The phase diagram of approximation rates for deep neural networks

Dmitry Yarotsky
Skolkovo Institute of Science and Technology
d.yarotsky@skoltech.ru

Anton Zhevnerchuk
Skolkovo Institute of Science and Technology
Anton.Zhevnerchuk@skoltech.ru

Abstract

We explore the phase diagram of approximation rates for deep neural networks and prove several new theoretical results. In particular, we generalize the existing result on the existence of deep discontinuous phase in ReLU networks to functional classes of arbitrary positive smoothness, and identify the boundary between the feasible and infeasible rates. Moreover, we show that all networks with a piecewise polynomial activation function have the same phase diagram. Next, we demonstrate that standard fully-connected architectures with a fixed width independent of smoothness can adapt to smoothness and achieve almost optimal rates. Finally, we consider deep networks with periodic activations ("deep Fourier expansion") and prove that they have very fast, nearly exponential approximation rates, thanks to the emerging capability of the network to implement efficient lookup operations.

1 Introduction

There is a subtle interplay between different notions of complexity for neural networks. One, most obvious, aspect of complexity is the network size measured in terms of the number of connections and neurons. Another is characteristics of the network architecture (e.g., shallow or deep). A third is the type of the activation function used in the neurons. Yet another, important but sometimes overlooked aspect is the precision of operations performed by neurons. All these complexities are connected by tradeoffs: if we fix a particular problem solvable by neural networks, then we have some freedom in decreasing one complexity at the cost of others. The question we address is: *what are the limits of this freedom*? In the present paper we perform a systematic theoretical study of this question in the context of network expressiveness. We fix the classical approximation problem and explore the opportunities potentially present in solving it within different neural network scenarios.

Specifically, suppose that we have a class F of maps from the d-dimensional cube $[0,1]^d$ to $\mathbb{R}$, and we want the network to approximate elements of F in the uniform norm $\|\cdot\|_\infty$. We will make the standard assumption that F is a Sobolev- or Hölder ball of smoothness $r > 0$ (i.e., a ball of "r times differentiable functions", see Section 2). Then, for a particular type of approximation model, we examine the optimal *approximation rate*, i.e. the relation beween the approximation accuracy and the required number W of model parameters. Typically, this relation has the form of a power law

$$\|f - \widetilde{f}_W\|_\infty = O(W^{-p}), \quad \forall f \in F, \tag{1}$$

where $\widetilde{f}_W$ is an approximation of f by a model with W parameters, and p is a constant (which we will also refer to as the *rate*). In standard fully-connected networks, there is one parameter (weight) per each connection and neuron, so W can be equivalently viewed as the size of the model. Our approach in this paper will be to analyze how the rates p depend on various approximation conditions (e.g., network depth, activation functions, etc.).

There are several important general ideas explaining which approximation rates p we can reasonably expect in Eq.(1). In the context of abstract approximation theory, we can forget (for a moment) about

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

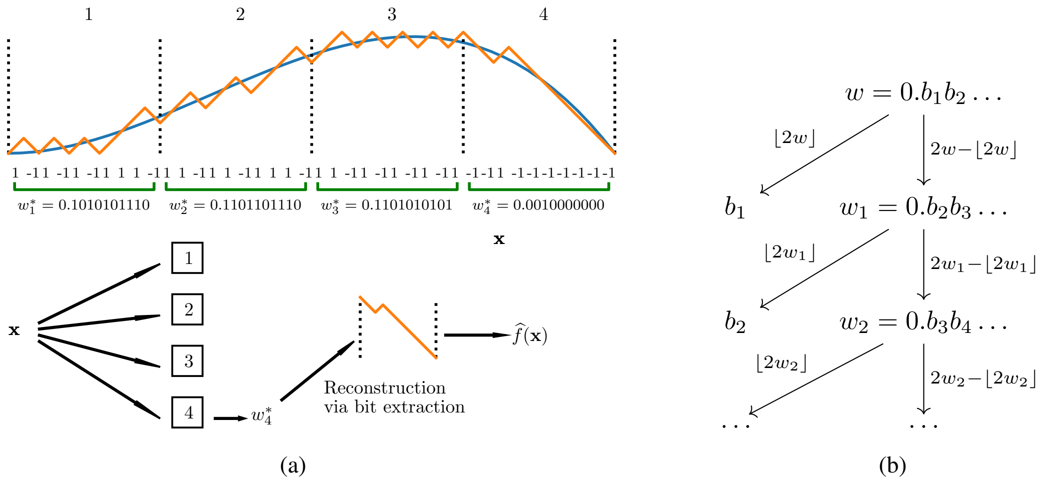


Figure 1: **(a)** A high-rate approximation from [10]. The domain $[0, 1]^d$ is divided into patches and an approximation to f is encoded in each patch by a single network weight using a binary-type representation. Then, the network computes the approximation $\widetilde{f}(\mathbf{x})$ by finding the relevant weight and decoding it using the bit extraction technique of [11] (here, $d = 1, r = 1$, and $p = \frac{2r}{d} = 2$). **(b)** Sequential bit extraction by a deep network [11]. (The floor function $\lfloor \cdot \rfloor$ can be approximated by ReLU with arbitrary accuracy via $\lfloor w \rfloor \approx \frac{1}{\delta}(w-1)_+ - \frac{1}{\delta}(w-1-\delta)_+$ with a small δ.)

the network-based implementation of $\widetilde{f}_W$ and just think of it as some approximate parameterization of F by vectors $\mathbf{w} \in \mathbb{R}^W$. Let us view the approximation process $f \mapsto \widetilde{f}_W$ as a composition of the *weight assignment* map $f \mapsto \mathbf{w}_f \in \mathbb{R}^W$ and the *reconstruction* map $\mathbf{w}_f \mapsto \widetilde{f}_W \in \mathcal{F}$, where $\mathcal{F}$ is the full normed space containing F. If both the weight assignment and reconstruction maps were linear, and so their composition $f \mapsto \widetilde{f}_W$, the l.h.s. of Eq.(1) could be estimated by the *linear W-width* of the set F (see [1]). For a Sobolev ball of d-variate functions f of smoothness r, the linear W-width is asymptotically $\sim W^{-r/d}$, suggesting the approximation rate $p = \frac{r}{d}$. Remarkably, this argument extends to *non-linear* weight assignment and reconstruction maps under the assumption that the weight assignment is *continuous*. More precisely, it was proved in [2] that, under this assumption, p in Eq.(1) cannot be larger than $\frac{r}{d}$.

An even more important set of ideas is related to estimates of Vapnik-Chervonenkis dimensions of deep neural networks. The concept of expressiveness in terms of VC-dimension (based on finite set shattering) is weaker than expressiveness in terms of uniform approximation, but upper bounds on the VC-dimension directly imply upper bounds on feasible approximation rates. In particular, the VC-dimension of networks with piecewise-polynomial activations is $O(W^2)$ ([3]), which implies that p cannot be larger than $\frac{2r}{d}$ – note the additional factor 2 coming from the power 2 in the VC bound. We refer to the book [4] for a detailed exposition of this and related results.

Returning to approximations with networks, the above arguments suggest that the rate p in Eq.(1) can be up to $\frac{r}{d}$ assuming the continuity of the weight assignment, and up to $\frac{2r}{d}$ without assuming the continuity, but assuming a piecewise-polynomial activation function such as ReLU. We then face the constructive problem of showing that these rates can indeed be fulfilled by a network computation. One standard general strategy of proving the rate $p = \frac{r}{d}$ is based on polynomial approximations of f (in particular, via the Taylor expansion). A survey of early results along this line for networks with a single hidden layer and suitable activation functions can be found in [5]. An interesting aspect of piecewise-linear activations such as ReLU is that the rate $p = \frac{r}{d}$ cannot be achieved with single-layer networks, but can be achieved with deeper networks implementing approximate multiplication and polynomials ([6, 7, 8, 9]).

It was shown in [10] that ReLU networks can also achieve rates p beyond $\frac{r}{d}$. The result of [10] is stated in terms of the modulus of continuity of f; when restricted to Hölder functions with constant $r \le 1$, it implies that on such functions ReLU networks can provide rates p in the interval $(\frac{r}{d}, \frac{2r}{d}]$, in agreement with the mentioned upper bound $\frac{2r}{d}$. The construction is quite different from the case

$p = \frac{r}{d}$ and has a "coding theory" rather than "analytic" flavor, see Fig.1. In agreement with continuous approximation theory and existing VC bounds, the construction inherently requires discontinuous weight assignment (as a consequence of coding finitely many values) and network depth (necessary for the bit extraction part). In this sense, at least in the case of $r \leq 1$ one can distinguish two qualitatively different "approximation phases": the shallow continuous one corresponding to $p = \frac{r}{d}$ (and lower values), and the deep discontinuous one corresponding to $p \in (\frac{r}{d}, \frac{2r}{d}]$. It was shown in [8, 12] that the shallow rate $p = \frac{r}{d}$, but not faster rates, can be achieved if the network weights are discretized with the precision of $O(\log(1/\epsilon))$ bits, where ϵ is the approximation accuracy.

We remark in passing that in recent years there has also been a substantial amount of related research on other aspects of deep network expressiveness: e.g. (just to give a few examples) on performance scaling with input dimension [13, 14], depth separation [15, 16], generalization from finite training sets [17], approximation on manifolds [18], approximation of discontious functions [8] and specific signal structures [19, 20]. These topics are outside the scope of the present paper.

Contribution of this paper. The developments described above leave many questions open. One immediate question is whether and how the deep discontinuous approximation phase generalizes to higher values of smoothness ($r > 1$). Another natural question is how much the network architectures providing the maximal rate $p = \frac{2r}{d}$ depend on the smoothness class. Yet another question is how sensitive the phase diagram is with respect to changing ReLU to other activation functions. In the present paper we resolve some of these questions and, moreover, offer new perspectives on the tradeoffs between different aspects of complexity in neural networks. Specifically:

- In Section 3, we prove that the approximation phase diagram indeed generalizes to arbitrary smoothness $r > 0$, with the deep discontinuous phase occupying the region $\frac{r}{d} < p \leq \frac{2r}{d}$.
- In Section 4, we prove that the standard fully-connected architecture with a sufficiently large constant width H only depending on the dimension d, say $H = 2d + 10$, can implement approximations that are asymptotically almost optimal for *arbitrary* smoothness r. This property can be described as "universal adaptivity to smoothness" exhibited by such architectures.
- In Section 5, we discuss how the ReLU phase diagram can change if ReLU is replaced by other activation functions. In particular, we prove that the deep discontinuos phase can be constructed for any activation that has a point of nonzero curvature. This implies that the phase diagram for any piecewise polynomial activation is the same as for ReLU.
- In Section 6 we consider what we call *"deep Fourier expansion"* – approximation by a deep network with a periodic activation function, which can be seen as a generalization of the usual Fourier series approximation. We prove that such networks can provide much faster, exponential rates compared to the polynomial rates of ReLU networks. The key element of the proof is a new version of the bit extraction procedure replacing sequential extraction by a dichotomy-based lookup.
- In Section 7 we analyze the distribution of information in the networks implementing the discussed modes of approximation. In particular, we show that in the deep discontinuous ReLU phase the total information $\epsilon^{-d/r}$ is uniformly distributed over the $\epsilon^{-1/p}$ encoding weights, with $\epsilon^{1/p-d/r}$ bits per weight, while in the "deep Fourier" it is all concentrated in a single encoding weight.

2 Preliminaries

Smooth functions. The paper revolves about what we informally describe as "functions of smoothness r", for any $r > 0$. It is convenient to precisely define them in terms of Hölder spaces. Any $r > 0$ can be uniquely represented as $r = k + \alpha$ with an integer $k \geq 0$ and $0 < \alpha \leq 1$. We define the respective Hölder space $\mathcal{C}^{k,\alpha}([0,1]^d)$ as the space of k times continuously differentiable functions on $[0,1]^d$ having a finite norm

$$\|f\|_{\mathcal{C}^{k,\alpha}([0,1]^d)} = \max\Big\{ \max_{\mathbf{k}:|\mathbf{k}|\leq k} \max_{\mathbf{x}\in[0,1]^d} |D^{\mathbf{k}} f(\mathbf{x})|, \max_{\mathbf{k}:|\mathbf{k}|=k} \sup_{\substack{\mathbf{x},\mathbf{y}\in[0,1]^d,\\ \mathbf{x}\neq\mathbf{y}}} \frac{|D^{\mathbf{k}} f(\mathbf{x}) - D^{\mathbf{k}} f(\mathbf{y})|}{\|\mathbf{x}-\mathbf{y}\|^\alpha} \Big\}.$$

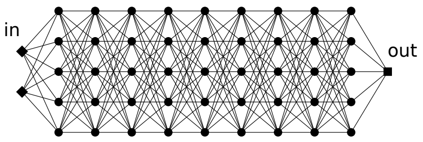


Figure 2: A standard deep fully-connected architecture with width 5.

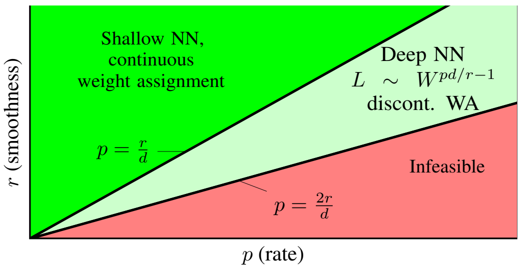


Figure 3: The phase diagram of approximation rates for ReLU networks.

Here $D^{\mathbf{k}} f$ denotes the partial derivative of f. We choose the sets F appearing in Eq.(1) to be the unit balls in these Hölder spaces and denote them by $F_{r,d}$.

Neural networks. We consider conventional feedforward neural networks with layouts given by directed acyclic graphs. Each hidden unit performs a computation of the form $\sigma(\sum_{k=1}^{K} w_k z_k + h)$, where z_k are the signals from the incoming connections, and w_k and h are the weights associated with this unit. In addition to input units and hidden units, the network is assumed to have a single output unit performing a computation similar to that of hidden units, but without the activation function. In Sections 3 and 4 we assume that the activation function is ReLU: $\sigma(x) = a_+ \equiv \max(0, a)$. In general, we refer to networks with an activation function σ as *σ-networks*.

In general, we do not make any special connectivity assumptions about the network architecture. The exception is Section 4 where we consider a particular family of architectures in which hidden units are divided into a sequence of layers, and each layer has a constant number of units. Two units are connected if and only if they belong to neighboring layers. The input units are connected to the units of the first hidden layer and only to them; the output unit is connected to the units of the last hidden layer, and only to them. We refer to this as a *standard deep fully-connected architecture with constant width* (see Fig.2).

Whenever we mention a *piecewise linear* or *piecewise polynomial* activation, we mean that $\mathbb{R}$ can be divided into *finitely many* intervals on which the activation is linear or polynomial (respectively). Without this condition of finiteness, activations could be made drastically more expressive (e.g., by joining a dense countable subset of polynomials [21]).

Approximations. In the accuracy–complexity relation (1) we assume that approximations $\widetilde{f}_W$ are obtained by assigning f-dependent weights to a network architecture η_W *common* to all $f \in F$. In particular, this allows us to speak of the weight assignment map $G_W : f \mapsto \mathbf{w}_f \in \mathbb{R}^W$ associated with a particular architecture η_W. We say that the weight assignment is continuous if this map is continuous with respect to the topology of uniform norm $\|\cdot\|_\infty$ on F. We will be interested in considering different approximation rates p, and we interpret Eq.(1) in a precise way by saying that a rate p can be achieved iff

$$\inf_{\eta_W, G_W} \sup_{f \in F} \|f - \widetilde{f}_{\eta_W, G_W}\|_\infty \le c_{F,p} W^{-p}, \tag{2}$$

where $\widetilde{f}_{\eta_W, G_W}$ denotes the approximation obtained by the weight assignment G_W in the architecture η_W. Here and in the sequel we generally denote by $c_{a,b,\ldots}$ various positive constants possibly dependent on $a, b, \ldots$ (typically on smoothness r and dimension d). Throughout the paper, we will treat r and d as fixed parameters in the asymptotic accuracy-complexity relations.

