Kolmogorov n-Widths for Multitask Physics-Informed Machine Learning (PIML) Methods: Towards Robust Metrics

Michael Penwarden[1,2], Houman Owhadi[3], Robert M. Kirby[1,2]

[1] *Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, USA.*

[2] *Kahlert School of Computing, University of Utah, Salt Lake City, UT 84112, USA.*

[3] *Department of Computing and Mathematical Sciences, Caltech, Pasadena, CA 91125, USA*

Abstract

Physics-informed machine learning (PIML) as a means of solving partial differential equations (PDE) has garnered much attention in the Computational Science and Engineering (CS&E) world. This topic encompasses a broad array of methods and models aimed at solving a single or a collection of PDE problems, called multitask learning. PIML is characterized by the incorporation of physical laws into the training process of machine learning models in lieu of large data when solving PDE problems. Despite the overall success of this collection of methods, it remains incredibly difficult to analyze, benchmark, and generally compare one approach to another. Using Kolmogorov n-widths as a measure of effectiveness of approximating functions, we judiciously apply this metric in the comparison of various multitask PIML architectures. We compute lower accuracy bounds and analyze the model's learned basis functions on various PDE problems. This is the first objective metric for comparing multitask PIML architectures and helps remove uncertainty in model validation from selective sampling and overfitting. We also identify avenues of improvement for model architectures, such as the choice of activation function, which can drastically affect model generalization to "worst-case" scenarios, which is not observed when reporting task-specific errors. We also incorporate this metric into the optimization process through regularization, which improves the models' generalizability over the multitask PDE problem.

Keywords: Physics-informed Neural Networks (PINNs), DeepONets, Kolmogorov n-width, Multitask Learning

1. Introduction

Physics-informed machine learning (PIML) has emerged as a popular framework for solving partial differential equations (PDEs). PIML has been particularly effective and efficient in solving ill-posed and inverse problems over traditional methods [1]. The most ubiquitously used PIML implementation at present is the physics-informed neural network (PINN)[2]. This approach is often selected due to its flexibility in discretization and has been shown to be successful across a wide class of application domains [3, 4, 5, 6, 7, 8, 9]. Innumerable strategies to enhance PINN training have been proposed such as adaptive sampling [10, 11, 12], adaptive weighting [13, 14], adaptive activation functions [15, 16], additional loss terms [17], domain decomposition [18, 19, 20, 21], metalearning [22, 23, 24], and network architecture modification to obey characteristics [25, 26]. A thorough summary of PINN training challenges and their proposed solutions is provided in [25]. The physics-informed concept has also been incorporated into neural operator learning

---

Email addresses: mpenwarden@sci.utah.edu (Michael Penwarden), owhadi@caltech.edu (Houman Owhadi), kirby@cs.utah.edu (Robert M. Kirby)

in works such as physics-informed DeepONets [27, 28, 29] and physics-informed neural operators (PINO) [30, 31]. Comprehensive surveys of the state of PIML and future directions can be found in [32, 33].

The optimization process of PINNs and physics-informed methods, in general, limits the upper accuracy bound and can cause training challenges due to the interplay between minimizing the PDE residuals and the non-convex optimization of deep neural networks, which induces non-unique solutions [34]. Training challenges in PINNs can happen for various reasons, the most common being poor sampling, unbalanced loss term weights, or poor optimization schemes. Even with a well-tuned PINN, "stiff" PDEs with sharp transitions [35], multi-scale problems [36], or highly nonlinear time-varying PDEs [37] can still pose problems for the standard PINN. Despite the overall success of these methods, many practical issues and avenues of improvement still exist, particularly in the case of forward problems [38, 13]. Many works still handpick problems or tasks in a family of problems that show the most promising results and do not provide a lower accuracy bound or performance over the continuous task family. Therefore, we propose using Kolmogorov n-widths applied to physics-informed methods as a measure with which we can analyze, benchmark, and improve models for multitask problems. Other connections between Kolmogorov n-widths and PINNs have been proposed as a measure of the rate of convergence [25]; however, only approximations using the rate of decay of temporal snapshot singular values are made as opposed to the true *inf sup inf* evaluation performed here.

Our first major contribution is a proposed methodology for computing Kologmroov n-widths as a metric and regularizer for PIML models, which requires tri-optimization. Our second contribution is an experimental study of different multitask PIML architectures for which we can compute benchmarks to compare against. We also provide insights into the ability of different architectures to learn generalizable basis functions through choice in various parameters, such as the activation function. The paper is organized as follows: in Section 2, we first summarize PIML and existing architectures that target multitask PDE problems such as physics-informed DeepONets and multi-head PINNs. In Section 3, we introduce Kolmogorov n-widths and how they can be numerically approximated for PIML models using a novel optimization procedure. Two applications of this metric are proposed: the first as an analysis tool with which we can benchmark different architectures and compute lower accuracy bounds, and the second is a way to add regularization into training to improve the lower accuracy bound and the generalization of the learnable neural network basis functions. In Section 4, we provide numerical results on various multitask problems, study the effect of different network sizes and training iterations, and study the effect of regularization on the learned basis functions. We summarize and conclude our results in Section 5.

2. Background

Physics-informed machine learning (PIML) [1, 32, 33] was introduced as an alternative to traditional PDE solvers such as finite differences, finite elements, finite volumes, spectral methods, etc. Despite significant efforts with traditional approaches, it is still difficult to incorporate noisy data, solve high-dimensional and inverse problems, or overcome great costs attributed to complex mesh generation, all requiring elaborate code to function. Alternatively, PIML represents the modern confluence of two powerful computational modeling paradigms: data-intensive machine learning and model-informed simulation science. Using machine learning as the foundation, this approach benefits from the rich expressivity of deep neural networks (DNNs) as a surrogate for PDE solutions. They also allow for "meshfree" discretization and capitalize on machine learning technologies such as automatic differentiation and stochastic optimization to easily incorporate physical laws with data [39]. Figure 1 illustrates the spectrum of situations PIML can be placed in, ranging from pure physics-informed learning to pure regression. This work focuses solely on the physics-informed paradigm for multitask forward problems since they result in challenging training cases and compete with dataless traditional PDE solvers [40, 34].

2.1. Physics-Informed Neural Networks

Physics-informed neural networks (PINNs), proposed in [2, 41, 42], are the most common paradigm of PIML. The NN component of PINN is generally taken as a feed-forward NN (FF-NN), which is the case
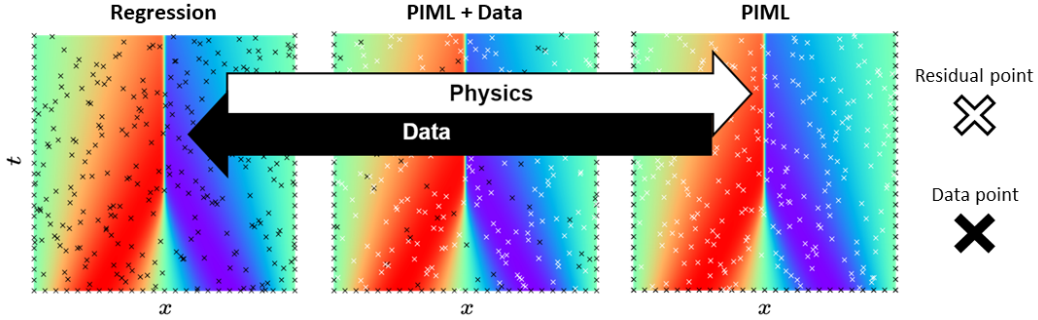
Figure 1: Physics-Informed Machine Learning as a spectrum of data and physics. Residual points are where the PDE residual is evaluated and minimized during optimization.

in this work, but includes and is not limited to convolutional NN (CNN) [43, 44, 45, 46, 47], recurrent NN (RNN) [48, 49, 50], among others. In the original PINNs work, when presented with a PDE specified over a spatiotemporal domain $\Omega \times T$, the solution is computed (i.e., the differential operator is satisfied) as in other mesh-free methods like RBF-FD [51, 52] at a collection of collocation points. To perform this optimization, we re-write a generalized PDE system in residual form as $\mathcal{R}(u) = \mathcal{S} - \frac{\partial}{\partial t} u - \mathcal{F}(u)$ where $\mathcal{S}$ and $\mathcal{F}$ are source functions and nonlinear operators respectively and compute the derivatives with automatic-differentiation (AD). PINN optimization is expressed as follows: given a neural network $u_{\boldsymbol{\theta}}(\mathbf{x}, t)$ with specified activation functions and learnable parameters $\boldsymbol{\theta}$, find $\boldsymbol{\theta}^*$ that minimizes the summed error of the boundary/initial condition and PDE residual term. The PINN optimization problem is explicitly defined as follows:

$$\boldsymbol{\theta}^* \leftarrow \underset{\boldsymbol{\theta}}{\arg\min} \frac{\lambda_{\mathcal{B}}}{N} \sum_{j=1}^{N} |u(\mathbf{x}_j, t_j) - u_{\boldsymbol{\theta}}(\mathbf{x}_j, t_j)|^2 + \frac{\lambda_{\mathcal{R}}}{M} \sum_{j=1}^{M} |\mathcal{R}(u_{\boldsymbol{\theta}}(\mathbf{x}_j, t_j))|^2. \tag{1}$$

Given the optimization scheme in Equation refeq:PINN, we now draw the parallel of neural network architecture to the linear combination of global basis functions. In Figure 2, diagram (A) outlines the framework for a standard PINN with physics-informed training as described in Equation 1. In diagram (B) and future diagrams, physics-informed training is implied but not explicitly shown. In (B), we see that the "body" of the neural network acts as a surrogate for learning global basis functions, where each neuron in the final hidden layer is one basis function, similar to a set of global basis functions in spectral methods. This is a unique and overlooked aspect of PINNs, which are commonly compared to traditional methods that use basis functions with compact support instead of global functions. The final layer of the network can be viewed as the basis coefficients, which are solution-specific.

2.2. Multitask PDE Problems

Multitask learning [53] is when multiple tasks are solved simultaneously while exploiting commonalities across the tasks. This learning paradigm improves generalization by incorporating domain information contained in *related* tasks as an inductive bias. In the context of PDEs, this can arise in many forms, such as solving a task family in which the initial condition, boundary condition, forcing or source functions, PDE parameters, etc., change. Examples of use cases for multitask PDE problems for PINNs can be found in [22, 54, 55, 56, 57]. PINNs themselves are not a multitask model without modification since they each solve one task individually. In this work, we are primarily interested in the representation of learned global basis functions in multitask PIML models arising from solving various multitask PDE problems and how well they generalize to difficult unseen tasks.

