SUPN: Shallow Universal Polynomial Networks

Anonymous authors
Paper under double-blind review

Abstract

Deep neural networks (DNNs) and Kolmogorov–Arnold networks (KANs) are popular methods for function approximation due to their flexibility and expressivity. However, they typically require a large number of trainable parameters to produce a suitable approximation. Beyond making the resulting network less transparent, overparameterization creates a large optimization space, likely producing local minima in training that have quite different generalization errors. In this case, network initialization can have an outsize impact on the model's out-of-sample accuracy. For these reasons, we propose shallow universal polynomial networks (SUPNs). These networks replace all but the last hidden layer with a single layer of polynomials with learnable coefficients, leveraging the strengths of DNNs and polynomials to achieve sufficient expressivity with far fewer parameters. We prove that SUPNs converge at the same rate as the best polynomial approximation of the same degree, and we derive explicit formulas for quasi-optimal SUPN parameters. We complement theory with an extensive suite of numerical experiments involving SUPNs, DNNs, KANs, and polynomial projection in one, two, and ten dimensions, consisting of over 13,000 trained models. On the target functions we numerically studied, for a given number of trainable parameters, the approximation error and variability are often lower for SUPNs than for DNNs and KANs by an order of magnitude. In our examples, SUPNs even outperform polynomial projection on non-smooth functions.

1 Introduction

Many scientific applications involve computationally expensive input–output maps, denoted f, resulting from physical systems, such as state-space evolution (Peherstorfer & Willcox, 2016) or the relationship between a model's parameters and certain quantities of interest (Morrow & Stoyanov, 2020). Accordingly, much work has been devoted toward constructing less costly approximations equipped with error estimates or convergence guarantees. One approach is focused on accelerating a numerical solver, such as a finite-element method, by simplifying a model's governing equations (Jakeman et al., 2025) or constructing a reduced-order model (Benner et al., 2017) that diminishes the number of unknowns. Though this approach is focused on the physical system and not directly on f, such improvements will nonetheless reduce the expense of constructing f. Another approach is to approximate f directly. In practice, this requires sampling its inputs and outputs, and an approximation constructed this way is called a *data-driven surrogate model*. The efficiency of this approach depends on the dimensionality and nonlinearity of f. Many methods in this category are concerned primarily with the high-dimensional case because a system may have a very large number of parameters mapped to some quantity of interest. In this paper, we present a highly expressive method for approximating nonlinear and low-regularity functions, which outperforms state-of-the-art alternatives. For transparency and to avoid the known issue of weak baselines in machine learning research (McGreivy & Hakim, 2024), we compare our method to deep neural networks (DNNs), Kolmogorov–Arnold networks (KANs), and polynomial projection on an extensive set of numerical experiments with documented hyperparameter choices.

Several different varieties of data-driven surrogate models exist, and we will purposefully save the technical formulation for a later section. Although an exhaustive list is beyond the scope of this paper, we offer a brief description. A data-driven surrogate of an input–output map can be broadly categorized as linear or

nonlinear, depending on how the parameters appear within the surrogate's ansatz. Linear methods express the resultant surrogate as a linear combination of chosen basis functions. This category includes polynomial chaos expansions (Ghanem & Spanos, 1991; Xiu & Karniadakis, 2002), sparse grids (Nobile et al., 2008; Barthelmann et al., 2000), operator inference (Peherstorfer & Willcox, 2016; Benner et al., 2015), spectral methods (Canuto et al., 2006; Gottlieb & Orszag, 1977), and radial basis functions (Buhmann, 2003). In contrast, the parameters of nonlinear methods are embedded within the surrogate's constituent functions, which are generally nonlinear. The parameters of a nonlinear surrogate are determined with an iterative optimization procedure. Functional tensor trains (Gorodetsky et al., 2019) and deep neural networks (DNNs) are examples of this category.

Although linear surrogates are well-established methods with a rich body of theory, DNNs have attracted significant contemporary interest due to their flexibility and expressivity, resulting in a proliferation of architectures (Cybenko, 1989; Hochreiter & Schmidhuber, 1997; Ronneberger et al., 2015; Liu et al., 2025; Vaswani et al., 2017) and bespoke adaptations tailored to specific problems, such as AlphaFold (Jumper et al., 2021). Nonetheless, the search for a sufficiently accurate DNN is often hampered by vanishing or exploding gradients, an unwieldy number of trainable parameters, and difficulty approximating high-frequency target functions (Zhang et al., 2025). Theoretical results establish that feedforward networks, also known as multi-layer perceptrons (MLPs), can approximate continuous functions arbitrarily well (Pinkus, 1999). An alternative to the MLP, KANs are based on the Kolmogorov–Arnold representation theorem (Arnold, 2009; Kolmogorov, 1957; Braun & Griebel, 2009). In contrast to MLPs, KANs have learnable parameters within their activation functions, which are parametrized as, e.g., splines (de Boor, 1978) or Chebyshev polynomials (Atkinson, 1989, p. 211). In fact, KANs with a linear spline basis are equivalent to MLPs with a ReLU activation function (Actor et al., 2025). Neural networks (NNs) have also been extended to learn discretizations of operators between function spaces. Kernel-based methods such as the Fourier neural operator (FNO) (Li et al., 2021) and kernel neural operator (KNO) (Lowery et al., 2024) seek to learn a sequence of kernel integral operators. Deep operator networks (DeepONets) (Lu et al., 2021) are an alternative approach consisting of two NNs, one learning a set of basis functions and the other learning a set of coefficients.

We avoid the previously discussed issues in deep learning of functions by adopting a fundamentally shallow approach. We introduce a single-layer tanh-activated network with an initial lift consisting of Chebyshev polynomials, which we call a *shallow universal polynomial network* (SUPN). This function-approximation method leverages the power of both polynomial approximation and NNs (via a nonlinear activation function) to produce an approximation that is both parsimonious and robust with respect to network parameter initialization. Indeed, the principal theoretical contributions of this paper (Section 3.2) go beyond standard universal approximation theorems and establish that SUPNs converge at the same rate as a related polynomial approximation. Our numerical experiments indicate that SUPNs are competitive against DNNs and KANs on smooth target functions, and they are competitive against polynomial projection on nonsmooth targets. Moreover, we observed that, without fail, SUPNs perform at least as well as—and often much better than—DNNs or KANs when the target function has tensor-product structure.

Figure 1 shows a visual comparison of SUPNs with other methods. SUPNs differ from MLPs through the Chebyshev lift, and they differ from KANs through the use of fixed activation functions and a single layer. In particular, previous discussions of Chebyshev KANs (Sidharth et al., 2024; Shukla et al., 2024) adopted a deep-learning approach that composed tanh-activated Chebyshev layers together. In addition, Xu et al. (2025) proposed a Chebyshev feature map where the learnable parameters are not basis coefficients but instead generalized polynomial "degrees"; those networks still use a deep MLP after the Chebyshev lift.

The term *polynomial networks* has historically referred to methods that utilize polynomial activation functions. Despite the similarity in naming conventions, our method has little in common with these prior models, such as polynomial neural networks (PNNs) (Das, 1995; Kak, 1993; Hopfield, 1982), deep PNNs (Kileel et al., 2019; Chrysos et al., 2021; Kubjas et al., 2024), and sum-product networks (Poon & Domingos, 2011). What unites these methods is that they are reformulations or extensions of the so-called Group Method of Data Handling (Ivakhnenko, 1971), which builds hierarchical polynomial regressions and is equivalent to an MLP with polynomial activation functions. However, it is well-established that MLPs with polynomial activation functions are not universal approximators (Leshno et al., 1993; Sonoda & Murata, 2017; Pinkus, 1999). In

contrast, our proposed method, based on linear combinations of tanh-activated polynomials, *is* a universal approximator.

To motivate the derivation of SUPNs, enable their analysis, and demonstrate their approximation capabilities, the rest of the paper is organized as follows. Section 2 provides a general framework for data-driven surrogate modeling and error estimation, as well as basic formulations of polynomial projection, MLPs, and KANs. Section 3 presents the mathematical formulation of SUPNs, establishes two basic universal approximation theorems, and proves SUPN convergence rates based on polynomial error bounds. Section 4 discusses the second-order optimizer utilized for training. Section 5 compares SUPNs, polynomial projection, DNNs, and KANs in one, two, and ten dimensions on standard test functions (e.g., Runge and Rastrigin) and specific examples from KAN papers. Existing literature currently lacks a systematic comparison of machine learning methods against polynomial approximations. Section 6 summarizes our findings and discusses possible extensions of this work. For convenience, Appendix A summarizes the notation used throughout this paper.

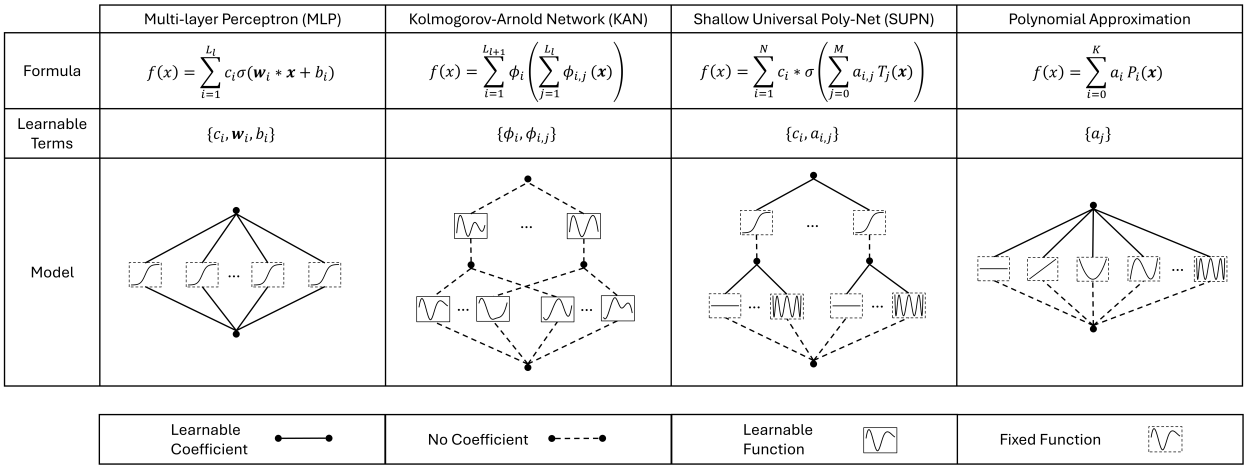

Figure 1: Illustration of function approximators considered in this work. Note that MLPs and KANs are shown in their single layer form for comparison, but are most often used in a deep (many-layer) configuration compared to SUPNs, which are always a single layer.

2 Surrogate models

Consider a model f mapping D bounded inputs to Q outputs. We can denote this model as $f : \mathbb{R}^D \supset \Omega \to \mathbb{R}^Q$ where $\Omega \subset \mathbb{R}^D$ is compact.[1] The first step of surrogate modeling is to define a space $\mathcal{F}$ of candidate functions, parameterized by an ansatz $\bar{g} = \bar{g}(\cdot\,; \boldsymbol{\theta})$ and $\boldsymbol{\theta} \in \mathbb{R}^P$ such that $\mathcal{F}_{\boldsymbol{\theta}} = \{\bar{g}(\cdot\,; \boldsymbol{\theta}) : \boldsymbol{\theta} \in \mathbb{R}^P\}$. At the highest level, the goal of surrogate modeling is to find a suitable approximation $\hat{f} \in \mathcal{F}_{\boldsymbol{\theta}}$. With $p \geq 1$ and μ a positive measure on Ω, the best approximation of f by $\mathcal{F}_{\boldsymbol{\theta}}$ in the μ-weighted $L^p(\Omega)$ norm is

$$\hat{f} \coloneqq \underset{g \in \mathcal{F}_{\boldsymbol{\theta}}}{\arg\min} \, \|f - g\|_{L^p_\mu} \,. \tag{1}$$

A minimum exists (rather than an infimum) because $\mathcal{F}_{\boldsymbol{\theta}}$ is finite-dimensional and the norm is a continuous function bounded below. An alternative form of (1), which will become useful later, is

$$\hat{f} = \bar{g}(\cdot\,; \hat{\boldsymbol{\theta}}), \qquad \hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^P}{\arg\min} \, \|f - \bar{g}(\cdot\,; \boldsymbol{\theta})\|_{L^p_\mu} \,. \tag{2}$$

The choice of norm and $\bar{g}$ in (2) are crucial to the accuracy of $\hat{f}$. Common choices of $\bar{g}$ are degree-n polynomials (denoted $\mathbb{P}_n$), sines and cosines with maximum frequency n, or neural networks of a given width

[1] Without loss of generality, we may take $\Omega = [-1, 1]^D$ since a compact interval $[a, b]$ can be mapped to $[-1, 1]$ with a straightforward affine transformation. We also often take $Q = 1$ for convenience, since the multi-output case constructs a scalar-valued result for each output dimension.

and depth. Well-established theory exists for these ansatzes, and we will review some of this theory later in this section.

In practice, surrogates are constructed instead from a finite amount of data because the explicit functional form of f is rarely available. In the supervised learning context, one collects $K \in \mathbb{N}$ input–output pairs $\{(\boldsymbol{x}_k, f(\boldsymbol{x}_k))\}_{k\in[K]}$, where $[K] \coloneqq \{1, \dots, K\}$. Data-driven surrogates approximate the continuous L^p norms with a corresponding discrete ℓ^p norm. The discrete minimization problem becomes

$$\begin{aligned} \tilde{f} &= \bar{g}(\cdot\,; \tilde{\boldsymbol{\theta}}), & \tilde{\boldsymbol{\theta}} &= \operatorname*{arg\,min}_{\boldsymbol{\theta}\in\mathbb{R}^P} \|\bar{\boldsymbol{g}}(\boldsymbol{\theta}) - \boldsymbol{f}\|_{\ell^p_{\boldsymbol{w}}(\mathbb{R}^K)}, \\ \bar{\boldsymbol{g}}(\boldsymbol{\theta}) &= (\bar{g}(\boldsymbol{x}_k; \boldsymbol{\theta}))_{k\in[K]}\,, & \boldsymbol{f} &= (f(\boldsymbol{x}_k))_{k\in[K]}\,, \end{aligned} \tag{3}$$

which is a direct adaptation of the best-approximation problem (2). The weights $\boldsymbol{w} \in \mathbb{R}^K$ are chosen to approximate the integral in the L^p norm using the discrete ℓ^p norm:

$$\|\boldsymbol{v}\|_{\ell^p_{\boldsymbol{w}}(\mathbb{R}^K)} = \begin{cases} \left(\sum_{k\in[K]} w_k |v_k|^p\right)^{1/p}, & \text{if } p < \infty \\ \max_{k\in[K]} |v_k|, & \text{if } p = \infty \end{cases}.$$

The final step is to minimize (3). Popular optimization methods include grid search and gradient-based optimizers. Grid searches discretize the feasible set for $\boldsymbol{\theta}$ and take the discrete minimum. However, grid searches often do not provide optimality guarantees, and P is typically quite large, so the number of samples required to cover the sample space (e.g., with low discrepancy) is practically prohibitive. Alternatively, when using gradient-based optimizers, the ℓ^p norm should be differentiable. As a result, practical implementation uses (3) with the ℓ^2 norm, yielding the familiar loss-minimization problem with mean-squared error (MSE):

$$\tilde{\boldsymbol{\theta}} = \operatorname*{arg\,min}_{\boldsymbol{\theta}\in\mathbb{R}^P} \mathcal{L}(\boldsymbol{\theta}), \qquad \mathcal{L}(\boldsymbol{\theta}) = \sum_{k\in[K]} w_k \left(\bar{g}(\boldsymbol{x}_k; \boldsymbol{\theta}) - f(\boldsymbol{x}_k)\right)^2 . \tag{4}$$

In the special case where $\bar{g}$ depends linearly on $\boldsymbol{\theta}$, the minimizer $\tilde{\boldsymbol{\theta}}$ is the solution to a linear system, which is significantly cheaper to solve than the nonlinear case.

2.1 Ansatzes

We now present three prominent choices of the ansatz $\bar{g}$. These ansatzes have associated theory that will be crucial in the theoretical analysis of Section 3, and they will be used for comparison in Section 5.

2.1.1 Polynomials

Consider how to approximate a function $f : \Omega \to \mathbb{R}$ with polynomials in some norm. For this paper, there are two important cases to consider: the L^∞ norm and the L^2 norm. The first case is addressed in the following theorems, which will be needed in the proofs of Section 3.

Theorem 2.1 (Stone–Weierstrass (Cotter, 1990; Stone, 1937; Atkinson, 1989))**.** *Let $f : \Omega \to \mathbb{R}$ be continuous and $\Omega \subset \mathbb{R}^D$ be compact. For any $\epsilon > 0$, there exists a polynomial $q : \Omega \to \mathbb{R}$ such that $\|f - q\|_{L^\infty(\Omega)} < \epsilon$.*

Theorem 2.2 (Jackson's inequality (Jackson, 1930))**.** *Let $f \in C^k([-1, 1], \mathbb{R})$ with $f^{(k)}$ α-Hölder continuous, i.e. there exists $\gamma > 0$ and $\alpha \in (0, 1]$ such that $|f^{(k)}(x) - f^{(k)}(y)| \le \gamma|x - y|^\alpha$. There exists a constant $\beta > 0$ such that, for every $n \ge 1$, there is a polynomial $q_n \in \mathbb{P}_n$ for which*

$$\|f - q_n\|_{L^\infty} \le \beta\, n^{-(k+\alpha)} .$$

Now consider a polynomial approximation of $f \in L^2(\Omega)$ with degree at most P in the unweighted L^2 norm. For simplicity, we address $D = 1$. This corresponds to (2) with the L^2 norm and

$$\bar{g}_P(x; \boldsymbol{\theta}) = \sum_{p\in[P-1]_0} \theta_p\, L_p(x), \tag{5}$$

where $[P-1]_0 \coloneqq \{0, \dots, P-1\}$ and L_p is the Legendre polynomial of degree p. The Legendre polynomials satisfy

$$\langle L_n, L_m \rangle_{L^2(\Omega)} = \delta_{nm} \frac{2}{2n+1} \,.$$

where

$$\langle f, g \rangle_{L^2(\Omega)} \coloneqq \int_\Omega f(x) g(x) \,\mathrm{d}x \,, \qquad \|f\|^2_{L^2(\Omega)} \coloneqq \langle f, f \rangle_{L^2(\Omega)} \,.$$

In this case, the coefficients of the best approximation are

$$\hat{\theta}_p = \frac{\langle L_p, f \rangle_{L^2(\Omega)}}{\langle L_p, L_p \rangle_{L^2(\Omega)}} = \langle L_p, f \rangle_{L^2(\Omega)} \frac{2p+1}{2} \,,$$

and we have

$$\lim_{P \to \infty} \|f - \bar{g}_P\|_{L^2} = 0 \,.$$

The data-driven setting approximates $\langle L_p, f \rangle_{L^2(\Omega)}$ with a quadrature rule

$$\hat{\theta}_p \approx \tilde{\theta}_p = \frac{2p+1}{2} \sum_{k \in [K]} w_k L_p(x_k) f(x_k) \,. \tag{6}$$

Projection in higher dimensions It will later become necessary to consider $D \geq 2$. In that case, the notion of "degree" must be generalized to allow for differing amounts of cross-interaction terms among the input space. This is accomplished by replacing the sum over $p \in [P-1]_0$ with a sum over $\boldsymbol{p} \in \Lambda \subset \mathbb{N}_0^d$, where Λ is a lower set and the multi-dimensional Legendre polynomials are products of the univariate ones:

$$L_{\boldsymbol{p}}(\boldsymbol{x}) = \prod_{d=1}^{D} L_{p_d}(x_d) \,.$$

In this case, the approximation space becomes

$$\mathbb{P}_\Lambda = \operatorname{span}\{L_{\boldsymbol{p}}(\boldsymbol{x}) : \boldsymbol{p} \in \Lambda\} \,.$$

