ON ITERATIVE NEURAL NETWORK PRUNING, REINITIALIZATION, AND THE SIMILARITY OF MASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Iterative pruning methods, such as lottery ticket pruning, provide evidence for the ability to use models with state-of-the-art properties on low-compute devices such as mobile phones. We examine how recently documented, fundamental phenomena in pruned deep learning models is affected by changes in pruning procedure. We address questions of the uniqueness high-sparsity sub-networks and their dependence on pruning method by analyzing differences in connectivity structure and learning dynamics. In convolutional layers, we document the emergence of structure induced by magnitude-based unstructured pruning in conjunction with weight rewinding that resembles the effects of structured pruning.

1 INTRODUCTION

Deep neural architectures have seen a dramatic increase in size (Amodei & Hernandez, 2018). Although not entirely understood, it is known that over-parametrized networks exhibit high generalization performance, with recent empirical evidence showing that the generalization gap tends to close with increased number of parameters (Zhang et al., 2017; Belkin et al., 2018; Hastie et al., 2019; Allen-Zhu et al., 2018; Du et al., 2018; Du & Lee, 2018). While advantageous under this point of view, the proliferation of parameters in neural architectures may induce adverse consequences. The computational cost to train state-of-the-art models has raised the barrier to entry for many researchers hoping to contribute (Le, 2013; Sculley et al., 2018; Strubell et al., 2019). Additionally, the cost of storage and compute has restricted the use of these large-scale models on smaller systems such as wireless devices (Gokhale et al., 2014). This inaccessibility is of particular concern for machine learning applications in the developing world, where many people use mobile phones their only means of internet access (De-Arteaga et al., 2018; Bahia & Suardi, 2019). To enable private, secure computation on devices with lower computational resources, model compression (Breiman et al., 1984; LeCun et al., 1990b; Gong et al., 2014; Hinton et al., 2015) is commonly used.

Recently, there has been increasing interest model compression by pruning, led by the existence of sub-networks with favorable training properties within over-parametrized models, "lottery tickets" (Frankle & Carbin, 2018). To further understanding of the dynamics of pruning, this work investigates the dependence of the properties of these sparse, lucky sub-networks on the choice of pruning technique. We set out to answer the questions:

1. Do pruning methods other than magnitude-based unstructured pruning give rise to winning tickets?
2. If so, do they identify the same lucky sub-network, or are there many equiperforming winning tickets within the same over-parametrized network?

We develop methods to further analyze trainable sparse networks found through iterative pruning techniques, and provide new insight towards a deeper understanding of their properties and of the processes that generate them. Without addressing issues related to scalability and emergence of lottery tickets in large-scale domains, we focus instead on the empirical characterization of weight evolution and emergence of distinctive connectivity patterns in small architectures, such as LeNet (LeCun et al., 1990a), on simple datasets, such as MNIST (LeCun et al., 1994). This also helps develop analysis techniques for sanity checks via visual inspection. We then apply the same techniques to larger architectures and more complex tasks to explore whether findings hold in different regimes.

This work provides empirical evidence showing that:

1. multiple "lottery tickets" can exist within an over-parametrized network;
2. it is possible to find a lucky sub-network through a variety of choices of pruning techniques;
3. lottery ticket-style weight rewinding, coupled with unstructured pruning, gives rise to connectivity patterns similar to the ones obtained with structured pruning along the input dimension, suggesting an input feature selection effect. This is not the case when finetuning;
4. random structured pruning outperforms random unstructured pruning, meaning that networks are more resistant to the removal of random units/channels than to the removal of random individual connections;
5. different iterative pruning techniques learn vastly different functions of their input, and similarly performing networks make different mistakes on held-out test sets, hinting towards the utility of ensembling in this setting.

2 METHOD

All networks in this section are trained for 30 epochs using SGD with constant learning rate 0.01, batch size of 32, without explicit regularization. The pruning fraction per iteration is held constant at 20% of remaining connections/units per layer. Experiment and analysis code is available[1].

2.1 PRUNING METHODS

This works explores a variety of pruning techniques that may differ along the following axes:

Neuronal Importance Definition: In magnitude-based pruning, units/connections are removed based on the magnitude of synaptic weights. Usually, low magnitude parameters are removed. As an alternative, one can remove high magnitude weights (Zhou et al., 2019). Non-magnitude-based pruning techniques can be based other rules for neuronal importance such as activations or gradients.

Local vs. global: Local pruning removes a fixed percentage of units/connections from each layer, comparing each unit/connection to the other units/connections in the layer. Global pruning compares all parameters across layers and selects prunes global fraction, which is beneficial when layers have unequal parameter distribution. A middle-ground approach is to pool together only parameters belonging to layers of the same kind, to avoid mixing, say, convolutional and fully-connected layers.

Unstructured vs. structured: Unstructured pruning removes individual connections, while structured pruning removes entire units or channels. Structured pruning along the input axis is conceptually similar to input feature importance selection. Structured pruning along the output axis is analogous to output suppression. In this work, we compare: magnitude-based $\{L_1, \text{random}\}$ unstructured (US), $\{L_1, L_2, L_{-\infty}, \text{random}\}$ structured (S), and hybrid pruning. The hybrid techniques consists of pruning convolutional layers with L_1 structured pruning and fully-connected layers with L_1 unstructured pruning. "fc-only" identifies experiments in which only the fully-connected layers are pruned. Structured pruning is performed along the input axis. All techniques are local. We implement these pruning techniques and train our models using PyTorch (Paszke et al., 2017).

2.2 FINETUNING VS. REINITIALIZING

A point of contention in the literature revolves around the necessity to rewind weights after each pruning iteration, as opposed to simply finetuning the pruned model. We avoid performance-based arguments in favor of studying how this choice affects the left-over connectivity structure. For comparisons to alternative rewinding techniques, see Appendix A. Unless otherwise specified, all results refer to setup with full weight rewinding. When rewinding the weights according to Frankle & Carbin (2018) and Zhou et al. (2019) biases are pruned; in this work, we do *not* prune biases.

