Proceedings of Machine Learning Research vol 145:1–20, 2022 3rd Annual Conference on Mathematical and Scientific Machine Learning

Hierarchical partition of unity networks: fast multilevel training

Nathaniel Trask NATRASK@SANDIA.GOV
Center for Computing Research, Sandia National Laboratories,Albuquerque, NM, USA
Amelia Henriksen
Center for Computing Research, Sandia National Laboratories,Albuquerque, NM, USA
Carianne Martinez
Applied Information Sciences Center, Sandia National Laboratories, Albuquerque, NM, USA
Eric C. Cyr ECCYR@SANDIA.GOV
Center for Computing Research, Sandia National Laboratories,Albuquerque, NM, USA

Editors: Bin Dong, Qianxiao Li, Lei Wang, Zhi-Qin John Xu

Abstract

We present a probabilistic mixture of experts framework to perform nonparametric piecewise polynomial approximation without the need for an underlying mesh partitioning space. Deep neural networks traditionally used for classification provide a means of localizing polynomial approximation, and the probabilistic formulation admits a trivially parallelizable expectation maximization (EM) strategy. We then introduce a hierarchical architecture whose EM loss naturally decomposes into coarse and fine scale terms and small decoupled least squares problems. We exploit this hierarchical structure to formulate a V-cycle multigrid-inspired training algorithm. A suite of benchmarks demonstrate the ability of the scheme to: realize for smooth data algebraic convergence with respect to number of partitions, exponential convergence with respect to polynomial order; exactly reproduce piecewise polynomial functions; and demonstrate through an application to data-driven semiconductor modeling the ability to accurately treat data spanning several orders of magnitude.

Keywords: nonparametric regression, deep learning, partition of unity, nonlinear approximation

1. Introduction

We consider the approximation of functions $y \in V$ from scattered data $\mathcal{D} := \{\boldsymbol{x}_i, y_i\}_{i=1}^{N_d}$, where $\boldsymbol{x}_i \subset \mathbb{R}^d$ and V is a Banach space. For datasets with latent low-dimensional manifold structure where $\boldsymbol{x}_i \in \mathcal{M}$, $\dim(\mathcal{M}) = l \ll d$, deep neural networks (DNNs) have been shown to "break the curse of dimensionality" Poggio et al. (2017); Bach (2017). Namely, they provide approximation whose accuracy scales with l, independent of d. This, coupled with the universal approximation property of DNNs, has led to a flurry of interest using DNNs to both construct surrogates in high-dimensional spaces Tripathy and Bilionis (2018); Schwab and Zech (2019) and to solve high-dimensional partial differential equations and inverse problems Han et al. (2018). Additionally, others have shown success using DNNs as meshfree nonlinear approximation spaces to solve forward and inverse problems Lagaris et al. (1998); Raissi et al. (2019); Berg and Nyström (2018). For all of these applications however, DNNs fail to exhibit convergence with increasing network capacity and produce $O(1)$ errors. While the universal approximation property provides guarantees of arbitrarily small error for optimally trained large networks of sufficient size, when training with first-order optimizers a sub-optimal approximation is typically realized and often struggles with

approximation of multiscale data Beck et al. (2019); Wang et al. (2021a,b). In more traditional numerical analysis, mesh refinement forms a cornerstone of verification and validation for modeling and simulation Oberkampf and Roy (2010). Demonstration of solution convergence as the approximation resolution is improved, coupled with an error analysis establishing consistency, indicates convergence to a true solution and provides necessary trust in high consequence engineering environments. If DNNs are to be used in similar high consequence scientific settings, deep surrogates, PDE solvers, and inverse problems must provide similar refinement guarantees.

In the last five years, a number of approximation theoretic works have established the existence of DNNs which converge algebraically with respect to width or depth. Yartotsky established convergence of networks with respect to width and depth Yarotsky (2017, 2018). He et al. (2018) established an interpretation of shallow ReLU feedworward networks as adaptive piecewise linear finite element space, while Opschoor et al. (2020) established that DNNs may actually emulate a much wider range of traditional approximators, such as spectral approximants, free-knot splines, hp-finite element spaces, and radial basis functions. Despite these advances in understanding the potential approximation power of DNNs, these theoretical works do not establish convergence of networks trained with gradient descent. Cyr et al. (2020) introduced a block coordinate descent optimizer and initialization scheme which allowed demonstration of first-order convergence with respect to network width. Motivated by Opschoor et al. (2020)'s construction that DNNs may emulate both partitions of space and monomials, Lee et al. (2021) introduced POU-nets which explicitly incorporate the partition and polynomial emulation into the network architecture: a classification architecture provides a partition of space into N partitions, while a polynomial basis of dimension M associated with each partition forms the localized approximation.

Global polynomial approximants are well-known to require careful choice of nodal degrees of freedom to construct numerically stable approximation, rendering them ineffective for unstructured data Boyd (2001); Ibrahimoglu (2016). Localized polynomial approximants, such as splines or hp-finite element spaces, provide stable approximation but require human-in-the-loop construction of a mesh (partition of space) which is challenging in complex geometries or high dimensions. POU-nets therefore provide a potential means of automating mesh generation while exploiting the desirable approximation properties of polynomials. As we will discuss later however, the training of POU-nets is sensitive to the choice of hyper-parameters with no reliable way to train, and in this work we aim to develop a robust generalization of hybrid DNN/polynomial approximation. For all results presented, we use a single set of hyperparameters and vary only the number of partitions.

We will show that, in expectation, POU-Nets admit interpretation as a mixture of experts (MoE) model Masoudnia and Ebrahimpour (2014) using deep networks as gating functions and polynomials as expert models. Jordan and Jacobs (1994) showed that mixture of expert models may be assembled into a hierarchical architecture and trained with expectation maximization. While the focus of their work was to identify models tailored toward different subpopulations of a data set, we will show that the gating functions in MoE may be used to localize polynomial approximation. Additionally, the hierarchical architecture employed by Jordan prompts a question about whether POU-nets may be reformulated in a manner exploiting hierarchical structure. Approximation of multiscale data has been a major challenge for DNNs Wang et al. (2021b). Yet in traditional scientific computing multilevel approaches are ubiquitous to handling and exploiting multiscale problem structure: e.g. algebraic/geometric multigrid Trottenberg et al. (2000), wavelets Graps (1995); Mallat (1999), fast multipole methods Greengard and Gropp (1990), and fast Fourier methods Cooley and Tukey (1965) all combine hierarchical approximation with fast algorithms. In this work we

present a multilevel scheme for training a hierarchy of POU-nets, inspired by the V-cycle used in geometric multigrid. Reformulating POU-nets as a probabilistic model allow an expectation-maximization (EM) strategy for optimizing alternatively between coarse and fine scales, combining a nested approximation architecture with a multilevel optimization scheme. As a result, the proposed approach, compared to a single level EM strategy, is more robust to getting trapped in suboptimal local extrema during training. This reformulation also admits a number of major advances beyond Lee et al. (2021): the EM strategy allows a trivially parallelizable inversion of a large number of small matrices (solve N decoupled rank M matrices) rather than the single large matrix handle in POU-nets (solve one rank $N \times M$ matrix). This hierarchical parallelism provides new opportunities to accelerate training and inference of DNNs, which currently are dominated by a paradigm of GPU-accelerated back propagation. Other contributions include closed form expressions for uncertainty which follow from the probabilistic formulation.

We proceed by first reviewing aspects of POU-nets in the deterministic setting as presented in Lee et al. (2021). Then a probabilistic reformulation is introduced that matches the deterministic one in expectation. Next, we introduce the hierarchical setting and describe in section Section 4.1 the connection to traditional multilevel schemes, noting that the ELBO used in EM decouples between scales during the V-cycle. Finally, a number of experiments in 1D and 2D explore the ability of the approach to reliably produce localized approximations of both smooth and low regularity data.