2.3. Multi-Head PINNs

Multi-head PINNs (MH-PINNs) were introduced in [58] as a framework for solving multiple PDE problems with PINNs without having to train K number of independent networks. This modification makes
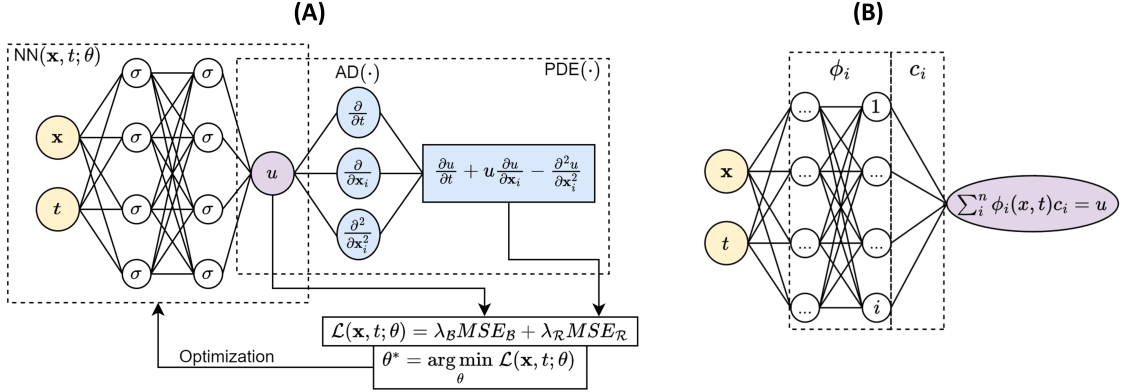
Figure 2: (A) Diagram of full Physics-Informed Neural Network including the PDE residual formulation using automatic differentiation and optimization process. (B) PINN solution as a sum of basis functions (ϕ_i) and coefficients (c_i).
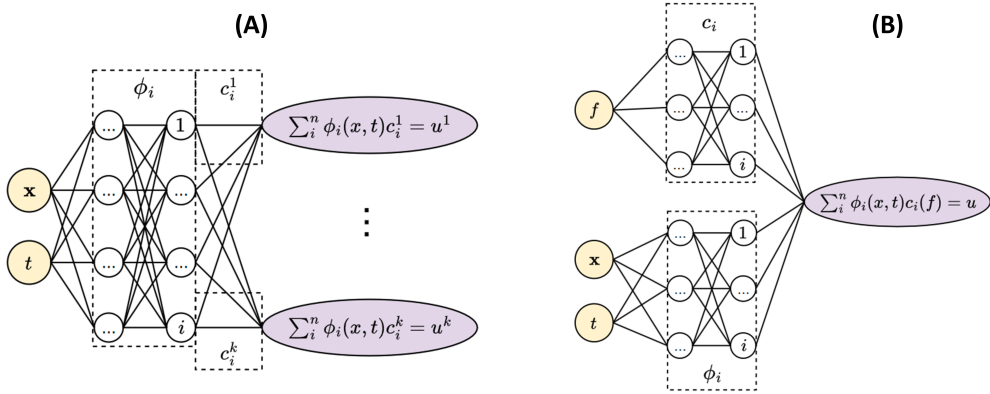

Figure 3: (A) Multi-head PINN where each "head" is a different linear combination (c_i) of the body network basis functions (ϕ_i). (B) Physics-Informed DeepONet architecture where the "branch" network represents the coefficients (c_i) and the "trunk" network represents the basis functions (ϕ_i).

MH-PINN a potent tool for multitask learning, generative modeling, and few-shot learning. Given this modification, we can modify the PINN optimization problem in Equation 1 as the new problem in Equation 2. The primary difference is that MH-PINNs have an outer sum over all tasks K and that each head creates a unique network prediction indexed as $u_{\boldsymbol{\theta}}^{(i)}$ for each task.

$$\boldsymbol{\theta}^* \leftarrow \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{K} \left(\frac{\lambda_{\mathcal{B}}}{N} \sum_{j=1}^{N} \left| u^{(i)}(\mathbf{x}_j, t_j) - u_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_j, t_j) \right|^2 + \frac{\lambda_{\mathcal{R}}}{M} \sum_{j=1}^{M} \left| \mathcal{R}\left(u_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_j, t_j) \right) \right|^2 \right) \quad (2)$$

In Figure 3, we denote two types of multitask PIML models. The first model (A) is one in which the "body" represents the global basis function, and the head represents the task-specific operations. Given this framework, a shared basis for the multitask problem is learned, and optimal coefficients are found to solve different problems, similar to spectral methods. Therefore, the problem is: how can we evaluate if our model has learned an optimal basis over the entire family of multitask problems, given it trains on a discrete set? Unlike in spectral methods, we perform complex optimization on a highly expressive neural network to form the set of basis functions instead of using predefined orthogonal ones such as Fourier series or Chebyshev polynomials, making this evaluation difficult. In our proposed methodology, we will return to this concept with Kolmogorov n-widths in Section 3.

2.4. PI-DeepONets

Deep Operator Networks (DeepONets) are a novel paradigm of scientific machine learning (SciML) in which a neural network is used as a surrogate model for function-to-function operators [59]. In the original formulation, the DeepONet performs nonlinear regression between two functions using experimental data. In PDE terms, an example would be learning a mapping between a forcing function and the resulting solution, $\mathcal{G} : f \rightarrow u$, for a fixed PDE and domain. A standard DeepONet consists of two networks: the branch, which represents basis coefficients conditioned on the input function, and the trunk, which represents the basis functions conditioned upon the output location in space and time. This framework, shown in Figure 3 (B), is based on the universal approximation theorem for operators [60].

Operator learning can be beneficial compared to models such as PINNs, which require a new network instance to be optimized for each problem. Neural operators predict new solutions at test time without the need for additional training. More recently, physics-informed DeepONets (PI-DONs) were introduced in [27, 28, 29] as a way to offset the large data requirement of this type of model. This optimization problem, stated in Equation 3, is similar to MH-PINNs with the distinction that our output is also conditioned upon the input function $f(\cdot)$ as well as the spatiotemporal coordinates $(\mathbf{X}, t)$. Both models seek to learn an optimal set of global basis functions for the multitask PDE problem. We, therefore, propose Kolmogorov n-widths as a concrete metric of comparison for models of this physics-informed multitask paradigm.

$$\boldsymbol{\theta}^* \leftarrow \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{K} \left(\frac{\lambda_{\mathcal{B}}}{N} \sum_{j=1}^{N} \left| u^{(i)}(\mathbf{x}_j, t_j) - u_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_j, t_j, f(\cdot)) \right|^2 + \frac{\lambda_{\mathcal{R}}}{M} \sum_{j=1}^{M} \left| \mathcal{R}\left(u_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_j, t_j, f(\cdot)) \right) \right|^2 \right) \quad (3)$$

3. Methods

3.1. Kolmogorov n-widths

In approximation theory, Kolmogorov n-widths are a measure of effectiveness of approximating functions. In other words, it is a measure of how well any n-dimensional subspace can approximate the solution manifold, defined in [61, 62, 63].

Definition 1. *Let $\mathcal{M}_n$ be any n-dimensional subspace of the normed linear space $\mathcal{M}$ with norm $||\cdot||_{\mathcal{M}}$. For each $x \in \mathcal{M}$,*

$$\delta(\mathcal{M}_n, x) = \inf_{y_n \in \mathcal{M}_n} ||x - y_n||_{\mathcal{M}} \quad (4)$$

is the distance of $\mathcal{M}_n$ to x, shown in Figure 4 (A). If $\delta(\mathcal{M}_n, x) = ||x - y_n^*||_{\mathcal{M}}$, then y_n^* is the "best" approximation from $\mathcal{M}_n$ to x. However, we are more often interested in how well an n-dimensional subspace $\mathcal{M}_n$ approximates another subset $\mathcal{A} \subset \mathcal{M}$, such as a solution manifold to a multitask PDE problem. In this case, we define

$$\delta(\mathcal{M}_n, \mathcal{A}) = \sup_{x \in \mathcal{A}} \inf_{y_n \in \mathcal{M}_n} ||x - y_n||_{\mathcal{M}} \quad (5)$$

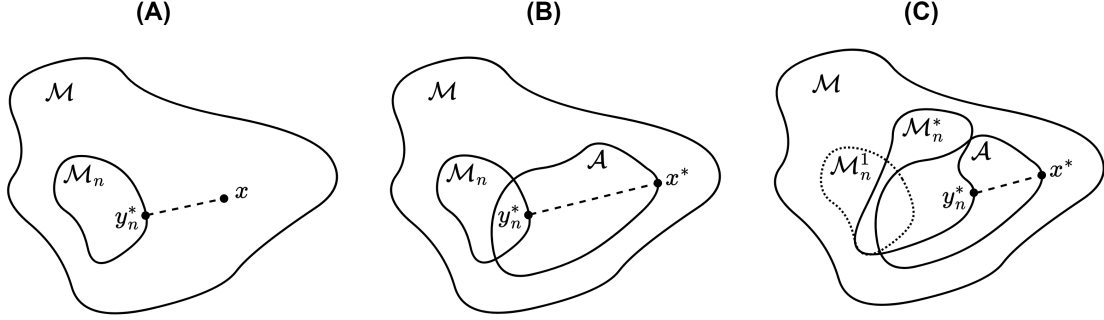*as the deviation from $\mathcal{M}_n$ to $\mathcal{A}$, shown in Figure 4 (B).*

Figure 4: (A) Distance of $\mathcal{M}_n$ to point $x \in \mathcal{M}$ where y_n^* is the best approximation, i.e., satisfies Equation 4. (B) Distance of $\mathcal{M}_n$ to $\mathcal{A}$ where y_n^* and x^* achieve the *sup inf* of $\mathcal{M}_n$ to $\mathcal{A}$, i.e., satisfy $\sup_{x \in \mathcal{A}} \inf_{y_n \in \mathcal{M}_n} ||x - y_n||_\mathcal{M}$. (C) Distance of $\mathcal{M}_n^*$ to $\mathcal{A}$ where y_n^* and x^* achieve the *inf sup inf* of $\mathcal{M}_n^*$ to $\mathcal{A}$, i.e., satisfy $\inf_{\mathcal{M}_n \in \mathcal{M}} \sup_{x \in \mathcal{A}} \inf_{y_n \in \mathcal{M}_n} ||x - y_n||_\mathcal{M}$.

