Parameter-Efficient Approximation by Exploitation of Sparsity

Isaac Liao
EECS
Massachusetts Institute of Technology
Cambridge, MA 02139
iliao@mit.edu

Abstract

We propose a novel neural network architecture which is highly expressive, in that it is capable of approximating a huge class of sparse neural networks of a size smaller by a logarithmic factor, regardless of the arrangement of the sparse weights. In many cases, uniform approximation bounds for dense neural networks from n inputs to n outputs are derived through the construction of a sparse subnetwork, thus taking at least n^2 parameters in the original dense network despite that sometimes the sparse subnetwork only needs $O(n)$ operations. Our architecture can then reproduce this sparse subnetwork and replicate this uniform approximation bound for itself, but instead using only $O(n \log n)$ parameters, an improvement upon dense networks. Our architecture's improved parameter efficiency could then drastically reduce the memory requirements of neural networks with large hidden dimension, allowing us to respond by scaling up the size of the model (ie. the hidden dimension and the number of nonlinearities) by a factor of $O(n/\log n)$, to reuse the freed memory in a more expressive way.

1 Introduction

Machine learning researchers have been interested in sparsity for various reasons, including the simplification of models (Grünwald, 2007), to provide regularization (Bartoldson et al., 2020) or robustness (Cosentino et al., 2019), and to reduce computational costs (Hoefler et al., 2021); our research aims towards the last of these, more specifically the reduction of memory. Modern machine learning tends to operate in the regime of huge models, requiring vast overparameterization to achieve state-of-the-art results. As a consequence, machine learning has become extremely memory-hungry from the sheer quantity of parameters needed to be stored in the model. Lowering the number of parameters required to achieve the same performance would therefore allow machine learning to become more accessible to those with restricted computational resources, and would also allow those with more resources to scale to larger problems.

Recently, there has been increased interest in sparsity of machine learning models, as it has been shown that most of the weights in modern machine learning models can be removed without ill effect on performance, thus drastically reducing the size of models as measured by parameter count. Frankle and Carbin (2018) proposes that there are sparse subnetworks within large dense networks which achieve similar performance to their dense counterparts, and shows this to be true by providing an algorithm for pruning the weights of dense networks to produce performant sparse networks. In essence, this is an assertion that the most parameter-efficient architectures for neural networks tend to be sparse ones, and a proposition for an algorithm for finding such an architecture.

Since pruning methods like these begin with many weights and remove most of them away before training, the memory requirements of the pruning process are similar to that of the original unpruned

Preprint. Under review.

network. Therefore, while the pruned model can be trained with much less memory, the memory required for the construction of the neural network as a whole remains roughly unchanged when using pruning methods, calling into question the efficacy of pruning methods for this particular purpose. To alleviate this, we propose an alternative perspective on the construction of sparse networks: namely that diverse assortments of sparse neural network structures need not be achieved by cutting down huge dense structures, but can instead be configured from within a fixed structure of not much larger size via learning. The key takeaway is that the known efficiency of gradient descent in searching through a combinatorially large space of possible neural network weights can also be leveraged to search through this combinatorially large assortment of sparse structures.

2 Related Work

Historically, statistical learning theory has often taken the view that real-world data is generated by completely unknown processes, such that there was less motivation to study sparse computations as opposed to dense computation structures. More recently, there has been an increasing amount of interest in the assumption that real-world data is generated by sparse computations, making the discovery of these sparse structures a central goal of sparsity research. This assumption arises from the observation that anything that can be practically computed is practically computable with a sparse computation, whereas this is not necessarily true for computations structured in a dense fashion. Therefore, the assumption that we may choose to seek a sparse computation rarely undermines a problem's tractability.

Much research has therefore been devoted to the creation of sparse neural networks, and numerous approaches have been proposed, not limited to the following:

1. Frankle and Carbin (2018) state the well-known lottery ticket hypothesis: that when randomly initializing and training a dense neural network, there exists some sparse subnetwork which can be trained in the same manner with the same initialization to produce better performance. They show that a such a sparse neural network can be found by pruning low-magnitude weights from a dense neural network.

