RECONCILING SPARSE AND STRUCTURED PRUNING: A SCIENTIFIC STUDY OF BLOCK SPARSITY

Arlene Siswanto
MIT CSAIL
siswanto@mit.edu

Jonathan Frankle
MIT CSAIL
jfrankle@mit.edu

Michael Carbin
MIT CSAIL
mcarbin@mit.edu

ABSTRACT

We systematically study block sparsity as a middle ground that interpolates between (a) sparse pruning of individual weights in an unstructured manner and (b) structured pruning at larger granularities, such as neurons or channels. These two forms of pruning are known to exhibit different properties. Unstructured pruning reaches higher sparsities than neuron or channel pruning before losing accuracy; however, neuron and channel pruning is easier to exploit to reduce the real-world costs of inference. We investigate block sparsity as a potential middle ground between these alternatives. Unstructured pruning is also known to be sensitive to reinitialization, while neuron- and channel-pruned networks can train to the same accuracy even after being reinitialized. We use block sparsity to examine the extent to which these two regimes apply at intermediate pruning granularities. We observe an intrinsic tradeoff between granularity and accuracy: increasing block size results in a gradual decrease in accuracy, even for the smallest blocks. Surprisingly, we also obtain similar accuracy across all pruning granularities when reinitializing; for larger blocks, this is closer to the accuracy without reinitializing, potentially explaining the neuron and channel pruning results.

1 INTRODUCTION

Pruning is a well-studied compression technique for removing up to 80-95% of weights in a neural network with little or no loss in model accuracy (LeCun et al., 1990; Han et al., 2015). The random connectivity resulting from the *unstructured pruning* of individual weights, however, may result in negligible or even reduced performance gains due to irregular memory accesses (Elsen et al., 2020; Wen et al., 2016). As models become larger, recent efforts have been made towards the acceleration of sparse matrix computation (Pool, 2020; Gray et al., 2017; Lagunas, 2020b).

Structured pruning, which follows rules that govern permissible sparsity patterns, can more easily exploit pruning for performance improvements. Constraining the positions of nonzero values within a weight matrix through structured pruning can facilitate such large contiguous memory accesses while still reducing overall computation (Lagunas, 2020a; Yang et al., 2019). Accessing GPU memory takes an order of magnitude longer than floating point calculations; as a result, accessing several values at once and performing calculations in parallel tends to increase efficiency.

Conventional structured pruning differs from unstructured pruning in more ways than just its effects on performance. Notably, prior work demonstrates that pruning entire neurons and entire channels substantially decreases the maximum sparsity obtainable without affecting accuracy. In addition, Liu et al. (2019) show that randomly reinitializing neuron- and channel-pruned networks does not affect test accuracy, suggesting that the learned values of weights does not hold significance; the same behavior cannot be found in unstructured pruning (Han et al., 2015; Frankle & Carbin, 2019).

Research questions. We seek to understand the middle ground between unstructured and structured pruning. We do so through *block sparsity*, a standard technique that prunes the network in blocks of granularities that can range in size from a handful of weights to an entire neuron (Figure 1). In particular, we study the following questions:

1. For a given block, what is the maximum sparsity attainable while matching the accuracy of the original trained network (*maximal matching sparsity*)?

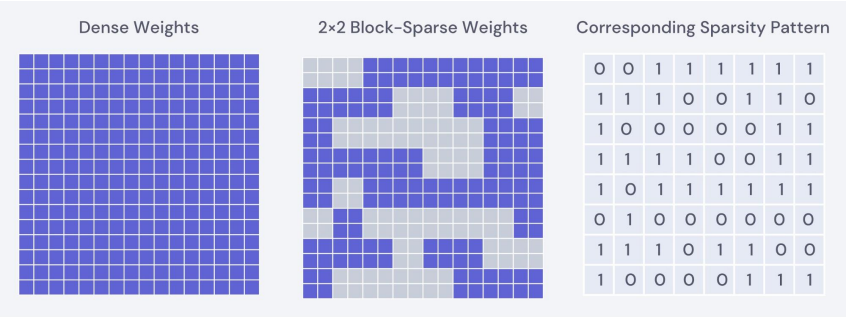

Figure 1: Weights from block-sparse pruning with 2×2 blocks. Blocks take on aggregate scores based on the weights that reside within them, and entire blocks are pruned according to this score.

2. What explains the discrepancy between the effect of random reinitialization on unstructured pruning and highly structured pruning?