2. Deterministic POU-nets

We summarize here the deterministic approach pursued in Lee et al. (2021), where an approximation of y is given as

$$y_{POU}(\boldsymbol{x};\theta) = \sum_{p=1}^{N} \phi_p(\boldsymbol{x},\theta) q_p(\boldsymbol{x}), \tag{1}$$

where N is the number of partitions, ϕ_p is a partition of unity ($\phi_p > 0$, $\sum_p \phi_p = 1$), and q_p is a member of the Banach space $Q = span(\Phi_1, ..., \Phi_M) \subset V$. For this paper, we will consider only the space of m^{th} order polynomials $Q = \pi_m(\mathbb{R}^d)$, though other spaces may be used and are likely to be advantageous. For polynomial Q, with the POU chosen as indicator functions on compactly supported disjoint sets ($\phi_p = \mathbb{1}_{\Omega_p}$, $\Omega_p \cap \Omega_q = \emptyset$ for all p, q) traditional piecewise polynomial approximation spaces are recovered. Instead, we introduce a parameterization of ϕ which may be calibrated from data. If during training parameters are obtained such that ϕ has compact support, one obtains $hp-$convergence as follows.

Theorem 1 *Lee et al. (2021) Consider y_{POU} with $Q = \pi_m(\mathbb{R}^d)$. If $y(\cdot) \in C^{m+1}(\Omega)$ and q_p solves*

$$\min_{q'_p \in Q} \sum_d \left| y_d - \sum_{p=1}^{N} \phi_p(\boldsymbol{x}_d,\theta) q'_p(\boldsymbol{x}) \right|^2 \tag{2}$$

for all p to yield y_{POU}, then

$$\|y^*_{POU} - y\|^2_{\ell_2(\mathcal{D})} \leq C_{m,y} \max_p diam\left(supp(\phi_p)\right)^{m+1}, \tag{3}$$

where $\|\cdot\|_{\ell_2(\mathcal{D})}$ denotes the root-mean-square norm over the training data pairs in $\mathcal{D}$.

In Lee et al. (2021), each step of training consists of a dense linear solve corresponding to the least squares problem Equation 2, followed by a gradient update of θ . The method therefore alternates between updating the polynomial and partition models, aiming to discover a set of partitions which provide optimal piecewise polynomial approximation.

In practice, however, it is not straightforward to find a parameterization of ϕ which provides compact support. In Lee et al. (2021), the authors considered two parameterizations: an RBF network which, by construction, provides smooth localized (but not compactly supported) partitions; and a ResNet which provides good approximation of piecewise polynomial functions but requires a scheduling strategy and careful calibration of hyperparameters to obtain compact partitions during training.

3. Probabilistic POU-Net

POU-nets may be recast in a probabilistic context by describing the partitions in terms of a latent categorical random variable $Z(\boldsymbol{x}) \sim Cat(\pi(\boldsymbol{x}))$ prescribing the probability at a given point $\boldsymbol{x}$ of evaluating the i^{th} expert model $\mathcal{E}_i(\boldsymbol{x})$. By $\pi(\boldsymbol{x})$ we denote a set of discrete probabilities that sum to one for a given $\boldsymbol{x}$.

$$p(Y(\boldsymbol{x}) = y) = \sum_{i=1}^{N} p(\mathcal{E}_i = y)p(Z(\boldsymbol{x}) = i) \tag{4}$$

$$p(Z(\boldsymbol{x}) = i) = \pi_i(\boldsymbol{x}; \theta) \tag{5}$$

$$p(\mathcal{E}_i = y) = p\left(Y(\boldsymbol{x}) = y | Z(\boldsymbol{x}) = i\right) = \mathcal{N}(y; \mu_i(\boldsymbol{x}), \sigma_i^2) \tag{6}$$

$$\mu_i(\boldsymbol{x}) = \mathbf{c}_i^{\mathsf{T}} \boldsymbol{\Phi}(\boldsymbol{x}) \tag{7}$$

where $\mathbf{c}_i \in \mathbb{R}^{dim(Q)}$ and σ_i^2 are expert coefficients to be determined on each partition, consisting of a vector of polynomial coefficients and a Gaussian noise model, respectively. We parameterize the categorical distribution using a deep multinomial classification network consisting of a residual network composed with a softmax activation. The variable θ denotes the networks weights and biases.

The expectation and variance of this mixture of experts model is given in closed form

$$\mathbb{E}[Y](\boldsymbol{x}) = \sum_{i=1}^{N} \pi_i(\boldsymbol{x}; \theta)\mu_i(\boldsymbol{x}) \tag{8}$$

$$\text{Var}[Y] = \sum_{i=1}^{N} \pi_i(\boldsymbol{x}; \theta)(\sigma_i^2 + \mu_i(\boldsymbol{x})^2) - \mathbb{E}[Y](\boldsymbol{x})^2. \tag{9}$$

We note that for the choice $\pi_i = \phi_i$ and $\mu_i = q_i$ this coincides with the deterministic POU-net

$$\mathbb{E}[Y](\boldsymbol{x}) = y_{POU}(\boldsymbol{x}). \tag{10}$$

Unlike Lee et al. (2021) however, the variance formula allows both a heteroscedastic uncertainty estimator and a means of training the model via maximum likelihood estimation. Noting that the partitions act as a latent variable, allows the application of an expectation maximization (EM) optimization strategy.

The complete data log-likelihood function, given by

$$\log L(\theta; X, Y, Z) = \sum_{d=1}^{N_d} \sum_{i=1}^{N} \mathbb{1}_{Z(\boldsymbol{x}_d)=i} \log \left[\pi_i(\boldsymbol{x}_d; \theta)\mathcal{N}(y_d; \mu_i(\boldsymbol{x}_d), \sigma_i^2)\right] \tag{11}$$

is bounded from below by the evidence based lower bound (ELBO)

$$L_{\text{ELBO}}(\theta, \mu, \sigma; \mathcal{D}) = \sum_{d=1}^{N_d} \sum_{i=1}^{N} p(Z = i|Y = y_d) \log \frac{\pi_i(\boldsymbol{x}_d; \theta)\mathcal{N}(y_d; \mu_i(\boldsymbol{x}_d), \sigma_i^2)}{p(Z = i|Y = y_d)}, \tag{12}$$

where we use $\mathcal{D}$ to denote data for X and Y.

Maximization of the ELBO is guaranteed to improve the log-likelihood. To achieve this, we apply the EM method. In the *E-step* we use Bayes rule and the law of total probability to compute the posterior probability

$$w_{id} := p(Z = i|Y = y_d) = \frac{\pi_i(\boldsymbol{x}_d)\mathcal{N}(y_d; \mu_i(\boldsymbol{x}_d), \sigma_i^2)}{\sum_I \pi_I(\boldsymbol{x}_d)\mathcal{N}(y_d; \mu_I(\boldsymbol{x}_d), \sigma_I^2)}. \tag{13}$$

In the *M-step*, we compute the ELBO as follows, neglecting the denominator of Equation 12 as it is approximated by a constant and thus does not effect the maximization problem.

$$L_{\text{ELBO}}(\theta, \mu, \sigma; \mathcal{D}) = \sum_{d=1}^{N_d} \sum_{i=1}^{N} w_{id} \log \left[\pi_i(\boldsymbol{x}_d; \theta)\mathcal{N}(y_d; \mu_i(\boldsymbol{x}_d), \sigma_i^2)\right] \tag{14}$$

$$= \sum_{d=1}^{N_d} \sum_{i=1}^{N} w_{id} \left[\log \pi_i(\boldsymbol{x}_d; \theta) - \frac{1}{2} \log \sigma_i^2 - \frac{1}{2} \left(\frac{y_d - \mu_i(\boldsymbol{x}_d)}{\sigma_d^2}\right)^2 - \frac{1}{2} \log 2\pi\right]. \tag{15}$$

To maximize this loss, stationarity requires

$$\frac{\partial L_{\text{ELBO}}}{\partial \mu} = 0, \qquad \frac{\partial L_{\text{ELBO}}}{\partial \sigma} = 0, \qquad \frac{\partial L_{\text{ELBO}}}{\partial \theta} = 0. \tag{16}$$

For the first condition, for each i we seek an optimal coefficient vector $\mathbf{c}_i$, such that $\mu_i(\boldsymbol{x}) = \sum_{\alpha=1}^{M} c_{i,\alpha}\Phi_\alpha(\boldsymbol{x})$. A straightforward calculation eveals the following closed-form expression for the optimal $\boldsymbol{c}_i$ as the solution of a weighted least squares problem for each partition i.

$$\sum_{d=1}^{N_d} w_{id}\Phi_\alpha(\boldsymbol{x_d})\Phi_\beta(\boldsymbol{x_d})c_{i,\beta} = \sum_{d=1}^{N_d} w_{id}\Phi_\alpha(\boldsymbol{x_d})y_d \tag{17}$$

The optimal polynomial fit is thus given by a least squares problem weighted by the posterior estimation of whether each data point belongs to the partition under consideration. This is remarkable, as it implies that the optimal polynomial representation may be obtained in a trivially parallel manner by solving N decoupled least squares problems each of rank M, in contrast to the deterministic POU-net setting where one solves a single $N \times M$ weighted least squares problem.

