On the Sparsity of Deep Neural Networks in the Overparameterized Regime: An Empirical Study

Rahul Parhi [1] Jack H. Wolf [1] Robert D. Nowak [1]

Abstract

Sparsity and low-rank structures have been incorporated into neural networks to reduce computational complexity and to improve generalization and robustness. Recent theoretical developments show that both are natural characteristics of data-fitting solutions cast in a new family of Banach spaces referred to as $\mathscr{R}\,\mathrm{BV}^2$ spaces, the spaces of second-order bounded variation in the Radon domain. Moreover, sparse and deep ReLU networks are solutions to infinite dimensional variational problems in compositions of these spaces. This means that these learning problems can be recast as parametric optimizations over neural network weights. Remarkably, standard weight decay and variants correspond exactly to regularizing the $\mathscr{R}\,\mathrm{BV}^2$-norm in the function space. Empirical validation in this paper confirm that weight decay leads to sparse and low-rank networks, as predicted by the theory.

1. Introduction

One approach to studying overparameterization in shallow neural networks is the so-called "continuum-width" limit, in which a function is *synthesized* as a combination of continuously many neurons. Such functions can be expressed as an integral of a neural activation function against a finite (Radon) measure over the weights and biases of the neuron; see Barron (1993); Bach (2017) for further details and related work. It has been known for a long time that data fitting problems with this class of functions admit sparse solutions, with as many neurons as data points. This raises several questions: 1) What is the space of all functions that can be constructed in this manner? 2) What properties does this space have? 3) What happens if the learning problem is posed in this function space? 4) Is there an equivalent sparsity-promoting regularizer for training neural networks? Recently, we have answered these questions and its extensions to deep neural networks. This work, which we review in the next section, sets the stage for the empirical study presented in this paper.

Our theoretical work shows that finite-width neural networks are optimal solutions to data-fitting problems cast in certain function spaces. These are the spaces of second-order bounded variation in the Radon domain ($\mathscr{R}\,\mathrm{BV}^2$). The norms in these spaces are computed by measuring the smoothness (in terms of derivatives) in the Radon domain. It turns out that regularizing the $\mathscr{R}\,\mathrm{BV}^2$-norm of a shallow, single-output network is equivalent to the squared ℓ^2-norm regularization of the network weights. This implies that training an overparameterized shallow network using weight decay implements $\mathscr{R}\,\mathrm{BV}^2$-norm regularization. Experiments confirm that training overparameterized networks with weight decay yields sparse solutions where the number of neurons is roughly equal to the number of training data, regardless of the overparameterization level.

The theory also shows that finite-width deep neural networks are solutions data-fitting in a new class of function spaces defined by compositions of $\mathscr{R}\,\mathrm{BV}^2$ spaces. In addition to having sparse weights, the neural network solutions also have low-rank weight matrices between ReLU layers. In this setting, the $\mathscr{R}\,\mathrm{BV}^2$-norm of each layer can be expressed in terms of squared ℓ^1- and ℓ^2-weight norms, suggesting a potentially important modification to standard weight decay. Our experiments explore similarities and differences between training with standard weight decay and this modification. While our experiments are rather limited, we find that the modified weight decay results in sparser networks and appears to eliminate some redundancies in the solutions obtained using standard weight decay. We also explore the low-rank nature of neural networks trained with different regularizers and with architectures with and without explicit low-rank weight structures.

Sparsity and low-rank structures have been incorporated into neural networks to reduce computational complexity and to improve generalization and robustness, for example (LeCun et al., 1990; Hinton et al., 2012; Frankle & Carbin, 2018; Ba & Caruana, 2014; Golubeva et al., 2021; Huh et al., 2021; Littwin et al., 2021; Evci et al., 2020; Timpl et al.,

[1]University of Wisconsin–Madison, Madison, WI, USA. Correspondence to: Rahul Parhi <rahul@ece.wisc.edu>.

Presented at the ICML 2021 Workshop on Overparameterization: Pitfalls & Opportunities. Copyright 2021 by the author(s).