2. Another line of work aims to grow and remove weights and/or neurons dynamically during training of a sparse neural network (Hoefler et al., 2021).

3. Lastly, yet another line of work proposes to decompose dense matrices present in neural networks into smaller operations, such as singular value decompositions (Sainath et al., 2013), tensor train decompositions (Zhao et al., 2019), and/or randomized sparse decompositions (Liao et al., 2022).

Our work takes inspiration and technology from the third approach, the learning of matrix decompositions, to achieve the goal of the first approach, which is to assume that there is some sparse computation which works well and to try to find it. Our work will heavily depend on the past work of Liao et al. (2022), which provides a sparse linear neural network architecture which we will draw heavily from and use in our experiments, along with some theoretical work associated with this architecture, its initialization, and its expressive power.

3 Architecture

We propose to extend the sparse neural network architecture defined in (Liao et al., 2022), which offers to replace $n \times n$ dense matrices with $O(\log n)$ or $O(\log^2 n)$ sparse matrices, of $O(n)$ weights each, with the idea that the sparse computation graph formed allows for matrix multiplication with $O(n \log n)$ or $O(n \log^2 n)$ memory and time complexity rather than $O(n^2)$ like for dense matrices. Their architecture allows them to build much larger linear operations than what dense matrices allow for, in situations where the number of parameters is constrained. Section 3.1 details the construction of their sparse decomposition, and Section 3.2 explains how we may insert biases and nonlinearities to turn these matrix decompositions into neural networks with exceedingly large input and output dimensions.

3.1 Linear Sparsity

Liao et al. (2022) defines a linear operator $\tilde{G}(\theta) \in \mathbb{R}^{n \times n}$ parameterized by θ as follows:

$$\tilde{G}(\theta) = \prod_{i=1}^{N} B(\theta^{(i)}) P_i \tag{1}$$

where $B(\theta^{(i)})$ are block diagonal matrices with each block k by k for $k \in \mathbb{N}$ and filled with parameters in $\theta^{(i)}$, P_i are randomly chosen permutations, and $N \in \mathbb{N}$ is the depth. For our work, we choose $k = 2$. This architecture can be extended to the case where the number of weights n per layer is larger than the number of input and output neurons, simply by adding any required input neurons with zero activation before applying $\tilde{G}(\theta)$, and removing extra output neurons after applying $\tilde{G}(\theta)$.

Like (Liao et al., 2022), we initialize each block matrix to a random orthogonal matrix, such that $\tilde{G}(\theta)$ is initialized to an orthogonal matrix as well. This initially removes issues relating to gradient magnitudes for learning. We encourage the layers of the network to retain any unused information for later, ie. to maintain full-rankness, through regularization by penalizing the block matrices' deviation from orthogonal matrices. The regularization term for any one 2 by 2 block matrix B is the following quartic:

$$2||I - B^T B||_F^2 \tag{2}$$

which is the mean squared error of the squared singular values of B from 1. The global regularization term is then computed by taking the mean of this expression over all the blocks B in all the $B(\theta^{(i)})$.

3.2 Nonlinear Sparsity

Modifying the compositionally sparse linear architecture of $\tilde{G}(\theta)$ from Section 3.1, we may insert biases and nonlinearities after every block diagonal matrix to construct compositionally sparse nonlinear architectures as well. More specifically, we may create networks as follows:

$$h_\theta^{(0)}(x) = x \tag{3}$$

$$h_\theta^{(i)}(x) = \tanh(b_i + B(\theta^{(i)}) P_i h_\theta^{(i-1)}(x)) \tag{4}$$

$$h_\theta(x) = b_N + B(\theta^{(N)}) P_N h_\theta^{(N-1)}(x) \tag{5}$$

where b_i are learned biases, which may be initialized to zero. Note that at initialization, the Jacobian of h_θ at the origin is simply $\tilde{G}(\theta)$, which itself is an orthogonal matrix, counteracting issues relating to gradient magnitudes for learning.