Similar to Gaussian mixture models, the global optimizer of the second condition is also given in closed-form as the empirical standard deviation weighted by the posterior:

$$\sigma_i^2 = \frac{\sum_d w_{id} \left(y_d - \mu_i(\boldsymbol{x}_d)\right)^2}{\sum_{d'} w_{id'}} \tag{18}$$

The remaining term in Equation 16 however does not admit a closed form expression due to the deep neural network parameterization of the categorical distribution. We define a loss by taking only the term depending on θ and apply a gradient optimizer.

$$\mathcal{L}(\theta; \mathcal{D}) = \sum_{i,d} w_{id} \log \pi_i(\boldsymbol{x_d}; \theta) = -H(w_{id}, \pi_i(\boldsymbol{x_d}; \theta)) \tag{19}$$

Note that this is equivalent to minimizing the cross entropy $H(w, \pi)$, providing a reinforcement mechanism driving the POUs π toward the posterior distribution $p(Z = i|Y = y_d)$. While we do not pursue this connection further in this work, we remark that this admits interpretation as a policy gradient method in reinforcement learning Sutton et al. (1999): the posterior distribution and Z serve the role of a reward and policy, respectively, driving the network toward a distribution on Z which provides optimal localized approximation.

At each step of training, we update the posterior weights, solve the least squares problems, and perform a gradient update of Equation 19. Optionally, one may update σ via Equation 18 as well, however in Appendix B we provide a study demonstrating that updating σ at each step leads to partitions with less effective approximation properties. This is because a tight estimate of σ over-constrains the space near suboptimal local minima with the optimal representation of noise competing against optimal polynomial fit. Instead, we fix $\sigma = 1$ during training and update σ at the end of training if an uncertainty estimate is desired. For the gradient update, we have empirically observed that applying the Adam optimizer Kingma and Ba (2014) provides the best results. Alternating between EM updates (which jump nonlocally around parameter space) and a gradient update violates the assumptions in the momentum update of Adam. Nonetheless we obtain good results using Adam in our numerical experiments and postpone design of a more careful gradient update to Equation 19 to future work.

4. Hierarchical partition of unity

We now provide an extension of the previous section to construct a hierarchy of models which are amenable to a multilevel training strategy. For ease of exposition we present a two-level method but the approach extends to arbitrary numbers of refinements. We denote the number of coarse partitions N_c, the number of fine refinements per coarse partition N_f, and will obtain a total of $N = N_f N_c$ partitions across the two-level scheme.

We consider a coarse partition $Z_1 \sim Cat\,(\pi(\boldsymbol{x}))$, and now introduce a fine partition Z_2 defined conditionally via

$$p(Z_1 = i) = \pi_i(\boldsymbol{x}, \theta) \tag{20}$$

$$p(Z_2 = j|Z_1 = i) = \pi_{ij}(\boldsymbol{x}, \theta_i) \tag{21}$$

$$p(Z_1 = i, Z_2 = j) = p(Z_1 = i)p(Z_2 = j|Z_1 = i). \tag{22}$$

In total, $N_c + 1$ networks are used to parameterize Z_1 and Z_2: one network on the coarse level, and one network for each of N_c coarse partitions. Each network's architecture consists of a ResNet composed with a softmax activation. Note that the parameterization of the fine level neural networks is denoted by θ_i, on the i^{th} coarse partition. When required, we denote the set of all fine level parameters $\Theta = \{\theta_i\}_{i=1}^{N_c}$.

We construct both a coarse mixture of experts

$$p(Y_1(\boldsymbol{x}) = y) = \sum_{i=1}^{N_c} p\left(Y_1(\boldsymbol{x}) = y | Z_1(\boldsymbol{x}) = i\right) p(Z_1(\boldsymbol{x}) = i) \tag{23}$$

$$= \sum_{i=1}^{N_c} p(\mathcal{E}_i = y) p(Z_1(\boldsymbol{x}) = i) \tag{24}$$

$$p(\mathcal{E}_i = y) = p\left(Y_1(\boldsymbol{x}) = y | Z_1(\boldsymbol{x}) = i\right) = \mathcal{N}(y; \mu_i(\boldsymbol{x}), \sigma_i^2) \tag{25}$$

$$\mu_i(\boldsymbol{x}) = \mathbf{c}_i^\intercal \boldsymbol{\Phi}(\boldsymbol{x}), \tag{26}$$

and a fine mixture of experts

$$p(Y_2(\boldsymbol{x}) = y) = \sum_{i=1}^{N_c} p(\mathcal{E}_{ij} = y) p(Z_1(\boldsymbol{x}) = i, Z_2(\boldsymbol{x}) = j) \tag{27}$$

$$p(\mathcal{E}_{ij} = y) = p\left(Y_2(\boldsymbol{x}) = y | Z_1(\boldsymbol{x}) = i, Z_2(\boldsymbol{x}) = j\right) = \mathcal{N}(y; \mu_{ij}(\boldsymbol{x}), \sigma_{ij}^2) \tag{28}$$

$$\mu_{ij}(\boldsymbol{x}) = \mathbf{c}_{ij}^\intercal \boldsymbol{\Phi}(\boldsymbol{x}), \tag{29}$$

Through marginalization of the fine scale model, we obtain an alternative expression for the coarse scale probabilities in terms of the fine scale probabilities.

$$p(Y_2 = y | Z_1 = i) = \mathbb{E}_{Z_2 \sim \pi_{ij}}[p(Y_2 = y | Z_1 = i, Z_2 = j)] \tag{30}$$

$$= \sum_j p(Y_2 = y | Z_1 = i, Z_2 = j) p(Z_2 = j | Z_1 = i) \tag{31}$$

$$= \sum_j \pi_{ij}(\boldsymbol{x}, \theta_i) \mathcal{N}(y_d; \mu_{ij}(\boldsymbol{x}_d), \sigma_{ij}^2) \tag{32}$$

Following the same expectation maximization procedure from the previous section, we obtain the following three estimators for posterior distributions

$$w_{id} := p(Z_1 = i | Y_1 = y_d) = \frac{\pi_i(\boldsymbol{x}_d, \theta) \mathcal{N}(y_d; \mu_i(\boldsymbol{x}_d), \sigma_i^2)}{\sum_I \pi_I(\boldsymbol{x}_d) \mathcal{N}(y_d; \mu_I(\boldsymbol{x}_d), \sigma_I^2)}, \tag{33}$$

$$w_{ijd} := p(Z_1 = i, Z_2 = j | Y_2 = y_d) = \frac{\pi_i(\boldsymbol{x}_d, \theta) \pi_{ij}(\boldsymbol{x}, \theta_i) \mathcal{N}(y_d; \mu_{ij}(\boldsymbol{x}_d), \sigma_{ij}^2)}{\sum_{I,J} \pi_I(\boldsymbol{x}_d) \pi_{IJ}(\boldsymbol{x}, \theta_I) \mathcal{N}(y_d; \mu_{IJ}(\boldsymbol{x}_d), \sigma_{IJ}^2)}, \tag{34}$$

$$\hat{w}_{id} := p(Z_1 = i | Y_2 = y_d) = \frac{\sum_j \pi_i(\boldsymbol{x}_d) \pi_{ij}(\boldsymbol{x}, \theta_i) \mathcal{N}(y_d; \mu_{ij}(\boldsymbol{x}_d), \sigma_{ij}^2)}{\sum_{I,j} \pi_I(\boldsymbol{x}_d) \pi_{Ij}(\boldsymbol{x}, \theta_I) \mathcal{N}(y_d; \mu_{Ij}(\boldsymbol{x}_d), \sigma_{Ij}^2)}. \tag{35}$$

The estimator $\hat{w}_{id}$ is constructed by marginalizing the fine level estimator on to the coarse level. This will be used in the construction of the multilevel scheme.