[1] in anonymized form, at `github.com/iclr-8dafb2ab/iterative-pruning-reinit`

3 RESULTS

In all figures, unless otherwise specified, the error bars and shaded envelopes correspond to one standard deviation (half up, half down) from the mean, over 5 experiments with seeds 0-4.

3.1 PRUNING METHODS

We begin by exploring the performance of sub-networks generated by different iterative pruning techniques starting from a base LeNet architecture, where lottery tickets are known to uncontroversially exist and be easy to find. Each point in Fig. 1 represents the test accuracy after exactly 30 epochs of training (not the best test accuracy achieved across the 30 epochs). Multiple techniques are able to identify trainable sub-networks up to high levels of sparsity. For a note on how the fraction of pruned weights is computed, see Appendix B. Different types of magnitude-based structured pruning seem to perform only marginally better than pruning random channels, leading us to conclude that either the channels learn highly redundant transformations and are therefore equivalent under pruning, or there exists a hierarchy of importance among channels but it is not correlated to any of the L_n norms tested in these experiments.

We quantify the overlap in sub-networks found by two different pruning methods started with pairwise-identical initializations by computing the Jaccard similarity coefficient (Intersection over Union) between the masks (Jaccard, 1901). Although the different structured pruning techniques find sub-networks that perform similarly (Fig.1), a deeper investigation into the connectivity structure of the found sub-networks shows that there exist multiple lucky sub-networks with similar performance yet little to no overlap, as evident from the growth in Jaccard distance over pruning iterations in Fig. 3,.

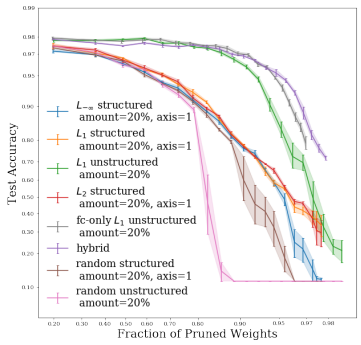


Figure 1: Test accuracy on MNIST test set for SGD-trained LeNet models, pruned using six methods, and rewound to initial weight values after each pruning iteration.

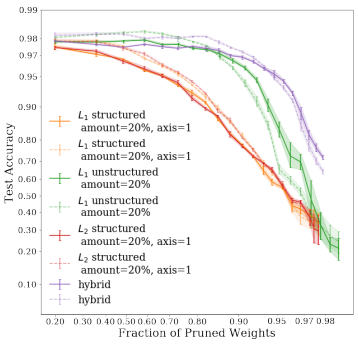


Figure 2: Test accuracy on the MNIST test set for SGD-trained LeNet models, pruned using four methods. The transparent curves correspond to finetuning; the dark ones to rewinding.

The study of the Jaccard distance between masks surfaces another interesting phenomenon: as shown in Fig. 4, the connectivity patterns obtained from magnitude-based *un*structured pruning are relatively more similar to those that one would expect from structured pruning along the input dimension, especially in the convolutional layers, than to random unstructured patterns. This suggests that a form of input (or, at times, especially in larger models, output) feature selection is automatically being learned. This observation is further confirmed via visual inspection of the pruning masks; the first two columns of Fig. 5 show the weights and masks in the second convolutional layer of LeNet, obtained by applying structured and unstructured L_1 pruning. Unstructured pruning ends up automatically removing entire rows of filters corresponding to unimportant input channels. Structured pruning, instead, in this case, does so because, in these experiments, it is explicitly instructed to prune along the input dimension. This suggests that the low-magnitude weights pruned by L_1-unstructured pruning may be those that process low-importance hidden representations and whose outputs contribute the least to the network's output, which in turn results in lower gradients and smaller weight updates, causing these weights to remain small, and thus be subject to pruning.

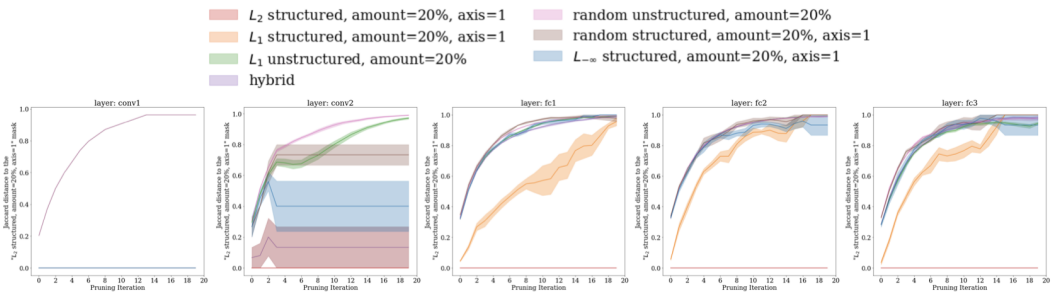


Figure 3: Jaccard distance between the masks (*i.e.* connectivity structures) found by pruning LeNet using L_2-structured pruning, and those found by other pruning methods listed in the legend, conditional on identical seed for meaningful comparison in light of neural network degeneracy. The comparison is conducted for each layer individually. L_1-structured pruning yields the most similar masks to L_2-structured pruning, as expected.

Not only do the different pruning methods lead to different masks, but they also lead to sub-networks that learn partially complementary solutions, opening up the opportunity for the ensembling of different sparse sub-networks. Table 2 records the average accuracy (over the 5 experiment seeds) of sub-networks obtained at each pruning iteration through a set of pruning techniques, as well as the average accuracy obtained after simply averaging all their predictions, or the predictions of structured and unstructured L_1 pruning. Note that, after each pruning iterations, the levels of sparsity in the sub-networks will be different as they depends on the type of pruning applied.