4 Theoretical Properties

4.1 Linear Representability

Let F_{linear} denote the set of functions which can be computed by sparse linear neural networks of depth $O(N/\log n)$, at most n weights per layer, and indegree/outdegree at most 2. Liao et al. (2022) shows that with probability $1 - (n!)^{O(1)}$ over random selection of P_i, $\tilde{G}(\theta)$ can compute any function in F_{linear} given the right choice of θ. Because of this result, any demonstration that $\tilde{G}(\theta)$ can compute a linear operations merely requires showing that there exists a sparse linear neural network in F_{linear} which computes it. In this manner, we deduce by construction of networks in F_{linear} that $\tilde{G}(\theta)$ with n double the number of input and output nodes can represent matrix multiplication by:

- any rank k matrix with $O(n \log n(k + \log n))$ parameters, where the space of rank k matrices has dimension kn;
- any permutation matrix with $O(n \log n)$ parameters, where this space contains $n! \in e^{\Theta(n \log n)}$ matrices;
- any sparse matrix with at most k weights per row and column, including convolutions (without weight sharing) with $O(kn \log n)$ parameters;

- the discrete Fourier transform matrix with $O(n \log^2 n)$ parameters (we must separate real and complex components and have double the number of neurons); and
- any dense matrix with $O(n^2 \log n)$ parameters.

I do not detail the constructions here, because they are uninteresting, and to save space. From these examples, we see that oftentimes the number of parameters required is only a logarithmic factor larger than the time complexity of the best known algorithm to compute the solution.

4.2 Nonlinear Representability

Let F_{tanh} denote the set of functions $f_{\text{tanh}} : \mathbb{R}^i \to \mathbb{R}^o$ which can be computed by sparse tanh networks of depth $O(N/\log n)$, at most n weights per layer, and indegree/outdegree at most 2. Similarly to Section 4.1, we seek to show in this section that with probability $1 - (n!)^{O(1)}$ over random selection of P_i, h_θ is capable of arbitrarily accurately mimicking any function in F_{tanh} under choice of θ, thus motivating that h_θ is highly expressive and parameter efficient. Since this result shows h_θ to imitate other compositionally sparse tanh networks, then we may convert any constructive proofs for approximation bounds for sparse tanh networks into similar approximation bounds for the h_θ network. This makes the h_θ network "universally capable of approximation" in a certain sense. More specifically, let there be some set F of ground truth continuous functions $f : \mathbb{R}^i \to \mathbb{R}^o$ to approximate, and let us have a continuous loss function $\ell : \mathbb{R}^o \times \mathbb{R}^o \to \mathbb{R}$. Take any scheme for constructively proving a uniform approximation bound over some compact subset $S \subset \mathbb{R}^i$ of the input space, in the form of a function $C : F \to F_{\text{tanh}}$ that gives the bound

$$\inf_{f_{\text{tanh}} \in F_{\text{tanh}}} \sup_{x \in S} \ell(f(x), f_{\text{tanh}}(x)) \leq \sup_{x \in S} \ell(f(x), C(f)(x)) \tag{6}$$

Then, we must also have a corresponding bound for h_θ:

$$\inf_{\theta} \sup_{x \in S} \ell(f(x), h_\theta(x)) \leq \sup_{x \in S} \ell(f(x), C(f)(x)) \tag{7}$$

This is because we may construct a function $h_{\theta_\lambda} : \mathbb{R}^i \to \mathbb{R}^o$, where the weights θ_λ are parameterized by $\lambda \in \mathbb{R}$, such that h_{θ_λ} uniformly converges to the desired function $f_{\text{tanh}} \in F_{\text{tanh}}$ when $\lambda \to \infty$. We may show this through the following.