Following the same procedure from the previous section, we obtain two weighted least squares problems for the coarse and fine scale expert models.

$$\sum_{d=1}^{N_d} w_{id} \Phi_\alpha(\boldsymbol{x_d}) \Phi_\beta(\boldsymbol{x_d}) c_{i,\beta} = \sum_{d=1}^{N_d} w_{id} \Phi_\alpha(\boldsymbol{x_d}) y_d \tag{36}$$

$$\sum_{d=1}^{N_d} w_{ijd} \Phi_\alpha(\boldsymbol{x_d}) \Phi_\beta(\boldsymbol{x_d}) c_{ij,\beta} = \sum_{d=1}^{N_d} w_{ijd} \Phi_\alpha(\boldsymbol{x_d}) y_d \tag{37}$$

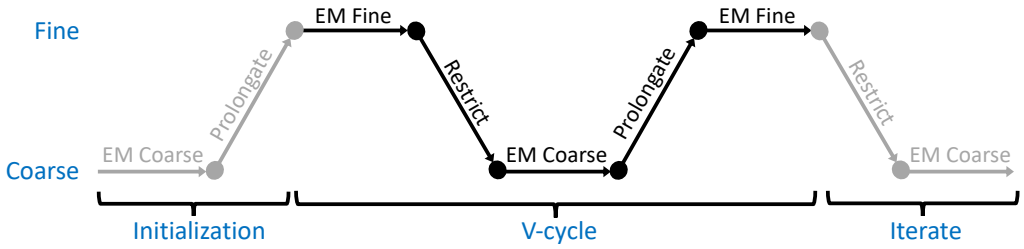

Figure 1: A two-level "V-cycle" for training the hierarchical POU networks. For details in pseudo-code see Algorithm 4.1, where the figure depicts the case $S_{pre} = 0$, $S_{coarse} = 1$ and $S_{post} = 1$. The horizontal axis subdivides the algorithm into stages mimicking a V-cycle AMG scheme; EM-Fine, Restrict, EM-Coarse, and Prolongate. The algorithmic kernel comprised of these stages, highlighted with black, defines the V-cycle. The light grey stages are representative of an initialization phase, and an iteration of repeated V-cycles.

Finally, analogous to Equation 19, we define the following three losses

$$\mathcal{L}_c(\theta; \mathcal{D}) = \sum_{i,d} w_{id} \log \pi_i(\boldsymbol{x_d}; \theta), \tag{38}$$

$$\mathcal{L}_f(\theta_i; \mathcal{D}) = \sum_{i,j,d} w_{ijd} \log \left(\pi_i(\boldsymbol{x_d}; \theta)\pi_{ij}(\boldsymbol{x_d}; \theta_i)\right), \tag{39}$$

$$\mathcal{L}_{f2c}(\theta; \mathcal{D}) = \sum_{i,d} \hat{w}_{id} \log \pi_i(\boldsymbol{x_d}; \theta). \tag{40}$$

The first and second losses align the coarse and fine partitions with their corresponding posterior estimators. The third loss aligns the coarse partitions with the marginalized fine scale model, allowing the coarse partition to access fine scale information.

4.1. Multilevel Training of Hierarchical POUs

Our approach is motivated by multilevel methods used to solve PDEs Brandt (1977); Trottenberg et al. (2000); Briggs et al. (2000), optimization problems Nash (2000); Gaedke-Merzhäuser et al. (2020) and, recently, to train neural networks Gunther et al. (2020); Kirby et al. (2020); Moon and Cyr (2021); Gaedke-Merzhäuser et al. (2020). Figure 1 is a schematic description of Algorithm 4.1 for a two-level V-cycle. In the algorithm we mimick the Relaxation, Restriction, Correction and Prolongation structure of a typical multigrid scheme.

The intuition behind the multilevel scheme is that we desire a hierarchy of partitions which simultaneously provide good approximations at both coarse and fine length scales, while maintaining that the coarse approximation should approximate the fine approximation. This allows steps of gradient descent to evolve at multiple rates, with partitions on each scale evolved using separate instances of Adam with separate momenta.

In nonlinear multilevel methods for solving PDEs, the prolongation step consists of a data remap of the coarse correction to produce a correction on the fine level. However, due to the marginalization formula in Equation 30, no remap is necessary as the coarsened probability distribution may be

Function `TwoLevel` (θ *: coarse weights,* Θ *: fine weights,* $S_{pre}, S_{post}, S_{coarse}$) **:**

```
// Pre-relaxation
for i = 1...S_pre do
    w_ijd ← FinePosterior(θ, Θ, c_ij)  // See Posterior Definition in Eq. 34
    c_ij ← WeightedLeastSquares(w_ijd)  // See Eq. 37
    Θ ← Gradient-Descent(w_ijd, θ, Θ, L_f)  // See Eq. 38
end
// Restrict
ŵ_id ← Σ_j w_ijd  // See Eq. 35
θ ← Gradient-Descent(ŵ_id, θ, L_f2c)  // See Eq. 38
// Coarse correction
for i = 1...S_coarse do
    w_id ← CoarsePosterior(θ, c_i)  // See Posterior Definition in Eq. 33
    c_i ← WeightedLeastSquares(w_id)  // See Eq. 36
    θ ← Gradient-Descent(w_id, θ, L_c)  // See Eq. 38
end
// Prolongation:  Use the most recent value of θ on the fine level
// Post-relaxation
for i = 1...S_post do
    w_ijd ← FinePosterior(θ, Θ, c_ij)  // See Posterior Definition in Eq. 34
    c_ij ← WeightedLeastSquares(w_ijd)  // See Eq. 37
    Θ ← Gradient-Descent(w_ijd, θ, Θ, L_f)  // See Eq. 38
end
```

Algorithm 1: Mimicking the steps of a V-cycle multigrid scheme, the two level training consists of: solving on the fine scale (pre-relaxation); using the fine scale prediction to update the coarse scale (restrict); solving on the coarse scale (coarse correction); and then optionally solving again on the fine scale (post-relaxation) using the θ from the coarse level (prolongation).

directly evaluated from the fine one. Polynomial coefficients are fully decoupled on each partition and are computed using a GPU-accelerated weighted least squares solve.

To provide further motivation for the effectiveness of alternating between coarse and fine scales, the hierarchical architecture admits the following decomposition of the ELBO loss (Eq. 39).

$$\mathcal{L}_f(\theta_i; \mathcal{D}) = \sum_{i,j,d} w_{ijd} \log\left(\pi_i(\boldsymbol{x_d}; \theta)\pi_{ij}(\boldsymbol{x_d}; \theta_i)\right) \tag{41}$$

$$= \sum_{i,j,d} w_{ijd} \log\left(\pi_i(\boldsymbol{x_d}; \theta)\right) + \sum_{i,j,d} w_{ijd} \log\left(\pi_{ij}(\boldsymbol{x_d}; \theta_i)\right) \tag{42}$$

$$= \sum_{id} \underbrace{\left(\sum_{j} w_{ijd}\right)}_{\hat{w}_{id}} \log\left(\pi_i(\boldsymbol{x_d}; \theta)\right) + \sum_{i,j,d} w_{ijd} \log\left(\pi_{ij}(\boldsymbol{x_d}; \theta_i)\right) \tag{43}$$

$$= \mathcal{L}_{f2c}(\theta; \mathcal{D}) + \sum_{i,j,d} w_{ijd} \log\left(\pi_{ij}(\boldsymbol{x_d}; \theta_i)\right). \tag{44}$$

Therefore, the ELBO decomposes into the marginalized ELBO plus a correction of the fine scales.

5. Numerical experiments

We gather results in one and two dimensions to establish the performance of the approach. Data and code to reproduce experiments may be found at https://anonymous.4open.science/r/pounets-4084/hierarchicalPOU.ipynb. All parameters necessary to reproduce experiments may be found in Appendix A.