3 The phase diagram of ReLU networks

Our first main result is the phase diagram of approximation rates for ReLU networks, shown in Fig.3. The "shallow continuous phase" corresponds to $p = \frac{r}{d}$, the "deep discontinuous phase" corresponds to $\frac{r}{d} < p \le \frac{2r}{d}$, and the infeasible region corresponds to $p > \frac{2r}{d}$. Our main new contribution is the exact location of the deep discontinuous phase for all $r > 0$. The precise meaning of the diagram is explained by the following series of theorems (partly established in earlier works).

Theorem 3.1 (The shallow continuous phase)**.** *The approximation rate $p = \frac{r}{d}$ in Eq.*(2) *can be achieved by ReLU networks having $L \le c_{r,d} \log W$ layers, and with a continuous weight assignment.*

This result was proved in [6] in a slightly weaker form, for integer r and with error $O(W^{-r/d} \log^{r/d} W)$ instead of $O(W^{-r/d})$. The proof is based on ReLU approximations of local Taylor expansions of f. The extension to non-integer r is immediate thanks to our definition of general r-smoothness in terms of Hölder spaces. The logarithmic factor $\log^{r/d} W$ can be removed by observing that the computation of the approximate Taylor polynomial can be isolated from determining its coefficients and hence only needs to be implemented once in the network rather than for each local patch as in [6] (see Remark A.1; the idea of isolation of operations common to all patches is developed much further in the proof Theorem 3.3 below, and is applicable in the special case $p = \frac{r}{d}$).

Theorem 3.2 (Feasibility of rates $p > \frac{r}{d}$)**.**

1. *Approximation rates $p > \frac{2r}{d}$ are infeasible for networks with piecewise-polynomial activation function and, in particular, ReLU networks;*

2. *Approximation rates $p \in (\frac{r}{d}, \frac{2r}{d}]$ cannot be achieved with continuous weights assignment;*

3. *If an approximation rate $p \in (\frac{r}{d}, \frac{2r}{d}]$ is achieved with ReLU networks, then the number of layers L in η_W must satisfy $L \ge c_{p,r,d} W^{pd/r-1}/\log W$ for some $c_{p,r,d} > 0$.*

These statements follow from existing results on continuous nonlinear approximation ([2] for statement 2) and from upper bounds on VC-dimensions of neural networks ([3] for statement 1 and [22] for statement 3), see [6, Theorem 1] for a derivation. The extensions to arbitrary r are straightforward.

The main new result in this section is the existence of approximations with $p \in (\frac{r}{d}, \frac{2r}{d}]$:

Theorem 3.3 (The deep discontinuous phase)**.** *For any $r > 0$, any rate $p \in (\frac{r}{d}, \frac{2r}{d}]$ can be achieved with deep ReLU networks with $L \le c_{r,d} W^{pd/r-1}$ layers.*

This result was proved in [10] in the case $r \le 1$. We generalize this to arbitrary r by combining the coding-based approach of [10] with Taylor expansions. We give a sketch of proof below; the full proof is given in Section A.

Sketch of proof. We use two length scales for the approximation: the coarser one $\frac{1}{N}$ and the finer one $\frac{1}{M}$, with $M \gg N$. We start by partitioning the cube $[0,1]^d$ into $\sim N^d$ patches (particularly, simplexes) of linear size $\sim \frac{1}{N}$, and then sub-partitioning them into patches of linear size $\sim \frac{1}{M}$. In each of the finer M-patches Δ_M we approximate the function $f \in F_{r,d}$ by a Taylor polynomial P_{Δ_M} of degree $\lceil r \rceil - 1$. Then, from the standard Taylor remainder bound, we have $|f(\mathbf{x}) - P_{\Delta_M}(\mathbf{x})| = O(M^{-r})$ on Δ_M. This shows that if ϵ is the required approximation accuracy, we should choose $M \sim \epsilon^{-1/r}$.

Now, if we tried to simply save the Taylor coefficients for each M-patch in the weights of the network, we would need at least $\sim M^d$, i.e. $\sim \epsilon^{-d/r}$, weights in total. This corresponds to the classical rate $p = \frac{r}{d}$. In order to save on the number of weights and achieve higher rates, we collect Taylor coefficients of all M-patches lying in one N-patch and encode them in a single *encoding weight* associated with this N-patch. Given $p > \frac{r}{d}$, we choose $N \sim \epsilon^{-1/(pd)}$, so that in total we create $\sim \epsilon^{-1/p}$ encoding weights, each containing information about $\sim (M/N)^d$, i.e. $\sim \epsilon^{-(d/r-1/p)}$, Taylor coefficients. The number of encoding weights then matches the desired complexity $W \sim \epsilon^{-1/p}$.

To encode the Taylor coefficients we actually need to discretize them first. Note that to reconstruct the Taylor approximation in an M-patch with accuracy ϵ, we need to know the Taylor coefficients of order k with precision $\sim M^{-(r-k)}$. We implement an efficient sequential encoding/decoding procedure for the approximate Taylor coefficients of orders $k < \lceil r \rceil$ for all M-patches lying in the given N-patch Δ_N. Specifically, choose some sequence $(\Delta_M)_t$ of the M-patches in Δ_N so that neighboring elements of the sequence correspond to neighboring patches. Then, the order-k Taylor coefficients at $(\Delta_M)_{t+1}$ can be determined with precision $\sim M^{-(r-k)}$ from the respective and higher order coefficients at $(\Delta_M)_t$ using $O(1)$ predefined discrete values. This allows us to encode all the approximate Taylor coefficients in all the M-patches of Δ_N by a single $O((M/N)^d)$-bit number.

To reconstruct the approximate Taylor polynomial for a particular input $\mathbf{x} \in \Delta_M \subset \Delta_N$, we sequentially reconstruct all the coefficients for the sequence $(\Delta_M)_t$, and, among them, select the coefficients at the patch $(\Delta_M)_{t_0} = \Delta_M$. The sequential reconstruction can be done by a deep subnetwork with the help of the bit extraction technique [11]. The depth of this subnetwork is proportional to the number of M-patches in Δ_N, i.e. $\sim (M/N)^d$, which is $\sim \epsilon^{-(d/r-1/p)}$ according to our definitions of N and M. If $p \le \frac{2r}{d}$, then $\frac{d}{r} - \frac{1}{p} \le \frac{1}{p}$ and hence this depth is smaller or comparable to the number of encoding weights, $\epsilon^{-1/p}$. However, if $p > \frac{2r}{d}$, then the depth is asymptotically larger than the number of encoding weights, so the total number of weights is dominated by the depth of the decoding subnetwork, which is $\gtrsim \epsilon^{-d/(2r)}$, and the approximation becomes less efficient than at $p = \frac{2r}{d}$. This explains why $p = \frac{2r}{d}$ is the boundary of the feasible region.

Once the (approximate) Taylor coefficients at $\Delta_M \ni \mathbf{x}$ are determined, an approximate Taylor polynomial $\widetilde{P}_{\Delta_M}(\mathbf{x})$ can be computed by a ReLU subnetwork implementing efficient approximate multiplications [6]. □

4 Fixed-width networks: universal adaptivity to smoothness

The network architectures constructed in the proof of Theorem 3.3 to provide the faster rates $p \in (\frac{r}{d}, \frac{2r}{d}]$ are relatively complex and r-dependent. We can ask if such rates can be supported by some simple conventional architectures. It turns out that we can achieve nearly optimal rates using standard fully-connected architectures with sufficiently large constant widths only depending on d:

Theorem 4.1. *Let η_W be standard fully-connected ReLU architectures with width $2d+10$ and W weights. Then*

$$\inf_{G_W} \sup_{f \in F_{r,d}} \|f - \widetilde{f}_{\eta_W, G_W}\|_\infty \le c_{r,d} W^{-2r/d} \log^{2r/d} W. \tag{3}$$

The rate in Eq.(3) differs from the optimal rate with $p = \frac{2r}{d}$ only by the logarithmic factor $\log^{2r/d} W$. We give a sketch of proof of Theorem 4.1 below, and details are provided in Section B.

An interesting result proved in [23, 24] (see also [25] for a related result for ResNets) states that standard fully-connected ReLU architectures with a fixed width H can approximate any d-variate continuous function if and only if $H \ge d+1$. Theorem 4.1 shows that with slightly larger widths, such networks can not only adapt to any function, but also adapt to its smoothness. The results of [23, 24] also show that Theorem 4.1 cannot hold with d-independent widths.

Sketch of proof of Theorem 4.1. The proof is similar to the proof of Theorem 3.3, but requires a different implementation of the reconstruction of $\widetilde{f}(\mathbf{x})$ from encoded Taylor coefficients. The network constructed in Theorem 3.3 traverses M-knots of an N-patch and computes Taylor coefficients at the new M-knot by updating the coefficients at the previous M-knot. This computation can be arranged within a fixed-width network, but its width depends on r, since we need to store the coefficients from the previous step, and the number of these coefficients grows with r (see [10] for the constant-width fully-connected implementation in the case of $r \le 1$, in which the Taylor expansion degenerates into the 0-order approximation).

To implement the approximation using an r-independent network width, we can decode the Taylor coefficients afresh at each traversed M-knot, instead of updating them. This is slightly less efficient and leads to the additional logarithmic factor in Eq.(3), as can be seen in the following way. First, since we need to reconstruct the Taylor coefficients of degree k with precision $O(M^{-(r-k)})$, we need to store $\sim \log M$ bits for each coefficient in the encoding weight. Since $M \sim \epsilon^{-1/r}$, this means a $\sim \log(1/\epsilon)$-fold increase in the depth of the decoding subnetwork. Moreover, an approximate Taylor polynomial must be computed separately for each M-patch. Multiplications can be implemented with accuracy ϵ by a fixed-width ReLU network of depth $\sim (\log(1/\epsilon))$ (see [6]). Computation of an approximate polynomial of the components of the input vector $\mathbf{x}$ can be arranged as a chain of additions and multiplications in a network of constant width independent of the degree of the polynomial – assuming the coefficients of the polynomial are decoded from the encoding weight and supplied as they become required. This shows that we can achieve accuracy ϵ with a network of constant width independent of r at the cost of taking the larger depth $\sim \epsilon^{-d/(2r)} \log(1/\epsilon)$ (instead of simply $\sim \epsilon^{-d/(2r)}$ as in Theorem 3.3). Since W is proportional to the depth, we get $W \sim \epsilon^{-d/(2r)} \log(1/\epsilon)$. By inverting this relation, we obtain Eq.(3). □

5 Activation functions other than ReLU

We discuss now how much the ReLU phase diagram of Section 3 can change if we use more complex activation functions. We note first that statement 1 of Theorem 3.2 holds not only for ReLU, but for any piecewise-polynomial activation functions, so that the region $p > \frac{2r}{d}$ remains infeasible for any such activation. Also, since all piecewise-linear activation functions are essentially equivalent (see e.g. [6, Proposition 1]), the phase diagram for any piecewise-linear activation is the same as for ReLU.

Our main result in this section states that Theorem 3.3 establishing the existence of the deep discontinuous phase remains valid for any activation that has a point of nonzero curvature.

Theorem 5.1. *Suppose that the activation function σ has a point x_0 where the second derivative $\frac{d^2\sigma}{dx^2}(x_0)$ exists and is nonzero. Then, any rate $p \in (\frac{r}{d}, \frac{2r}{d})$ can be achieved with deep σ-networks with $L \le c_{r,d} W^{pd/r-1}$ layers.*

The proof is given in Section C; its idea is to reduce the approximation by σ-networks to deep polynomial approximations. Then, we can follow the lines of the proof of Theorem 3.3 with some adjustments (in particular, we replace the usual bit extraction dynamic as in Fig.1(b) by a polynomial dynamical system). We remark that in general, if constrained by degree, polynomials poorly approximate ReLU and other piecewise linear functions [26], but in our setting the polynomials are constrained by their compositional complexity rather than degree, in which case a polynomial approximation of ReLU can be much more accurate.

Combined with Statement 1 of Theorem 3.2, Theorem 5.1 implies, in particular, that the phase diagram for general piecewise polynomial activation functions is the same as for ReLU:

Corollary 5.1. *Let σ be a continuous piecewise polynomial activation function. Then the rates $p < \frac{2r}{d}$ are feasible for σ-networks, and the rates $p > \frac{2r}{d}$ are infeasible.*

A remarkable class of functions that can be seen as a far-reaching generalization of polynomials are the Pfaffian functions [27]. Level sets of these functions admit bounds on the number of their connected components that are similar to analogous bounds for algebraic sets, and this is a key property in establishing upper bounds on VC dimensions of networks. In particular, it was proved in [28] that the VC-dimension of networks with the standard sigmoid activation function $\sigma(x) = 1/(1 + e^{-x})$ is upper-bounded by $O(W^2k^2)$, where k is the number of computation units (see also [4, Theorem 8.13]). Since $k \le W$, the bound $O(W^2k^2)$ implies the slightly weaker bound $O(W^4)$. Then, by mimicking the proof of statement 1 of Theorem 3.2 and replacing there the bound $O(W^2)$ for piecewise-polynomial activation by the bound $O(W^4)$ for the standard sigmoid activation, we get

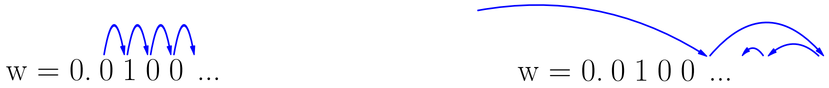
Theorem 5.2. *For networks with the standard sigmoid activation function $\sigma = 1/(1 + e^{-x})$, the rates $p > \frac{4r}{d}$ are infeasible.*

It appears that there remains a significant gap between the upper and lower VC dimension bounds for networks with $\sigma(x) = 1/(1 + e^{-x})$ (see a discussion in [4, Chapter 8]). Likewise, we do not know if the approximation rates up to $p = \frac{4r}{d}$ are indeed feasible with this σ.

All the above results ignore both precision and magnitude of the network weights. In fact, the rates $p > \frac{2r}{d}$ can be excluded for rather general activation functions if we put some mild constraints on the growth of the weights. In Section D we explain this point using a covering number bound from [4, Theorem 14.5].

6 "Deep Fourier expansion"

Note that the usual Fourier series expansion $f(\mathbf{x}) \sim \sum_{\mathbf{n}\in\mathbb{Z}^d} a_\mathbf{n} e^{2\pi i \mathbf{n}\cdot\mathbf{x}}$ for a function f on $[0,1]^d$ can be viewed as a neural network with one hidden layer, the sin activation function, and predefined weights in the first layer. Standard convergence bounds for Fourier series (see e.g. [29]) correspond to the shallow continuous rate $p = \frac{r}{d}$, in agreement with the linearity of the standard assignment



(a) With ReLU

(b) With a periodic activation σ

Figure 4: Standard sequential (a) and dichotomy-based (b) bit extraction. Bit extraction is used to decode information from network weights and is crucial in achieving non-classical rates $p > \frac{r}{d}$. Standard bit extraction ([11], see Fig.1) is available with the threshold or ReLU activation functions. The bits are decoded one-by-one, which requires a significant networks depth and caps feasible rates at $p = \frac{2r}{d}$. In contrast, "deep Fourier expansion" of Theorem 6.1 is essentially based on a more efficient dichotomy-based lookup that becomes available if neurons can implement a periodic activation function (see Section E).

of Fourier coefficients. We can ask what happens to the expressiveness of this approximation if we generalize it by removing all constraints on the architecture and weights, i.e., consider a general deep network with the $\sin$ activation function.

It turns out that such a model is drastically more expressive than both standard Fourier expansion and deep ReLU networks. The key factor in this is the *periodicity* of the activation function $\sigma = \sin$; the particular form of σ is not that important. Our main result below assumes that the network can use both ReLU and σ as activation functions; we refer to these networks as *mixed ReLU/σ* networks.

Theorem 6.1. *Fix r, d. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a Lipschitz periodic function with period T. Suppose that $\sigma(x) > 0$ for $x \in (0, T/2)$ and $\sigma(x) < 0$ for $x \in (T/2, T)$, and also that $\max_{x\in\mathbb{R}} \sigma(x) = -\min_{x\in\mathbb{R}} \sigma(x)$. Then:*

1. *For any number W, we can find a mixed ReLU/σ network architecture η_W with W weights, and a corresponding weight assignment G_W, such that*
$$\sup_{f\in F_{r,d}} \|f - \widetilde{f}_{\eta_W,G_W}\|_\infty \leq \exp\big(-c_{r,d}W^{1/2}\big) \tag{4}$$
with some r, d-dependent constant $c_{r,d} > 0$.
2. *Moreover, the above architecture η_W has only one weight whose value depends on $f \in F_{r,d}$; for all other weights the assignment G_W is f-independent.*

In contrast to the previously considered power law rates (2), the rate (4) is exponential and corresponds to $p = \infty$, so that the ReLU-infeasible sector $p > \frac{2r}{d}$ is fully feasible for mixed ReLU/periodic networks. Moreover, statement 2 of the above theorem means that all information about the approximated function f can be encoded in a single network weight.