2021; Neyshabur, 2020; Wang et al., 2021) and see Hoefler et al. (2021) for a more comprehensive review. Our theory and experiments show that both sparsity and low-rank structure are natural consequences of learning in $\mathscr{R}\,\mathrm{BV}^2$ function spaces and that weight decay results in sparse solutions. Special sparsity-promoting regularization techniques and/or pruning are unnecessary to learn sparse solutions. Remarkably, the sparsifying properties of standard weight decay appear to be overlooked in the literature, even though the connection between weight decay and ℓ^1-regularization has been known for years (Grandvalet, 1998).

2. Deep ReLU Network Representer Theorem

Many past works have considered training neural networks with arbitrarily many neurons and sparsity-promoting regularization methods; for example, see (Bengio et al., 2006; Rosset et al., 2007; Bach, 2017). An important question not fully answered in such past work is characterizing the space of all functions that can be represented in such a way. The answer was first obtained for shallow, univariate ReLU networks in (Savarese et al., 2019; Parhi & Nowak, 2020) and later for shallow, multivariate networks (Ongie et al., 2020; Parhi & Nowak, 2021b). Finally, our recent work developed a complete characterization for deep ReLU networks (Parhi & Nowak, 2021a). We briefly review this theory and its implications that will be examined in our experiments.

The function spaces naturally associated with ReLU networks are characterized by measuring smoothness in the Radon domain. The Radon transform of $f : \mathbb{R}^d \to \mathbb{R}$ is

$$\mathscr{R}\{f\}(\boldsymbol{\gamma}, t) = \int_{\mathbb{R}^d} f(\boldsymbol{x})\delta(\boldsymbol{\gamma}^\mathsf{T}\boldsymbol{x} - t)\,\mathrm{d}\boldsymbol{x},\ (\boldsymbol{\gamma}, t) \in \mathbb{S}^{d-1} \times \mathbb{R}$$

where δ is the univariate Dirac impulse and $\mathbb{S}^{d-1}$ denotes the surface of the Euclidean sphere in $\mathbb{R}^d$. This transform extracts information about f along all hyperplanes. For each $\boldsymbol{\gamma} \in \mathbb{S}^{d-1}$, we have a univariate function in t and the smoothness of that function is measured in terms of its second-order total variation, which is essentially the L^1-norm of its second derivative (in a distributional sense to allow for Dirac impulses). For example, applying these steps to a single ReLU neuron produces a Dirac impulse with magnitude equal to the ℓ^2-norm of the vector of weights into the neuron, which characterizes its slope/smoothness.

More precisely, the space is defined via the operator

$$\mathscr{R}\,\mathrm{TV}^2(f) = c_d \|\partial_t^2 \Lambda^{d-1} \mathscr{R} f\|_{\mathcal{M}(\mathbb{S}^{d-1}\times\mathbb{R})} \qquad (1)$$

where c_d is a constant depending on the dimension d of the domain of the function f, ∂_t^2 denotes the second order differentiation, $\Lambda^{d-1}\mathscr{R}$ is the filtered Radon transform, and the $\mathcal{M}$-norm denotes the total variation norm (in the sense of measures) and can be viewed as a generalization of the L^1-norm to allow for distributions such as the Dirac impulse. We refer the reader to Parhi & Nowak (2021b, Section 3) for more details. It was shown in Parhi & Nowak (2021a;b) that $\mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^d)$, the space of functions with finite $\mathscr{R}\,\mathrm{TV}^2(\cdot)$, is a Banach space.

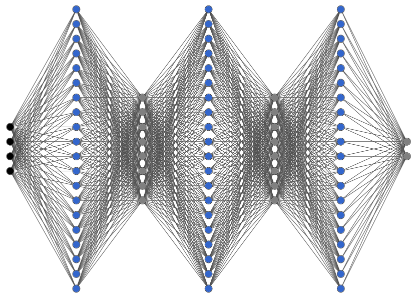


Figure 1. Architecture of the deep neural network in Theorem 1 in the case of $L = 3$ hidden layers. The black nodes denote input nodes, the blue nodes denote ReLU nodes, and the gray nodes denote linear nodes. Bias nodes and skip connection nodes are omitted for clarity.