5.1. Adaptive approximation of smooth functions

We approximate the function $y(x) = \exp\left[-\frac{1}{2}\left(\frac{x-0.5}{0.05}\right)\right]^2$ using three coarse partitions each refined with two fine scale partitions ($N = 6$). In Figure 2, we observe that the partitions evolve from randomly initialized into a disjoint partition of space, providing localized polynomial approximation refined about the Gaussian peak.

We next demonstrate the impact of the multilevel strategy by approximating the function

$$y(x) = \sin 2\pi x + \exp\left[-\frac{1}{2}\left(\frac{x-0.5}{0.05}\right)\right]^2 \tag{45}$$

using $N_c = 15$ coarse partitions with $N_f = 3$ refinements ($N = 45$). We compare to an otherwise identical single level POUnet, consisting of either $N = 15$ or $N = 45$ partitions. Therefore, we obtain a comparison of both the fine and coarse scales to a corresponding single level architecture. In Figure 3, we observe an order of magnitude increase in accuracy for the fine scale approximation. One may, as a post-processing step, fit higher order polynomials while keeping the partitions fixed to achieve spectral convergence.

5.2. Adaptive approximation of piecewise smooth functions

Next we approximate the piecewise quadratic function

$$y(x;p) = \left(2\left|px - \left\lfloor px + \frac{1}{2}\right\rfloor\right| - 1\right)^2. \tag{46}$$

This function was considered in Lee et al. (2021), where it was demonstrated that deterministic POU-nets are able to provide approximation with error under 1% as p is increased, while for a dense network accuracy degrades for large p. In Figure 4, we are able to achieve accuracy to machine precision with a single set of parameters.

5.3. Two-dimensional reduced regularity data

In Figure 5 we define

$$s(x,y) = y - (x - \frac{1}{2})^2$$

and regress the two functions

$$f_1(x,y) = \sin 2\pi s(x,y) \tag{47}$$

and

$$f_2(x,y) = \left(2\left|4s(x,y) - \left\lfloor 4s(x,y) + \frac{1}{2}\right\rfloor\right| - 1\right)^2, \tag{48}$$

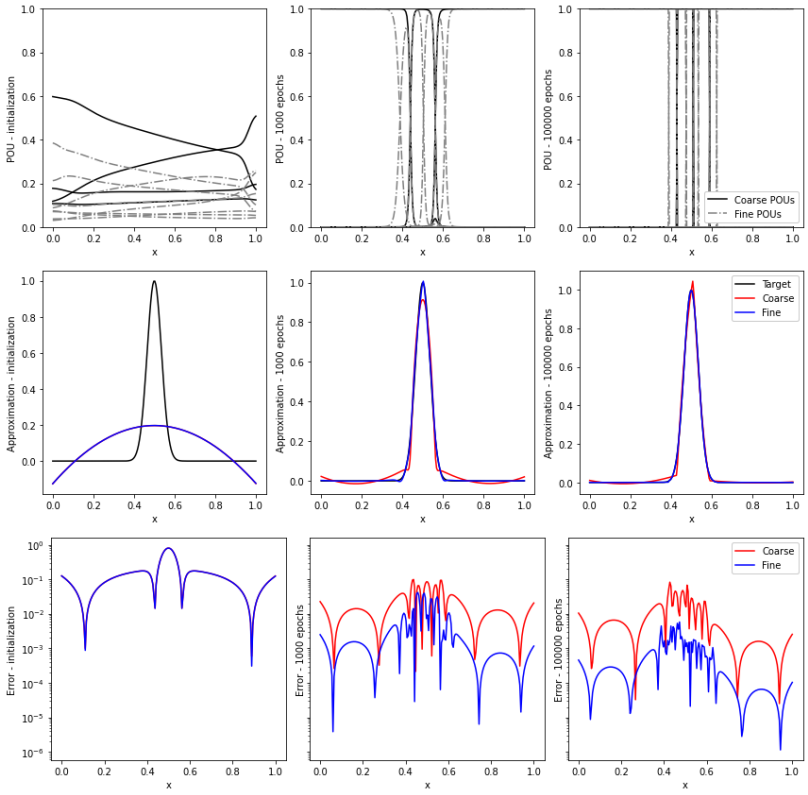

Figure 2: Evolution of partitions and approximations from initialization *(left column)* to 1000 steps *(center column)* and 100000 steps *(right column)*. Initially random partitions rapidly assemble into disjoint compactly supported subdomains *(top row)*. Clustering near features provides localized approximation to data *(center row)* and accurate pointwise error *(bottom row)*.

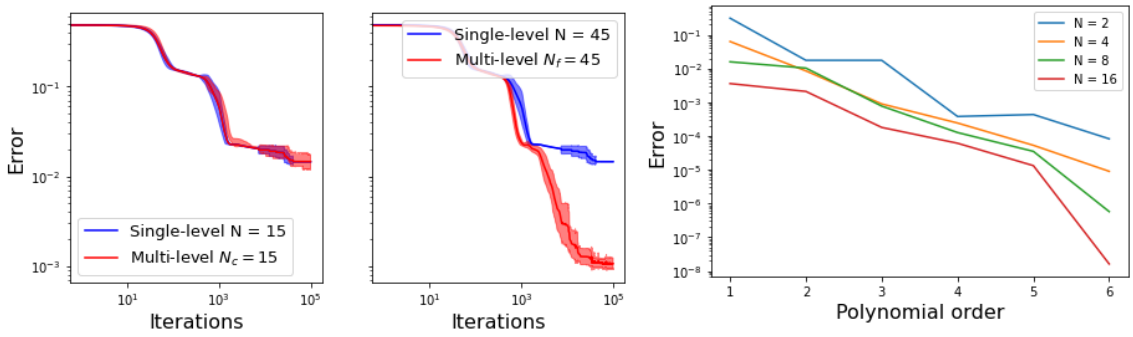

Figure 3: Convergence comparison of $N = 15$ *(left)* or $N = 45$ *(center)* single stage partitions *(blue)* to two-stage $(N_c = 15, N_f = 3)$ scheme with same total partitions $N = 45$ *(red)* and otherwise identical parameters. An order of magnitude increase in accuracy is realized for multistage scheme. When sequentially higher-order polynomials are used on each partition *(right)* exponential convergence is observed. Log-normal statistics over five runs are presented.

where $f_1 \in C_\infty$ and $f_2 \in C_0$. We use these to study convergence in two dimensions in comparison to traditional finite element spaces; namely a p^{th}-order discontinuous Galerkin (DG) space on a Cartesian grid consisting of N total elements. Following Wendland (Theorem 3.2) Wendland (2004), *local polynomial reproductions* consisting of compactly supported basis functions which reproduce m^{th}-order polynomials admit the optimal scaling

$$||f - f_{lpr}||_\infty(\Omega) \leq CN^{-\frac{m+1}{d}}|f|_{C^{m+1}},$$

where $|\cdot|_{C^m}$ denotes the H^m seminorm. This scaling on the dimension d is based upon a packing argument requiring $\frac{|\Omega|}{N}\ h^d$ to fill space. This provides one notion of the "curse-of-dimensionality", where the algebraic convergence rate with respect to the total number of partitions scales inversely with the latent dimension. The comparison in Figure 5 demonstrates algebraic convergence for the POU approximation at a faster rate than a corresponding DG space with the same number of elements/partitions and equal polynomial order. As shown in the figure, this is because the learnt partitions may take advantage of underlying low-dimensional structure - in this case the quasi-1D dependence on $s(x, y)$.