Let us have some desired function $f_{\text{tanh}} \in F_{\text{tanh}}$, which must represent some sparse tanh network of depth $d \in O(N/\log n)$, at most n weights per layer, and indegree/outdegree at most 2. Then, separate this tanh network into its d sparse affine layers with at most n weights each. Our strategy will be to partition the N layers of the h_θ network into d subnetworks of N/d layers each, and have each subnetwork arbitrarily accurately approximate one corresponding linear operation A_i of the f_{tanh} network with high probability (the offset of the affine layer can be accounted for since every layer j of h_θ contains offsets b_j). (Liao et al., 2022) shows that if there were no activations nor biases, this can be done for every layer with probability $1 - (n!)^{O(1)}$ under random sampling of P_i, ie. that there exists some θ such that $A_1 = B(\theta^{(1)})P_1 B(\theta^{(2)})P_2 \dots B(\theta^{(N/d)})P_{N/d}$. We may then add the biases and activations back and use this solution θ to recreate the linear operation of A_1 to arbitrary accuracy over any compact set. Firstly, we set the biases to zero. Then, note that the Jacobian of this subnetwork at the origin is already A_1, and thus the subnetwork represents the linear operation of A_1 to arbitrary accuracy over neighborhoods surrounding the origin. Using this fact, we may scale down the first learned linear operation $B(\theta^{(N/d)})$ by a chosen parameter λ and simultaneously scale up the last learned operation $B(\theta^{(1)})$ by λ, to keep the Jacobian the same but extend the domain where the linear approximation near the origin is accurate. By taking $\lambda \to \infty$, we then achieve arbitrarily small uniform approximation error between A_i and the subnetwork over any compact set of inputs. Finally, since the tanh activations connecting between sunetworks are also uniformly continuous, the entirity of h_θ then uniformly converges to f_{tanh} over any compact set of inputs, upon taking $\lambda \to \infty$.

We have thus shown that every function $f_{\text{tanh}} \in F_{\text{tanh}}$ is a limit point of the set of possible h_θ. Finally, by continuity of ℓ, we have the desired bound

$$\inf_{\theta} \sup_{x \in S} \ell(f(x), h_\theta(x)) \leq \inf_{f_{\text{tanh}} \in F_{\text{tanh}}} \sup_{x \in S} \ell(f(x), f_{\text{tanh}}(x)) \leq \sup_{x \in S} \ell(f(x), C(f)(x)) \tag{8}$$

all of which only applies with probability $1 - (n!)^{O(1)}$ under random sampling of P_i.

This finally proves the desired universal representability result: given a target function F that can be uniformly approximated over a compact set by some unknown neural network of depth $d \in O(N/\log n)$, at most n weights per layer, and indegree/outdegree at most 2, the target function can then also be uniformly approximated by h_θ too, which is not much more expensive as the unknown neural network, merely being $O(\log n)$ times deeper.

In the same manner as in Section 3.1, with this result we may deduce that $h_\theta(x)$, with every layer size $O(1)$ times larger than the number of input and output nodes, can represent given the correct θ:

- n-way multiplexers with $O(n \log^2 n)$ parameters;
- n-bit binary adders with $O(n \log^2 n)$ parameters (via the Kogge-Stone adder); and
- n-bit bitshift operators with $O(n \log^2 n)$ parameters,

5 Experiments

We perform several experiments to show that the architecture of $\tilde{G}(\theta)$ from Section 3.1 is indeed capable of performing computations like those in Section 4.1 in a parameter-efficient manner. We do not use our architecture from Section 4.2 to approach nonlinear target functions from Section 3.2 because we could not learn any of them even with vanilla dense networks of much larger parameter count; they are too difficult to learn and the loss would never decrease. We will thus only describe our experiments with linear computations.

