LOFT: Finding Lottery Tickets through Filter-wise Training

Qihan Wang[⋆], **Chen Dun**[⋆], **Fangshuo Liao**[⋆], **Chris Jermaine, Anastasios Kyrllidis**
Department of Computer Science
Rice University

Abstract

In this paper, we explore how one can efficiently identify the emergence of "winning tickets" using distributed training techniques, and use this observation to design efficient pretraining algorithms. Our focus in this work is on convolutional neural networks (CNNs), which are more complex than simple multi-layer perceptrons, but simple enough to exposure our ideas. To identify good filters within winning tickets, we propose a novel filter distance metric that well-represents the model convergence, without the need to know the true winning ticket or fully training the model. Our filter analysis behaves consistently with recent findings of neural network learning dynamics. Motivated by such analysis, we present the *LOttery ticket through Filter-wise Training* algorithm, dubbed as LOFT. LOFT is a model-parallel pretraining algorithm that partitions convolutional layers in CNNs by filters to train them independently on different distributed workers, leading to reduced memory and communication costs during pretraining. Experiments show that LOFT i) preserves and finds good lottery tickets, while ii) it achieves non-trivial savings in computation and communication, and maintains comparable or even better accuracy than other pretraining methods.

1 Introduction

The Lottery Ticket Hypothesis (LTH) [8] claims that neural networks (NNs) contain subnetworks ("*winning tickets*") that can match the dense network's performance when fine-tuned in isolation. Yet, identifying such subnetworks often requires proper pretraining of the dense network. How to efficiently find such subnetworks remains a widely open question: since LTH relies on a *pretraining* phase, it is a *de facto* criticism that finding such pretrained models could be a burdensome task, especially when one focuses on large NNs.

This burden has been eased with efficient training methodologies, which are often intertwined with pruning steps. Simply put, one has to answer two fundamental questions: *"When to prune?"* and *"How to pretrain such large models?"*. Focusing on *"When to prune?"*, one can prune before [23, 22, 35], after [21, 13, 7, 12, 24, 29, 11, 36, 41], and/or during pretraining [8, 33, 27, 2, 6, 30, 28].[1] Works like SNIP [23, 22] and GraSP [35] aim to prune without pretraining, while suffering some accuracy loss. Pruning after training often leads to favorable accuracy, with the expense of fully training a large model. A compromise between the two approaches exist in *early bird tickets* [39], where one could potentially avoid the full pretraining cost, but still identify "winning tickets", by performing a smaller number of training epochs and lowering the precision of computations. *This finding reveals the opportunity to design more efficient pretraining algorithms that target specifically at identifying the winning tickets for larget models.*

[1] LTH approaches, while they originally imply pruning after training, include pruning at various stages during pretraining to find the sparse subnetworks.

Has it Trained Yet? Workshop at the Conference on Neural Information Processing Systems (NeurIPS 2022).

Focusing on *"How to pretrain large models?"*, modern large-scale neural networks come with significant computational and memory costs. Researchers often turn to distributed training methods, such as data parallel and model parallel [43, 1, 34, 3, 42, 9, 10, 4], to enable heavy pretraining towards finding winning tickets, by using clusters of compute nodes. Yet, data parallelism needs to update the whole model on each worker—which still results in a large memory and computational cost. To handle such cases, researchers utilize model parallelism, such as Gpipe [17], to reduce the per node computational burden. Traditional model parallelism enjoys similar convergence behaviour as centralized training, but needs to synchronize at every training iteration to exchange intermediate activations and gradient information between workers, thus often incurring high communication cost.

Our approach and contributions. In this work, we propose a new model-/data-parallel pretraining method on the one-shot pruning setting that can efficiently reveal winning tickets for convolutional neural networks (CNNs). In particular, we center on the following questions:

"What is a characteristic of a good pretrained CNN that contains the winning ticket? How will such a criterion inform our design towards efficient pretraining?"