Definition 2.1 (Lower set (Chkifa et al., 2018))**.** *A set $\Lambda \subset \mathbb{N}_0^D$ is called lower (or downward-closed) if and only if, for each $\boldsymbol{i} \in \Lambda$, $\{\boldsymbol{j} : \boldsymbol{j} \leq \boldsymbol{i}\} \subset \Lambda$. Here, $\boldsymbol{j} \leq \boldsymbol{i}$ if and only if $j_d \leq i_d$ for all $d \in [D]$.*

Example 1. *Two common examples of lower sets are the hyperbolic-cross and total-degree spaces, shown in Figure 2 and given by, respectively,*

$$\Lambda_{HC}(M) = \left\{ \boldsymbol{i} \in \mathbb{N}_0^D : \prod_{d=1}^{D} (i_d + 1) \leq M + 1 \right\} , \tag{7}$$

$$\Lambda_{TD}(M) = \left\{ \boldsymbol{i} \in \mathbb{N}_0^D : \sum_{d=1}^{D} i_d \leq M \right\} . \tag{8}$$

2.1.2 Multi-layer perceptrons

A multi-layer perceptron (MLP) is given by

$$\begin{cases} \tilde{f}(\boldsymbol{x}) & = \boldsymbol{W}_L \boldsymbol{y}_L(\boldsymbol{x}) \\ \boldsymbol{y}_{k+1}(\boldsymbol{x}) & = \sigma(\boldsymbol{W}_k \boldsymbol{y}_k(\boldsymbol{x}) + \boldsymbol{b}_k), \qquad k \in [L-1] \\ \boldsymbol{y}_1(\boldsymbol{x}) & = \sigma(\boldsymbol{W}_0 \, \boldsymbol{x} + \boldsymbol{b}_0) \end{cases} . \tag{9}$$

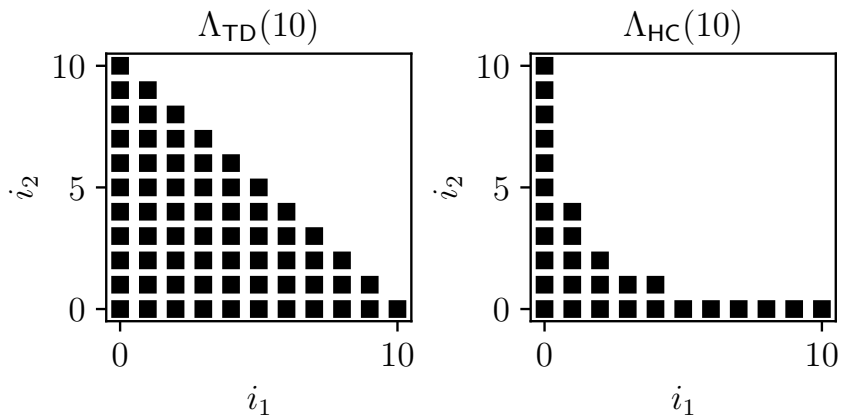

Figure 2: Two examples of lower sets, the total-degree space (left) and hyperbolic cross-section (right).

Here, L is the network depth, $N = \dim(\boldsymbol{b}_k)$ is the uniform network width, and $\sigma(x)$ is a non-polynomial activation function applied component-wise. The network parameters $\boldsymbol{\theta}$ are the concatenation of the entries of each $\boldsymbol{W}_k$ and $\boldsymbol{b}_k$. By counting the size of each $\boldsymbol{W}_k$ and $\boldsymbol{b}_k$, the total number of parameters is $P = N(D+2) + (L-1)(N^2+N)$.

We now state two universal approximation theorems for MLPs that will be foundational for the proofs in Section 3.

Theorem 2.3 (Shallow, wide MLPs (Pinkus, 1999))**.** *Let both $f : \Omega \to \mathbb{R}^Q$ and $\sigma : \mathbb{R} \to \mathbb{R}$ be continuous. Let the network depth be fixed at $L = 1$. Then σ is non-polynomial if and only if, for any $\epsilon > 0$, there exists $N \in \mathbb{N}$, $\boldsymbol{W}_0 \in \mathbb{R}^{N \times D}$, $\boldsymbol{b}_0 \in \mathbb{R}^N$, and $\boldsymbol{W}_1 \in \mathbb{R}^{Q \times N}$ such that $\tilde{f}$ in (9) satisfies*

$$\sup_{\boldsymbol{x} \in \Omega} \|f(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x})\|_{\ell^\infty(\mathbb{R}^Q)} < \epsilon$$

where $\sigma(\cdot)$ is applied componentwise.

Theorem 2.4 (Deep, narrow MLPs (Kidger & Lyons, 2020))**.** *Let $f : \Omega \to \mathbb{R}^Q$ be continuous, and let $\sigma : \mathbb{R} \to \mathbb{R}$ be non-affine. Let the network width N be fixed such that $N \geq D + Q + 2$. Then there exists $L \in \mathbb{N}$, weights $\{\boldsymbol{W}_k\}_{k \in [L]_0}$, and biases $\{\boldsymbol{b}_k\}_{k \in [L-1]_0}$ such that $\tilde{f}(\boldsymbol{x})$ in (9) satisfies*

$$\sup_{\boldsymbol{x} \in \Omega} \|f(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x})\|_{\ell^\infty(\mathbb{R}^Q)} < \epsilon\,.$$

2.1.3 Kolmogorov–Arnold networks

A deep KAN (Liu et al., 2025, Eq. 2.10–2.12) has the form

$$\tilde{f}(\boldsymbol{x}) = \boldsymbol{\Phi}_L \circ \boldsymbol{\Phi}_{L-1} \circ \cdots \boldsymbol{\Phi}_0(\boldsymbol{x}) \tag{10}$$

where the KAN layer $\boldsymbol{\Phi}_\ell : \mathbb{R}^{N_\ell} \to \mathbb{R}^{N_{\ell+1}}$ is defined as

$$(\boldsymbol{\Phi}_\ell(\boldsymbol{z}))_k = (\boldsymbol{W}_\ell \operatorname{SiLU}(\boldsymbol{z}))_k + \sum_{j \in [N_\ell]} \phi^{(p)}_{\ell,k,j}(z_j), \tag{11}$$

where $\ell \in [L]_0$, $k \in [N_{\ell+1}]$, and $\boldsymbol{W}_\ell \in \mathbb{R}^{N_{\ell+1} \times N_\ell}$. For a given spline order p, the functions ϕ are univariate B-splines of the form

$$\phi^{(p)}(x) = \sum_{n \in [G]} \alpha_n B_{n,p}(x),$$

where G is the size of the grid partition $\{x_n\}_{n\in[G]}$. The splines themselves are defined recursively via

$$B_{n,0} \coloneqq \begin{cases} 1, & x_n \leq x < x_{n+1} \\ 0, & \text{otherwise} \end{cases}$$

and, for $p > 0$,

$$B_{n,p}(x) \coloneqq \frac{x - x_n}{x_{n+p} - x_n} B_{n,p-1}(x) + \frac{x_{n+p+1} - x}{x_{n+p+1} - x_{n+1}} B_{n+1,p-1}(x).$$

The network parameters $\boldsymbol{\theta}$ are the union of all $\boldsymbol{W}_\ell$ and all B-spline coefficients.

2.2 Empirical accuracy estimation

To evaluate the effectiveness a surrogate model, we identify some useful metrics such as *best approximation error*, *sampling error*, and *training robustness*. We will utilize these metrics in Section 5. For clarity of presentation, we take $D = 1$ here.

2.2.1 Best approximation error

This error is determined purely by the choice of $\boldsymbol{\theta} \in \mathbb{R}^P$. The error under consideration is

$$\epsilon(\hat{\boldsymbol{\theta}}) \coloneqq \inf_{g\in\mathcal{F}} \|f - g\|_{L^p_\mu} = \|f - \hat{f}\|_{L^p_\mu}\,, \tag{12}$$

where $\hat{f}$ is defined in (2). Crucially, $\epsilon(\hat{\boldsymbol{\theta}})$ is free from any sampling or training error.

2.2.2 Sampling error

This error is due to the replacement of the continuous L^2 norm in (2) with an empirical norm. Using this norm, we seek to minimize the empirical training loss (4). The corresponding generalization error is given by

$$\epsilon(\tilde{\boldsymbol{\theta}}) = \|f - \tilde{f}\|_{L^2_\mu}$$

where $\tilde{f}$ is given by (3). The meaningful indicator about the effect of sampling would be a comparison between $\epsilon(\hat{\boldsymbol{\theta}})$ and $\epsilon(\tilde{\boldsymbol{\theta}})$. Fixing K, we examine three potential choices of grids on $[-1, 1]$, where $k \in [K]$:

- *Uniform sampling*:

$$x_k \overset{\text{iid}}{\sim} \mathcal{U}([-1,1]), \qquad w_k = 2/K$$

- *Equidistant sampling*:

$$x_k = -1 + \frac{2(k-1)}{K-1}, \qquad w_k = 2/K$$

- *Gauss–Legendre quadrature* (Atkinson, 1989, p. 276):

$$\{x_k\}_{k\in[K]} = L_K^{-1}(\{0\}), \qquad w_k = \frac{-2}{(K+1)L_{K+1}(x_k)L'_K(x_k)}$$

2.2.3 Training error and robustness

When numerically optimizing the parameters of a nonlinear surrogate, typically we can only discover various local minima of (4) instead of a global minimum. However, different local minima of the training loss can result in dramatically different generalization error. Premature stopping of the optimizer further compounds this issue (see Section 4). Much attention has been given to the proper initialization of network weights for desirable generalization error, e.g. through explicit probability distributions (Cyr et al., 2020; Fokina &

Oseledets, 2020; Lee et al., 2024; Lu et al., 2020; Glorot & Bengio, 2010; He et al., 2015) or *meta-learned* initializations (Finn et al., 2017; Nichol et al., 2018; Liu et al., 2022; Penwarden et al., 2023). We examine training robustness by randomly initializing the network many times and reporting the variance of the test error:

$$\operatorname*{Var}_{\boldsymbol{\theta}_{\text{init}}\sim\mu}\left[\epsilon(\tilde{\boldsymbol{\theta}})\right]$$

where μ is the Kaiming uniform distribution (He et al., 2015).

3 Shallow universal polynomial networks (SUPNs)

For one input dimension, we define a *shallow universal polynomial network* as any function of the form

$$f_{N,M}(x) \coloneqq \sum_{n\in[N]} c_n \tanh\left(\sum_{m\in[M-1]_0} a_{n,m}\, T_m(x)\right), \qquad x \in [-1,1], \tag{13}$$

where c_n, and $a_{n,m}$ are trainable parameters and $T_m(x)$ is the degree-m Chebyshev polynomial. Intuitively, a SUPN replaces the first $L-1$ layers of a deep MLP with a Chebyshev polynomial, yielding $P = N(M+1)$. This substantially decreases the number of trainable network parameters, which accelerates training and improves interpretability of the neural network. For our purposes, T_m may be computed either trigonometrically or via the three-term recurrence relation (Atkinson, 1989, p. 211). In principle, any polynomial basis on Ω could be used in (13), but we observed that the Chebyshev basis performs well in practice. We hypothesize that the special approximation properties of Chebyshev polynomials may be responsible for this performance. For instance, among all degree-n polynomials with unit L^∞ norm, T_n has largest leading coefficient, and each local extremum is ± 1, which provides stability when learning $a_{n,m}$. Again, SUPNs differ from Chebyshev KANs through the absence of repeated compositions.

In higher dimensions, we define a multi-dimensional SUPN as

$$f_{N,\Lambda}(\boldsymbol{x}) \coloneqq \sum_{n\in[N]} c_n \tanh\left(\sum_{\boldsymbol{m}\in\Lambda} a_{n,\boldsymbol{m}}\, T_{\boldsymbol{m}}(\boldsymbol{x})\right) \tag{14}$$

where $\Lambda \subset \mathbb{N}_0^D$ is a lower set and $T_{\boldsymbol{m}}(\boldsymbol{x}) = \prod_{d=1}^D T_{m_d}(x_d)$, as in Section 2.1.1. In this multi-dimensional setting, the number of trainable parameters for a SUPN is $P = N(|\Lambda|+1)$; for convenience, we may use M and $|\Lambda|$ interchangeably when context is clear. Multi-dimensional SUPNs are able to model cross-interactions between the inputs with a single layer. In contrast, KANs require compositions to model interaction terms as each layer (11) only has *sums* of univariate functions. In this paper, we only use isotropic Λ, but we plan to incorporate anisotropy in Λ as future work, which would further increase the scalability of SUPNs to higher dimensions.

Remark 3.1 (Multiple outputs)**.** *For clarity of presentation, this paper considers only the scalar-output case. However, SUPNs can straightforwardly handle multiple outputs by inserting an additional index on the inner and outer indices corresponding to the output dimension. This is mathematically equivalent to separate scalar SUPNs for each output, but the implementation will be more efficient due to vectorization.*

3.1 Universal approximation of SUPNs

We present two theorems for the universal approximation properties of (13), which rely on the fixed-width and fixed-depth universal approximation theorems for MLPs. We emphasize that these theorems do not construct the network attaining a given ϵ-accuracy, but they are necessary to establish the suitability of (13) as an approximation basis.

Theorem 3.1 (fixed M, variable N)**.** *Let $f : \Omega \to \mathbb{R}^Q$ be continuous with $\Omega \subset \mathbb{R}^D$ compact. Fix $M \geq D+1$. For any $\epsilon > 0$, there exists $N \in \mathbb{N}$ such that $\|f - f_{N,M}\|_{L^\infty} < \epsilon$.*

Proof. Let $\epsilon > 0$ be arbitrary. By Theorem 2.3, there exists $N \in \mathbb{N}$, $\boldsymbol{W}_0 \in \mathbb{R}^{N\times D}$, $\boldsymbol{b}_0 \in \mathbb{R}^N$, and $\boldsymbol{W}_1 \in \mathbb{R}^{Q\times N}$ such that $\|f - \tilde{f}_N\|_{L^\infty} < \epsilon$, where

$$\tilde{f}_N(\boldsymbol{x}) = \boldsymbol{W}_1 \tanh(\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{b}),$$

with $\tanh(\cdot)$ applied componentwise. Each entry of $\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{b}$ is a linear polynomial and can be exactly represented by the inner sum of (14) with $\{1, x_1, \dots, x_D\}$. This basis corresponds to $\Lambda = \{\boldsymbol{0}, \boldsymbol{e}_1, \dots, \boldsymbol{e}_D\}$, giving $M = |\Lambda| = D + 1$. □ □

Theorem 3.2 (fixed N, variable M)**.** *Let $f : \Omega \to \mathbb{R}^Q$ be continuous, with $\Omega \subset \mathbb{R}^D$ compact. Fix $N \geq D + Q + 2$. For any $\epsilon > 0$, there exists $M \in \mathbb{N}$ such that $\|f - f_{N,M}\|_{L^\infty} < \epsilon$.*

Proof. Fix $N \geq D+Q+2$, and let $\epsilon > 0$. By Theorem 2.4, there exists $L \in \mathbb{N}$, $\{\boldsymbol{W}_k\}_{k\in[L]_0}$, and $\{\boldsymbol{b}_k\}_{k\in[L-1]_0}$ such that

$$\sup_{\boldsymbol{x}\in\Omega} \|f(\boldsymbol{x}) - \tilde{f}_L(\boldsymbol{x})\|_{\ell^\infty(\mathbb{R}^Q)} < \epsilon/2,$$

where $\tilde{f}_L$ is given by (9). Let $\tilde{f}_{L-1}(\boldsymbol{x})$ denote the output of layer, i.e. $\tilde{f}_L(\boldsymbol{x}) = \boldsymbol{W}_L \tanh\big(\tilde{f}_{L-1}(\boldsymbol{x})\big)$. By Theorem 2.1 and the uniform continuity of tanh and $\tilde{f}_{L-1}$, there exists a multi-index set Λ and a vector-valued polynomial $p \in \mathbb{P}_\Lambda(\Omega; \mathbb{R}^Q)$ such that, for all $\boldsymbol{x} \in \Omega$,

$$\|f_{N,M}(\boldsymbol{x}) - \tilde{f}_L(\boldsymbol{x})\|_{\ell^\infty(\mathbb{R}^Q)} < \epsilon/2\,,$$

where $f_{N,M}(\boldsymbol{x}) = \boldsymbol{W}_L \tanh(p(\boldsymbol{x}))$ and $M = |\Lambda|$. The triangle inequality completes the proof. □ □

Remark 3.2. *In principle, any non-polynomial activation function $\sigma(x)$ would suffice in Theorem 3.1 and Theorem 3.2 for universal approximation. However, there are practical and theoretical reasons for using* tanh*. In practice, we observed numerical instabilities when $\sigma(x)$ is not bounded, such as the ReLU activation. Furthermore, the relationship between SUPN and polynomial approximation accuracy in Theorem 3.3 relies on a Taylor expansion whose remainder is bounded by the derivatives of* tanh*.*

3.2 Polynomial approximation bounds

In this section, we present several theorems that bound the best-approximation error of SUPNs by the best-approximation error of polynomials. These theorems relax the assumptions of Section 3.1 and have considerably stronger accuracy guarantees. Theorem 3.3 shows that the best polynomial approximation of $f \in W^{k,p}_\mu(\Omega)$ in the $W^{k,p}_\mu$ norm, where

$$W^{k,p}_\mu(\Omega) \coloneqq \Big\{ f : \Omega \to R \;\Big|\; \|f\|_{W^{k,p}_\mu} < \infty \Big\} \qquad \text{and} \qquad \|f\|_{W^{k,p}_\mu} \coloneqq \max_{|\boldsymbol{\beta}|\leq k,\ \boldsymbol{\beta}\in\mathbb{N}_0^d} \left\| \frac{\mathrm{d}^{|\boldsymbol{\beta}|} f}{\mathrm{d}x_1^{\beta_1}\cdots \mathrm{d}x_d^{\beta_d}} \right\|_{L^p_\mu(\Omega)},$$

can be approximated uniformly by SUPNs. Its proof motivates the use of $\sigma(x) = \tanh(x)$ as our activation function. Corollary 3.1 asserts that if a P-term Chebyshev series achieves a given accuracy, then there is a P-term SUPN that achieves essentially the same accuracy. Hence, the approximation power of SUPNs can at least match polynomials. Theorem 3.4 and Theorem 3.5 derive explicit expressions for a quasi-optimal SUPN in the L^2_μ and L^∞ norms. Finally, part (b) of Theorem 3.5 obtains a best-approximation convergence rate that is polynomially fast in the smoothness of f.