Many prior works have proposed various kinds of block sparsity (e.g., Mao et al., 2017; Li et al., 2017; Narang et al., 2017; Wen et al., 2016; Anwar et al., 2015; Elsen et al., 2020); we systematically study the effect of block sparsity on accuracy across a range of patterns and granularities. Completely novel to our work, we use these results as a basis for probing the lingering and mysterious disconnect between the role of reinitialization in unstructured and structured pruning.

Results. Our experiments uncover a tradeoff between accuracy and block size—consistent with pruning literature. Increasing block size gradually decreases accuracy; there is no singular size at which the drop in accuracy notably spikes. We also make the surprising observation that, for any given sparsity, accuracy when randomly reinitializing pruned networks is similar across pruning granularities at all but the very highest sparsities, in all cases at or below that of using the learned weights. Based on these results, we believe that the observation of Liu et al. (2019) about the apparent resilience of neuron- or channel- pruned networks to reinitialization is a result of accuracy decreasing dramatically to the point at which it is no better than this shared reinitialized performance.

2 Methodology

We apply block-sparse pruning to two benchmark image classification tasks to compare maximal matching sparsities. The first is a fully connected LeNet with 256 neurons in each of two hidden layers on MNIST (LeCun et al., 1998). Fully connected layers are stored as 2-dimensional tensors; blocks of choice are squares and long strips with power-of-two side lengths. The second is a ResNet-20 (He et al., 2015) on CIFAR-10 dataset (Krizhevsky et al.). Convolutional layers in this residual network are stored as 4-dimensional tensors; blocks of choice are those within individual 3×3 kernels as well as blocks spanning power-of-two kernels. We follow the standard three-step pipeline (Han et al., 2015), removing roughly 20% of the least important weights during each of 15-20 runs:

1. *Train the network and determine which weights are important.* The importance of a weight is the mean of the magnitudes of all weights residing within the same block (L1/lasso).
2. *Prune the globally least important weights.* We use a *binary mask* $M \in \{0, 1\}^d$ to remove the least important weights from the network. Given a parameterization θ' of a network, the result of $M \odot \theta'$, where $\odot$ is the elementwise product operator, is the new parameterization in which the values of pruned weights are set to 0.
3. *Fine-tune the network.* We train the unpruned weights using previously learned weight values by replaying the learning rate schedule (Renda et al., 2020).

3 The Effect of Block-Sparse Pruning on Accuracy

Understanding how various block sizes affect pruned accuracy offers insight into how to best maximize efficiency gains from sparse computation. For example, the largest block with the same maximal matching sparsity as unstructured pruning could immediately be used as a drop-in replacement.

