UNIVERSAL APPROXIMATION AND MODEL COMPRESSION FOR RADIAL NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a class of fully-connected neural networks whose activation functions, rather than being pointwise, rescale feature vectors by a function depending only on their norm. We call such networks *radial* neural networks, extending previous work on rotation equivariant networks that considers rescaling activations in less generality. We prove universal approximation theorems for radial neural networks, including in the more difficult cases of bounded widths and unbounded domains. Our proof techniques are novel, distinct from those in the pointwise case. Additionally, radial neural networks exhibit a rich group of orthogonal change-of-basis symmetries on the vector space of trainable parameters. Factoring out these symmetries leads to a practical lossless model compression algorithm. Optimization of the compressed model by gradient descent is equivalent to projected gradient descent for the full model.

1 INTRODUCTION

Inspired by biological neural networks, the theory of artificial neural networks has largely focused on pointwise (or "local") nonlinear layers (Rosenblatt, 1958; Cybenko, 1989), in which the same function $\sigma\colon \mathbb{R} \to \mathbb{R}$ is applied to each coordinate independently:

$$\mathbb{R}^n \to \mathbb{R}^n, \qquad v = (v_1\,,\,\ldots\,,\,v_n) \;\mapsto\; (\sigma(v_1)\,,\,\sigma(v_2)\,,\,\ldots\,,\,\sigma(v_n)). \tag{1.1}$$

In networks with pointwise nonlinearities, the standard basis vectors in $\mathbb{R}^n$ can be interpreted as "neurons" and the nonlinearity as a "neuron activation." Research has generally focused on finding functions σ which lead to more stable training, have less sensitivity to initialization, or are better adapted to certain applications (Ramachandran et al., 2017; Misra, 2019; Milletarí et al., 2018; Clevert et al., 2015; Klambauer et al., 2017). Many σ have been considered, including sigmoid, ReLU, arctangent, ELU, Swish, and others.

However, by setting aside the biological metaphor, it is possible to consider a much broader class of nonlinearities, which are not necessarily pointwise, but instead depend simultaneously on many coordinates. Freedom from the pointwise assumption allows one to design activations that yield expressive function classes with specific advantages. Additionally, certain choices of non-pointwise activations maximize symmetry in the parameter space of the network, leading to compressibility and other desirable properties.

In this paper, we introduce *radial* neural networks which employ non-pointwise nonlinearities called *radial rescaling* activations. Such networks enjoy several provable properties including high model compressibility, symmetry in optimization, and universal approximation. Radial rescaling activations are defined by rescaling each vector by a scalar that depends only on the norm of the vector:

$$\rho : \mathbb{R}^n \to \mathbb{R}^n, \qquad v \;\mapsto\; \lambda(|v|)v, \tag{1.2}$$

where λ is a scalar-valued function of the norm. Whereas in the pointwise setting, only the linear layers mix information between different components of the latent features, for radial rescaling, all coordinates of the activation output vector are affected by all coordinates of the activation input vector. The inherent geometric symmetry of radial rescalings makes them particularly useful for designing equivariant neural networks (Weiler & Cesa, 2019; Sabour et al., 2017; Weiler et al., 2018a;b).

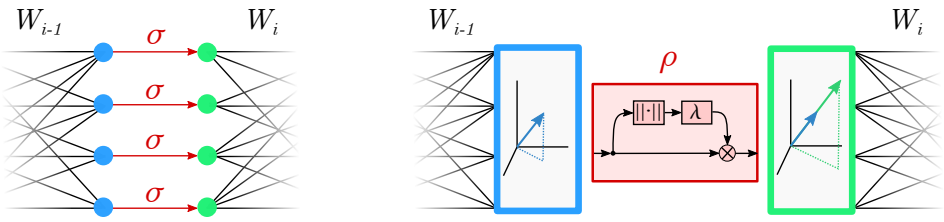

Figure 1: (Left) Pointwise activations distinguish a specific basis of each hidden layer and treat each coordinate independently, see equation 1.1. (Right) Radial rescaling activations rescale each feature vector by a function of the norm, see equation 1.2.

We note that radial neural networks constitute a simple and previously unconsidered type of multi-layer radial basis functions network (Broomhead & Lowe, 1988), namely, one where the number of hidden activation neurons (often denoted N) in each layer is equal to one. Indeed, pre-composing equation 1.2 with a translation and post-composing with a linear map, one obtains a special case of the local linear model extension of a radial basis functions network.

In our first set of main results, we prove that radial neural networks are in fact *universal approximators*. Specifically, we demonstrate that any asymptotically affine function can be approximated with a radial neural network, suggesting potentially good extrapolation behavior. Moreover, this approximation can be done with bounded width. Our approach to proving these results departs markedly from techniques used in the pointwise case. Additionally, our result is not implied by the universality property of radial basis functions networks in general, and differs in significant ways, particularly in the bounded width property and the approximation of asymptotically affine functions.

In our second set of main results, we exploit parameter space symmetries of radial neural networks to achieve *model compression*. Using the fact that radial rescaling activations commute with orthogonal transformations, we develop a practical algorithm to systematically factor out orthogonal symmetries via iterated QR decompositions. This leads to another radial neural network with fewer neurons in each hidden layer. The resulting model compression algorithm is *lossless*: the compressed network and the original network both have the same value of the loss function on any batch of training data.

Furthermore, we prove that the loss of the compressed model after one step of gradient descent is equal to the loss of the original model after one step of *projected gradient descent*. As explained below, projected gradient descent involves zeroing out certain parameter values after each step of gradient descent. Although training the original network may result in a lower loss function after fewer epochs, in many cases the compressed network takes less time per epoch to train and is faster in reaching a local minimum.

To summarize, our main contributions are:

- A formalization of radial neural networks, a new class of neural networks;
- Universal approximation results for radial neural networks, including: a) approximation of asymptotically affine functions, and b) bounded width approximation;
- A lossless compression algorithm for radial neural networks and a theorem providing the relationship between optimization of the original and compressed networks.
- Experiments verifying all theoretical results and showing that radial networks outperform pointwise networks on a noisy image recovery task.