Similarly, we can say that if $\delta(\mathcal{M}_n, \mathcal{A}) = ||x^* - y_n^*||_\mathcal{M} = \delta(\mathcal{M}_n, x^*)$, then y_n^* is the "best" approximation given by $\mathcal{M}_n$ to the "worst" or most challenging instance of $x \in \mathcal{A}$ for $\mathcal{M}_n$ to approximate. Given two n-dimensional subspaces $\mathcal{M}_n$ and $\mathcal{N}_n$, we can consider the ratio

$$\xi(\mathcal{A}, \mathcal{M}_n, \mathcal{N}_n) = \frac{\delta(\mathcal{M}_n, \mathcal{A})}{\delta(\mathcal{N}_n, \mathcal{A})} \tag{6}$$

as a metric for comparison when considering which approximating space we prefer. However, again, we are often more interested in how well our approximation $\mathcal{M}_n$ compares to other viable n-dimensional subspaces in $\mathcal{M}$, not just one. Therefore, let us define the Kolmogorov n-width as

$$\mathcal{K}(\mathcal{M}, \mathcal{A}) = \inf_{\mathcal{M}_n \in \mathcal{M}} \sup_{x \in \mathcal{A}} \inf_{y_n \in \mathcal{M}_n} ||x - y_n||_\mathcal{M} \tag{7}$$

which is the deviation from $\mathcal{M}_n^*$ to $\mathcal{A}$, shown in Figure 4 (C).

An n-dimensional subspace $\mathcal{M}_n^*$ is "optimal" if $\delta(\mathcal{M}_n^*, \mathcal{A}) = \mathcal{K}(\mathcal{M}, \mathcal{A})$. However, it is difficult to obtain $\mathcal{K}(\mathcal{M}, \mathcal{A})$ and determine the existence of optional approximating functions. In terms of PIML models, let us separate our learnable model parameters ($\boldsymbol{\theta}$) into two groups: the parameterization of the basis $\phi(W^1)$ and the parameterization of the coefficients $c(W^2)$, shown in Figures 2 & 3 for different architectures. Next, consider the generalized PDE problem

$$\mathcal{F}u(x) = f(x)$$
$$\text{where } u(x) = \sum_{i=1}^{n} c_i \phi_i(x). \tag{8}$$

For this prototypical PDE problem where the boundary conditions are enforced by the basis functions ϕ_i, we can define the Kolmogorov n-width as

$$\inf_{W^1} \sup_{c} \inf_{W^2} \left|\left| u(c) - \tilde{u}(W^1, W^2) \right|\right|_\mathcal{M}, \ c_i \in [a, b]. \tag{9}$$

Colloquially, this metric for PIML multitask problems can be described as an error estimate for the best learnable model basis ($\inf_{W^1}$) on the most challenging problem in the bounded multitask solution space ($\sup_c$) with the best basis coefficients ($\inf_{W^2}$). This is also the lower accuracy bound for a given architecture.

3.2. Multitask PIML Metric

To compute the Kolmogorov n-width metric, we perform a two-step optimization process. First, we perform the standard optimization for the given multitask PIML model, assuming we provide a sufficient

sampling of problems in the multitask PDE space over which to train. This optimization step is explicitly defined in Equation 2 for multi-head PINNs and in Equation 3 for physics-informed DeepONets. This results in learned basis functions, i.e., the output of the body network in MH-PINN or trunk network in PI-DON, which we store and hold constant for the second optimization step. By doing this, we have evaluated inf_{W^1} in Equation 9. The second step is to perform bi-optimization, which competitively trains the basis coefficients between the reference solution and the PIML model solution. Note that our multitask PDE problems are bounded, which is implemented by passing the coefficient variables through a sigmoid function and scaling it to the appropriate parametric interval for the problem. This step can be written as

$$\max_c \min_{W^2} \left\| \sum_{i=1}^N c_i \phi_i(\mathbf{x}, t) - \sum_{i=1}^M W_i^2 \phi_i^{W^{1^*}}(\mathbf{x}, t) \right\|_2 \tag{10}$$

where we take the norm in Equation 9 as the two-norm and assume through sufficient optimization the numerically derived *max min* approximates the *sup inf*. This process is provided in Algorithm 1, where any optimizer can be used.

Remark 1. *While there is no strict rule on which optimization scheme to use, it is important to note that this choice can play a large role in the final result. Different simultaneous optimization schemes result in different training trajectories. In Section 3.4, we briefly discuss this in the context of PIML.*

Algorithm 1 Algorithm for computing Kolmogorov n-width metric for multitask PIML architectures

---

Require: $\mathcal{P} : \mathcal{P} \in$ Error estimator
Require: $\mathcal{D} : \mathcal{D} \in$ Optimizer
Require: $\mathcal{L} =$ Physics-informed multitask loss function
Require: $\mathcal{K}(\mathcal{P}(\cdot)) =$ Kolmogorov n-width
    Initialize model parameters $\boldsymbol{\theta} = \{W^1, W^2\} \in \mathbb{R}$
    for $i \leftarrow 1$ **to** N **do**
        $\boldsymbol{\theta} \leftarrow \mathcal{D}(\mathcal{L}(\boldsymbol{\theta}, \mathbf{x}, t))$ $\triangleright$ Model optimization
    end for
    Store learned basis functions ($\phi(W^1)$)
    for $i \leftarrow 1$ **to** M **do**
        $c \leftarrow \mathcal{D}(\mathcal{K}(\mathcal{P}(\phi, c, W^2)))$ $\triangleright \mathcal{K}$ bi-optimization
        $W^2 \leftarrow \mathcal{D}(\mathcal{K}(\mathcal{P}(\phi, c, W^2)))$ $\triangleright \mathcal{K}$ bi-optimization
    end for

3.3. Multitask PIML Regularization

    To perform multitask PIML learning with Kolmogorov n-width regularization, we propose simultaneous tri-optimization. Similar to computing the metric in Equation 10, we write our approximation of the Kolmogorov n-width as a *min max min*. This value is then added as a regularizer into the standard physics-informed optimization for the chosen model, such as Equation 2 for MH-PINNs and Equation 3 for PI-DON.

$$\min_{W^1} \max_c \min_{W^2} \left\| \sum_{i=1}^N c_i \phi_i(\mathbf{x}, t) - \sum_{i=1}^M c_i(W_i^2) \phi_i(\mathbf{x}, t; W^1) \right\|_2 \tag{11}$$
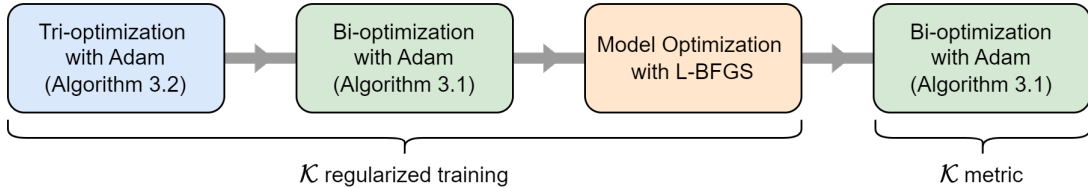
This process is provided in Algorithm 2.

Figure 5: Multi-step optimization process used to incorporate L-BFGS training into our Kolmogorov n-width regularization scheme.

Algorithm 2 Algorithm for training multitask PIML architecture with Kolmogorov n-width regularization

---

Require: $\mathcal{P} : \mathcal{P} \in$ Error estimator
Require: $\mathcal{D} : \mathcal{D} \in$ Optimizer
Require: $\mathcal{L} =$ Physics-informed multitask loss function
Require: $\mathcal{K}(\mathcal{P}(\cdot)) =$ Kolmogorov n-width
  Initialize model parameters $\boldsymbol{\theta} = \{W^1, W^2\} \in \mathbb{R}$
  for $i \leftarrow 1$ **to** N **do**
    $\boldsymbol{\theta} \leftarrow \mathcal{D}(\mathcal{L}(\boldsymbol{\theta}, \mathbf{x}, t) + \mathcal{K}(\mathcal{P}(\phi, c, W^2)))$ $\triangleright$ Model tri-optimization
    $c \leftarrow \mathcal{D}(\mathcal{K}(\mathcal{P}(\phi, c, W^2)))$ $\triangleright \mathcal{K}$ tri-optimization
    $W^2 \leftarrow \mathcal{D}(\mathcal{K}(\mathcal{P}(\phi, c, W^2)))$ $\triangleright \mathcal{K}$ tri-optimization
  end for

In practice, training physics-informed models is commonly done in two optimization stages: Adam [64] and then L-BFGS [65]. Adam is a first-order gradient-based stochastic optimizer with a low memory requirement. Adam is used in the first stage of training as a warm-up because it has been empirically shown to handle the large jumps in model parameter updates that come with randomly initialized weights and biases of the model. L-BFGS is a second-order (Hessian) gradient-based optimizer with a low memory requirement. L-BFGS is used in the second stage since it has been empirically shown to converge to much lower physics-informed losses than Adam only. However, it is often unstable in the early stages of training, far away from a local minimum's basin of attraction. Therefore, Adam is used to overcome the initial instability in this method, and L-BFGS is used to refine $\boldsymbol{\theta}$ to achieve orders of magnitude lower loss. Following this logic and based on trial and error, we find that the tri-optimization process in Algorithm 2 works with three Adam optimizers. However, when training with L-BFGS as the model optimizer, it becomes unstable and fails due to the updates being made by the other two optimizers. Therefore, to obtain the best model performance, which requires we use L-BFGS, we propose the following optimization steps in Figure 5. First, we perform Algorithm 2 with Adam to approximate the parameter sets $\{\boldsymbol{\theta}, c, W^2\}$. We then hold $\boldsymbol{\theta}$ constant and perform Algorithm 1 to refine our "worst possible" regularization case parameterized by c. We then hold c constant and perform L-BFGS training on $\{\boldsymbol{\theta}, W^2\}$, which is standard to finish physics-informed training as discussed. Finally, we perform Algorithm 1 again with the model basis refined by L-BFGS to compute the final Kolmogorov n-width metric for reporting. Alternatively, if L-BFGS training is unnecessary, Algorithm 2 can be used as is.