Theorem 3.3 (Polynomial approximation proximity)**.** *Let $f \in W^{k,p}_\mu(\Omega; \mathbb{R})$, with μ a probability measure, $\Omega \subset \mathbb{R}^D$ compact, $k \in \mathbb{N}_0$, and $1 \leq p \leq \infty$. For $\Lambda \subset -\mathbb{N}_0^D$ and with $\mathbb{P}_\Lambda \coloneqq \operatorname{span}_{\boldsymbol{m}\in\Lambda} T_{\boldsymbol{m}}$, define*

$$\epsilon_\Lambda(f) \coloneqq \inf_{q\in\mathbb{P}_\Lambda} \|f - q\|_{W^{k,p}_\mu}.$$

For any $\delta > 0$, there is a polynomial $\tilde{q} \in \mathbb{P}_\Lambda$ and a SUPN of the form (14) *with $P = |\Lambda| + 1$ parameters such that*

$$\|f - \tilde{q}\|_{W^{k,p}_\mu} < \begin{cases} (1+\delta)\epsilon_\Lambda(f), & \text{if } \epsilon_\Lambda(f) > 0 \\ \delta, & \text{if } \epsilon_\Lambda(f) = 0 \end{cases} \qquad \text{and} \tag{15}$$

$$\|f_{N,\Lambda} - \tilde{q}\|_{W^{k,\infty}} < \begin{cases} \delta\,\epsilon_\Lambda(f), & \text{if } \epsilon_\Lambda(f) > 0 \\ \delta, & \text{if } \epsilon_\Lambda(f) = 0 \end{cases}. \tag{16}$$

Proof. We deal explicitly with $k = 0$, $\epsilon_\Lambda(f) > 0$. An analogous proof holds for $\epsilon_\Lambda(f) = 0$. We give the proof for $k > 0$ in Appendix F. By definition of $\epsilon_\Lambda(f)$, there is some polynomial $\tilde{q} \in \mathbb{P}_\Lambda$ satisfying (15). Let $\tilde{q} = \sum_{\boldsymbol{m}\in\Lambda} \alpha_{\boldsymbol{m}} T_{\boldsymbol{m}}$, and define $R \coloneqq \sum_{\boldsymbol{m}\in\Lambda} |\alpha_{\boldsymbol{m}}|$. If $R = 0$, set $c_1 = 0$ in (14) with $N = 1$ to achieve $f_{N,\Lambda} = q = 0$, achieving (16). Otherwise, consider $R > 0$. Note that,

$$\sup_{\boldsymbol{x}\in\Omega} |\tilde{q}(\boldsymbol{x})| \leq \sup_{\boldsymbol{x}\in\Omega} \sum_{\boldsymbol{m}\in\Lambda} |\alpha_{\boldsymbol{m}}||T_{\boldsymbol{m}}(\boldsymbol{x})| \leq R\,.$$

We will construct a SUPN with $(N, M) = (1, |\Lambda|)$ that well-approximates $\tilde{q}$. Let $\sigma(y) = \tanh(y)$. Around $y = 0$, $\sigma(y) \approx y$, and using $|\sigma''(y)| < 2|y|$, Taylor's Theorem with the integral form of the remainder gives, for any $r > 0$,

$$|\sigma(y) - y| < r^3, \qquad\qquad |y| < r\,.$$

Now consider the $(N, M) = (1, |\Lambda|)$ SUPN in (14), with the parameter assignment

$$a_{1,\boldsymbol{m}} = \frac{\alpha_{\boldsymbol{m}}}{S}, \qquad\qquad c_1 = S, \qquad\qquad S \coloneqq \sqrt{\frac{R^3}{\delta\epsilon_\Lambda(f)}}\,.$$

Then, for any $\boldsymbol{x} \in \Omega$,

$$|f_{N,\Lambda}(\boldsymbol{x}) - \tilde{q}(\boldsymbol{x})| = S\left|\sigma\left(\frac{\tilde{q}(\boldsymbol{x})}{S}\right) - \frac{\tilde{q}(\boldsymbol{x})}{S}\right| \leq S\left(\frac{R}{S}\right)^3 = \delta\,\epsilon_\Lambda(f)\,.$$

The $k > 0$ proof introduces no significantly new ideas but is more technical; see Appendix F. □

The next result uses the L^∞ (or $W^{k,\infty}$) approximability by SUPNs of the best-approximating polynomial to establish explicit error bounds of $f_{N,\Lambda}$ in the $W^{k,p}_\mu$ norm.

Corollary 3.1. *Under assumptions of Theorem 3.3, for any $\delta > 0$, there is a SUPN of the form (14) with $P = |\Lambda| + 1$ parameters such that*

$$\|f - f_{N,\Lambda}\|_{W^{k,p}_\mu} < \begin{cases} (1+\delta)\epsilon_\Lambda(f), & \text{if } \epsilon_\Lambda(f) > 0 \\ \delta, & \text{if } \epsilon_\Lambda(f) = 0 \end{cases}\,. \tag{17}$$

Proof. Use $\delta/2$ and the triangle inequality in Theorem 3.3. □

In contrast to the generality of approximability in Sobolev spaces we have shown above, if we specialize to $k = 0$ and $p = 2$, we can construct the near-optimal SUPN weights under the L^2_μ norm. This can be done by explicitly using the argumentation of Section 3.2, even if μ is not the Chebyshev measure.

Theorem 3.4 (Constructive SUPN in L^2_μ norm)**.** *The SUPN guaranteed under Theorem 3.3 with $k = 0$ and $p = 2$ is given by*

$$a_{1,\boldsymbol{m}} = \frac{\tilde{\alpha}_{\boldsymbol{m}}}{S}, \qquad\qquad c_1 = S, \qquad\qquad S \coloneqq \begin{cases} \sqrt{\frac{R^3}{\delta}} & \text{if } \epsilon_\Lambda(f) = 0 \\ \sqrt{\frac{R^3}{\delta\,\epsilon_\Lambda(f)}} & \text{if } \epsilon_\Lambda(f) > 0 \end{cases}$$

where $\{\tilde{\alpha}_{\boldsymbol{m}}\}_{\boldsymbol{m}\in\Lambda}$ satisfy

$$\sum_{\boldsymbol{m}\in\Lambda} \alpha_{\boldsymbol{m}}\phi_{\boldsymbol{m}} = \sum_{\boldsymbol{m}\in\Lambda} \tilde{\alpha}_{\boldsymbol{m}} T_{\boldsymbol{m}}$$

and also

$$\alpha_{\boldsymbol{m}} = \langle \phi_{\boldsymbol{m}}, f\rangle_{L^2_\mu}, \qquad\qquad R = \sum_{\boldsymbol{m}\in\Lambda} |\tilde{\alpha}_{\boldsymbol{m}}|,$$

$$\langle \phi_{\boldsymbol{m}}, \phi_{\boldsymbol{n}}\rangle_{L^2_\mu} = \delta_{\boldsymbol{m},\boldsymbol{n}}\,, \qquad\qquad \phi_{\boldsymbol{m}} \in \mathbb{P}_\Lambda\,.$$

Furthermore, the error $\epsilon_\Lambda(f)$ can be computed via

$$\epsilon_\Lambda(f) = \sqrt{\|f\|^2_{L^2_\mu} - \sum_{\boldsymbol{m}\in\Lambda} |\alpha_{\boldsymbol{m}}|^2} \tag{18}$$

Proof. Apply Theorem 3.3 with the optimal L^2_μ projection coefficients. □

Now, the optimal L^2_μ projection onto Chebyshev polynomials with the measure $\mathrm{d}\mu = \mathrm{d}x/\sqrt{1-x^2}$ is nearly optimal under the L^∞ norm (Geddes, 1978). Hence, we can explicitly construct the network weights guaranteed by the universal approximation theorem (Theorem 3.2). A convergence rate easily follows from Jackson's inequality.

Theorem 3.5 (Constructive SUPN in L^∞ norm and convergence rate)**.** *Let $f \in C(\Omega)$ with $\Omega \subset \mathbb{R}$ compact and $\delta > 0$ be arbitrary.*

(a) We can construct a SUPN (13) with $P = M + 1$ parameters satisfying

$$\|f - f_{1,M}\|_{L^\infty} < \left(4 + \frac{4}{\pi^2}\log(M+1)\right) \inf_{q \in \mathbb{P}_M} \|f - q\|_{L^\infty} + \delta$$

using weights from Theorem 3.4 with $S \coloneqq \sqrt{R^3/\delta}$ and μ being the Chebyshev measure.

(b) If $f \in C^k(\Omega)$ and $f^{(k)}$ is α-Hölder continuous, then there exists $\beta > 0$ such that

$$\|f - f_{1,M}\|_{L^\infty} < \frac{\beta \log(M+1)}{M^{k+\alpha}} + \delta\,.$$

Proof. As shown in Rivlin (1974), the Chebyshev series

$$\tilde{f} = \sum_{m \in [M]_0} \alpha_m T_m(x), \qquad \alpha_m = \frac{\langle T_m, f\rangle_{L^2_\mu}}{\langle T_m, T_m\rangle_{L^2_\mu}}, \qquad \mathrm{d}\mu(x) = \frac{\mathrm{d}x}{\sqrt{1-x^2}}$$

is nearly optimal as a degree-M minimax polynomial, satisfying

$$\|f - \tilde{f}\|_{L^\infty} < \left(4 + \frac{4}{\pi^2}\log(M+1)\right) \inf_{q \in \mathbb{P}_M} \|f - q\|_{L^\infty}\,.$$

Using $S \coloneqq \sqrt{R^3/\delta}$ and the weights from Theorem 3.4 within Section 3.2 yields

$$\|\tilde{f} - f_{1,M}\|_{L^\infty} < \delta,$$

which completes the proof of part (a). Applying part (a) with Jackson's inequality (Theorem 2.2) yields part (b). □

4 Second-order optimizer

As discussed in Section 2.2.3, best approximation error can easily be obscured by error in the training method. The use of second-order optimization methods for training SUPNs is inspired by the similar mathematical structure of that problem and classical inverse problems where such methods have worked well. Second-order optimization methods approximate a Newton step on the optimality condition $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = 0$, as opposed to approximating the direction of steepest descent, as is done in first-order methods. Empirical advantages gained by the use of second-order methods over first-order methods include robustness (i.e., the performance of these methods requires less hyperparameter tuning), the ability to navigate past saddle points and flat regions, and more rapid rates of convergence. We emphasize that second-order methods have already been explored in the context of training deep neural networks; an empirical study of these methods can be found in, e.g., Xu et al. (2020). In our experiments, we use a fixed batch of data, so there is no stochasticity in the calculation of the gradients and Hessians (e.g., as a result of minibatching). However, trust region methods can also be applied in the case of stochastic gradients or Hessians; the complexity and convergence of trust region methods in this setting has been explored in Bollapragada et al. (2018) and O'Leary-Roseberry et al. (2019).

In the case of training SUPNs, DNNs, and KANs, which are generally nonconvex, the Hessians of these networks with respect to their parameters are generally not positive-definite. Trust-region methods are able to handle indefinite Hessian estimates. At every iteration n of a trust-region method, we approximate the loss function $\mathcal{L}(\boldsymbol{\theta})$ locally near the current parameters $\boldsymbol{\theta}_n$ using a model, which is often quadratic:

$$\mathcal{L}(\boldsymbol{\theta}_n + \boldsymbol{s}) \approx \mathcal{L}(\boldsymbol{\theta}_n) + (\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_n))^\top \boldsymbol{s} + \frac{1}{2} \boldsymbol{s}^\top \boldsymbol{H} \boldsymbol{s}, \tag{19}$$

where $\boldsymbol{H}$ is either the Hessian $\nabla^2_\theta \mathcal{L}(\boldsymbol{\theta}_n)$ or an approximation thereof. This model is (approximately) minimized with the constraint that $\|\boldsymbol{s}\|$ is less than some trust radius Δ_n:

$$\left.\begin{array}{ll} \min\limits_{\boldsymbol{s} \in \mathbb{R}^P} & (\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_n))^T \boldsymbol{s} + \frac{1}{2} \boldsymbol{s}^T \boldsymbol{H} \boldsymbol{s} \\ \text{s.t.} & \|\boldsymbol{s}\| < \Delta_n \end{array}\right\}. \tag{20}$$

Note that we have dropped $\mathcal{L}(\boldsymbol{\theta}_n)$ as it does not depend on the step $\boldsymbol{s}$. Letting $\boldsymbol{s}_n$ be a solution (or approximate solution) of the above optimization problem (called the "subproblem"), the proposed iterate $\boldsymbol{\theta}_n + \boldsymbol{s}_n$ is either accepted or rejected. Note that an exact subproblem solution is not required; finding an $\boldsymbol{s}$ that satisfies a fraction of Cauchy decrease is enough (i.e., a fraction of the decrease attained by the minimizer in the direction of the negative gradient). The trust-region radius is adjusted from one iteration to the next by assessing the accuracy of the model. Additional details and specific implementations can be found in, e.g., Nocedal & Wright (2006) and Conn et al. (2000).

For our experiments, we use a trust region method, where the subproblem solver is the Steihaug–Toint conjugate gradient (CG) method (Toint, 1981; Steihaug, 1983). We accelerate this optimization method using a limited-memory BFGS Hessian approximation (Liu & Nocedal, 1989) as a preconditioner. This optimization algorithm is implemented using the Rapid Optimization Library (Javeed et al., 2022). This method does not require fully forming a Hessian or computing its inverse; it only requires Hessian–vector products, which can be efficiently calculated with automatic differentiation.

Example 2. *In Figure 3, we show training losses for the Adam (Kingma & Ba, 2017), BFGS (Liu & Nocedal, 1989), and second-order trust-region methods after an initial 1000 epochs of burn-in with Adam (not shown) on the target $f(x) = |x|$. Adam and BFGS use a learning rate of 10^{-2}. Both DNNs and SUPNs converge attain a lower error in far fewer epochs with Newton–CG than with Adam or BFGS. This demonstrates the utility of using a second-order method to determine best-approximation error.*

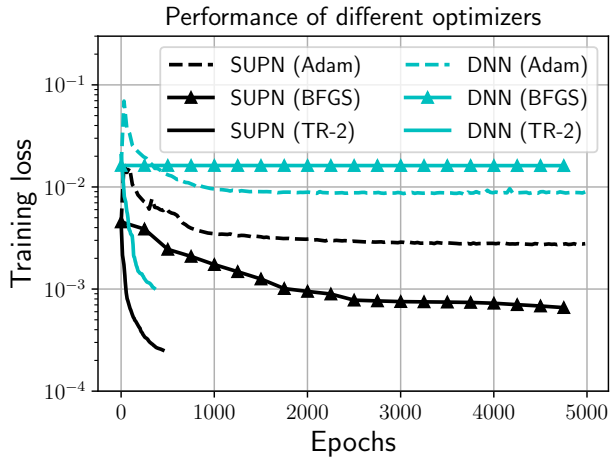

Figure 3: Training losses with Adam, BFGS, and a second-order trust-region method.

5 Numerical experiments

We present several experiments in one, two, and ten dimensions to demonstrate the utility of SUPNs. These experiments are designed to show how SUPNs, DNNs, KANs, and polynomial projection perform on functions

of varying regularity. Importantly, they demonstrate the best-approximation error $\epsilon(\hat{\boldsymbol{\theta}})$, which is free from error arising from insufficient sampling or premature stopping of the optimization method. SUPNs are shown to be superior to DNNs and KANs on smooth functions, and to polynomial projection on nonsmooth functions or functions without tensor-product structure. Furthermore, every method *except SUPNs* is the worst-performing method on at least one target function.

We briefly summarize the commonalities among all examples. Every DNN uses tanh activations in each layer except the output layer. KANs use a grid size of 5, a spline order of 3, and the SiLU activation function, in keeping with Liu et al. (2025). The appendix contains specific values for N and M in each example. All experiments were performed on Nvidia A100 GPUs, except the 10D case, which used H100 GPUs due to memory requirements. The loss functions for neural networks are highly nonconvex in general, so we use a large number of allowable optimization iterations. Training consisted of a burn-in period of 5000 epochs with Adam (learning rate 10^{-3}), followed by trust-region Newton–CG. The trust-region algorithm has a maximum of 1000 Newton steps, a gradient tolerance of 10^{-6} and a step tolerance of 5×10^{-5}. At each optimization step, CG has an absolute tolerance of 10^{-4}, a relative tolerance of 10^{-2}, and a maximum of 500 iterations. Finally, five independent realizations of initial network weights are performed for each experiment. Accuracy on a validation set is monitored during training, and the reported accuracy is the test-set accuracy at the minimum validation-set error, averaged over all runs. All data are noise-free since adding noise would create an artificial floor on the approximation error of all methods we consider here.

5.1 One dimension

We first compare best-approximation error and training robustness between SUPNs, DNNs, KANs, and polynomial projection on a suite of standard test problems. Then, we establish sampling requirements of SUPNs at models corresponding to low, medium, and high best-approximation accuracy. Next, we demonstrate that the order of convergence of SUPNs on the Runge function is independent of its bandwidth. Finally, we show that SUPNs can fit a highly oscillatory sine function from Abueidda et al. (2025, Fig. 5) with an extended domain.

5.1.1 Target functions

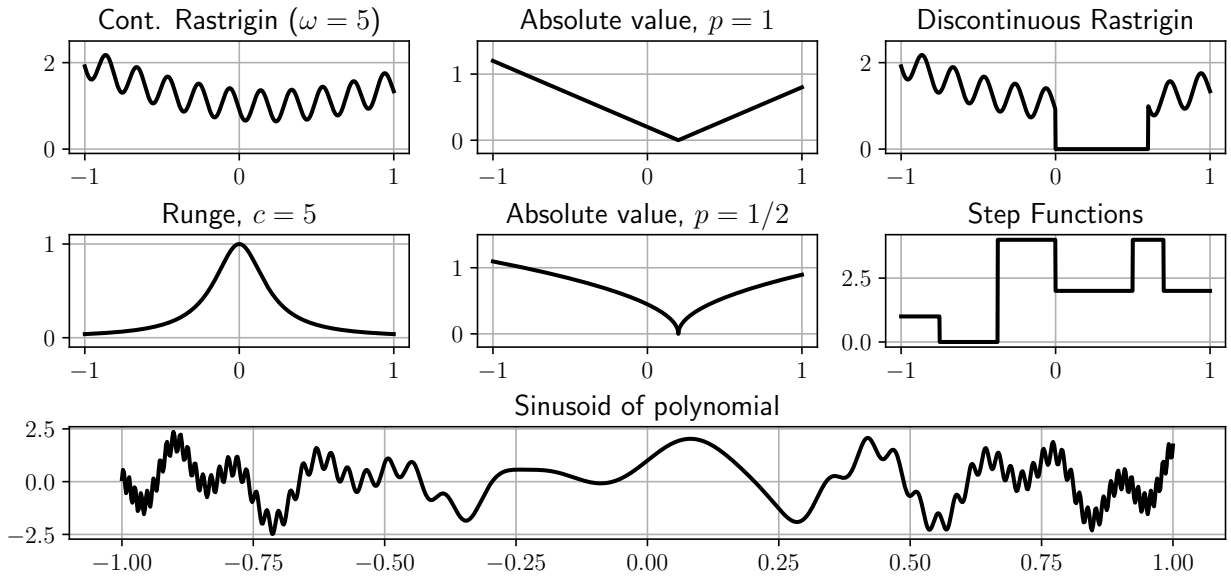

Figure 4: 1D target functions for approximation.

We consider seven 1D functions $f : [-1, 1] \to \mathbb{R}$, shown in Figure 4. These functions are defined below.

1. Continuous Rastrigin:

$$f_1(x;\omega) = 2(x-0.2)^2 - \frac{1}{2.77}\cos(2\pi\omega x - 1.22) + 1 \tag{21}$$

2. Discontinuous Rastrigin:

$$f_2(x) = \begin{cases} 0, & 0 \le x \le 0.6 \\ f_1(x;\omega=5), & \text{otherwise} \end{cases}$$

3. Absolute value of order $p \in (0,1]$:

$$f_3(x;p) = |x-0.2|^p$$

4. Linear combination of step functions:

$$f_4(x) = \begin{cases} 1, & -1 \le x < -0.75 \\ 0, & -0.75 \le x < -0.375 \\ 4, & -0.375 \le x < 0 \ \text{ or } \ 0.5 \le x < 0.7 \\ 2, & 0 \le x < 0.5 \ \text{ or } \ 0.7 \le x \le 1 \end{cases}$$

5. Runge function ($c \ge 1$):

$$f_5(x;c) = \frac{1}{1+(cx)^2} \tag{22}$$

6. Sinusoid of polynomial:

$$f_6(x) = \sin(2\pi^2 x) + \cos(\pi^3 x^2) + \cos(\pi^4 x^3)\sin(\pi^4 x^3)$$

5.1.2 Best-approximation error

We first examine the error of a local minimizer of (2). We use a 2000-point Gauss–Legendre quadrature rule in the loss function (4). The validation set consists of 3001 equidistant points on $[-1,1]$. The test set consists of 17001 equidistant points on $[-1,1]$, and we report the test error at the minimum validation error encountered during training. We vary N, M, width and depth to create networks with $P \le 2000$; exact details are in the appendix (Table 2). We perform five initializations of network weights from the Kaiming uniform distribution (He et al., 2015).

