Exploring the Approximation Capabilities of Multiplicative Neural Networks for Smooth Functions

Anonymous authors
Paper under double-blind review

Abstract

Multiplication layers are a key component in various influential neural network modules, including self-attention and hypernetwork layers. In this paper, we investigate the approximation capabilities of deep neural networks with intermediate neurons connected by simple multiplication operations. We consider two classes of target functions: generalized bandlimited functions, which are frequently used to model real-world signals with finite bandwidth, and Sobolev-Type balls, which are embedded in the Sobolev Space $\mathcal{W}^{r,2}$. Our results demonstrate that multiplicative neural networks can approximate these functions with significantly fewer layers and neurons compared to standard ReLU neural networks, with respect to both input dimension and approximation error. These findings suggest that multiplicative gates can outperform standard fully-connected layers and potentially improve neural network design.

1 Introduction

Deep learning with large neural networks has seen tremendous success in solving a wide range of tasks in recent years, including image classification (He et al., 2016; Dosovitskiy et al., 2021; Zhai et al., 2021), language processing (Vaswani et al., 2017; Devlin et al., 2019; Brown et al., 2020), interacting with open-ended environments (Silver et al., 2016; Arulkumaran et al., 2019), and code synthesis (Chen et al., 2021).

Recent empirical studies have shown that neural networks that incorporate multiplication operations between intermediate neurons (Durbin & Rumelhart, 1989; Urban & van der Smagt, 2015; Trask et al., 2018), such as self-attention layers (Vaswani et al., 2017), dynamic convolutions (Wu et al., 2019) and hypernetworks (Ha et al., 2016; Krueger et al., 2017; Littwin & Wolf, 2019), are particularly effective. For example, self-attention layers have been widely successful in computer vision (Dosovitskiy et al., 2021; Zhai et al., 2021) and language processing (Cheng et al., 2016; Parikh et al., 2016; Paulus et al., 2017; Vaswani et al., 2017). It has also been shown that one can achieve reasonable performance with Transformers even without applying non-linear activation functions (Levine et al., 2020). Additionally, hypernetworks, which use multiplication to generate network weights seem to improve the performance of neural networks on various meta-learning tasks (von Oswald et al., 2020; Littwin & Wolf, 2019; Bensadoun et al., 2021). In a different line of work, Haber & Ruthotto (2018); Eliasof et al. (2021) have shown that incorporating non-linear interactions can stabilize the training of deep networks, including convolutional and graph neural networks. However, there is still a lack of theoretical understanding as to why multiplicative operators are effective.

In this work, we study the expressive power of neural networks with multiplication layers. Specifically, we want to evaluate the number of neurons and layers needed to approximate a given function within a given error tolerance using a specific architecture. A classic result in the theory of deep learning shows that neural networks can approximate any smooth target function, known as the universal approximation property, with as few as one hidden layer (Cybenko, 1989; Hornik et al., 1989; ichi Funahashi, 1989; Leshno et al., 1991). However, these papers do not provide specific information about the type of architecture and number of parameters required to achieve a given level of accuracy. This is a crucial question, as a high requirement for these resources could limit the universality of neural networks and explain their limited success in some practical applications.

In a recent paper, Jayakumar et al. (2020) investigated the role of multiplicative interaction as a framework including various classical and contemporary neural network components, including gating, attention layers, hypernetworks, and dynamic convolutions. The authors conjectured that multiplicative layers are well-suited for modeling conditional computations and proved that networks with multiplicative connections are more expressive than those with ReLU fully-connected layers. However, they did not investigate to what extent the multiplicative networks serve as better function approximators than standard ReLU networks.

Previous work has demonstrated that functions in Sobolev spaces can be approximated by a one-hidden layer neural network with analytic activation functions (Mhaskar, 1996). However, the number of neurons required to approximate these functions with an error of at most ϵ in the L_∞ norm scales as $\mathcal{O}(\epsilon^{-d/r})$, where d is the input dimension, r is the smoothness degree of the target function, and $\epsilon > 0$ is the error rate. This raises the question of whether the curse of dimensionality, the phenomenon whereby the complexity of a model grows exponentially with the input dimension, is inherent to neural networks.

On the other hand, DeVore et al. (1989) proved that any continuous function approximator that approximates all Sobolev functions of order r and dimension d within error ϵ requires at least $\Omega(\epsilon^{-d/r})$ parameters in the L_∞ norm. This result meets the bound of Mhaskar (1996) and confirms that neural networks cannot avoid the curse of dimensionality for the Sobolev space when approximating in the L_∞ norm. A key question is whether neural networks can overcome this curse of dimensionality for certain sets of target functions, and what kind of architectures provide the best guarantees for approximating these functions.

To overcome the curse of dimensionality, various studies (Mhaskar et al., 2017; Poggio et al., 2020; Kohler & Krzyżak, 2017; Montanelli & Du, 2019; Blanchard & Bennouna, 2022b; Galanti & Wolf, 2020) have investigated the approximation capabilities of neural networks in representing other classes of functions with weaker notions of distance, such as the L_2 distance. For example, Mhaskar et al. (2017); Poggio et al. (2020) showed that smooth, compositionally sparse functions with a degree of smoothness r can be approximated with the L_∞ distance up to error ϵ using deep neural networks with $\mathcal{O}(d\epsilon^{-2/r})$ neurons. Other structural constraints have been applied to functions with structured input spaces (Mhaskar, 2010; Nakada & Imaizumi, 2022; Schmidt-Hieber, 2019), compositions of functions (Kohler & Krzyżak, 2017), piecewise smooth functions (Petersen & Voigtländer, 2017; Imaizumi & Fukumizu, 2018). A different line of research has focused on understanding the types of functions that certain neural network architectures can implement with regularity constraints. For example, E et al. (2021) showed that the space of 2-layer neural networks is equivalent to the Barron space when the size of their weights is restricted. They further showed that Barron functions can be approximated within ϵ using 2-layer networks with $\mathcal{O}(\epsilon^{-2})$ neurons. Another line of research has considered spectral conditions on the function space, allowing functions to be expressed as infinite-width limits of shallow networks (Barron, 1991; Klusowski & Barron, 2016). In (Blanchard & Bennouna, 2022b) they considered the space of Korobov functions, which are functions that are practically useful for solving partial differential equations (PDEs). They showed any Korobov function can be approximated up to error ϵ in L_2 distance with a 2-layer neural network with ReLU activation function with $\mathcal{O}(\epsilon^{-1}\log(1/\epsilon)^{1.5(d-1)+1})$ and with a $\mathcal{O}(\log(d))$-depth network with $\mathcal{O}(\epsilon^{-0.5}\log(1/\epsilon)^{1.5(d-1)+1})$ neurons.

In a recent paper, Montanelli et al. (2021) provided approximation guarantees were established for generalized bandlimited functions. These functions are commonly used to model signals that have a finite range of frequencies (e.g., waves, video, and audio signals), which is known as a finite bandwidth. The solutions to many PDEs in physics are bandlimited functions, as the physical phenomena modeled by these PDEs typically have a finite range of frequencies. For example, the solutions to the wave equation, which models the propagation of waves, are bandlimited functions. In (Montanelli et al., 2021), it was shown that any bandlimited function can be approximated to within error ϵ using a ReLU neural network of depth $\mathcal{O}(\log^2(1/\epsilon))$ with $\mathcal{O}(\epsilon^{-2}\log^2(1/\epsilon))$ neurons with the L_2 distance.

In this paper, we compare the approximation capabilities of multiplicative neural network architectures with those of standard ReLU networks with respect to the L_2 distance. In particular, we prove that a multiplicative neural network of depth $\mathcal{O}(\log(\frac{1}{\epsilon}))$ with $\mathcal{O}(\epsilon^{-2}\log(\frac{1}{\epsilon}))$ neurons can approximate any generalized bandlimited function up to an error of ϵ (with constants depending on the dimension and on the band). This result represents an improvement compared to the findings in (Montanelli et al., 2021) for traditional ReLU networks. Additionally, we also study the approximation guarantees of neural networks for approximating

functions in Sobolev-Type balls of order r. We show that for the same error tolerance ϵ, multiplicative neural networks can approximate these functions with depth $\mathcal{O}(d\epsilon^{-1/r})$ and $\mathcal{O}(d\epsilon^{-(2+1/r)})$ neurons, while standard ReLU neural networks require depth $\mathcal{O}(d^2\epsilon^{-2/r})$ and $\mathcal{O}(d^2\epsilon^{-(2+2/r)})$ neurons. These results demonstrate the superior performance of multiplicative gates compared to standard fully-connected layers. In Table 1 we contrast our new bounds with preexisting bounds on the approximation power of neural networks for the Sobolev space, bandlimited functions, and the Sobolev-Type ball.

Space	Model	# neurons	Depth	Reference
$\mathcal{W}^{r,p}$	C^∞, non-poly	$\mathcal{O}(\epsilon^{-d/r})$	$\mathcal{O}(1)$	(Mhaskar, 1996)
$\mathcal{W}^{r,\infty}$	ReLU	$\mathcal{O}(\epsilon^{-d/r}\log\frac{1}{\epsilon})$	$\mathcal{O}(\log\frac{1}{\epsilon})$	(Yarotsky, 2017)
Bandlimited functions	ReLU	$\mathcal{O}(\epsilon^{-2}\log^2\frac{1}{\epsilon})$	$\mathcal{O}(\log^2\frac{1}{\epsilon})$	(Montanelli et al., 2021)
Bandlimited functions	Multiplicative	$\mathcal{O}(\epsilon^{-2}\log\frac{1}{\epsilon})$	$\mathcal{O}(\log\frac{1}{\epsilon})$	This paper
$\mathcal{B}_{2r,2}\subsetneq\mathcal{W}^{r,2}$	ReLU	$\mathcal{O}(d^2\epsilon^{-(2+2/r)})$	$\mathcal{O}(d^2\epsilon^{-2/r})$	This paper
$\mathcal{B}_{2r,2}\subsetneq\mathcal{W}^{r,2}$	Multiplicative	$\mathcal{O}(d\epsilon^{-(2+1/r)})$	$\mathcal{O}(d\epsilon^{-1/r})$	This paper
$\mathcal{B}_{1,1}$	Sigmodial	$\mathcal{O}(d\epsilon^{-2})$	$\mathcal{O}(1)$	(Barron, 1993)

Table 1: Approximation results for Sobolev $\mathcal{W}^{r,p}$, Bandlimited and $\mathcal{B}_{2r,2}$ functions by ReLU and multiplicative neural networks. The number of neurons and the depth are given in $\mathcal{O}$ notation.

2 Problem Setup

We are interested in determining how complex (i.e., number of trainable parameters, number of neurons and layers) a model ought to be in order to theoretically guarantee approximation of an unknown target function f up to a given approximation error $\epsilon > 0$.

Formally, we consider a Banach space of functions $\mathcal{V}$ (for example, $L_p([0,1]^d)$), equipped with a norm $\|\cdot\|_{\mathcal{V}}$ (for example, $\|\cdot\|_{L_p([0,1]^d)}$), and a set of target functions $\mathcal{U}\subseteq\mathcal{V}$. We also consider a set of approximators $\mathcal{H}$ and seek to quantify the ability of these approximators to approximate $\mathcal{U}$ using the following quantity

$$d_{\mathcal{V}}(\mathcal{H},\mathcal{U}) \;=\; \inf_{\hat{f}\in\mathcal{H}}\sup_{f\in\mathcal{U}}\|\hat{f}-f\|_{\mathcal{V}},$$

which measures the maximal approximation error for approximating a target function $f\in\mathcal{U}$ using candidates $\hat{f}$ from $\mathcal{H}$. Typically, $\mathcal{H}$ is a parametric set of functions (e.g., neural networks of a certain architecture) and we denote by $\hat{f}_\theta\in\mathcal{H}$ a function that is parameterized by a vector of parameters $\theta\in\mathbb{R}^N$. For simplicity, we avoid writing θ in the subscript when it is obvious from context.