For each experiment, we define a ground truth square matrix $A \in \mathbb{R}^{n/2 \times n/2}$ and we optimize

$$\arg\min_\theta \mathbb{E}_{x \sim \mathcal{N}(0,I)} \left[\frac{2}{n} ||\tilde{G}(\theta)x - Ax||^2 \right] \tag{9}$$

via online stochastic gradient descent. We generate the data online by sampling batches of x from $\mathcal{N}(0, I)$ and computing the output Ax using the ground truth A at every step.

We use $n = 2048$, with input and output dimensions of $n/2 = 1024$ for all experiments. We will take the number of parameters in θ to be less than $n^2/4 \approx 1.0$M to show that we need fewer parameters than densely parameterized operations to be able to perform the same computations. This necessarily means that the VC dimension of $\tilde{G}(\theta)$ will be far lower than $n^2/4$, so we should expect to perform poorly if A is an arbitrary dense matrix. For all of these experiments, we double the number of neurons from $n/2$ to n at the input by appending zeroes to the input vector, and slice away the last $n/2$ neurons for the output.

We condition A in four ways such that Ax can be "computed efficiently" for arbitrary x, ie. that $A \in F_{\text{linear}}$, and we show that $\tilde{G}(\theta)$ can perform these computations despite its limited number of parameters and its inability to perform dense matrix multiplication in general. For case, we will define a conditioning difficulty parameter k which we may adjust from 1 to 5 to observe the effect on performance. We vary both the difficulty k and the $\tilde{G}(\theta)$ architecture depth N, and train each $\tilde{G}(\theta)$ for 7000 steps with Adam of learning rate 0.01 and regularization parameter 3, and then decrease the regularization parameter to 0.3 for another 3000 steps, and finally decrease the learning rate to 0.001 for another 2000 steps, and measure the final mean squared error achieved over the last 1000 steps.

5.1 Low Rank Matrices

We attempt to learn matrix multiplication by A of low rank k. We sample A randomly by multiplying together two random unit normal matrices $U, V \in \mathbb{R}^{n/2 \times k}$ as $A = UV^T/\sqrt{nk/2}$. By varying the rank k from 1 to 5 inclusive, we vary the number of degrees of freedom (dof) of A from 2047 to 10235. In Figure 1, we show that by varying N and thus the number of parameters in $\tilde{G}(\theta)$, we observe that it takes about 25k parameters for $\tilde{G}(\theta)$ to learn the solution, where this threshold increases with the difficulty k. Note that the receptive field of any output neuron can only reach the entire set of input neurons after a depth of 10, corresponding to $\approx$ 20k parameters, and thus in theory any such threshold must be above 20k, and in practice the threshold is not much higher than that at all. This signals that $\tilde{G}(\theta)$ is good at making efficient use of its depth, because it can learn the solution almost as soon as it exceeds the necessary depth.

