning techniques prefer to fine-tune a pruned model instead of training it from random weights.

Some more recent works found that retraining a pruned network from scratch can achieve the same accuracy and sometimes even outperform fine-tuned DNNs. These works refer to the pruned networks that match or even outperform their unpruned counterparts as winning tickets [10], [18]–[21] in reference to The Lottery Ticket Hypothesis, which is further discussed in Section III. An important step for implementing this hypothesis is to reset the weights of the pruned network to their initial random values. Follow up alternatives for this step are: (i) resetting the weights to the values they had at an early epoch of the unpruned network training, or (ii) keeping the final training weights and resetting only the learning rate schedule.

We can categorize pruning according to different criteria. A well-known distinction refers to structured versus unstructured pruning [22]. Unstructured pruning can remove parameters from a DNN without a specific geometry or constraint, which is difficult to compute efficiently in most existing hardware. For example, in a fully connected layer, removing some individual connections to a neuron (which is usually done by zeroing those connections) does not bring any reduction in inference time, since the MVM (Matrix-Vector Multiplication) still needs to take place to compute layer-outputs. Structured pruning assures the remaining parameters are kept at well-defined locations. For instance, in a convolutional neural network, structured pruning may remove an entire filter kernel but not individual kernel weights, thus preventing the formation of sparse filters. Here, the MVM assigned to the removed filter is not needed anymore, which saves CPUs or GPUs cycles, running models more efficiently. We may also classify a pruning method as local or global, which relates to removing the elements/parameters when taking (or not) into account their location in the network. Whenever the parameters are removed proportionally in all layers of the network, we say the pruning is local, otherwise, it will be global.

Recent studies have shown that DNNs can be pruned for aggressive levels of compression without harming the final accuracy of the pruned network [18], [20], [21]. However, although impressive, such results were observed in the contexts of: (i) unstructured pruning [18], which produces pruned networks without real performance gains in general purpose hardware; and (ii) structured pruning with predefined optimal architecture [20], which is unfeasible for most applications due to the large search space of the pruning rates per layer in deep neural networks. In addition to that, we do not know of existing studies comparing local and global pruning variations under the Lottery Ticket Hypothesis.

This paper contributes with novel empirical evidence that supports obtaining winning tickets when performing structured pruning of convolutional neural networks. We experimentally evaluate and discuss the emergence of winning tickets using Weight Rewinding and Learning Rate Rewinding on respect to iterative structured pruned networks with automatic architecture definition [13], [23] and compare the results of pruned networks found using these retraining approaches with the same networks trained with randomly initialized weights.

The remainder of this paper is organized as follows. Section II introduces some definitions aiming at a better understanding of the concepts used in this paper. Section III contains a discussion of related work. Section IV presents the experimental protocol and the obtained results. Finally, we offer our conclusions and directions for future work in Section V.

II. Theoretical Background

In this section, we present a brief recap about DNNs pruning, regarding its different types and methods.

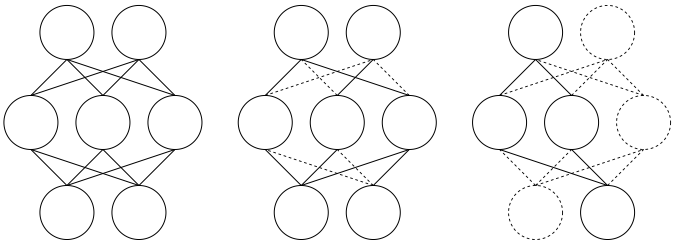
A. Unstructured pruning

Unstructured pruning, also known as sparsity-inducing or weight pruning, consists of removing connections between neurons or filters of adjacent layers. In doing so, zero values are introduced into the weight tensors, which has the effect of removing them. Despite the effectiveness of unstructured pruning methods [18], their application is still limited, since general purpose hardware (e.g. CPUs and GPUs) and libraries (e.g., BLAS) do not deal well with sparse matrices, penalizing the execution of pruned networks in this way [11].