We can define the vector-valued analogue of $\mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^d)$ as the D-fold Cartesian product

$$\mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^d; \mathbb{R}^D) := \mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^d) \times \cdots \times \mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^d).$$

This is a Banach space when equipped with the norm

$$\|f\|_{\mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^d;\mathbb{R}^D)} := \sum_{m=1}^{D} \|f_m\|_{\mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^d)},$$

where $f = (f_1, \dots, f_D)$. Lastly, we can then define a "deep" or compositional analogue of $\mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^d; \mathbb{R}^D)$ as $\mathscr{R}\,\mathrm{BV}^2_{\mathsf{deep}}(L)$

$$:= \left\{ f = f^{(L)} \circ \cdots \circ f^{(1)} : \begin{array}{l} f^{(\ell)} \in \mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^{d_{\ell-1}}; \mathbb{R}^{d_\ell}), \\ \ell = 1, \dots, L \end{array} \right\}.$$

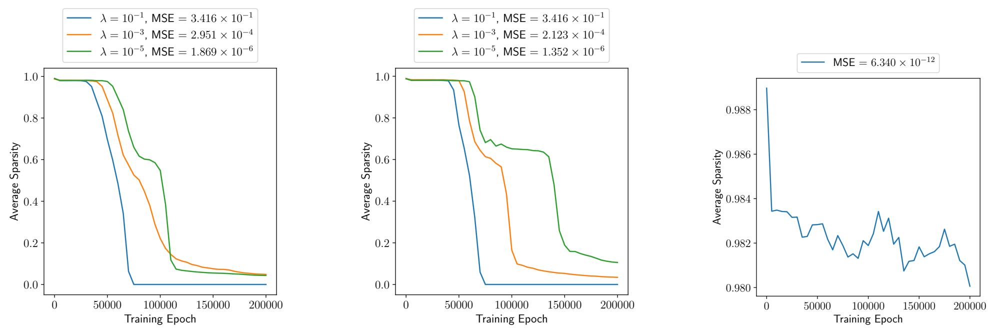
This definition reflects two standard architectural specifications for deep neural networks: the number of hidden layers, L, and the functional "widths", d_ℓ, of each layer. That is, each function in the composition will ultimately correspond to a layer in a deep neural network in the representer theorem for deep ReLU networks. We now state the deep ReLU network representer theorem of Parhi & Nowak (2021a).

Theorem 1 (see Parhi & Nowak (2021a, Theorem 3.2)) *Consider the problem of approximating the scattered data* $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N \subset \mathbb{R}^{d_0} \times \mathbb{R}^{d_L}$. *Let* $\ell(\cdot,\cdot)$ *be an arbitrary lower semi-continuous loss function and let* $\lambda > 0$ *be a regularization parameter. Then, there exists a solution to the variational problem*

$$\min_{f \in \mathscr{R}\,\mathrm{BV}^2_{\mathsf{deep}}(L)} \sum_{n=1}^{N} \ell(\boldsymbol{y}_n, f(\boldsymbol{x}_n)) + \lambda \sum_{\ell=1}^{L} \|f^{(\ell)}\|_{\mathscr{R}\,\mathrm{BV}^2(\mathbb{R}^{d_{\ell-1}};\mathbb{R}^{d_\ell})}$$

of the form $s(\boldsymbol{x}) = \boldsymbol{x}^{(L)}$, *where* $\boldsymbol{x}^{(L)}$ *is computed recursively via*

$$\begin{cases} \boldsymbol{x}^{(0)} := \boldsymbol{x}, \\ \boldsymbol{x}^{(\ell)} := \mathbf{V}^{(\ell)}\boldsymbol{\rho}(\mathbf{W}^{(\ell)}\boldsymbol{x}^{(\ell-1)} - \boldsymbol{b}^{(\ell)}) + \mathbf{C}^{(\ell)}\boldsymbol{x}^{(\ell-1)} + \boldsymbol{c}_0^{(\ell)}, \end{cases}$$



(a) Regularization as in (2) (b) Weight decay (c) No regularization

Figure 2. Average sparsity of weight matrices vs. training epoch.

where $\ell = 1, \dots, \ell$, $\boldsymbol{\rho}$ applies $\rho = \max\{0, \cdot\}$ component-wise, $\mathbf{V}^{(\ell)} \in \mathbb{R}^{d_\ell \times K^{(\ell)}}$, $\mathbf{W}^{(\ell)} \in \mathbb{R}^{K^{(\ell)} \times d_{\ell-1}}$, $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{K^{(\ell)}}$, $\mathbf{C}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$, and $\boldsymbol{c}_0^{(\ell)} \in \mathbb{R}^{d_\ell}$, where $K^{(\ell)} \leq N d_\ell$.