The sketch of proof of Theorem 6.1 is given in Section E, and details are provided in Section F. The main idea of the network design is to compute each digit of the output using a dynamical system controlled by the digits of the input. The faster rate can be interpreted as resulting from an efficient, dichotomy-based lookup that can be performed in networks including both ReLU and a periodic activation, see Fig.4.

It is well-known that some exotic activation functions allow to achieve rates even higher than those we have discussed. For example, a result of [21] based on the Kolmogorov Superposition Theorem ([1, p. 553]) shows the existence of a strictly increasing analytic activation function σ such that any $f \in C([0,1]^d)$ can be approximated with arbitrary accuracy by a three-layer σ-network with only $9d + 3$ units. However, in contrast to these results, our Theorem 6.1 holds for a very simple and general class of activation functions.

7 Distribution of information in the network

It is interesting to examine how information about the approximated function f is distributed in the network (see Table 1). The classical theorem of Kolmogorov [30] shows that the ϵ-entropy of the Hölder ball $F_{r,d}$ scales as $\epsilon^{-d/r}$ at small ϵ. This means that any family of networks achieving accuracy ϵ on this ball must include at least $\epsilon^{-d/r}$ bits of information about $f \in F_{r,d}$. This imposes

Approximation	Shallow ReLU	Deep ReLU	“Deep Fourier”
Rate (p)	$p = \frac{r}{d}$	$p \in (\frac{r}{d}, \frac{2r}{d}]$	$p = \infty$
Weight assignment	continuous	discontinuous	discontinuous
Network depth (L)	$\log(1/\epsilon)$	$\epsilon^{1/p-d/r}$	$\log(1/\epsilon)$
Number of weights, total (W)	$\epsilon^{-d/r}$	$\epsilon^{-1/p}$	$\log^2(1/\epsilon)$
Number of encoding weights	$\epsilon^{-d/r}$	$\epsilon^{-1/p}$	1
Bits / encoding weight	$\log(1/\epsilon)$	$\epsilon^{1/p-d/r}$	$\epsilon^{-d/r}\log(1/\epsilon)$

Table 1: Summary of the examined approximation modes. ϵ stands for the approximation accuracy $\|f - \widetilde{f}\|_\infty$ achieved uniformly on the Hölder ball $F_{r,d}$. The expressions in the bottom four rows show the orders of magnitude for various network characteristics w.r.t. ϵ.

constraints on the magnitude and/or precision of network weights: if the network is small and the weights have a limited space of values, the network simply cannot contain the necessary amount of information ([31, 8, 12]).

Classical linear models or “weakly nonclassical” models such as shallow ReLU networks contain $\epsilon^{-d/r}$ weights, and a weight precision of $O(\log(1/\epsilon))$ bits is sufficient to accomodate the total ϵ-entropy $\epsilon^{-d/r}$ ([12]). In contrast, the models in the “deep discontinuous ReLU” phase contain much fewer weights and accordingly need a much higher weight precision. Specifically, it follows from the proofs of Theorems 3.3 and 5.1 that the number of encoding weights in a network with rate $p \in (\frac{r}{d}, \frac{2r}{d}]$ is $\sim \epsilon^{-1/p}$, while each encoding weight must be specified with accuracy $c^{\epsilon^{1/p-d/r}}$ with some constant $c > 0$, i.e. must have $\sim \epsilon^{1/p-d/r}$ bits.

In the “deep Fourier” model, the encoding weight is unique. In the end of Section E we roughly estimate the information contained in this weight as $\epsilon^{-d/r}\log(1/\epsilon)$, again in agreement with the ϵ-entropy $\epsilon^{-d/r}$ of the Hölder ball $F_{r,d}$.

8 Discussion

Our results highlight tradeoffs between complexity of the network size and complexity of activations and/or arithmetic operations: the size can be decreased substantially at the cost of the other complexities. In addition to the increased precision of network operations, this requires the weight assignment to be discontinuous with respect to the fitted function f. While we do not discuss learning aspects in this paper, this discontinuity suggests that such networks should be hard to train by usual gradient-based methods, and would probably require other types of fitting algorithms.

The mentioned complexity tradeoffs are not unlimited: we have shown that for all piecewise polynomial activations the feasible rates span the sector $p \leq \frac{2r}{d}$. We do not know if this remains true for other standard nonpolynomial activations such as the standard sigmoid. This question seems to be essentially rooted in the optimality of Khovanskii’s fewnomial bounds, which is a long-standing problem in algebraic geometry [32, 33].

We have introduced the “deep Fourier” model – a hypothetical computational model assuming that the neurons can perfectly compute a periodic function of their inputs. This model allows to achieve exponential approximation rates while storing all information in a single weight. This result is purely theoretical; it doesn’t seem possible to implement such a model using practical technologies. Rather, we see the main interest of this result in the theoretical demonstration of a huge network size reduction compared to the usual shallow Fourier expansion, and in the associated novel bit extraction mechanism.

9 Broader impact

Not applicable.

10 Acknowledgments and Funding Transparency Statement

We thank Christoph Schwab for suggesting an extension of Theorem 6.1 to general periodic activations. We also thank the anonymous reviewers for several useful comments and suggestions. The research was not supported by third parties. The authors are not aware of any conflict of interest associated with this research.

References

[1] George G. Lorentz, Manfred v. Golitschek, and Yuly Makovoz. *Constructive Approximation (Advanced Problems)*. Springer, Berlin, 1996.

[2] Ronald A. DeVore, Ralph Howard, and Charles Micchelli. Optimal nonlinear approximation. *Manuscripta Mathematica*, 63:469–478, 12 1989.

[3] Paul W Goldberg and Mark R Jerrum. Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, 18(2-3):131–148, 1995.

[4] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. Cambridge university press, 2009.

[5] Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[6] Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

[7] Shiyu Liang and Rayadurgam Srikant. Why deep neural networks for function approximation? In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[8] Philipp Petersen and Felix Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108:296–330, 2018.

[9] Itay Safran and Ohad Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2979–2987. JMLR. org, 2017.

[10] Dmitry Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. In *Conference On Learning Theory*, pages 639–649, 2018.

[11] Peter L Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear VC-dimension bounds for piecewise polynomial networks. *Neural computation*, 10(8):2159–2173, 1998.

[12] Felix Voigtlaender and Philipp Petersen. Approximation in $L^p(\mu)$ with deep ReLU neural networks. In *2019 13th International conference on Sampling Theory and Applications (SampTA)*, pages 1–4. IEEE, 2019.

[13] Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

[14] 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.

[15] Matus Telgarsky. benefits of depth in neural networks. In *Conference on Learning Theory*, pages 1517–1539, 2016.

[16] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940, 2016.

[17] Johannes Schmidt-Hieber. Nonparametric regression using deep neural networks with ReLU activation function. *Annals of Statistics*, 48(4):1875–1897, 2020.

[18] Uri Shaham, Alexander Cloninger, and Ronald R Coifman. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537–557, 2018.

[19] Dmytro Perekrestenko, Philipp Grohs, Dennis Elbrächter, and Helmut Bölcskei. The universal approximation power of finite-width deep ReLU networks. *arXiv preprint arXiv:1806.01528*, 2018.

[20] Philipp Grohs, Dmytro Perekrestenko, Dennis Elbrächter, and Helmut Bölcskei. Deep neural network approximation theory. *arXiv preprint arXiv:1901.02220*, 2019.

[21] Vitaly Maiorov and Allan Pinkus. Lower bounds for approximation by mlp neural networks. *Neurocomputing*, 25(1-3):81–91, 1999.

[22] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and Pseudodimension Bounds for Piecewise Linear Neural Networks. *J. Mach. Learn. Res.*, 20:63–1, 2019.

[23] Boris Hanin and Mark Sellke. Approximating continuous functions by ReLU nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

[24] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, pages 6231–6239, 2017.

[25] Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. In *Advances in Neural Information Processing Systems*, pages 6169–6178, 2018.

[26] Matus Telgarsky. Neural networks and rational functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3387–3393. JMLR. org, 2017.

[27] A. G. Khovanskii. *Fewnomials*. Vol. 88 of Translations of Mathematical Monographs. American Mathematical Society, 1991.

[28] Marek Karpinski and Angus Macintyre. Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neural networks. *Journal of Computer and System Sciences*, 54(1):169–176, 1997.

[29] D. Jackson. *The theory of approximation*. Colloquium publications. The American mathematical society, 1930.

[30] A.N. Kolmogorov and V.M. Tikhomirov. ϵ-entropy and ϵ-capacity of sets in functional spaces. *A. M. S. Translations*, 17:277–364, 1961.

[31] Helmut Bölcskei, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. Memory-optimal neural network approximation. In *Wavelets and Sparsity XVII*, volume 10394, page 103940Q. International Society for Optics and Photonics, 2017.

[32] Bertrand Haas. A simple counterexample to Kouchnirenko's conjecture. *Beiträge zur Algebra und Geometrie*, 43(1):1–8, 2002.

[33] Alicia Dickenstein, J Maurice Rojas, Korben Rusek, and Justin Shih. Extremal real algebraic geometry and $\mathcal{A}$-discriminants. *Moscow Mathematical Journal*, 7(3):425–452, 2007.

A Theorem 3.3: proof details

We follow the paper [10] where Theorem 3.3 was proved for $r \leq 1$, and generalize it to arbitrary $r > 0$ using the strategy explained in Section 3. Given $p \in (\frac{r}{d}, \frac{2r}{s}]$ we show that it is possible to construct a network architecture with W weights and $L = O(W^{pd/r-1})$ layers which approximates every $f \in F_{r,d}$ with error $O(W^{-p})$. In Remark A.1 we deal with the case $p = \frac{r}{d}$.

We start by describing the space partition and related constructions. Then we give an overview of the network structure. Finally, we describe in more detail the network computation of the Taylor approximations, which is the main novel element of Theorem 3.3.

A.1 Space partitions

For an integer $N \geq 1$ we denote by $\mathcal{P}_N$ a standard triangulation of $\mathbb{R}^d$ into simplexes:

$$\Delta_{N,\mathbf{n},\rho} = \left\{ \mathbf{x} \in \mathbb{R}^d : 0 \leq x_{\rho(1)} - \frac{\mathbf{n}_{\rho(1)}}{N} \leq \cdots \leq x_{\rho(d)} - \frac{\mathbf{n}_{\rho(d)}}{N} \right\},$$

where $\mathbf{n} \in \mathbb{Z}^d$ and ρ is a permutation of d elements. The vertices of these simpixes are the points of the grid $(\mathbb{Z}/N)^d$. We call the set of all the vertices *the N-grid* and a particular vertex *an N-knot*. For an N-knot we call the union of simplexes it belongs to *an N-patch*. We denote a set of all N-knots $\mathbf{K}_N$.

Let $\phi : \mathbb{R}^d \to \mathbb{R}$ be the "spike" function defined as the continuous piecewise linear function such that:

1. ϕ is linear on every simplex from the triangulation $\mathcal{P}_1$;
2. $\phi(0) = 1$, $\phi(\mathbf{n}) = 0$ for all other $\mathbf{n} \in \mathbb{Z}^d$.

The function $\phi(\mathbf{x})$ can be computed by a feed-forward ReLU network with $O(d^2)$ weights (see [10, Section 4.2] for details). We treat d as a constant, so we can say that $\phi(\mathbf{x})$ can be computed by a network with a constant number of weights. Note that for integer N and $\mathbf{n} \in Z^d \cap [0, N]^d$, the function $\phi(N\mathbf{x} - \mathbf{n})$ is a continuous piecewise linear function which is linear in each simplex from $\mathcal{P}_N$, is equal to 1 at $\mathbf{x} = \frac{\mathbf{n}}{N}$, and vanishes at all other N-knots of $(\mathbb{Z}/N)^d$.

It is convenient to keep in mind two following simple propositions:

Proposition A.1. *Suppose we have K N-knots $\frac{\mathbf{n}_1}{N}, \ldots, \frac{\mathbf{n}_K}{N}$, $\mathbf{n}_i \in \mathbb{Z}^d$ and corresponding numbers $\ell_1, \ldots, \ell_K$. Then the function*

$$g(\mathbf{x}) = \sum_{k=1}^{K} \ell_k \phi(N\mathbf{x} - \mathbf{n}_k)$$

has the following properties:

1. *$g(\mathbf{x})$ is linear on each simplex from $\mathcal{P}_N$;*
2. *$g\left(\frac{\mathbf{n}_k}{N}\right) = \ell_k$ for $k = 1, \ldots N$. For other N-knots $\frac{\mathbf{n}}{N}$, h is zero: $h\left(\frac{\mathbf{n}}{N}\right) = 0$;*
3. *$g(\mathbf{x})$ can be computed exactly by a network with $O(K)$ weights and $O(1)$ layers.*

Proposition A.2. *Suppose we have K N-knots $\frac{\mathbf{n}_1}{N}, \ldots, \frac{\mathbf{n}_K}{N}$, $\mathbf{n}_i \in \mathbb{Z}^d$ and corresponding numbers $s_1, \ldots, s_K$. Suppose also that N-patches associated with $\frac{\mathbf{n}_1}{N}, \ldots, \frac{\mathbf{n}_K}{N}$ are disjoint. Then there exists function $h(\mathbf{x})$ with the following properties:*

1. *$h(\mathbf{x})$ is linear on each simplex from $\mathcal{P}_N$;*
2. *For $k = 1, \ldots N$, $h(\mathbf{x}) = s_k$ at an N-patch associated with $\frac{\mathbf{n}_i}{N}$;*
3. *$h(\mathbf{x})$ can be computed exactly by a network with $O(K)$ weights and $O(1)$ layers.*

Proof. Follows directly from Prop. A.1. We assign value s_k to all N-knots in N-patch associated with $\frac{\mathbf{n}_k}{N}$ and apply Prop. A.1. Since N-patches of interest are disjoint, each N-knot has at most one assigned value. □

A.2 The filtering subgrids

Given the total number of weights W, we set $N = W^{1/d}$. We will assume without loss of generality that N is integer. We consider triangulation $\mathcal{P}_N$ of $[0,1]^d$ on length scale $\frac{1}{N}$.

It is convenient to split the N-grid into 3^d disjoint subgrids with the $3\times$ grid spacing:

$$\mathbf{N_q} = \{\tfrac{\mathbf{n}}{N} : \mathbf{n} \in \big(\mathbf{q} + (3\mathbb{Z})^d\big) \cap [0,N]^d\}, \quad \mathbf{q} \in \{0,1,2\}^d.$$

Clearly, each subgrid contains $O(N^d)$ knots. Note that N-patches associated with N-knots in $\mathbf{N_q}$ are disjoint. It means, in particular, that any point $\mathbf{x} \in [0,1]^d$ lies in at most one such N-patch. It also means that Prop. A.2 is applicable to $\mathbf{N_q}$. We will use this observation in subsection A.3 for constructing an efficient approximation in a neighbourhood of $\mathbf{N_q}$ for a single $\mathbf{q}$. We call the union of these N-patches *a domain of* $\mathbf{N_q}$.

We compute the full approximation $\widetilde{f}$ as a sum

$$\widetilde{f}(\mathbf{x}) = \sum_{\mathbf{q}\in\{0,1,2\}^d} \widetilde{w}_{\mathbf{q}}(\mathbf{x})\widetilde{f}_{\mathbf{q}}(\mathbf{x}). \tag{5}$$

Function $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ computes $f(\mathbf{x})$ with error $O(W^{-p})$ for every $\mathbf{x}$ in the domain of $\mathbf{N_q}$. For $\mathbf{x}$ out of the domain of $\mathbf{N_q}$ it computes some garbage value. We describe $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ in subsection A.3. The final approximation $\widetilde{f}(\mathbf{x})$ is a weighted sum of $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ with weights $\widetilde{w}_{\mathbf{q}}(\mathbf{x})$. We choose such functions $\widetilde{w}_{\mathbf{q}}(\mathbf{x})$, that $\widetilde{w}_{\mathbf{q}}(\mathbf{x})$ vanishes outside the domain of $\mathbf{N_q}$ and

$$\sum_{\mathbf{q}\in\{0,1,2\}^d} \widetilde{w}_{\mathbf{q}}(\mathbf{x}) \equiv 1.$$

It follows that $\widetilde{f}(\mathbf{x})$ is a weighted sum (with weights with the sum 1) of terms approximating $f(\mathbf{x})$ with error $O(W^{-p})$. Consequently, $\widetilde{f}(\mathbf{x})$ approximates $f(x)$ with error $O(W^{-p})$.