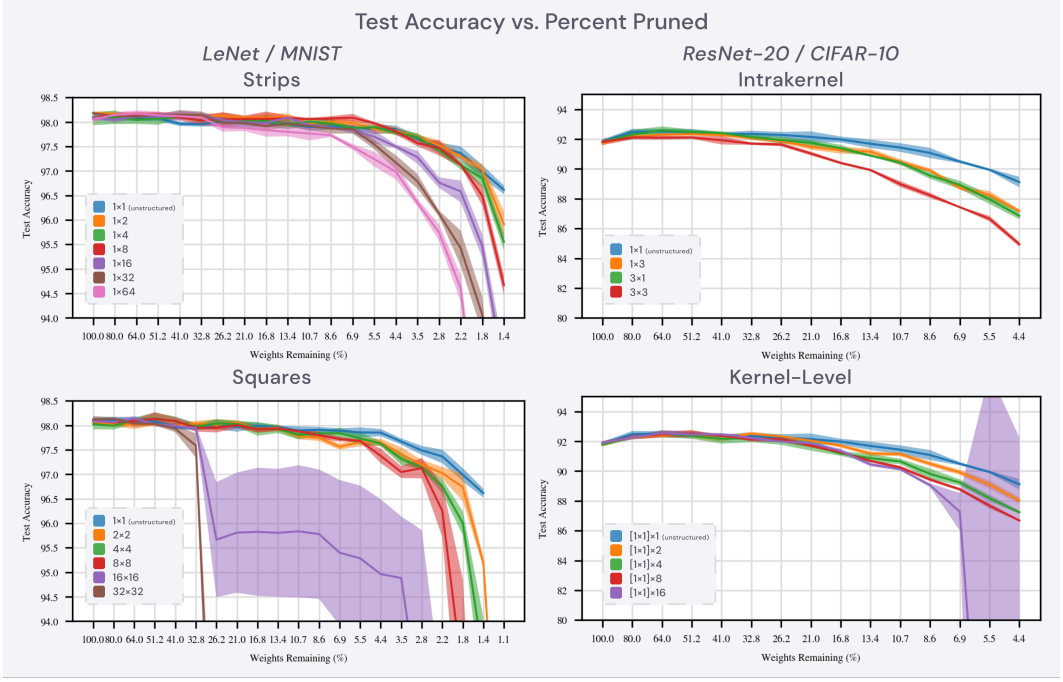

Figure 2: Effect of varying block granularity on accuracy. *Left:* Fully connected LeNet trained on MNIST. Blocks within a single kernel and blocks spanning power-of-two kernels are examined. *Right:* ResNet-20 trained on CIFAR-10. Power-of-two squares and long strips are examined.

Results. Figure 2 shows how test accuracy changes as a function of weights remaining for each block granularity. For MNIST, unstructured pruning has 5.5% of weights remaining at maximal matching sparsity. This sparsity is not matched by block-sparse pruning for any block size and shape. We obtain similar results from our CIFAR-10 experiments. Any choice of block leads to a lower maximal matching sparsity than the 13.4% of weights remaining in unstructured pruning. We conclude that there is a tradeoff between block size and accuracy; we cannot benefit from block structure without compromising accuracy. Upon further analysis, we also observe that square blocks obtain higher accuracy than long strips containing an equal number of weights, and that long blocks spanning multiple kernels obtain higher accuracy than blocks within a single kernel containing an equal number of weights. This is consistent with the hypothesis that scattering pruning across components (e.g., neurons or kernels) is preferable to removing many weights from a single component.

4 Reinitializing Block-Sparse Pruned Networks

In the pruning literature, it is conventional to retain the weights learned during network training for the fine-tuning phase (Han et al., 2015). In the context of unstructured pruning, Han et al. (2015) and Frankle & Carbin (2019) show that reinitializing the weights of pruned networks leads to lower final accuracy. However, Liu et al. (2019) find the opposite for neuron and channel pruning: randomly reinitializing the remaining unpruned weights has no impact on accuracy and trained accuracy from random reinitialization is no different from the learned weights. In this section, we investigate the extent of these two regimes by using block sparsity as a way to interpolate between the extremes of unstructured pruning and neuron or channel pruning. To do so, we repeat the experiments in Section 2, randomly reinitialize sample new weight values from the initialization distribution for the unpruned weights, fine-tune the network, and compare results to the original procedure.

Results. Comparing our original experiments to their randomly reinitialized counterparts, our primary observation is that random reinitialization results in decreased accuracy for both tasks. This can be seen in Figure 3. This behavior is consistent across all block sizes and shapes we tried, with the gap between the original and randomly reinitialized runs decreasing as block size increases. This suggests that the learned weight values remain important; we cannot simply take a block-