B. Structured pruning

Structured pruning, as the name suggests, comprises the removal of entire structures from the neural network [13] [24]. These structures can be neurons of fully connected networks or entire filters of convolutional neural networks. The major benefit of using structured pruning is that by not adding unstructured sparsity to the network, the pruned architecture

allows effective network acceleration in nearly any hardware and software libraries. Figure 1 shows how both unstructured and structured pruning affects the neural network architecture.



(a) Unpruned network (b) Unstructured pruned (c) Structured pruned

Fig. 1. Effect of different pruning types on the neural network architecture. Dashed units and connections means that they were pruned.

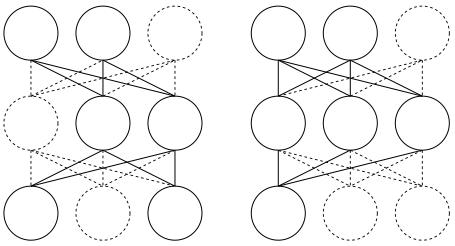
C. Pruning locality

Some pruning techniques, as in the works of Li *et al.* (2017) [13], and He *et al.* (2019) [24], require a user-defined pruning rate to be applied to the model. This pruning rate indirectly represents the number of weights, neurons, or filters to be removed during the pruning procedure. Regardless of the pruning rate applied, it is necessary to define from where, in the DNN, the elements will be removed, either locally or globally. Figure 2 illustrates how both local and global pruning affects the DNN architecture.

1) Local pruning: In the local pruning, the elements are removed taking into account their location. Regardless of the criterion adopted, the weights, neurons, or even entire convolutional filters are ranked and pruned proportionally in all layers of the DNN. Han *et al.* (2015) [11] and Frankle and Carbin (2019) [10] are examples of works that apply local pruning on some scenarios.

2) Global pruning: In the global pruning, as applied by Frankle *et al.* (2019) [18], Salama *et al.* (2019) [25], Renda *et al.* (2020) [20], the elements are removed without taking into account their location. Thus, to achieve a certain pruning rate, variable pruning rate fractions are applied in different layers based on the global ranking of the neural network elements. The final reduction of weights, neurons, or filters are equivalent to the initial pruning rate applied.

It is worth noticing that any continuous compression rate, used for pruning, should be mapped into a discrete number representing the number of elements to be effectively removed. Thus, the number of elements to be removed will not always match the compression rate chosen, but, instead, the nearest possible one. Besides, global pruning can break a model depending on the level of compression applied. This issue could happen, for example, when all weights, neurons, or filters are completely removed from a given layer. Furthermore, it is not allowed to remove elements from the output layer since it is responsible for the outcomes of the model.



(a) Locally pruned network (b) Globally pruned network

Fig. 2. Effect of different pruning localities on the neural network architecture. Dashed objects means that they were pruned.

D. Pruning Scheduling

In addition to the local pruning, the pruning schedule must be defined. The scheduling can be broadly distributed into three categories [15], [20], as follows:

- **One-Shot pruning.** Pruning is applied to the trained DNN all at once so that the defined pruning rate is achieved at the end of the process.
- **Iterative pruning.** In this approach, small fractions of the DNN are pruned at the end of an entire training cycle. Thus, the pruning rate is achieved at the end of several training and pruning iterations.
- **Gradual pruning.** In this approach, the DNN is pruned throughout the training process, so that at the end of the training a pruned neural network is produced.

Due to the integration between training and pruning in gradual pruning, there is no need to do an additional step to retrain the pruned network, since pruning is done in conjunction with the neural network training. However, in one-shot pruning and iterative pruning, whose elements are removed at the end of the training process, the pruned networks often need to be fine-tuned or retrained to compensate for the information loss.

Furthermore, in this paper we only evaluate pruned networks obtained by iterative pruning, since previous studies have shown that iterative pruning is able to generate pruned networks with a higher level of compression and lower accuracy reduction [10], [11], [20], [21] when compared to those obtained by one-shot pruning or gradual pruning.