3.4. Simultaneous Optimization & Solution Space Construction

In this section, we briefly discuss different simultaneous optimization schemes that could be used to compute the proposed metric, as well as investigate choices made when constructing the reference solutions during simultaneous optimization. Simultaneous optimization is ubiquitous in other machine learning domains such as Generative Adversarial Networks (GANs) [66] and has become prevalent in PIML with the use of self-adaptive weighting schemes [14, 67]. Similarly, we must perform simultaneous optimization in Algorithms 1 & 2. Given that part of this optimization is finding optimal basis functions from a deep

neural network, the overall approximation of the Kolmogorov n-width will be non-convex if done simultaneously. Non-convex min-max optimization, its challenges, and applications have been thoroughly studied here [68]. We choose to use an optimization scheme common for physics-informed training, which are the Adam and L-BFGS optimizers, where the max is taken as the negative loss to perform gradient ascent, similar to self-adaptive PINN weights in [14]. Another option is to use competitive gradient descent [69], which is a robust scheme used to pit two neural network agents against each other in a min-max problem and avoids oscillatory behavior, extended to PINNs here [70]. However, in our Kolmgorov n-width case, our reference solution does not result from a second neural network. Finally, it is important to keep in mind with all non-convex optimization that the result will be an approximation since the global minimum is not guaranteed and that different realizations of the same process may lead to different results due to different training trajectories, parameter initialization, saddle points, etc.

Next, we investigate the effect of different choices made during the construction of the solution space. In the context of our numerical results (see Section 4), we use a sum of sine functions with variable coefficients to construct 1D Poisson and 2D nonlinear Allen-Cahn solution spaces without a magnitude falloff in higher frequency components. Normally, an elliptic problem would result in a simpler solution space biased toward lower frequencies than the solution we construct here. This limitation on high-energy states is necessary in traditional methods for convergence, which is not the case for neural networks. We make this choice to highlight the effect of neural network activation function choice on the generalizability of the learned basis functions. In this sense, we expect that sine activation functions should not only perform well on the sampled tasks but also generalize better to "worst-case" scenarios compared to tanh. Tanh activation functions, in the spirit of the Universal Approximation Theorem [71] for neural networks, can learn a set of basis functions that fit periodic high-frequency features well. However, this does not guarantee the learned basis will generalize well to unseen high-frequency solutions in the solution space. We observe this numerically and elaborate further in Section 4.

Now, consider we want to rectify our construction of the solution space by adding back the features we would obtain by manufacturing the forcing term and deriving the solution. In the context of the Kolmogorov n-width metric we defined or PIML multitask architectures in Equation 9, we can add normalization with the forcing norm:

$$\inf_{W^1} \sup_c \inf_{W^2} \frac{||u(c) - \tilde{u}(W^1, W^2)||_{\mathcal{M}}}{||f(c)||_{\mathcal{M}}}, \ c_i \in [a, b]. \tag{12}$$

By including this normalization, we add back the bias toward low frequencies for Laplacian-looking operators. Finally, consider we also want to analyze our choice of bounds for basis function coefficients (c_i). We performed our numerical experiments using a unit hypercube, but a unit ball is another common choice. Given these potential choices, we compute the Kolmogorov n-width, using Algorithm 1 on MH-PINN with sine activation functions and the other settings defined in Section 4, shown in Figure 6. Using the normalization in Equation 3.4 to add back the low energy bias creates a much similar competitive optimization problem, which does not highlight the discrepancy in tanh high-frequency generalization since solutions in that space are eliminated. Additionally, the coefficient unit ball also simplifies the solution space, removing the possibility of outliers such as the edges of the hypercube. Interestingly, the optimization becomes over-constrained by adding the normalization and unit ball constraints, and the coefficients do not change from their initialization, causing the Kolmogorov n-width to be the lowest due to the model basis functions fitting a fixed function, effectively removing the sup_c. For these reasons, we do not use those options in our numerical results so we can compare PIML architectures on sufficiently difficult multitask problems.

In future work, we hope to forego manufactured solutions and use FEM, or other numerical approximation methods, to construct our reference solutions and use the FEM basis functions during simultaneous optimization to compute the PIML Kolmogorov n-widths. However, this poses a few difficulties, such as how to enforce the correct bounds on basis function coefficients for a FEM surrogate concerning the higher-level multitask bounds for the PDE problem. This would require determining the variation each FEM nodal coefficient can have in a numerically efficient way for PDE problems with initial conditions, boundary conditions, forcing functions, etc., that have their own varying bounds that constrain the multitask problem.
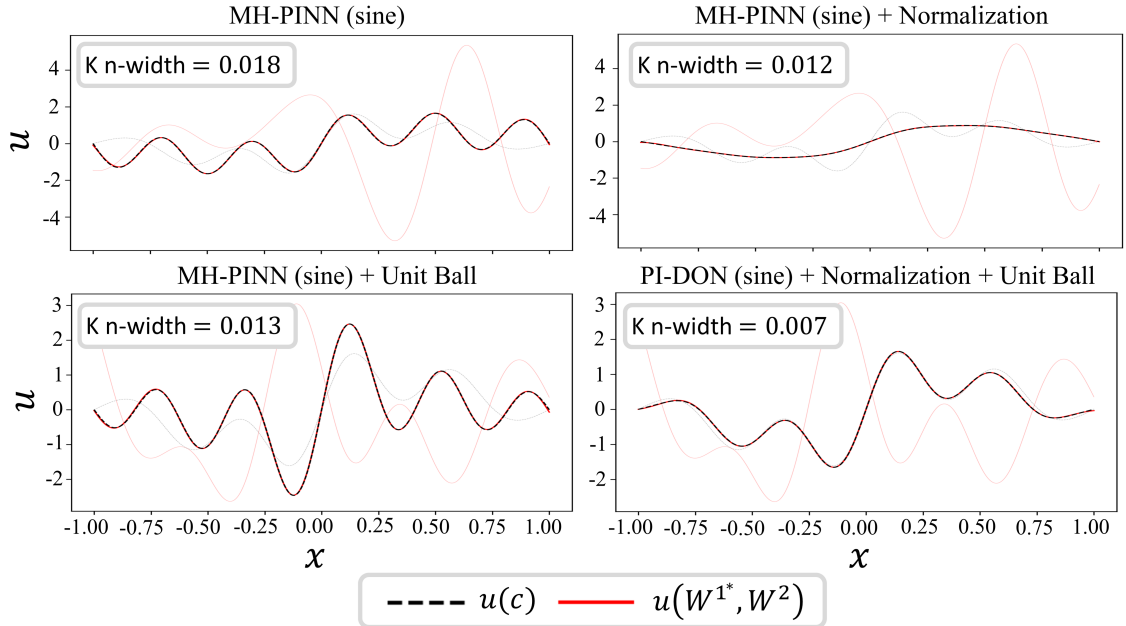
Figure 6: Plots of the Kolmogorov n-width competitive optimization results where the transparent lines represent the start of training and the solid lines represent the end of training.

We hope this step allows for real application of our Kolmogorov n-width method, such as in hypersonic vehicle [72], offshore structures [73] or earthquake localization [74] applications solved by DeepONets, which would require a solution surrogate. It is important to note that the solution surrogate, such as FEM, does not make training the PIML redundant since PIML has unique features like learning the operator, which can provide real-time inference on unseen tasks and outperform FEM in complex scenarios such as shape optimization [72]. Therefore, the importance of these models by leveraging traditional methods is a critical union to overcome real-world engineering challenges.

4. Numerical Experiments

In this section, we demonstrate the efficacy of our proposed methods on various multitask PDE problems. The code and data constituting these results are provided.[1] Ground truth solutions are generated using the method of manufactured solutions to derive the correct forcing term. For optimization of all models when using Kolmogorov n-width as a metric (see Algorithm 1), we use 1,000 Adam epochs followed by 5,000 L-BFGS epochs for the first model optimization step and then two separately defined Adam optimizers for 5,000 epochs for the bi-optimization step. For optimization of all models when using Kolmogorov n-width as a regularizer (see Algorithm 2 & Figure 5), we use 1,000 epochs of Adam for the tri-optimization warm-up procedure followed by the same epoch counts in the model optimization and bi-optimization steps as before. The total loss for any given model can be written as

$$\mathcal{L} = \lambda_{\mathcal{R}}\mathcal{L}_{\mathcal{R}} + \lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}} + \lambda_{\mathcal{K}}\mathcal{L}_{\mathcal{K}} \tag{13}$$

where the Lagrangian multipliers are $\lambda_{\mathcal{R}} = 1$ and $\lambda_{\mathcal{B}} = \lambda_{\mathcal{K}} = 10$ which help balance the loss function for optimal training. The terms $\mathcal{L}_{\mathcal{R}}$ and $\mathcal{L}_{\mathcal{B}}$ represent the loss of the PDE residual and boundary condition, respectively (explicitly stated in Equations 2 & 3). The term $\mathcal{L}_{\mathcal{K}}$ is 0 if unused, otherwise representing the

---

[1]The code and data accompanying this manuscript will be made publicly available at `https://github.com/mpenwarden/Knw-PIML` after publication
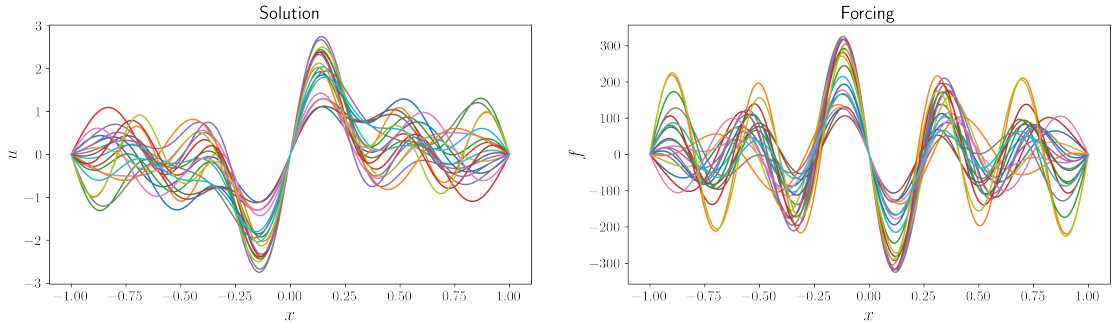
Figure 7: Plot of 20 1D Poisson solutions drawn from Equation 17 and the derived forcing function respectively.