In Figure 5, we demonstrate that SUPNs are able to attain a given error tolerance with significantly fewer parameters than DNNs or KANs, often by an order of magnitude. For the continuously differentiable targets, i.e. continuous Rastrigin and Runge, projection outperforms all the other methods given sufficiently many trainable parameters, but SUPNs outperform DNNs and KANs. For $f(x) = |x|$, SUPNs have a very slight advantage over DNNs and projection when $10^2 \le P \le 10^3$. For $f(x) = |x|^{1/2}$, SUPNs and DNNs track each other very closely, outperforming polynomial projection; the infinite derivative at $x = 0$ causes stagnation around $P \approx 500$. For discontinuous targets, SUPNs have the fewest parameters at the lowest error threshold. Similar trends hold for the L^∞ norm (except for discontinuous targets, where all methods fail); for brevity, further details are in the appendix (Appendix B).

Figure 6 shows the standard deviation of relative error over five realizations of initial network weights. The standard deviation decreases as the mean relative error decreases, except in a handful of very small networks, which give reliably high error. Comparing Figure 5 and Figure 6 at a fixed number of trainable parameters shows that SUPNs are almost always more robust than DNNs and KANs for a given parameter count. The caveat is $f(x) = |x|^{1/2}$, where DNNs and SUPNs are *equally* robust.

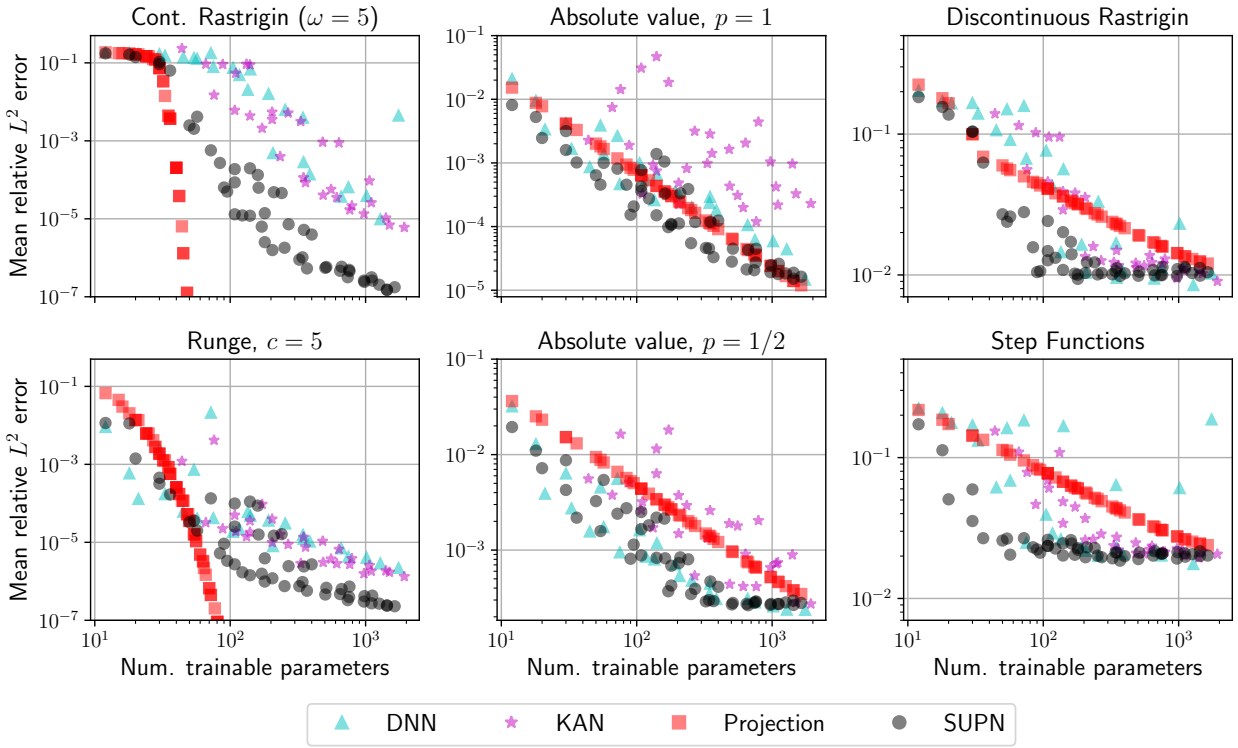

Figure 5: Relative L^2 errors vs. trainable parameters using SUPN, DNN, KAN, and polynomial projection for f_k, $k \in [5]$.

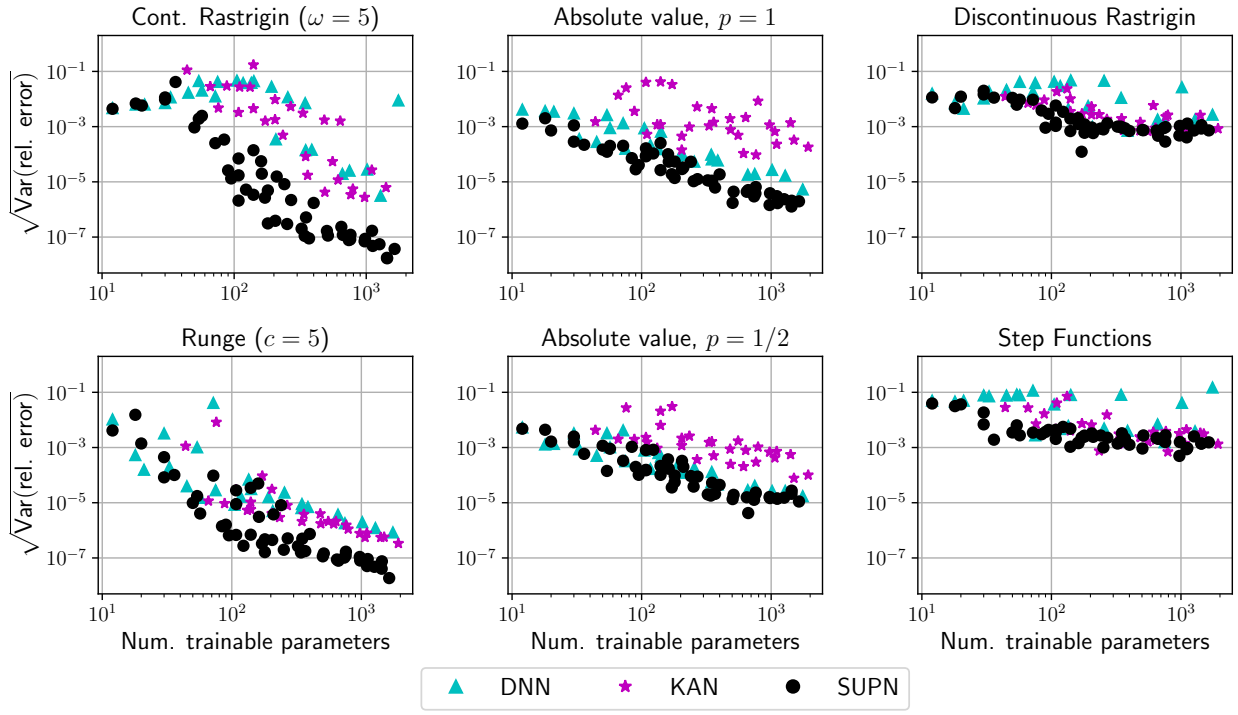

Figure 6: Standard deviation of relative error over 5 realizations of initial network weights.

5.1.3 Effect of hyperparameters

Figure 7 shows a heat map of SUPN, DNN, KAN, and polynomial projection best-approximation error on the sinusoidal target, $f_6(x)$. The bottom row shows how P varies with different choices of N and M. This plot indicates that overly narrow networks (small N) can saturate the approximation capability of a degree-M polynomial in practice (e.g., initializing differently than Theorem 3.4). Likewise, networks with a low polynomial degree M will require a large number N of basis functions. SUPNs have the desirable feature of continuously improving in accuracy in (N, M), instead of abruptly achieving high accuracy at some arbitrary (based on the target function) hyperparameter condition like DNNs or KANs. These observations are consistent with Theorem 2.3 and Theorem 2.4.

5.1.4 Sampling error

Next, we empirically investigate sampling requirements for SUPNs. We vary the ratio between the number of training data K and the number of SUPN parameters, $P = N(M+1)$. We examine models corresponding to qualitatively low, medium, and high best-approximation accuracy from Fig. 5, using the sampling techniques discussed in Section 2.2. Figure 8 indicates that Gauss–Legendre requires the fewest number of samples to achieve best-approximation error. The specific ratio appears to depend on the smoothness of the target function. We see up to $K = P$ for continuous Rastrigin, and $K = 2P$ for absolute value and discontinuous Rastrigin.

5.1.5 Convergence on Runge function

The Runge function (22) is a famous example of a function that can be difficult to approximate with naive methods (Atkinson, 1989). Consider approximating the parameterized Runge function with polynomial projection in the L^2 norm. It can be shown that the best approximation error (12) using the P-term

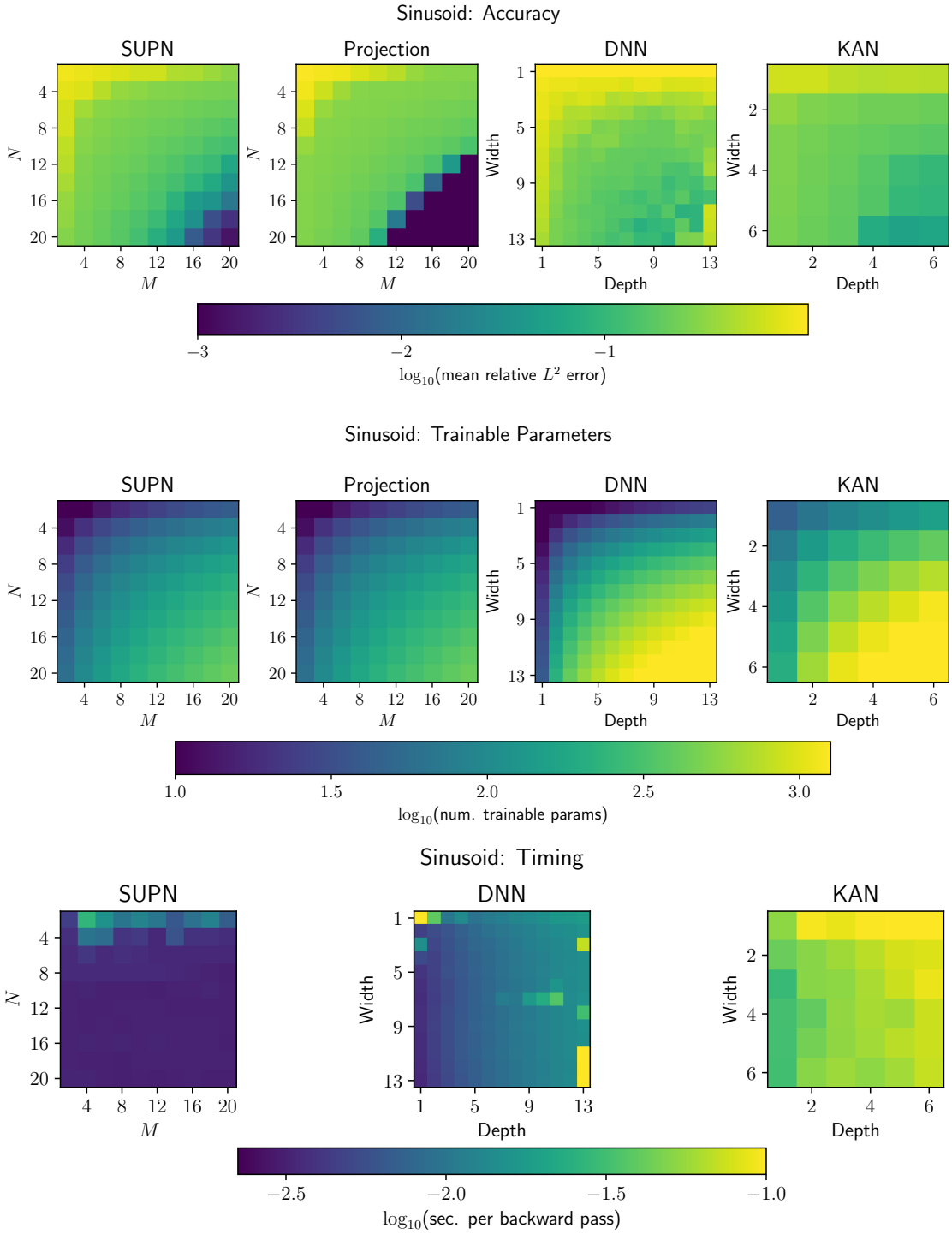

Figure 7: Top: Heat map of test errors for $f_6(x)$ over hyperparameter sweep. Middle: Corresponding parameter counts. Bottom: Time required to compute one backward pass; projection is omitted because it requires no backward passes. Direct numerical comparisons are in Figure 5. Compared to DNNs and KANs, SUPNs display continuous accuracy improvement in (N, M) along with far lower parameter counts and relative errors.

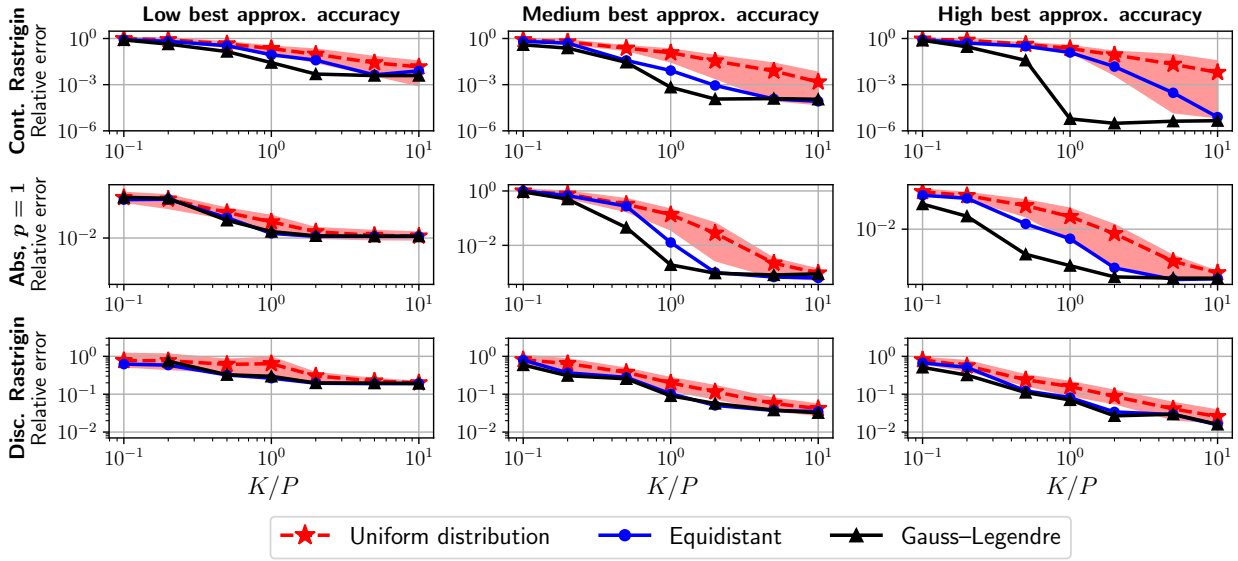

Figure 8: Mean L^2 error (over network initialization) of finite-sampling experiments for different combinations of target function, best approximation accuracy, and sampling strategy. For the "uniform" results, the line is the mean over 10 realizations of training data and 5 weight initializations, while the shaded region is the 10th to 90th percentile.

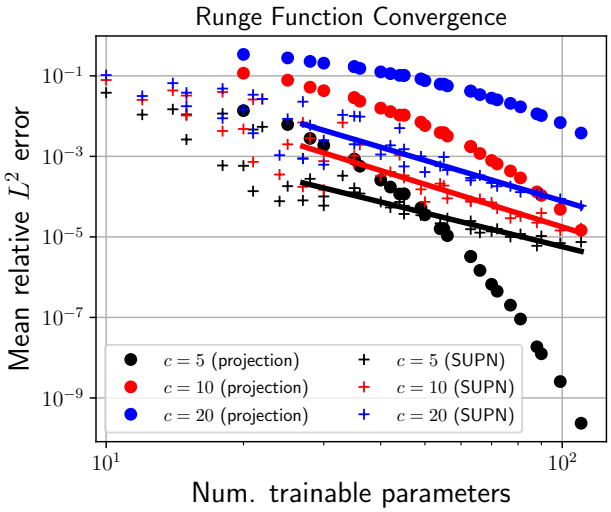

Figure 9: Convergence of SUPN and polynomial projection on Runge function for $c = 5, 10, 20$. Lines of best fit are shown for SUPN.

Legendre series (5) is bounded by

$$\epsilon(\hat{\boldsymbol{\theta}}) \lesssim \exp(-\beta P/c)$$

for some universal constant $\beta > 0$. See, e.g., Wang & Xiang (2012, Theorem 2.1) or Trefethen (2012, Theorem 8.2). This bound gives spectral convergence, but the convergence rate decays with c. Figure 9 shows spectral convergence for polynomial projection that degrades as c increases. In contrast, SUPNs display a finite order of convergence, but that order appears to be independent of c.

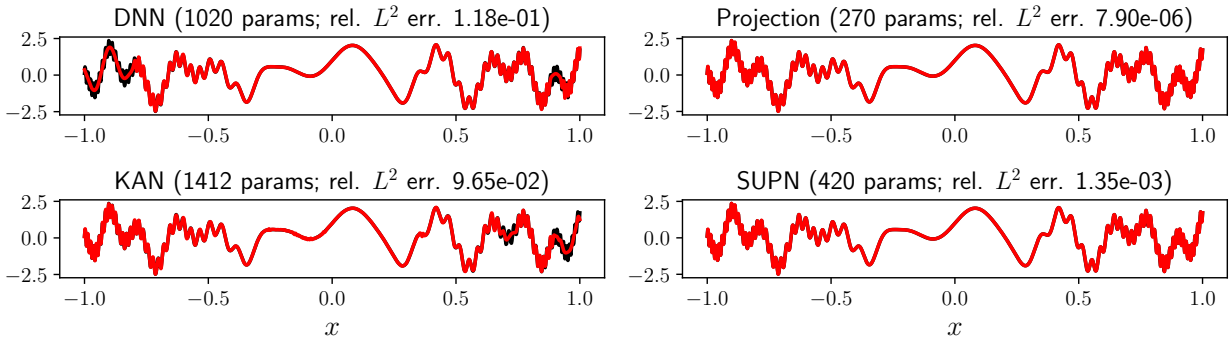

Figure 10: Surrogate predictions on sinusoid example corresponding to median-error model.