2 Related work

Radial rescaling activations. As noted, radial rescaling activations are a special case of the activations used in radial basis functions networks (Broomhead & Lowe, 1988). Radial rescaling functions have the symmetry property of preserving vector directions, and hence exhibit rotation equivariance. Consequently, examples of such functions, such as the squashing nonlinearity and Norm-ReLU, feature in the study of rotationally equivariant neural networks (Weiler & Cesa, 2019; Sabour et al., 2017; Weiler et al., 2018a;b; Jeffreys & Lau, 2021). However, previous works apply

the activation only along the channel dimension, and consider the orthogonal group $O(n)$ only for $n = 2, 3$. In contrast, we apply the activation across the entire hidden layer, and $O(n)$-equivariance where n is the hidden layer dimension. Our constructions echo the vector neurons formalism (Deng et al., 2021), in which the output of a nonlinearity is a vector rather than a scalar.

Universal approximation. Neural networks of arbitrary width and sigmoid activations have long been known to be universal approximators (Cybenko, 1989). Universality can also be achieved by bounded width networks with arbitrary depth (Lu et al., 2017b), and generalizes to other activations and architectures (Hornik, 1991; Yarotsky, 2022; Ravanbakhsh, 2020; Sonoda & Murata, 2017). While most work has focused on compact domains, some recent work also considers non-compact domains (Kidger & Lyons, 2020; Wang & Qu, 2022). The techniques used for pointwise activations do not generalize to radial rescaling activations, where all activation output coordinates are affected by all input coordinates. Consequently, individual radial neural network approximators of two different functions cannot be easily combined to an approximator of the sum of the functions. The standard proof of universal approximation for radial basis functions networks requires an unbounded increase the number of hidden activation neurons, and hence does not apply to the case of radial neural networks (Park & Sandberg, 1991).

Groups and symmetry. Appearances of symmetry in machine learning have generally focused on symmetric input and output spaces. Most prominently, equivariant neural networks incorporate symmetry as an inductive bias and feature weight-sharing constraints based on equivariance. Examples include G-convolution, steerable CNN, and Clebsch-Gordon networks (Cohen et al., 2019; Weiler & Cesa, 2019; Cohen & Welling, 2016; Chidester et al., 2018; Kondor & Trivedi, 2018; Bao & Song, 2019; Worrall et al., 2017; Cohen & Welling, 2017; Weiler et al., 2018b; Dieleman et al., 2016; Lang & Weiler, 2021; Ravanbakhsh et al., 2017). By contrast, our approach to radial neural networks does not depend on symmetries of the input domain, output space, or feedforward mapping. Instead, we exploit parameter space symmetries and thus obtain more general results that apply to domains with no apparent symmetry.

Model compression. A major goal in machine learning is to find methods to reduce the number of trainable parameters, decrease memory usage, or accelerate inference and training (Cheng et al., 2017; Zhang et al., 2018). Our approach toward this goal differs significantly from most existing methods in that it is based on the inherent symmetry of network parameter spaces. One prior method is *weight pruning*, which removes redundant weights with little loss in accuracy (Han et al., 2015; Blalock et al., 2020; Karnin, 1990). Pruning can be done during training (Frankle & Carbin, 2018) or at initialization (Lee et al., 2019; Wang et al., 2020). *Gradient-based pruning* removes weights by estimating the increase in loss resulting from their removal (LeCun et al., 1990; Hassibi & Stork, 1993; Dong et al., 2017; Molchanov et al., 2016). A complementary approach is *quantization*, which decreases the bit depth of weights (Wu et al., 2016; Howard et al., 2017; Gong et al., 2014). *Knowledge distillation* identifies a small model mimicking the performance of a larger model (Buciluǎ et al., 2006; Hinton et al., 2015; Ba & Caruana, 2013). *Matrix Factorization* methods replace fully connected layers with lower rank or sparse factored tensors (Cheng et al., 2015a;b; Tai et al., 2015; Lebedev et al., 2014; Rigamonti et al., 2013; Lu et al., 2017a) and can often be applied before training. Our method involves a type of matrix factorization based on the QR decomposition; however, rather than aim for rank reduction, we leverage this decomposition to reduce hidden widths via change-of-basis operations on the hidden representations. Close to our method are lossless compression methods which remove stable neurons in ReLU networks (Serra et al., 2021; 2020) or exploit permutation parameter space symmetry to remove neurons (Sourek et al., 2020); our compression instead follows from the symmetries of the radial rescaling activation. Finally, the compression results of Jeffreys & Lau (2021), while conceptually similar to ours, are weaker, as (1) the unitary group action is on disjoint layers instead of moving through all layers, and (2) the results are only stated for the squashing nonlinearity.

3 RADIAL NEURAL NETWORKS

In this section, we define radial rescaling functions and radial neural networks. Let $h : \mathbb{R} \to \mathbb{R}$ be a function. For any $n \geq 1$, set:

$$h^{(n)} : \mathbb{R}^n \to \mathbb{R}^n \qquad h^{(n)}(v) = h(|v|)\frac{v}{|v|}$$

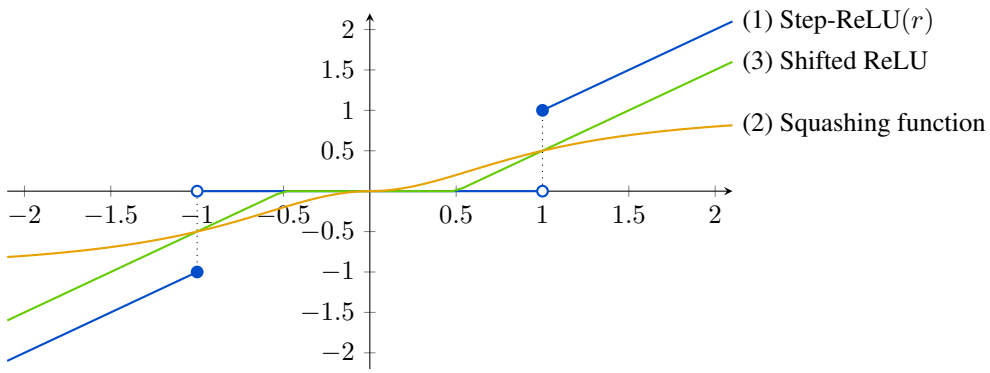

Figure 2: Examples of different radial rescaling functions in $\mathbb{R}^1$, see Example 1.

for $v \neq 0$, and $h^{(n)}(0) = 0$. A function $\rho : \mathbb{R}^n \to \mathbb{R}^n$ is called a *radial rescaling* function if $\rho = h^{(n)}$ for some piecewise differentiable $h : \mathbb{R} \to \mathbb{R}$. Hence, ρ sends each input vector to a scalar multiple of itself, and that scalar depends only on the norm of the vector[1]. It is easy to show that radial rescaling functions commute with orthogonal transformations.