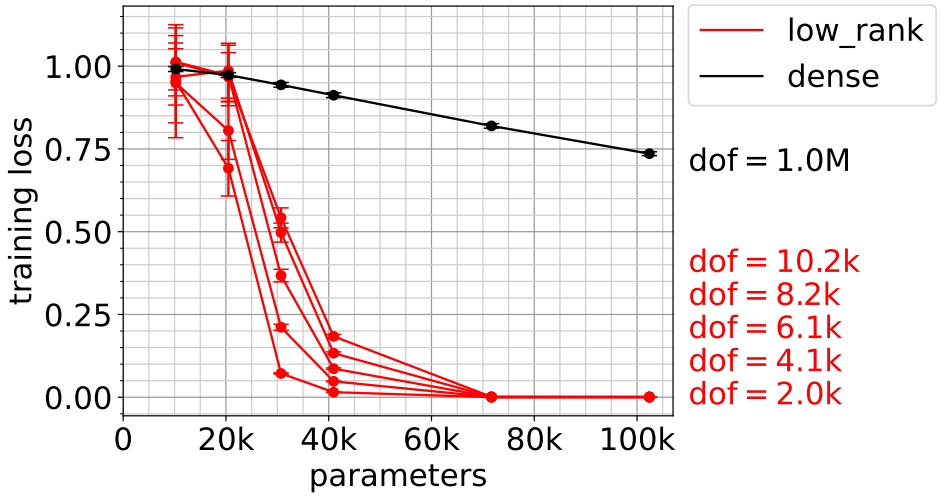

Figure 1: Ability of $\tilde{G}(\theta)$ with varying numbers of parameters to learn multiplication by random rank-k matrices where k varies from 1 to 5. Each red line represents one value of k, and is labeled with the corresponding number of degrees of freedom (dof) in the rank-k matrix. We observe a thresholding effect where $\tilde{G}(\theta)$ needs $\approx$ 25k parameters to find the solution. Notably, this threshold is higher than the number of degrees of freedom in the solution, but is still much lower than the number of degrees of freedom of a generic dense matrix. The black line shows the ability of $\tilde{G}(\theta)$ to learn multiplication by an iid random normal dense matrix with variance $1/\sqrt{n/2}$, for comparison.

5.2 Band Matrices/Convolutions

We attempt to learn one dimensional convolutions with receptive field size k, without parameter sharing. We sample the weights of the convolution from normal distributions of standard deviation $1/\sqrt{k}$. We vary k, and thus also the number of degrees of freedom of A from 1024 to 5120. In Figure 2, we again show by varying N and thus the number of parameters in $\tilde{G}(\theta)$, that it takes about 25k parameters for $\tilde{G}(\theta)$ to learn the solution, where this threshold increases with k. Interestingly, in this case we also observe a performance drop when excessively many parameters are given. We hypothesize that this is because in the infinite depth limit, the random orthogonal initializations of the block matrices cause the learning behavior of the $\tilde{G}(\theta)$ network to behave in a manner similar to if $\tilde{G}(\theta)$ were parameterized as a dense matrix. This would remove any biases of $\tilde{G}(\theta)$ towards learning solutions that are more compositionally sparse.

5.3 Sparse Matrices

We attempt to learn multiplication by random sparse matrices of $kn/2$ nonzero weights sampled from normal distributions of variance $1/\sqrt{k}$. The positions of the nonzero weights within the A matrix are uniformly randomly chosen. We sample the weights of the convolution from normal distributions. The rest of this experiment proceeds similarly to the experiment of Section 5.2, and Figure 3 shows the results.

5.4 Permutation Matrices

We attempt to learn to apply a randomly chosen permutation to the vector x. There is no difficulty parameter k for this task. The rest of this experiment proceeds similarly to the experiment of Section 5.2, and Figure 4 shows the results.