Function $\widetilde{w}_{\mathbf{q}}(\mathbf{x})$ is given by applying Prop. A.1 to N-knots from $\mathbf{N_q}$ with all values $\ell_1, \ell_2, \ldots, \ell_{|\mathbf{N_q}|}$ equals to 1. Clearly, $\widetilde{w}_{\mathbf{q}}(\mathbf{x})$ vanishes outside the domain of $\mathbf{N_q}$. Sum $\sum_{\mathbf{q}\in\{0,1,2\}^d} \widetilde{w}_{\mathbf{q}}(\mathbf{x})$ is linear on each simplex from $\mathcal{P}_N$ and equals to 1 at all N-knots, because each N-knot belongs to exactly one set $\mathbf{N_q}$. Consequently, this sum equals to 1 for every $\mathbf{x} \in [0,1]^d$. It follows from Prop. A.1 that network implementing $\widetilde{w}_{\mathbf{q}}(\mathbf{x})$ has $O(N^d) = O(W)$ weights and $O(1)$ layers.

Multiplication $\widetilde{w}_{\mathbf{q}}(\mathbf{x})\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ is implemented approximately, with error $O(W^{-p})$, by network given by [6, Proposition 3] and requires $O(\log W)$ additional weights.

A.3 The approximation for a subgrid

Here we describe how we construct $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ for a single $\mathbf{q} \in \{0,1,2\}^d$. Remind that $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ computes accurate approximation for $f(\mathbf{x})$ only on the domain of $\mathbf{N_q}$.

For any N-knot $\frac{\mathbf{n}}{N}$ in $\mathbf{N_q}$ we consider a cube with center at $\frac{\mathbf{n}}{N}$ and edge $\frac{2}{N}$:

$$\left\{ \mathbf{x} \in \mathbb{R}^d : \max_{1\le i\le d} \left| \mathbf{x}_i - \frac{\mathbf{n}_i}{N} \right| \le \frac{1}{N} \right\}.$$

We call such cube *an N-cube* and denote it by $\mathbf{C_n}$. Note that $\mathbf{C_n} = \frac{\mathbf{n}}{N} + \mathbf{C_0}$.

Remind that the domain of $\mathbf{N_q}$ consists of $|\mathbf{N_q}|$ disjoint N-patches associated with N-knots from $\mathbf{N_q}$. Each $\mathbf{x}$ from the domain of $\mathbf{N_q}$ belongs to exactly one such N-patch. We call this patch *an N-patch for* $\mathbf{x}$ and associated N-knot *an N-knot for* $\mathbf{x}$. Let us denote an N-knot for $\mathbf{x}$ by $\frac{\mathbf{n_q(x)}}{N}$.

We set $M = W^{p/r}$. Note that $M^{-r} = W^{-p}$ and, therefore, we need to construct an approximation of error $O(M^{-r})$. We will assume without loss of generality that M is integer and M is divisible by N. Then $\mathcal{P}_M$ is a subpartition of $\mathcal{P}_N$. We define M-knot and M-patch similarly to N-knot and N-patch. We denote a set of all M-knots by $\mathbf{K}_M$. Note that there are $O\left((M/N)^d\right)$ M-knots in each N-patch and N-cube. See Fig.5 for an illustration of all described constructions.

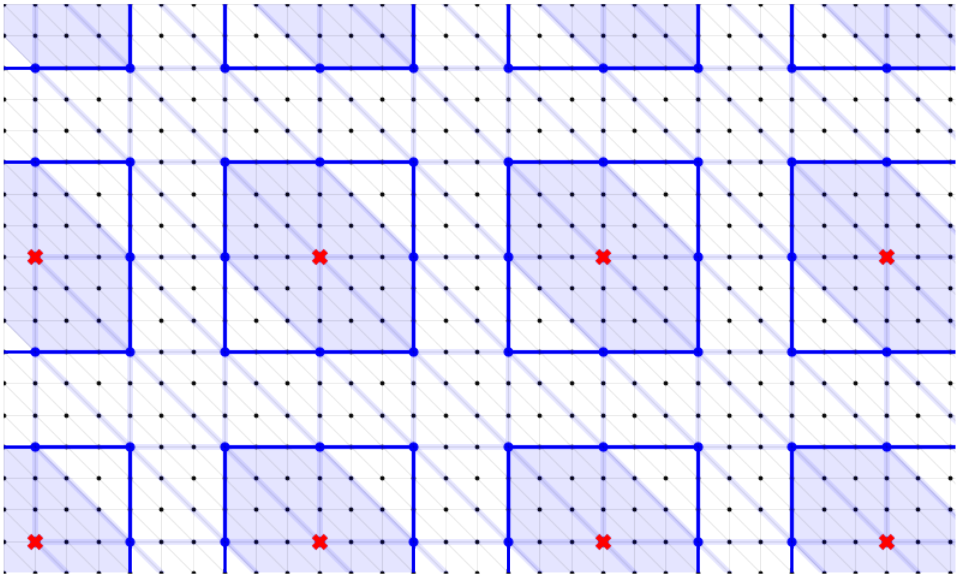


Figure 5: The partitions $\mathcal{P}_N$ and $\mathcal{P}_M$ for $d = 2$ and $\frac{M}{N} = 3$. The small black dots are the M-knots, and the thin black edges show the triangulation $\mathcal{P}_M$. The large blue dots are the N-knots; the light blue edges show the triangulation $\mathcal{P}_N$. The red crosses show the points of the subgrid $\mathbf{N_q}$. The filled blue region is the domain of $\mathbf{N_q}$. The bold blue squares show the N-cubes $\mathbf{C_n}$ for the points of $\mathbf{N_q}$.

Suppose that $\mathbf{x}$ lies in an M-patch associated with an M-knot $\frac{\mathbf{m}}{M}$. Consider a Taylor polynomial $P_{\mathbf{m}/M}(\mathbf{x})$ at $\frac{\mathbf{m}}{M}$ of order $\lceil r \rceil - 1$. Standard bounds for the remainder of Taylor polynomial imply that it approximates $f(\mathbf{x})$ with error $O(M^{-r})$ uniformly for $f \in F_{r,d}$. Taylor polynomial at $\frac{\mathbf{m}}{M}$ (and actually any polynomial) can be implemented with error $O(M^{-r})$ by a network with $O(\log M)$ weights and layers. We refer reader to [6, Proposition 3] and a proof of [6, Theorem 1] for details.

We can approximate $f(\mathbf{x})$ with error $O(M^{-r})$ with a weighted sum of Taylor polynomials $P_{\mathbf{m}/M}(\mathbf{x})$ at all M-knots:

$$\widetilde{f}(\mathbf{x}) = \sum_{\frac{\mathbf{m}}{M} \in \mathbf{K}_M} \phi\left(M\mathbf{x} - \mathbf{m}\right) P_{\mathbf{m}/M}(\mathbf{x}). \tag{6}$$

Note that $\phi\left(M\mathbf{x} - \mathbf{m}\right)$ vanishes outside an M-patch associated with $\frac{\mathbf{m}}{M}$ and

$$\sum_{\frac{\mathbf{m}}{M} \in \mathbf{K}_M} \phi\left(M\mathbf{x} - \mathbf{m}\right) \equiv 1.$$

There are M^d terms in (6) and calculating single term requires $O(\log M)$ weights. So, the total number of weights needed to implement (6) is $O(M^d \log M) = O(W^{pd/r} \log W)$. It is clearly infeasible for $p > \frac{r}{d}$. For $p = \frac{r}{d}$ it leads to approximation error $O(W^{-r/d} \log^{r/d} W)$ and makes a statement of [6, Theorem 1]. Note that in this construction Taylor coefficients at M-knots are the weights of network.

Note that terms of (6) are nonzero only for M-knots in an N-cube for $\mathbf{x}$. Suppose that $\mathbf{x}$ lies in the domain of $\mathbf{N_q}$ and, therefore, has well defined N-knot $\frac{\mathbf{n_q(x)}}{N}$. For such $\mathbf{x}$ we can write

$$\begin{aligned}\widetilde{f}_{\mathbf{q}}(\mathbf{x}) &= \sum_{\frac{\mathbf{m}}{M} \in \mathbf{K}_M \cap \mathbf{C}_{\mathbf{n_q}(x)}} \phi\left(M\mathbf{x} - \mathbf{m}\right) P_{\mathbf{m}/M}(\mathbf{x}) \\ &= \sum_{\frac{\mathbf{m}}{M} \in \mathbf{K}_M \cap \mathbf{C_0}} \phi\left(M\left(\mathbf{x} - \frac{\mathbf{n_q}(x)}{N}\right) - \mathbf{m}\right) P_{\mathbf{m}/M + \mathbf{n_q(x)}/N}(\mathbf{x})\end{aligned} \tag{7}$$

There are only $(M/N)^d = W^{pd/r-1}$ terms in (7). Therefore, if we know $\frac{\mathbf{n_q(x)}}{N}$ and Taylor coefficients for $\frac{\mathbf{m}}{M} + \frac{\mathbf{n_q(x)}}{N}$, then $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ can be implemented with error $O(M^{-r})$ by a network with $O\left((M/N)^d \log M\right) = O(W^{pd/r-1} \log W)$ weights.

For $\mathbf{x}$ in the domain of $\mathbf{N_q}$ it holds that $\widetilde{f}_{\mathbf{q}}(\mathbf{x}) = \widetilde{f}(\mathbf{x})$. It follows that $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ indeed approximates $f(\mathbf{x})$ with error $O(M^{-r}) = O(W^{-p})$ on the domain of $\mathbf{N_q}$.

If $\mathbf{x}$ lies in the domain of $\mathbf{N_q}$, then we can compute a single coordinate of $\frac{\mathbf{n_q}(x)}{N}$ with a network given by Prop. A.2. We need to take $\frac{\mathbf{n}_i}{N} \in \mathbf{N_q}$ and set s_i to be a corresponding coordinate of $\mathbf{n}_i$. We compute $\frac{\mathbf{n_q}(x)}{N}$ by applying this observation to all coordinates. Constructed network has $O(|\mathbf{N_q}|) = O(N^d) = O(W)$ weights and $O(1)$ layers.

In subsection A.4 we show, that (approximated) Taylor coefficients for $(M/N)^d$ M-knots $\frac{\mathbf{m}}{M} + \frac{\mathbf{n}}{N}$, $\frac{\mathbf{m}}{M} \in \mathbf{K}_M \cap \mathbf{C_0}$ can be computed by a network with $O\left((M/N)^d\right)$ weights and layers from $c_{r,d} \leq 2(d+1)^{\lceil r \rceil - 1}$ $\mathbf{n}$-dependent values. We call this values *encoding weights for* $\mathbf{n}$.

In subsection A.4 we describe how we construct encoding weights for a particular function f and an N-knot $\frac{\mathbf{n}}{N}$. We show that using approximated Taylor coefficients computed from encoding weights instead of real ones leads to error bounded by $O(M^{-r}) = O(W^{-p})$. For $\mathbf{x}$ in the domain of $\mathbf{N_q}$ we can calculate encoding weights for $\mathbf{n_q}(\mathbf{x})$ by a network given by Prop. A.2.

Let us finalize a structure of network computing $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$. For $\mathbf{x}$ in the domain of $\mathbf{N_q}$ it

1. Computes $\mathbf{n_q}(\mathbf{x})$ and encoding weights for $\mathbf{n}_q(\mathbf{x})$. This step is implemented by applying Prop. A.2 and requires $O(N^d) = O(W)$ weights and $O(1)$ layers;

2. Given encoding weights for $\mathbf{n}_q(x)$, computes (approximated) Taylor coefficients for all M-knots $\frac{\mathbf{m}}{M} + \frac{\mathbf{n_q(x)}}{N}$, $\frac{\mathbf{m}}{M} \in \mathbf{K}_M \cap \mathbf{C_0}$. This step requires a network with $O\left((M/N)^d\right) = O(W^{pd/r-1})$ weights and layers and described in subsection A.4;

3. Given (approximated) Taylor coefficients achieved at the previous step, computes an approximation for $P_{\mathbf{m}/M + \mathbf{n_q(x)}/N}$ for all M-knots $\frac{\mathbf{m}}{M} + \frac{\mathbf{n_q(x)}}{N}$, $\frac{\mathbf{m}}{M} \in \mathbf{K}_M \cap \mathbf{C_0}$. The approximation with error $O(M^{-r}) = O(W^{-p})$ for a single $P_{\mathbf{m}/M + \mathbf{n_q(x)}/N}$ can be implemented by a network with $O(\log M) = O(\log W)$ weights and layers. Total number of weights needed at this step is, therefore, $O\left(|\mathbf{K}_M \cap \mathbf{C_0}| \log M\right) = O\left((M/N)^d \log M\right) = O(W^{pd/r-1} \log W)$. Computation for different M-knots can be done in parallel, so the total number of layers is still $O(\log W)$;

4. Given $\mathbf{n_q}(\mathbf{x})$, computed at first step, computes $\phi\left(M\left(\mathbf{x} - \frac{\mathbf{n_q}(x)}{N}\right) - \mathbf{m}\right)$ for all M-knots $\frac{\mathbf{m}}{M} + \frac{\mathbf{n_q(x)}}{N}$, $\frac{\mathbf{m}}{M} \in \mathbf{K}_M \cap \mathbf{C_0}$. It requires $O\left(|\mathbf{K}_M \cap \mathbf{C_0}|\right) = O\left((M/N)^d\right) = O(W^{pd/r-1})$ weights and $O(1)$ layers;

5. Combines outputs of steps 3 and 4 in the final approximation with (7). Multiplication with accuracy $O(M^{-r})$ can be implemented by a network with $O(\log M)$ weights and layers, so this step requires $O(|\mathbf{K}_M \cap \mathbf{C_0}| \log M) = O\left((M/N)^d \log M\right) = O(W^{pd/r-1} \log W)$ weights and $O(\log M) = O(\log W)$ layers.

Clearly we can pass forward values achieved at early steps without increasing an asymptotic for needed number of weights and layers.

If we sum up the total number of weights needed at each step, we obtain $O\left(W + W^{pd/r-1} \log W\right)$. For $\frac{r}{d} < p < \frac{2r}{d}$ it is equivalent to $O(W)$ and matches the desired approximation rate. For $p = \frac{2r}{d}$ it is equivalent to $O(W \log W)$ and leads to the desired approximation rate up to a logarithmic factor. We show how to deal with it in subsection A.5.

The total number of needed layers is $O(W^{pd/r-1})$ and matches the desired.

A.4 Encoding and decoding Taylor coefficients

It is known that $\sim \epsilon^{-d/r}$ bits are needed to specify a function $f \in F_{r,d}$ with accuracy ϵ [30]. It follows from the bounds for Kolmogorov ε-entropy of $F_{r,d}$ derived in [30, § 4]. Here we describe how this specification can be implemented by a neural network.