Example 1. (1) Step-ReLU, where $h(r) = r$ if $r \geq 1$ and 0 otherwise. In this case, the radial rescaling function is given by

$$\rho : \mathbb{R}^n \to \mathbb{R}^n, \qquad v \mapsto v \text{ if } |v| \geq 1; \qquad v \mapsto 0 \text{ if } |v| < 1 \tag{3.1}$$

(2) The squashing function, where $h(r) = r^2/(r^2+1)$. (3) Shifted ReLU, where $h(r) = \max(0, r-b)$ for $r > 0$ and b is a real number. See Figure 2. We refer to Weiler & Cesa (2019) and the references therein for more examples and discussion of radial functions.

A *radial neural network* with L layers consists of positive integers n_i indicating the width of each layer $i = 0, 1, \ldots, L$; the trainable parameters, comprising of a matrix $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ of weights and a bias vector $b_i \in \mathbb{R}^{n_i}$ for each $i = 1, \ldots, L$; and a radial rescaling function $\rho_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ for each $i = 1, \ldots, L$. We refer to the tuple $\mathbf{n} = (n_0, n_1, \ldots, n_L)$ as the *widths vector* of the neural network. The hidden widths vector is $\mathbf{n}^{\text{hid}} = (n_1, n_2, \ldots, n_{L-1})$. The feedforward function $F : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ of a radial neural network is defined in the usual way as an iterated composition of affine maps and activations. Explicitly, set $F_0 = \text{id}_{\mathbb{R}^{n_0}}$ and the partial feedforward functions are:

$$F_i : \mathbb{R}^{n_0} \to \mathbb{R}^{n_i}, \qquad x \mapsto \rho_i\left(W_i \circ F_{i-1}(x) + b_i\right)$$

for $i = 1, \ldots, L$. Then the feedforward function is $F = F_L$. Radial neural networks are a special type of radial basis functions network; we explain the connection in Appendix F.

Remark 2. If $b_i = 0$ for all i, then we have $F(x) = W\left(\mu(x)x\right)$ where $\mu : \mathbb{R}^n \to \mathbb{R}$ is a scalar-valued function and $W = W_L W_{L-1} \cdots W_1 \in \mathbb{R}^{n_L \times n_0}$ is the product of the weight matrices. If any of the biases are non-zero, then the feedforward function lacks such a simple form.

4 Universal Approximation

We now consider two universal approximation results. The first approximates asymptotically affine functions with a network of unbounded width. The second generalizes to bounded width. Proofs appear in Appendix B. Throughout, $B_r(c) = \{x \in \mathbb{R}^n : |x - c| < r\}$ is the r-ball around a point c, and an affine map $\mathbb{R}^n \to \mathbb{R}^m$ is one of the from $L(x) = Ax + b$ for $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

4.1 Approximation of asymptotically affine functions

A continuous function $f : \mathbb{R}^n \to \mathbb{R}^m$ is *asymptotically affine* if there exists an affine map $L : \mathbb{R}^n \to \mathbb{R}^m$ such that, for every $\epsilon > 0$, there is a compact subset K of $\mathbb{R}^n$ such that $|L(x) - f(x)| < \epsilon$ for

[1] A function $\mathbb{R}^n \to \mathbb{R}$ that depends only on the norm of a vector is known as a *radial* function. Radial rescaling functions rescale each vector according to the radial function $v \mapsto \lambda(|v|) := \frac{h(|v|)}{|v|}$. This explains the connection to Equation 1.2.

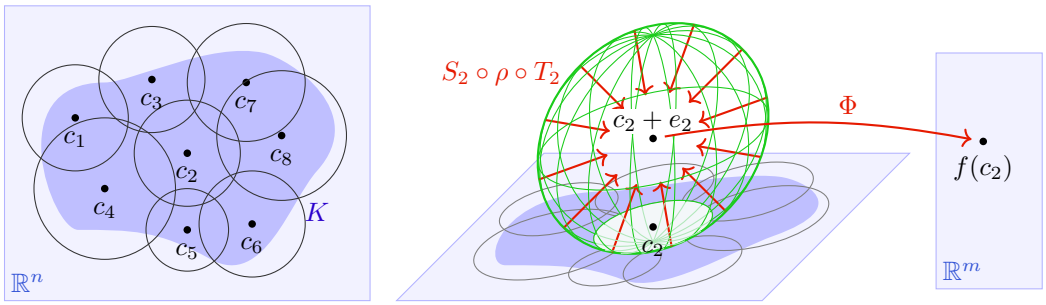

Figure 3: Two layers of the radial neural network used in the proof of Theorem 3. (Left) The compact set K is covered with open balls. (Middle) Points close to c_2 (green ball) are mapped to $c_2 + e_2$, all other points are kept the same. (Right) In the final layer, $c_2 + e_2$ is mapped to $f(c_2)$.

all $x \in \mathbb{R}^n \setminus K$. In particular, continuous functions with compact support are asymptotically affine. The continuity of f and compactness of K imply that, for any $\epsilon > 0$, there exist $c_1, \ldots, c_N \in K$ and $r_1, \ldots, r_N \in (0, 1)$ such that, first, the union of the balls $B_{r_i}(c_i)$ covers K and, second, for all i, we have $f\left(B_{r_i}(c_i) \cap K\right) \subseteq B_\epsilon(f(c_i))$. Let $N(f, K, \epsilon)$ be the minimal[2] choice of N.