III. Related Work

This paper is mainly related to the neural network pruning literature, more specifically structured pruning techniques and the Lottery Ticket Hypothesis. The "Optimal Brain Damage" [26] and "Optimal Brain Surgeon" [27] are among the pioneering works on pruning with the idea of using a sensitivity measure to estimate the impact of removing units from the networks. However, the use of sensitivity estimates involves the additional calculation of second order derivatives, making these methods computationally costly when applied to over-parameterized networks.

As an alternative to sensitivity-based methods, recent studies have advocated towards magnitude-based approaches [11], [13], [19], [24], which consider the magnitude of the weights

of a trained network as a proxy measure for unit importance. Since calculating the magnitude is computationally simpler and more efficient than calculating second order derivatives, such methods can scale up and be applied to deeper networks with hundreds of millions of parameters.

Among recent influential research on neural network pruning, Frankle and Carbin (2019) [10] formulated the Lottery Ticket Hypothesis, which states that it is possible to find sparse subnetworks (winning tickets) from an over-parameterized neural network and to train them from scratch so that the sparse sub networks can achieve similar or even greater accuracy than their unpruned counterpart. The steps for finding winning tickets are the following:

(1) Randomly initialize a neural network.
(2) Train it until convergence.
(3) Prune a fraction of the network using the magnitude of the weights as a criterion.
(4) Reset the remaining weights of the pruned network to its initial random values.
(5) Train the pruned network until convergence.

By iteratively applying these steps, the authors were able to achieve aggressively pruned networks that reach or even surpass their unpruned counterparts' accuracy, while converging upon only a fraction of the original model required number of epochs. Although the impressive results on fully connected networks and simple convolutional networks, this approach was not able to find winning tickets on deep convolutional neural networks. Aiming to overcome such limitation, the authors proposed an updated version of the Lottery Ticket Hypothesis and the steps for finding Winning Tickets in a follow-up work [18]. Instead of resetting the weights to its initial random values, the weights are now reset to the values they had at an early epoch of the unpruned network training. With this retraining approach, named as "weight rewinding", it was possible to find winning tickets on deep convolutional neural networks.

Because of the effectiveness of the Lottery Ticket Hypothesis, further work focused on theoretical aspects of the winning tickets. Zhou *et al.* (2019) [19] empirically evaluated the pruning masks that emerge with the winning tickets and discovered the existence of supermasks, which are pruning masks [1] that can be applied to untrained networks, performing better than chance. Morcos *et al.* (2019) [21] focused at evaluating how well a winning ticket performs when re-used on a different dataset.

More recently, Renda *et al.* (2020) [20] introduced the Learning Rate Rewinding method, a novel retraining approach, as a variation of weight rewinding. Unlike the weight rewinding, the authors propose to keep the final training weights (similar to fine-tuning) and reset only the learning rate when retraining the pruned network. The results indicate that the learning rate rewinding produces winning tickets that matches or outperforms those produced by weight rewinding.

In light of the aforementioned related works, we observed that most claims and conclusions are based on experiments with unstructured pruning, with the exception of Renda *et al.* (2020) [20] who present preliminary experiments with structured pruning. These experiments are however limited to using per-layer pruning rates hand-picked by Li et al. (2017) [13] and a set of pruning rates derived from these hand-picked rates.

To the best of our knowledge, no prior work evaluated the emergence of "winning tickets" when applying the concepts of weight rewinding and learning rate rewinding in the context of structured pruning where the target architecture is automatically determined throughout the pruning process. In the next section, we present the experimental methodology adopted in this paper.

IV. Evaluation

In this section, we evaluate the emergence of winning tickets on deep convolutional networks and compare the efficacy of two retraining approaches: weight rewinding and learning rate rewinding, for finding winning tickets. Randomly initialized structured pruned networks are used as baselines in the comparison. In the following subsections, we describe the experimental setup and discuss the obtained results.

A. Experimental Setup

1) Hardware and Libraries: All the experiments were conducted on Ubuntu 16.04.6 LTS Linux Kernel 4.15.0-107-generic equipped with Intel Core i7-8700K CPU @ 3.70GHz processor, 2x16GiB DIMM Synchronous 2666 MHz memory and GeForce 2080 RTX Ti graphic processing unit. The neural networks and pruning algorithms were implemented using PyTorch [28] v1.3.1 and datasets from the torchvision package v0.4.2.