Theorem 1 says a solution to a variational problem over the infinite-dimensional space $\mathscr{R}\,\mathrm{BV}^2_{\mathsf{deep}}(L)$ is *exactly* a deep ReLU network with rank bounded weight matrices and skip connections, as seen in Figure 1. The low-rank structure arises naturally from the specification of the "functional width" d_ℓ of each layer, while the number of ReLU units per layer is not explicitly constrained. The important take-away message is that we do not assume a neural network structure for the solution, rather the representer theorem shows that neural networks are optimal solutions to data-fitting in $\mathscr{R}\,\mathrm{BV}^2_{\mathsf{deep}}(L)$. This is analogous to the result that kernel machines are solutions to data-fitting problems in reproducing kernel Hilbert spaces.

Parhi & Nowak (2021a) show that the variational problem in Theorem 1 can be recast as the following finite-dimensional neural network training problem (provided the widths of the architecture satisfy the requirements in Theorem 1):

$$\min_{\boldsymbol{\theta}=\{\boldsymbol{v}_k, \boldsymbol{w}_k, \boldsymbol{c}_j\}} \sum_{n=1}^{N} \ell(\boldsymbol{y}_n, f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)) + \lambda \left(\sum_j \|\boldsymbol{c}_j\|_1 + \sum_k \|\boldsymbol{v}_k\|_1^2 + \|\boldsymbol{w}_k\|_2^2 \right), \tag{2}$$

where $(\boldsymbol{v}_k, \boldsymbol{w}_k)$ are the weights into and out of the kth ReLU neuron (the sum is over all ReLUs in the network) and $\boldsymbol{c}_j$ are the weights out of the jth linear neuron (corresponding to skip connections, and the sum is over all such neurons).

3. Experiments

The previously discussed theory characterizes the global minimizers of regularized neural network training objective (2). Since the objective is non-convex (and thus finding global minimizers may be a challenge), it important to verify the implications of the theory in practice. The neural networks are implemented using PyTorch and trained using the Adam optimizer with a learning rate of 10^{-3}. The loss function is the mean squared error (MSE) loss. At the beginning of training, all parameters are initialized uniformly at random on $[-1, 1]$. In each layer we have $K^{(\ell)} = 1296$ and $d_\ell = 16$. The data set used for training is of size 81 with input values lying in the 4-dimensional grid $\{0, 0.5, 1\}^4$ and output values generated uniformly at random on $[-1, 1]$.

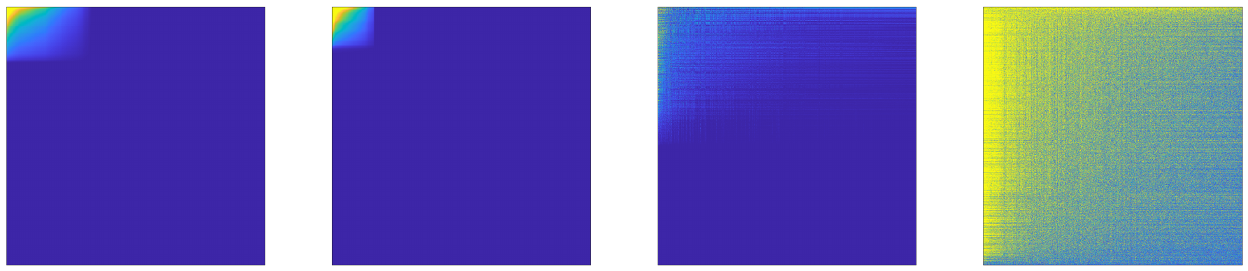
Sparsity of Weight Matrices During Training To investigate the sparsity of the weight matrices during training, we train deep ReLU networks with the architecture in Theorem 1 with $L = 2$ hidden layers and regularization parameter $\lambda \in \{10^{-1}, 10^{-3}, 10^{-5}\}$ for 200000 epochs. We compare the regularization from (2) with weight decay, i.e., squared ℓ^2-norm on all parameters, and no regularization. We compute the sparsity of matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ as

$$\mathsf{sparsity}(\mathbf{A}) := \frac{1}{mn} \sum_{k=1}^{m} \sum_{\ell=1}^{n} \mathbb{1}\{|a_{k,\ell}| > 10^{-3}\},$$

where $a_{k,\ell}$ is the (k,ℓ)th entry of $\mathbf{A}$ and $\mathbb{1}\{\cdot\}$ is 1 when the input is true and 0 otherwise. We chose the threshold of 10^{-3} since the maximum value in any matrix was typically between 0.1 and 0.2.