Theorem 3 (Universal approximation)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be an asymptotically affine function. For any $\epsilon > 0$, there exists a compact set $K \subset \mathbb{R}^n$ and a function $F : \mathbb{R}^n \to \mathbb{R}^m$ such that:*

1. *F is the feedforward function of a radial neural network with $N = N(f, K, \epsilon)$ layers whose hidden widths are $(n + 1, n + 2, \ldots, n + N)$.*
2. *For any $x \in \mathbb{R}^n$, we have $|F(x) - f(x)| < \epsilon$.*

We note that the approximation in Theorem 3 is valid on all of $\mathbb{R}^n$, not only on K. To give an idea of the proof, first fix $c_1, \ldots, c_N \in K$ and $r_1, \ldots, r_N \in (0, 1)$ as above. Let $e_1, \ldots, e_N$ be orthonormal basis vectors extending $\mathbb{R}^n$ to $\mathbb{R}^{n+N}$. For $i = 1, \ldots, N$ define affine maps $T_i : \mathbb{R}^{n+i-1} \to \mathbb{R}^{n+i}$ and $S_i : \mathbb{R}^{n+i} \to \mathbb{R}^{n+i}$ by

$$T_i(z) = z - c_i + h_i e_i \qquad S_i(z) = z - (1 + h_i^{-1})\langle e_i, z\rangle e_i + c_i + e_i$$

where $h_i = \sqrt{1 - r_i^2}$ and $\langle e_i, z\rangle$ is the coefficient of e_i in z. Setting ρ_i to be Step-ReLU (Equation 3.1) on $\mathbb{R}^{n+i}$, these maps are chosen so that the composition $S_i \circ \rho_i \circ T_i$ maps the points in $B_{r_i}(c_i)$ to $c_i + e_i$, while keeping points outside this ball the same. We now describe a radial neural network with widths $(n, n+1, \ldots, n+N, m)$ whose feedforward function approximates f. For $i = 1, \ldots, N$ the affine map from layer $i - 1$ to layer i is given by $z \mapsto T_i \circ S_{i-1}(z)$, with $S_0 = \mathrm{id}_{\mathbb{R}^n}$. The activation at each hidden layer is Step-ReLU. Let L be the affine map such that $|L - f| < \epsilon$ on $\mathbb{R}^n \setminus K$. The affine map from layer N to the output layer is $\Phi \circ S_N$ where $\Phi : \mathbb{R}^{n+N} \to \mathbb{R}^m$ is the unique affine map determined by $x \mapsto L(x)$ if $x \in \mathbb{R}^n$, and $e_i \mapsto f(c_i) - L(c_i)$. See Figure 3 for an illustration of this construction. Theorem 3 has the following straightforward corollary:

Corollary 4. *Radial neural networks are dense in the space of all continuous functions with respect to the topology of compact convergence, and hence satisfy cc-universality.*

4.2 Bounded width approximation

We now turn our attention to a bounded width universal approximation result.

Theorem 5. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be an asymptotically affine function. For any $\epsilon > 0$, there exists a compact set $K \subset \mathbb{R}^n$ and a function $F : \mathbb{R}^n \to \mathbb{R}^m$ such that:*

1. *F is the feedforward function of a radial neural network with $N = N(f, K, \epsilon)$ hidden layers whose widths are all $n + m + 1$.*
2. *For any $x \in \mathbb{R}^n$, we have $|F(x) - f(x)| < \epsilon$.*

[2] In many cases, the constant $N(f, K, \epsilon)$ can be bounded explicitly. For example, if K is the unit cube in $\mathbb{R}^n$ and f is Lipschitz continuous with Lipschitz constant R, then $N(f, K, \epsilon) \leq \left\lceil \frac{R\sqrt{n}}{2\epsilon} \right\rceil^n$.

The proof, which is more involved than that of Theorem 3, relies on using orthogonal dimensions to represent the domain and the range of f, together with an indicator dimension to distinguish the two. We regard points in $\mathbb{R}^{n+m+1}$ as triples (x, y, θ) where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ and $\theta \in \mathbb{R}$. The proof of Theorem 5 parallels that of Theorem 3, but instead of mapping points in $B_{r_i}(c_i)$ to $c_i + e_i$, we map the points in $B_{r_i}((c_i, 0, 0))$ to $(0, \frac{f(c_i)-L(0)}{s}, 1)$, where s is chosen such that different balls do not interfere. The final layer then uses an affine map $(x, y, \theta) \mapsto L(x) + sy$, which takes $(x, 0, 0)$ to $L(x)$, and $(0, \frac{f(c_i)-L(0)}{s}, 1)$ to $f(c_i)$.

We remark on several additional results; see Appendix B for full statements and proofs. The bound of Theorem 5 can be strengthened to $\max(n, m) + 1$ in the case of functions $f : K \to \mathbb{R}^m$ defined on a compact domain $K \subset \mathbb{R}^n$ (i.e., ignoring asymptotic behavior). Furthermore, with more layers, it is possible to reduce that bound to $\max(n, m)$.

5 Model compression

In this section, we prove a model compression result. Specifically, we provide an algorithm which, given any radial neural network, computes a different radial neural network with smaller widths. The resulting compressed network has the same feedforward function as the original network, and hence the same value of the loss function on any batch of training data. In other words, our model compression procedure is *lossless*. Although our algorithm is practical and explicit, it reflects more conceptual phenomena, namely, a change-of-basis action on network parameter spaces.

5.1 Parameter space symmetries

Suppose a fully connected network has L layers and widths given by the tuple $\mathbf{n} = (n_0, n_1, n_2, \ldots, n_{L-1}, n_L)$. In other words, the i-th layer has input width n_{i-1} and output width n_i. The parameter space is defined as the vector space of all possible choices of parameter values. Hence, it is given by the following product of vector spaces:

$$\mathsf{Param}(\mathbf{n}) = \left(\mathbb{R}^{n_1 \times n_0} \times \mathbb{R}^{n_2 \times n_1} \times \cdots \times \mathbb{R}^{n_L \times n_{L-1}}\right) \times \left(\mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \cdots \times \mathbb{R}^{n_L}\right)$$