Kolmogorov n-width regularization (see Equation 9). Each neural network (MH-PNN "body" or PI-DON "branch" & "trunk") is width 20 and depth two unless otherwise stated. Given that this study aims to compare fundamental aspects of different PIML architectures, small networks are sufficient in most cases. However, we also study the effect of varying sizes and training epochs on the computed metrics. Model performance is reported in relative L^2 error given by

$$\frac{||u - u_{\boldsymbol{\theta}}||_2}{||u||_2}. \tag{14}$$

When reporting the Kolmogorov n-width using the 2-norm, we also report it as the relative L^2 error since this metric has become standardized across PINN literature to report performance and allows for direct comparison with the mean of the multitask problem.

4.1. 1D Poisson equation

Let us consider the following PDE problem

$$\frac{\partial^2 u}{\partial x^2} = f(x), x \in [-1, 1], \tag{15}$$

$$u(-1) = u(1) = 0. \tag{16}$$

To form the dataset, we assume the solution takes the form

$$u(x) = \sum_{k=1}^{5} c_k sin(kx\pi) \tag{17}$$

with the corresponding forcing term

$$f(x) = -\pi^2 \sum_{k=1}^{5} k^2 c_k sin(kx\pi) \tag{18}$$

where c_k are independent and identically distributed (i.i.d) random variables drawn from the uniform distribution $U(0, 1)$. We use a uniform distribution of 512 residual points for physics-informed training. The generated dataset is shown in Figure 7.

Table 1 provides the results of different PIML architectures on the 1D Poisson problem. The (sine) and (tanh) nomenclature denote the type of activation function used in the network representing the basis functions (i.e., the "body" and "trunk" nets). Adding the $\mathcal{K}$ term denotes that our proposed Kolmogorov n-width regularization was used. The Kolmogorov n-width metric and mean and standard deviation of the sampled multitask problems are reported as the relative L_2 error for comparison. Note that the results

Table 1: Table of Kolmogorov n-width values for different PIML architectures for multitask 1D Poisson problem. The K n-width and mean and standard deviation of the sampled multitask problems are reported as the relative L_2 error for comparison.

Model architecture	K n-width	Sampled Multitask Error
MH-PINN (sine)	0.019	0.005 ± 0.003
MH-PINN (tanh)	0.301	0.015 ± 0.008
MH-PINN (sine) $+ \mathcal{K}$	0.008	0.010 ± 0.005
MH-PINN (tanh) $+ \mathcal{K}$	0.153	0.055 ± 0.003
PI-DON (sine)	0.232	0.022 ± 0.011
PI-DON (tanh)	0.472	0.087 ± 0.057
PI-DON (sine) $+ \mathcal{K}$	0.026	0.039 ± 0.038
PI-DON (tanh) $+ \mathcal{K}$	0.208	0.081 ± 0.062

for Kolmogorov n-width converge because the solution coefficients c_k are individually bounded; this is done in the context of machine learning by treating the coefficients as any other neural network learnable parameter, which is then passed through a sigmoid activation function to bound them on the interval $(0, 1)$ during optimization.

Comparing MH-PINN (sine) to MH-PINN (tanh), if one were to report the error over the discretely sampled multitask set, then one would observe the tanh architecture has a reported mean error $3.1\times$ that of the sine architecture. However, when comparing the architectures with the Kolmogorov n-width, the difference is now $16.2\times$ greater. Similarly, the MH-PINN (sine) architecture has a sampled error $4.4\times$ greater than PI-DON (sine), which changes to $12.2\times$ when reported as the Kolmogorov n-width. Furthermore, the Kolmogorov n-width evaluation, representing the lower accuracy bound, is, in both cases, an order of magnitude larger than the discretely sampled distribution of problems. In the regularization case, the sampled error increases, indicating less overfitting of the sampled tasks, which causes poor generalization to "worst-case" scenarios in the solution space. This indicates that purely reporting the sampled mean can lead to deceptive conclusions since any given architecture could perform better than another on a subset of problems, but when considering the solution space as a whole, that comparison could change. We claim this is a more appropriate and judicious metric to report as it is not subject to selective task sampling biases.

Remark 2. *Note that we are comparing MH-PINNs to PI-DONs on the grounds of their learned global basis functions; they are functionally different in that the former is a neural solver and the latter a neural operator [33]. We do not comment on their overall use, and we hope this metric can be used to make further improvements and insights on all types of architectures.*

In Figure 8, the results of Table 1 are shown as violin plots with the Kolmogorov n-widths denoted for all cases. The violin plots visualize the full distribution of the reported sampled errors in Table 1. We observe that in all cases, using the proposed $\mathcal{K}$ regularization decreased the final Kolmogorov n-width of the trained model. This is particularly true in the cases of the sine activation where the Kolmogorov n-width value is brought in line with the mean of the sample distribution, meaning there is no gap in reporting both metrics.

While the Kolmogorov n-width represents the lower accuracy bound, it is a numerical approximation conditioned on the assumption of an error estimator to learn the best possible basis coefficients. Note that this differs from the PIML architectures themselves, which are dataless (aside from the boundary conditions) and use physics-informed training to find a solution. In this sense, one can expect that the upper bound of the reported distribution is higher than our numerically computed Kolmogorov n-width due to the training difficulties inherent in PIML. These training challenges and accuracy limitations are well known [75, 40, 34, 38], and the physics-informed loss does not necessarily correlate to the error. This discrepancy is interesting because if these PIML architectures use their learned basis functions to the best of their ability, they should not result in such high errors, indicating a suboptimal use of their basis.

Figure 9 shows the Kolmogorov n-width metric results at the end of training, with the more transparent lines representing the beginning of training. Here, we can clearly see the approximation discrepancies made by using sine activation functions over tanh, which struggle to fit the higher frequency features. We can also
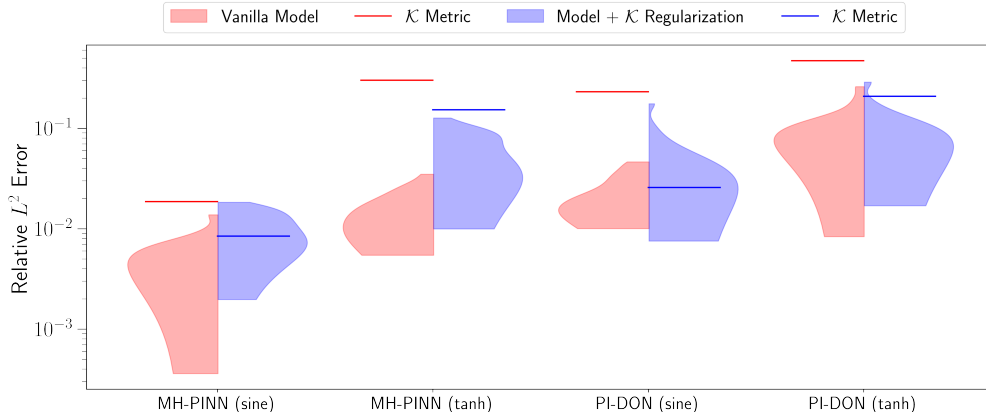
Figure 8: Violin plots showing the distribution of errors for the set of sampled multitask 1D Poisson problems. The resultant Kolmogorov n-width ($\mathcal{K}$) metric is also shown for the same set of basis functions used to solve the sampled problems.

observe how incorporating the regularization vastly improves the model's approximation of the worst case in the solution space for a given realization of learned basis functions.

Finally, we study the effect of architecture and optimization criteria choices on the observed gap between reporting the sampled mean versus the Kolmogorov n-width metrics. All models are MH-PINN (tanh) without regularization. In Figure 10, the relative difference, i.e. $\frac{\mathcal{K}_{L_2} - \overline{S}_{L_2}}{\mathcal{K}_{L_2}}$, is reported for different widths, depths, and optimization epochs (for the L-BFGS step). Note that a value of 0 indicates no discrepancy between the mean of the sampled multitask error ($\overline{S}_{L_2}$) and the Kolmogorov n-width ($\mathcal{K}_{L_2}$). The relative difference is clearly more dependent on the depth than the width, which appears almost invariant for shallow networks.

The lowest value found was 0.45 for depth = 4, width = 60, and epochs = 5,000; however, this is still around a $2\times$ difference of the metrics. Furthermore, we observe that more training epochs cause the discrepancy to increase. This implies that the lower values from the larger networks with fewer epochs are due to undertraining, which causes the sampled error and, thus, also the ratio to be lower. This causes the model to overfit the sampled tasks even more, given its greater expressivity, and perform worse in terms of Kolmogorov n-width. This study validates that regardless of network choices, a significant gap in Kolmogorov n-width and sampled error is still observed and that it is important to consider the entire solution space instead of potentially selectively sampled tasks with any sized network.