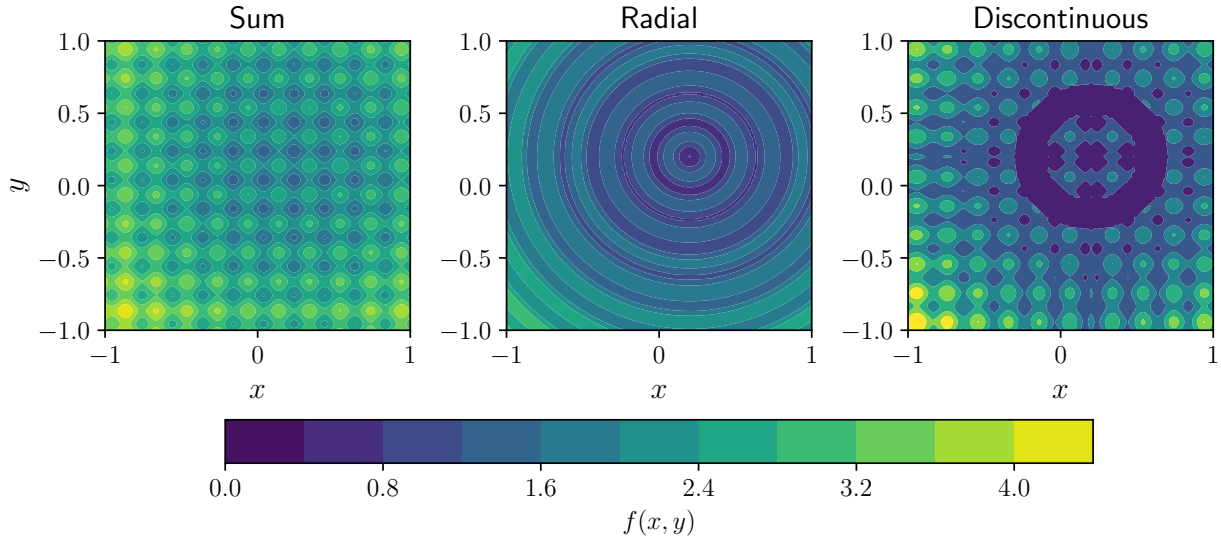

Figure 11: 2D target functions for approximation.

5.1.6 Sinusoid of polynomial

This function uses an extended domain of the problem in Abueidda et al. (2025, Sec. 3.1), in which the authors observe MLPs struggling to fit high-frequency components of the function. We display these results separately from the other functions because larger networks are required than for the experiments in Figure 5. Figure 10 demonstrates that SUPNs and polynomial projection have low error with a small number of parameters, while KANs and especially MLPs struggle at higher frequencies.

5.2 Two dimensions

We now present best-approximation results for functions with two input dimensions. The results of a finite-sampling study are in the appendix (Fig. 17) and qualitatively similar to the one-dimensional case. For both SUPN and projection, we use the isotropic hyperbolic cross-section Λ_{HC} (7) and total-degree space Λ_{TD} (8) for admissible polynomial degrees.

5.2.1 Target functions

We consider three 2D functions $f : [-1, 1]^2 \to \mathbb{R}$, based on the one-dimensional Rastrigin function (21) with $\omega = 5$. These target functions are shown in Figure 11 and are selected to have different amounts of regularity.

1. Continuous Rastrigin (sum):

$$f_7(x, y) = f_1(x; \omega = 5) + f_1(y; \omega = 5)$$

2. Continuous Rastrigin (radial):

$$f_8(x, y) = f_1(r; \omega = 5), \qquad r = \sqrt{(x - 0.2)^2 + (y - 0.2)^2}$$

3. Discontinuous Rastrigin ($\omega = 5$):

$$f_9(x, y) = \begin{cases} 0, & r \in [0.3, 0.5] \\ f_1(|x - 0.2|) f_1(|y - 0.2|), & \text{otherwise} \end{cases}$$

5.2.2 Best-approximation error

In the loss function (4), we use a tensor-product Gauss–Legendre rule with 200 nodes in each dimension. The validation set consists of a 130×130 grid of equispaced points, and the test set consists of a 450×450 grid of equispaced points. Due to the computational cost of a tensor-product grid, the validation and test sets are commensurate in size with the training set, unlike in Section 5.1.2.

Figure 12 shows that SUPNs are a competitive method in 2D. For a sum of one-dimensional Rastrigin functions, SUPNs dramatically outperform DNNs and KANs. SUPNs with a properly chosen index set Λ even outperform polynomial projection with a misspecified Λ. Interestingly, DNNs outperform all methods on the radial target, which we explain by the lack of tensor-product structure in f_8 causing difficulty for polynomial-based methods. However, with a transform into polar coordinates (r, θ), the radial results would become virtually identical to the top left panel of Fig. 5. Moreover, the horizontal spread of KANs on the radial target is much wider than for SUPNs, so the performance of KANs is less predictable with respect to its width and depth. On the discontinuous target, SUPNs have the lowest error among all methods at a given parameter count. Robustness on these 2D examples is qualitatively similar to Fig. 6 and may be found in the Appendix (Fig. 16).

5.3 Ten dimensions

We now demonstrate that SUPNs can scale into moderately large dimensions. We consider the anisotropic and nonlinear target function

$$\begin{aligned} f_{\text{aniso}}(\boldsymbol{x}) = \exp(x_1 - 0.7)\sin(1.3x_2) + 0.2\cos(2\pi x_3) + 0.01|x_4 - 0.27|x_5 \\ + 0.1|x_6|x_7 + 0.05\exp(-(x_8 - 0.3)^2/16) + 0.1x_9x_{10} \end{aligned}$$

where $\boldsymbol{x} \in [-1, 1]^{10}$. We use 10^5 samples from the Halton sequence for training. For validation, we use the next 2×10^5 Halton elements, and for testing, the *next* 2×10^5 Halton elements. As in Section 5.2, all index sets are isotropic.

Figure 13 displays the results. Asymptotically, SUPNs are an order of magnitude more accurate than both KANs and DNNs. To ensure we show best-approximation error rather than sampling error, we also show projection using 10^7 Halton elements to compute the L^2 inner products. With only 10^5 samples, there is significant quadrature error in the higher-order Legendre expansion coefficients ($P \gtrsim 10^3$).

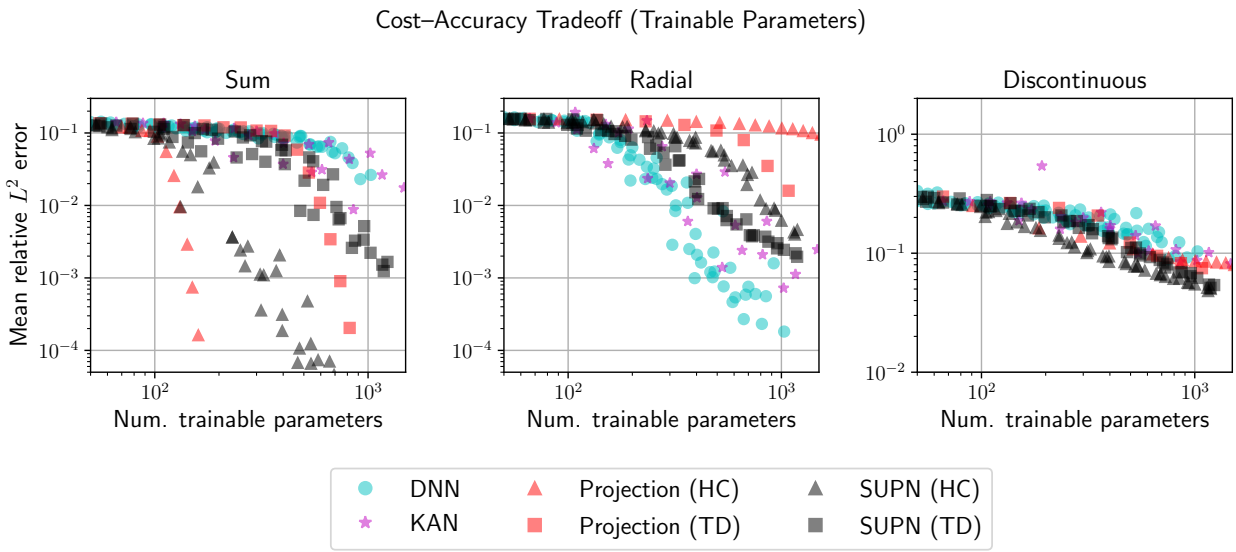

Figure 12: Mean L^2 error versus number of trainable parameters P for 2D examples (f_k, $k = 7, 8, 9$).

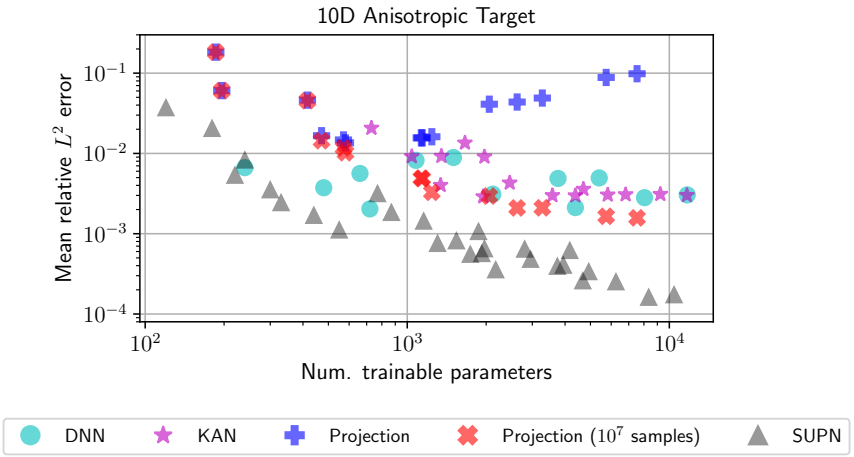

Figure 13: Mean relative L^2 error on anisotropic target function with $D = 10$ input dimensions.

6 Conclusion and future work

We have presented shallow universal polynomial networks, a parsimonious and provably convergent surrogate model that avoids common issues arising from highly over-parameterized models. Intuitively, SUPNs replace the learned bases of a DNN or KAN with a learned polynomial, which dramatically reduces the parameter count of the network. The early layers of an NN learn the same simplistic building blocks, creating unnecessarily complex loss landscapes which can inhibit the learning of high frequency features shown in Section 5. An extensive set of numerical experiments validates that SUPNs outperform DNNs and KANs with on one-, two-, and ten-dimensional target functions that exhibit tensor-product structure. Additionally, the performance of SUPNs is significantly more robust with respect to network initialization.

In future work, we see SUPNs as potential building-blocks for operator learning in one, two, and three spatial dimensions, which are the predominant use-case of neural networks in physics-informed cases. Although MLPs work well in high dimensions, they cannot match traditional approximation methods in lower dimensions. Physics-informed SUPNs may overcome issues fitting higher frequencies in PDEs such as the Helmholtz equation without the need for domain decomposition such as in Moseley et al. (2023). Additionally, we plan to investigate adaptive methods for populating Λ so that SUPNs can exploit anisotropy to scale to even higher dimensions.

Broader Impact Statement

This work does not directly cause the negative impacts listed in the TMLR Ethics Guidelines.

References

Diab W. Abueidda, Panos Pantidis, and Mostafa E. Mobasher. DeepOKAN: Deep operator network based on Kolmogorov Arnold networks for mechanics problems. *Comput. Meth. Appl. Mech. Eng.*, 436:117699, 2025. ISSN 0045-7825. doi: 10.1016/j.cma.2024.117699.

Jonas A. Actor, Graham Harper, Ben Southworth, and Eric C. Cyr. Leveraging KANs for expedient training of multichannel MLPs via preconditioning and geometric refinement, 2025.

Vladimir I. Arnold. *On the representation of functions of several variables as a superposition of functions of a smaller number of variables*, pp. 25–46. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-01742-1_5.

Kendall E. Atkinson. *An Introduction to Numerical Analysis*. Wiley, 1989.

Volker Barthelmann, Erich Novak, and Klaus Ritter. High dimensional polynomial interpolation on sparse grids. *Adv. Comput. Math.*, 12(4):273–288, Mar 2000. ISSN 1572-9044. doi: 10.1023/A:1018977404843.

Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Rev.*, 57(4):483–531, 2015. doi: 10.1137/130932715.

Peter Benner, Mario Ohlberger, Albert Cohen, and Karen Willcox. *Model Reduction and Approximation: Theory and Algorithms*. SIAM, 2017.

Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled Newton methods for optimization. *IMA J. Numer. Anal.*, 39(2):545–578, 2018. doi: 10.1093/imanum/dry009.

Jürgen Braun and Michael Griebel. On a constructive proof of Kolmogorov's superposition theorem. *Constr. Approx.*, 30(3):653–675, 2009. doi: 10.1007/s00365-009-9054-2.

Martin D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003.

C Canuto, M Y Hussaini, A Quarteroni, and T A Zang Jr. *Spectral Methods: Fundamentals in Single Domains*. Springer, 2006.

Abdellah Chkifa, Nick Dexter, Hoang Tran, and Clayton G. Webster. Polynomial approximation via compressed sensing of high-dimensional functions on lower sets. *Math. Comput.*, 87:1415–1450, 2018. doi: 10.1090/mcom/3272.

Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Jiankang Deng, Yannis Panagakis, and Stefanos Zafeiriou. Deep polynomial neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(8): 4021–4034, 2021. doi: 10.1109/TPAMI.2021.3058891.

Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust Region Methods*. SIAM, 2000.

N.E. Cotter. The Stone–Weierstrass theorem and its application to neural networks. *IEEE Trans. Neural Netw.*, 1(4):290–295, 1990. doi: 10.1109/72.80265.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.*, 2(4): 303–314, 12 1989. doi: 10.1007/BF02551274.

Eric C. Cyr, Mamikon A. Gulian, Ravi G. Patel, Mauro Perego, and Nathaniel A. Trask. Robust training and initialization of deep neural networks: An adaptive basis viewpoint. In *Proc. 1st Math. SciML Conf.*, volume 107 of *Proc. Mach. Learn. Res.*, pp. 512–536, 2020.

Sanjoy Das. The polynomial neural network. *Inf. Sci.*, 87(4):231–246, 1995. ISSN 0020-0255. doi: https://doi.org/10.1016/0020-0255(95)00133-6.

Carl de Boor. *A Practical Guide to Splines*. Springer, 1978.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Int. Conf. Mach. Learn.*, pp. 1126–1135, 2017.

Daria Fokina and Ivan Oseledets. Growing axons: Greedy learning of neural networks with application to function approximation, 2020.

K. O. Geddes. Near-minimax polynomial approximation in an elliptical region. *SIAM J. Numer. Anal.*, 15 (6):1225–1233, 1978. doi: 10.1137/0715083.

Roger G. Ghanem and Pol D. Spanos. *Stochastic Finite Element Method: Response Statistics*, pp. 101–119. Springer, 1991.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. 13th Int. Conf. Artif. Intell. Stat.*, volume 9 of *Proc. Mach. Learn. Res.*, pp. 249–256, 2010. URL `https://proceedings.mlr.press/v9/glorot10a.html`.

Alex Gorodetsky, Sertac Karaman, and Youssef Marzouk. A continuous analogue of the tensor-train decomposition. *Comput. Meth. Appl. Mech. Eng.*, 347:59–84, 2019. doi: 10.1016/j.cma.2018.12.015.

David Gottlieb and Steven A. Orszag. *Numerical Analysis of Spectral Methods*. SIAM, 1977.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *2015 IEEE Int. Conf. Comput. Vis.*, pp. 1026–1034, 2015. doi: 10.1109/ICCV.2015.123.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci. USA*, 79(8):2554–2558, 1982. doi: 10.1073/pnas.79.8.2554.

A. G. Ivakhnenko. Polynomial theory of complex systems. *IEEE Trans. Syst. Man Cybern.*, SMC-1(4): 364–378, 1971. doi: 10.1109/TSMC.1971.4308320.

Dunham Jackson. *The Theory of Approximation*, chapter 1. AMS, 1930. Theorem VII.

J. D. Jakeman, M. Perego, D. T. Seidl, T. A. Hartland, T. R. Hillebrand, M. J. Hoffman, and S. F. Price. An evaluation of multi-fidelity methods for quantifying uncertainty in projections of ice-sheet mass change. *Earth Syst. Dyn.*, 16(2):513–544, 2025. doi: 10.5194/esd-16-513-2025.

A. Javeed, D.P. Kouri, D. Ridzal, and G. von Winckel. Get ROL-ing: An introduction to Sandia's Rapid Optimization Library. In *7th International Conference on Continuous Optimization*, 2022.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zidek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, Aug 2021. doi: 10.1038/s41586-021-03819-2.

Subhash C. Kak. Feedback neural networks: New characteristics and a generalization. *Circuits Syst. and Signal Process.*, 12(2):263–278, Jun 1993. ISSN 1531-5878. doi: 10.1007/BF01189877.

Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks, 2020.

Joe Kileel, Matthew Trager, and Joan Bruna. On the expressive power of deep polynomial neural networks. *Adv. Neural Inf. Process. Syst.*, 32, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

A. N. Kolmogorov. On the representations of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Dokl. Akad. Nauk USSR*, volume 114, pp. 953–956, 1957. URL `http://mi.mathnet.ru/dan22050`.

Kaie Kubjas, Jiayi Li, and Maximilian Wiesmann. Geometry of polynomial neural networks. *Algebr. Stat.*, 15(2):295–328, 2024. doi: 10.2140/astat.2024.15.295.

Hyunwoo Lee, Yunho Kim, Seung Yeop Yang, and Hayoung Choi. Improved weight initialization for deep and narrow feedforward neural network. *Neural Netw.*, 176:106362, 2024. ISSN 0893-6080. doi: 10.1016/j.neunet.2024.106362.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Netw.*, 6(6):861–867, 1993. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80131-5.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(1):503–528, 1989. doi: 10.1007/BF01589116.

Xu Liu, Xiaoya Zhang, Wei Peng, Weien Zhou, and Wen Yao. A novel meta-learning initialization method for physics-informed neural networks. *Neural Comput. Appl.*, 34(17):14511–14534, 2022. doi: 10.1007/s00521-022-07294-2.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–Arnold networks, 2025.

Matthew Lowery, John Turnage, Zachary Morrow, John D. Jakeman, Akil Narayan, Shandian Zhe, and Varun Shankar. Kernel neural operators (KNOs) for scalable, memory-efficient, geometrically-flexible operator learning, 2024.

Lu Lu, Shin Yeonjong, Su Yanhui, and George Em Karniadakis. Dying ReLU and initialization: Theory and numerical examples. *Commun. Comput. Phys.*, 28(5):1671–1706, 2020. doi: 10.4208/cicp.OA-2020-0165.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nat. Mach. Intell.*, 6(10):1256–1269, 2024. doi: 10.1038/s42256-024-00897-5.

Zachary Morrow and Miroslav Stoyanov. A method for dimensionally adaptive sparse trigonometric interpolation of periodic functions. *SIAM J. Sci. Comput.*, 42(4):A2436–A2460, 2020. doi: 10.1137/19M1283483.

Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Finite basis physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations. *Adv. Comput. Math.*, 49(4):62, 2023. doi: 10.1007/s10444-023-10065-9.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.

F. Nobile, R. Tempone, and C. G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2309–2345, 2008. doi: 10.1137/060663660.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

Thomas O'Leary-Roseberry, Nick Alger, and Omar Ghattas. Inexact Newton methods for stochastic nonconvex optimization with applications to neural network training, 2019.

Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Comput. Meth. Appl. Mech. Eng.*, 306:196–215, 2016. ISSN 0045-7825. doi: 10.1016/j.cma.2016.03.025.

Michael Penwarden, Shandian Zhe, Akil Narayan, and Robert M Kirby. A metalearning approach for physics-informed neural networks (PINNs): Application to parameterized PDEs. *J. Comput. Phys.*, 477:111912, 2023. doi: 10.1016/j.jcp.2023.111912.

Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numer.*, 8:143–195, 1999. doi: 10.1017/S0962492900002919.

Hoifung Poon and Pedro Domingos. Sum–product networks: A new deep architecture. In *2011 IEEE Int. Conf. Comput. Vis. Workshops*, pp. 689–690, 2011. doi: 10.1109/ICCVW.2011.6130310.

T. J. Rivlin. *Chebyshev Polynomials*, chapter 3. Wiley, 1974. Theorem 3.3.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Int. Conf. on Med. Image Comput. and Computer-Assisted Intervention*, pp. 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28.

Khemraj Shukla, Juan Diego Toscano, Zhicheng Wang, Zongren Zou, and George Em Karniadakis. A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks. *Comput. Meth. Appl. Mech. Eng.*, 431:117290, 2024. doi: 10.1016/j.cma.2024.117290.