5.4. Application: semiconductor device modeling

Finally, we consider an application in which one regresses a map governing the electrical current as a function of voltage drop for a transistor device. Data-driven models for semiconductor devices have attracted attention as a means of embedding richer representations of physics when first principles models are either inaccurate or computationally intractable, and the regressed "IV-curve" response is embedded as a component into a circuit simulator and used to perform system scale design. This data set is subtly challenging for non-parametric regression, as the response varies from ideal diode behavior with exponential scaling at small voltage drops, to resistive behavior with linear scaling at large voltage drops. Accurate prediction of device performance hinges on accurate prediction over ten orders of magnitude variation in the current. To handle this broad range

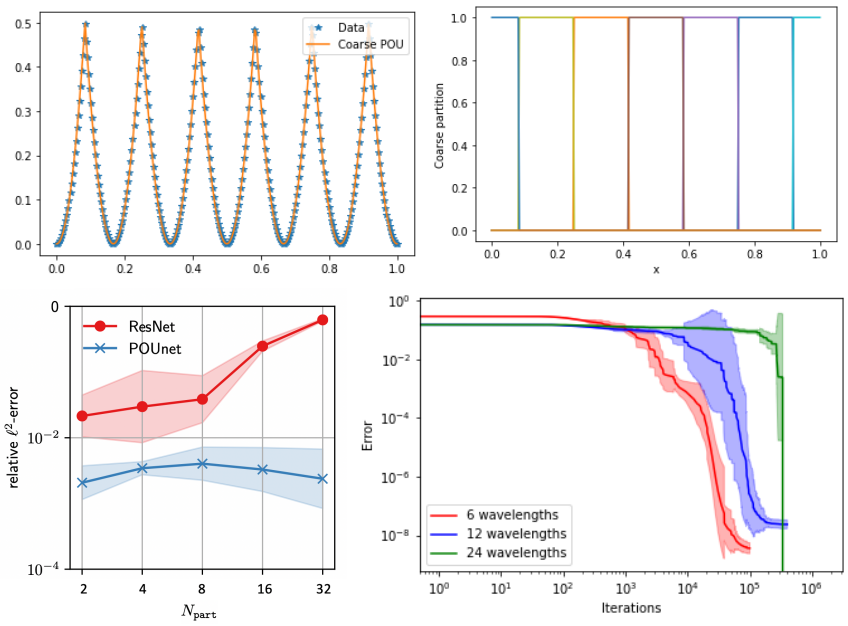

Figure 4: Approximation of piecewise quadratic function (*top left*) and resulting piecewise disjoint partitions (*top right*). While Lee et al. (2021) demonstrated poor performance for a ResNet and sustained $< 1\%$ accuracy for deterministic POU-nets as the number of piecewise quadratic regions was increased (*bottom left*), the proposed hierarchical approach is able to reliably achieve machine precision for a single set of parameters (*bottom right*). Statistics are presented as running minimum over ten runs assuming log-normal distribution. For large numbers of peaks, after many iterations the solution diverges. While this can be avoided by choosing a smaller learning rate, we present all results with a single set of hyperparameters.

of scales, previous researchers (Aadithya et al. (2020); Gao et al. (2020)) manually partitioned the response into an exponential and linear regime and introduced scaling of data within each to achieve a good approximation. While effective, this human-in-the-loop approach precludes automation, and we demonstrate superior accuracy may be obtained without human intervention. All results are calculated using a log scaling of the current and a transform of the voltages to map onto a unit interval. For this example only, we perform a grid search hyperparameter tuning to obtain the most compact architecture capable of matching both the logarithmic and linear current response.

In Figure 6 we demonstrate predictions of the linear- and log-scale current response using either piecewise quadratic or quartic polynomials. In Table 1 we compare the performance of the multilevel scheme against an otherwise identical single level POU-Net. In general, the multistage training provides a surrogate with up to two orders of magnitude better accuracy, and often will provide a coarse scale prediction that outperforms the single level model despite using half the number of partitions. Finally, in Figure 7 we consider a three-terminal bipolar junction transistor device for which the current is a function of two voltage drops, posing a two-dimensional regression problem.

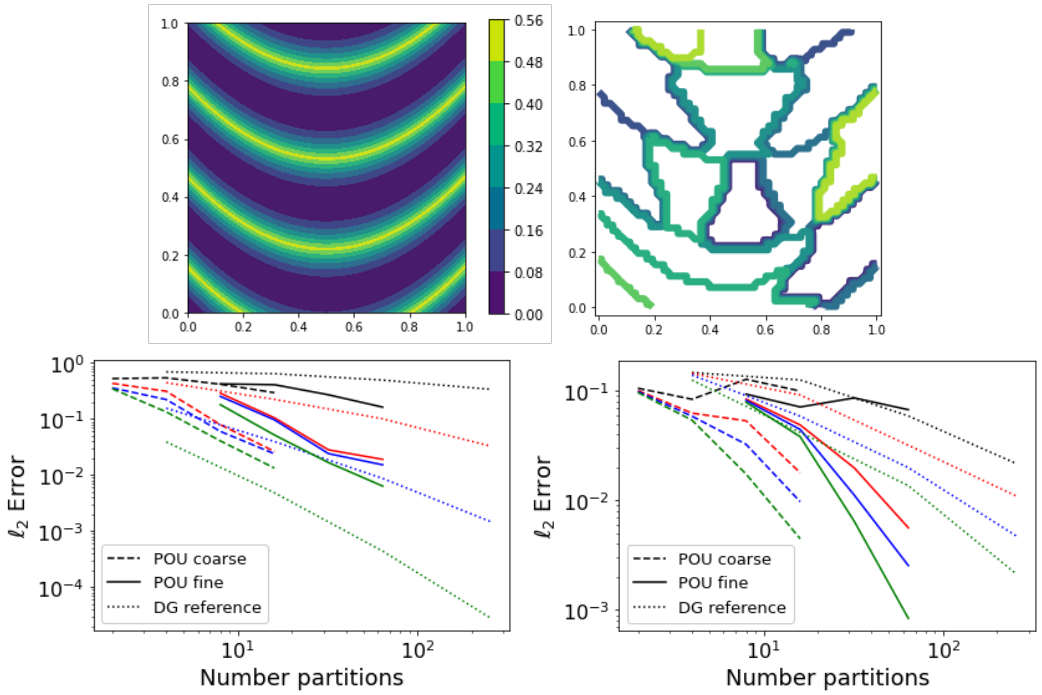

Figure 5: **2D convergence study for reduced regularity data:** We consider approximation of C_0 and C_∞ functions in two dimensions. *Top left:* C_0 function approximated. Maximum value (yellow contours) coincides with discontinuity in first derivative. *Top right:* Partitions associated with C_0 function, with boundaries visualized by plotting contours associated with argmax of POU. Partition boundaries align with discontinuities. *Bottom:* Algebraic convergence with respect to number of partitions for C_∞ data *(left)* and C_0 data *(right)* with comparison to corresponding discontinuous polynomial (DG) space. Black, red, blue and green lines correspond to first, second, third and fourth order polynomial spaces respectively. For smooth data, we observe a nominal improvement compared to a traditional space, with algebraic convergence rates matching or exceeding DG space. For C_0 data the DG approximation is hindered by reduced regularity, while adaptivity of the current approach allows realization of an order of magnitude reduction in partitions required to reach comparable accuracy.

		ℓ_∞			ℓ_2		
N	Porder	mean - stdev	mean	mean + stdev	mean - stdev	mean	mean + stdev
8	2	3.1E-3	3.8E-3	4.7E-3	1.1E-3	1.3E-3	1.5E-3
8	4	6.2-5	2.0E-4	6.4E-4	2.3E-5	7.5E-5	2.4E-4
(4,2) coarse	2	6.6E-7	1.8E-4	3.3E-4	4.7E-5	7.7E-5	1.2E-4
(4,2) coarse	4	9.6E-5	2.5E-4	6.5E-4	2.4E-5	7.3E-5	2.3E-4
(4,2) fine	2	1.1E-3	1.4E-3	1.8E-3	3.9E-4	5.1E-4	6.4E-4
(4,2) fine	4	2.0E-6	6.6E-6	2.2E-5	6.6E-7	2.3E-6	8.3E-6

Table 1: Comparison of single level POU against multilevel POU for semiconductor problem. Fine scale prediction of multilevel POU is two orders of magnitude more accurate with a lower standard deviation, while the coarse multilevel prediction provides a more accurate prediction than the single training while using half as many partitions. Statistics are computed over five runs assuming a log normal distribution of error.

This is the only test we performed which required moving to a larger network (presumably due to the higher dimension).