2.1 Target Function Spaces

It is generally impossible to approximate arbitrary target functions using standard neural networks, as demonstrated in Theorem 7.2 in (Devroye et al., 1996). As a result, we often consider specific spaces of target functions that satisfy certain smoothness assumptions in order to obtain non-trivial results. In this work, we focus specifically on target functions $\mathcal{U}$ over the unit cube $B=[0,1]^d$ that satisfy the following types of smoothness assumptions.

Sobolev spaces. Sobolev spaces are one of the most extensively studied classes of functions in approximation theory (DeVore & Lorentz, 1993; Yarotsky, 2017; Liang & Srikant, 2017). These spaces consist of smooth functions with bounded derivatives up to a certain order and are particularly useful in the study of partial differential equations (PDEs).

We first define the L_p norm of a given function $f:\Omega\to\mathbb{R}$ as $\|f\|_{L_p(\Omega)}=(\int_\Omega|f(x)|^p\,\mathrm{d}x)^{1/p}$, where Ω is a measurable space equipped with a Sigma-algebra Σ and a measure μ. A function f is said to be in $L_p(\Omega)$ if $\|f\|_{L_p(\Omega)}<\infty$.

Let $r \in \mathbb{N}$ and $p \in [1, \infty)$. The Sobolev space $\mathcal{W}^{r,p}(B)$ consists of functions $f : B \to \mathbb{R}$ with r-distributional derivatives in L_p. The Sobolev norm $\|\cdot\|_{\mathcal{W}^{r,p}(B)}$ is defined as

$$\|f\|_{\mathcal{W}^{r,p}(B)} \;=\; \sum_{\alpha:|\alpha|_1 \leq r} \|D^\alpha f\|_{L_p(B)},$$

where $\alpha = (\alpha_1, \ldots, \alpha_d) \in \{0, \ldots, r\}^d$, $|\alpha|_1 = \alpha_1 + \cdots + \alpha_d$, and $D^\alpha f$ is the respective distributional derivative. When $p = \infty$, the essential supremum norm is used. We also present the semi-norm:

$$|f|_{r,p} \;=\; \sum_{\alpha \,:\, |\alpha|=r} \|D^\alpha f(x)\|_{L_p(B)}.$$

Classic results in the literature show that the number of parameters needed to approximate functions in $\mathcal{W}^{r,\infty}$ up to error ϵ is lower bounded by $\Omega(\epsilon^{-d/r})$ (DeVore et al., 1989). This exponential dependence on d is known as the "curse of dimensionality".

Generalized Bandlimited functions. Generalized bandlimited functions are functions whose spectrum, or the set of frequencies that make up the function, is limited to a certain band or range of frequencies. This property makes bandlimited functions well-suited for certain applications, such as signal processing, where it is important to ensure that the signal does not contain frequencies outside of a certain range.

Generalized bandlimited functions are defined using their generalized Fourier transform given by an analytic function $K : \mathbb{R} \to \mathbb{C}$ (known as the kernel) and a scalar $M \geq 1$ (the band). Formally, for a given function $f : \mathbb{R}^d \to \mathbb{R}$, we denote the set of all functions $F : [-M, M]^d \to \mathbb{C}$ that retrieve f from the frequencies $\omega \in [-M, M]^d$ using the kernel K as follows:

$$S_{f,K} \;=\; \left\{F : [-M, M]^d \to \mathbb{R} \mid f(x) = \textstyle\int_{[-M,M]^d} F(\omega)K(\omega \cdot x)\ \mathrm{d}\omega\right\}.$$

We define $F_f = \arg\min_{F \in S_{f,K}} \|F\|_{L^2([-M,M]^d)}$ to be the smallest L_2-norm function in $S_{f,K}$. For instance, when $K(x) = \exp(ix)$, the function F_f corresponds to the normalized standard Fourier transform of f, which is given by $F = \frac{1}{(2\pi)^d}(\mathcal{F}f)(\omega) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} f(x) \exp(-i\omega \cdot x)\ \mathrm{d}x$.

The space $\mathcal{H}_{K,M}(B)$ of generalized bandlimited functions is a Hilbert space of functions that can be represented as a weighted sum of the function K over a finite domain. This space is equipped with an inner product and a norm, which allow us to measure the similarity and magnitude of these functions, respectively. We define $\mathcal{H}_{K,M}(B)$ as the functions $f : B \to \mathbb{R}$ such that

$$\mathcal{H}_{K,M}(B) \;=\; \left\{\forall x \in B, f(x) = \textstyle\int_{[-M,M]^d} F(\omega)K(\omega \cdot x)\ \mathrm{d}\omega \mid F : [-M, M]^d \to \mathbb{C} \text{ is in } L^2([-M, M]^d)\right\}.$$

The inner product and norm in this space are defined as follows: $\langle f, g\rangle_{\mathcal{H}_{K,M}(B)} = \int_{[-M,M]^d} F_f(\omega)\overline{F_g}(\omega)\ \mathrm{d}\omega$ and norm $\|f\|_{\mathcal{H}_{K,M}(B)} = \|F_f\|_{L^2([-M,M]^d)}$.

One of the key properties of generalized bandlimited functions is that they can be completely reconstructed from a discrete set of samples. This is known as the Shannon-Nyquist theorem (Shannon, 1984), and it is an important result in the field of signal processing and communication. An interesting consequence of this theorem is that even in high-dimensions, where seemingly unpredictable geometrical phenomena may occur (e.g. Blum et al. (2020), Chapter 2), still perfectly reconstruct a function given it's values at the Nyquist frequency. For more details, see Appendix A.

Sobolev-Type Balls. Sobolev-Type balls are sets of functions that satisfy smoothness constraints on higher-order derivatives (Barron, 1993; Jones, 1992; Pinkus, 1985; Wahba, 1990; Blanchard & Bennouna, 2022a). One such constraint is that the magnitude of the function's Fourier transform, $|\mathcal{F}f(\omega)|$, decays fast enough as $|\omega|$ approaches infinity. These constraints are imposed by comparing the magnitude of the Fourier transform of the function, $\mathcal{F}f(\omega)$, with the magnitude of $|\omega|^r$ for some number $r > 0$. In this paper we define a generalized form of Sobolev-type balls:

$$\mathcal{B}_{r,\rho} \;=\; \left\{f : \mathbb{R}^d \to \mathbb{R}, f \in L_2(\mathbb{R}^d) : \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} |(\mathcal{F}f)(\omega)|\ \mathrm{d}\omega, \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} |\omega|^r |(\mathcal{F}f)(\omega)|^\rho\ \mathrm{d}\omega \leq 1\right\}.$$

In (Barron, 1993), they explored the ability of neural networks to approximate functions in the space $P_1 = \{f \in L_2(\mathbb{R}^d) \mid \int_{\mathbb{R}^d} |\omega||(\mathcal{F}f)(\omega)| < \infty\}$. We now demonstrate that any function in P_1 has a normalized representation in $\mathcal{B}_{1,1}$. For this, we show that the conditions on P_1 allow us to bound $\|\mathcal{F}f\|_{L_1(\mathbb{R}^d)}$. Namely,

$$\begin{aligned}\int_{\mathbb{R}^d} |(\mathcal{F}f)(\omega)| \, \mathrm{d}\omega &= \int_{[-1,1]^d} |(\mathcal{F}f)(\omega)| \, \mathrm{d}\omega + \int_{\mathbb{R}^d \setminus [-1,1]^d} |(\mathcal{F}f)(\omega)| \, \mathrm{d}\omega \\ &\leq C_1 \left(\int_{[-1,1]^d} |(\mathcal{F}f)(\omega)|^2 \, \mathrm{d}\omega \right)^{1/2} + \int_{\mathbb{R}^d \setminus [-1,1]^d} |\omega||(\mathcal{F}f)(\omega)| \, \mathrm{d}\omega \\ &\leq C_1 \|f\|_2 + \int_{\mathbb{R}^d} |\omega||(\mathcal{F}f)(\omega)| \, \mathrm{d}\omega,\end{aligned}$$

for $C_1 > 0$. Therefore, any function in the space P_1 can be scaled to a function in $\mathcal{B}_{1,1}$. Specifically, they showed that sigmoidal neural networks with a bounded depth and $\mathcal{O}(d\epsilon^{-2})$ neurons can approximate such functions with error at most ϵ.

In this paper we will focus on the space $\mathcal{B}_{2r,2}$. In Lemma 2 (Appendix C) we show that this space is embedded as a proper subspace of $\mathcal{W}^{r,2}$, with the following norm:

$$\|f\|_{\mathcal{B}_{2r,2}} = \left(\frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} (1 + |\omega|^{2r}) |(\mathcal{F}f)(\omega)|^2 \, \mathrm{d}\omega \right)^{1/2}. \tag{1}$$

It is worth mentioning that Pinkus (1985) showed that using traditional basis functions, functions of the space $P_2 = \{f \in L_2(\mathbb{R}^d) \mid \int_{\mathbb{R}^d} |\omega|^{2r} |(\mathcal{F}f)(\omega)|^2 < \infty\}$ may be approximated with error at most ϵ using $\mathcal{O}(\epsilon^{-d/2r})$ parameters. In this paper, we show that by enforcing an additional constraint on the L_1 norm of the Fourier coefficients, we can circumvent exponential dependence on the dimension d.

2.2 Neural Network Architectures

In the previous section, we described a setting in which a class of candidate functions $\mathcal{H}$ serve as approximators to a class of target functions $\mathcal{U}$. In this work, we compare the approximation guarantees of standard multilayer perceptrons and a generic set of neural networks that incorporate multiplication operations. Our goal is to understand whether multiplication layers can provide better guarantees to approximate bandlimited functions and members of $\mathcal{B}_{2r,2}$.

Multilayer perceptrons. A multilayer perceptron is a type of neural network architecture that consists of L layers of linear transformations interspersed with element-wise non-linear activation functions (e.g., the ReLU function). Typically, the last layer does not include a non-linear activation.

Definition 1 (Multilayer perceptron)**.** *A multilayer perceptron is a function $f = y_{L,1} : \mathbb{R}^{p_0} \to \mathbb{R}$ defined by a set of univariate functions $\bigcup_{i=0}^{L} \{y_{i,j}\}_{j=1}^{p_i}$. Each function $y_{i,j} : \mathbb{R}^{p_0} \to \mathbb{R}$ (also known as a neuron) is recursively computed in the following manner*

$$\begin{aligned} y_{L,j}(x) &= \langle w_{L,j}, y_{L-1}(x) \rangle + b_{L,j} \\ y_{i,j}(x) &= \sigma(\langle w_{i,j}, y_{i-1}(x) \rangle + b_{i,j}) \\ y_{0,j}(x) &= x_j, \end{aligned}$$

where $i \in [L-1]$, $j \in [p_i]$, and $w_{i,j} \in \mathbb{R}^{p_{i-1}}$ and $b_{i,j} \in \mathbb{R}$ are the weights and a bias of the neuron $y_{i,j}$ and $y_i = (y_{i,1}, \ldots, y_{i,p_i})$. The function $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function.

In this work, we focus on neural networks with ReLU activations, which are defined as $\sigma(x) = \max(0, x)$. However, it is worth noting that other activation functions have been proposed in the literature, such as sigmoidal functions that are measurable functions η that satisfy $\eta(x) \to 0$ as $x \to -\infty$ and $\eta(x) \to 1$ as $x \to \infty$.