S. S. Sidharth, A. R. Keerthana, R. Gokul, and K. P. Anas. Chebyshev polynomial-based Kolmogorov–Arnold networks: An efficient architecture for nonlinear function approximation, 2024.

Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Appl. Comput. Harmon. Anal.*, 43(2):233–268, 2017. doi: 10.1016/j.acha.2015.12.005.

Trond Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.*, 20(3):626–637, 1983. doi: 10.1137/0720042.

Marshall H. Stone. Applications of the theory of Boolean rings to general topology. *Trans. Am. Math. Soc.*, 41:375–481, 1937. doi: 10.1090/S0002-9947-1937-1501905-7.

Philippe L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In *Sparse Matrices and Their Uses*. IMA, 1981.

Lloyd N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Adv. Neural Inf. Process. Syst.*, 30, 2017.

Haiyong Wang and Shuhuang Xiang. On the convergence rates of Legendre approximation. *Math. Comput.*, 81:861–877, 2012. doi: 10.1090/S0025-5718-2011-02549-4.

Dongbin Xiu and George Em Karniadakis. The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.*, 24(2):619–644, 2002. doi: 10.1137/S1064827501387826.

Peng Xu, Fred Roosta, and Michael W. Mahoney. Second-order optimization for non-convex machine learning: An empirical study. In *Proc. 2020 SIAM Int. Conf. Data Mining*, pp. 199–207, 2020. doi: 10.1137/1.9781611976236.23.

Zhongshu Xu, Yuan Chen, and Dongbin Xiu. Chebyshev feature neural network for accurate function approximation. *J. Mach. Learn. Model. Comput.*, 6(2), 2025. doi: 10.1615/JMachLearnModelComput.2025056536.

Shijun Zhang, Hongkai Zhao, Yimin Zhong, and Haomin Zhou. Why shallow networks struggle to approximate and learn high frequencies. *Inf. Inference*, 14(3):iaaf022, 2025. doi: 10.1093/imaiai/iaaf022.

A Symbols and Notation

D	Input dimension	$\mathcal{L}(\cdot)$	Loss function
Q	Output dimension	$\boldsymbol{\theta}$	Learnable parameters
P	Parameter dimension	$\hat{\boldsymbol{\theta}}$	Best-approximation parameters
Ω	Compact subset of $\mathbb{R}^D$	$\tilde{\boldsymbol{\theta}}$	Empirically trained parameters
T_i	Chebyshev polynomial of deg. i	σ	Activation function
L_i	Legendre polynomial of deg. i	$B_{j,i}$	B-spline of deg. i on partition j
K	Training set size	ϵ	Generalization error
$[N]$	$\{1, 2, \ldots, N\}$	$[N]_0$	$\{0, 1, \ldots, N\}$
$\mathbb{P}_n$	Polynomials of degree n		

Table 1: Symbols and notation.

B Supplementary 1D results

B.1 L^∞ best-approximation error

In Figure 14, we observe similar convergence trends in learnable parameter count for L^∞ as we do in L^2 for continuous function approximation cases. The exceptions are that KANs perform noticeably worse on $f(x) = |x|^p$, and SUPNs are now evenly matched on the Runge function. For discontinuous cases, all methods suffer in L^∞.

B.2 Architecture details

We specify the architectures of the 1D networks in Table 2. We performed polynomial projection with the same number of degrees of freedom as SUPNs.

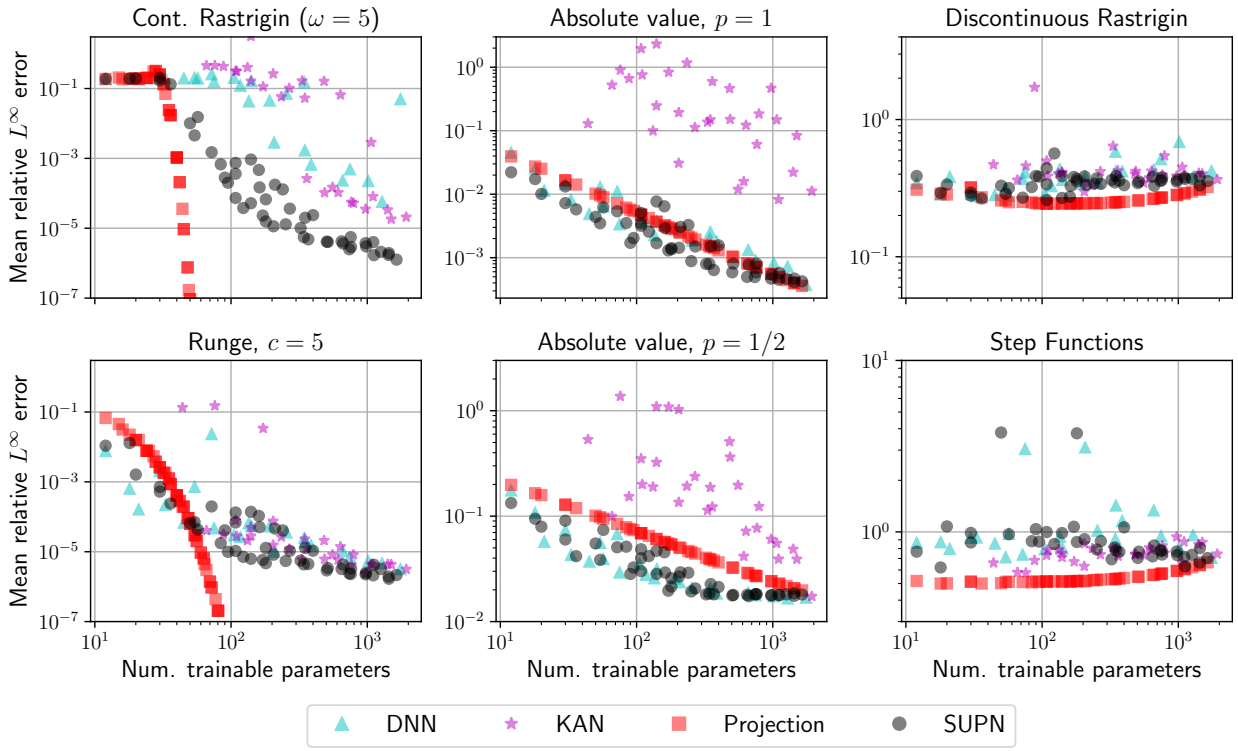

Figure 14: Relative L^∞ errors vs. trainable parameters using SUPN, DNN, KAN, and polynomial projection for f_k, $k \in [6]$.

	SUPN	DNN	KAN
N (width)	3, 5, 9, 18, 27, 35, 40	2, 3, 5, 9, 12	[6]
M (depth)	3, 5, 9, 18, 27, 35, 40	2, 3, 5, 9, 12	[5]

Table 2: Network architectures for 1D examples.

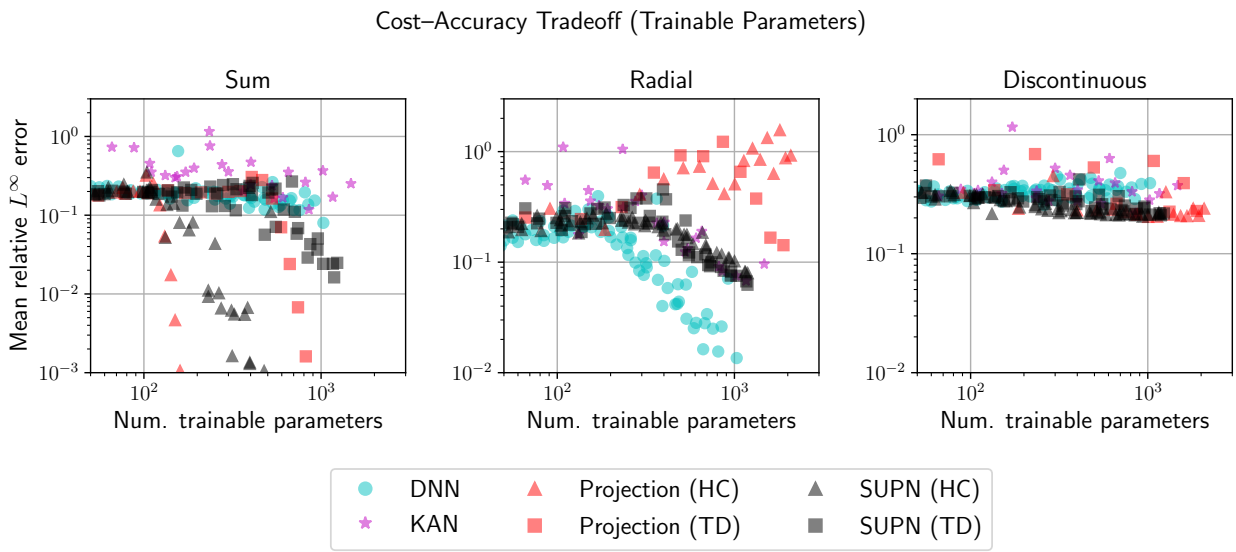

Figure 15: Relative L^∞ errors in 2D.

C Supplementary 2D results

C.1 L^∞ best-approximation error

L^∞ convergence on the 2D examples (Fig. 15) is qualitatively similar to the L^2 case, with two exceptions. First, uniform approximation of a discontinuous function by polynomials is error-prone. Second, the L^∞ accuracy of projection on the radial example has seriously degraded from the L^2 setting, but it appears that (for Λ_{TD}, at least) accuracy begins to improve after $P \approx 1000$. This is consistent with the lack of tensor-product structure in the radial target function.

C.2 L^2 robustness

Figure 16 shows that for $P \gtrsim 100$, SUPNs have between 5-100x less variability in their performance than DNNs and KANs at the same error tolerance. These results comport with those for the one-dimensional examples. Compared to the 1D robustness plots, we do observe an early *increase* in variability up to $P \approx 100$. In this regime, networks have sufficiently few parameters to yield reliable predictions that are nonetheless poor.

C.3 Finite sampling

Figure 17 shows the results of a finite sampling experiment. The horizontal axis is the ratio of the univariate grid size to the square-root of the number of network parameters. The largest ratio needed to obtain best-approximation accuracy is $K_{1\mathrm{D}} \approx 5\sqrt{P}$. Since our networks have $P < 1600$, then our tensor-product Gauss–Legendre rule with 200 nodes in each dimension is sufficient.

C.4 Architecture details

Table 3 describes the architecture for the 2D targets. Here, L refers to the contour of either $\Lambda_{\mathrm{HC}}(L)$ or $\Lambda_{\mathrm{TD}}(L)$.

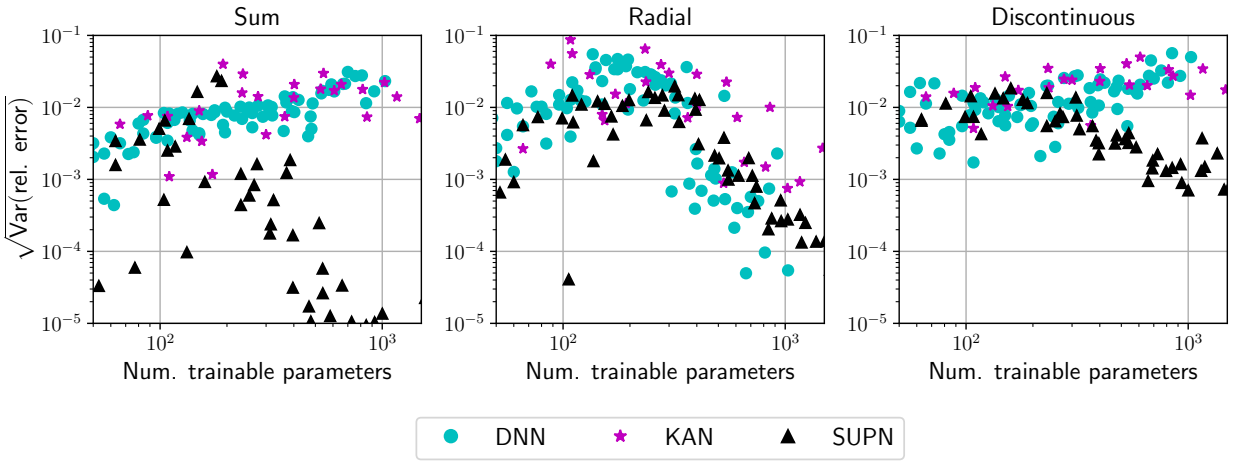

Figure 16: Standard deviation of relative error over 5 realizations of initial network weights for 2D examples. To reduce clutter, only the better-performing of Λ_{TD} and Λ_{HC} are shown for SUPN.

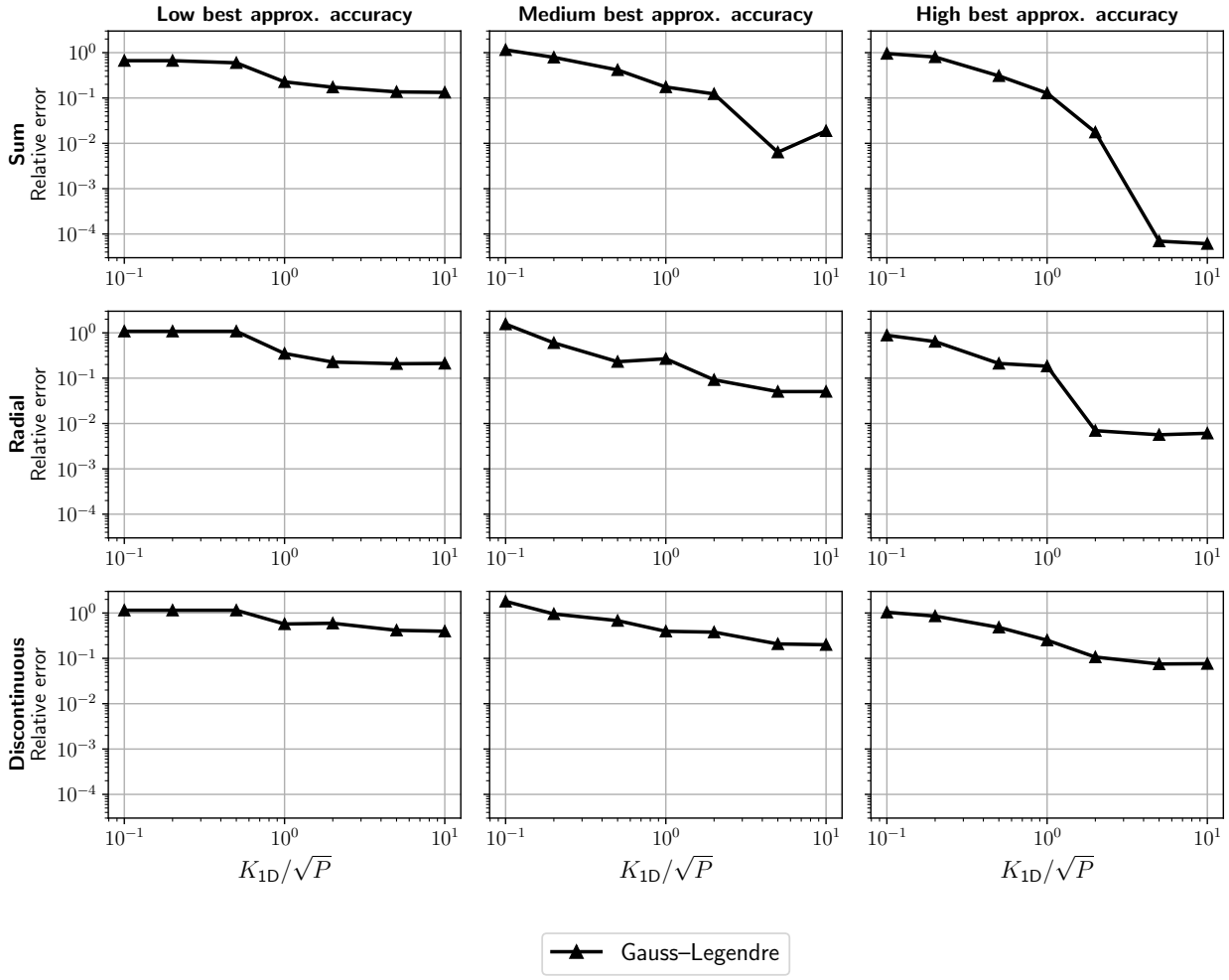

Figure 17: Finite-sampling error on the 2D examples.

	SUPN (HC)	SUPN (TD)	DNN	KAN
N (width)	1, 3, 5, 7, 9, 11, 13, 15	1, 3, 5, 7, 9, 11, 13, 15	[10]	[5]
L (depth)	0, 3, 7, 11, 16, 22, 28, 34	0, 2, 4, 6, 8, 10, 12, 14	[10]	[5]

Table 3: Network architectures for 2D examples.

	SUPN (HC)	DNN	KAN
N (width)	5, 10, 15, 20	20, 40, 60	5, 10, 15
L (depth)	[7]	[4]	[5]

Table 4: Network architectures for 10D example.

D Supplementary 10D results

D.1 Architecture details

Table 4 describes the architecture for the 10D targets.

E Best-approximation predictions

In this section, we plot the predictions corresponding to the 1D step function and continuous Rastrigin targets. These plots provide a qualitative complement to the best-approximation error plots in our results. We sweep over $(N, M) \in [10] \times [10]$. For KANs and DNNs, N denotes network width, and M denotes network depth.

E.1 Step functions

Figures 18–21 demonstrate that the assumption of continuity on the target function is truly necessary for L^∞ convergence. All methods can exhibit persistent spikes over very small sub-intervals of Ω at the discontinuities, analogous to Gibbs effects. Note that we distinguish between the training and testing grid. These spikes are not present on the training grid, since the error of the approximation is include din the loss and therefore minimized at that grid. However, at finer resolutions during testing, there is no constraint (it is necessary to not have a constraint to fit the discontinuities in the first place during training). To fit the discontinuities at the training grid, the basis functions become extreme which can randomly induce spikes in the underlying approximation which is seen in L^∞ but not L^2 since they contribute very little in that norm. This is a limitation of all methods, but is particularly bad for SUPNs, and could be a future research direction to mitigate the effect while still maintaining the ability to fit discontinuities at the training grid level.

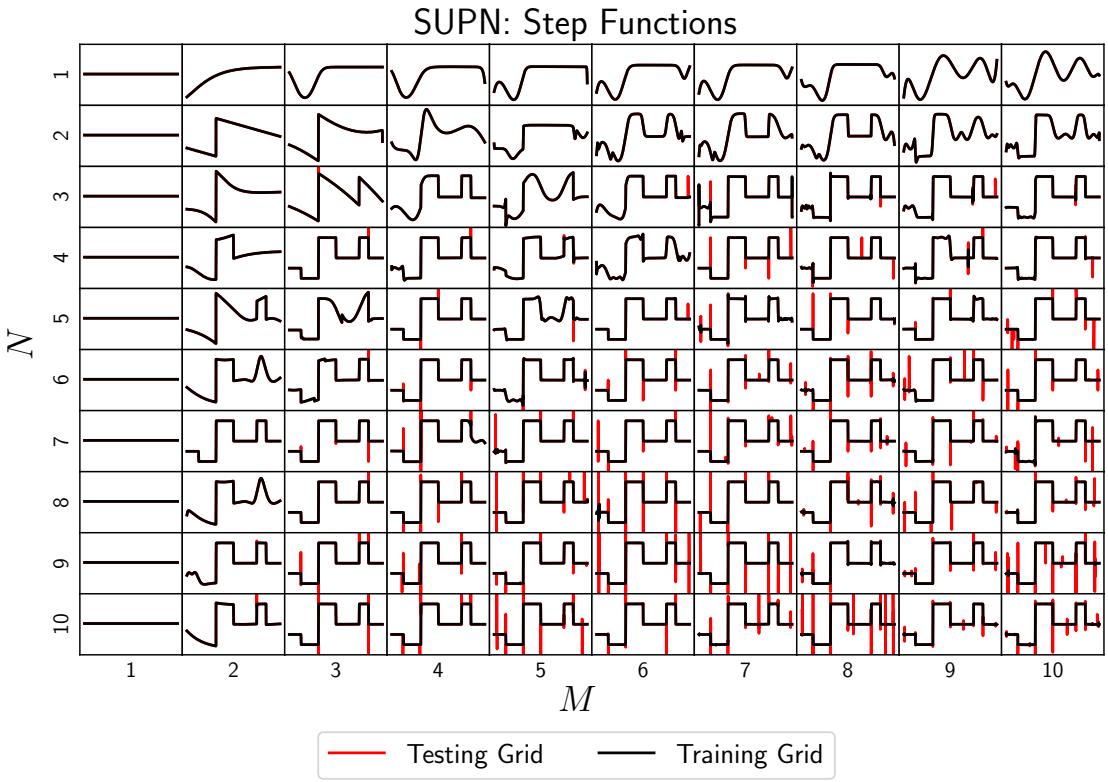

Figure 18: Plot of best approximation with SUPN on the step functions over (N, M) hyperparameter sweep.