We denote an element therein as a pair of tuples $(\mathbf{W}, \mathbf{b})$ where $\mathbf{W} = (W_i \in \mathbb{R}^{n_i \times n_{i-1}})_{i=1}^L$ are the weights and $\mathbf{b} = (b_i \in \mathbb{R}^{n_i})_{i=1}^L$ are the biases. To describe certain symmetries of the parameter space, consider the following product of orthogonal groups, with sizes corresponding to the widths of the hidden layers:

$$O(\mathbf{n}^{\text{hid}}) = O(n_1) \times O(n_2) \times \cdots \times O(n_{L-1})$$

There is a change-of-basis action of $O(\mathbf{n}^{\text{hid}})$ on the parameter space $\mathsf{Param}(\mathbf{n})$. Explicitly, the tuple of orthogonal matrices $\mathbf{Q} = (Q_i)_{i=1}^{L-1} \in O(\mathbf{n}^{\text{hid}})$ transforms the parameter values $(\mathbf{W}, \mathbf{b})$ to $\mathbf{Q} \cdot \mathbf{W} := \left(Q_i W_i Q_{i-1}^{-1}\right)_{i=1}^L$ and $\mathbf{Q} \cdot \mathbf{b} := (Q_i b_i)_{i=1}^L$, where $Q_0 = \text{id}_{n_0}$ and $Q_L = \text{id}_{n_L}$.

5.2 Model compression

In order to state the compression result, we first define the reduced widths. Namely, the reduction $\mathbf{n}^{\text{red}} = (n_0^{\text{red}}, n_1^{\text{red}}, \ldots, n_L^{\text{red}})$ of a widths vector $\mathbf{n}$ is defined recursively by setting $n_0^{\text{red}} = n_0$, then $n_i^{\text{red}} = \min(n_i, n_{i-1}^{\text{red}} + 1)$ for $i = 1, \ldots, L-1$, and finally $n_L^{\text{red}} = n_L$. For a tuple $\boldsymbol{\rho} = (\rho_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i})_{i=1}^L$ of radial rescaling functions, we write $\boldsymbol{\rho}^{\text{red}} = \left(\rho_i^{\text{red}} : \mathbb{R}^{n_i^{\text{red}}} \to \mathbb{R}^{n_i^{\text{red}}}\right)$ for the corresponding tuple of restrictions, which are all radial rescaling functions. The following result relies on Algorithm 1 below.

Theorem 6. *Let $(\mathbf{W}, \mathbf{b}, \boldsymbol{\rho})$ be a radial neural network with widths $\mathbf{n}$. Let $\mathbf{W}^{\text{red}}$ and $\mathbf{b}^{\text{red}}$ be the weights and biases of the compressed network produced by Algorithm 1. The feedforward function of the original network $(\mathbf{W}, \mathbf{b}, \boldsymbol{\rho})$ coincides with that of the compressed network $(\mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}}, \boldsymbol{\rho}^{\text{red}})$.*

Algorithm 1: QR Model Compression (`QR-compress`)

input : $\mathbf{W}, \mathbf{b} \in \mathsf{Param}(\mathbf{n})$
output : $\mathbf{Q} \in O(\mathbf{n}^{\text{hid}})$ and $\mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}} \in \mathsf{Param}(\mathbf{n}^{\text{red}})$

$\mathbf{Q}, \mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}} \leftarrow [\,], [\,], [\,]$ `// initialize output lists`
$A_1 \leftarrow [b_1 \quad W_1]$ `// matrix of size` $n_1 \times (n_0 + 1)$
for $i \leftarrow 1$ **to** $L - 1$ **do** `// iterate through layers`
  $Q_i, R_i \leftarrow$ `QR-decomp(`A_i `, mode = 'complete')` `//` $A_i = Q_i \text{Inc}_i R_i$
  Append Q_i to $\mathbf{Q}$
  Append first column of R_i to $\mathbf{b}^{\text{red}}$ `// reduced bias for layer` i
  Append remainder of R_i to $\mathbf{W}^{\text{red}}$ `// reduced weights for layer` i
  Set $A_{i+1} \leftarrow [b_{i+1} \quad W_{i+1} Q_i \text{Inc}_i]$ `// matrix of size` $n_{i+1} \times (n_i^{\text{red}} + 1)$
end
Append the first column of A_L to $\mathbf{b}^{\text{red}}$ `// reduced bias for last layer`
Append the remainder of A_L to $\mathbf{W}^{\text{red}}$ `// reduced weights for last layer`
return $\mathbf{Q}, \mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}}$

We explain the notation of the algorithm. The inclusion matrix $\text{Inc}_i \in \mathbb{R}^{n_i \times n_i^{\text{red}}}$ has ones along the main diagonal and zeros elsewhere. The method `QR-decomp` with `mode = 'complete'` computes the complete QR decomposition of the $n_i \times (1 + n_{i-1}^{\text{red}})$ matrix A_i as $Q_i \text{Inc}_i R_i$ where $Q_i \in O(n_i)$ and R_i is upper-triangular of size $n_i^{\text{red}} \times (1 + n_{i-1}^{\text{red}})$. The definition of n_i^{red} implies that either $n_i^{\text{red}} = n_{i-1}^{\text{red}} + 1$ or $n_i^{\text{red}} = n_i$. The matrix R_i is of size $n_i^{\text{red}} \times n_i^{\text{red}}$ in the former case and of size $n_i \times (1 + n_{i-1}^{\text{red}})$ in the latter case.

Example 7. Suppose the widths of a radial neural network are $(1, 8, 16, 8, 1)$. Then it has $\sum_{i=1}^{4} (n_{i-1} + 1) n_i = 305$ trainable parameters. The reduced network has widths $(1, 2, 3, 4, 1)$ and $\sum_{i=1}^{4} (n_{i-1}^{\text{red}} + 1)(n_i^{\text{red}}) = 34$ trainable parameters. Another example appears in Figure 4.

We note that the tuple of matrices $\mathbf{Q}$ produced by Algorithm 1 does not feature in the statement of Theorem 6, but is important in the proof (which appears in Appendix C). Namely, an induction argument shows that the i-th partial feedforward function of the original and reduced models are related via the matrices Q_i and Inc_i. A crucial ingredient in the proof is that radial rescaling activations commute with orthogonal transformations.