First we introduce some notation. Suppose we have an M-knot $\frac{\mathbf{m}}{M}$. Taylor expansion $P_{\mathbf{m}/M}(\mathbf{x})$ of $f(\mathbf{x})$ at $\frac{\mathbf{m}}{M}$ is given by

$$P_{\mathbf{m}/M}(\mathbf{x}) = \sum_{\mathbf{k}:|\mathbf{k}|\leq\lceil r\rceil-1} \frac{D^{\mathbf{k}} f\left(\frac{\mathbf{m}}{M}\right)}{\mathbf{k}!}\left(\mathbf{x} - \frac{\mathbf{m}}{M}\right)^{\mathbf{k}}.$$

We use usual convention $\mathbf{k}! = \prod_{i=1}^{d} k_i$ and $\left(\mathbf{x} - \frac{\mathbf{m}}{M}\right)^{\mathbf{k}} = \prod_{i=1}^{d}\left(x_i - \frac{m_i}{M}\right)^{k_i}$. We denote

$$a_{\mathbf{m},\mathbf{k}} = D^{\mathbf{k}} f\left(\frac{\mathbf{m}}{M}\right).$$

We denote an approximated Taylor coefficients to be defined further in this section by $\widehat{a}_{\mathbf{m},\mathbf{k}}$. Corresponding approximated Taylor expansion is given by

$$\widehat{P}_{\mathbf{m}/M}(\mathbf{x}) = \sum_{\mathbf{k}:|\mathbf{k}|\leq\lceil r\rceil-1} \frac{\widehat{a}_{\mathbf{m},\mathbf{k}}}{\mathbf{k}!}\left(\mathbf{x} - \frac{\mathbf{m}}{M}\right)^{\mathbf{k}}.$$

For any $\mathbf{x}$ in the M-patch associated with $\frac{\mathbf{m}}{M}$

$$\left|f(\mathbf{x}) - P_{\mathbf{m}/M}(\mathbf{x})\right| \leq c_{r,d} M^{-r},$$

for all $f \in F_{r,d}$ and some constant $c_{r,d}$, which does not depend on M and $\mathbf{m}$.

We first show how we construct encoding weights associated with an N-knot $\frac{\mathbf{n}}{N}$. Our construction is quite similar to one from the proof of [30, Theorem XIV], where bounds for Kolmogorov ε-entropy of $F_{r,d}$ were derived. Then we discuss how approximated Taylor coefficients at M-knots in the N-cube $\mathbf{C_n}$ are computed from encoding weights by a network.

Our goal is to construct such approximated Taylor coefficients $\widehat{a}_{\mathbf{m},\mathbf{k}}$, that for any $\mathbf{x}$ in the M-patch associated with $\frac{\mathbf{m}}{M}$ holds $|\widehat{P}_{\mathbf{m}/M}(\mathbf{x}) - P_{\mathbf{m}/M}(\mathbf{x})| \leq c_{r,d} M^{-r}$ for some M-independent constant $c_{r,d}$. The following proposition states sufficient condition on such $\widehat{a}_{\mathbf{m},\mathbf{k}}$.

Proposition A.3. *Suppose that*

$$|a_{\mathbf{m},\mathbf{k}} - \widehat{a}_{\mathbf{m},\mathbf{k}}| \leq M^{|\mathbf{k}|-r} \quad \forall\, \mathbf{k} : |\mathbf{k}| \leq \lceil r\rceil - 1. \tag{8}$$

Then for any $\mathbf{x}$ in an M-patch associated with $\frac{\mathbf{m}}{M}$

$$\left|\widehat{P}_{\mathbf{m}/M}(\mathbf{x}) - P_{\mathbf{m}/M}(\mathbf{x})\right| \leq (d+1)^{\lceil r\rceil-1} M^{-r}.$$

Proof.

$$\begin{aligned}\left|\widehat{P}_{\mathbf{m}/M}(\mathbf{x}) - P_{\mathbf{m}/M}(\mathbf{x})\right| &\leq \sum_{\mathbf{k}:|\mathbf{k}|\leq\lceil r\rceil-1} \frac{1}{\mathbf{k}!}\left|\widehat{a}_{\mathbf{m},\mathbf{k}} - a_{\mathbf{m},\mathbf{k}}\right| \left|\left(\mathbf{x} - \frac{\mathbf{m}}{M}\right)^{\mathbf{k}}\right| \\ &\leq \sum_{\mathbf{k}:|\mathbf{k}|\leq\lceil r\rceil-1} M^{|\mathbf{k}|-r} M^{-|\mathbf{k}|} \\ &\leq (d+1)^{\lceil r\rceil-1} M^{-r}.\end{aligned}$$

□

Suppose that two M-knots $\frac{\mathbf{m}_1}{M}$ and $\frac{\mathbf{m}_2}{M}$ are adjacent and we have $\widehat{a}_{\mathbf{m}_1,\widehat{\mathbf{k}}}$, $|\widehat{\mathbf{k}}| \leq \lceil r\rceil - 1$ satisfying (8). Another convenient proposition we use further shows how to construct an accurate approximation for Taylor coefficients at $\frac{\mathbf{m}_2}{M}$.

Proposition A.4. *Suppose that two M-knots $\frac{\mathbf{m}_1}{M}$ and $\frac{\mathbf{m}_2}{M}$ are adjacent. Suppose that approximated Taylor coefficients $\widehat{a}_{\mathbf{m}_1,\widehat{\mathbf{k}}}$, $|\widehat{\mathbf{k}}| \le \lceil r \rceil - 1$ at $\frac{\mathbf{m}_1}{M}$ satisfy* (8). *Then we can find such $c_{\mathbf{k},\widehat{\mathbf{k}}}$ and $\widetilde{a}_{m_2,\mathbf{k}}$, $|\mathbf{k}|, |\widehat{\mathbf{k}}| \le \lceil r \rceil - 1$, that*

1. *For all* $\mathbf{k} : |\mathbf{k}| \le \lceil r \rceil - 1$
$$\widetilde{a}_{m_2,\mathbf{k}} = \sum_{\widehat{\mathbf{k}}:|\widehat{\mathbf{k}}|\le\lceil r\rceil-1} c_{\mathbf{k},\widehat{\mathbf{k}}} \cdot \widehat{a}_{\mathbf{m}_1,\widehat{\mathbf{k}}};$$
2. *For all* $\mathbf{k} : |\mathbf{k}| \le \lceil r \rceil - 1$
$$|a_{\mathbf{m}_2,\mathbf{k}} - \widetilde{a}_{\mathbf{m}_2,\mathbf{k}}| < 4M^{|\mathbf{k}|-r}; \tag{9}$$
3. *Coefficients $c_{\mathbf{k},\widehat{\mathbf{k}}}$ depend only on the relative position of $\frac{\mathbf{m}_1}{M}$ and $\frac{\mathbf{m}_2}{M}$.*

Proof. Remind that M-knots $\frac{\mathbf{m}_1}{M}$ and $\frac{\mathbf{m}_2}{M}$ are adjacent. Let us consider first component of $\mathbf{m}_1$ and $\mathbf{m}_2$ independently and assume without loss of generality that $\mathbf{m}_1 = (m_1, \overline{\mathbf{m}})$ and $\mathbf{m}_2 = (m_1 + 1, \overline{\mathbf{m}})$.

Standard bounds for a remainder of Taylor series partial sum imply, that for any $\mathbf{k} = (k_1, \dots, k_d)$ and $f \in F_{r,d}$

$$\left| D^{(k_1,\dots,k_d)} f\left(\frac{\mathbf{m}_2}{M}\right) - \sum_{n=0}^{\lceil r\rceil-1-|\mathbf{k}|} \frac{D^{(k_1+n,\dots,k_d)} f\left(\frac{\mathbf{m}_1}{M}\right)}{n!} \cdot \frac{1}{M^n} \right| \le M^{|\mathbf{k}|-r}.$$

In our notation

$$\left| a_{\mathbf{m}_2,(k_1,\dots,k_d)} - \sum_{n=0}^{\lceil r\rceil-1-|\mathbf{k}|} \frac{a_{\mathbf{m}_1,(k_1+n,\dots,k_d)}}{n!} \cdot \frac{1}{M^n} \right| \le M^{|\mathbf{k}|-r}. \tag{10}$$

From the proposition that coefficients $\widehat{a}_{\mathbf{m}_1,\mathbf{k}}$ satisfy (8) it follows that

$$\begin{aligned} \left| \sum_{n=0}^{\lceil r\rceil-1-|\mathbf{k}|} \frac{\left(a_{\mathbf{m}_1,(k_1+n,\dots,k_d)} - \widehat{a}_{\mathbf{m}_1,(k_1+n,\dots,k_d)}\right)}{n!} \cdot \frac{1}{M^n} \right| &\le \sum_{n=0}^{\lceil r\rceil-1-|\mathbf{k}|} \frac{M^{|\mathbf{k}|+n-r}}{n!} \cdot \frac{1}{M^n} \\ &= M^{|\mathbf{k}|-r} \sum_{n=0}^{\lceil r\rceil-1-|\mathbf{k}|} \frac{1}{n!} \\ &< eM^{|\mathbf{k}|-r} < 3M^{|\mathbf{k}|-r}. \end{aligned} \tag{11}$$

Combining (10) and (11) we obtain

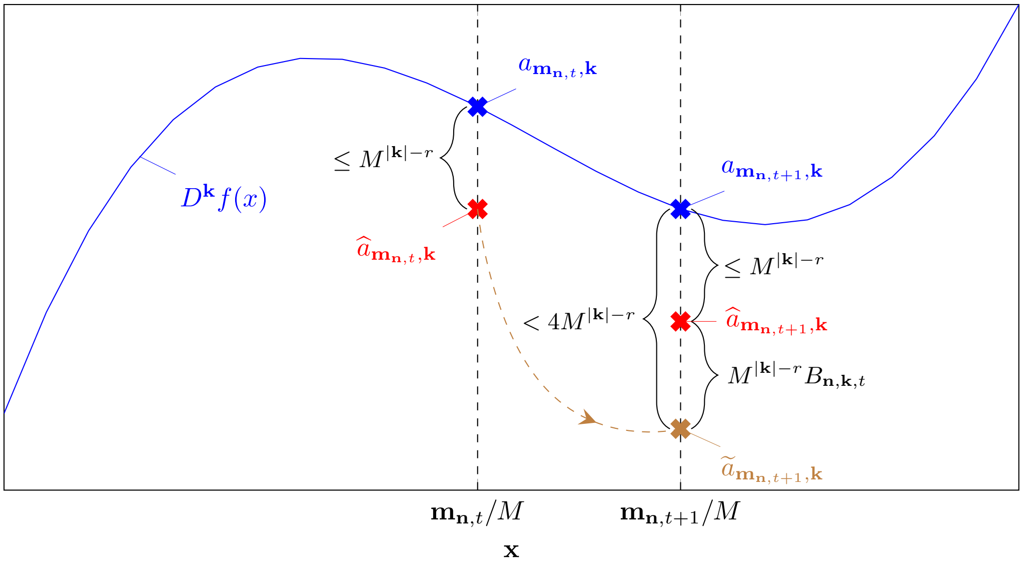
$$\left| a_{\mathbf{m}_2,(k_1,\dots,k_d)} - \sum_{n=0}^{\lceil r\rceil-1-|\mathbf{k}|} \frac{\widehat{a}_{\mathbf{m}_1,(k_1+n,\dots,k_d)}}{n!} \cdot \frac{1}{M^n} \right| < 4M^{|\mathbf{k}|-r}.$$

It follows that if for each $\mathbf{k} = (k_1, \dots, k_d)$ we set

$$\widetilde{a}_{\mathbf{m}_2,(k_1,\dots,k_d)} = \sum_{n=0}^{\lceil r\rceil-1-|\mathbf{k}|} \frac{\widehat{a}_{\mathbf{m}_1,(k_1+n,\dots,k_d)}}{n!} \cdot \frac{1}{M^n}, \tag{12}$$

then $\widetilde{a}_{\mathbf{m}_2,\mathbf{k}}$ satisfy (9). It remains to note that coefficients in (12) depend only on the relative position of $\frac{\mathbf{m}_1}{M}$ and $\frac{\mathbf{m}_2}{M}$, but not on $f \in F_{r,d}$, values $\widehat{a}_{\mathbf{m}_1,\mathbf{k}}$ or M-knots $\frac{\mathbf{m}_1}{M}$ and $\frac{\mathbf{m}_2}{M}$ themselves. □

Now we are ready to describe how we find $\widehat{a}_{\mathbf{m},\mathbf{k}}$ for all M-knots $\frac{\mathbf{m}}{M}$ from a given N-cube $\mathbf{C}_\mathbf{n}$. We enumerate M-knots lying in $\mathbf{C}_\mathbf{n}$ with numbers $t = 1, \dots, (2M/N + 1)^d$ and denote them $\frac{\mathbf{m}_{\mathbf{n},t}}{M}$. We inductively construct $\widehat{a}_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$ satisfying (8) for all M-knots $\frac{\mathbf{m}_{\mathbf{n},t}}{M}$. We choose such an enumeration, that two consequent M-knots are adjacent.



$a_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$
$\leq M^{|\mathbf{k}|-r}$
$D^{\mathbf{k}}f(x)$
$\widehat{a}_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$
$a_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$
$\leq M^{|\mathbf{k}|-r}$
$< 4M^{|\mathbf{k}|-r}$
$\widehat{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$
$M^{|\mathbf{k}|-r}B_{\mathbf{n},\mathbf{k},t}$
$\widetilde{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$
$\mathbf{m}_{\mathbf{n},t}/M$
$\mathbf{m}_{\mathbf{n},t+1}/M$
$\mathbf{x}$

Figure 6: An illustration of determining approximated Taylor coefficients at $\frac{\mathbf{m}_{\mathbf{n},t+1}}{M}$ from known approximated Taylor coefficients at $\frac{\mathbf{m}_{\mathbf{n},t}}{M}$. The blue line is $D^{\mathbf{k}}f(x)$ and the blue crosses are its values $a_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$ and $a_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$ at M-knots $\frac{\mathbf{m}_{\mathbf{n},t}}{M}$ and $\frac{\mathbf{m}_{\mathbf{n},t+1}}{M}$ respectively. Red crosses are desired approximations $\widehat{a}_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$ and $\widehat{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$ for $a_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$ and $a_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$ satisfying (8). Given $\widehat{a}_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$, we first apply Prop. A.4 to get $\widetilde{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$ satisfying (9). This step is illustrated by the brown dashed arrow and brown cross is $\widetilde{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$. Then we choose such $B_{\mathbf{n},\mathbf{k},t} \in \{-3,\ldots,3\}$, that $\widehat{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}} = \widetilde{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}} + M^{|\mathbf{k}|-r}B_{\mathbf{n},\mathbf{k},t}$ satisfy (8).

We set $\widehat{a}_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}} = a_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}}$. Such $\widehat{a}_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}}$ clearly satisfy (8). Suppose that we have constructed $\widehat{a}_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$ satisfying (8). Since M-knots $\frac{\mathbf{m}_{\mathbf{n},t}}{M}$ and $\frac{\mathbf{m}_{\mathbf{n},t+1}}{M}$ are adjacent, we can apply Prop. A.4 to get $\widetilde{a}_{m_{\mathbf{n},t+1},\mathbf{k}}$, $|\mathbf{k}| \leq \lceil r \rceil - 1$ satisfying (9). It follows that there exist such integers $B_{\mathbf{n},\mathbf{k},t}$, that $|B_{\mathbf{n},\mathbf{k},t}| \leq 3$ and

$$\left| a_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}} - \widetilde{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}} - M^{|\mathbf{k}|-r}B_{\mathbf{n},\mathbf{k},t} \right| \leq M^{|\mathbf{k}|-r}.$$

We set

$$\widehat{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}} = \widetilde{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}} + M^{|\mathbf{k}|-r}B_{\mathbf{n},\mathbf{k},t}. \tag{13}$$

Then coefficients $\widehat{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$ satisfy (8) as desired. See Fig.6 for an illustration of algorithm of determining $\widehat{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$.

For a single $\mathbf{k}$ we encode $(2M/N+1)^d$ values $B_{\mathbf{n},\mathbf{k},t}$ by a single base-7 number $b_{\mathbf{n},\mathbf{k}}$

$$b_{\mathbf{n},\mathbf{k}} = \sum_{t=1}^{(2M/N+1)^d} 7^{-t}\left(B_{\mathbf{n},\mathbf{k},t} + 3\right)$$

Numbers $b_{\mathbf{n},\mathbf{k}}$ and $\widehat{a}_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}} = a_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}}$ are encoding weights for $\mathbf{n}$. There are $c_{r,d} \leq 2(d+1)^{\lceil r \rceil - 1}$ encoding weights.

Now we describe how a network reconstruct all $\widehat{a}_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$ from encoding weights. Numbers $B_{\mathbf{n},\mathbf{k},t}$ can be reconstructed from $b_{\mathbf{n},\mathbf{k}}$ by a ReLU network with $O\left((M/N)^d\right)$ weights and layers. We refer to [10, 5.2.2], where similar reconstruction is described for ternary numbers. Given $\widehat{a}_{\mathbf{m}_{\mathbf{n},t},\mathbf{k}}$ and $B_{\mathbf{n},\mathbf{k},t}$, we first compute $\widetilde{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$ with (12) and then we compute $\widehat{a}_{\mathbf{m}_{\mathbf{n},t+1},\mathbf{k}}$ with (13). We need $O(1)$ weights and layers at each step, so the total number of needed weights and layers is $O\left((M/N)^d\right)$.