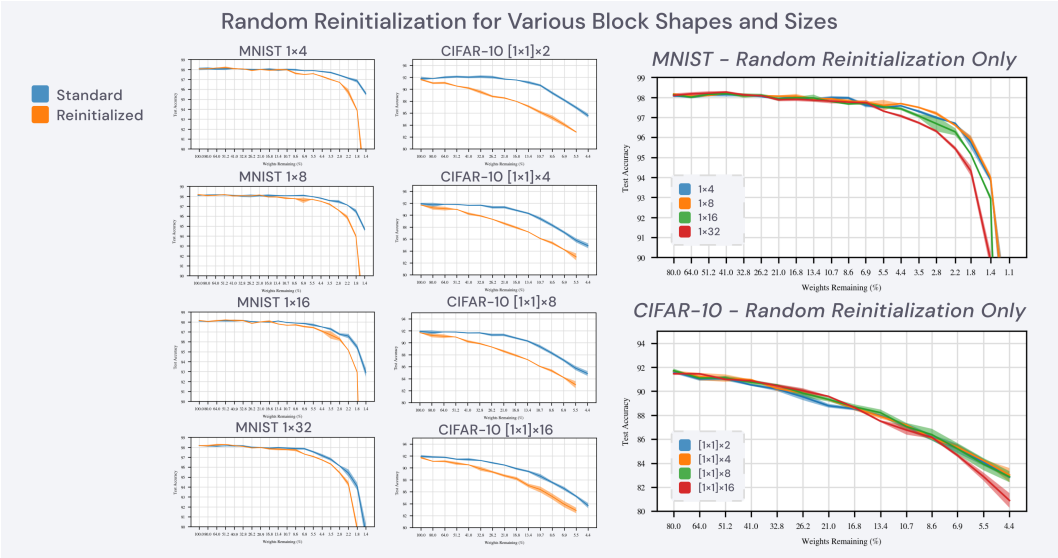

Figure 3: Random reinitialization with LeNet trained on MNIST and ResNet-20 trained on CIFAR-10 using various block granularities. *Left:* Individual comparison with non-randomly reinitialized baseline. *Right:* Comparison among randomly reinitialized curves only.

sparse pruned neural network, randomly reinitialize it, and expect it to train to the full accuracy of the original network.

Interestingly, for the block structures under consideration, random reinitialization seems to obtain similar accuracy across all granularities at all but the highest sparsities. Figure 3 (right) provides an aggregate comparison of randomly reinitialized networks across the various block shapes and sizes. The implication is that the granularity of pruning does not substantially affect accuracy after reinitialization as long as the network is pruned to the same sparsity. This result is not obvious: one might predict that randomly reinitializing a network would have led to a higher accuracy for smaller block sizes and a lower accuracy for larger block sizes, just as we saw in Section 2 when retaining the original weights; instead, we only observe this to be the case at the highest sparsities, those beyond which networks could reach full accuracy even with the trained weights. As we can see that the original blue curves in Figure 3 (left) move closer to the randomly reinitialized orange curves, while the orange curves generally remain consistent across experiments (Figure 3, right).

5 Discussion and Conclusions

Our experiments explore and exploit the balance that block sparsity can strike between unstructured pruning of individual weights and highly structured pruning of neurons and channels. Increasing block size results in a gradual decrease in accuracy, rather than a sharp drop when pruning neurons or channels. At the same time, there is no tradeoff-free use of block sparsity: it is unable to maintain accuracy at the same percentage pruned as unstructured pruning for all but the smallest of blocks. As a result, the viability of block-sparse pruning involves finding a balance between computational and memory needs and accuracy depending on priorities.

We are also motivated to understand why random reinitialization seems to not affect accuracy for neuron and channel pruning yet decreases accuracy in unstructured pruning. Across block sizes, random reinitialization performs worse than using the trained weights, although this gap is more pronounced in smaller blocks than larger blocks. Accuracy from random reinitialization is consistent across granularities except at the most extreme sparsities, potentially signifying a lower bound threshold characteristic of suboptimally pruned networks. Our data implies that the reason behind similar accuracies found by Liu et al. (2019) in structured settings may be that neuron and channel pruning reduce the trained accuracy of pruned subnetworks to the extent that it is comparable to random reinitialization.

REFERENCES

Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2015.

Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. Fast sparse convnets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14629–14638, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural network. *International Conference on Learning Representations*, 2019.

Scott Gray, Alec Radford, and Diederik P. Kingma. GPU kernels for block-sparse weights. *arXiv:1711.09224*, 2017.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset. `http://www.cs.toronto.edu/~kriz/cifar.html`. Dataset release.

François Lagunas. Sparse neural networks (2/n): Understanding GPU performance. `https://medium.com/huggingface/sparse-neural-networks-2-n-gpu-performance-b8bc9ce950fc`, May 2020a. Affiliated with Hugging Face.

François Lagunas. Block sparse matrices for smaller and faster language models. `https://huggingface.co/blog/pytorch_block_sparse`, September 2020b. Hugging Face code release.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal Brain Damage. *Advances in Neural Information Processing Systems*, 1990.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *International Conference on Learning Representations*, 2017.

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

Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J. Dally. Exploring the regularity of sparse structure in convolutional neural networks. *arXiv:1705.08922*, 2017.

Sharan Narang, Eric Undersander, and Gregory Diamos. Block-sparse recurrent neural networks. *arXiv:1711.02782*, 2017.

Jeff Pool. Accelerating sparsity in the NVIDIA Ampere Architecture. *NVIDIA GPU Technology Conference*, 2020. Presentation slides at the conference.

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

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in Neural Information Processing Systems*, 2016.

Chen Yang, Zhenghong Yang, Abdul Mateen Khattak, Liu Yang, Wenxin Zhang, Wanlin Gao, and Minjuan Wang. Structured pruning of convolutional neural networks via L1 regularization. *IEEE*, 2019.