2) DNN Architecture and Datasets: For the experiments, we selected the well known convolutional neural network model VGG16 [6], which has ReLU activations [29] and Batch Normalization layers [30]. The model is trained by 160 epochs on the CIFAR-10 benchmark dataset [31], since this network and dataset are used for evaluating pruning methods in prior works [13], [15], [32], [33]. The CIFAR-10 dataset contains 60,000 32×32 color images organized in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. In addition, during the training we apply standard data augmentation as proposed by He *et al.* (2016) [34]. SGD with momentum is used as the optimization method, with an initial learning rate of 0.1, which decay (dividing by 10) at epochs 80 and 120 as in Li *et al.* (2019) [15].

3) Pruning Approaches: We assess the emergence of "winning tickets" by pruning the VGG16 network using iterative pruning in two scenarios: local pruning and global pruning. For both scenarios, we use the L1-norm of the convolutional filters as pruning heuristic, removing at each pruning iteration those

[1] A pruning mask is a binary matrix that shows whether a given weight or set of weights should be pruned (0) or not (1). The pruned network ($f(x; w \odot m)$) results from the Hadamard product ($\odot$) of the weights matrix (w) and the pruning mask (m).

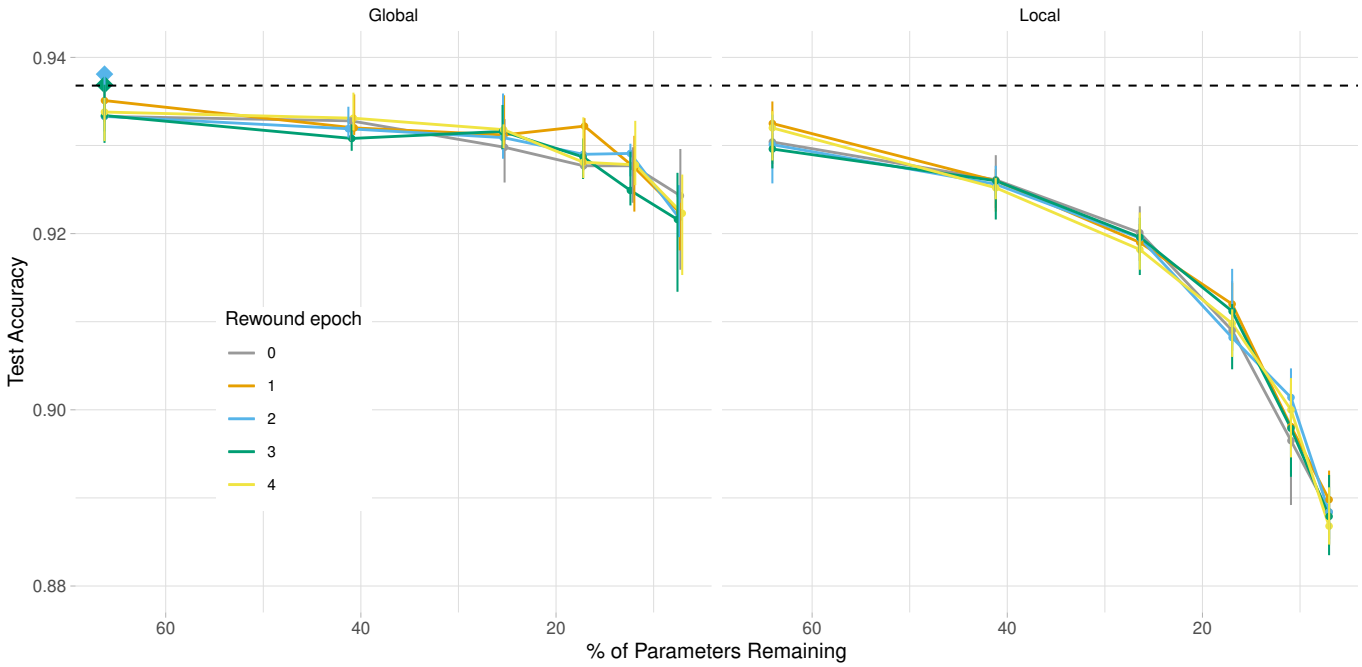


Fig. 3. Accuracy of VGG16 network trained on CIFAR-10 while being iteratively pruned using weight rewinding. Winning tickets are marked with diamond shape and the accuracy of the unpruned network is also shown (dashed line).