For given $\mathbf{x} \in [0,1]^d$ and $\mathbf{q} \in \{0,1,2\}^d$ we obtain encoding weights for $\mathbf{n}_{\mathbf{q}}(\mathbf{x})$ by applying Prop. A.2. Note that Prop. A.4 implies that coefficients in (12) depend only on the relative position of M-knots

$\frac{\mathbf{m}_{\mathbf{n},t}}{M}$ and $\frac{\mathbf{m}_{\mathbf{n},t+1}}{M}$. It follows that if we choose similar enumeration of M-knots for all N-cubes $\mathbf{C}_\mathbf{n}$, $\frac{\mathbf{n}}{N} \in \mathbf{N}_\mathbf{q}$, then we can use a network described in previous paragraph for all possible values of $\mathbf{n}_\mathbf{q}(\mathbf{x})$.

Note that encoding weights $b_{\mathbf{n},\mathbf{k}}$ can be represented as $\sim (M/N)^d$-bits numbers while encoding weights $\widehat{a}_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}} = a_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}}$ can be arbitrary real numbers. Remind that described construction requires $|\widehat{a}_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}} - a_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}}| \sim M^{|\mathbf{k}|-r}$. It follows that if we want to encode $\widehat{a}_{\mathbf{m}_{\mathbf{n},1},\mathbf{k}}$ by a finite number of bits as well, then we need $\sim \log M$ additional bits to achieve desired accuracy.

A.5 Getting rid of logarithmic factor

Remind that logarithmic factor arises in the construction described in A.3 in case $p = \frac{2r}{d}$. This is because we construct $\widetilde{f}_\mathbf{q}(\mathbf{x})$ in form (7) with $O(W)$ terms and we need $O(\log W)$ weights to implement an approximated Taylor sum arising in each term.

Note that for a particular $\mathbf{x}$ most terms in (7) vanishes since $\phi(M(\mathbf{x} - \frac{\mathbf{n}_\mathbf{q}(x)}{N}) - \mathbf{m}) = 0$ and there is no need to compute $P_{\mathbf{m}/M + \mathbf{n}_\mathbf{q}(\mathbf{x})/N}(\mathbf{x})$ for such terms. If we perform Taylor sum calculation for only a constant number of non-vanishing terms, then the total number of needed weights reduces to $O(W + \log W)$. We can apply technique used in A.2 for detecting nonvanishing terms from input $\mathbf{x}$.

We split all M-knots lying in N-cube $\mathbf{C}_\mathbf{0}$ into a disjoint union of 3^d sets

$$\mathbf{M}_\mathbf{s} = \{\tfrac{\mathbf{m}}{M} : \mathbf{m} \in \big(\mathbf{s} + (3\mathbb{Z})^d\big) \cap \mathbf{C}_\mathbf{0}\}, \quad \mathbf{s} \in \{0,1,2\}^d.$$

M-patches associated with M-knots in $\mathbf{M}_\mathbf{s}$ are disjoint. We call their union the domain of $\mathbf{M}_\mathbf{s}$. If $\mathbf{x} - \frac{\mathbf{n}_\mathbf{q}(\mathbf{x})}{N}$ lies in the domain of $\mathbf{M}_\mathbf{s}$, there is exactly one such $\frac{\mathbf{m}_{\mathbf{q},\mathbf{s}}(\mathbf{x})}{M}$, that $\mathbf{x} - \frac{\mathbf{n}_\mathbf{q}(\mathbf{x})}{N}$ lies in the M-patch associated with $\frac{\mathbf{m}_{\mathbf{q},\mathbf{s}}}{M}$. We can rewrite (7) as

$$\widetilde{f}_\mathbf{q}(\mathbf{x}) = \sum_{\mathbf{s} \in \{0,1,2\}^d} \left[\widetilde{f}_{\mathbf{q},\mathbf{s}}(\mathbf{x}) \sum_{\frac{\mathbf{m}}{M} \in \frac{\mathbf{n}_\mathbf{q}(\mathbf{x})}{N} + \mathbf{M}_\mathbf{s}} \phi\left(M\left(\mathbf{x} - \tfrac{\mathbf{n}_\mathbf{q}(x)}{N}\right) - \mathbf{m}\right) \right]. \tag{14}$$

Here $\widetilde{f}_{\mathbf{q},\mathbf{s}}(\mathbf{x})$ is a function, which calculates $P_{\mathbf{m}_{\mathbf{q},\mathbf{s}}(\mathbf{x})/M + \mathbf{n}_\mathbf{q}/N}(\mathbf{x})$ if $\mathbf{x} - \frac{\mathbf{n}_\mathbf{q}(\mathbf{x})}{N}$ lies in the domain of $\mathbf{M}_\mathbf{s}$, and some garbage value otherwise. We also require that $\widetilde{f}_{\mathbf{q},\mathbf{s}}(\mathbf{x})$ computes an approximation for a Taylor series partial sum only once. The total number of partial sums computed by network implementing $\widetilde{f}_\mathbf{q}(\mathbf{x})$ in form (14) is therefore reduced to 3^d. The total number of weights needed to implement $\widetilde{f}_\mathbf{q}(\mathbf{x})$ reduces from $O(W \log W)$ to $O(W)$.

To compute such $\widetilde{f}_{\mathbf{q},\mathbf{s}}(\mathbf{x})$ we only need to determine approximated Taylor coefficients for $\frac{\mathbf{m}_{\mathbf{q},\mathbf{s}}(\mathbf{x})}{M} + \frac{\mathbf{n}_\mathbf{q}(\mathbf{x})}{N}$ among all coefficients. For each $\frac{\mathbf{m}}{M} \in \mathbf{M}_\mathbf{s}$ we construct function $\widehat{w}_{\mathbf{s},\mathbf{m}}(\mathbf{x})$, which equals to 1 in the M-patch associated with $\frac{\mathbf{m}}{M}$ and vanishes in other patches of the domain of $\mathbf{M}_\mathbf{s}$. Knowing values $\widehat{w}_{\mathbf{s},\mathbf{m}}(\mathbf{x} - \frac{\mathbf{n}_\mathbf{q}(\mathbf{x})}{N})$ we clearly can get Taylor coefficients for $\frac{\mathbf{m}_{\mathbf{q},\mathbf{s}}(\mathbf{x})}{M} + \frac{\mathbf{n}_\mathbf{q}(\mathbf{x})}{N}$ from all Taylor coefficients computed by network.

Remark A.1. *Similar reasoning can be applied to the case $p = \frac{r}{d}$. In this case we do not consider an M-grid at all, but we still can split N-grid into 3^d disjoint sets and compute approximated Taylor sum once for each set. In this case weight assignment map is continuous and even linear on f.*

B Theorem 4.1: proof details

We follow the network construction used in the proof of Theorem 3.3 and described in Subsections A.2,A.3. We want to show that this construction can be realized within a ReLU network of width $2d + 10$. As explained in Section 4, we slightly modify the construction, so that we don't update the Taylor coefficients at new M-patches, but rather compute them afresh. This will give a slight increase in the size of the network. Accordingly, we define parameters N, M in terms of the required accuracy ϵ rather than the number of weights: specifically, we set $M = \epsilon^{-1/r}$ and $N = \epsilon^{-1/(2r)}$.

Following [10], we think of the width-$(2d+10)$ network as $2d+10$ "channels" that are interconnected and can exchange information. We reserve d channels for passing forward the scalar components of the input vector $\mathbf{x}$ and one channel for accumulating the approximation $\widetilde{f}(\mathbf{x})$. The other channels are used for intermediate computations.

The first step in computing the approximation $\widetilde{f}(\mathbf{x})$ is the finite decomposition 5 of $\widetilde{f}$ over $\mathbf{q}$-subgrids. The decomposition can be implemented in the width-$(2d+10)$ network in the serial fashion, so we only need to consider computation of a single term $\widetilde{w}_{\mathbf{q}}(\mathbf{x})\widetilde{f}_{\mathbf{q}}(\mathbf{x})$.

The weight $\widetilde{w}_{\mathbf{q}}(\mathbf{x})$ is just a linear combination of $O(N^d)$ functions $\phi(N\mathbf{x}-\mathbf{n})$, and ϕ can be computed by a constant-size chain of linear and ReLU operations (see [10, Section 4.2]). Thus, $\widetilde{w}_{\mathbf{q}}(\mathbf{x})$ can be computed by a subnetwork using just 2 channels and depth $O(\epsilon^{-d/(2r)})$. On the other hand, we will show below that $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ can be computed by a subnetwork using $d+8$ channels and depth $O(\epsilon^{-d/(2r)}\log(1/\epsilon))$. We can then pass the values $\widetilde{w}_{\mathbf{q}}(\mathbf{x})$ and $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ to the third subnetwork computing an $O(\epsilon)$-approximation to the product $\widetilde{w}_{\mathbf{q}}(\mathbf{x})\widetilde{f}_{\mathbf{q}}(\mathbf{x})$. This approximate product can be computed by a width-4 subnetwork of depth $O(\log(1/\epsilon))$ (see [6, Proposition 3]). Thus the total computation of the term $\widetilde{w}_{\mathbf{q}}(\mathbf{x})\widetilde{f}_{\mathbf{q}}(\mathbf{x})$, and hence of the whole approximation $\widetilde{f}(\mathbf{x})$ can be done with necessary accuracy ϵ within the width-$(2d+10)$ network of depth $L=O(\epsilon^{-d/(2r)}\log(1/\epsilon))$. By inverting this relation, we get $\epsilon=O(L^{-2r/d}\log^{2r/d}L)$, as desired.

We return now to the computation of $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$. It is based on the expansion (7) and can be performed as described later in that subsection. We examine now indivudual steps and how they can be implemented in our fixed-depth network.

1. The N-knot positions $\mathbf{n}_{\mathbf{q}}(\mathbf{x})$ associated with $\mathbf{x}$ are computed using a linear combination of $O((M/N)^d)$ functions of the form $\phi(N\mathbf{x}-\mathbf{n}_k)$. This computation can be performed in a subnetwork of width 2 and depth $O(\epsilon^{-d/2r})$. We reserve d channels to pass forward the scalar components of $\mathbf{n}_{\mathbf{q}}(\mathbf{x})$. Additionally, we reserve one channel for passing forward the encoding weight corresponding to this $\mathbf{n}_{\mathbf{q}}(\mathbf{x})$. The encoding weight gets transformed as it passes along the network and bits get decoded from it. Additional 3 channels are sufficient for bit decoding (see [10] for a description of the decoding procedure).

2. We traverse the $O((M/N)^d)$ M-knots of the N-patch corresponding to $\mathbf{n}_{\mathbf{q}}$ and decode from the encoding weight the Taylor coefficients of degree up $\lceil r\rceil-1$ at these knots. It is sufficient to know these coefficients with precision $O(\epsilon^r)$, so each Taylor coefficient can be encoded by $K_{\max}=O(\log(1/\epsilon))$ bits $\{b_k\}_{k=0}^{K_{\max}}$, and reconstructed by accumulating the linear combination $\sum_{k=0}^{K_{\max}}2^{-k}b_k$. Thus, the total required number of bits in the encoding weight is $O(\epsilon^{-d/(2r)})\log(1/\epsilon)$. Also, all the necessary coefficients can be reconstructed using $O(\epsilon^{-d/(2r)})\log(1/\epsilon)$ layers of width 4.

3. At each M-knot $\mathbf{m}/M+\mathbf{n}_{\mathbf{q}}(\mathbf{x})/N$ in the N-patch, we compute the respective Taylor polynomial $P_{\mathbf{m}/M+\mathbf{n}_{\mathbf{q}}(\mathbf{x})/N}(\mathbf{x})=\sum_{\mathbf{k}:|\mathbf{k}|\le\lceil r\rceil-1}a_{\mathbf{k}}(\mathbf{x}-(\mathbf{m}/M+\mathbf{n}_{\mathbf{q}}(\mathbf{x})/N))^{\mathbf{k}}$. The values of $\mathbf{x}$ and $\mathbf{n}_{\mathbf{q}}(\mathbf{x})$ are provided from the reserved channels, and $\mathbf{m}$ is defined in the network weights. We don't need to know all the coefficients at once, since the polynomial can be computed serially, one monomial after another, and one multiplication after another. To ensure accuracy ϵ, each multiplication requires depth $O(\log(1/\epsilon))$ and width 4. The total polynomial can then be accumulated using a subnetwork of depth $O(\log(1/\epsilon))$ and width 5.

4. Computation of the values $\phi\big(M(\mathbf{x}-\frac{\mathbf{n}_{\mathbf{q}}(x)}{N})-\mathbf{m}\big)$ can be performed in 2 channels using $O(\epsilon^{-d/(2r)})$ layers in total.

5. Once the factors are computed, each product $\phi\big(M(\mathbf{x}-\frac{\mathbf{n}_{\mathbf{q}}(x)}{N})-\mathbf{m}\big)P_{\mathbf{m}/M+\mathbf{n}_{\mathbf{q}}(\mathbf{x})/N}(\mathbf{x})$ can be computed with accuracy $O(\epsilon)$ in a subnetwork of width 4 and with $O(\log(1/\epsilon))$ layers, which gives $O(\epsilon^{-d/(2r)}\log(1/\epsilon))$ layers in total.

Summarizing, we see that the computation of $\widetilde{f}_{\mathbf{q}}(\mathbf{x})$ can be implemented with accuracy $O(\epsilon)$ in a subnetwork occupying $d+8$ channels and spanning $O(\epsilon^{-d/(2r)}\log(1/\epsilon))$ layers, as claimed.

C Theorem 5.1: proof

We generally follow the proof of Theorem 3.3 given in Sections 3 and A, and adapt it to the new setting. We start by reducing the approximation by σ-networks to deep polynomial approximations. We show that the ReLU activation function can be efficiently approximated by iterated polynomials, which allows us to reproduce some parts of the proof of Theorem 3.3 simply by approximating the ReLU. However, other parts, in particular the decoder subnetwork and the selection of the encoding weight, will require more significant changes.

Step 1: Reduction to polynomial approximation.

Lemma C.1. *Suppose that the activation function σ has a point x_0 where the second derivative $\frac{d^2\sigma}{dx^2}(x_0)$ exists and is nonzero. Then, for any multivariate polynomial u, there exists a network architecture such that the polynomial u can be approximated with any accuracy on any bounded set by a σ-network with this architecture by suitably assigning the weights.*

Proof. For $u(x) = x^2$, the desired σ-network is

$$\widetilde{u}_\delta(x) = (\tfrac{d^2\sigma}{dx^2}(x_0))^{-1}\tfrac{1}{\delta^2}\big(\sigma(x_0+x\delta)+\sigma(x_0-x\delta)-2\sigma(x_0)\big)$$

with a small δ. For any other polynomial, the network can be constructed by using $\widetilde{u}_\delta$, the polarization identity $xy = \frac{1}{2}((x+y)^2-(x-y)^2)$, and linear operations. □

In view of this lemma, in the sequel we will treat σ-networks as if capable of exactly implementing any polynomial using some finite architecture. Also, we note that under our assumption on the activation functions and in contrast to ReLU-networks, multiplications can be implemented with any accuracy by fixed-size subnetworks.

Step 2: Fast polynomial approximation of thresholds and ReLU. Consider the polynomial $u(x) = \frac{1}{2}x(3-x^2)$, which in particular has the following properties:

1. $u(0) = 0$ and $u(\pm 1) = \pm 1$;
2. u is monotone increasing on $[-1, 1]$;
3. $\frac{du}{dx}(\pm 1) = 0$.

Let u_n be the n'th iterate of u:

$$u_n = \underbrace{u\circ\ldots\circ u}_{n}. \tag{15}$$

Lemma C.2.

1. $|u_n(x) - \operatorname{sgn}(x)| \le |\operatorname{sgn}(x) - x|^{2^{n/2}}$ *for any* $x \in [-1, 1]$.
2. $|xu_n(x) - |x|| \le 2^{-n/2}$ *for any* $x \in [-1, 1]$.