6 Projected Gradient Descent

The typical use case for model compression algorithms is to produce a smaller version of the fully trained model which can be deployed to make inference more efficient. It is also worth considering whether compression can be used to accelerate training. For example, for some compression algorithms, the compressed and full models have the same feedforward function after a step of gradient descent is applied to each, and so one can compress before training and still reach the same minimum. Unfortunately, in the context of radial neural networks, compression using Algorithm 1 and then training does not necessarily give the same result as training and then compression (see Appendix D.6 for a counterexample). However, `QR-compress` does lead to a precise mathematical relationship between optimization of the two models: the loss of the compressed model after one

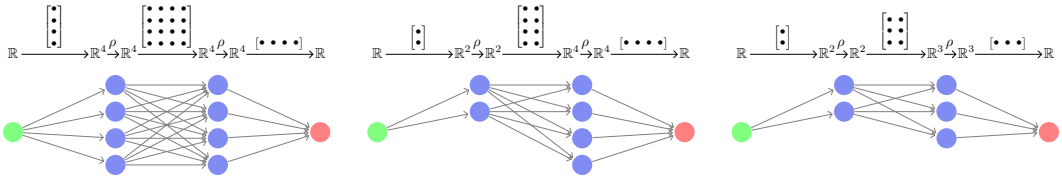

Figure 4: Model compression in 3 steps. Layer widths can be iteratively reduced to 1 greater than the previous. The number of trainable parameters reduces from 33 to 17.

step of gradient descent is equivalent to the loss of (a transformed version of) the original model after one step of projected gradient descent. Proofs appear in Appendix D.

To state our results, fix a tuple of widths $\mathbf{n}$ and a tuple $\boldsymbol{\rho} = (\rho_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i})_{i=1}^L$ of radial rescaling functions. The loss function $\mathcal{L} : \mathsf{Param}(\mathbf{n}) \to \mathbb{R}$ associated to a batch of training data $\{(x_j, y_j)\} \subseteq \mathbb{R}^{n_0} \times \mathbb{R}^{n_L}$ is defined as taking parameter values $(\mathbf{W}, \mathbf{b})$ to the sum $\sum_j \mathcal{C}(F(x_j), y_j)$ where $\mathcal{C} : \mathbb{R}^{n_L} \times \mathbb{R}^{n_L} \to \mathbb{R}$ is a cost function on the output space, and $F = F_{(\mathbf{W},\mathbf{b},\boldsymbol{\rho})}$ is the feedforward of the radial neural network with parameters $(\mathbf{W}, \mathbf{b})$ and activations $\boldsymbol{\rho}$. Similarly, we have a loss function $\mathcal{L}_{\text{red}}$ on the parameter space $\mathsf{Param}(\mathbf{n}^{\text{red}})$ with reduced widths vector. For any learning rate $\eta > 0$, we obtain gradient descent maps:

$$\begin{aligned} \gamma : \mathsf{Param}(\mathbf{n}) &\to \mathsf{Param}(\mathbf{n}) & \gamma_{\text{red}} : \mathsf{Param}(\mathbf{n}^{\text{red}}) &\to \mathsf{Param}(\mathbf{n}^{\text{red}}) \\ (\mathbf{W}, \mathbf{b}) &\mapsto (\mathbf{W}, \mathbf{b}) - \eta \nabla_{(\mathbf{W},\mathbf{b})} \mathcal{L} & (\mathbf{V}, \mathbf{c}) &\mapsto (\mathbf{V}, \mathbf{c}) - \eta \nabla_{(\mathbf{V},\mathbf{c})} \mathcal{L}_{\text{red}} \end{aligned}$$

We will also consider, for $k \geq 0$, the k-fold composition $\gamma^k = \gamma \circ \gamma \circ \cdots \circ \gamma$ and similarly for γ_{red}. The *projected gradient descent* map on $\mathsf{Param}(\mathbf{n})$ is given by:

$$\gamma_{\text{proj}} : \mathsf{Param}(\mathbf{n}) \to \mathsf{Param}(\mathbf{n}), \qquad (\mathbf{W}, \mathbf{b}) \mapsto \mathrm{Proj}\left(\gamma(\mathbf{W}, \mathbf{b})\right)$$

where the map Proj zeroes out all entries in the bottom left $(n_i - n_i^{\text{red}}) \times n_{i-1}^{\text{red}}$ submatrix of $W_i - \nabla_{W_i} \mathcal{L}$, and the bottom $(n_i - n_i^{\text{red}})$ entries in $b_i - \nabla_{b_i} \mathcal{L}$, for each i. Schematically:

$$W_i - \nabla_{W_i} \mathcal{L} = \begin{bmatrix} * & * \\ * & * \end{bmatrix} \mapsto \begin{bmatrix} * & * \\ 0 & * \end{bmatrix}, \qquad b_i - \nabla_{b_i} \mathcal{L} = \begin{bmatrix} * \\ * \end{bmatrix} \mapsto \begin{bmatrix} * \\ 0 \end{bmatrix}$$

To state the following theorem, let $\mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}}, \mathbf{Q} = \texttt{QR-compress}(\mathbf{W}, \mathbf{b})$ be the outputs of Algorithm 1 applied to $(\mathbf{W}, \mathbf{b}) \in \mathsf{Param}(\mathbf{n})$. Hence $(\mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}}) \in \mathsf{Param}(\mathbf{n}^{\text{red}})$ are the parameters of the compressed model, and $\mathbf{Q} \in O(\mathbf{n}^{\text{hid}})$ is an orthogonal parameter symmetry. We also consider the action (Section 5.1) of $\mathbf{Q}^{-1}$ applied to $(\mathbf{W}, \mathbf{b})$.