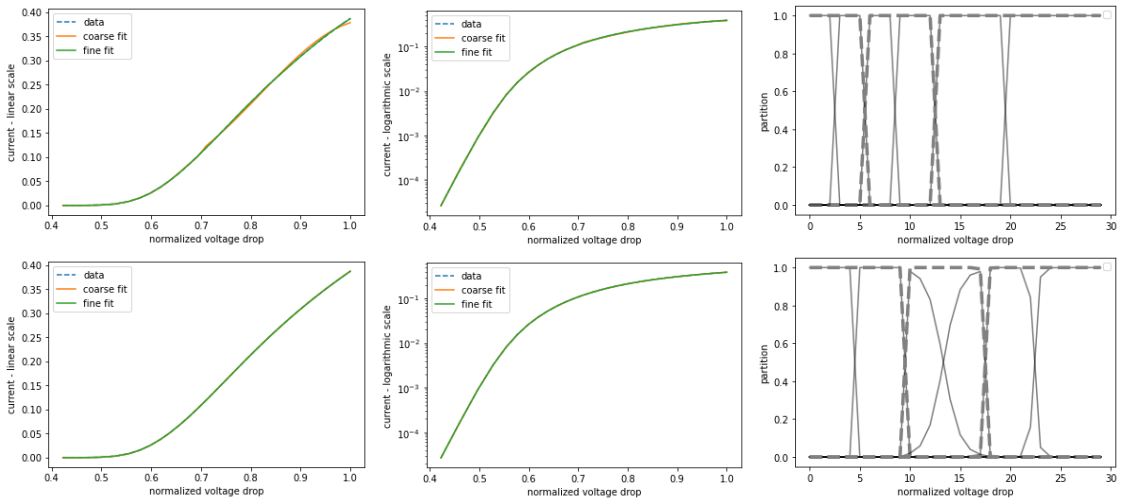

Figure 6: Prediction of IV-curve for transistor device using 4 coarse partitions with 2 refinements, using either quadratic (*top*) or quartic (*bottom*) polynomials. To serve as an accurate surrogate for circuit simulation, the approximation must reproduce a response on linear (*left*) and logarithmic (*center*) scales, posing a challenging regression problem spanning five orders of magnitude. Interestingly, training using different ordering of polynomials provides qualitatively different partitions of space (*right*).

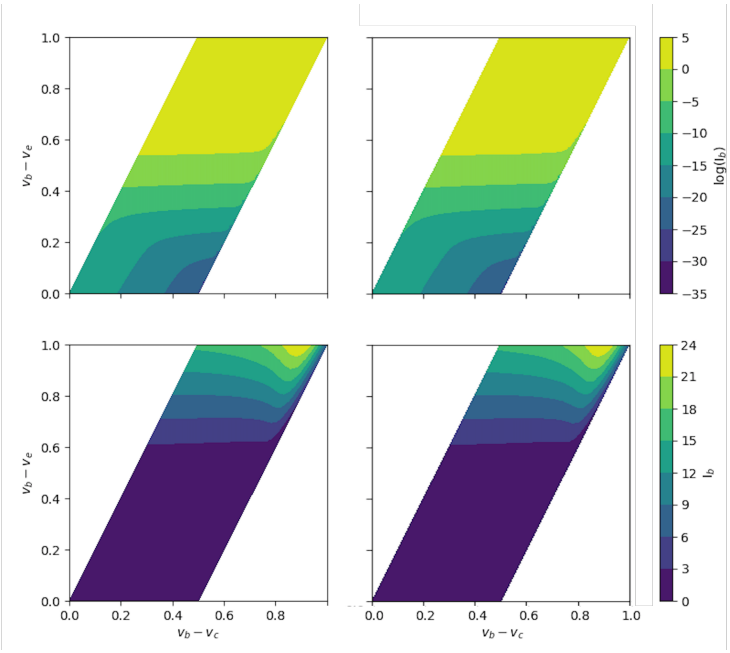

Figure 7: Prediction of IV-response for bipolar junction transistor device, requiring prediction across two-dimensional input space. *Left:* Training data, *Right:* Prediction, *Top:* Log-scale current, *Bottom:* Linear-scale current.

6. Conclusion

We have presented a new probabilistic generalization of POU-nets which may be applied in a hierarchical manner and trained with a V-cycle inspired multilevel scheme. Benchmarks show that this provides a marked improvement compared to either deep neural network or deterministic POU-nets for a range of benchmarks.

While for simplicity the current work has focused on a two level scheme, in principle the approach could be extended to an arbitrary number of levels, similar to traditional multigrid methods. We pursue this in future work, anticipating improvements both in quality of solution as well as exposing exploitable parallelism. As the calculation decouples into a large number of small least squares problems and loss functions, it is a candidate for both MPI-style domain decomposition and GPU-accelerated solution of the linear systems.

An additional further direction of research is to generalize this strategy to solve partial differential equations (PDEs) in high dimensions. While initial works have shown promise using DNNs to solve PDEs, realizing the convergence guarantees typical of finite element methods remains elusive. In future work we apply the ideas introduced here to the solution of both physics-informed neural networks Raissi et al. (2019) and structure-preserving architectures Trask et al. (2022).

Acknowledgments

N. Trask acknowledges funding under the Collaboratory on Mathematics and Physics-Informed Learning Machines for Multiscale and Multiphysics Problems (PhILMs) project funded by DOE Office of Science (Grant number DE-SC001924). N. Trask and E. Cyr acknowledge support from the DOE Early Career program. A. Henriksen acknowledges support from the Sandia data science postdoctoral fellowship. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DENA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. SAND number: SAND2022-4027 O

References

K Aadithya, Paul Kuberry, B Paskaleva, P Bochev, K Leeson, Alan Mar, Ting Mei, and E Keiter. Development, demonstration, and validation of data-driven compact diode models for circuit simulation and analysis. *arXiv preprint arXiv:2001.01699*, 2020.

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

Christan Beck, Arnulf Jentzen, and Benno Kuckuck. Full error analysis for the training of deep neural networks. *arXiv preprint arXiv:1910.00121*, 2019.

Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

Achi Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977.

William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

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 *Mathematical and Scientific Machine Learning*, pages 512–536. PMLR, 2020.

Lisa Gaedke-Merzhäuser, Alena Kopaničáková, and Rolf Krause. Multilevel minimization for deep residual networks. *arXiv preprint arXiv:2004.06196*, 2020.

Xujiao Gao, Andy Huang, Nathaniel Trask, and Shahed Reza. Physics-informed graph neural network for circuit compact model development. In *2020 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pages 359–362. IEEE, 2020.

Amara Graps. An introduction to wavelets. *IEEE computational science and engineering*, 2(2): 50–61, 1995.

Leslie Greengard and William D Gropp. A parallel version of the fast multipole method. *Computers & Mathematics with Applications*, 20(7):63–71, 1990.

Stefanie Gunther, Lars Ruthotto, Jacob B Schroder, Eric C Cyr, and Nicolas R Gauger. Layer-parallel training of deep residual neural networks. *SIAM Journal on Mathematics of Data Science*, 2(1):1–23, 2020.

Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. Relu deep neural networks and linear finite elements. *arXiv preprint arXiv:1807.03973*, 2018.

Bayram Ali Ibrahimoglu. Lebesgue functions and lebesgue constants in polynomial interpolation. *Journal of Inequalities and Applications*, 2016(1):1–15, 2016.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Andrew Kirby, Siddharth Samsi, Michael Jones, Albert Reuther, Jeremy Kepner, and Vijay Gadepally. Layer-parallel training with gpu concurrency of deep residual neural networks via nonlinear multigrid. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, sep 2020.

Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

Kookjin Lee, Nathaniel A Trask, Ravi G Patel, Mamikon A Gulian, and Eric C Cyr. Partition of unity networks: deep hp-approximation. *arXiv preprint arXiv:2101.11256*, 2021.

Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014.

Euhyun Moon and Eric C Cyr. Parallel training of gru networks with a multi-grid solver for long sequences. In *International Conference on Learning Representations*, 2021.

Stephen G Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14(1-2):99–116, 2000.

William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press, 2010.

Joost AA Opschoor, Philipp C Petersen, and Christoph Schwab. Deep relu networks and high-order finite element methods. *Analysis and Applications*, 18(05):715–770, 2020.

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

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Christoph Schwab and Jakob Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq. *Analysis and Applications*, 17(01): 19–55, 2019.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Nathaniel Trask, Andy Huang, and Xiaozhe Hu. Enforcing exact physics in scientific machine learning: a data-driven exterior calculus on graphs. *Journal of Computational Physics*, page 110969, 2022.