Prior works show that filter-wise pruning is more preferable compared with weight pruning for CNNs [17, 16, 5, 37, 25]. Our approach operates by decomposing the full network into narrow subnetworks via filter-wise partition during pretraining. These subnetworks, which are randomly recreated intermittently during the pretraining process, are trained independently and their updates are periodically aggregated into the global model. Because each subnetwork is much smaller than the full model, our approach also enables scaling beyond the memory limit of a single GPU. Our methodology allows the discovery of winning tickets with less memory and a lower communication budget. The contributions of our work are summarized as follows:

- We propose a metric to quantify the distance between tickets in different stages of pretraining, allowing us to characterize the convergence to winning tickets throughout the pretraining process.
- We identify that such convergence behavior suggests an alternative way of pretraining: we propose a novel data-/model-parallel pretraining method through filter-wise partition of CNNs and iterative training of such subnetworks.
- We theoretically show that our proposed method achieves CNN weight that is close to the weight found by gradient descent in a simplified scenario.
- We empirically show that our method provides a better or comparable winning ticket, while being memory and communication efficient.

2 Identifying Winning Tickets Early in the Training Process

The CNN model [14, 19, 31] is composed of convolutional layers, batch norm layers [18], pooling layers, and a final linear classifier layer. Our goal is to retrieve a *structured* winning ticket, through partitioning and pruning the filters in the convolutional layers. Let p_i denote the number of input channels for the i-th convolutional layer. Correspondingly, the output channel of the i-th layer is the same as the input channel of the $(i+1)$-th layer, which is p_{i+1}. Let h_i,w_i be the height and width of the input feature maps, respectively. Then, the i-th convolutional layer transforms the input feature map $x_i \in \mathbb{R}^{p_i \times h_i \times w_i}$ into the output feature map $x_{i+1} \in \mathbb{R}^{p_{i+1} \times h_{i+1} \times w_{i+1}}$ by performing 2D convolutions on the input feature map with p_{i+1} filters of size 3×3, where the j-th filter is denoted as $\mathcal{F}_{i,j} \in \mathbb{R}^{p_i \times 3 \times 3}$. Thus the total filter weight for the i-th layer is $\mathcal{F}_i \in \mathbb{R}^{p_{i+1} \times p_i \times 3 \times 3}$. Formally, prunning $1/k$ of the filters in the i-th layer is equivalent to discarding p_{i+1}/k filters. Thus the resulted total pruned filter weight is in $\mathbb{R}^{p_{i+1} \cdot (k-1)/k \times p_i \times 3 \times 3}$ and the output feature map x_{i+1} is in $\mathbb{R}^{p_{i+1} \cdot (k-1)/k \times h_{i+1} \times w_{i+1}}$.

2.1 Evaluate the distance of two pretrained models

Borrowing techniques from search system rankings [20], we propose a *filter distance* metric, based on a position-weighted version of Spearman's footrule [32]. In particular, consider the filters at epochs X and Y on the i-th layer, namely $\mathcal{F}_i^{(X)}, \mathcal{F}_i^{(Y)}$. We calculate the ℓ_2-norm of $\mathcal{F}_{i,j}^{(X)}, \mathcal{F}_{i,j}^{(Y)}$ for each filter index $j \in [p_{i+1}]$ and sort them by magnitude. We denote the two sorted list with length p_{i+1} as $R^{(X)}$ and $R^{(Y)}$, with each containing the ℓ_2-norm of the filters, $\left\|\mathcal{F}_{i,j}^{(X)}\right\|_2$ and $\left\|\mathcal{F}_{i,j}^{(Y)}\right\|_2$.

We represent the change in ranking from $R^{(X)}$ to $R^{(Y)}$ as σ. Namely, if $x \in R^{(X)}$ is the i-th element in $R^{(X)}$, then, the ranking of x in $R^{(Y)}$ is denoted as $\sigma(i)$. The original Spearman's footrule defines the displacement of element i as $|i - \sigma(i)|$, leading to the total displacement of all elements as:

$$F(\sigma) = \sum_i |i - \sigma(i)|.$$

Given weights w_i's for the elements, the weighted displacement for element i becomes $w_i \cdot \left|\sum_{j<i} w_j - \sum_{\sigma(j)<\sigma(i)} w_j\right|$, leading to the total weighted displacement as follows: [2]

$$F_w(\sigma) = \sum_i w_i \cdot (|\sum_{j<i} w_j - \sum_{\sigma(j)<\sigma(i)} w_j|).$$