Multiplicative neural networks. In this work, we are interested in comparing the approximation abilities of standard ReLU networks, with that of neural networks that incorporate multiplication layers (also known as

product units (Durbin & Rumelhart, 1989)). In order to fully understand the added benefits of multiplication gates, we ask the following question: *Are multiplications between neurons sufficient to substitute the non-linear activations in multilayer perceptrons?*

Definition 2 (Multiplicative network)**.** *A multiplicative neural network is a function* $f = y_{L,1} : \mathbb{R}^{p_0} \to \mathbb{R}$ *defined by a set of univariate functions* $\bigcup_{i=1}^{L}\{y_{i,j}\}_{j=1}^{p_i}$*. Each function* $y_{i,j} : \mathbb{R}^{p_0} \to \mathbb{R}$ *(also known as a neuron) is recursively computed in the following manner*

$$\begin{aligned} y_{i,j}(x) &= \langle w_{i,j}, y_{i-1}(x)\rangle + a_{i,j}y_{i-1,j_1}(x)y_{i-1,j_2}(x) + b_{i,j} \\ y_{0,j}(x) &= x_j, \end{aligned}$$

where $a_{i,j}, b_{i,j} \in \mathbb{R}$*,* $w_{i,j} \in \mathbb{R}^{p_{i-1}}$ *are trainable parameters and* $y_i = (y_1, \ldots, y_{p_i})$*.*

Multiplicative networks differ from multilayer perceptrons as they rely on (non-linear) multiplicative gates between neurons instead of element-wise activation functions, such as ReLU or sigmoid, to model complex functions. The total number of trainable parameters in a multiplicative layer is $p_{i+1}p_i + 2p_i$, which is p_i more parameters than a fully-connected layer. Following Yarotsky (2017); Blanchard & Bennouna (2022a) we measure the complexity of a neural network using its depth L and the number of neurons $G = \sum_{i=1}^{L-1} p_i$ it includes.

Self-Attention and multiplicative layers. Let us describe a single-headed self-attention operation in the original Transformer (Vaswani et al., 2017). Each layer $i \in [L]$ of a depth-L Transformer encoder is defined as follows. The input to the ith layer is a sequence of N tokens, denoted by $x_i = \{x_{i,j}\}_{j=1}^{N}$, where each $x_{i,j} \in \mathbb{R}^{d_x}$ represents the jth token of the ith layer. To compute the output of the ith layer at a particular position $e \in [N]$, we use the following formula:

$$f_{\mathrm{SA}}^{i,e}(x_i) = \sum_{j=1}^{N} \mathrm{softmax}_j\left(\tfrac{1}{\sqrt{d_a}}\langle W^{Q,i}x_{i,e}, W^{K,i}x_{i,e}\rangle\right) W^{V,i}x_{i,j},$$

where $\mathrm{softmax}_j(f(x)) = \exp(f(x)_j)/\sum_{j'}\exp(f(x)_{j'})$ is the softmax operator, and the trainable weight matrices $W^{K,i}$, $W^{Q,i}$, $W^{V,i} \in \mathbb{R}^{d_a \times d_x}$ convert the representation from its dimension d_x into the attention dimension $d_a = d_x$, creating 'Key', 'Query', and 'Value' representations, resp. As can be seen, the self-attention layers use multiplicative connections when computing the following inner product $\langle W^{Q,i}x_{i,e}, W^{K,i}x_{i,e}\rangle$. This operation computes multiplications between the coordinates of transformations of the same token $x_{i,e}$. In other words, it can be thought of as computing a multiplicative layer on an input vector $x_{i,e}$. As a side note, in addition to self-attention layers, transformers also incorporate commonly used layers such as fully-connected layers, residual connections, and normalization layers, which are not the focus of this paper.

3 Representation Power of Multiplicative Neural Networks

In this section, we explore the expressive power of neural networks with multiplication layers. We first demonstrate that these networks can easily represent polynomial functions, which we then use to approximate bandlimited functions. This allows us to approximate functions in the space $\mathcal{B}_{2r,2}$ without suffering from the curse of dimensionality. Specifically, we prove the following lemma:

Lemma 1. *For any polynomial* $p_n : \mathbb{R} \to \mathbb{R}$ *of degree* n *of the form* $p_n(x) = \sum_{k=0}^{n} c_k x^k$*, there exists a multiplicative neural network* $f_n^{\mathrm{POL}} : \mathbb{R}^3 \to \mathbb{R}$*, of depth* $L_n = \mathcal{O}(n)$ *with* $G_n = \mathcal{O}(n)$ *neurons that satisfies* $f_n^{\mathrm{POL}}(x, x, c_0) = p_n(x)$ *for* $c_0 \in \mathbb{R}$*, and* $x \in \mathbb{R}$*.*

With this lemma in hand, we can show how to use multiplicative networks to approximate analytic kernel functions $K : \mathbb{R} \to \mathbb{C}$ by leveraging their ability to represent polynomials. By approximating a certain class of polynomials, we can demonstrate how to use these networks to approximate analytic kernels. Once we have the ability to approximate analytic kernels, we can use Maurey's Theorem (see Lemma 3) to express any function $f \in \mathcal{H}_M$ as a finite sum of kernel superpositions, and construct a network that approximates this sum.

Approximating of real-valued analytic functions. In this section, we show how to approximate certain real-valued analytic functions. For this purpose, we recall the notion of the Bernstein s-ellipse, which is a geometric shape defined on the complex plane that is useful in approximation theory.

Definition 3. *Let $M \geq 1$, $s > 1$ be two scalars. The Bernstein s-ellipse on $[-M, M]$ is defined as follows*

$$E_s^M \;=\; \Big\{x + iy \in \mathbb{C} : \tfrac{x^2}{(a_s^M)^2} + \tfrac{y^2}{(b_s^M)^2} = 1\Big\},$$

whose semi-major and semi-minor axes are $a_s^M = M\frac{s+s^{-1}}{2}$ and $b_s^M = M\frac{s-s^{-1}}{2}$.

The parameter s controls the shape of the ellipse, and the parameter M determines its size. For example, when $s = 2$, the ellipse is a circle centered at the origin with a radius M. As s increases, the ellipse becomes more elongated and its semi-minor axis decreases. The semi-major and semi-minor axes of the ellipse determine its maximal and minimal values, respectively. Before stating our result in Theorem 2, we recall the following theorem of Trefethen (2019):

Theorem 1 (Theorem 8.2 of Trefethen (2019))**.** *Let $M > 0, s > 2$ be scalars and $K : [-M, M] \to \mathbb{R}$ be an analytic function that is analytically continuable to the ellipse E_s^M, where it satisfies $\sup_{x\in E_s^M} |K(x)| \leq C_K$ for some constant $C_K > 0$. For every $n \in \mathbb{N}$, there exists a polynomial $h_n : \mathbb{R} \to \mathbb{C}$ of degree n, such that,*

$$\|h_n - K\|_{L^\infty([-M,M])} \;\leq\; \frac{2C_K s^{-n}}{s-1}.$$

Theorem 1 states that any analytic function K that is bounded on the Bernstein s-ellipse E_s^M can be accurately approximated by a polynomial h_n of degree n, with the error decreasing exponentially as the degree n increases.

As we show next, the use of multiplication layers in neural networks can improve the efficiency of function approximation in certain cases. The following theorem shows that deep multiplicative networks can approximate real-valued analytic functions on bounded intervals.

Theorem 2. *Let $M \geq 1$, $s > 2$, $C_K > 0$ and $\epsilon \in (0, 1)$ be scalars. Then, for any real-valued analytic function $K : [-M, M] \to \mathbb{R}$ that is analytically continuable to the ellipse E_s^M where $|K(x)| \leq C_K$, there exists a deep multiplicative network $f^{\mathrm{MA}} : [-M, M]^3 \to \mathbb{R}$ (MA stands for 'Multiplicative Analytic') of depth $L_\epsilon = \mathcal{O}(\frac{1}{\log_2 s} \log_2 \frac{C_K}{\epsilon})$ with $G_\epsilon = \mathcal{O}(\frac{1}{\log_2 s} \log_2 \frac{C_K}{\epsilon})$ neurons that satisfies*

$$\|f^{\mathrm{MA}}(x, x, x) - K(x)\|_{L^\infty([-M,M])} \;\leq\; \epsilon.$$

Theorem 2 establishes that deep multiplicative networks with second-degree multiplications can approximate any real-valued analytic function that is bounded on the Bernstein s-ellipse E_s^M with error bounded by a quantity that decreases exponentially with the depth of the network.

In (Montanelli et al., 2021), the authors show that the kernel may be approximated with depth $L_\epsilon = \mathcal{O}(\frac{1}{\log_2^2 s} \log_2^2 \frac{C_K}{\epsilon})$ with $G_\epsilon = \mathcal{O}(\frac{1}{\epsilon^2 \log_2^2 s} \log_2^2 \frac{C_K}{\epsilon})$ neurons. In contrast, our approach achieves a $\mathcal{O}(\log(1/\epsilon))$ improvement in depth and a $\mathcal{O}(\log(1/\epsilon^2))$ in number of neurons, resulting in an exponentially faster convergence to the target function.

Approximation of bandlimited functions. As a next step, we study the ability of neural networks to approximate bandlimited functions. We start by showing how bandlimited functions can be approximated using analytic kernels. For this purpose, we will use Maurey's theorem (Pisier, 1980-1981; Vitali D. Milman, 1986) that states that for Hilbert subspaces with a bounded norm, any function in the convex hull can be easily approximated using points from the subspace, where the rate of approximation is dependent on the number of points used.

The following theorem shows that a deep multiplicative network can approximate in $B = [0, 1]^d$, a bandlimited function up to a given error tolerance using a relatively small number of neurons and depth.

Theorem 3. *Let $\epsilon \in (0, 1)$, $M > 1, d \geq 2$, and $K : \mathbb{R} \to \mathbb{C}$ be an analytic kernel that holds the assumptions of Theorem 2 with respect to $s > 2, C_K > 0$, and bounded by a constant $D_K \in (0, 1]$ on $[-dM, dM]$. Let f*

be a real-valued function in $\mathcal{H}_{K,M}(B)$. Further, let $F : [-M, M]^d \to \mathbb{C}$ be a square-integrable function such that $f(x) = \int_{[-M,M]^d} F(\omega)K(\omega \cdot x)\ \mathrm{d}\omega$. We define $C_F = \int_{\mathbb{R}^d} |F(\omega)|\ \mathrm{d}\omega = \int_{[-M,M]^d} |F(\omega)|\ \mathrm{d}\omega$. Then, there exists a deep multiplicative network $f^{\mathrm{MBL}} : B \to \mathbb{R}$ (MBL stands for 'Multiplicative bandlimited') of depth $L_\epsilon = \mathcal{O}\Big(\frac{1}{\log_2 s} \log_2 \frac{C_F C_K}{\epsilon}\Big)$ with $G_\epsilon = \mathcal{O}\Big(\frac{C_F^2}{\epsilon^2 \log_2 s} \log_2 \frac{C_F C_K}{\epsilon}\Big)$ neurons that satisfies $\|f^{\mathrm{MBL}} - f\|_{L^2(B)} \leq \epsilon$.

The above theorem shows that one can approximate bandlimited functions up to error ϵ using multiplicative neural networks of depth $L_\epsilon = \mathcal{O}(\log(1/\epsilon))$ using $G_\epsilon = \mathcal{O}\big(\epsilon^{-2} \log(1/\epsilon)\big)$ neurons. In comparison, Montanelli et al. (2021) showed that one can approximate bandlimited functions to the same level of approximation using standard ReLU networks of depths $L_\epsilon = \mathcal{O}\big(\log^2(1/\epsilon)\big)$ with $G_\epsilon = \mathcal{O}\big(\epsilon^{-2} \log^2(1/\epsilon)\big)$ neurons. This result demonstrates the inherent parameter efficiency of multiplicative neural networks in comparison with standard ReLU networks.