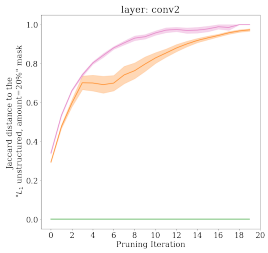


Figure 4: Jaccard distance between the mask for the second convolutional layer found by pruning LeNet using L_1-unstructured pruning, and the mask found by other pruning methods listed in the legend above, conditional on identical seed. Surprisingly, L_1-unstructured behaves more like L_1-structured pruning than a random unstructured pattern. This behavior is further investigated in Fig. 5.

3.2 Finetuning vs. Reinitializing

The "Lottery Ticket Hypothesis" Frankle & Carbin (2018) postulates that rewinding weights to the initial (or early-stage (Frankle et al., 2019)) values after pruning is key to identifying lucky sub-networks. In this section, we attempt to test this hypothesis in the experimental setting of small networks (LeNet) and simple tasks (MNIST). We investigate, at the individual parameter level, how the this procedure ends up differing from finetuning after pruning. Although higher quality lottery tickets can be obtained by rewinding the weight values to their value after a small number of training iterations (Frankle et al., 2019), this method was not tested in this work. To promote understanding, we do not to focus on extracting state-of-the-art performing lottery tickets, in favor of minimizing the influence of exogenous choices and ad-hoc heuristics for last-mile performance gains.

Finetuning is found to yield pruned networks that are comparable in performance with reinitialized networks (at least in the simple-task-small-network regime) when given the same training budget and the networks are pruned using the same technique, as shown in Fig. 2. When finetuning with L_1-unstructured pruning, the magnitude of weights tends to continue growing, achieving higher final values at later pruning iterations than their reinitialized counterparts. Despite achieving similar average performance in this simplified scenario, the interesting observation lies in the stark difference between masks created by the same pruning technique when the weights are rewound or finetuned.

(a) 2^{nd} iteration of L_1-S pruning with rewinding

(b) 5^{th} iteration of L_1-US pruning with rewinding

(c) 5^{th} iteration of L_1-US pruning with $\sigma \cdot \text{sign}(w_i)$ reinitialization

(d) 5^{th} iteration of L_1-US pruning with finetuning

(e) 5^{th} iteration of L_1-S pruning with rewinding

(f) 10^{th} iteration of L_1-US pruning with rewinding

(g) 10^{th} iteration of L_1-US pruning with $\sigma \cdot \text{sign}(w_i)$ reinitialization

(h) 10^{th} iteration of L_1-US pruning with finetuning

Figure 5: Masked weights in the 2^{nd} convolutional layer of LeNet. Rows represent two different pruning iterations. The columns represent four different pruning and weight treatments. The convolutional layer has 6 input channels and 16 output channels, with 3×3 filters. Masked out weights appear in gray. Active, positive weights are depicted in red, negative weights in blue.

Fig. 5 shows that finetuning removes the natural tendency of L_1-unstructured pruning with reinitialization to approximate the behavior of L_1-structured pruning. We can therefore primarily attribute the emergence of that interesting connectivity pattern to the choice of reinitializing the weights. In terms of mask structure, the difference between the effects of finetuning and lottery-ticket reinitialization grows across all layers and logarithmically with pruning iterations (see Appendix A).

3.3 Do these observations hold in larger networks and domains?

These empirical results have been observed to hold in AlexNet and VGG-11 architectures on MNIST and CIFAR-10 (see Appendix E). In even larger models, the nature of lottery tickets is still contended and special care is required in ad-hoc training and pruning procedures to facilitate their discovery.

4 Conclusion

We show evidence against the uniqueness of winning tickets in various networks and tasks, identifying different lucky sub-networks of competitive performance within the same parent network and controlling for degeneracy by fixing experimental seeds. We also provide empirical results showing that rewinding weights to the original values at initialization after each pruning iteration yields sparsified networks that may not only be superior to finetuned sparse models from a performance perspective, but also appear to possess structural advantages that might make them more suitable for hardware-level implementations with inference-time speed-up effects.

We offer methods to experimentally analyze and compare the effects of different pruning techniques on the performance of neural networks and on the nature of the masks generated by each. With the explicit intent not to adopt training tricks to induce lottery tickets in state-of-the-art deep architectures, this work intentionally restricts its focus to smaller models and domains. This work can be further extended to better understand the role of new pruning methods, training tricks such as late resetting, and applcations beyond image classification in terms of the similarity of masks obtained.

References

Z. Allen-Zhu, Y. Li, and Y. Liang. Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers. *arXiv e-prints*, November 2018.

Dario Amodei and Danny Hernandez. AI and compute. May 2018. URL `https://openai.com/blog/ai-and-compute/`.

Kalvin Bahia and Stefano Suardi. The state of mobile internet connectivity 2019. Technical report, GSM Association, 2019.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine learning and the bias-variance trade-off. *arXiv e-prints*, December 2018.

L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984. ISBN 9780412048418. URL https://books.google.com/books?id=JwQx-WOmSyQC.

M De-Arteaga, W Herlands, D B Neill, and others. Machine learning for the developing world. *ACM Transactions on Management Information Systems*, 2018. URL https://dl.acm.org/doi/abs/10.1145/3210548.

Simon S. Du and Jason D. Lee. On the Power of Over-parametrization in Neural Networks with Quadratic Activation. *arXiv e-prints*, art. arXiv:1803.01206, Mar 2018.

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient Descent Provably Optimizes Over-parameterized Neural Networks. *arXiv e-prints*, art. arXiv:1810.02054, Oct 2018.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. March 2018. URL http://arxiv.org/abs/1803.03635.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. The lottery ticket hypothesis at scale. March 2019. URL http://arxiv.org/abs/1903.01611.