To put emphasis on the correct ranking of the top elements, we set the position weight for the i-th ranking element as $1/i$. To further simplify calculations, we approximate $\sum_{i=1}^{n} \frac{1}{i} \approx \ln(n) - \ln(1)$ where $\ln(\cdot)$ is the natural logarithm. The above lead to the following definition for our *filter distance*:

$$F_{\text{filter}}(\sigma) = \sum_i \tfrac{1}{i} \cdot |\ln(i) - \ln(\sigma(i))|\,.$$

For the case where the two lists of pruned filters do not contain the same elements, we can naturally define the distance when the i-th element is not in the other list to be $|\ln(l+1) - \ln(i)|$; l is the length of the pruned filter list. This filter distance metric is fundamentally different from the mask distance proposed in [38]. We compare against these early-pruning methods in the experiments.

2.2 Rethinking the Property of Winning Tickets

Through empirical analysis, we observe that *training the CNN weights until loss converges is not necessary for the discovery of winning tickets.* However, many existing pretraining algorithms do not exclude heavy training over the whole CNN model. *These facts demand a new pretraining algorithm, targeting specifically at efficiently finding winning tickets.*

We propose sampling and training different sets of tickets during different stages of the pretraining. In this way, the algorithm is expected to "touch" upon the potential winning tickets at certain iterations. We conjecture (this is empirically shown in our experiments) that important filters in such winning tickets can be preserved and further recovered at the end of pretraining using our approach. These observations led us to the definition of the LOFT algorithm.

3 LOFT: Finding Lottery Tickets Through Filter-wise Training

Algorithm 1 LOFT Algorithm

Parameter: T synchronization iterations in pretraining, S workers, ℓ local iterations, W CNN weights,

$h(W) \leftarrow$ randomly initialized CNN.
for $t = 0, \ldots, T-1$ **do**
  $\{h_s(W_s)\}_{s=1}^{S} = \texttt{filterPartition}(h(W),\ S)$
  Distribute each $h_s(W_s)$ to a different worker.
  for $s = 1, \ldots, S$ **do**
    Train $h_s(W_s)$ for ℓ iterations using local SGD.
  end for
  $h(W) = \texttt{aggregate}\left(\{h_s(W_s)\}_{s=1}^{S}\right)$.
end for

We treat "*sampling and training sets of tickets*" as a filter-wise decomposition of a given CNN, where each ticket is a subnetwork with a subset of filters. The LOFT algorithm that implements our ideas is shown in Algorithm 1. Each block within a CNN typically consists of two identical convolutional layers, conv_i and conv_{i+1}. Our methodology operates by partitioning the filters of these layers, $\mathcal{F}_i$ and $\mathcal{F}_{i+1}$, to different subnetworks –see `filterPartition()` step in Algorithm 1– in a structured, disjoint manner. These subnetworks are trained independently –see local SGD steps in Algorithm 1– before aggregating their updates into the global model by directly placing the filters back to their original place—see `aggreegate()` step in Algorithm 1. *The full CNN is never trained directly.*

Our methodology of choosing tickets/subnetworks avoids partitioning layers that are known to be most sensitive to pruning, such as strided convolutional blocks [26]. Parameters that are not

[2] Using other norm calculation like ℓ_2-norm will not affect the overall characteristic of filter distance in analysis.

partitioned are shared among subnetworks, so their values must be averaged when the updates of tickets/subnetworks are aggregated into the global model.

Compared with common distributed protocols, our pretraining methodology $i)$ reduces the communication costs, since we only communicate the tickets/subnetworks; and $ii)$ reduces the computational and memory costs on each worker, since we only locally train the sampled tickets/subnetworks that are smaller than the global model. From a different pespective, our approach allows pretraining networks beyond the capacity of a single-GPU: The global model could be a factor of $O(S)$ wider than each subnetwork, allowing the global model size to be extended far beyond the capacity of single GPU. *The ability to train such "ultra-wide" models is quite promising for pruning purposes.*

After pretraining with LOFT, we perform standard pruning on the whole network to recover the winning ticket, and use standard training techniques over this winning ticket until the end of training.