Theorem 8. *Let $\mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}}, \mathbf{Q} = \texttt{QR-compress}(\mathbf{W}, \mathbf{b})$ be the outputs of Algorithm 1 applied to $(\mathbf{W}, \mathbf{b}) \in \mathsf{Param}(\mathbf{n})$. Set $\mathbf{U} = \mathbf{Q}^{-1} \cdot (\mathbf{W}, \mathbf{b}) - (\mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}})$. For any $k \geq 0$, we have:*

$$\gamma^k(\mathbf{W}, \mathbf{b}) = \mathbf{Q} \cdot \gamma^k(\mathbf{Q}^{-1} \cdot (\mathbf{W}, \mathbf{b})) \qquad \gamma_{\text{proj}}^k(\mathbf{Q}^{-1} \cdot (\mathbf{W}, \mathbf{b})) = \gamma_{\text{red}}^k(\mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}}) + \mathbf{U}.$$

We conclude that gradient descent with initial values $(\mathbf{W}, \mathbf{b})$ is equivalent to gradient descent with initial values $\mathbf{Q}^{-1} \cdot (\mathbf{W}, \mathbf{b})$ since at any stage we can apply $\mathbf{Q}^{\pm 1}$ to move from one to the other. Furthermore, projected gradient descent with initial values $\mathbf{Q}^{-1} \cdot (\mathbf{W}, \mathbf{b})$ is equivalent to gradient descent on $\mathsf{Param}(\mathbf{n}^{\text{red}})$ with initial values $(\mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}})$ since at any stage we can move from one to the other by $\pm \mathbf{U}$. Neither $\mathbf{Q}$ nor $\mathbf{U}$ depends on k.

7 Experiments

In addition to our theoretical results, we provide an implementation of Algorithm 1 in order to validate the claims of Theorems 6 and 8 empirically, as well as a demonstration that a radial network outperforms a MLP on a noisy image recovery task. Full experimental details are in Appendix E.

(1) Empirical verification of Theorem 6. We learn the function $f(x) = e^{-x^2}$ from samples using a radial neural network with widths $\mathbf{n} = (1, 6, 7, 1)$ and activation the radial shifted sigmoid $h(x) = 1/(1 + e^{-x+s})$. Applying `QR-compress` gives a compressed radial neural network with widths $\mathbf{n}^{\text{red}} = (1, 2, 3, 1)$. Theorem 6 implies that the respective neural functions F and F_{red} are equal. Over 10 random initializations, the mean absolute error is negligible up to machine precision: $(1/N) \sum_j |F(x_j) - F_{\text{red}}(x_j)| = 1.31 \cdot 10^{-8} \pm 4.45 \cdot 10^{-9}$.

(2) Empirical verification of Theorem 8. The claim is that training the transformed model with parameters $\mathbf{Q}^{-1} \cdot (\mathbf{W}, \mathbf{b})$ and objective $\mathcal{L}$ by projected gradient descent coincides with training the reduced model with parameters $(\mathbf{W}^{\text{red}}, \mathbf{b}^{\text{red}})$ and objective $\mathcal{L}_{\text{red}}$ by usual gradient descent. We verified this on synthetic data as above. Over 10 random initializations, the loss functions after training match: $|\mathcal{L} - \mathcal{L}_{\text{red}}| = 4.02 \cdot 10^{-9} \pm 7.01 \cdot 10^{-9}$.

(3) The compressed model trains faster. Our compression method may be applied before training to produce a smaller model class which *trains* faster without sacrificing accuracy. We demonstrate

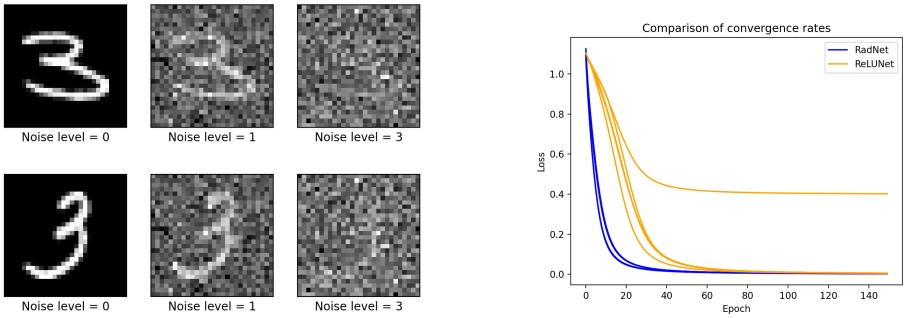

Figure 5: (Left) Different levels of noise. (Right) Training five Step-ReLU radial networks and five ReLU MLPs on data with `n`=3 original images, `m`=100 noisy copies of each.

this in learning the function $\mathbb{R}^2 \to \mathbb{R}^2$ sending (t_1, t_2) to $(e^{-t_1^2}, e^{-t_2^2})$ using a radial neural network with widths $(2, 16, 64, 128, 16, 2)$ and activation the radial sigmoid $h(r) = 1/(1 + e^{-r})$. Applying `QR-compress` gives a compressed network with widths $\mathbf{n}^{\text{red}} = (2, 3, 4, 5, 6, 2)$. We trained both models until the training loss was ≤ 0.01. Over 10 random initializations on our system, the reduced network trained in 15.32 ± 2.53 seconds and the original network trained in 31.24 ± 4.55 seconds.

(4) Noisy image recovery. A Step-ReLU radial network performs better than an otherwise comparable network with pointwise ReLU on a noisy image recovery task. Using samples of MNIST with significant added noise, the network must identify from which original sample the noisy sample derives (see Figure 5). We observe that the radial network 1) is able to obtain a better fit, 2) has faster convergence, and 3) generalizes better than the pointwise ReLU. We hypothesize the radial nature of the random noise makes radial networks well-adapted to the task. Our data takes `n` = 3 original MNIST images with the same label, and produces `m` = 100 noisy images for each, with a 240 train / 60 test split. Over 10 trials, each training for 150 epochs, the radial network achieves training loss 0.00256 $\pm 3.074 \cdot 10^{-1}$ with accuracy 1 ± 0, while the ReLU MLP has training loss 0.295 $\pm 2.259 \cdot 10^{-1}$ with accuracy 0.768 $\pm 2.199 \cdot 10^{-1}$. On the test set, the radial network has loss 0.00266 $\pm 3.749 \cdot 10^{-4}$ with accuracy 1 ± 0, while the ReLU MLP has loss 0.305 $\pm 2.588 \cdot 10^{-1}$ with accuracy 0.757 $\pm 2.464 \cdot 10^{-1}$. The convergence rates are illustrated in Figure 5, with the radial network outperforming the ReLU MLP, and 150 epochs are sufficient for all methods to converge.