V Gokhale, J Jin, A Dundar, B Martini, and E Culurciello. A 240 g-ops/s mobile coprocessor for deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 696–701, June 2014. URL http://dx.doi.org/10.1109/CVPRW.2014.106.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. December 2014. URL http://arxiv.org/abs/1412.6115.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J. Tibshirani. Surprises in High-Dimensional Ridgeless Least Squares Interpolation. *arXiv e-prints*, art. arXiv:1903.08560, Mar 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop (2015)*, March 2015. URL http://arxiv.org/abs/1503.02531.

Paul Jaccard. Etude de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 01 1901. doi: 10.5169/seals-266450.

Q V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8595–8598. ieeexplore.ieee.org, May 2013. URL http://dx.doi.org/10.1109/ICASSP.2013.6639343.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, R E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a Back-Propagation network. In D S Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, pp. 396–404. Morgan-Kaufmann, 1990a. URL http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In D S Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, pp. 598–605. Morgan-Kaufmann, 1990b. URL http://papers.nips.cc/paper/250-optimal-brain-damage.pdf.

Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. *The MNIST database of handwritten digits.*, 1994. URL http://yann.lecun.com/exdb/mnist/.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

D Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. Winner's curse? on pace, progress, and empirical rigor. In *ICLR 2018 Workshop*, February 2018. URL https://openreview.net/forum?id=rJWF0Fywf.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3645–3650. aclweb.org, 2019. URL https://www.aclweb.org/anthology/P19-1355.pdf.

Chiyuan Zhang, Qianli Liao, Alexander Rakhlin, Karthik Sridharan, Brando Miranda, Noah Golowich, and Tomaso Poggio. No . 067 april 4 , 2017 theory of deep learning iii : Generalization properties of sgd by. 2017.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In H Wallach, H Larochelle, A Beygelzimer, F dAlché-Buc, E Fox, and R Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 3592–3602. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8618-deconstructing-lottery-tickets-zeros-signs-and-the-supermask.pdf.

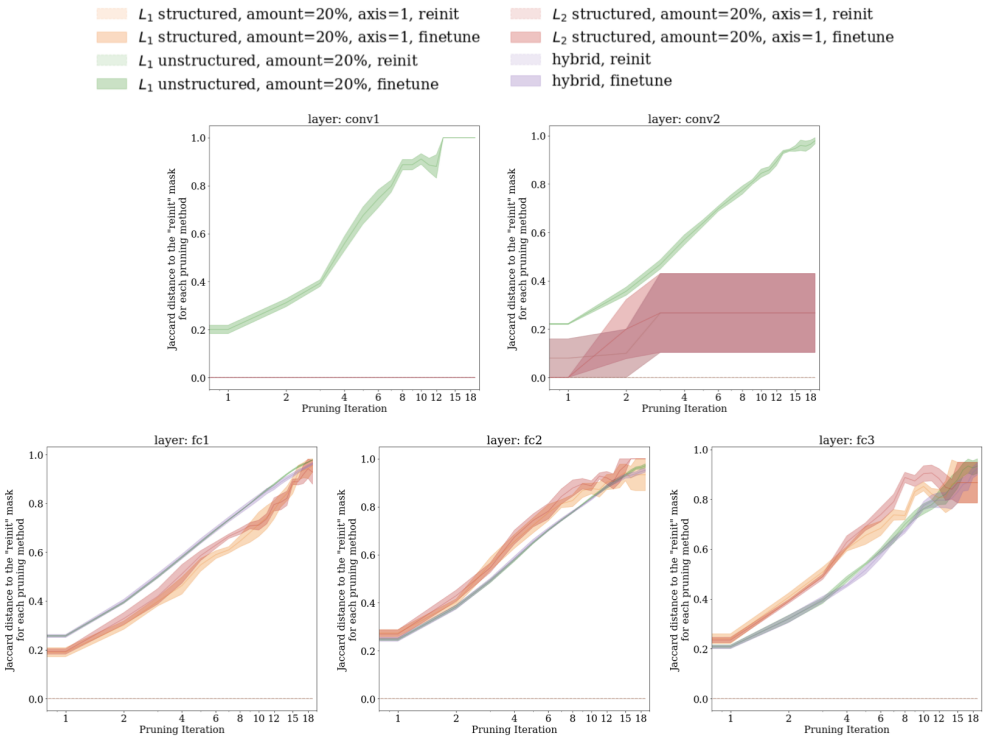


Figure 6: Jaccard distance between the masks found by pruning LeNet and rewinding weights, and those found by finetuning after pruning, conditional on identical pruning techniques and seeds. The comparison is conducted for each layer individually. Note the logarithmic scale on the x-axis.

A FINETUNING VS. REINITIALIZING (CONTINUED)

The Jaccard distance, or any other similarity measure among masks (*e.g.*, the Hamming distance), can be adopted to quantify the effects of the choice of finetuning or reinitializing weights after pruning. As expected, the nature of the connectivity structure that emerges in an iterative series of pruning and weight handling steps depends not only on the pruning choice but also on how weights are handled after pruning. Finetuning yields significantly different masks from rewinding, for all pruning techniques, and the difference (quantified in terms of the Jaccard distance) appears to grow logarithmically in the number of pruning iterations (Fig. 6).

A.1 SIGN-BASED REINITIALIZION

Does the exact weight value at initialization carry any significance, or is all we care about its sign?

As proposed by Zhou et al. (2019), we experiment with reinitializing the pruned sub-network by resetting each unpruned parameter i to $\sigma_{L_i} \cdot \text{sign}(w_i)$, where w_i is the weight value of parameter i at initialization, and σ_{L_i} is the empirical standard deviation of the weights in layer L that contains the parameter i. This is in contrast to rewinding to the initial weight values w_i, as originally suggested by Frankle & Carbin (2018), or finetuning without weight resetting.