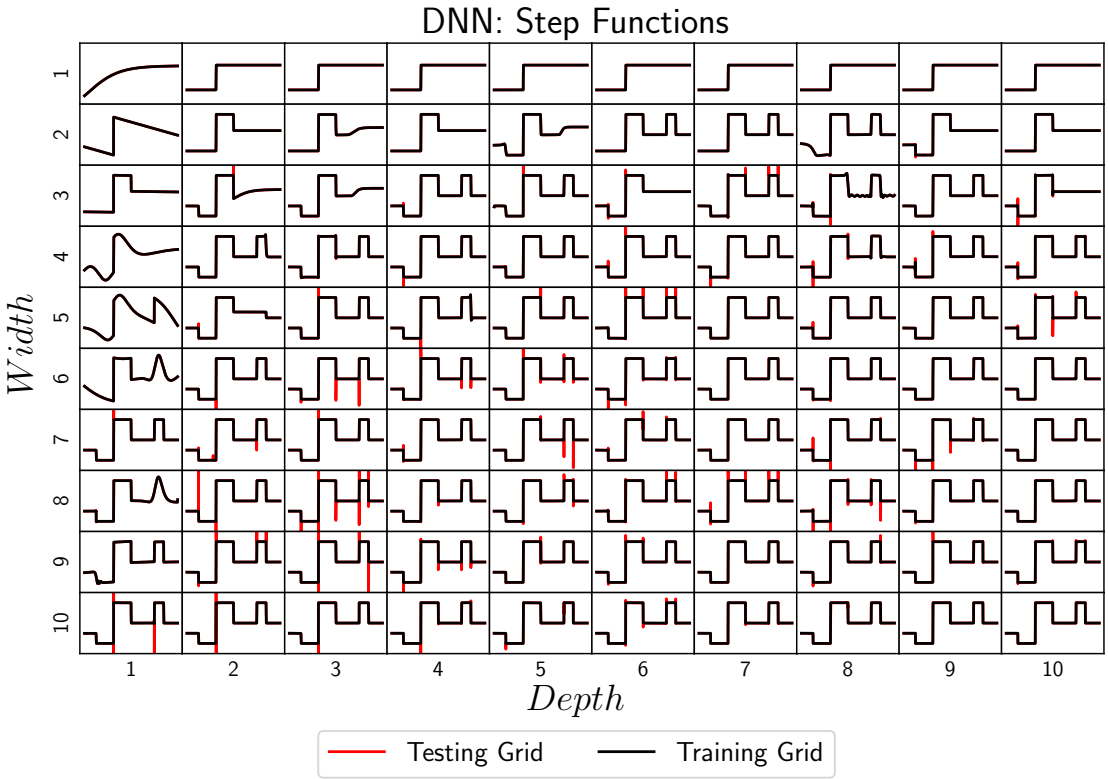

Figure 19: Plot of best approximation with DNN on the step functions over $(Width, Depth)$ hyperparameter sweep.

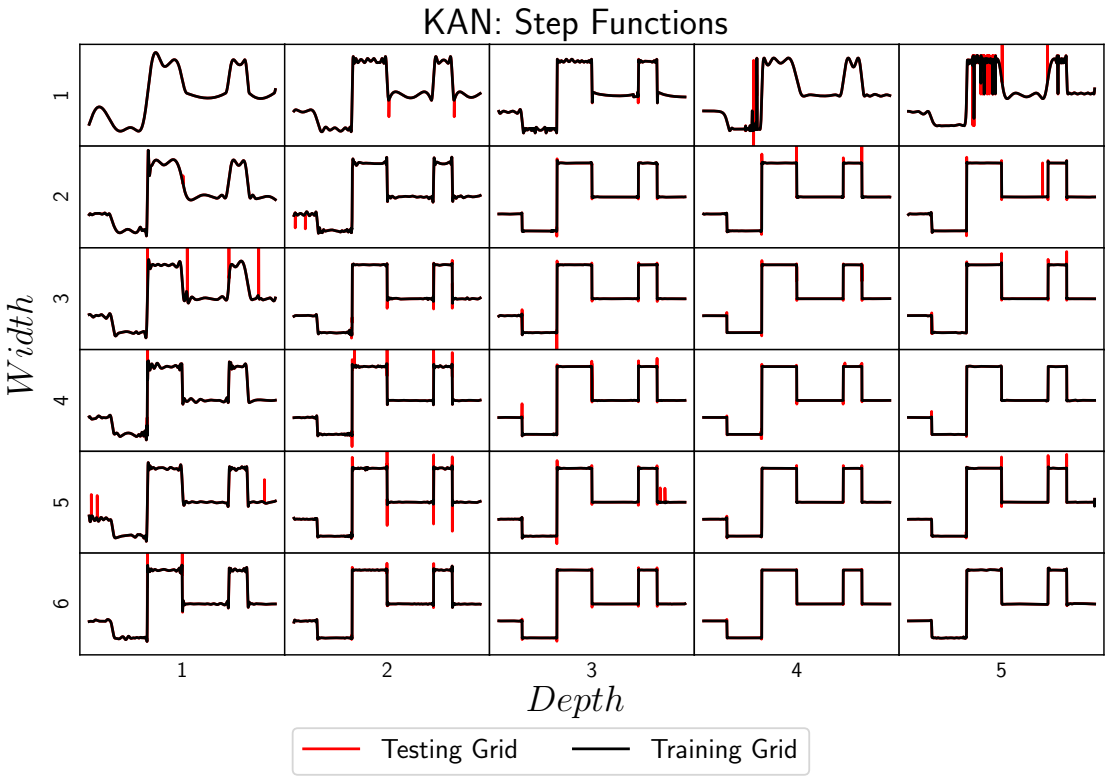

Figure 20: Plot of best approximation with KAN on the step functions over (N, M) hyperparameter sweep.

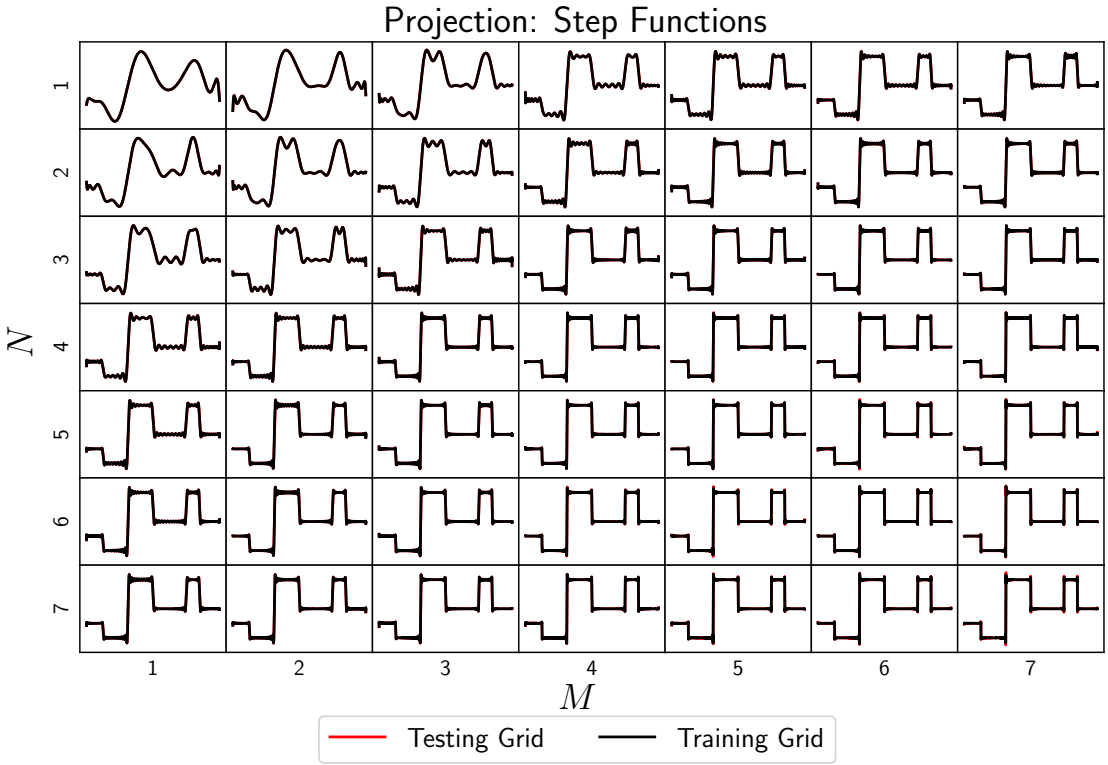

Figure 21: Plot of best approximation with polynomial projection on the step functions over (N, M) hyper-parameter sweep.

E.1.1 Zoomed-in Discontinuity Approximations

Figures 22–24 provide a more nuanced perspective to the discontinuity approximation and how it drives the lack of L_∞ convergence in Figure 14. Consider the initialization 0 results in Figure 22 for the DNN on the test discretization. Despite the approximation perfectly fitting the training points, the method has no knowledge of the analytical location of the discontinuity, so it is misaligned with the target which is observable at higher sampling. Because of this, the L_∞ error for the DNN in this case is 4. Regardless of the spiking behavior observed in all NN-based models (which can be more clearly seen in Figures 22–24), they have no chance of ever converging in L_∞ due to simply lacking the exact location of the discontinuity at higher discretizations. Other noteworthy observations from these plots is how smooth polynomial projection is compared to the NN-based methods when zoomed in, as well as the fact KAN sometimes fails to pass through the training points as seen in Figure 24 initialization 0.

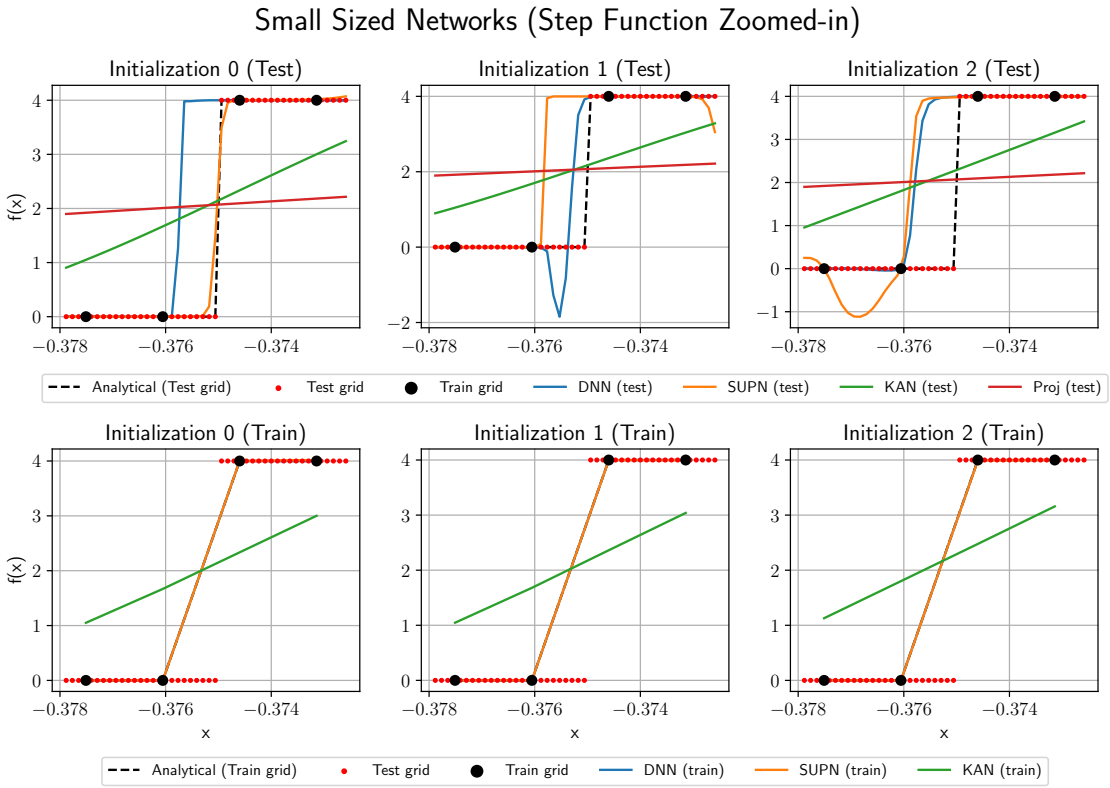

Figure 22: Plot of zoomed in discontinuity at $x = -0.375$ for the step function target. The network approximations shown are drawn from the **5th largest** width/depth or M/N models shown in Figures 18–20, i.e., along the diagonal, for three different model initializations.

E.2 Continuous Rastrigin

Figures 25–28 visually show the approximation methods as a function of architecture size at both the training and testing discretizations. Unlike Appendix E.1, we see very little generalization error between the training and testing discretizations, and spiking is not induced at higher discretizations. This gives credence to the idea that the NN-based models learn extreme basis functions to fit the discontinuity more accurately than polynomial projection. For continuous functions, this phenomena is not observed and models converge with respect to their expressability under proper optimization.

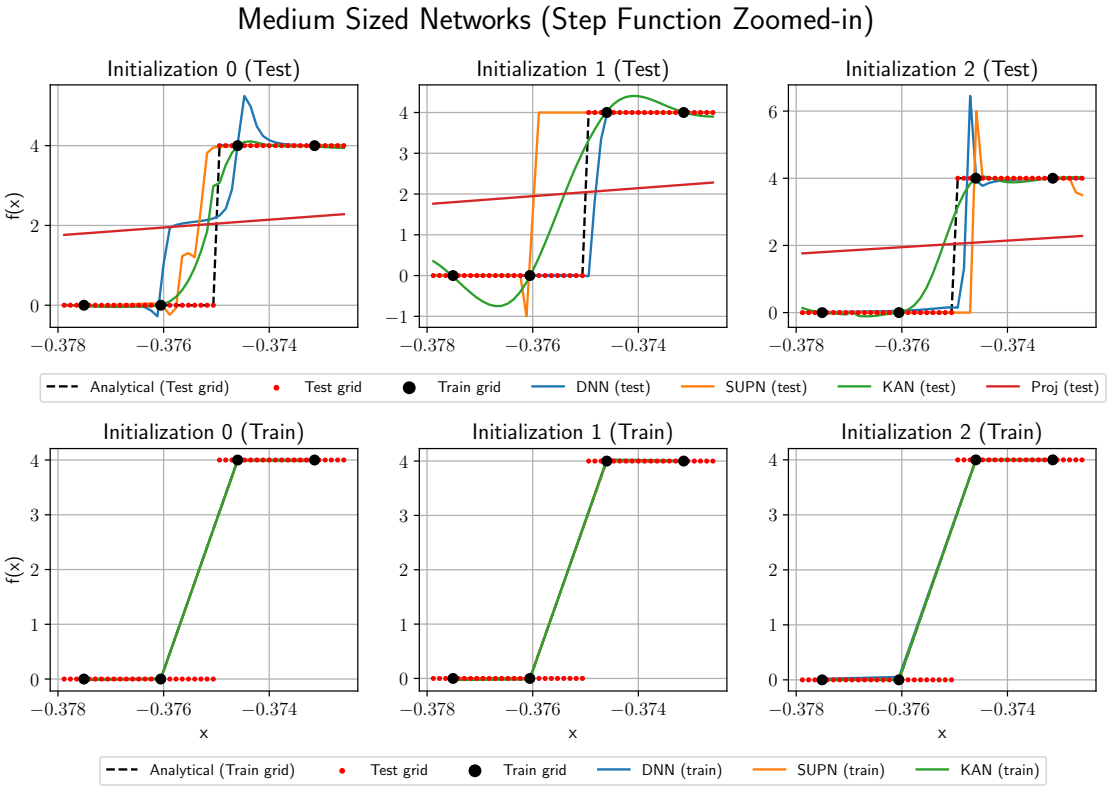

Figure 23: Plot of zoomed in discontinuity at $x = -0.375$ for the step function target. The network approximations shown are drawn from the **3rd largest** width/depth or M/N models shown in Figures 18–20, i.e., along the diagonal, for three different model initializations.

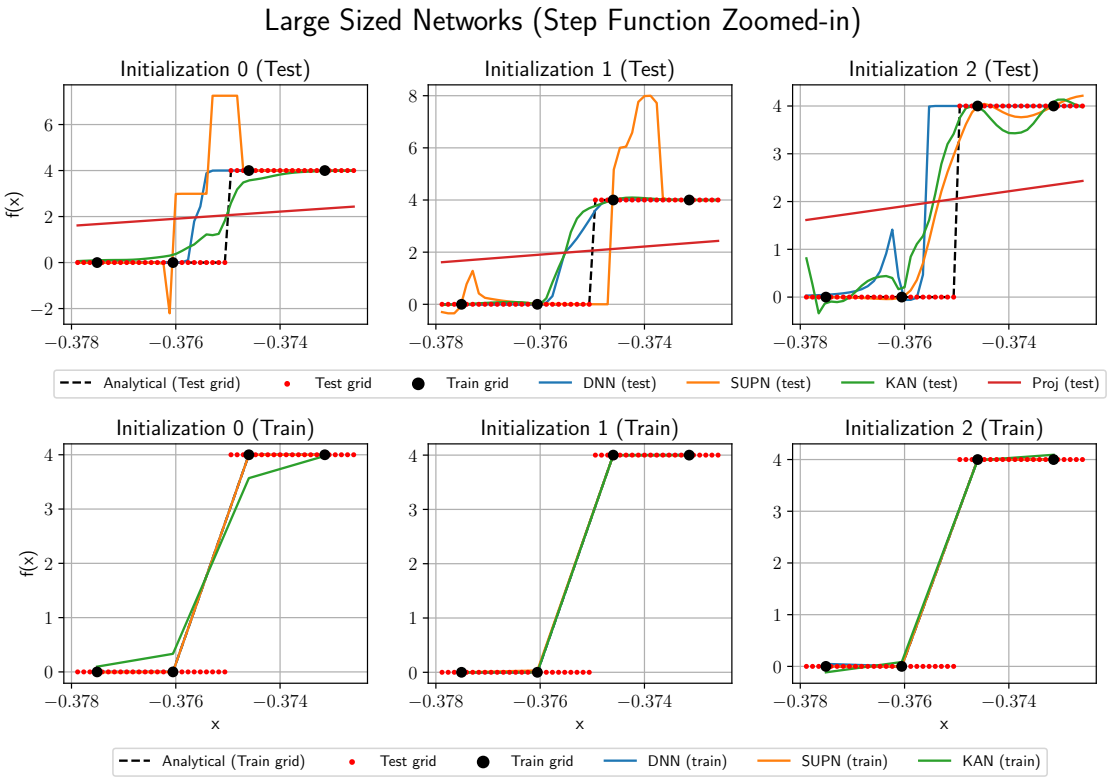

Figure 24: Plot of zoomed in discontinuity at $x = -0.375$ for the step function target. The network approximations shown are drawn from the **largest** width/depth or M/N models shown in Figures 18–20, i.e., along the diagonal, for three different model initializations.