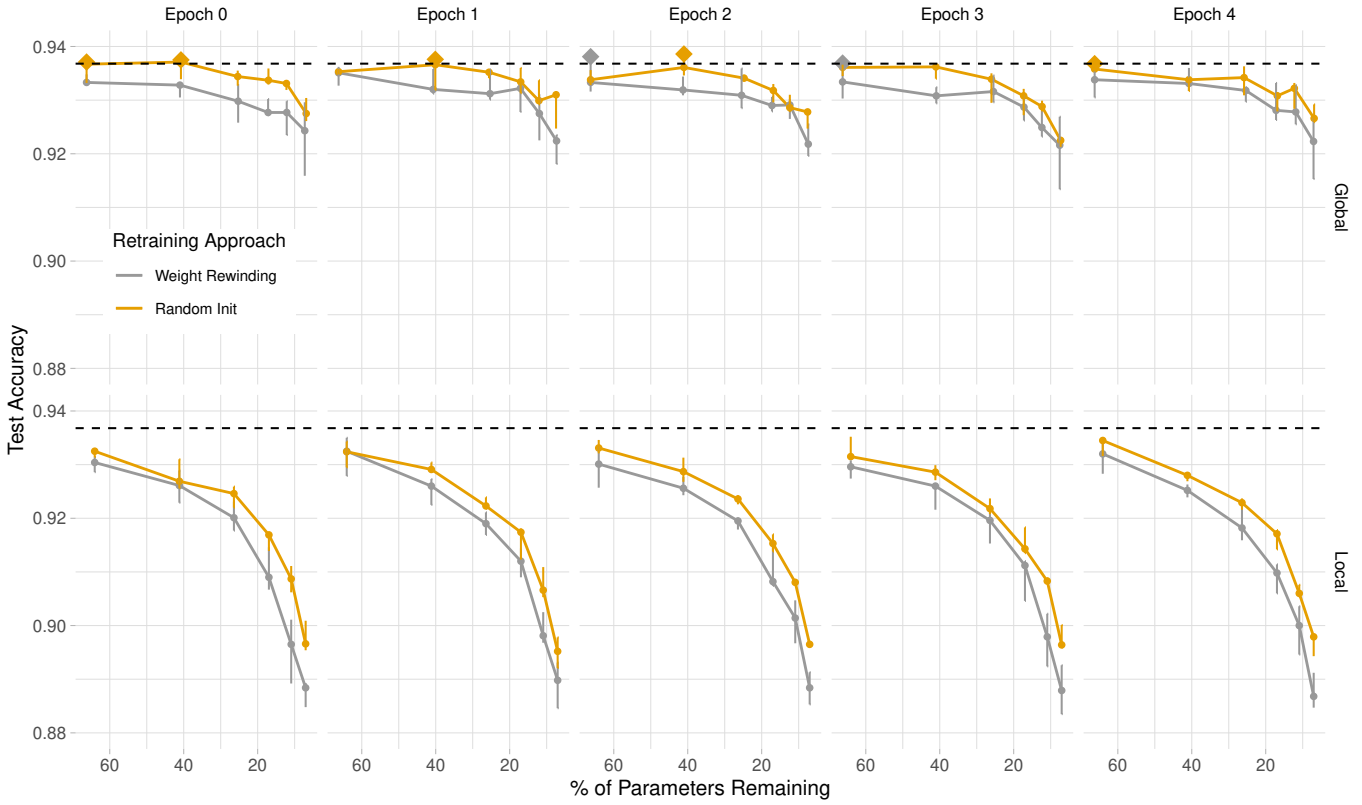


Fig. 4. Accuracy comparison between weight rewinding and random initialization for VGG16 network trained on CIFAR-10 while being iteratively pruned. Winning tickets are marked with diamond shape and the accuracy of the unpruned network is also shown (dashed line).