Theoretical result. We perform theoretical analysis on a one-hidden-layer CNN, and show that *the trajectory of the neural network weight in* LOFT *stays near to the trajectory of gradient descent*.

Theorem 1. *Let $f(\cdot,\cdot)$ be a one-hidden-layer CNN with the second layer weight fixed. Assume the number of hidden neurons satisfies $m = \Omega\left((n^4K^2/\lambda_0^4\delta^2)\cdot\max\{n,d\}\right)$ and the step size satisfies $\eta = O\left(\frac{\lambda_0}{n^2}\right)$: Then, under mild assumptions, with probability at least $1-O(\delta)$ we have:*

$$\mathbb{E}_{[\mathbf{M}_T]}\left[\left\|\mathbf{W}_T-\hat{\mathbf{W}}_T\right\|_F^2\right]+\eta\sum_{t=0}^{T-1}\mathbb{E}_{[\mathbf{M}_T]}\left[\left\|f\left(\mathbf{X},\mathbf{W}_t\right)-f\left(\mathbf{X},\hat{\mathbf{W}}_t\right)\right\|_2^2\right]\leq O\left(\frac{n^2\sqrt{d}}{\lambda_0^2\kappa m^{\frac{1}{4}}\sqrt{\delta}}+\frac{2\eta^2T\theta^2(1-\xi)\lambda_0}{S}\right).$$

Remarks. Intuitively, this theorem states that the sum of the expected weight difference in the T-th iteration (i.e., $\mathbb{E}_{[\mathbf{M}_T]}[\|\mathbf{W}_T-\hat{\mathbf{W}}_T\|_F^2]$) and the aggregation of the step-wise difference of the neural network output between LOFT and gradient descent (i.e., $\sum_{t=0}^{T-1}\mathbb{E}_{[\mathbf{M}_T]}\|f\left(\mathbf{X},\mathbf{W}_t\right)-f\left(\mathbf{X},\hat{\mathbf{W}}_t\right)\|_2^2$) is bounded and controlled by the quantity on the right-hand side. I.e., both the weights found by LOFT as well as the output of LOFT are close to the ones found by regular training. Notice that increasing the number of filters m and the number of subnetworks S will drive the bound of the summation to zero.

SETTING	NO-PRUNE	METHODS	PRUNING RATIO 80%	50%	30%	COMM COST	IMPROV.
RESNET34 CIFAR-100	75.93	GPIPE-2	75.51	76.00		131.88G	
		LOFT-2	76.11	77.07		104.78G	1.26×
		GPIPE-4	75.51	76.00		461.60G	
		LOFT-4	75.05	76.51		144.66G	3.19×
PRERESNET-18 IMAGENET	70.71	GPIPE-2	66.71	69.14	70.29	20954.24G	
		LOFT-2	65.41	69.12	69.64	791.09G	21.60×
		GPIPE-4	66.71	69.14	70.29	52385.59G	
		LOFT-4	65.60	68.93	69.77	1284.84G	40.77×

Table 1: Left: Fine-tuned accuracy for different pretraining methods at different pruning ratios. NO-PRUNE corresponds to full CNN training without pruning. Right: Total communication costs (COMM) of model parallel baseline (GPipe) [17] and LOFT during pretraining. Number after method name represents the number of parallel worker used.

4 Experiments

We show that LOFT can preserve the winning tickets and non-trivially reduce costs during pretraining. We illustrate that LOFT does not recover the winning tickets by chance: LOFT converges to winning tickets faster and provide better tickets for all pretraining length.

Experimental Setup. For each setting, we consider a workflow where we pre-train for 20 epochs and then fine-tune for 90 epochs. We consider 3 networks: PreActResNet-18, PreActResNet-34 [15], and WideResNet-34 [40] to characterize our performance on models of different sizes and structures. We test these settings on the CIFAR-100, and ImageNet.

Table 1 shows the performance comparison for LOFT and Gpipe under various settings. While LOFT inherits the memory efficiency from model-parallel training methods, it further reduces the communication cost from $1.26\times$ up to $40.77\times$, as shown in Table 1. This is achieved by $i)$ changing

the way of decomposing the network such that each worker can host an independent subnetwork and train locally without communication, which greatly reduces the communication frequency; and ii) each worker only exchange the weight of the subnetwork after each round of local training instead of transmitting activation maps and gradients.