The overall test accuracy of our experiments (Fig. 7) seems to support the observation of Zhou et al. (2019) that, as long as the sign matches the sign at initialization (*and* special care is taken in re-scaling the standard deviation), resetting the weights to different values won't affect the sub-networks trainability and performance. Removing the factor of σ causes the network not to converge; as expected, keeping the weights within a numerically favorable range is important, and naively focusing on the sign alone leads to poor performance.

Adopting mask similarity to test for equivalence among reinitialization techniques, the method of Zhou et al. (2019) is found to induce structure that differs more substantially from the connectiv-

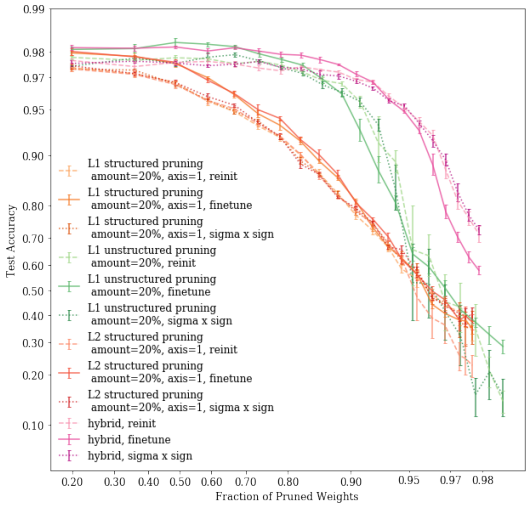


Figure 7: Test accuracy comparison on the MNIST test set for SGD-trained LeNet models, pruned using four different pruning techniques corresponding to the four colors, and reinitialized using three different techniques corresponding to the line styles.

	0	0.2	0.4	0.5	0.6	0.7	0.8	0.9	0.95	0.97	0.98	0.99
LeNet	60	48	36	30	24	18	12	6	3	2	1	0.6
AlexNet	61,100	48,880	36,660	30,550	24,440	18,330	12,220	6,110	3,055	1,833	1,222	611
VGG11	132,863	106,290	79,718	66,431	53,145	39,859	26,572	13,286	6,643	3,985	2,657	1,328

Table 1: Number of parameters (in thousands) left in each model at each pruning fraction.

ity patterns generated by the method of Frankle & Carbin (2018) than simple finetuning does, across most layers and a set of common choices of magnitude-based pruning techniques (Fig. 8).

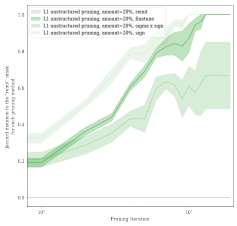
B Effective Pruning Rate

A subtle implementation detail involves the way in which the fraction of pruned weights is computed, especially in convolutional layers pruned with structured pruning techniques. Take, for example, the LeNet architecture used in this work. When applying pruning to produce results like the ones displayed in Fig. 1, there are two ways to define the number of weights that get pruned (displayed along the x-axis). The definition used in the main portion of the paper uses the fraction of weights *explicitly* pruned by the decision rule that produces the mask for each layer. However, this doesn't account for the *implicitly* pruned weights, *i.e.* those weights that, due to downstream pruning in following layers, are now disconnected from the output of the neural network. For this reason, they receive no gradient and their value never changes from that at initialization. They can technically take on any value, including 0, without affecting the output of the neural network. In other words, they could effectively be pruned without any loss of performance. If we include these weights in the effective fraction of pruned weights, Fig. 1 (now converted into Fig. 9(a) for visual coherence) is modified into Fig. 9(b). When looking at effective sparsity, structure pruning appears to be competitive, especially at high pruning fractions, because its effective sparsity naturally tends to be higher. This also suggests that since the effective pruning rate per iteration is higher than 20% for structured pruning techniques, one could experiment with lowering it, to avoid aggressive pruning and consequent loss in performance.

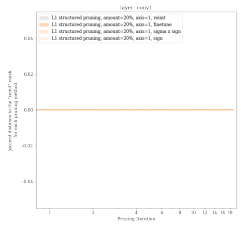
The number of parameters corresponding to various fractions of pruned weights in the models investigated in this work is listed in Table 1.

C Pruning After 1 Training Epoch

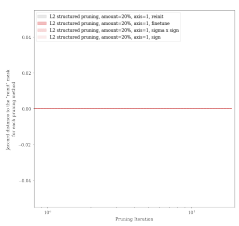
Is it necessary to train for many epochs before pruning to find high quality masks?