Approximation of Smooth Functions. We now turn to show results for Sobolev-Type functions. We use the results on bandlimited functions shown in Theorem 3 to approximate functions in $\mathcal{B}_{2r,2} \subsetneq \mathcal{W}^{r,2}$. We show that using slightly stronger assumptions, we get an approximation rate comparable to those shown by Barron (1993) (where the network is a shallow sigmodial network model) using multiplicative neural networks (i.e. without non-linear activations). Further, since these are in fact in $\mathcal{W}^{r,2}$, we may better characterize such functions.

Theorem 4. *Let $d \geq 2, r \in \mathbb{N}$, $f \in \mathcal{B}_{2r,2}$ and $\epsilon > 0$. There exists a deep ReLU network f^{RS} (standing for "ReLU Sobolev") with a depth of $L_\epsilon = \mathcal{O}(d^2 \epsilon^{-2/r})$ and $G_\epsilon = \mathcal{O}(d^2 \epsilon^{-(2+2/r)})$ neurons, such that $\big\|f^{\mathrm{RS}} - f\big\|_{L_2(B)} \leq \epsilon$.*

Theorem 5. *Let $d \geq 2, r \in \mathbb{N}$, $f \in \mathcal{B}_{2r,2}$ and $\epsilon > 0$. There exists a deep ReLU network f^{MS} (standing for "Multiplicative Sobolev") with a depth of $L_\epsilon = \mathcal{O}(d\epsilon^{-1/r})$ and $G_\epsilon = \mathcal{O}(d\epsilon^{-(2+(1/r))})$ neurons, such that $\big\|f^{\mathrm{MS}} - f\big\|_{L_2(B)} \leq \epsilon$.*

Proof. Let $f \in \mathcal{B}_{2r,2}$. We would like to approximate f using a bandlimited function f_M and then approximate f_M using a multiplicative neural network $f^{\mathrm{BL}} = f^{\mathrm{MS}}$. Let $M > 1$ be a band.

We recall the Inverse Fourier transform given by $(\mathcal{F}^{-1}g)(x) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} g(\omega) \exp(i\omega \cdot x)\ \mathrm{d}\omega$. We define the bandlimiting of $f : \mathbb{R}^d \to \mathbb{R}$ as $f_M = \mathcal{F}^{-1}(\mathcal{F}f \mathbb{1}_{[-M,M]^d})$, such that $f_M \in \mathcal{H}_{K,M}(B)$ for $K(x) = \exp(ix)$ and $F = \frac{1}{(2\pi)^d} \mathcal{F}f$. We have

$$\begin{aligned} \|f - f_M\|_{L_2(B)} &\leq \left(\frac{1}{(2\pi)^d} \big\|\mathcal{F}f - \mathcal{F}f\mathbb{1}_{[-M,M]^d}\big\|^2_{L_2(\mathbb{R}^d)}\right)^{1/2} \\ &= \left(\frac{1}{(2\pi)^d} \int_{\mathbb{R}^d \setminus [-M,M]^d} |\mathcal{F}f(\omega)|^2\ \mathrm{d}\omega\right)^{1/2}. \end{aligned}$$

For any $\omega \in \mathbb{R}^d \setminus [-M, M]^d$, we have $|M^{-1}\omega|^{2r} \geq 1$. Therefore,

$$\begin{aligned} \left(\frac{1}{(2\pi)^d} \int_{\mathbb{R}^d \setminus [-M,M]^d} |\mathcal{F}f(\omega)|^2\ \mathrm{d}\omega\right)^{1/2} &\leq \left(\frac{1}{(2\pi)^d} \int_{\mathbb{R}^d \setminus [-M,M]^d} \big|M^{-1}\omega\big|^{2r} |\mathcal{F}f(\omega)|^2\ \mathrm{d}\omega\right)^{1/2} \\ &\leq M^{-r} \left(\frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\omega|^{2r} |\mathcal{F}f(\omega)|^2\ \mathrm{d}\omega\right)^{1/2} \\ &\leq M^{-r}, \end{aligned}$$

where the final inequality is due to $f \in \mathcal{B}_{2r,2}$. For any $\epsilon > 0$, we set $M = (2/\epsilon)^{1/r}$. We will construct a neural network f^{MS} to approximate the bandlimited function f_M, such that

$$\|f_M - f^{\mathrm{MS}}\|_{L_2(B)} \leq \epsilon/2.$$

Assuming we have constructed such f^{MS}, then by the triangle inequality we then arrive at

$$\|f - f^{\text{MS}}\|_{L_2(B)} \;\leq\; \|f - f_M\|_{L_2(B)} + \|f_M - f^{\text{MS}}\|_{L_2(B)} \;\leq\; M^{-r} + \epsilon/2 \;\leq\; \epsilon.$$

We define a function $F : [-M, M]^d \to \mathbb{C}$ as follows

$$F(\omega) \;=\; \frac{1}{(2\pi)^d}(\mathcal{F}f)(\omega).$$

We then have the following identity

$$f_M(x) \;=\; \int_{[-M,M]^d} F(\omega)K(\omega \cdot x)\;\mathrm{d}\omega \;=\; \int_{[-M,M]^d} \tfrac{1}{(2\pi)^d}\mathcal{F}f(\omega)\exp(i\omega \cdot x)\;\mathrm{d}\omega.$$

We may now work under the conditions of Theorem 3. We consider that $C_F = \int_{[-M,M]^d}|F(\omega)|\;\mathrm{d}\omega = \frac{1}{(2\pi)^d}\int_{[-M,M]^d}|\mathcal{F}f(\omega)|\;\mathrm{d}\omega \leq 1$, where the inequality is due to the definition of $\mathcal{B}_{2r,2}$. The kernel $K(t) = \exp(it)$ takes as input $t = \omega \cdot x$, for $x \in B$ and $\omega \in [-M, M]^d$. Therefore, $t \in [-dM, dM]$, and $K : [-dM, dM] \to \mathbb{R}$. We note that K is continuable to the Bernstein 4-ellipse E_4^{dM}. We notice that $a_4^{dM} = dM(4 + 4^{-1})/2$ is the larger axis, and therefore the maximal norm of K on E_s^{dM} is given by $K(a_4^{dM})$:

$$C_K \;=\; \max_t|K(t)| \;\leq\; \exp(dM\tfrac{4+4^{-1}}{2}).$$

Further, for any $t \in \mathbb{R}$ we have $|K(t)| \leq 1 = D_K$.

We now approximate f_M with a multiplicative network. Using the results of Theorem 3, there exists a deep multiplicative neural network f^{MS} that approximates the bandlimited function f_M in $L_2(B)$ with error bounded by $\epsilon/2$ and depth

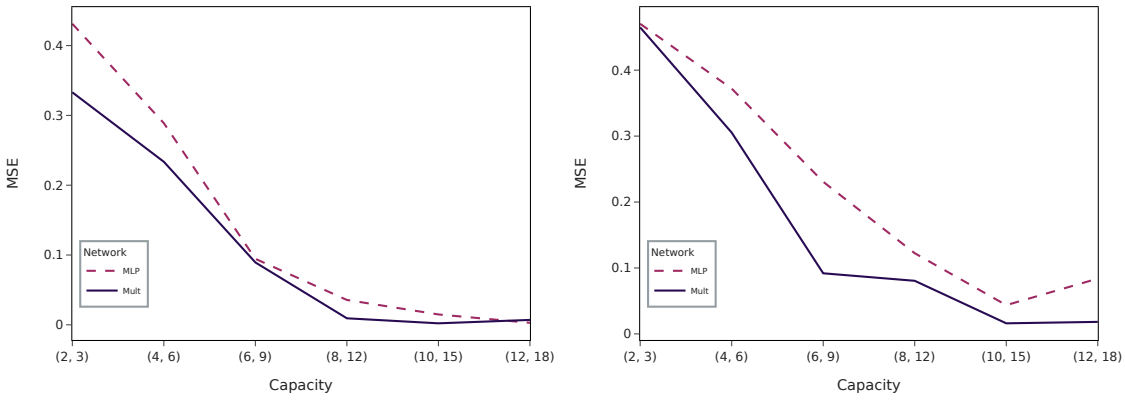
$$\begin{aligned} L_\epsilon \;&\leq\; C_1\frac{1}{\log_2 4}\log_2\frac{2C_KC_F}{\epsilon} \\ &\leq\; C_2\frac{1}{\log_2 4}\log_2\frac{\exp(dM\frac{4+4^{-1}}{2})}{\epsilon} \\ &\leq\; C_2\Big(3d\epsilon^{-1/r} + \log_2(1/\epsilon)\Big) \\ &\leq\; C_3 \cdot d\epsilon^{-1/r}, \end{aligned} \tag{2}$$

for some constants $C_1, C_2, C_3 > 0$. In addition, the number of neurons can be bounded by

$$\begin{aligned} G_\epsilon \;&\leq\; C_1'C_F^2 \cdot \epsilon^{-2}\log_2\frac{2C_KC_F}{\epsilon} \\ &\leq\; C_2' \cdot \epsilon^{-2}\log_2\frac{\exp(dM\frac{4+4^{-1}}{2})}{\epsilon} \\ &\leq\; C_2' \cdot \epsilon^{-2}(3d(2/\epsilon)^{1/r} + \log_2(1/\epsilon)) \\ &\leq\; C_3' \cdot d\epsilon^{-(2+1/r)}, \end{aligned} \tag{3}$$

for some constants $C_1', C_2', C_3' > 0$. □

Theorems 4-5 provide insights into several interesting properties of the Sobolev-Type ball $\mathcal{B}_{2r,2}$. First, we see that approximating these functions does not suffer from the curse of dimensionality that occurs when approximating the full Sobolev space. Secondly, for both ReLU networks and multiplicative networks, the bound becomes tighter as the smoothness r increases, which is a desirable property for approximation error bounds. In fact, as r approaches infinity, the bound approaches the one proposed in (Barron, 1993). Lastly, we show that for the same error tolerance ϵ, multiplicative neural networks can approximate a target function$f \in \mathcal{B}_{2r,2}$ with a depth of $\mathcal{O}(d\epsilon^{-1/r})$ and $\mathcal{O}(d\epsilon^{-(2+1/r)})$ neurons, while standard ReLU neural networks require a depth of $\mathcal{O}(d^2\epsilon^{-2/r})$ and $\mathcal{O}(d^2\epsilon^{-(2+2/r)})$ neurons. This result demonstrates that multiplicative neural networks have stronger approximation guarantees when approximating functions in the Sobolev-Type space.



T_{12} T_{15}

Figure 1: **Approximating Chebychev polynomials with networks of increasing capacities**. We present the minimal test mean squared error (MSE) loss for approximating Chebychev polynomials T_n using networks with increasing capacity, utilizing 10 initialization seeds. The network capacity (x-axis) is defined as $(\text{depth}, \text{width})$ and is increased linearly.

4 Experiments

In previous sections, we demonstrated that multiplicative networks have lower capacity requirements for approximating certain target functions when compared to multilayer perceptrons. To validate these results empirically, we analyzed the influence of network capacity such as depth and width on the network's approximation error. To do so, we approximated Chebyshev polynomials of varying degrees T_n using multilayer perceptrons and multiplicative networks with increasing capacities (i.e., depth, width).

As the degree of the Chebyshev polynomials increased, we expected multilayer perceptrons to struggle in approximating high-degree polynomials. Conversely, as proven in Lemma 1, T_n can be perfectly constructed using a multiplicative network with depth $L_n = n$ and width $G_n = 3n$.

In Figure 1, we approximated T_{12} and T_{15} using networks whose depth and width varied simultaneously. The models were trained with the Adam optimizer and a learning rate of 0.001 for 100 epochs using a batch size of 64 to minimize the Mean Squared Error (MSE) loss. We used 1000 training and 1000 test samples that were randomly and uniformly selected from $[0, 1]$. In each plot, the minimum test MSE losses of each model are reported based on 10 different initialization seeds. As shown, multiplicative networks achieved better results (lower errors) compared to multilayer perceptrons with the same depth and width.