References

[1] A. Agarwal and J. Duchi. Distributed delayed stochastic optimization. In *Advances in NeurIPS*, pages 873–881, 2011.

[2] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. In *ICLR*, 2018.

[3] T. Ben-Nun and T. Hoefler. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. *arXiv e-prints*, page arXiv:1802.09941, 2018.

[4] C.-C. Chen, C.-L. Yang, and H.-Y. Cheng. Efficient and Robust Parallel DNN Training through Model Parallelism on Multi-GPU Platform. *arXiv e-prints*, page arXiv:1809.02839, 2018.

[5] Arthur da Cunha, Emanuele Natale, and Laurent Viennot. Proving the Strong Lottery Ticket Hypothesis for Convolutional Neural Networks. In *ICLR 2022 - 10th International Conference on Learning Representations*, Virtual, France, April 2022.

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

[7] Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4860–4874, 2017.

[8] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2018.

[9] A. Gholami, A. Azad, P. Jin, K. Keutzer, and A. Buluc. Integrated Model, Batch and Domain Parallelism in Training Neural Networks. *arXiv e-prints*, page arXiv:1712.04432, 2017.

[10] L. Guan, W. Yin, D. Li, and X. Lu. XPipe: Efficient Pipeline Model Parallelism for Multi-GPU DNN Training. *arXiv e-prints*, page arXiv:1911.04610, 2019.

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

[12] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in NeurIPS*, 28, 2015.

[13] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645. Springer, 2016.

[16] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2006–2015, 2020.

[17] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. Chen, D. Chen, H. Lee, J. Ngiam, Q. Le, Y. Wu, and Z. Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[19] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in NeurIPS*, volume 25, 2012.

[20] R. Kumar and S. Vassilvitskii. Generalized distances between rankings. In *WWW*, page 571–580, 2010.

[21] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in NeurIPS*, pages 598–605, 1990.

[22] N. Lee, T. Ajanthan, S. Gould, and P. Torr. A signal propagation perspective for pruning neural networks at initialization. In *ICLR*, 2019.

[23] N. Lee, T. Ajanthan, and P. Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2018.

[24] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[25] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters for Efficient ConvNets. *arXiv e-prints*, page arXiv:1608.08710, August 2016.

[26] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the Value of Network Pruning. *arXiv e-prints*, page arXiv:1810.05270, 2018.

[27] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through ℓ_0 regularization. In *ICLR*, 2018.

[28] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[29] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th ICLR, ICLR 2017-Conference Track Proceedings*, 2019.

[30] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *ICML*, pages 4646–4655. PMLR, 2019.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[32] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 100(3/4):441–471, 1987.

[33] Suraj Srinivas and R Venkatesh Babu. Generalized dropout. *arXiv preprint arXiv:1611.06791*, 2016.

[34] S. Stich. Local SGD converges fast and communicates little. In *ICLR*, 2019.

[35] C. Wang, G. Zhang, and R. Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2019.

[36] Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *ICML*, pages 6566–6575. PMLR, 2019.

[37] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction, 2021.

[38] H. You, C. Li, P. Xu, Y. Fu, Y. Wang, X. Chen, R. Baraniuk, Z. Wang, and Y. Lin. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.

[39] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G. Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv e-prints*, page arXiv:1909.11957, September 2019.

[40] S. Zagoruyko and N. Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[41] Wenyuan Zeng and Raquel Urtasun. Mlprune: Multi-layer pruning for automated neural network compression.(2019). In *URL https://openreview. net/forum*, 2019.

[42] W. Zhu, C. Zhao, W. Li, H. Roth, Z. Xu, and D. Xu. LAMP: Large Deep Nets with Automated Model Parallelism for Image Segmentation. *arXiv e-prints*, page arXiv:2006.12575, 2020.

[43] M. Zinkevich, M. Weimer, L. Li, and A. Smola. Parallelized stochastic gradient descent. In *Advances in NeurIPS*, pages 2595–2603, 2010.