Proof. 1. Make the change of variables $x = v(y) = 1 - y$ and let $\widetilde{u} = v^{-1}\circ u\circ v$. Then $\widetilde{u}(y) = \frac{1}{2}y^2(3-y)$. It is easy to check that $\widetilde{u}(y) \le y^{\sqrt{2}}$ for any $y \in [0, 1]$. Since both $\widetilde{u}(y)$ and $y^{\sqrt{2}}$ are monotone increasing on $[0, 1]$, there is a similar inequality for their n'th iterates: $\widetilde{u}_n(y) \le y^{2^{n/2}}$. This gives the desired bound for $x \in [0, 1]$. The bound for $x \in [-1, 0]$ follows by symmetry.

2. By Statement 1, $|xu_n(x) - |x|| \le |x||\operatorname{sgn}(x) - x|^{2^{n/2}} \le 2^{-n/2}$ for $x \in [-1, 1]$. □

The lemma implies, in particular, that a size-$O(n)$ σ-network can provide an approximation of accuracy $2^{-n/2}$ for the functions $|x|$ and x_+ on the segment $[-1, 1]$. The ReLU x_+ is approximated by $\frac{1}{2}(xu_n(x) + x)$.

Step 3: Reduction to $\widetilde{f}_{\mathbf{q}}$. In the original proof of faster rates for ReLU given in Section A, the first step was to represent the approximation $\widetilde{f}$ by a finite expansion (5) over 3^d subgrids indexed by $\mathbf{q} \in \{0, 1, 2\}^d$:

$$\widetilde{f}(\mathbf{x}) = \sum_{\mathbf{q}\in\{0,1,2\}^d} \widetilde{w}_{\mathbf{q}}(\mathbf{x})\widetilde{f}_{\mathbf{q}}(\mathbf{x}).$$

In the original proof, the "filtering functions" $\widetilde{w}_{\mathbf{q}}$ were linear combinations of $O(N^d)$ shifted and rescaled piecewise linear "spike" functions ϕ (see Prop. A.1). The function ϕ can be constructed using several linear and ReLU operations (see [10, Section 4.2]).

We observe now that, using Lemma C.2, we can very efficiently approximate the spike functions ϕ and then the full filtering functions $\widetilde{w}_{\mathbf{q}}$ by polynomials, simply by approximating each ReLU by the polynomial $\frac{1}{2}(xu_n(x) + x)$. Indeed, such an approximation of ϕ has accuracy $O(2^{-n/2})$ for a size-$O(n)$ σ-network (propagation of the error in the computation can be controlled in the standard way, using the Lipshitz continuity of ReLU). We need to remember, however, that the approximation in Lemma C.2 is valid only on the segment $[-1, 1]$, while the shifted and rescaled spike $\phi(N\mathbf{x} - \mathbf{n})$ requires ReLUs to act on a domain of size $O(N)$ if $\mathbf{x} \in [0, 1]^d$. The domain adaptation can be achieved simply by rescaling the ReLU using the identity $(Nx)_+ = Nx_+$. As a result, by adding approximations for all the spikes, we can approximate $\widetilde{w}_{\mathbf{q}}$ by a σ-network of size $O(nN^d)$ and depth $O(n)$ with uniform accuracy $O(2^{-n/2}N^{d+1})$. Let $N = W^{(1-\delta)/d}$ with some small $\delta > 0$, and $n = c\log_2 W$ with some $c > 2\frac{(1-\delta)(d+1)+r}{d}$. Then the accuracy of the $\widetilde{w}_{\mathbf{q}}$ network is within the desired bound $\epsilon = O(W^{-r/d})$, while the size of the $\widetilde{w}_{\mathbf{q}}$ network is $O(W^{1-\delta}\log W)$, also within the desired bound W.

This shows that our task is essentially reduced to implementing the functions $\widetilde{f}_{\mathbf{q}}$. We examine now the ReLU implementation of $\widetilde{f}_{\mathbf{q}}$ summarized into 5 steps in the end of Section A.3. We observe that steps 3-5 (computation of the weighted sum of Taylor approximations in a N-patch) can be easily implemented with the activation function σ instead of ReLU, by invoking again Lemma C.2 where necessary. In contrast, steps 1-2 require more significant modifications since they directly involve encoding weights that need to be handled with a high precision ($\sim W^{pd/r-1}$ bits). We first describe the suitable modification of step 2 (bit extraction), and then of step 1 (finding $\mathbf{n}_{\mathbf{q}}(\mathbf{x})$ and the respective encoding weight).

Step 4: Bit extraction. The standard bit extraction procedure (see [11] and Fig. 1) decodes a binary sequence from the encoding weight using threshold activation functions ($\lfloor\cdot\rfloor$) or their approximations by ReLU. In our present setting of polynomial approximation, we use instead a polynomial dinamical system. Specifically, consider the polynomial $v(x) = 2 - 3x^2$. Consider the disjoint intervals $I_0 = [\frac{1}{2}, 1], I_1 = [-1, -\frac{1}{2}]$ and observe that they are contained in the interval $[-1, 1]$ which is in turn contained in either of the images $v(I_0), v(I_1)$. Consider a sequence $w_1, \ldots, w_n$ defined by $w_k = v(w_{k-1})$ with some initial value w_1.

Lemma C.3. *For any binary sequence $b_1, \ldots, b_n \in \{0, 1\}$, there exists an interval $I \subset [-1, 1]$ of length at least 6^{-n} such that for any initial value $w_1 \in I$ we have $w_k \in I_{b_k}$ for all $k = 1, \ldots, n$.*

Proof. The interval I can be constructed by sequentially forming pre-images, $I^{(k-1)} = v^{-1}(I^{(k)}) \cap I_{b_{k-1}}$, where $k = n, n-1, \ldots, 2$, and $I^{(n)} = I_{b_n}$. Then $I = I^{(1)}$; the lower bound on the length of I follows since $|\frac{dv}{dx}| \le 6$ on $[-1, 1]$. □

The lemma shows that we can decode a length-n binary sequence by a σ-network of size $O(n)$ starting from an encoding weight defined with precision 6^{-n}. In contrast to the original ($\lfloor\cdot\rfloor$-based) bit extraction, the values decoded in the present polynomial procedure contain some uncertainty: we only know that w_k belong to one of the intervals I_0 or I_1. However, this uncertainty is not important: first, we can reduce it to an arbitrary magnitude by small-size subnetworks implementing a polynomial u_n from Lemma C.2; second, by Proposition A.4 and Eq.(8), some level of uncertainty in the Taylor coefficients $\widehat{a}_{\mathbf{m},\mathbf{k}}$ is tolerable.

Step 5: Computation of the encoding weight corresponding to given input x. In the proof for ReLU networks, the position $\mathbf{n}_{\mathbf{q}}(\mathbf{x})$ of the N-knot containing the given point $\mathbf{x}$, and the respective encoding weight, were determined exactly thanks to the ability of ReLU networks to exactly represent functions piecewise linear on the standard triangulation (see Proposition A.2). This is no longer possible with general activation functions σ or polynomials; any σ-network trying to determine the encoding weight will inevitably do it with some error. However, though the precision requirement for encoding weights is high, we can use part 1 of Lemma C.2 to bring this error to an acceptable level without substantially increasing the network size.

Indeed, consider a particular N-knot $\frac{\mathbf{n}}{N} \in \mathbf{N}_\mathbf{q}$ and first construct a map $z_\mathbf{n}(\mathbf{x})$ such that $z_\mathbf{n}(\mathbf{x}) \in [\frac{1}{2}, 1]$ for $\mathbf{x}$ belonging to the corresponding N-patch, while $z_\mathbf{n}(\mathbf{x}) \in [-1, -\frac{1}{2}]$ for $\mathbf{x}$ belonging to the other N-patches. Arguing as in Step 3, such a map can be implemented by a σ-network of size $O(\log W)$, by approximating the respective ReLU map.

Next, let $z_{\mathbf{n},n} = u_n \circ z_\mathbf{n}$, where u_n is given in Eq.(15). Using Lemma C.2 with $n = O(\log D)$, we can ensure that $|z_{\mathbf{n},n}(\mathbf{x}) - 1| < 7^{-D}$ on the $\mathbf{n}$'th patch while $|z_{\mathbf{n},n}(\mathbf{x}) + 1| < 7^{-D}$ on the other patches. Here, D corresponds to the number of iterations in Lemma C.3 and is proportional to the depth of the decoding subnetwork, i.e. $D \sim (M/N)^d \sim W^{pd/r-1}$ so that $\log D = O(\log W)$.

We can now combine all the maps $z_{\mathbf{n},n}$ into the map $Z(\mathbf{x}) = \frac{1}{2}\sum_{\mathbf{n}:\mathbf{n}/N \in \mathbf{N}_\mathbf{q}}(z_{\mathbf{n},n}(\mathbf{x}) + 1)w_\mathbf{n}$, where $w_\mathbf{n}$ is the desired encoding weight in the $\mathbf{n}$'th patch. By construction, for any $\mathbf{x}$ in the $\mathbf{n}$'th patch we have $|Z(\mathbf{x}) - w_\mathbf{n}| = O(N^d 7^{-D})$, which satisfies the accuracy requirement 6^{-D} of Lemma C.3. On the other hand, the size of the σ-network implementing $Z(\mathbf{x})$ is $O(N^d \log W)$. Choosing $N \sim W^{(1-\delta)/d}$ with arbitrarily small $\delta > 0$, this size fits the available budget W.

D Expressiveness of networks with Lipschitz activation functions and slowly growing weights

In this section we clarify why, as mentioned in Section 5, under mild assumptions on the growth of network weights, networks with any bounded Lipschitz activation function (in particular, the standard sigmoid $\sigma(x) = 1/(1 + e^{-x})$) can only achieve the approximation rates $p \leq \frac{2r}{d}$. This follows from existing upper bounds on the covering numbers for such networks, in particular [4, Theorem 14.5].

Specifically, consider a neural network with the following properties. Suppose that the network neurons have (possibly different) Lipschitz activation functions σ such that $|\sigma(x)| \leq b$ and $|\sigma(x) - \sigma(y)| \leq a|x - y|$ for all $x, y \in \mathbb{R}$. Suppose that there is a constant $V > 1/a$ such that for any weight vector $\mathbf{w}$ associated with a particular neuron, its l^1-norm $\|\mathbf{w}\|_1$ is bounded by V. Assume that the network has $L \geq 2$ layers, with connections only between adjacent layers, and has W weights. Assume finally that the neurons in the first layer have non-decreasing activation functions. Let F denote the family of functions on $[0, 1]^d$ implementable by such a network.

For any finite subset $S \subset [0, 1]^d$ consider the restriction $F|_S$ as a subset of $\mathbb{R}^{|S|}$ equipped with the uniform norm $\| \cdot \|_\infty$. We define the *covering number* $N_\infty(\epsilon, F, S)$ as the smallest number of ϵ-balls in $\mathbb{R}^{|S|}$ covering the set $F|_S$. Then, for any integer $m > 0$, we define the covering number $N_\infty(\epsilon, F, m) = \max_{S \subset [0,1]^d, |S|=m} N_\infty(\epsilon, F, m)$. We then have the following bound.

Theorem D.1 (Theorem 14.5 of [4]). $N_\infty(\epsilon, F, m) \leq \big(\frac{4embW(aV)^L}{\epsilon(aV-1)}\big)^W$.

To obtain the desired bound on approximaton rates for Hölder balls $F_{r,d}$, we can now lower-bound $N_\infty(\epsilon, F, m)$ using the ϵ-capacity of Hölder balls. Specifically, observe that the Hölder ball $F_{r,d}$ contains a set Φ_ϵ of at least $M_\epsilon = 2^{c_{r,d}\epsilon^{-d/r}}$ functions separated by $\| \cdot \|_\infty$-distance 4ϵ (with some constant $c_{r,d} > 0$). These functions can be constructed by a standard argument in which we choose in $[0, 1]^d$ a grid S_ϵ of size $c_{r,d}\epsilon^{-d/r}$ (with a spacing $\sim \epsilon^{1/r}$), and then place a properly rescaled spike function with the sign $+$ or $-$ at each point of the grid. The functions of Φ_ϵ are mutually 4ϵ-separated when restricted to the grid S_ϵ. If our family F of network-implementable functions can ϵ-approximate any function from the balls $F_{r,d}$, then any ϵ-net for $F|_{S_\epsilon}$ is a 2ϵ-net for $\Phi_\epsilon|_{S_\epsilon}$, and thus must contain at least M_ϵ elements. Hence, $M_\epsilon \leq N_\infty(\epsilon, F, S_\epsilon) \leq N_\infty(\epsilon, F, c_{r,d}\epsilon^{-d/r})$, i.e.

$$c_{r,d}\epsilon^{-d/r} \leq W \log_2 \big(\tfrac{4ec_{r,d}\epsilon^{-d/r-1}bW(aV)^L}{aV-1}\big). \tag{16}$$

Assuming that $1/\epsilon, W, L, V$ grow while the other parameters are held constant, this bound implies that

$$\epsilon \geq c_{r,d,a,b}(WL)^{-r/d}\ln^{-r/d} V$$

with some $c_{r,d,a,b} > 0$.

Now suppose that V is a function of W, i.e. the magnitude of the weights is allowed to depend on the network size. Suppose that the network achieves the approximation rate p, i.e.

$$\epsilon \leq C_{r,d,a,b}W^{-p}. \tag{17}$$

Since $L \leq W$, comparing Eq.(16) with Eq.(17), we then find that

$$\ln V \geq c'_{r,d,a,b}W^{pd/r-2}. \tag{18}$$

Thus, the rates $p > \frac{2r}{d}$ require V to very rapidly grow with W. This observation agrees with the main result of Section 5 – Theorem 6.1 – describing approximation with arbitrary rates p by networks with a periodic activation function. In the proof of this theorem, the network weights are defined with the help of rapidly growing constants a_k given in Eq.(19). In particular, we have $\log a_K \sim 2^K$ with $K \sim W^{1/2}$, which agrees with the lower bound (18).

E Theorem 6.1: sketch of proof

We can assume without loss of generality that the period $T = 2$ and $\max_{x\in\mathbb{R}}\sigma(x) = -\min_{x\in\mathbb{R}}\sigma(x) = 1$ (these values can always be effectively adjusted in each neuron by rescaling the input and output weights). We divide the proof into three steps.

Step 1: reduction to patch-encoders and patch-classifiers. Recall the concepts of coarser partition on the scale $\frac{1}{N}$ and the finer partition on the scale $\frac{1}{M}$ used in the proofs of Theorem 3.3 and 4.1. In those theorems, both N and M were $\sim W^a$ with some constant powers a. In contrast, we choose now $N = 1$, and we'll set M to grow much faster (roughly exponentially) with W: this will be possible thanks to the much more efficient decoding available with the $\sin$ activation.

Specifically, note first that we can implement an almost perfect approximation of the parity function $\theta : x \mapsto (-1)^{\lfloor x\rfloor}$ using a constant size networks, by computing $a\sigma(x)$ with a large a and then thresholding the result at 1 and -1 using ReLU operations (the approximation only fails in small neighborhoods of the integer points). If the cube $[0,1]^d$ is partitioned into cubic M-patches, we can apply rescaled versions of θ coordinate-wise to create a binary dictionary of these patches. Specifically, we can construct a network of size $\sim d\log_2 M$ that maps a given $\mathbf{x} \in [0,1]^d$ to a size-K binary sequence encoding the place of the patch $\Delta_M \ni \mathbf{x}$ in the cube $[0,1]^d$, with $K \sim d\log_2 M$. We call this network the *patch-encoder.*

Given a function $f \in F_{r,d}$, we approximate it by a function $\widetilde{f}$ which is constant in each M-patch. Suppose for simplicity and without loss of generality that the smoothness $r \leq 1$, then this approximation has accuracy $\epsilon \sim M^{-r}$. Let $\widetilde{f}_{\Delta_M}$ be the value that the approximation returns on the patch Δ_M. It is sufficient to define $\widetilde{f}_{\Delta_M}$ with precision $\sim M^{-r}$. Consider the binary expansion of f_{Δ_M} that provides this precision: $\widetilde{f}_{\Delta_M} = -1 + \sum_{k=0}^R \widetilde{f}_{\Delta_M,k}2^{-k}$, where $R \sim r\log_2 M$ and $\widetilde{f}_{\Delta_M,k} \in \{0,1\}$. Suppose that for each k we can construct a network that maps each patch Δ_M to the corresponding bit $\widetilde{f}_{\Delta_M,k}$. Summing these *patch-classifiers* with coefficients 2^{-k}, we then reconstruct the full approximation $\widetilde{f}$.