4.2. 2D nonlinear Allen-Cahn equation

Let us consider the following stationary PDE problem

$$\lambda \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + u(u^2 - 1) = f(x, y), (x, y) \in [0, 1] \times [0, 1], \tag{19}$$

$$u(x, 0) = u(x, 1) = u(0, y) = u(1, y) = 0, \tag{20}$$

where $\lambda = 0.1$ and the solution $u(x, y)$ takes the form

$$u(x, y) = \sum_{k=1}^{5} c_k sin(kx\pi) sin(ky\pi) \tag{21}$$

where c_k are (i.i.d) random variables drawn from the uniform distribution $U(0, 1)$. We use a uniform grid of size 51×51 to evaluate the residual points for physics-informed training. The generated dataset is shown in Figure 11.
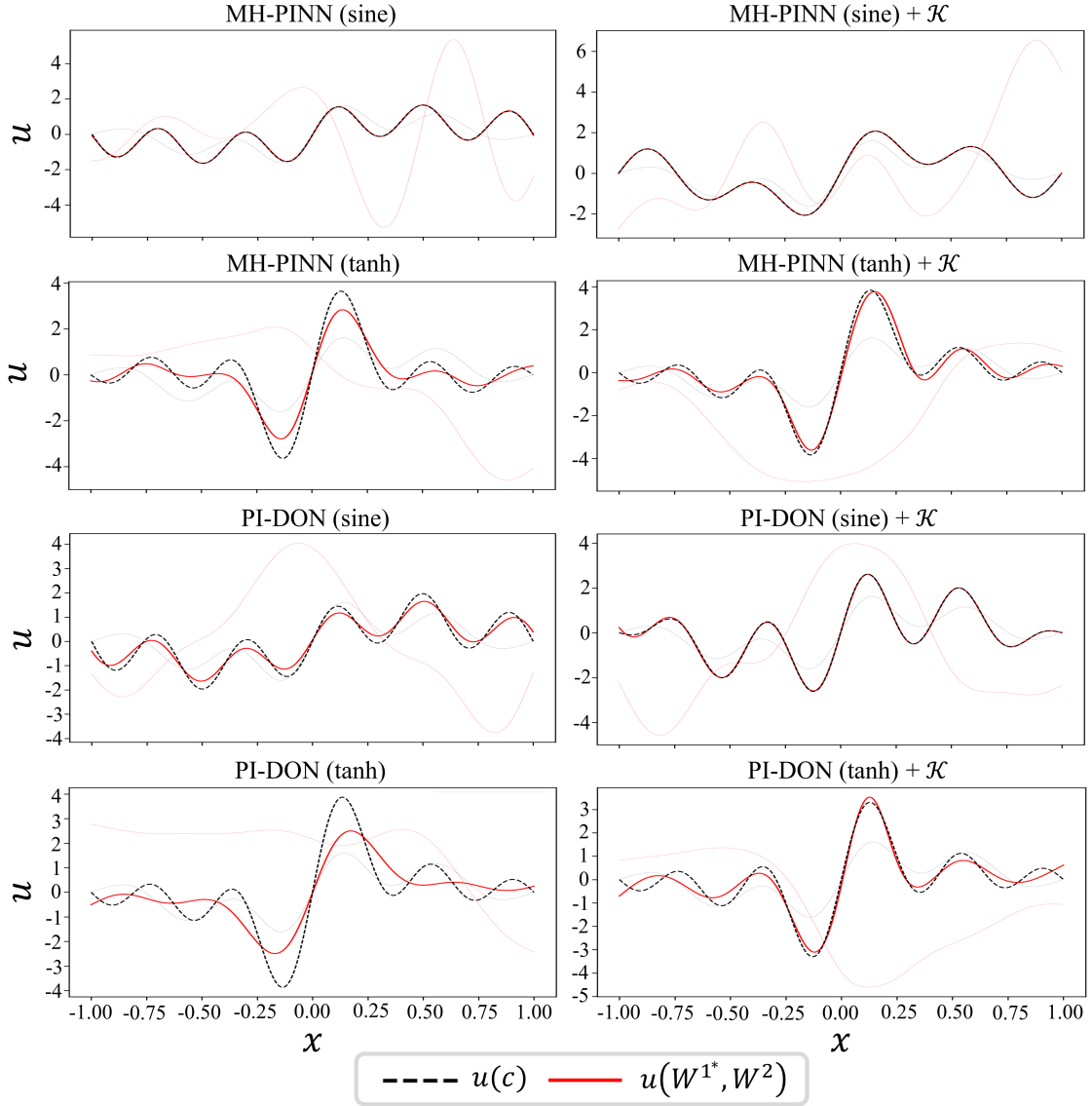
Figure 9: Plots of the Kolmogorov n-width competitive optimization results for the 1D Poisson problem. The *minmax* optimization procedure (shown in Algorithm 1) is performed to find the "worst-case" scenario in the solution space for a fixed set of learned basis functions $\phi(W^{1^*})$ with the best possible coefficients (W^2).
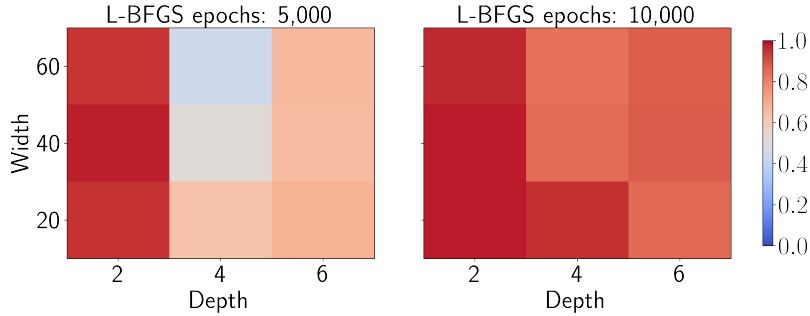
14

Figure 10: Heat map of the relative difference between the Kolmogorov n-width metric and mean of the sampled multitask problem metric where a value of 0 represents no reporting discrepancy.

Remark 3. *For ease of implementation of competitively training the solution and learned basis functions to compute the Kolmogorov n-width and to highlight different architecture choices such as sine versus tanh activation functions, we have chosen to construct $u(\cdot)$ out of sine functions instead of $f(\cdot)$. While this is a trivial change in the linear 1D Poisson problem, we acknowledge that this, in some sense, simplifies the nonlinear problem since we would expect a more complex set of solution basis functions if $f(\cdot)$ were constructed out of a sum of sine functions. Our main goal is to showcase our proposed metric to provide insight and benchmarking for PIML architectures, not solve the most challenging problem. In future work, we hope to study more complex constructions of the solution basis, such as using FEM as the reference solution and basis instead of the method of manufactured solutions.*
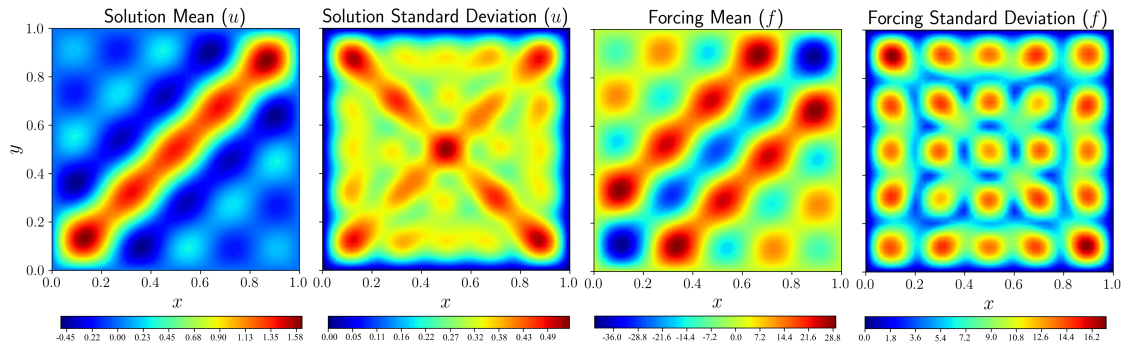

Figure 11: Plot of 20 2D nonlinear Allen-Cahn solutions drawn from Equation 21 and the derived forcing functions respectively.

Table 2 provides the final results of all considered architectures for the 2D nonlinear Allen-Cahn Problem. Overall, we observe similar trends in the gap between the Kolmogorov n-width metric and the sampled multitask mean. For example, while the sampled means are almost equivalent for MH-PINN (sine) and (tanh), the Kolmogorov n-width is almost five times larger. This indicates our method should be invariant to the spatial dimension, at least to the extent that the Kolmogorov n-width still indicates the models overfit the set of discrete problems regardless of dimension. Interestingly, MH-PINN (sine) has no gap between the two metrics, indicating a good set of learned basis functions. We also observe that unlike MH-PINN (sine), PI-DON (sine) suffers very bad generalizability despite having the advantage of sine activation functions and an incredibly low sampled mean error. This feature is also present in the 1D Poisson results, where the PI-DON architecture fails to generalize well to the worst case despite achieving similar performance on the sampled tasks. It could be the case that the additional complexity of the "branch" network, which allows the PI-DON to be a neural operator and predict new solutions without additional training, causes poor generalization of the learned basis functions, which is masked when only reporting the sampled task error instead of the Kolmogorov n-width. These kinds of insights can not only be used to compare competing

15

Table 2: Table of Kolmogorov n-width values for different PIML architectures for multitask 2D nonlinear Allen-Cahn problem.

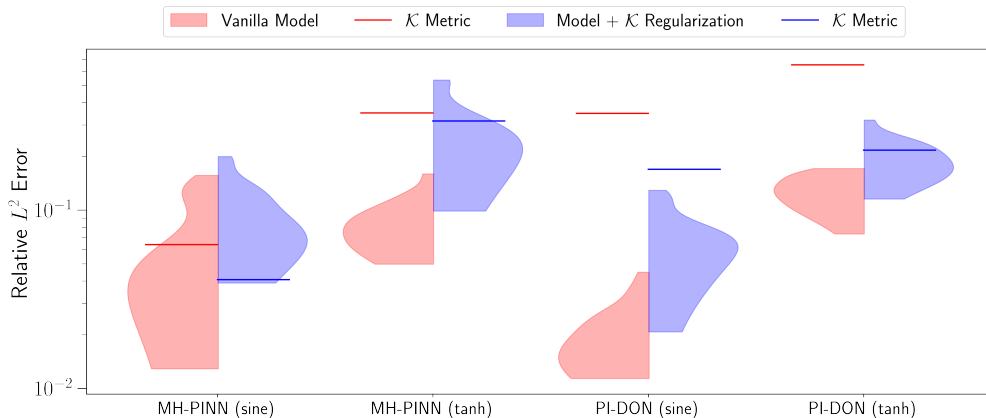| Model settings	Relative K n-width	Sampled Relative L^2 Error
MH-PINN (sine)	0.064	0.063 ± 0.044
MH-PINN (tanh)	0.397	0.085 ± 0.027
MH-PINN (sine) $+ \mathcal{K}$	0.060	0.092 ± 0.043
MH-PINN (tanh) $+ \mathcal{K}$	0.098	0.249 ± 0.120
PI-DON (sine)	0.347	0.020 ± 0.009
PI-DON (tanh)	0.652	0.127 ± 0.027
PI-DON (sine) $+ \mathcal{K}$	0.169	0.065 ± 0.028
PI-DON (tanh) $+ \mathcal{K}$	0.217	0.193 ± 0.049

Figure 12: Violin plots showing the distribution of errors for the set of sampled multitask 2D nonlinear Allen-Cahn problems. The resultant Kolmogorov n-width ($\mathcal{K}$) metric is also shown for the same set of basis functions used to solve the sampled problems.

architectures but also show where improvements can be made, for example, to the generalizability of the "trunk" net in PI-DONs. Finally, as in the 1D case, we still note that our Kolmogorov n-width regularization procedure improves the metric in all cases.