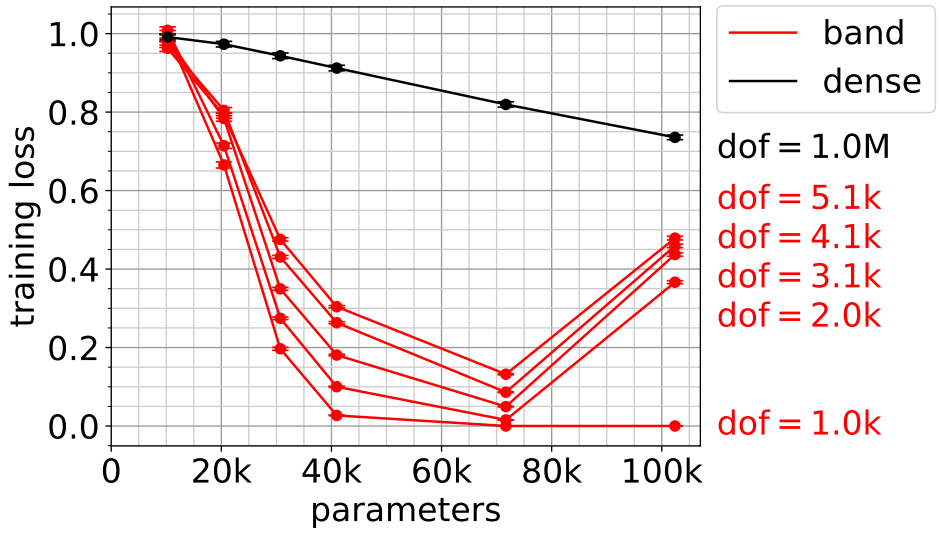

Figure 2: Ability of $\tilde{G}(\theta)$ with varying numbers of parameters to learn multiplication by band matrices of k bands. Other details are identical to those in Figure 1. In this plot, we also observe that if $\tilde{G}(\theta)$ is given too many parameters, it begins to lose its ability to learn the band matrices, as evidenced by the upward slopes towards the right side of the plot.

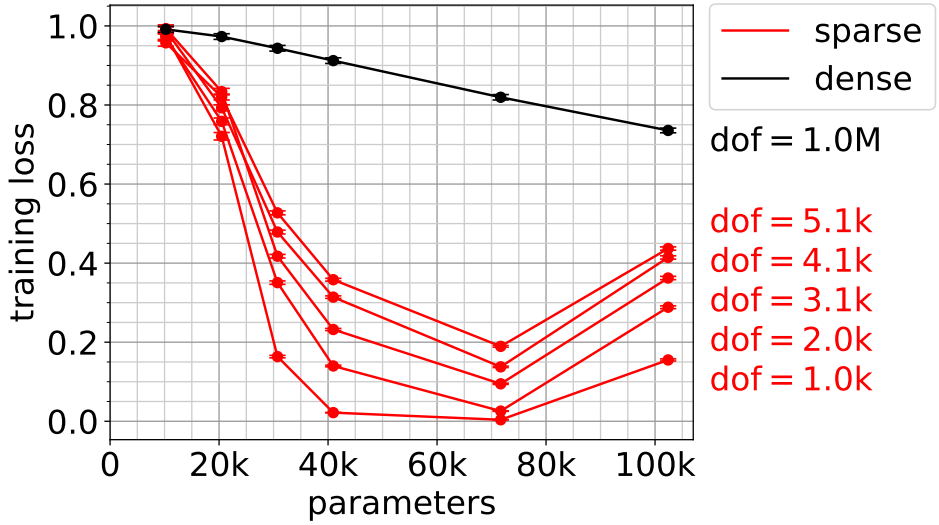

Figure 3: Ability of $\tilde{G}(\theta)$ with varying numbers of parameters to learn multiplication by random sparse matrices of $kn/2$ nonzero weights. Other details are identical to those in Figure 2.

6 Discussion

We have shown in Section 4.2 that the $\tilde{G}(\theta)$ linear neural network architecture defined originally in (Liao et al., 2022) can be extended to create a nonlinear architecture with similar theoretical properties. Namely, we define an architecture h_θ which can approximate to arbitrary accuracy any sparse tanh network out of a large class F_{tanh}, shallower by a logarithmic factor in hidden dimension. This allows us to transform many uniform approximation bounds for tanh networks contingent on the construction of an unknown sparse tanh subnetwork, into uniform approximation bounds for h_θ, without having to figure out the unknown structure of the sparse tanh subnetwork.