We have thus reduced the task to efficiently implementing an arbitrary binary classifier on the M-partition of $[0,1]^d$. The patch-encoder constructed above efficiently encodes each M-patch by a binary K-bit sequence. We can then think of the classifier as an assignment $A : \{0,1\}^K \to \{0,1\}$ that must be implemented by our network. We show below in Step 2 that this can be done by a size-$O(K)$ network, with the assignment encoded in a single weight w_A. The full number of network weights (including the patch-encoder and the patch-classifiers on all R scales) can then be bounded by $W = O(KR)$, i.e. $W = O(rd\log_2^2 M)$. The relations $\epsilon \sim M^{-r}$ and $W \sim rd\log_2^2 M$ then yield $\epsilon \sim 2^{-c'W^{1/2}}$ (with $c' \sim \sqrt{r/d}$), as claimed in Eq.(4).

Note, however, that the proof strategy that we have described requires the network to have R f-dependent encoding weights (one per each patch-classifier), while Statement 2 of the theorem claims a unique f-dependent weight. In Step 3, we will resolve this issue by showing that these R weights can be decoded from a single weight with a subnetwork of size $O(R)$.

To make these arguments fully rigorous, we need to handle the issue of our approximation to the parity function θ becoming invalid near the boundaries of the patches. This is done in Section F using partitions of unity; the resulting complications do not affect the asymptotic.

Step 2: implementation of a patch-classifier. We explain now how an arbitrary assignment $A : \{0,1\}^K \to \{0,1\}$ can be implemented by a network of size $O(K)$ with a single encoding weight w_A. Let us define two sequences, a_k and l_k:

$$l_1 = \tfrac{1}{2}, \quad a_1 = 2, \quad l_k = \min(\tfrac{l_{k-1}}{2}, \tfrac{l_{k-1}}{a_k c_\sigma}), \quad a_k = \tfrac{4}{l_{k-1}}, \tag{19}$$

where c_σ is the Lipschitz constant of σ. Consider iterations $g_1 \circ g_2 \circ \ldots \circ g_K(w_*)$, in which each g_k can be either the identity function $g_k(w) = w$, or $g_k(w) = \sigma(a_k w)$, with some initial value w_*. For each $\mathbf{z} \in \{0,1\}^K$, let us define $H_{K,w_*}(\mathbf{z})$ as the sgn of the value obtained by substituting the respective functions:

$$H_{K,w_*}(\mathbf{z}) = \operatorname{sgn} \circ \begin{cases} \mathrm{Id}, & z_1 = 0, \\ \sigma(a_1 \cdot), & z_1 = 1 \end{cases} \circ \begin{cases} \mathrm{Id}, & z_2 = 0, \\ \sigma(a_2 \cdot), & z_2 = 1 \end{cases} \circ \ldots \circ \begin{cases} \mathrm{Id}, & z_K = 0, \\ \sigma(a_K \cdot), & z_K = 1 \end{cases} (w_*)$$

Lemma E.1. *For any assignment $A : \{0,1\}^K \to \{0,1\}$ there exists $w_A \in \mathbb{R}$ such that $H_{K,w_A}(\mathbf{z}) = A(\mathbf{z})$ for all $\mathbf{z} \in \{0,1\}^K$.*

Proof. Proof by induction on K, but of a slightly sharper statement: the desired w_A not only exist, but fill (at least) an interval $I_K \subset [-1,1]$ of length l_K.

The base $K = 1$ follows immediately from the 2-periodicity of σ and the hypothesis that $\sigma(x) > 0$ for $x \in [0,1]$ while $\sigma(x) < 1$ for $x \in [1,2]$. Suppose we have proved the statement for $K-1$. Given an assignment $A : \{0,1\}^K \to \{0,1\}$, consider it as a pair of assignments $A_0 : \{0,1\}^{K-1} \to \{0,1\}, A_1 : \{0,1\}^{K-1} \to \{0,1\}$. By the induction hypothesis, we can find two intervals $I_{K-1}^{(0)}$ and $I_{K-1}^{(1)}$ of length l_{K-1} such that $H_{K-1,w_0}(\mathbf{z}) = A_0(\mathbf{z})$ and $H_{K-1,w_1}(\mathbf{z}) = A_1(\mathbf{z})$ for all $w_0 \in I_{K-1}^{(0)}, w_1 \in I_{K-1}^{(1)}$ and $\mathbf{z} \in \{0,1\}^{K-1}$. Consider the set

$$I = \{w \in \mathbb{R} : w \in I_{K-1}^{(0)} \text{ and } \sigma(a_K w) \in I_{K-1}^{(1)}\}. \tag{20}$$

Then for any $w \in I$, we have the desired property $H_{K,w}(\mathbf{z}) = A(\mathbf{z}), \forall \mathbf{z} \in \{0,1\}^K$. We need to show now that I contains an interval of length l_K. Observe that, by the relation $a_K = \frac{4}{l_{K-1}}$ from Eq.(19), the length l_{K-1} of the interval $I_{K-1}^{(0)}$ is twice as large as the period $\frac{2}{a_K}$ of the function $\sigma(a_K \cdot)$. Using the assumption that $\max \sigma(x) = -\min \sigma(x) = 1$, we see that the function $\sigma(a_K \cdot)$ attains both values 1 and -1 on its period. It follows then by continuity of σ that there exists a point w' at a distance not more than $\frac{l_{K-1}}{4}$ from the center of $I_{K-1}^{(0)}$ such that $\sigma(a_K w')$ attains any given value from the interval $[-1,1]$. Let this value be the center w'' of the interval $I_{K-1}^{(1)}$. Since the function $\sigma(a_K \cdot)$ is Lipschitz with constant $a_K c_\sigma$, we have $|\sigma(a_K w) - \sigma(a_K w')| < \frac{l_{K-1}}{2}$ for any w such that $|w - w'| < \frac{l_{K-1}}{2 a_K c_\sigma}$. Then it follows from the definition $l_K = \min(\frac{l_{K-1}}{2}, \frac{l_{K-1}}{a_K c_\sigma})$ that the length-l_K interval centered at w' is contained in I given by Eq.(20). □

This lemma shows that the network can implement any classifier A if the network can somehow branch into applying either Id or $\sigma(a_k \cdot)$ depending on the signal bit $b \in \{0,1\}$ that is output by the patch-encoder subnetwork. This branching can be easily implemented by forming the linear combination $(1-b)x + b\sigma(a_k x)$, and also noting that a product of any $x \in \{0,1\}$ and $y \in [-1,1]$ admits the ReLU implementation $xy = \max(0, 2x + y - 1) - x$.

Remark. The construction in Lemma E.1 can be interpreted as a dichotomy-based lookup if we think of the assignment A as a binary sequence of size $S = 2^K$. In each of the network steps we divide the sequence in half, ultimately locating the desired bit in $K \sim \log_2 S$ steps. We can compare this with the less efficient bit extraction procedure of [11] (for which it is however sufficient to only have the ReLU activation in the network). In this latter procedure, the bits are extracted from the encoding weight one-by-one, and so the lookup requires $\sim S$ steps.

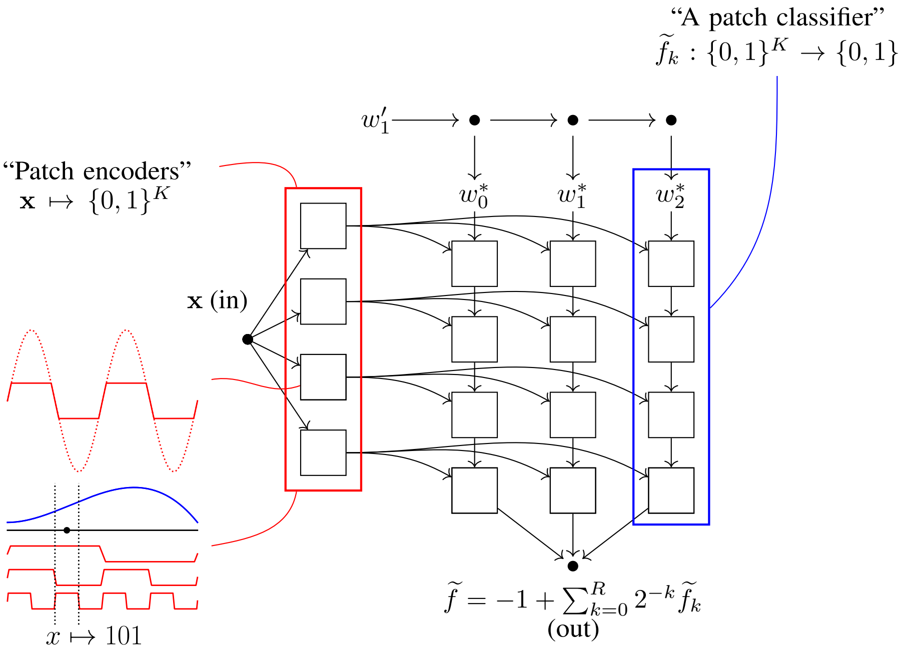

"A patch classifier"
$\widetilde{f}_k : \{0,1\}^K \to \{0,1\}$
w'_1
w^*_0
w^*_1
w^*_2
"Patch encoders"
$\mathbf{x} \mapsto \{0,1\}^K$
$\mathbf{x}$ (in)
$\widetilde{f} = -1 + \sum_{k=0}^{R} 2^{-k} \widetilde{f}_k$
(out)
$x \mapsto 101$

Figure 7: The network layout overview for the "deep Fourier" approximation (see Section E).

Step 3: ensuring a unique f-dependent weight. Steps 1 and 2 have shown that the desired network can be constructed using at most R f-dependent weights, say $w^*_1, \ldots, w^*_R$. We observe now that these values can be approximated arbitrarily well by a serial application of the rescaled activation σ:

Lemma E.2. *Let $w^*_1, \ldots, w^*_R \in [-1, 1]$. Fix $a > 0$ and consider the sequence $w'_1, \ldots, w'_R$ defined by $w'_k = \sigma(aw'_{k-1})$ with some initial w'_1. Then we can find an initial $w'_1 \in [-1, 1]$ such that $|w^*_k - w'_k| < \frac{2}{a}$ for all $k = 1, \ldots, R$.*

Proof. We use induction on R. The base $R = 1$ is trival. Suppose we have already proved the statement for $R - 1$. Let $\widetilde{w}$ be the corresponding initial value for the sequence $\widetilde{w}, \sigma(a\widetilde{w}), \sigma(a\sigma(a\widetilde{w})), \ldots$ approximating the sequence $w^*_2, \ldots, w^*_R$. The function $\sigma(a\cdot)$ has period $\frac{2}{a}$, and attains all values from $[-1, 1]$ on any interval of this length. It follows that we can find w'_1 such that $\sigma(aw'_1) = \widetilde{w}$ and $|w^*_1 - w'_1| < \frac{2}{a}$. This gives the desired w'_1. □

Thus, by taking some sufficiently large (W-dependent) a, we can generate all the R encoding weights w^*_k with sufficient accuracy from a single weight by using an f-independent network of complexity $O(R)$, which is within the desired bound (4).

In Figure 7 we show the overall network layout.

Information in the encoding weight. Let us make a rough estimate of the amount of information contained in the constructed encoding weight. First, we can estimate it as R (the number of patch classifiers) times the information in the weight w_A corresponding to a single patch classifier (see Lemma E.1). We have $R \sim r \log_2 M \sim \log_2 \epsilon$. The information in w_A can be roughly estimated as $-\log_2 l_K$, where l_K is the length of the interval I_K appearing in the proof of Lemma E.1. From relations (19), for small l_{k-1}, by combining $l_k = \frac{l_{k-1}}{a_k c_\sigma}$ and $a_k = \frac{4}{l_{k-1}}$ we get $l_k = \frac{l^2_{k-1}}{4c_\sigma}$, which leads to $\log_2 l_K \sim 2^K$. Since $K \sim d \log_2 M \sim \frac{d}{r} \log_2 \epsilon$, this gives $-\log_2 l_K \sim \epsilon^{-d/r}$. Summarizing, the total information can be roughly estimated as $\epsilon^{-d/r} \log(1/\epsilon)$.

F Theorem 6.1: proof details

Examining the sketch of proof given in Section E, we see that the only significant gap in the given argument is the treatment of boundaries of the patches. Namely, recall that we use approximations to the parity function $\theta(x) = (-1)^{\lfloor x \rfloor}$. The approximations can be defined by a finite expression in terms of linear, ReLU and sin operations:

$$\widetilde{\theta}_a(x) = \min(1, \max(-1, a\sigma(x))).$$

By taking a large, we can make $\widetilde{\theta}_a$ to equal θ outside some small neighborhood of $\mathbb{Z}$. Now, recall that we choose patches Δ_M as cubes $[\frac{m_1}{M}, \frac{m_1+1}{M}] \times [\frac{m_2}{M}, \frac{m_2+1}{M}] \times \ldots [\frac{m_d}{M}, \frac{m_d+1}{M}]$. Assume without loss of generality that $M = 2^U$ with some integer U. The patch-encoding functions $g_{u,k} : \mathbf{x} \mapsto \widetilde{\theta}_a(2^u x_k)$ (with $u = 1, 2, \ldots, U$ and $k = 1, 2, \ldots, d$) map the cubes Δ_M to the values ± 1 everywhere except near the boundaries of these cubes. If we could slightly "shrink" the cubes Δ_M so that they were disjoint, we could adjust a in $\widetilde{\theta}_a$ so that the functions $g_{u,k}$ were perfectly equal to ± 1 on the whole cubes. The remaining construction of patch-classifying networks in Section 5 then becomes fully functional and yields the desired asymptotic relation (4).

Thus, we need to show how to reduce the problem to the case of disjoint patches. This can be done by using suitable filtering functions, similarly to the proofs of Theorems 3.3 and 4.1. Fix some $a_0 > 1$ and consider the functions $\Psi_0, \Psi_1 : \mathbb{R} \to [0, 1]$ defined by

$$\Psi_0(x) = \tfrac{1}{2}(1 + \widetilde{\theta}_{a_0}(2Mx)), \quad \Psi_1 = 1 - \Psi_0.$$

The functions Ψ_0 and Ψ_1 form a a two-element partition of unity. Furthermore, since $a_0 > 1$, there is $\delta > 0$ such that

$$\Psi_0(x) = 0 \text{ for } x \in (\tfrac{3}{4M} - \delta, \tfrac{3}{4M} + \delta) + \mathbb{Z}/M, \tag{21}$$

$$\Psi_1(x) = 0 \text{ for } x \in (\tfrac{1}{4M} - \delta, \tfrac{1}{4M} + \delta) + \mathbb{Z}/M. \tag{22}$$

Taking the product of the partitions of unity over the d coordinates, we can write for $\widetilde{f} : [0, 1]^d \to \mathbb{R}$:

$$\widetilde{f} = \sum_{\mathbf{q} \in \{0,1\}^d} (\prod_{s=1}^{d} \Psi_{q_s}) \widetilde{f}.$$

Thanks to Eqs.(21),(22), for each $\mathbf{q} \in \{0, 1\}^d$, the filtering function $\prod_{s=1}^{d} \Psi_{q_s}$ vanishes in $[0, 1]^d$ outside an $\frac{1}{M}$-grid of disjoint cubic patches, exactly as desired. We can then look for the approximation $\widetilde{f}$ in the form

$$\widetilde{f} = \sum_{\mathbf{q} \in \{0,1\}^d} (\prod_{s=1}^{d} \Psi_{q_s}) \widetilde{f}_{\mathbf{q}},$$

where $\widetilde{f}_{\mathbf{q}}$ has the required values only on the patches $[0, 1]^d \backslash \operatorname{supp}(\prod_{s=1}^{d} \Psi_{q_s})$ and can be constructed as described in Section 5.

Having implemented these approximations $\widetilde{f}_{\mathbf{q}}$, the final approximation is obtained by implementing approximate products with the filters Ψ_{q_s} and performing summation over $\mathbf{q} \in \{0, 1\}^d$. As shown in [6, Proposition 3], multiplication with accuracy ϵ requires a ReLU subnetwork with $O(\log(1/\epsilon))$ connections. This is asymptotically negligible compared to our bound for the total complexity of the patch-classifiers (which is $O(\log^2(1/\epsilon))$).