In Figure 12, the results of Table 2 are shown as violin plots for the 2D nonlinear Allen-Cahn problem. Similar to the 1D Poisson problem, we see a large gap between the sampled distribution and the Kolmogorov n-width. However, in the 2D case, the PI-DON architectures (both sine & tanh) perform much better on the sampled tasks than in 1D, almost equivalent to the MH-PINN distributions. This would imply the models learned a similar set of basis functions given their similar performance. However, we can see that in terms of the Kolmogorov n-width, despite achieving similar performance, the basis are quite different.

Figure 9 shows the Kolmogorov n-width metric results at the end of training. The black outlined plots are the "worst-case" scenario, and the red outlined plots are the point-wise error between the "worst-case" scenario and the best approximation made with the learned model basis functions. Similar to the 1D case, we also observe that the tanh activation function models struggle to fit higher-frequency features. Interestingly, the point-wise error plots using the regularization method show more complex features than the standard periodic errors without regularization. This, in fact, is indicative that their learned basis functions are richer, which can be seen when plotting, and we further study in Figure 14.

Finally, we perform a study of the learned 2D basis functions for the nonlinear Allen-Cahn problem to help quantify and explain the differences observed in the metrics. Instead of manually plotting and searching for visual differences and trends between the output of final layer neurons in the network representing ϕ, we seek to quantify them with their singular value spectra. In [25], the authors propose to use the decay rate of singular values of temporal snapshots of the solution can be used as an *a priori* guideline for the
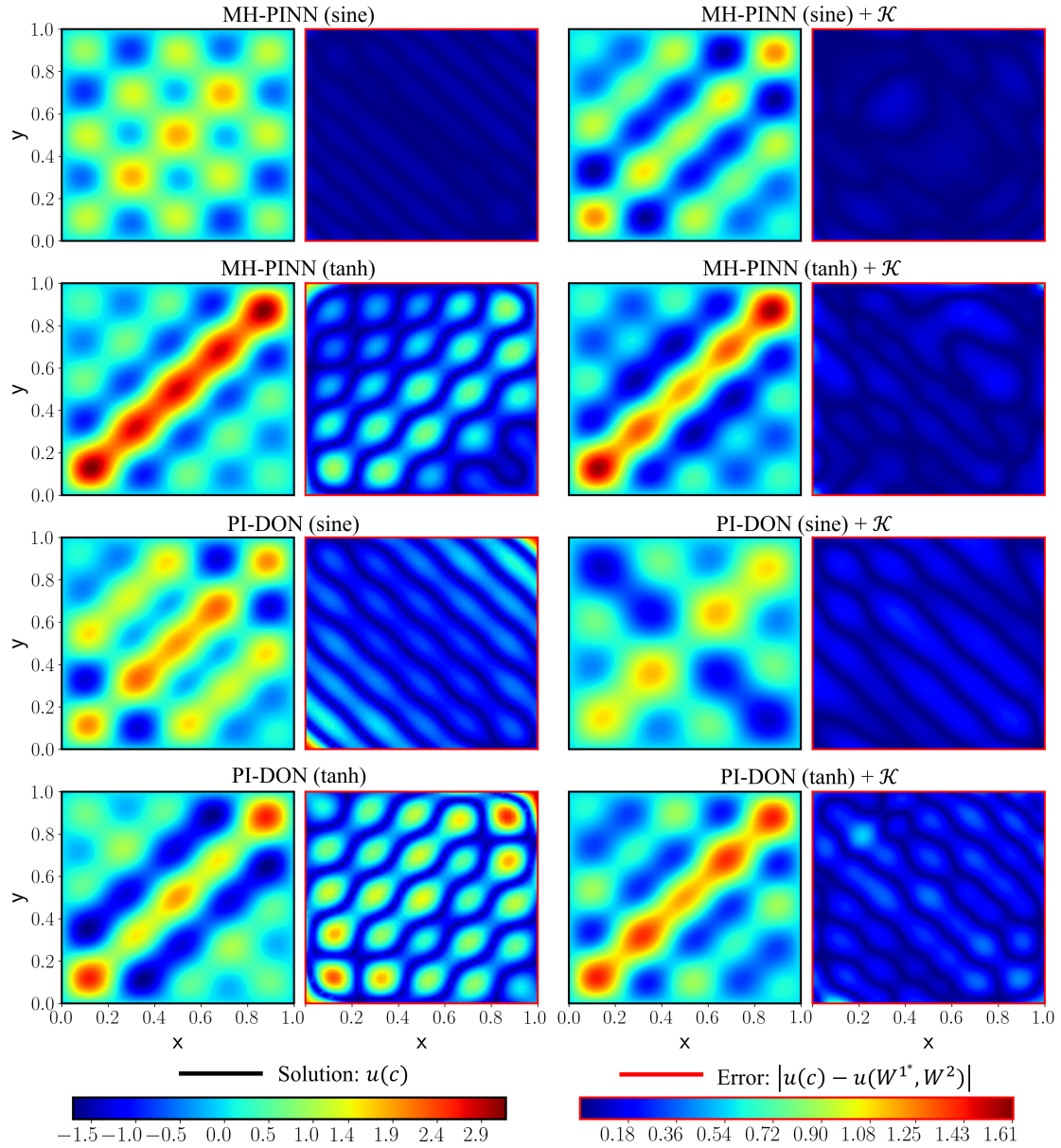
Figure 13: Plots of the Kolmogorov n-width competitive optimization results for the 2D nonlinear Allen-Cahn problem. The *minmax* optimization procedure (shown in Algorithm 1) is performed to find the "worst-case" scenario in the solution space for a fixed set of learned basis functions $\phi(W^{1^*})$ with the best possible coefficients (W^2).
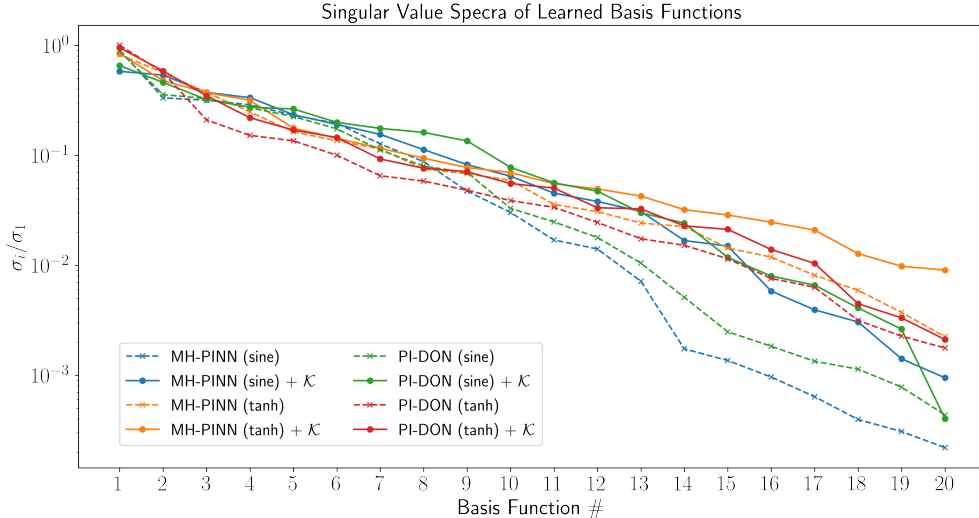
Figure 14: Plots of the rate of decay of (normalized) singular values of the set of learned basis functions. The results are computed by performing singular value decomposition (SVD) on the vectorized set of 2D basis functions where the matrix is of shape: **neurons in the final layer of the ϕ network $\times$ vec(spatial discretization)**, in our case **20×2601**.

complexity of PINN training. In a similar vein of investigation, we instead look at the decay rate of singular values of the learned basis functions instead of solution snapshots. This gives us insight into the richness of the learned basis for different architecture and training schemes, such as our regularization procedure. We observe that in all cases, the $\mathcal{K}$ regularization procedure forces the learned basis to become richer and more expressive to fit the features in the "worst-case" scenario, which, in combination with the prior discussions on the metric, results in a more generalizable architecture. Interestingly, the sine activation function models have a higher rate of decay than tanh despite performing better or equal in terms of sampled and n-width error. This indicates it is not just about expressivity but also about learning the right set of expressive basis, for which sine, in this case, is expected to perform better in terms of Kolmogorov n-width, given our solution space is constructed from sine functions. This also implies orthogonality does not guarantee generalization in the sense of Kolmogorov n-widths since a completely orthogonal set of basis would have no singular value decay, but tanh, which has less decay, performs worse than sine. In future work, we hope to investigate the effect of enforcing orthogonality of PIML basis functions under the lens of Kolmogorov n-width and singular value spectra to understand the nature of physics-informed training better. We also hope to provide a more comprehensive study of existing PIML training modifications and their effect on the learned basis functions, such as Fourier feature embeddings [76] or adaptive activation functions [15].

Remark 4. *While the authors of [25] claim the singular values of snapshots represent the Kolmogorov n-width, it does not say anything about the approximating functions since they only look at the known solution space, not the model approximation space. In our work, we are fully estimating the inf sup inf to obtain our metric.*

5. Conclusions

We have introduced a novel optimization procedure for computing the Kolmogorov n-width, a measure of the effectiveness of approximating functions, for analyzing multitask PIML architectures. We have found that multitask PIML models overfit the discrete training tasks which can result in poor generalization to "worst-case" scenarios that exist in the entire task family. This can also lead to a form of selective sampling, where reporting the sampled task errors can result in misleading conclusions about the efficacy of the model and its learned basis functions. We objectively compare and benchmark different multitask PIML architectures

using the proposed metric. We also find that neural network choices such as activation function have dramatic performance effects and spectral bias only seen under the lens of Kolmogorov n-widths. We also successfully propose and implement a novel regularization scheme using Kolmogorov n-widths to condition the model against overfitting and improve the generalization of the learned basis functions when solving multitask PDE problems. Future investigations will explore constructing the reference solution with FEM basis functions to compute the Kolmogorov n-width as a step closer to real-world applications. We also hope to comprehensively study the effect existing PIML architecture modifications like adaptive activation functions or Fourier feature embeddings have on the learned basis functions and to benchmark a wider array of problems and architectures.