Rohit K Tripathy and Ilias Bilionis. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of computational physics*, 375:565–588, 2018.

Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Elsevier, 2000.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021b.

Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

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

Dmitry Yarotsky. Optimal approximation of continuous functions by very deep relu networks. In *Conference on learning theory*, pages 639–649. PMLR, 2018.

Appendix A. Hyperparameters

For all results, the partition of unity architectures are initialized using the Box initialization outlined in Cyr et al. (2020).

For all one dimensional studies, we take for POU architectures a multilayer perceptron of width ten and depth two with tanh activation composed with a softmax. For the final two-dimensional exemplar, we take an otherwise identical slightly larger POU network of width twenty and depth two. All training is performed with a two level scheme, with a timestep of size 0.5e-3 for the Adam optimizer. While better performance may be obtained for some problems by selecting a smaller learning rate, we chose to perform all tests using a single set of parameters to demonstrate accuracy is not do to hyperparameter tuning.

For the piecewise quadratic benchmark, we choose number of total fine partitions to be twice the number of piecewise polynomial regions.

Over the course of peer-review, demonstration code for one of the examples may be found at https://anonymous.4open.science/r/ pounets-4084/hierarchicalPOU.ipynb. Due to memory constraints on the anonymous github we can only host a single jupyter notebook demoing the implementation, but the accepted article will include scripts running each experiment as well as all datasets used.

Appendix B. Effect of noise model for noisy and non-noisy data

In this Section, we consider approximation of the 1D function

$$y(x) = \sin 2\pi x + C(x)\epsilon,$$

$$\epsilon \sim \mathcal{N}(0, 1),$$

considering either no noise ($C(x) = 0$), or with the heteroskedastic noise

$$C(x) = \begin{cases} 0, & \text{if } x \leq \frac{1}{2} \\ 0.1, & \text{if } \frac{1}{2} \geq x \leq \frac{3}{4} \\ 0.5, & \text{if } \frac{3}{4} \geq x \end{cases}$$

For this study, we consider 500 uniformly sampled data points, $N_c = 8$ coarse partitions with $N_f = 2$ refinements or $N = 16$ total partitions and a quadratic polynomial space. The learning rate is kept at the $5e-3$ value used throughout the paper. Figure 9 demonstrates the performance between either using Equation 18 to update the noise model at each step of training or keeping it fixed to one. Figure 8 provides statistics describing the evolution of the error during training, demonstrating that the trends are consistent independent of initialization.

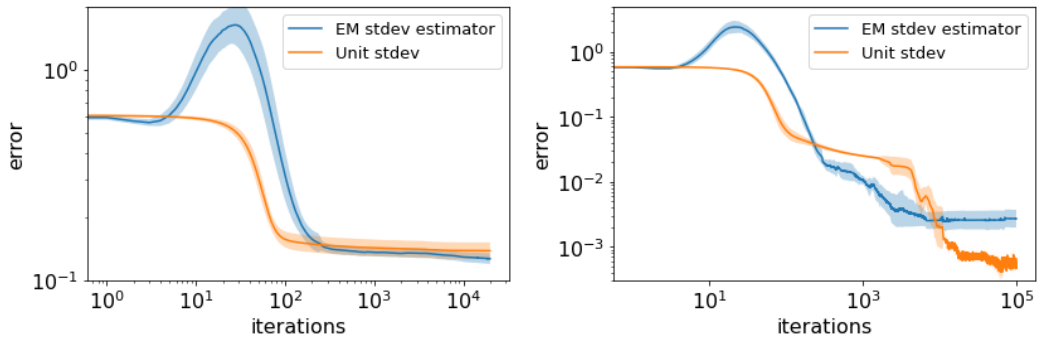

Figure 8: Mean and two standard deviations over 10 experiments demonstrate results in Figure 9 are repeatable and insensitive to random initialization. For noisy data *(left)*, both updates to noise model provide comparable accuracy. For clean data *(right)*, keeping $\sigma = 1$ fixed during training consistently provides an order of magnitude increase in accuracy.

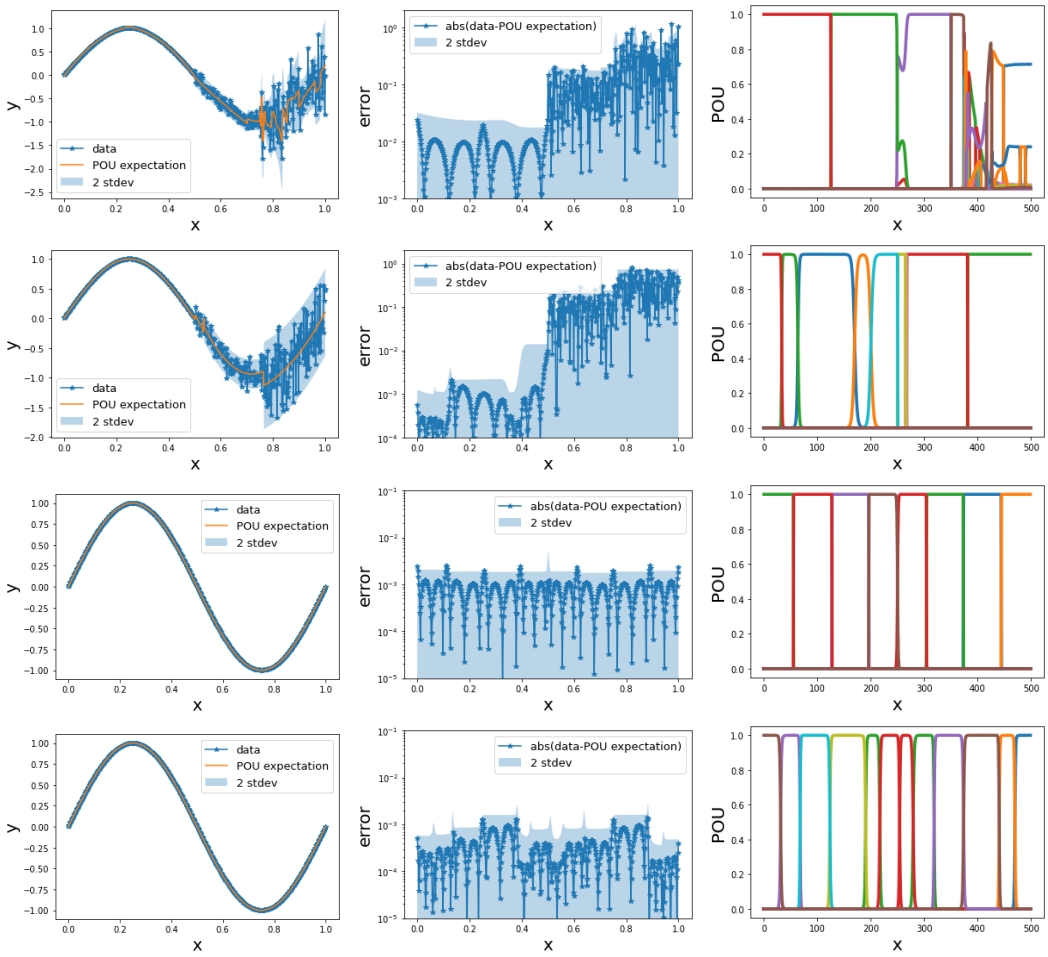

Figure 9: **Noise model fitting study:** We compare fitting 1D data with heterogeneous noise *(rows 1+2)* and without noise *(rows 3+4)*, updating noise at each iteration using either the EM estimator in Equation 18 *(rows 1+3)* or fixing $\sigma = 1$ during training and updating the noise model after training is completed *(rows 2+4)*. For all, the analytic expression variance (Equation 9) provides and accurate error estimator *(column 2)*. For noisy data, applying the EM estimator results in a tight fit to noise but with noisy partitions which provide poor approximation properties, while the second strategy provides a comparably good representation of noise while maintaining improved accuracy. For clean data, the EM estimator provides a uniform bound on the error *(row 3, column 2)* but with an overall larger mean error of 2.46×10^{-3} vs. 4.03×10^{-4}.