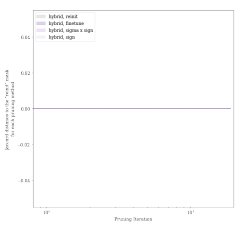
(a) First convolutional layer



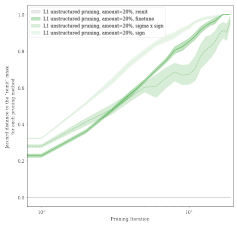
(b) First convolutional layer



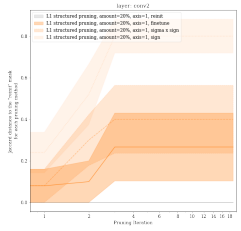
(c) First convolutional layer



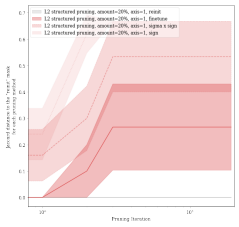
(d) First convolutional layer



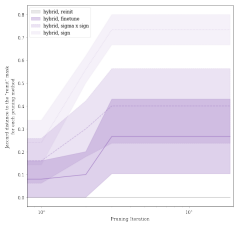
(e) Second convolutional layer



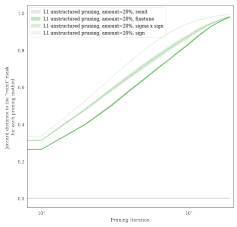
(f) Second convolutional layer



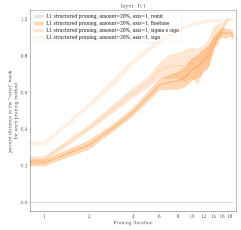
(g) Second convolutional layer



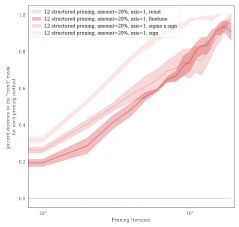
(h) Second convolutional layer



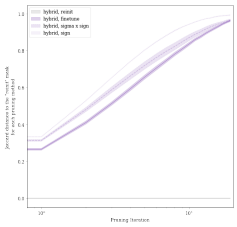
(i) First fully-connected layer



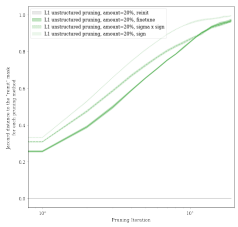
(j) First fully-connected layer



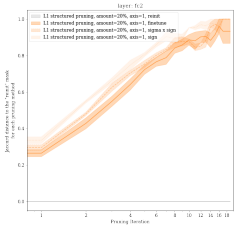
(k) First fully-connected layer



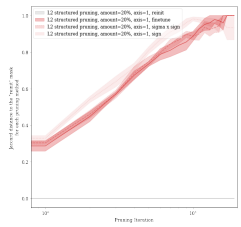
(l) First fully-connected layer



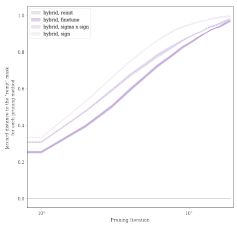
(m) Second fully-connected layer



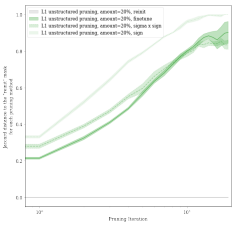
(n) Second fully-connected layer



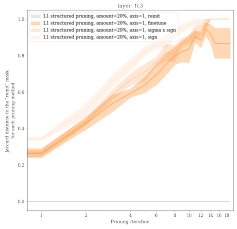
(o) Second fully-connected layer



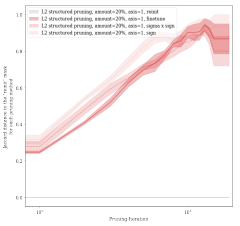
(p) Second fully-connected layer



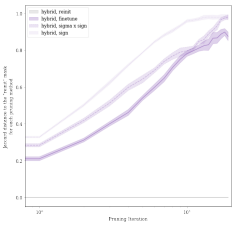
(q) Third fully-connected layer



(r) Third fully-connected layer

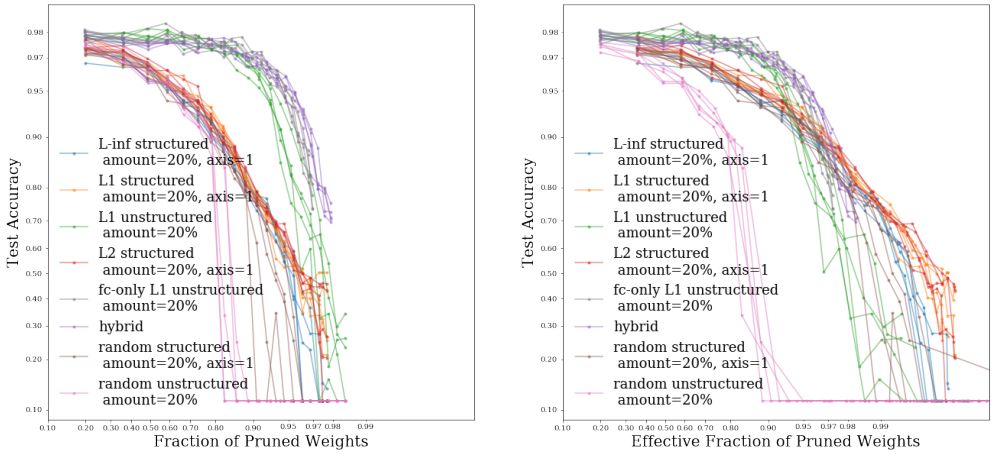


(s) Third fully-connected layer



(t) Third fully-connected layer

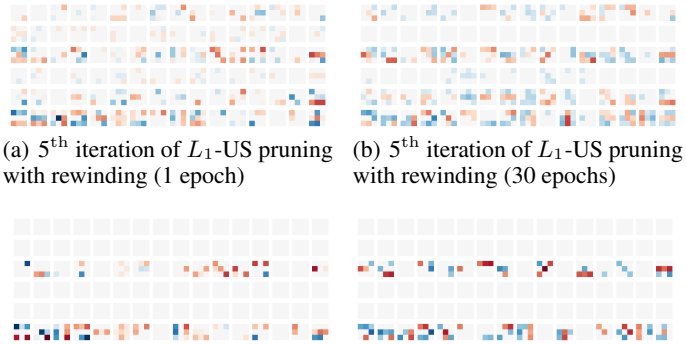
Figure 8: Growth of the Jaccard distance between the mask found by rewinding after pruning and three other techniques of handling weights (in order from highest to lowest opacity in the figures: finetuning, rewinding to $\sigma \cdot \text{sign}(w_i)$, and rewinding to $\text{sign}(w_i)$. Each column of sub-figures corresponds to a pruning technique (same color code as in the rest of the paper); each row corresponds to a layer in LeNet. Note the logarithmic scale on the x-axis.



(a) Number of pruned weights corresponds to locations where the mask is explicitly zero.

(b) Unchanged units included in the effective number of pruned weights.