In Figure 2, we approximated T_n of degrees $n = 9, 11, 13$ using multilayer perceptrons and multiplicative networks with depth n and width 10. We used the same training and testing processes as in the previous experiment. As can be seen, multiplicative networks converged to a zero loss faster and provided better approximation rates for the given polynomials. For the code for generating the experiments, see Appendix B.

5 Conclusions

Previous papers have studied the approximation guarantees of standard fully-connected neural networks to approximate functions in the Barron space $\mathcal{B}_{1,1}$ (Barron, 1993), the space of bandlimited functions (Montanelli et al., 2021), and the Korobov space (Blanchard & Bennouna, 2022a). These studies have shown that fully-connected networks can approximate a wide range of smooth functions without suffering from the curse of dimensionality and have provided insights into the tradeoffs between the width and depth of neural networks

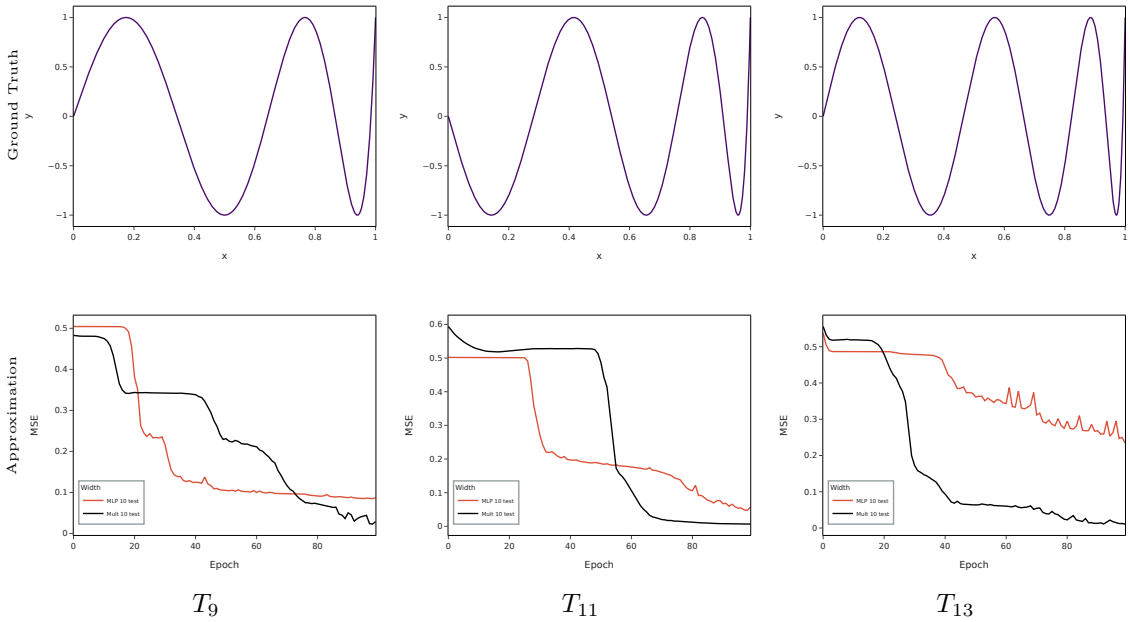


Figure 2: **Approximating Chebychev polynomials.** We present the minimal test mean squared error (MSE) loss for approximating Chebychev polynomials T_n, using networks of depth n and width 10, with 10 initialization seeds. The first row displays the actual polynomials, and the second row shows the approximation errors using both multilayer perceptrons and multiplicative networks.

in learning certain types of functions. However, these results are limited to variants of fully-connected network and do not provide information about other types of architectures.

In this paper, we extend these results by exploring the approximation guarantees of both multiplicative neural networks and standard fully-connected networks to approximate bandlimited functions and members of the Sobolev-Type ball $\mathcal{B}_{2r,2}$. Our results show that multiplicative neural networks achieve stronger approximation guarantees compared to standard ReLU networks. In addition, we show that, unlike the Barron space and the space of bandlimited functions, $\mathcal{B}_{2r,2}$ is a subset of the Sobolev space $\mathcal{W}^{r,2}$. Therefore, our results demonstrate that it is possible to avoid the curse of dimensionality for wide subsets of the Sobolev space.

References

David W. Albrecht, Xuan Thinh Duong, and Alan Mcintosh. Operator theory and harmonic analysis. *Springer Proceedings in Mathematics & Statistics*, 2021.

Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar: an evolutionary computation perspective. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019.

A. R. Barron. Approximation and estimation bounds for artificial neural networks. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, COLT '91, pp. 243–249, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1558602135.

Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inf. Theory*, 39:930–945, 1993.

Raphael Bensadoun, Shir Gur, Tomer Galanti, and Lior Wolf. Meta internal learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 20645–20656. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/ac796a52db3f16bbdb6557d3d89d1c5a-Paper.pdf`.

Moise Blanchard and Mohammed Amine Bennouna. Shallow and deep networks are near-optimal approximators of korobov functions. In *ICLR*, 2022a.

Moise Blanchard and Mohammed Amine Bennouna. Shallow and deep networks are near-optimal approximators of korobov functions. In *International Conference on Learning Representations*, 2022b. URL `https://openreview.net/forum?id=AV8FPoMTTa`.

Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of Data Science*. Cambridge University Press, 2020. doi: 10.1017/9781108755528.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Conference on Empirical Methods in Natural Language Processing*, 2016.

George V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, jun 2019.

Ronald A. DeVore and George G. Lorentz. Constructive approximation. In *Grundlehren der mathematischen Wissenschaften*, 1993.

Ronald A. DeVore, Ralph Howard, and Charles A. Micchelli. Optimal nonlinear approximation. *manuscripta mathematica*, 63:469–478, 1989.

Luc Devroye, László Györfi, and Gábor Lugosi. A probabilistic theory of pattern recognition. In *Stochastic Modelling and Applied Probability*, 1996.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Richard Durbin and David E. Rumelhart. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Comput.*, 1(1):133–142, mar 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.1.133. URL https://doi.org/10.1162/neco.1989.1.1.133.

Weinan E, Chao Ma, and Lei Wu. The barron space and the flow-induced function spaces for neural network models. *Constructive Approximation*, 55:369–406, 2021.

Moshe Eliasof, Eldad Haber, and Eran Treister. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. In *Neural Information Processing Systems*, 2021.

William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. In *Advances in Neural Information Processing Systems 33*. Curran Associates, Inc., 2020.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1): 014004, January 2018. doi: 10.1088/1361-6420/aa9a90.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pp. 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

Ken ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

Masaaki Imaizumi and Kenji Fukumizu. Deep neural networks learn non-smooth functions effectively. In *International Conference on Artificial Intelligence and Statistics*, 2018.

Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative interactions and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rylnK6VtDH.

Lee K. Jones. A simple lemma on greedy approximation in hilbert space and convergence rates for projection pursuit regression and neural network training. *Annals of Statistics*, 20:608–613, 1992.

Jason M. Klusowski and Andrew R. Barron. Risk bounds for high-dimensional ridge function combinations including neural networks. *arXiv: Statistics Theory*, 2016.

Michael Kohler and Adam Krzyżak. Nonparametric regression based on hierarchical interaction models. *IEEE Transactions on Information Theory*, 63(3):1620–1630, 2017. doi: 10.1109/TIT.2016.2634401.

David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a non-polynomial activation function can approximate any function. *New York University Stern School of Business Research Paper Series*, 1991.

Yoav Levine, Noam Wies, Or Sharir, Hofit Bata, and Amnon Shashua. Limits to depth efficiencies of self-attention. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22640–22651. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/ff4dfdf5904e920ce52b48c1cef97829-Paper.pdf.

Shiyu Liang and Rayadurgam Srikant. Why deep neural networks for function approximation? In *ICLR*, 2017.

Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

H.N. Mhaskar. Eignets for function approximation on manifolds. *Applied and Computational Harmonic Analysis*, 29(1):63–87, 2010. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2009.08.006. URL https://www.sciencedirect.com/science/article/pii/S106352030900089X.

Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. When and why are deep networks better than shallow ones? In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pp. 2343–2349. AAAI Press, 2017.

Hrushikesh Narhar Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, 8:164–177, 1996.

Hadrien Montanelli and Qiang Du. New error bounds for deep relu networks using sparse grids. *SIAM Journal on Mathematics of Data Science*, 1(1):78–92, 2019. doi: 10.1137/18M1189336. URL https://doi.org/10.1137/18M1189336.

Hadrien Montanelli, Haizhao Yang, and Qiang Du. Deep ReLU networks overcome the curse of dimensionality for bandlimited functions. *Journal of Computational Mathematics*, 39:801–815, 2021.

Ryumei Nakada and Masaaki Imaizumi. Adaptive approximation and generalization of deep neural network with intrinsic dimensionality. *J. Mach. Learn. Res.*, 21(1), jun 2022. ISSN 1532-4435.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2249–2255, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1244. URL https://aclanthology.org/D16-1244.

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 *Advances in Neural Information Processing Systems 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.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization, 2017. URL https://arxiv.org/abs/1705.04304.

Philipp Christian Petersen and Felix Voigtländer. Optimal approximation of piecewise smooth functions using deep relu neural networks. *Neural networks : the official journal of the International Neural Network Society*, 108:296–330, 2017.

Allan Pinkus. *N-Widths in Approximation Theory*. Springer-Verlag, 1985.

Gilles Pisier. Remarks on an unpublished result of B. Maurey, 1980-1981. URL http://www.numdam.org/item/SAF_1980-1981____A5_0/.

Tomaso Poggio, Andrzej Banburski, and Qianli Liao. Theoretical issues in deep networks. *Proceedings of the National Academy of Sciences*, 117(48):30039–30045, 2020. doi: 10.1073/pnas.1907369117. URL https://www.pnas.org/doi/abs/10.1073/pnas.1907369117.

John J. F. Fournier Robert A. Adams. *Sobolev spaces*. Pure and Applied Mathematics. Academic Press, 2 edition, 2003. ISBN 9780120441433,9781435608108,0120441438. URL http://gen.lib.rus.ec/book/index.php?md5=72db1ed30d242bd3ebcb73bcd099a6b4.

Johannes Schmidt-Hieber. Deep relu network approximation of functions on a manifold, 2019. URL https://arxiv.org/abs/1908.00695.

Claude E. Shannon. Communication in the presence of noise. *Proceedings of the IEEE*, 72:1192–1201, 1984.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. ISSN 0028-0836. doi: 10.1038/nature16961.

Andrew Trask, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units, 2018. URL https://arxiv.org/abs/1808.00508.

Lloyd N. Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2019. doi: 10.1137/1.9781611975949. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611975949.

Sebastian Urban and Patrick van der Smagt. A neural transfer function for a smooth and differentiable transition between additive and multiplicative interactions, 2015. URL https://arxiv.org/abs/1503.05724.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Gideon Schechtman Vitali D. Milman. *The Maurey-Pisier Theorem*, pp. 85–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 1986. ISBN 978-3-540-38822-7. doi: 10.1007/978-3-540-38822-7_13. URL https://doi.org/10.1007/978-3-540-38822-7_13.

Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020.

Grace Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990. doi: 10.1137/1.9781611970128. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611970128.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkVhlh09tX.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural networks : the official journal of the International Neural Network Society*, 94:103–114, 2017.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1204–1213, 2021.