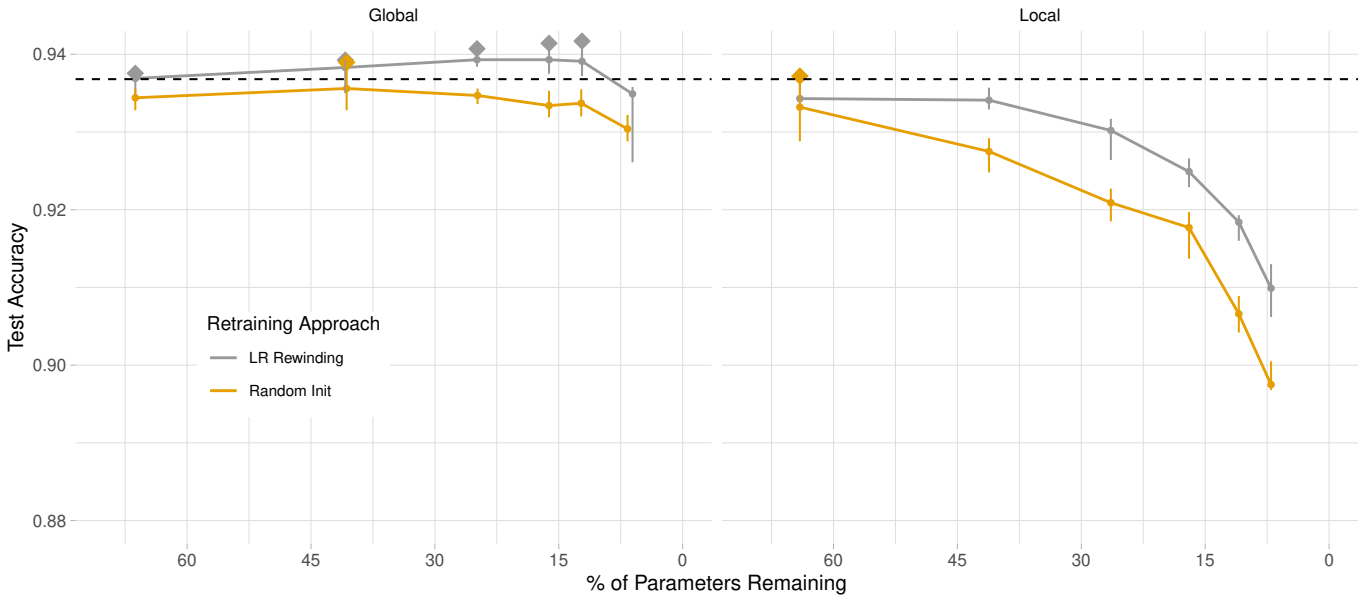


Fig. 5. Accuracy of VGG16 network while being iteratively pruned using learning rate rewinding. Winning tickets are marked with diamond shape and the accuracy of the unpruned network is also shown (dashed line).

filters with the lowest norms according to a chosen pruning rate. We performed 10 pruning iterations and, at each iteration, we applied 20% pruning rate, as in prior work [10], [20], [21]. Since the accuracies in the final four pruning iterations were lower than that of the unpruned network and therefore no winning tickets were found, those iterations were omitted in the results presentation.

B. Results and Discussion

The results obtained using each retraining techniques are presented in this section. The plots in Figures 3, 4 and 5 show the median, maximum and minimum test accuracies from five different training runs. For comparison, we also consider retraining the pruned networks obtained by both weight rewinding and learning rate rewinding with random initial weights, as discussed next.

1) Weight Rewinding: For evaluating the emergence of winning tickets by weight rewinding on structured pruned networks, we consider the retraining technique as presented in Frankle and Carbin (2019) [10] and Frankle *et al.* (2019) [18] and we consider the epochs 0 to 4 as weight rewinding epochs.