We plot the sparsity averaged across all weight matrices vs. the training epoch in Figure 2 of a typical run of this setup. We see that training with the regularization as in (2) as well as training with weight decay results in the sparsity weight matrices decreasing during training, though training with the regularization as in (2) results in sparser weight matrices. Meanwhile, training with no regularization results in non-sparse weight matrices. Also note the MSEs (data-fitting errors) reported above the plots. For reference, we remark that the mean squared value of the output values of the data set is 0.313, so all trained networks except those trained with $\lambda = 10^{-1}$ have essentially interpolated the data set.

Sparsity of Weight Matrices in Trained Networks To investigate the sparsity of the weight matrices, we train deep ReLU networks with the architecture in Theorem 1 with



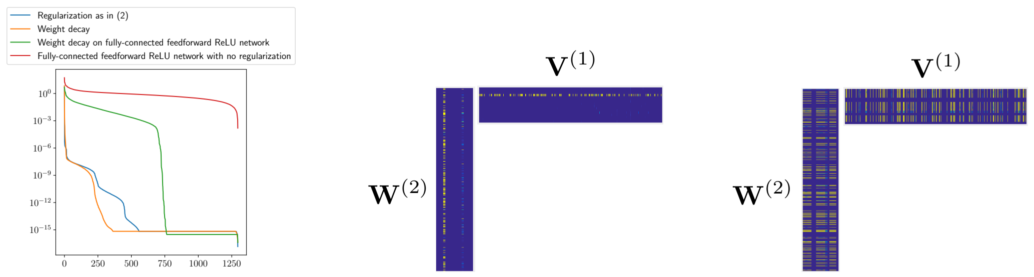
(a) $\mathbf{A}^{(2)} = \mathbf{W}^{(2)}\mathbf{V}^{(1)}$ after training with regularization as in (2).

(b) $\mathbf{A}^{(2)} = \mathbf{W}^{(2)}\mathbf{V}^{(1)}$ after training with weight decay.

(c) $\mathbf{A}^{(2)}$ of a standard fully-connected feedforward ReLU network trained with weight decay.

(d) $\mathbf{A}^{(2)}$ of a standard fully-connected feedforward ReLU network trained with no regularization.

Figure 3. Weight matrices between ReLU layers of trained networks.



(a) Singular values of weight matrices between ReLU layers of trained networks.

(b) Learned matrices from regularization as in (2).

(c) Learned matrices from weight decay.

Figure 4. Rank of weight matrices between ReLU layers of trained networks.

$L = 2$ hidden layers and regularization parameter $\lambda = 10^{-3}$ for 50000 epochs. We compare the regularization from (2) with weight decay. We also train a standard fully-connected feedforward ReLU network[1] with skip connections, with and without weight decay, for 50000 epochs.

We plot the weight matrices between ReLU layers of a typical run in Figure 3, sorted so that ℓ^2-norms of the rows and columns are decreasing (effectively re-ordering the ReLU neurons in each layer to help in visualizing the sparsity of the learned networks). The brighter colors correspond to larger magnitudes of the weights. We see that training with the regularization in (2), with weight decay with the architecture from Theorem 1, and with weight decay for a standard fully-connected feedforward ReLU network results in fairly sparse weight matrices. Meanwhile, training a standard fully-connected feedforward ReLU network without weight decay results in non-sparse weight matrices.