Figure 9: Test accuracy achieved after 30 training iterations by LeNet models trained with SGD and pruned with the methods listed in the legend. Each dot corresponds to a version of the model at the end of training. Each line connects models obtained by iterative pruning from the same seed. The two plots differ in the way the fraction of pruned weights is computed.



(a) 5^{th} iteration of L_1-US pruning with rewinding (1 epoch)

(b) 5^{th} iteration of L_1-US pruning with rewinding (30 epochs)

(c) 10^{th} iteration of L_1-US pruning with rewinding (1 epoch)

(d) 10^{th} iteration of L_1-US pruning with rewinding (30 epochs)

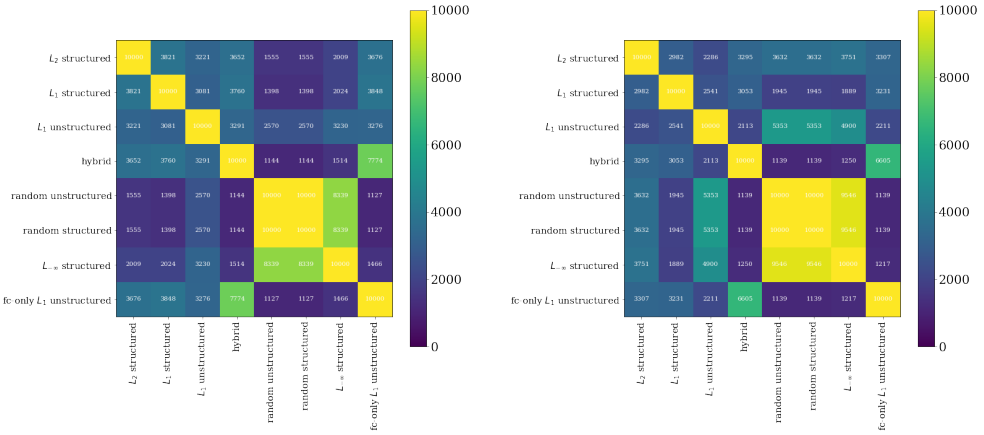
Figure 10: Difference in mask structure in the second convolutional layer that emerges when pruning after a single epoch (left) versus 30 epoch (right) of training (seed: 0).

To speed up the winning ticket search, we evaluate the option of shortening the training phases to only 1 epoch of training per iteration.

We compare the masks obtained with this quicker method to the masks found after training the network at each pruning iteration for 30 epochs (Fig. 10). After a single epoch of training, the ordinality of weight magnitudes has not yet settled to the solution corresponding to the values after 30 epochs of training. Although the majority of ranking swaps among weights, which can bring parameters from the lowest magnitude quartile all the way to the top one, and vice versa, happen in the first few epochs of training, minor movements later on in training might still be highly relevant in obtaining a robust ranking for magnitude-based pruning.

D Complementarity of Learned Solutions

Pruning the same network using different pruning techniques gives rise to sparse sub-networks that differ not only in structure but also in the learned function that they compute. Given sufficient com-



(a) 18^{th} pruning iteration (b) 20^{th} pruning iteration

Figure 11: Number of examples in the MNIST test set over which the sub-networks obtained through each pruning technique agree on the prediction, on average (over 5 experimental seeds).

Pruning Iteration	L_2 S	L_1 S	L_1 US	hybrid	random US	random S	$L_{-\infty}$ S	fc-only L_1 US	all	hybrid + fc-only + L_1 US
1	97.4	97.5	97.8	97.8	97.4	97.3	97.1	97.9	98.2	98.1
2	97.2	97.0	97.7	97.6	97.0	96.9	96.9	97.8	98.2	97.9
3	96.5	96.6	97.8	97.5	96.1	96.4	96.2	97.7	98.1	97.9
4	95.8	95.6	97.9	97.6	95.6	95.1	95.4	97.8	97.9	97.9
5	95.0	95.1	97.6	97.5	93.8	94.0	94.1	97.8	97.7	97.9
6	94.2	93.8	97.6	97.4	92.2	92.9	93.3	97.6	97.6	97.7
7	91.7	92.7	97.4	97.5	89.2	91.0	91.3	97.6	97.4	97.8
8	89.5	90.9	97.3	97.4	45.8	88.2	88.9	97.5	97.2	97.7
9	87.6	86.9	97.0	97.3	14.0	83.7	86.2	97.5	97.2	97.5
10	82.0	82.2	96.5	96.9	11.3	79.2	81.3	97.3	96.8	97.3
11	77.2	77.7	95.9	96.8	11.3	60.1	76.3	97.0	96.6	97.1
12	72.5	72.5	94.5	96.4	11.3	45.5	74.1	96.9	96.3	97.0
13	69.0	65.1	90.2	95.8	11.3	41.1	65.6	96.1	95.7	96.4
14	65.4	58.3	83.5	95.3	11.3	32.0	58.4	95.9	95.1	96.3
15	55.7	54.7	72.7	94.4	11.3	18.5	48.3	94.7	94.6	95.7
16	47.1	43.7	69.5	92.5	11.3	11.3	24.7	93.3	94.0	94.7
17	45.9	41.6	47.9	88.3	11.3	11.3	21.9	90.7	91.9	92.8
18	36.7	37.8	32.4	81.8	11.3	11.3	14.5	87.4	91.0	91.6
19	30.2	35.9	23.0	76.4	11.3	11.3	11.9	84.3	89.3	90.3
20	29.4	33.1	21.2	71.9	11.3	11.3	11.7	78.6	85.0	86.0