As we can see in Figure 3, rewinding to the initial random weights (epoch 0) or the weights from early training epochs (1 to 4) does not have a significant impact on the final accuracy of the pruned model by structured pruning, either for local or global pruning. On average, the sub-networks obtained by weight rewinding presented lower accuracy than the unpruned network, even in the initial pruning iterations. Although few, winning tickets emerged when global pruning was applied and the weights were rewound to the weights of epoch 2 and epoch 3 of the unpruned network training. However, these winning tickets emerged in the first iteration of pruning, when only 20% of the convolutional filters were removed, so that it cannot be considered an aggressive pruning level.

We can observe in Figure 3 that, although global pruning generates pruned networks that generally present greater accuracy, it is less stable than local pruning and presents a greater variation in accuracy in more aggressive pruning levels (i.e. lower percentage of remaining parameters). This is due to the fact that global pruning in combination with structured pruning can cause the removal of many specific layer filters, which makes it impossible for the pruned network to learn the classification rule from and insufficient number of image features.

In Figure 4, we compare the pruned networks generated by weight rewinding with their counterparts retrained with random initialized weights. We can observe that for almost all the subnetworks generated, those trained with random weights match the accuracy or outperforms the subnetworks obtained by weight rewinding. This can be observed both when the weights are rewound to the random initial weights (Epoch 0), and when the weights are rewound to the weights from early training epochs (Epochs 1 to 4), in either global or local pruning. In addition, it is still possible to notice a greater occurrence of winning tickets (diamond shape points in the plots) in the subnetworks trained with random weights.

These results suggest that, despite the unprecedented results achieved for unstructured pruned networks [10], [18], weight rewinding is not efficient for finding winning tickets in the context of structured pruning and is not better than retraining the pruned networks with random weights.

2) Learning Rate Rewinding: In Figure 5, we present the results from applying the learning rate (LR) rewinding retraining technique as proposed by Renda *et al.* (2020) [20].

Unlike weight rewinding, by applying learning rate rewinding we can notice a higher occurrence of winning tickets. These winning tickets emerged even in later iterations of pruning, i.e. for more aggressive pruning levels.

While with weight rewinding it was only possible to identify subnetworks that nearly matched the accuracy of the original unpruned network, the winning tickets generated by learning rate rewinding presents greater accuracy than the unpruned network. This phenomenon occurs even in subnetworks that have less than 85% of parameters remaining, when compared with the unpruned network. However, these winning tickets only emerged through global pruning. We can also observe that, as occurred with weight rewinding, globally pruned networks tend to be less stable, presenting a greater variation in final accuracy between different runs for the most aggressive pruning levels.

Furthermore, it is noteworthy that the subnetworks obtained through learning rate rewinding have higher accuracy when compared to their retrained counterparts with random weights. This is an indication that the learning rate rewinding retraining technique has a significant impact on the final accuracy of the structured pruned networks.

In addition, despite we considered an architecture (VGG-16 [6]) different from those investigated by Renda *et al.* (2020) [20] (ResNet-56 and ResNet-34 [35]), they were all convolutional neural networks. Thus, without loss of generality, we were able to find winning tickets associated to compression levels that were more aggressive than those presented by the authors in structured pruned networks.

For comparison, Renda *et al.* (2020) [20] apply structured pruning with learning rate Rewinding for a small range of compression levels, from approximately 86.96% to 58.82% of the remaining parameters on ResNet-56 and 92.60% to 79.36% of the remaining parameters on ResNet-34. Winning tickets were only identified at the lower compression levels ($\approx 86.96\%$ and $\approx 77.52\%$ of the remaining parameters on ResNet-56 and $\approx 92.60\%$ and $\approx 86.96\%$ on ResNet-34)[2]. Differently from them, we considered a larger compression interval, from approximately 66.27% to 1.02% of the remaining parameters on VGG-16 and our winning tickets were identified at higher compression levels ($\approx 66.27\%$, $\approx 40.85\%$, $\approx 24.90\%$, $\approx 16.18\%$ and $\approx 12.19\%$ of the remaining parameters).