8 Conclusions and Discussion

This paper demonstrates that radial neural networks are universal approximators and that their parameter spaces exhibit a rich symmetry group, leading to a model compression algorithm. The results of this work combine to build a theoretical foundation for the use of radial neural networks, and suggest that radial neural networks hold promise for wider practical applicability. Furthermore, this work makes an argument for considering non-pointwise nonlinearities in neural networks.

There are two main limitations of our results, each providing an opportunity for future work. First, our universal approximation constructions currently work only for Step-ReLU radial rescaling radial activations; it would be desirable to generalize to other activations. Additionally, Theorem 6 achieves compression only for networks whose widths satisfy $n_i > n_{i-1} + 1$ for some i. Networks which do not have increasing widths anywhere, such as encoders, would not be compressible.

Further extensions of this work include: First, little is currently known about the stability properties of radial neural networks during training, as well as their sensitivity to initialization. Second, radial rescaling activations provide an extreme case of symmetry; there may be benefits to combining radial and pointwise activations within a single network, for example, through 'block' radial rescaling functions. Our techniques may yield weaker compression properties for more general radial basis functions networks; radial neural networks may be the most compressible such networks. Third, radial rescaling activations can be used within convolutional or group-equivariant NNs. Finally, based on the theoretical advantages and experiments laid out in this paper, future empirical work will further explore applications in which we expect radial networks to outperform alternate methods. Such potential applications include data spaces with circular or distance-based class boundaries.

ETHICS STATEMENT

Our work is primarily focused on theoretical foundations of machine learning, however, it does have a direct application in the form a model compression. Model compression is largely beneficial to the world since it allows for inference to run on smaller systems which use less energy. On the other hand, when models may be run on smaller systems such as smartphones, it is easier to use deep models covertly, for example, for facial recognition and surveillance. This may make abuses of deep learning technology easier to hide.

REPRODUCIBILITY STATEMENT

The theoretical results of this paper, namely Theorem 3, Theorem 5, Theorem 6, and Theorem 8, may be independently verified through their proofs, which we include in their entirety in the appendices, including all necessary definitions, lemmas, and hypotheses in precise and complete mathematical language. The empirical verification of Section 7 may be reproduced using the code included with the supplementary materials. In addition, Algorithm 1 is written in detailed pseudocode, allowing readers to recreate our algorithm in a programming language of their choosing.

REFERENCES

Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *arXiv:1312.6184*, 2013.

Erkao Bao and Linqi Song. Equivariant neural networks and equivarification. *arXiv:1906.07172*, 2019.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv:2003.03033*, 2020.

David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 535–541, 2006.

Yu Cheng, X Yu Felix, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shih-Fu Chang. Fast neural networks with circulant projections. *arXiv:1502.03436*, 2, 2015a.

Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE international conference on computer vision*, pp. 2857–2865, 2015b.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv:1710.09282*, 2017.

Benjamin Chidester, Minh N. Do, and Jian Ma. Rotation equivariance and invariance in convolutional neural networks. *arXiv:1805.12301*, 2018.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning (ICML)*, pp. 2990–2999, 2016.

Taco S Cohen and Max Welling. Steerable CNNs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Taco S. Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pp. 1321–1330, 2019.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Congyue Deng, O. Litany, Yueqi Duan, A. Poulenard, A. Tagliasacchi, and L. Guibas. Vector Neurons: A General Framework for SO(3)-Equivariant Networks. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. doi: 10.1109/iccv48922.2021.01198.

Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2016.

Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *arXiv preprint arXiv:1705.07565*, 2017.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv:1803.03635*, 2018.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv:1412.6115*, 2014.

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

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Babak Hassibi and David G Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4 (2):251–257, 1991.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

George Jeffreys and Siu-Cheong Lau. Kähler Geometry of Quiver Varieties and Machine Learning. *arXiv:2101.11487*, 2021. URL http://arxiv.org/abs/2101.11487.

Ehud D Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE transactions on neural networks*, 1(2):239–242, 1990.

Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks. In *Conference on learning theory*, pp. 2306–2327. PMLR, 2020.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Risi Kondor and Shubhendu Trivedi. On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups. In *International conference on machine learning (ICML)*, 2018.

Leon Lang and Maurice Weiler. A Wigner-Eckart theorem for group equivariant convolution kernels. In *International Conference on Learning Representations (ICLR)*, 2021.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv:1412.6553*, 2014.

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

Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip HS Torr. A signal propagation perspective for pruning neural networks at initialization. *arXiv preprint arXiv:1906.06307*, 2019.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 5334–5343, 2017a.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017b.

Mirco Milletarí, Thiparat Chotibut, and Paolo E Trevisanutto. Mean field theory of activation functions in deep neural networks. *arXiv preprint arXiv:1805.08786*, 2018.

Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems (NeurIPS) 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Siamak Ravanbakhsh. Universal equivariant multilayer perceptrons. In *International Conference on Machine Learning*, pp. 7996–8006. PMLR, 2020.

Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing. In *International Conference on Machine Learning*, pp. 2892–2901. PMLR, 2017.

Roberto Rigamonti, Amos Sironi, Vincent Lepetit, and Pascal Fua. Learning separable filters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2754–2761, 2013.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv:1710.09829*, 2017.

Thiago Serra, Abhinav Kumar, and Srikumar Ramalingam. Lossless compression of deep neural networks. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 417–430. Springer, 2020.

Thiago Serra, Xin Yu, Abhinav Kumar, and Srikumar Ramalingam. Scaling up exact neural network compression by relu stability. *Advances in Neural Information Processing Systems*, 34, 2021.

Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017.

Gustav Sourek, Filip Zelezny, and Ondrej Kuzelka. Lossless compression of structured convolutional models via lifting. *arXiv preprint arXiv:2007.06567*, 2020.

Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv:1511.06067*, 2015.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

Ming-Xi Wang and Yang Qu. Approximation capabilities of neural networks on unbounded domains. *Neural Networks*, 145:56–67, 2022.

Maurice Weiler and Gabriele Cesa. General $E(2)$-Equivariant Steerable CNNs. *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS)*, 2018a.

Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 849–858, 2018b.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5028–5037, 2017.

Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4820–4828, 2016.

Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1):407–474, 2022.

Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic DNN weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 184–199, 2018.