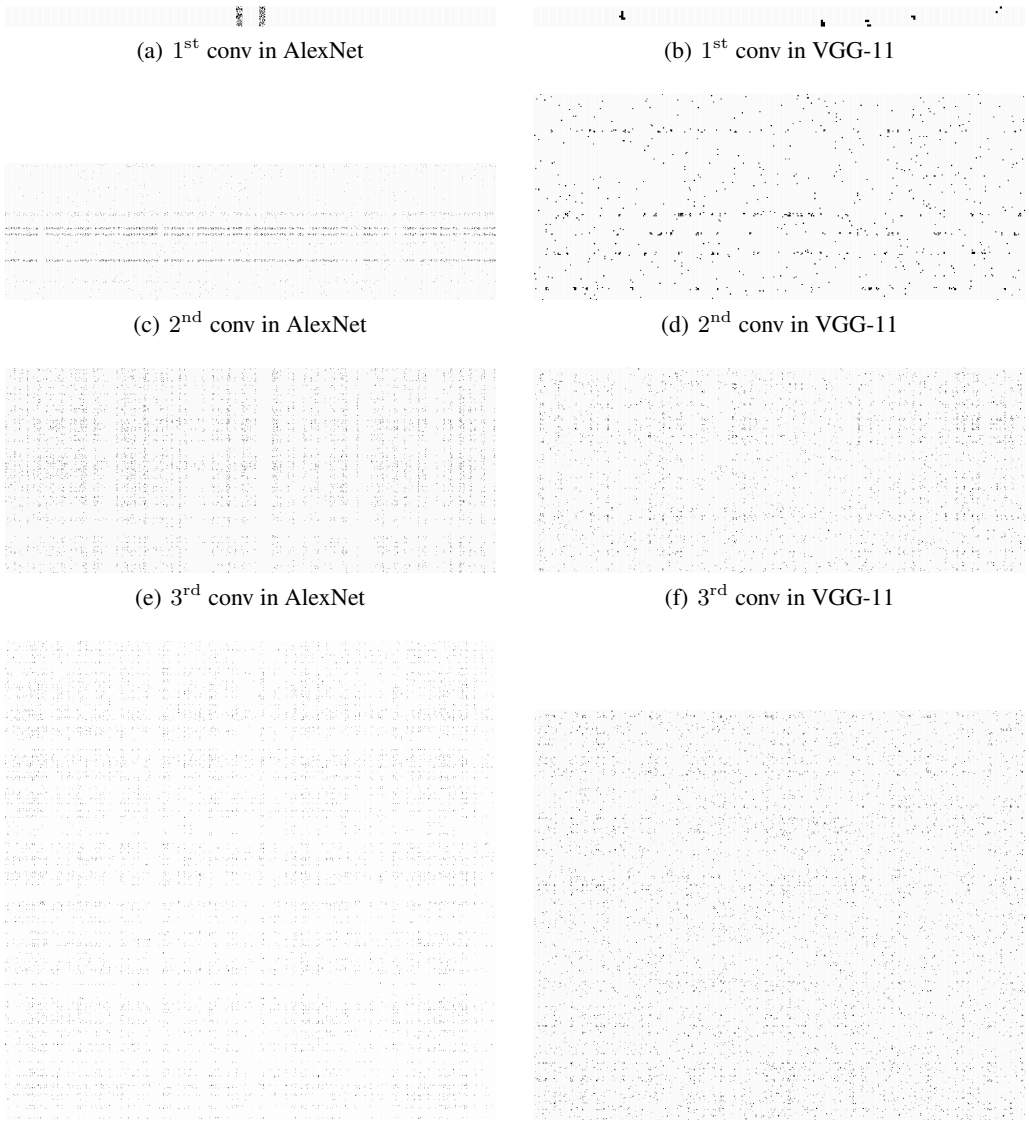
Table 2: Sub-network accuracies at each pruning iteration. Ensembling of sub-networks obtained through different pruning techniques can yield higher performance, hinting at the complementarity of information learned by each sub-network.

pute and memory budget, one can consider combining the predictions made by each sub-network to boost performance. On the left side of Table 2, for each pruning iteration, the average accuracy of a pruned LeNet model is listed, along with, on the right, the accuracies obtained by simply averaging the predictions of all eight individual sub-networks ("all") and by averaging the predictions obtained from the three most promising pruning techniques (last column).

The similarity of solutions can also be explored by looking at the heat maps of the agreement in average class prediction across sub-networks obtained through different pruning techniques. For reference, Fig. 11 provides these visualizations for the 18^{th} and 20^{th} pruning iterations.

E ALEXNET AND VGG ON MNIST AND CIFAR-10

In this section, we confirm qualitative observations, previously reported on LeNet models, on the structure of connectivity patterns that emerge from the application of L_1 *un*structured pruning in the context of AlexNet and VGG-11 architectures.



(a) 1^{st} conv in AlexNet

(b) 1^{st} conv in VGG-11

(c) 2^{nd} conv in AlexNet

(d) 2^{nd} conv in VGG-11

(e) 3^{rd} conv in AlexNet

(f) 3^{rd} conv in VGG-11

(g) 4^{th} conv in AlexNet

(h) 4^{th} conv in VGG-11

Figure 12: Binary weight masks for the first four convolutional layers of AlexNet trained on MNIST (left) and VGG-11 trained on CIFAR-10 (right), at the 20th and last pruning iteration for L_1 unstructured pruning (seed: 0). Despite the unstructured nature of the pruning technique, structure emerges along the input and output dimensions, which resembles the effect of structured pruning.

We train two individual sets of experiments starting from the base AlexNet model, one on MNIST, one on CIFAR-10. VGG models are trained on CIFAR-10 exclusively.

Fig. 12 shows the binary masks obtained in the first four convolutional layers of the two models after 20 pruning iterations, with pruning rate of 20% of remaining connections at each iteration. Here, the reported AlexNet properties refer to the MNIST-trained version. Preferential structure along the input and output dimensions (in the form of rows or columns of unpruned filters) is visible across the various layers, although visual inspection becomes hard and inefficient as the number of parameters per layer grows.