A Examples

Example 1. *Let $f \in L_2(\mathbb{R}^d)$ such that f is Bandlimited function with band π (i.e., $\text{supp}(\mathcal{F}f) \subset [-\pi, \pi]^d$). Let $\phi(x) = \text{sinc}(x) = \sin(\pi x)/\pi x$, and $\phi_k = \phi(\cdot - k)$ for all $k \in \mathbb{Z}^d$. By the Shannon-Nyquist theorem (Shannon, 1984), we have:*

$$f(x) \;=\; \sum_{k\in\mathbb{Z}^d} \langle f, \phi_k\rangle \phi_k(x) \;=\; \sum_{k\in\mathbb{Z}^d} f(k)\phi(x-k).$$

This means that every Bandlimited function can be completely determined using a discrete set of integer samples. This result is particularly surprising for high-dimensional functions (d is large) since the maximal distance between a point x and a sampling point grows with d. For example, the vertex of the unit cube in $\mathbb{R}^d$ is of distance $\sqrt{d}/2$ away from its center. Despite this, we can still recover samples from the integer vertices, even when the distances scale as $\sqrt{d}$. This property is useful when recovering high-dimensional functions using neural networks. See Figure 3 for illustration.

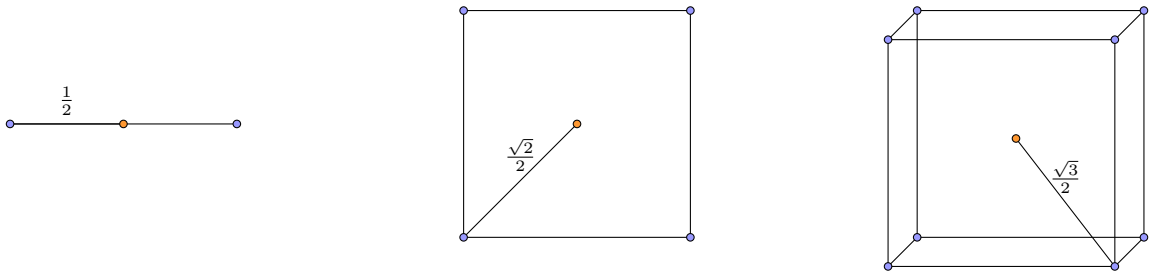


Figure 3: Illustration of Example 1. A bandlimited function f can be reconstructed using a discrete set of values, despite the fact that the distances in each cube grow as $\mathcal{O}(\sqrt{d})$.

B Code

In this section, we provide the code used to generate the experiments. The network architectures are described in Section 2.2. The "chebychev(n, x)" function calculates the Chebyshev polynomial of degree n for a given input x. The "MLPPolynomialApproximation" and "MultPolynomialApproximation" classes are Pytorch Lightning (Falcon et al., 2019) implementations for the multilayer perceptrons and multiplicative networks, respectively. The "MultiplicativeLayer" class is based on the standard linear layer in PyTorch (Paszke et al., 2019). Since the neurons in each layer are interchangeable, we arbitrarily connected the first two neurons in each layer using a multiplicative operation. In each network, the layers are followed by ReLU activations.

```
from torch.nn.parameter import Parameter, UninitializedParameter
from torch.nn import init
from torch.nn import functional as F
import torch
from torch import Tensor
import torch.nn as nn
import torch.optim as optim
import pytorch_lightning as pl
from torch.utils.data import DataLoader, TensorDataset
from tqdm import tqdm
import numpy as np
from sklearn.model_selection import train_test_split

def chebychev(n, x):
    coefs = [0 for _ in range(n)] + [1]
```

```
    return np.polynomial.chebyshev.Chebyshev(coefs)(x)

class MultiplicativeLayer(nn.Module):
    __constants__ = ['in_features', 'out_features']
    in_features: int
    out_features: int
    weight: Tensor

    def __init__(self, in_features: int=3, out_features: int=3, bias: bool = True,
                 device=None, dtype=None) -> None:
        factory_kwargs = {'device': device, 'dtype': dtype}
        super(AttentionLayer, self).__init__()
        self.in_features = in_features
        self.out_features = out_features

        self.weight = Parameter(torch.empty((out_features, in_features), **factory_kwargs)
            )
        self.bias = Parameter(torch.empty(out_features, **factory_kwargs))
        self.mult_weight = Parameter(torch.empty(1, **factory_kwargs))
        self.reset_parameters()

    def reset_parameters(self) -> None:
        init.kaiming_uniform_(self.weight, a=math.sqrt(5))
        if self.bias is not None:
            fan_in, _ = init._calculate_fan_in_and_fan_out(self.weight)
            bound = 1 / math.sqrt(fan_in) if fan_in > 0 else 0
            init.uniform_(self.bias, -bound, bound)
            init.uniform_(self.mult_weight, -bound, bound)

    def forward(self, x: Tensor) -> Tensor:
        bs, _ = x.shape
        x = F.linear(x, self.weight, self.bias)
        mult_output = torch.prod(x.gather(1, torch.tensor([0, 1]).repeat(bs, 1)), dim=1)
        mask = torch.zeros_like(x)
        mask[:, 1] = self.mult_weight*mult_output
        x = x + mask
        return x

class MultPolynomialApproximation(pl.LightningModule):
    def __init__(self, width=3, depth=3, poly_deg=3, use_relu=False):
        super().__init__()
        self.width = width
        self.depth = depth
        self.poly_deg = poly_deg
        self.layers = nn.ModuleList()
        self.layers.append(nn.Linear(1, width))
        self.train_loss = {}
        self.val_loss = {}
        for i in range(self.depth):
            self.layers.append(AttentionLayer(width, width))
        self.output = nn.Linear(width, 1)
        self.use_relu = use_relu

    def forward(self, x):
```

```
        for layer in self.layers:
            x = layer(x)
            if self.use_relu:
                x = torch.relu(x)
        return self.output(x)

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.forward(x)
        loss = torch.mean((y_hat - y) ** 2)
        self.log("train_loss", loss)
        return {"loss": loss}

    def training_epoch_end(self, outputs) -> None:
        loss = sum(output['loss'] for output in outputs) / len(outputs)
        self.train_loss[self.current_epoch] = loss.item()

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.forward(x)
        loss = torch.mean((y_hat - y) ** 2)
        self.log("val_loss", loss, prog_bar=True)
        return {"val_loss": loss}

    def validation_epoch_end(self, outputs) -> None:
        avg_loss = torch.stack([x["val_loss"] for x in outputs]).mean()
        self.val_loss[self.current_epoch] = avg_loss.item()

    def validation_end(self, outputs):
        avg_loss = torch.stack([x["val_loss"] for x in outputs]).mean()
        self.log("val_loss", avg_loss)
        self.val_loss[self.current_epoch] = avg_loss.item()
        return {"val_loss": avg_loss}

    def prepare_data(self, val_split=0.2):
        N = 1000
        X = torch.rand(N, 1)
        Y = chebychev(self.poly_deg, X[:, 0]).view(-1, 1)
        X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size=val_split)
        self.train_dataset = torch.utils.data.TensorDataset(X_train, Y_train)
        self.val_dataset = torch.utils.data.TensorDataset(X_val, Y_val)

    def train_dataloader(self):
        return torch.utils.data.DataLoader(self.train_dataset, batch_size=64, shuffle=True
            )

    def val_dataloader(self):
        return torch.utils.data.DataLoader(self.val_dataset, batch_size=32, shuffle=False)

    def configure_optimizers(self):
        optimizer = optim.Adam(self.parameters(), lr=1e-3)
        return optimizer
```

```
class MLPPolynomialApproximation(pl.LightningModule):
    def __init__(self, width=5, depth=3, poly_deg=3):
        super().__init__()
        self.width = width
        self.depth = depth
        self.poly_deg = poly_deg
        self.layers = nn.ModuleList()
        self.layers.append(nn.Linear(1, width))
        self.train_loss = {}
        self.val_loss = {}
        for i in range(self.depth):
            self.layers.append(nn.Linear(width, width))
        self.output = nn.Linear(width, 1)

    def forward(self, x):
        for layer in self.layers:
            x = torch.relu(layer(x))
        return self.output(x)

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.forward(x)
        loss = torch.mean((y_hat - y) ** 2)
        self.log("train_loss", loss)
        return {"loss": loss}

    def training_epoch_end(self, outputs) -> None:
        loss = sum(output['loss'] for output in outputs) / len(outputs)
        self.train_loss[self.current_epoch] = loss.item()

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.forward(x)
        loss = torch.mean((y_hat - y) ** 2)
        self.log("val_loss", loss, prog_bar=True)
        return {"val_loss": loss}

    def validation_epoch_end(self, outputs) -> None:
        avg_loss = torch.stack([x["val_loss"] for x in outputs]).mean()
        self.val_loss[self.current_epoch] = avg_loss.item()

    def validation_end(self, outputs):
        avg_loss = torch.stack([x["val_loss"] for x in outputs]).mean()
        self.log("val_loss", avg_loss)
        self.val_loss[self.current_epoch] = avg_loss.item()
        return {"val_loss": avg_loss}

    def prepare_data(self, val_split=0.2):
        N = 1000
        X = torch.rand(N, 1)
        Y = chebychev(self.poly_deg, X[:, 0]).view(-1, 1)
        X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size=val_split)
        self.train_dataset = torch.utils.data.TensorDataset(X_train, Y_train)
        self.val_dataset = torch.utils.data.TensorDataset(X_val, Y_val)
```

```
def train_dataloader(self):
    return torch.utils.data.DataLoader(self.train_dataset, batch_size=64, shuffle=True
        )

def val_dataloader(self):
    return torch.utils.data.DataLoader(self.val_dataset, batch_size=32, shuffle=False)

def configure_optimizers(self):
    optimizer = optim.Adam(self.parameters(), lr=1e-3)
    return optimizer
```

C Proofs

Definition 4. *For two Banach spaces V_1 and V_2, we say that V_1 is embedded in V_2 if $\|u\|_{V_2} \leq C\|u\|_{V_1}$ for some constant C and for $u \in V_1$.*

Lemma 2. *Let $r > 0$ and $d \geq 2$. Then, $\mathcal{B}_{2r,2}$ is embedded as a proper subspace of $\mathcal{W}^{r,2}$.*

Proof. Since $f \in \mathcal{B}_{2r,2}$, we have $f \in L_2(\mathbb{R}^d)$, and therefore, $\|f\|_2 < \infty$. Additionally, by Robert A. Adams (2003)(Theorem 5.2), there exists a constant $C_1 > 0$ such that

$$\|f\|_{\mathcal{W}^{r,2}} \;\leq\; C_1(\|f\|_2 + |f|_{r,2}),$$

where C_1 is dependant on r and the dimension d. Therefore, it remains to prove that $|f|_{r,2}$ is bounded. For this, we use some properties of multivariate function spaces.

We begin with the following claim that we will use in advance. Let $\omega \in \mathbb{R}^d, d \geq 2$, we have

$$|\omega|^{2r} \;=\; \|\omega\|_{l_2}^{2r} \;=\; \left(\sum_{m=1}^{d} \omega_m^2\right)^r \;=\; \sum_{\|\alpha\|_1 = r} \binom{r}{\alpha_1, \ldots, \alpha_d} \cdot (\omega^\alpha)^2, \tag{4}$$

where $\omega^\alpha = \prod_{i=1}^d \omega^{\alpha_i}$ for $\alpha \in \mathbb{R}^d$.