Rank of Weight Matrices in Trained Networks To investigate the rank of the weight matrices of a trained network, we consider the same setup discussed previously. In Figure 4(a), we plot the singular values of the weight matrix $\mathbf{A}^{(1)}$ between ReLU layers of a typical run. We see that with the regularization as in (2) and weight decay with the architecture from Theorem 1, the learned weight matrices are very low rank, with the matrix learned from the regularization in (2) being almost rank 1. Meanwhile, the weight matrix in the case of the fully-connected feedforward ReLU network with trained with weight decay is about rank 750 and trained without regularization is full-rank, so weight decay is, to a lesser extent, enforcing low-rank structure. In Figure 4(b) we plot the learned matrices with regularization as in (2). We see that the learned matrix $\mathbf{A}^{(2)} = \mathbf{W}^{(2)}\mathbf{V}^{(1)}$ is almost rank 1 since only one pattern is being learned. In Figure 4(c) we plot the learned matrices from training with weight decay and see that the learned matrix $\mathbf{A}^{(2)} = \mathbf{W}^{(2)}\mathbf{V}^{(1)}$ is low rank since the same pattern is learned multiple times.

4. Conclusion

We considered the training deep ReLU networks with new, principled regularization proposed by Parhi & Nowak (2021a). We found that the problem (2) resulted in networks with sparse and low-rank weight matrices. We compared this regularization to training the same architecture with weight decay and found, to a lesser extent, that the networks also exhibited sparse and low-rank weight matrices. We compared these results with a standard fully-connected feedforward ReLU network trained with and without weight decay. Training with weight decay still resulted in somewhat sparse and low-rank structure, while training without weight decay had dense and full-rank weight matrices.

[1] $\mathbf{A}^{(2)} \in \mathbb{R}^{1296\times1296}$ is *not* factorized as $\mathbf{A}^{(2)} = \mathbf{W}^{(2)}\mathbf{V}^{(1)}$.

References

Ba, L. J. and Caruana, R. Do deep nets really need to be deep? In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2654–2662, 2014.

Bach, F. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

Barron, A. R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

Bengio, Y., Le Roux, N., Vincent, P., Delalleau, O., and Marcotte, P. Convex neural networks. *Advances in neural information processing systems*, 18:123, 2006.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Golubeva, A., Neyshabur, B., and Gur-Ari, G. Are wider nets better given the same number of parameters? In *International Conference on Learning Representations*, 2021.

Grandvalet, Y. Least absolute shrinkage is equivalent to quadratic penalization. In *International Conference on Artificial Neural Networks*, pp. 201–206. Springer, 1998.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554*, 2021.

Huh, M., Mobahi, H., Zhang, R., Cheung, B., Agrawal, P., and Isola, P. The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*, 2021.

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

Littwin, E., Saremi, O., Zhai, S., Thilak, V., Goh, H., Susskind, J. M., and Yang, G. Implicit acceleration and feature learning ininfinitely wide neural networks with bottlenecks. *arXiv preprint arXiv:2107.00364*, 2021.

Neyshabur, B. Towards learning convolutions from scratch. *Advances in Neural Information Processing Systems*, 33, 2020.

Ongie, G., Willett, R., Soudry, D., and Srebro, N. A function space view of bounded norm infinite width ReLU nets: The multivariate case. In *International Conference on Learning Representations*, 2020.

Parhi, R. and Nowak, R. D. The role of neural network activation functions. *IEEE Signal Processing Letters*, 27: 1779–1783, 2020.

Parhi, R. and Nowak, R. D. What kinds of functions do deep neural networks learn? Insights from variational spline theory. *arXiv preprint arXiv:2105.03361*, 2021a.

Parhi, R. and Nowak, R. D. Banach space representer theorems for neural networks and ridge splines. *Journal of Machine Learning Research*, 22(43):1–40, 2021b.

Rosset, S., Swirszcz, G., Srebro, N., and Zhu, J. ℓ_1 regularization in infinite dimensional feature spaces. In *International Conference on Computational Learning Theory*, pp. 544–558. Springer, 2007.

Savarese, P., Evron, I., Soudry, D., and Srebro, N. How do infinite width bounded norm networks look in function space? In *Conference on Learning Theory*, pp. 2667–2690. PMLR, 2019.

Timpl, L., Entezari, R., Sedghi, H., Neyshabur, B., and Saukh, O. Understanding the effect of sparsity on neural networks robustness. In *ICML Workshop on Overparameterization: Pitfalls and Opportunities*, 2021.

Wang, H., Agarwal, S., and Papailiopoulos, D. Pufferfish: Communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems*, 3, 2021.