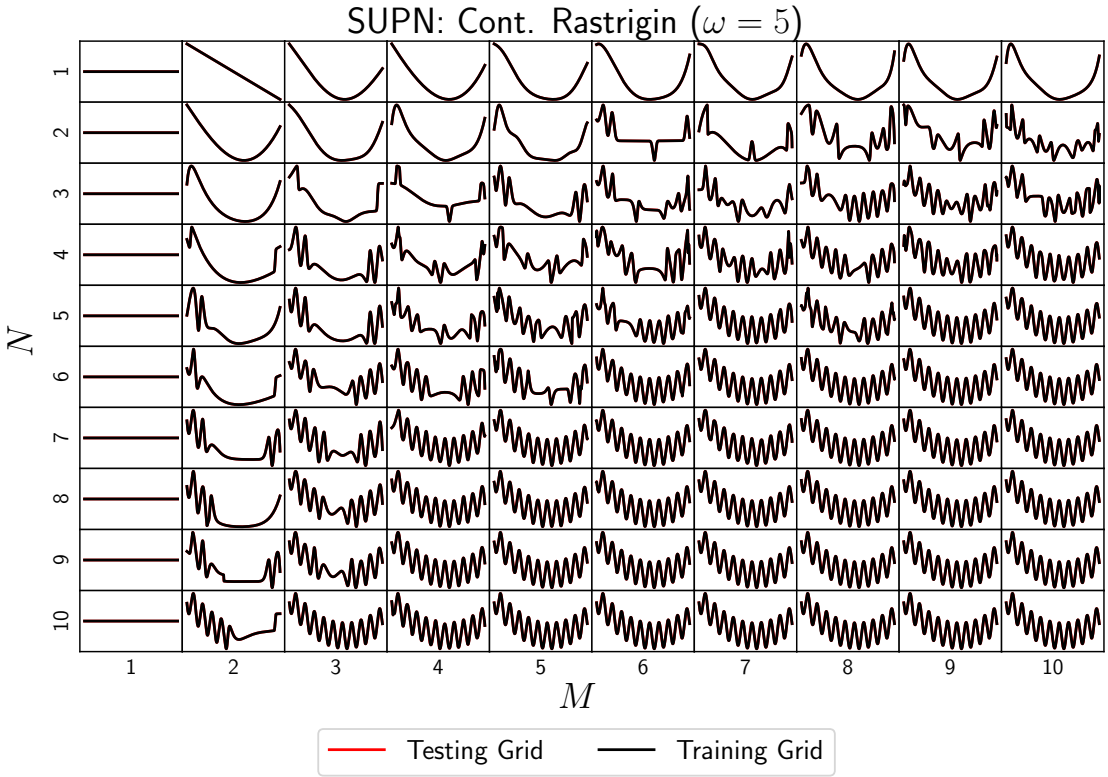

Figure 25: Plot of best approximation with SUPN on the continuous Rastrigin function over (N, M) hyperparameter sweep.

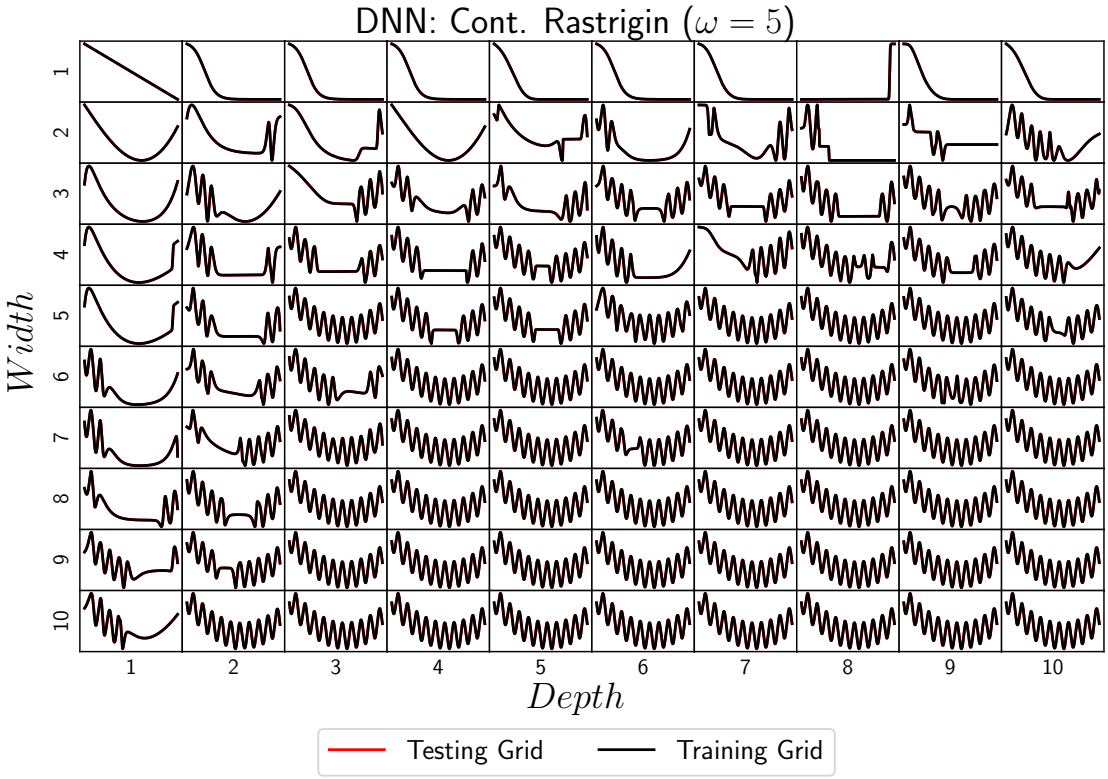

Figure 26: Plot of best approximation with DNN on the continuous Rastirgin function over $(Width, Depth)$ hyperparameter sweep.

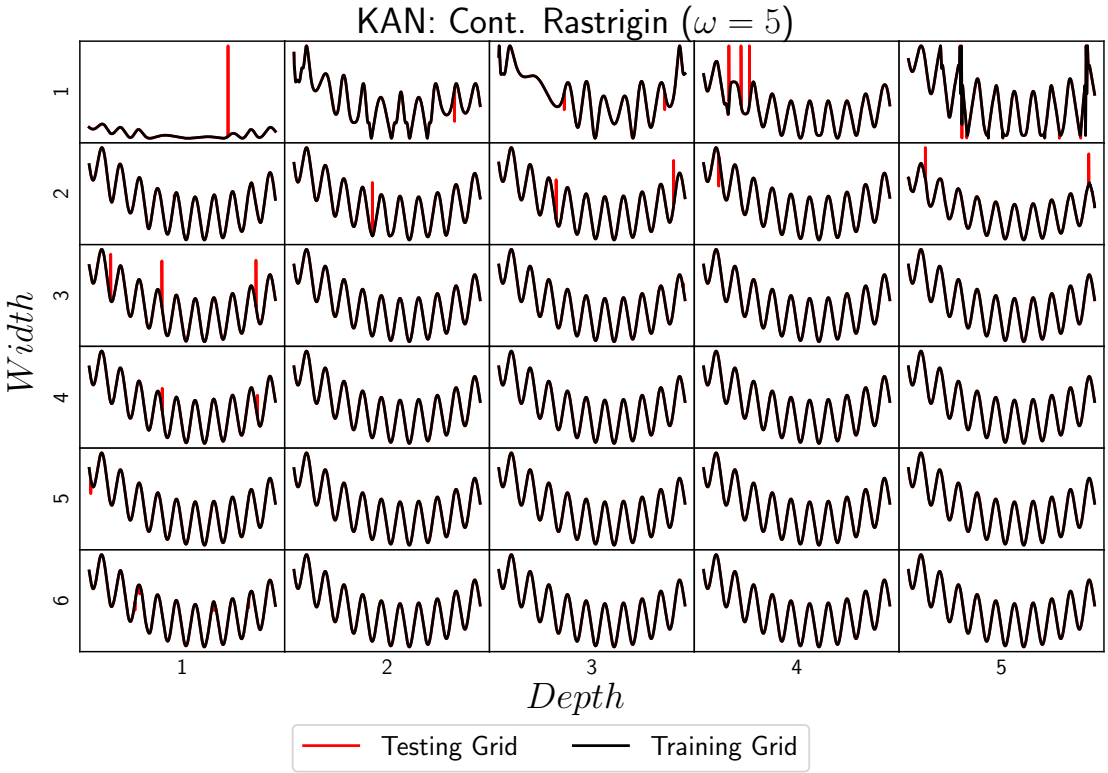

Figure 27: Plot of best approximation with KAN on the continuous Rastirgin function over $(Width, Depth)$ hyperparameter sweep.

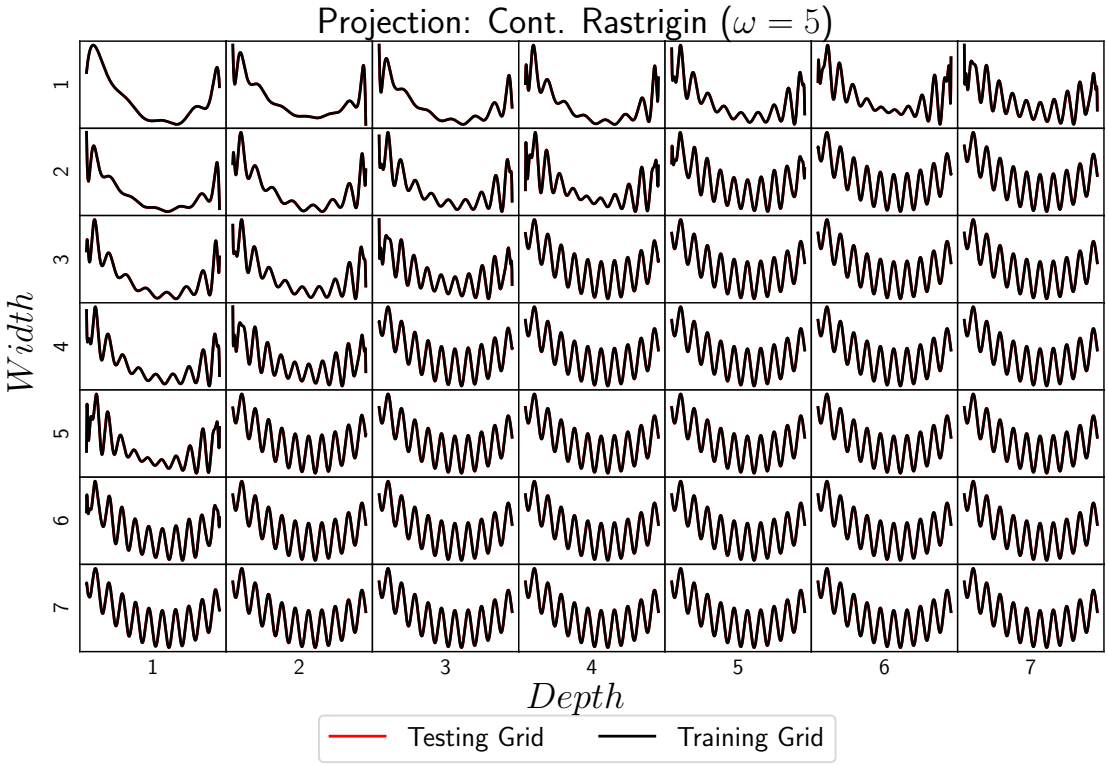

Figure 28: Plot of best approximation with polynomial projection on the continuous Rastrigin function over (N, M) hyperparameter sweep. Each grid cell uses $N(M+1)$ terms, i.e. the same as an SUPN.

F Proof of Sobolev-norm SUPN accuracy bounds

We provide the proof of Theorem 3.3 for $k > 0$ and $\epsilon_\Lambda(f) > 0$. This proof is conceptually the same as the $k = 0$ proof, but involves more opaque constants. By definition of $\epsilon_\Lambda(f)$, there is some polynomial $\tilde{q} \in \mathbb{P}_\Lambda$ satisfying (15). Let $\tilde{q} = \sum_{\boldsymbol{m}\in\Lambda} \alpha_{\boldsymbol{m}} T_{\boldsymbol{m}}$, and define

$$R := \|\tilde{q}\|_{W^{k,\infty}(\Omega)} = \max_{|\boldsymbol{\beta}|\leq k} \sup_{x\in\Omega} \left| \frac{\partial^{\boldsymbol{\beta}}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}} \tilde{q}(x) \right| < \infty.$$

If $R = 0$, set $c_1 = 0$ in (14) with $N = 1$ to achieve $f_{N,\Lambda} = q = 0$, achieving (16). Otherwise, consider $R > 0$. With $\sigma(y) = \tanh y$, we have that,

$$\sigma^{(n)}(0) = \begin{cases} 0, & n \text{ even} \\ \frac{2^{n+1}(2^{n+1}-1)B_{n+1}}{n+1}, & n \text{ odd} \end{cases}$$

where B_n is the nth Bernoulli number. An additional property of σ we require is that there are absolute numbers, (c_n, d_n), such that,

$$\left|\sigma^{(n)}(y) - \sigma^{(n)}(0)\right| \leq c_n|y|, \qquad |y| \leq d_n \leq 1.$$

For example, the above is true for any function with a bounded derivative of order $n+2$, which includes the tanh function. To make computations simpler, we use the looser but sufficient estimate,

$$\left|\sigma^{(n)}(y)\right| \leq a_n, \qquad |y| \leq \tilde{d}_n,$$

where $a_n = \max\{1, \sigma^{(n)}(0)\}$, where $\tilde{d}_n \in (0, d_n)$.

To compute an order-k derivative of an SUPN, Faá di Bruno's formula states that, for a function q with sufficient differentiable smoothness,

$$\begin{aligned} \frac{d^k}{dx^k}\sigma(q(x)) &= \sum_{\ell=0}^{k} \sigma^{(\ell)}(q(x)) B_{k,\ell}\left(q^{(1)}, \ldots, q^{(k-\ell+1)}\right) \\ &= \sigma'(q(x)) B_{k,1}\left(q^{(1)}, \ldots, q^{(k)}\right) + \sum_{\substack{\ell=0 \\ \ell\neq 1}}^{k} \sigma^{(\ell)}(q(x)) B_{k,\ell}\left(q^{(1)}, \ldots, q^{(k-\ell+1)}\right), \end{aligned}$$

where $B_{k,\ell}$ is the k-variate (exponential) incomplete Bell polynomial of degree ℓ. We note that $B_{k,1}(s_1, \ldots, s_k) = s_k$ for all k, and $B_{k,0} = 0$ for any $k > 0$, so that our formula simplifies to,

$$\frac{d^k}{dx^k}\sigma(q(x)) = \sigma'(q(x)) q^{(k)}(x) + \sum_{\ell=2}^{k} \sigma^{(\ell)}(q(x)) B_{k,\ell}\left(q^{(1)}, \ldots, q^{(k-\ell+1)}\right),$$

We will need suprema of the incomplete Bell polynomials over a compact isotropic hypercube of edgelength R:

$$\overline{B}_k := \max_{\ell=0,\ldots,k} \|B_{k,\ell}(z_1, \ldots, z_k)\|_{L^\infty([-R,R]^k)}.$$

A final property of the Bell polynomials we need is that for $r \geq 2$, $B_{r,\ell}(z_1, \ldots, z_r)$ is homogeneous of order ℓ, i.e., $B_{r,\ell}(cz_1, \ldots, cz_r) = c^\ell B_{r,\ell}(z_1, \ldots, z_r)$.

For simplicity, we take the multi-index $\boldsymbol{\beta}$ specifying the derivative to be a (multiple of a) cardinal unit vector, i.e., $\boldsymbol{\beta} = r\boldsymbol{e}_s$, where $r \in [1, k]$, and $\boldsymbol{e}_s \in \mathbb{R}^d$ is the cardinal unit vector in direction s. Then $\partial_{\boldsymbol{\beta}} f = \partial_s^r f$. For

any $r \in [1, k]$,

$$\frac{\partial^r}{\partial x_s^r}\sigma(q(\boldsymbol{x})) - q^{(r,s)}(\boldsymbol{x}) = \underbrace{(\sigma'(q(x)) - 1)\, q^{(r)}(x)}_{\text{(A)}} + \underbrace{\sum_{\ell=2}^{r} \sigma^{(\ell)}(q(x)) B_{r,\ell}\left(q^{(1,s)}, \ldots, q^{(r-\ell+1,s)}\right)}_{\text{(B)}}$$

where for shorthand we've written $q^{(r,s)}$ to denote the r-th partial derivative of q in the sth variable. Now, define q as,

$$q(x) = \frac{1}{S}\tilde{q}(x), \qquad S = \max\{(\delta\epsilon_\Lambda(f))^{-1/k}, (\delta\epsilon_\Lambda(f))^2\} \times \max\left\{1, \frac{R}{\min\{\tilde{d}_1, \ldots, \tilde{d}_k\}}, R^{3/2}, k\left(\overline{B}_k \max\{a_2, \ldots, a_k\}\right)^{1/k}\right\}$$

Then:

$$\begin{aligned}
|(\text{A})| &\leq |\sigma'(q(x)) - 1|\, R/S \leq |q(x)|^2 R/S \leq R^3/S^3 \leq \delta\epsilon_\Lambda(f)\frac{1}{S} \\
|(\text{B})| &\leq \sum_{\ell=2}^{r} |\sigma^{(\ell)}(q(x))| \left|B_{r,\ell}\left(q^{(1,s)}, \ldots, q^{(r-\ell+1,s)}\right)\right| \\
&\leq \sum_{\ell=2}^{r} a_\ell \left|B_{r,\ell}\left(q^{(1,s)}, \ldots, q^{(r-\ell+1,s)}\right)\right| \\
&\leq \sum_{\ell=2}^{r} a_\ell S^{-\ell} \left|B_{r,\ell}\left(\tilde{q}^{(1,s)}, \ldots, \tilde{q}^{(r-\ell+1,s)}\right)\right| \\
&\leq \sum_{\ell=2}^{r} a_\ell S^{-\ell} \overline{B}_r \\
&\leq \frac{1}{S}\sum_{\ell=2}^{k} \frac{a_\ell \overline{B}_k}{S^{\ell-1}}\,.
\end{aligned}$$

If $\delta\epsilon_\Lambda(f) \geq 1$, then

$$|(\text{A})| \leq 1/S \sum_{\ell=2}^{k} \frac{a_\ell \overline{B}_k}{(k\,\overline{B}_k \max\{a_2, \ldots, a_k\})^{(\ell-1)/k}} (\delta\epsilon_\Lambda(f))^{(\ell-1)/k} \leq (1/S)$$

The same estimate holds for all $\boldsymbol{\beta}$ satisfying $|\boldsymbol{\beta}| \leq k$ by taking iterated derivatives in each variable. Now consider the $(N, M) = (1, |\Lambda|)$ SUPN in (14), with the parameter assignment

$$a_{1,\boldsymbol{m}} = \frac{\alpha_{\boldsymbol{m}}}{S}, \qquad c_1 = S,$$

so that $f_{N,\Lambda}(\boldsymbol{x}) = S\sigma(q(\boldsymbol{x}))$. We have shown, for any $\boldsymbol{x} \in \Omega$ and $|\boldsymbol{\beta}| \leq k$,

$$\left|\frac{\partial^{\boldsymbol{\beta}}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}} f_{N,\Lambda}(\boldsymbol{x}) - \frac{\partial^{\boldsymbol{\beta}}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}}\tilde{q}(\boldsymbol{x})\right| = S\left|\frac{\partial^{\boldsymbol{\beta}}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}}\sigma(q(\boldsymbol{x})) - \frac{\partial^{\boldsymbol{\beta}}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}} q(\boldsymbol{x})\right| \leq \delta\epsilon_\Lambda(f)\,,$$

which completes the proof.