By Jensen's inequality,

$$|f|_{r,2}^2 \;=\; \left(\sum_{\alpha:|\alpha|=r} \|D^\alpha f\|_2\right)^2 \;\leq\; d^r \sum_{\alpha:|\alpha|=r} \|D^\alpha f\|_2^2$$

By Parseval's Identity in $\mathbb{R}^d$ (Albrecht et al., 2021):

$$\|D^\alpha f\|_2^2 \;=\; \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\mathcal{F}(D^\alpha f)(\omega)|^2 \,\mathrm{d}\omega \;=\; \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |(\mathcal{F}f)(\omega)|^2 |\omega^\alpha|^2 \,\mathrm{d}\omega.$$

Hence, by (4)

$$\begin{aligned}
\sum_{\alpha:|\alpha|=r} \|D^\alpha f\|_2^2 \;&=\; \sum_{\alpha:|\alpha|=r} \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |(\mathcal{F}f)(\omega)|^2 |\omega^\alpha|^2 \,\mathrm{d}\omega \\
&\leq\; \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |(\mathcal{F}f)(\omega)|^2 \left(\sum_{\alpha:|\alpha|=r} \binom{r}{\alpha_1, \ldots, \alpha_d} \cdot |\omega^\alpha|^2\right) \mathrm{d}\omega \\
&=\; \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\mathcal{F}f(\omega)|^2 |\omega|^{2r} \,\mathrm{d}\omega.
\end{aligned}$$

where the inequality follows from the fact that $\binom{r}{\alpha_1,\ldots,\alpha_d} \geq 1$. Finally, we conclude that

$$\begin{aligned}\|f\|_{\mathcal{W}^{r,2}} \;\leq\; C_1(\|f\|_2 + |f|_{r,2}) \;&\leq\; C_1\Big(\|f\|_2 + \tfrac{d^r}{(2\pi)^{d/2}}\left(\textstyle\int_{\mathbb{R}^d} |(\mathcal{F}f)(\omega)|^2 |\omega|^{2r}\ \mathrm{d}\omega\right)^{1/2}\Big)\\ &\leq\; C_2\|f\|_{\mathcal{B}_{2r,2}},\end{aligned}$$

for some $C_1, C_2 > 0$, where the second inequality is due to the definition of $\|\cdot\|_{\mathcal{B}_{2r,2}}$ as given in equation 1.

To see that $\mathcal{B}_{2r,2}$ is a proper subspace of $\mathcal{W}^{r,2}$, let us define as an example, the function $f \in \mathcal{W}^{r,2}\big(\mathbb{R}^d\big)$ through its Fourier transform:

$$\mathcal{F}f(\omega) = \begin{cases} 1 & |\omega| \leq 1 \\ |\omega|^{-(r+(d+\epsilon)/2)} & |\omega| > 1 \end{cases}$$

for any $\epsilon > 0$.

Indeed, $f \in \mathcal{W}^{r,2}\big(\mathbb{R}^d\big)$ since:

$$\begin{aligned}\int_{\mathbb{R}^d} |\omega|^{2r} |\mathcal{F}f(\omega)|^2\ \mathrm{d}\omega \;&=\; \int_{|\omega|\leq 1} |\omega|^{2r}|\mathcal{F}f(\omega)|^2 d\omega + \int_{|\omega|>1} |\omega|^{2r}|\mathcal{F}f(\omega)|^2\ \mathrm{d}\omega\\ &\leq\; 1 + \int_{|\omega|>1} |\omega|^{-(d+\epsilon)}\ \mathrm{d}\omega < \infty.\end{aligned}$$

However, for this example $\mathcal{F}f(\omega) \notin L_1$ in the cases where the dimension is relatively higher than the smoothness index. That is, whenever

$$r + (d+\epsilon)/2 \leq d \Leftrightarrow 2r + \epsilon \leq d.$$

This implies that f does not satisfy the condition $\frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} |\mathcal{F}f(\omega)| d\omega \leq 1$ which is required in $\mathcal{B}_{2r,2}$. We see that the L_1 condition on the Fourier transform of functions in $\mathcal{B}_{2r,r}$ allows us to circumvent the curse of dimensionality when approximating in Sobolev spaces. □

Lemma 1. *For any polynomial $p_n : \mathbb{R} \to \mathbb{R}$ of degree n of the form $p_n(x) = \sum_{k=0}^n c_k x^k$, there exists a multiplicative neural network $f_n^{\mathrm{POL}} : \mathbb{R}^3 \to \mathbb{R}$, of depth $L_n = \mathcal{O}(n)$ with $G_n = \mathcal{O}(n)$ neurons that satisfies $f_n^{\mathrm{POL}}(x, x, c_0) = p_n(x)$ for $c_0 \in \mathbb{R}$, and $x \in \mathbb{R}$.*

Proof. Let $p_n(x) = \sum_{k=0}^n c_k x^k$ be a polynomial of degree n and $p_{n,i} = \sum_{k=0}^i c_k x^k$ its partial sum up to term i. We construct a network f_n^{POL} whose ith layer satisfies:

$$(f_n^{\mathrm{POL}})_i(x, x, c_0) = (x, x^{i+1}, p_{n,i}(x)).$$

At layer i, the model weights for the three neurons are defined as:

$$w_{i,1} = (1, 0, 0), \quad a_{i,2} = 1 \text{ for } (j_1, j_2) = (1, 2)\ , \quad w_{i,3} = (0, c_i, 1)$$

The rest of the weights are zeros.

We argue that at any layer $i \geq 0$, neuron 1 contains x, neuron 2 contains x^{i+1} and neuron 3 contains $p_{n,i}(x)$. Let $y_{i+1,1}, y_{i+1,2}, y_{i+1,3}$ be the three neurons in layer $i+1$. Since the only non-zero weight affecting the first neuron is in $w_{i+1,1}$, $y_{i+1,1} = y_{i,1} = x$ by the assumption. The only non-zero weight affecting the second neuron appears in $a_{i=1,2}$, and therefore $y_{i+1,2} = y_{i,1}y_{i,2} = x^{i+1}$. Lastly, $y_{i+1,3}$ depends only on $w_{i+1,3}$, and so $y_{i+1,3} = c_{i+1}y_{i,2} + y_{i,3} = p_{n,i}(x) + c_{i+1}x^{i+1} := p_{n,i+1}(x)$. We conclude using the fact that $p_{n,n}(x) = p_n(x)$. □

Theorem 2. *Let $M \geq 1$, $s > 2$, $C_K > 0$ and $\epsilon \in (0, 1)$ be scalars. Then, for any real-valued analytic function $K : [-M, M] \to \mathbb{R}$ that is analytically continuable to the ellipse E_s^M where $|K(x)| \leq C_K$, there exists a deep multiplicative network $f^{\mathrm{MA}} : [-M, M]^3 \to \mathbb{R}$ (MA stands for 'Multiplicative Analytic') of depth $L_\epsilon = \mathcal{O}(\frac{1}{\log_2 s}\log_2 \frac{C_K}{\epsilon})$ with $G_\epsilon = \mathcal{O}(\frac{1}{\log_2 s}\log_2 \frac{C_K}{\epsilon})$ neurons that satisfies*

$$\|f^{\mathrm{MA}}(x, x, x) - K(x)\|_{L^\infty([-M,M])} \;\leq\; \epsilon.$$

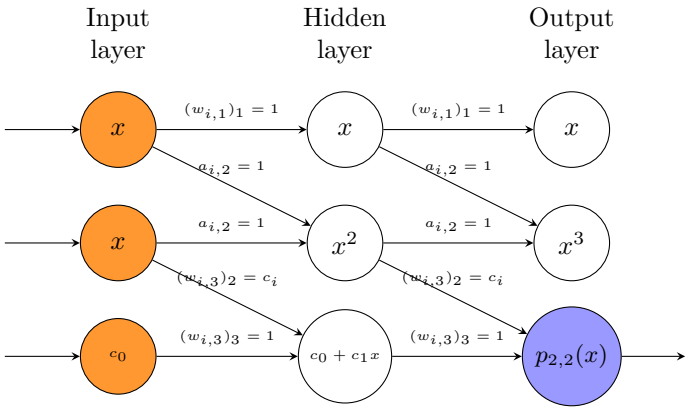


Figure 4: Illustration of the multiplicative network in Proof of Lemma 1, where the polynomial degree $n = 2$.

Proof. Let $M \geq 1$, $s > 2$, $C_K > 0$, $\epsilon \in (0, 1)$ and let K be an analytic function with the required assumptions. As a first step, we approximate K with a polynomial h_n of degree $n \geq 2$ (to be defined later). Then, we realize h_n with a deep multiplicative network using Lemma 1. We recall the polynomial h_n described in Theorem 1. For any integer $n \geq 2$, we have

$$\|h_n - K\|_{L^\infty([-M,M])} \;\leq\; \tfrac{C_K s^{-n}}{s-1} \;=\; \mathcal{O}(C_K s^{-n}).$$

By choosing $n = \frac{1}{\log_2 s} \log_2 \frac{C_K}{\epsilon(s-1)} \leq \frac{1}{\log_2 s} \log_2 \frac{C_K}{\epsilon}$ we obtain the following relation:

$$\begin{aligned} \frac{\epsilon(s-1)}{C_K} \;&=\; 2^{\log_2(\frac{\epsilon(s-1)}{C_K})} \\ &=\; 2^{-\frac{1}{\log_2 s}(\log_2(\frac{C_K}{\epsilon(s-1)}))\log_2 s} \\ &=\; s^{-\frac{1}{\log_2 s}(\log_2(\frac{C_K}{\epsilon(s-1)}))} \;=\; s^{-n}. \end{aligned} \tag{5}$$

In particular,

$$\|h_n - K\|_{L^\infty([-M,M])} \;\leq\; \tfrac{C_K s^{-n}}{s-1} \;\leq\; \epsilon,$$

for $s \geq 1$. Given that h_n is a polynomial of degree n, by Lemma 1, there exists f^{POL} and $c_0 = h_{n,0}$ such that $h_n(x) = f^{\mathrm{POL}}(x, x, c_0)$, and achieve the desired result. □

Let us now recall Maurey's Theorem (Pisier, 1980-1981; Vitali D. Milman, 1986) which will assist us in the proof of Theorem 3.

Lemma 3 (Maurey's theorem)**.** *Let $\mathcal{V}$ be a Hilbert space with norm $\|\cdot\|_{\mathcal{V}}$. Suppose there exists $Q \subset \mathcal{V}$ such that for every $q \in Q$, $\|q\|_{\mathcal{V}} \leq b$ for some $b > 0$. Then, for every f in the convex hull of Q and every integer $n \geq 1$, there exists a f_n in the convex hull of n points in Q and a constant $c > b^2 - \|f\|_{\mathcal{V}}^2$ such that $\|f_n - f\|_{\mathcal{V}}^2 \leq \frac{c}{n}$.*

Theorem 3. *Let $\epsilon \in (0, 1)$, $M > 1, d \geq 2$, and $K : \mathbb{R} \to \mathbb{C}$ be an analytic kernel that holds the assumptions of Theorem 2 with respect to $s > 2, C_K > 0$, and bounded by a constant $D_K \in (0, 1]$ on $[-dM, dM]$. Let f be a real-valued function in $\mathcal{H}_{K,M}(B)$. Further, let $F : [-M, M]^d \to \mathbb{C}$ be a square-integrable function such that $f(x) = \int_{[-M,M]^d} F(\omega) K(\omega \cdot x)\, \mathrm{d}\omega$. We define $C_F = \int_{\mathbb{R}^d} |F(\omega)|\, \mathrm{d}\omega = \int_{[-M,M]^d} |F(\omega)|\, \mathrm{d}\omega$. Then, there exists a deep multiplicative network $f^{\mathrm{MBL}} : B \to \mathbb{R}$ (MBL stands for 'Multiplicative bandlimited') of depth $L_\epsilon = \mathcal{O}\Big(\frac{1}{\log_2 s} \log_2 \frac{C_F C_K}{\epsilon}\Big)$ with $G_\epsilon = \mathcal{O}\Big(\frac{C_F^2}{\epsilon^2 \log_2 s} \log_2 \frac{C_F C_K}{\epsilon}\Big)$ neurons that satisfies $\|f^{\mathrm{MBL}} - f\|_{L^2(B)} \leq \epsilon$.*