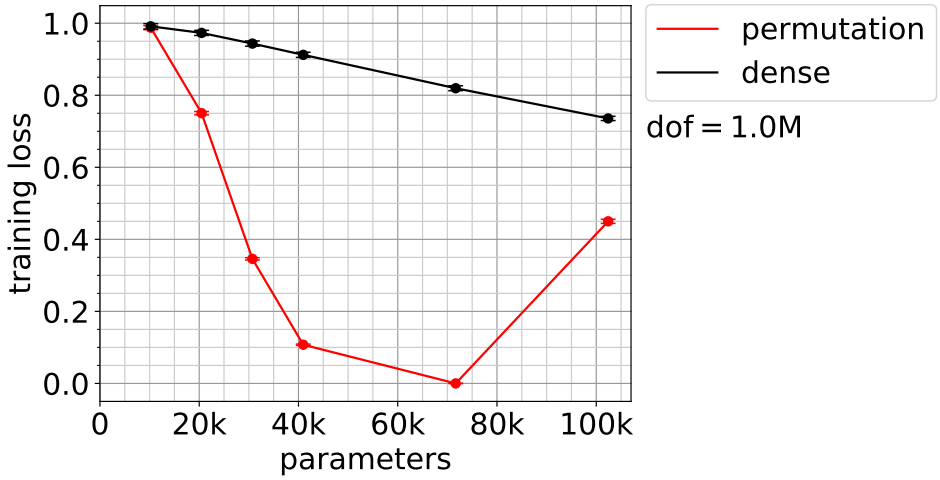

Figure 4: Ability of $\tilde{G}(\theta)$ with varying numbers of parameters to learn to to apply a permutation to the input. Other details are identical to those in Figure 2. The number of degrees of freedom in a permutation is not shown because it is zero, since the set of permutation matrices is discrete.

We have also shown through the experiments of Section 5 that the original $\tilde{G}(\theta)$ network is practically capable of learning a diverse range of the linear operations in F_{linear} despite having relatively few parameters, which was supported only theoretically in (Liao et al., 2022). These linear operations were sometimes extremely unrelated to one another (eg. sparse matrices and low-rank matrices), barring that they were all compositionally sparse in some way, and the single $\tilde{G}(\theta)$ architecture was still capable of learning each of them. We observed that after a certain threshold in the number of parameters, $\tilde{G}(\theta)$ was able to learn the linear operations to a much higher precision, and that this threshold is crossed shortly after $\tilde{G}(\theta)$ gains the necessary depth to reproduce the unknown sparse ground truth computation representing the linear operation. We have also discovered that when $\tilde{G}(\theta)$ is given exceedingly many parameters performance worsens, creating a "goldilocks" zone of network depths, and we present a hypothesis for why this happens. We leave further exploration of this phenomenon for future work.

7 Conclusion

Altogether, our theoretical extension and experimental testing of the $\tilde{G}(\theta)$ network in (Liao et al., 2022) motivates that the optimization of randomized sparse neural networks and matrix decompositions is a remarkably effective technique for the parameter-efficient discovery of sparse computation structures. In situations where the parameter count is a bottleneck, such techniques may be used to reduce the parameter count while maintaining much of the expressive power of a parameter-hungry dense neural network. In cases where memory is the issue, the memory freed by these techniques could allow us to upscale neural networks to problems of much higher dimension than ever before.

References

B. Bartoldson, A. Morcos, A. Barbu, and G. Erlebacher. The generalization-stability tradeoff in neural network pruning. *Advances in Neural Information Processing Systems*, 33:20852–20864, 2020.

J. Cosentino, F. Zaiter, D. Pei, and J. Zhu. The search for sparse, robust neural networks. *arXiv preprint arXiv:1912.02386*, 2019.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

P. Grünwald. *The Minimum Description Length Principle*. 01 2007. ISBN 9780262256292. doi: 10.7551/mitpress/4643.001.0001.

T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(241): 1–124, 2021.

I. Liao, R. R. Dangovski, J. N. Foerster, and M. Soljačić. Learning to optimize quasi-newton methods. *arXiv preprint arXiv:2210.06171*, 2022.

T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6655–6659, 2013. doi: 10.1109/ICASSP.2013.6638949.

Q. Zhao, M. Sugiyama, L. Yuan, and A. Cichocki. Learning efficient tensor representations with ring-structured networks. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 8608–8612. IEEE, 2019.