In conclusion, our experimental evaluation covered a wider range of compression levels and was able to identify more compressed winning tickets (i.e. more suitable for computing on hardware with limited resources).

V. Conclusion and Future Work

In this work we have evaluated the emergence of winning tickets by structured pruning on the VGG16 network trained on the CIFAR-10 dataset by comparing iteratively pruned networks found by weight rewinding, learning rate rewinding and their randomly initialized counterparts and considering local and global pruning. Our findings suggest that winning tickets can emerge by applying structured pruning with architectures discovered automatically during the global pruning process (i.e. the number of convolutional filters in each layer is not predetermined and these filters are removed regardless of the layer, according to the pruning rate and heuristics chosen), while prior work were limited to unstructured pruning and structured pruning with hand-chosen architectures (i.e. the number of convolutional filters in each layer of the resulting architecture is predefined).

Overall results indicate that, for both rewinding techniques, the local pruning fails for finding winning tickets. Moreover, few winning tickets emerged from weight rewinding, only for low pruning levels. When compared to their counterparts with randomly initialized weights they performed nearly the same or worse. However, the learning rate rewinding was able to find winning tickets even at aggressive pruning levels. The networks obtained through learning rate rewinding outperform their randomly initialized counterparts.

Future work might focus on identifying the characteristics that make these two retraining approaches to behave differently in structured and unstructured pruned networks and on evaluating weight rewinding and learning rate rewinding in the context of structured pruning for more datasets (e.g. CIFAR-100 [31] and ImageNet [36]) and neural network architectures (e.g. ResNets [35] and DenseNet [37]) and for different tasks (e.g data augmentation using Generative Adversarial Networks [38], in addition to classification) aiming to validate the generality of these rewinding techniques.

[2]The percentage of remaining parameters was calculated from the reported compression ratio values of Renda *et al.* (2020) [20].

Acknowledgement

This work was supported by a cooperation between UFCG and Hewlett Packard Enterprise (Hewlett-Packard Brasil Ltda.) using incentives of Brazilian Informatics Law (Law No. 8.2.48 of 1991).

References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[2] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2654–2662.

[3] Y. Li and Y. Liang, "Learning overparameterized neural networks via stochastic gradient descent on structured data," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 8157–8166.

[4] S. S. Du and J. D. Lee, "On the power of over-parametrization in neural networks with quadratic activation," *arXiv preprint arXiv:1803.01206*, 2018.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[8] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Advances in neural information processing systems*, 2014, pp. 855–863.

[9] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artificial Intelligence Review*, 2020.

[10] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*, 2019.

[11] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 1135–1143.

[12] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," in *International Conference on Learning Representations*, 2018.

[13] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *International Conference on Learning Representations*, 2017.

[14] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.

[15] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *International Conference on Learning Representations*, 2019.

[16] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 242–252.

[17] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9194–9203.

[18] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Stabilizing the lottery ticket hypothesis," *arXiv preprint arXiv:1903.01611*, 2019.

[19] H. Zhou, J. Lan, R. Liu, and J. Yosinski, "Deconstructing lottery tickets: Zeros, signs, and the supermask," in *Advances in Neural Information Processing Systems*, 2019, pp. 3597–3607.

[20] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," in *International Conference on Learning Representations*, 2020.

[21] A. Morcos, H. Yu, M. Paganini, and Y. Tian, "One ticket to win them all:

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-generalizing lottery ticket initializations across datasets and optimizers," in *Advances in Neural Information Processing Systems*, 2019, pp. 4932–4942.

[22] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, pp. 1–18, 2017.

[23] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 2074–2082.

[24] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[25] A. Salama, O. Ostapenko, T. Klein, and M. Nabi, "Pruning at a glance: Global neural pruning for model compression," 2019.

[26] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[27] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in neural information processing systems*, 1993, pp. 164–171.
performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[31] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009.

[32] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[33] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[35] ——, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[37] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[38] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.