Proof. Let $f \in \mathcal{H}_{K,M}(B)$. Further, let $F(\omega) = |F(\omega)| \cdot \exp(i\theta(\omega))$, the polar representation of $F(\omega)$. The following holds:

$$
\begin{aligned}
f(x) &= \mathrm{Re}\left(\int_{[-M,M]^d} F(\omega)K(\omega\cdot x)\ \mathrm{d}\omega\right)\\
&= \mathrm{Re}\left(\int_{[-M,M]^d} C_F\exp(i\theta(\omega))K(\omega\cdot x)\frac{|F(\omega)|}{C_F}\ \mathrm{d}\omega\right)\\
&= \int_{[-M,M]^d} C_F[\cos(\theta(\omega))K_R(\omega\cdot x)-\sin(\theta(\omega))K_I(\omega\cdot x)]\frac{|F(\omega)|}{C_F}\ \mathrm{d}\omega,
\end{aligned}
\tag{6}
$$

where K_R, K_I are the real and imaginary parts of K respectively. Given that the integral represents f as an infinite convex combination of functions in

$$
Q_{K,M} = \left\{\gamma[\cos(\beta)K_R(\omega\cdot x)-\sin(\beta)K_I(\omega\cdot x)]\ \middle|\ |\gamma|\le C_F, \beta\in\mathbb{R}, \omega\in[-M,M]^d\right\},
$$

then f is in the closure of the convex hull of $Q_{K,M}$. Due to the fact that $x\in[0,1]^d$ and $\omega\in[-M,M]^d$, $t=\omega\cdot x\in[-dM,dM]$. By the definition of D_K, functions in $Q_{K,M}$ are bounded in the $L^2(B)$-norm by $2C_FD_K\le 2C_F$. Using Lemma 3, there exist real coefficients $\{b_j\}$ and $\{\beta_j\}$, and vectors $\omega_j\in[-M,M]^d$ for $1\le j\le\lceil 1/\epsilon_0^2\rceil$, such that:

$$
f_{\epsilon_0}(x) = \sum_{j=1}^{\lceil 1/\epsilon_0^2\rceil} b_j[\cos(\beta_j)K_R(\omega_j\cdot x)-\sin(\beta_j)K_I(\omega_j\cdot x)],\quad \sum_{j=1}^{\lceil 1/\epsilon_0^2\rceil}|b_j|\le C_F,
$$

for $0<\epsilon_0<1$ to be defined at a later time, such that

$$
\|f_{\epsilon_0}(x)-f(x)\|_{L^2(B)}\le 2C_F\epsilon_0.
$$

We are now ready to approximate $f_{\epsilon_0}(x)$ using a deep multiplicative neural network f^{MBL} on B. We notice that $K_R(x)$ and $K_I(x)$ are analytic kernels that hold the assumptions of Theorem 2. They can therefore be approximated to ϵ_0 error using networks f^{RMA}, f^{IMA} of depth and number of neurons

$$
G_{\epsilon_0}\sim L_{\epsilon_0} = \mathcal{O}(\frac{1}{\log_2 s}\log_2\frac{C_K}{\epsilon_0}),
$$

where RMA stands for 'Real Multiplicative Analytic' and IMA stands for 'Imaginary Multiplicative Analytic'. Let us define the multiplicative network $f^{\mathrm{MBL}}(x)$ by

$$
f^{\mathrm{MBL}}(x) = \sum_{j=1}^{\lceil 1/\epsilon_0^2\rceil} b_j\left[\cos(\beta_j)f^{\mathrm{RMA}}(\omega_j\cdot x)-\sin(\beta_j)f^{\mathrm{IMA}}(\omega_j\cdot x)\right].
$$

This network has depth $L_{\epsilon_0}=\mathcal{O}(\frac{1}{\log_2 s}\log_2\frac{C_K}{\epsilon_0})$ and $G_{\epsilon_0}=\mathcal{O}(\frac{1}{\epsilon_0^2\log_2 s}\log_2\frac{C_K}{\epsilon_0})$ neurons.

$$
\begin{aligned}
\|f^{\mathrm{MBL}}(x)-f_{\epsilon_0}(x)\|_{L^\infty(B)} &\le \sum_{j=1}^{\lceil 1/\epsilon_0^2\rceil}|b_j|\|f^{\mathrm{RMA}}(\omega_j\cdot x)-K_R(\omega_j\cdot x)\|_{L^\infty(B)}\\
&\quad+\sum_{j=1}^{\lceil 1/\epsilon_0^2\rceil}|b_j|\|f^{\mathrm{IMA}}(\omega_j\cdot x)-K_I(\omega_j\cdot x)\|_{L^\infty(B)}\\
&\le 2C_F\epsilon_0,
\end{aligned}
$$

that implies

$$
\|f^{\mathrm{MBL}}(x)-f(x)\|_{L^2(B)}\le\|f^{\mathrm{MBL}}(x)-f_{\epsilon_0}(x)\|_{L^2(B)}+\|f_{\epsilon_0}(x)-f(x)\|_{L^2(B)}\le 4C_F\epsilon_0,
$$

where the last inequality is due to the fact that

$$\|g\|_{L^2(B)} = \left(\int_B |g|^2\right)^{1/2} \leq \|g\|_{L^\infty(B)} = \|g\|_{L^\infty(B)}.$$

Taking $\epsilon_0 = \epsilon/(4C_F)$ achieves the sought result. □

We further investigate how the constant C_K from Theorem 3 behaves as functions of M and s. Let $K(x) = \exp(ix)$ be an example kernel, $x \in [-M, M]^d$. We notice that $a_s^M = M(s + s^{-1})/2$ for $s > 2$, is the larger axis, and therefore the maximal norm of K on E_s^M is given by $K(a_s^M)$:

$$\max_x |K(x)| \leq \exp(M\tfrac{s+s^{-1}}{2}) = C_K(s, M).$$

In our setting, inputs are in the interval $t = \omega \cdot x \in [-dM, dM]$, so we may use the bounding constant

$$\max_t |K(t)| \ \leq \ \exp(dM\tfrac{s+s^{-1}}{2}) = C_K(s, dM).$$

The resulting network f^{MBL} then has depth

$$L_\epsilon \ = \ \mathcal{O}\bigg(\frac{1}{\log_2 s}\bigg(dM\frac{s + s^{-1}}{2} + \log_2 \frac{C_F}{\epsilon}\bigg)\bigg)$$

and

$$G_\epsilon = \mathcal{O}\bigg(\frac{C_F^2}{\epsilon^2 \log_2 s}\bigg(dM\frac{s + s^{-1}}{2} + \log_2 \frac{C_F}{\epsilon}\bigg)\bigg)$$

neurons. In this scenario, we see that both a large band M and a large dimension d will linearly affect the first term.

Theorem 4. *Let $d \geq 2, r \in \mathbb{N}$, $f \in \mathcal{B}_{2r,2}$ and $\epsilon > 0$. There exists a deep ReLU network f^{RS} (standing for "ReLU Sobolev") with a depth of $L_\epsilon = \mathcal{O}(d^2\epsilon^{-2/r})$ and $G_\epsilon = \mathcal{O}(d^2\epsilon^{-(2+2/r)})$ neurons, such that $\left\|f^{\mathrm{RS}} - f\right\|_{L_2(B)} \leq \epsilon$.*

Proof. Let $f \in \mathcal{B}_{2r,2}$, and $M > 1$. Similar to the proof of Theorem 5, we define the bandlimiting of $f : \mathbb{R}^d \to \mathbb{R}$ by

$$f_M \ = \ \mathcal{F}^{-1}(\mathcal{F}f\mathbb{1}_{[-M,M]^d}).$$

Let us define $F : [-M, M]^d \to \mathbb{C}$:

$$F(\omega) \ = \ \frac{1}{(2\pi)^d}(\mathcal{F}f)(\omega),$$

and $K(x) = \exp(ix)$. We then have the following identity:

$$f_M(x) \ = \ \int_{[-M,M]^d} F(\omega)K(\omega \cdot x)\ \mathrm{d}\omega \ = \ \int_{[-M,M]^d} \tfrac{1}{(2\pi)^d}\mathcal{F}f(\omega)\exp(i\omega \cdot x)\ \mathrm{d}\omega.$$

It is then easy to see that $f_M \in \mathcal{H}_{K,M}(B)$. We choose $M = (2/\epsilon)^{1/r}$. Using the same derivations as in the proof of Theorem 5, we seek to approximate f_M with a network f^{RS} such that

$$\|f_M - f^{\mathrm{RS}}\|_{L_2(B)} \ \leq \ \epsilon/2.$$

We then arrive at

$$\begin{aligned} \|f - f^{\mathrm{RS}}\|_{L_2(B)} \ &\leq \ \|f - f_M\|_{L_2(B)} + \|f_M - f^{\mathrm{RS}}\|_{L_2(B)} \\ &\leq \ M^{-r} + \epsilon/2 \ \leq \ \epsilon. \end{aligned} \tag{7}$$

We consider that $C_F = \int_{[-M,M]^d} |F(\omega)|\ \mathrm{d}\omega = \frac{1}{(2\pi)^d}\int_{[-M,M]^d} |\mathcal{F}f(\omega)|\ \mathrm{d}\omega \leq 1$, where the inequality is due to the definition of $\mathcal{B}_{2r,2}$. The kernel $K(t) = \exp(it)$ takes as input $t = \omega \cdot x$, for $x \in B$ and $\omega \in [-M, M]^d$.

Therefore, $t \in [-dM, dM]$, and $K : [-dM, dM] \to \mathbb{R}$. We note that K is continuable to the Bernstein 4-ellipse E_4^{dM}. We notice that $a_4^{dM} = dM(4 + 4^{-1})/2$ is the larger axis, and therefore the maximal norm of K on E_s^{dM} is given by $K(a_4^{dM})$:

$$C_K \;=\; \max_t |K(t)| \;\leq\; \exp(dM\tfrac{4+4^{-1}}{2}).$$

Further, for any $t \in \mathbb{R}$ we have $|K(t)| \leq 1 = D_K$.

Using Theorem 3.2 from (Montanelli et al., 2021) we can construct a deep ReLU network f^{RS} such that

$$\|f_M - f^{\mathrm{RS}}\|_{L_2(B)} \;\leq\; \epsilon/2$$

whose depth is

$$\begin{aligned} L_\epsilon \;&\leq\; C_1 \frac{1}{\log_2^2 4} \log_2^2 \frac{2C_F C_K}{\epsilon} \\ &\leq\; C_2 \frac{1}{\log_2^2 4} \left(\log_2 \frac{\exp(dM\frac{4+4^{-1}}{2})}{\epsilon} \right)^2 \\ &\leq\; 12C_2 \left(d\left(\frac{2}{\epsilon}\right)^{\frac{1}{r}} - \log_2 \frac{1}{\epsilon} \right)^2 \\ &\leq\; C_3 d^2 \epsilon^{-\frac{2}{r}}, \end{aligned} \tag{8}$$

for some constants $C_1, C_2, C_3 > 0$. In addition, the number of neurons can be bounded by

$$\begin{aligned} G_\epsilon \;&\leq\; C_1' C_F^2 \frac{1}{\epsilon^2 \log_2^2 4} \log_2^2 \frac{2C_K C_F}{\epsilon} \\ &\leq\; C_2' \frac{1}{\epsilon^2 \log_2^2 4} \left(\log_2 \frac{\exp(dM\frac{4+4^{-1}}{2})}{\epsilon} \right)^2 \\ &\leq\; \frac{12C_2'}{\epsilon^2} \left(d\left(\frac{2}{\epsilon}\right)^{\frac{1}{r}} - \log_2 \frac{1}{\epsilon} \right)^2 \\ &\leq\; C_3' d^2 \epsilon^{-(2+\frac{2}{r})}, \end{aligned} \tag{9}$$

for some constants $C_1', C_2', C_3' > 0$. □