Appendix A. Symbols & Notations

Table A.3: Symbols & Notations

$u(\cdot)$	PDE solution
ϕ_i	Basis Functions
c_i	Basis Coefficients
$\phi_i(W^1)$	Network Parameterization of Basis Functions
$c_i(W^2)$	Network Parameterization of Basis Coefficients
$\boldsymbol{\theta}$	Learnable network parameters
$\boldsymbol{\theta}^*$	Optimized network parameters
$u_{\boldsymbol{\theta}}(\cdot)$	PINN PDE prediction
$\mathcal{R}(\cdot)$	PDE residual
$\mathcal{F}(\cdot)$	Nonlinear operator
$\mathcal{S}(\cdot)$	Source function
$\mathcal{B}(\cdot)$	Boundary/initial condition
$\mathcal{L}(\cdot)$	Loss Function
$\mathcal{K}(\cdot)$	Kolmogorov n-width
Ω	Spatial domain of interest
T	Temporal domain of interest
$\mathbf{x}$	Spatial value, $\mathbf{x} \in \Omega$
t	Temporal value, $t \in T$

Bibliography

References

[1] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (6) (2021) 422–440. doi:10.1038/s42254-021-00314-5.
URL https://doi.org/10.1038/s42254-021-00314-5

[2] M. Raissi, P. Perdikaris, G. 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 (2019) 686–707.

[3] A. Mathews, M. Francisquez, J. W. Hughes, D. R. Hatch, B. Zhu, B. N. Rogers, Uncovering turbulent plasma dynamics via deep learning from partial observations, Physical Review E 104 (2) (2021) 025205.

[4] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering 358 (2020) 112623.

[5] A. Yazdani, L. Lu, M. Raissi, G. E. Karniadakis, Systems biology informed deep learning for inferring parameters and hidden dynamics, PLOS Computational Biology 16 (2020) 1–19. doi:10.1371/journal.pcbi.1007575.
URL https://doi.org/10.1371/journal.pcbi.1007575

[6] S. Wang, P. Perdikaris, Deep learning of free boundary and Stefan problems, Journal of Computational Physics 428 (2021) 109914. doi:https://doi.org/10.1016/j.jcp.2020.109914.
URL https://www.sciencedirect.com/science/article/pii/S0021999120306884

[7] K. Shukla, P. C. Di Leoni, J. Blackshire, D. Sparkman, G. E. Karniadakis, Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks, Journal of Nondestructive Evaluation 39 (2020) 1–20.

[8] Y. Chen, L. Lu, G. E. Karniadakis, L. D. Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials, Opt. Express 28 (8) (2020) 11618–11633. doi:10.1364/OE.384875.
URL https://opg.optica.org/oe/abstract.cfm?URI=oe-28-8-11618

[9] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, E. Kuhl, Physics-Informed Neural Networks for Cardiac Activation Mapping, Frontiers in Physics 8 (2020). doi:10.3389/fphy.2020.00042.
URL https://www.frontiersin.org/articles/10.3389/fphy.2020.00042

[10] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Review 63 (1) (2021) 208–228.

[11] A. Daw, J. Bu, S. Wang, P. Perdikaris, A. Karpatne, Rethinking the Importance of Sampling in Physics-informed Neural Networks, arXiv preprint arXiv:2207.02338 (2022).

[12] S. Subramanian, R. M. Kirby, M. W. Mahoney, A. Gholami, Adaptive Self-supervision Algorithms for Physics-informed Neural Networks, arXiv preprint arXiv:2207.04084 (2022).

[13] S. Wang, S. Sankaran, P. Perdikaris, Respecting causality is all you need for training physics-informed neural networks, arXiv preprint arXiv:2203.07404 (2022).

[14] L. McClenny, U. Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism, arXiv preprint arXiv:2009.04544 (2020).

[15] A. D. Jagtap, K. Kawaguchi, G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, Journal of Computational Physics 404 (2020) 109136.

[16] A. D. Jagtap, G. E. Karniadakis, How important are activation functions in regression and classification? a survey, performance comparison, and future directions, Journal of Machine Learning for Modeling and Computing 4 (1) (2023).

[17] J. Yu, L. Lu, X. Meng, G. E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Computer Methods in Applied Mechanics and Engineering 393 (2022) 114823.

[18] A. D. Jagtap, G. E. Karniadakis, Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations, Communications in Computational Physics 28 (2020) 2002–2041. doi:10.4208/cicp.OA-2020-0164.

[19] A. D. Jagtap, E. Kharazmi, G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Computer Methods in Applied Mechanics and Engineering 365 (2020) 113028. doi:https://doi.org/10.1016/j.cma.2020.113028.
URL https://www.sciencedirect.com/science/article/pii/S0045782520302127

[20] X. Meng, Z. Li, D. Zhang, G. E. Karniadakis, PPINN: Parareal physics-informed neural network for time-dependent PDEs, Computer Methods in Applied Mechanics and Engineering 370 (2020) 113250.

[21] Z. Hu, A. D. Jagtap, G. E. Karniadakis, K. Kawaguchi, Augmented physics-informed neural networks (apinns): A gating network-based soft domain decomposition methodology, Engineering Applications of Artificial Intelligence 126 (2023) 107183.

[22] M. Penwarden, S. Zhe, A. Narayan, R. M. Kirby, A metalearning approach for physics-informed neural networks (pinns): Application to parameterized pdes, Journal of Computational Physics 477 (2023) 111912. doi:https://doi.org/10.1016/j.jcp.2023.111912.
URL https://www.sciencedirect.com/science/article/pii/S0021999123000074

[23] S. Li, M. Penwarden, Y. Xu, C. Tillinghast, A. Narayan, R. M. Kirby, S. Zhe, Meta learning of interface conditions for multi-domain physics-informed neural networks, arXiv preprint arXiv:2210.12669 (2022).

[24] A. F. Psaros, K. Kawaguchi, G. E. Karniadakis, Meta-learning pinn loss functions, Journal of computational physics 458 (2022) 111121.

[25] R. Mojgani, M. Balajewicz, P. Hassanzadeh, Kolmogorov n–width and Lagrangian physics-informed neural networks: A causality-conforming manifold for convection-dominated PDEs, Computer Methods in Applied Mechanics and Engineering 404 (2023) 115810. `doi:https://doi.org/10.1016/j.cma.2022.115810`.
URL `https://www.sciencedirect.com/science/article/pii/S0045782522007666`

[26] U. Braga-Neto, Characteristics-Informed Neural Networks for Forward and Inverse Hyperbolic Problems, arXiv preprint arXiv:2212.14012 (2022).

[27] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed deeponets, Science advances 7 (40) (2021) eabi8605.

[28] S. Goswami, A. Bora, Y. Yu, G. E. Karniadakis, Physics-Informed Neural Operators, arXiv preprint arXiv:2207.05748 (2022).

[29] S. Goswami, M. Yin, Y. Yu, G. E. Karniadakis, A physics-informed variational deeponet for predicting crack path in quasi-brittle materials, Computer Methods in Applied Mechanics and Engineering 391 (2022) 114587.

[30] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, arXiv preprint arXiv:2111.03794 (2021).

[31] T. Konuk, J. Shragge, Physics-guided deep learning using fourier neural operators for solving the acoustic vti wave equation, 82nd EAGE Annual Conference & Exhibition (2021).
URL `https://api.semanticscholar.org/CorpusID:244213900`

[32] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics–informed neural networks: Where we are and what's next, Journal of Scientific Computing 92 (3) (2022) 88.

[33] Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, J. Zhu, Physics-informed machine learning: A survey on problems, methods and applications, arXiv preprint arXiv:2211.08064 (2022).

[34] M. Penwarden, A. D. Jagtap, S. Zhe, G. E. Karniadakis, R. M. Kirby, A unified scalable framework for causal sweeping strategies for physics-informed neural networks (pinns) and their temporal decompositions, Journal of Computational Physics 493 (2023) 112464. `doi:https://doi.org/10.1016/j.jcp.2023.112464`.

[35] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM Journal on Scientific Computing 43 (5) (2021) A3055–A3081.

[36] S. Wang, H. Wang, P. 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 (2021) 113938.

[37] R. Mattey, S. Ghosh, A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations, Computer Methods in Applied Mechanics and Engineering 390 (2022) 114474. `doi:https://doi.org/10.1016/j.cma.2021.114474`.
URL `https://www.sciencedirect.com/science/article/pii/S0045782521006939`

[38] S. Wang, X. Yu, P. Perdikaris, When and why pinns fail to train: A neural tangent kernel perspective, Journal of Computational Physics 449 (2022) 110768. `doi:https://doi.org/10.1016/j.jcp.2021.110768`.
URL `https://www.sciencedirect.com/science/article/pii/S002199912100663X`

[39] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, Cambridge, MA, USA, 2016.

[40] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Advances in Neural Information Processing Systems 34 (2021) 26548–26560.

[41] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, arXiv preprint arXiv:1711.10561 (2017).

[42] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations, arXiv preprint arXiv:1711.10566 (2017).

[43] H. Gao, L. Sun, J.-X. Wang, Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain, Journal of Computational Physics 428 (2021) 110079. `doi:https://doi.org/10.1016/j.jcp.2020.110079`.
URL `https://www.sciencedirect.com/science/article/pii/S0021999120308536`

[44] Z. Fang, A high-efficient hybrid physics-informed neural networks based on convolutional neural network, IEEE Transactions on Neural Networks and Learning Systems 33 (10) (2022) 5514–5526. `doi:10.1109/TNNLS.2021.3070878`.

[45] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, Journal of Computational Physics 394 (2019) 56–81. `doi:https://doi.org/10.1016/j.jcp.2019.05.024`.
URL `https://www.sciencedirect.com/science/article/pii/S0021999119303559`

[46] N. Geneva, N. Zabaras, Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks, Journal of Computational Physics 403 (2020) 109056. `doi:https://doi.org/10.1016/j.jcp.2019.109056`.
URL `https://www.sciencedirect.com/science/article/pii/S0021999119307612`

[47] N. Wang, H. Chang, D. Zhang, Theory-guided auto-encoder for surrogate construction and inverse modeling, Computer Methods in Applied Mechanics and Engineering 385 (2021) 114037. `doi:https://doi.org/10.1016/j.cma.2021.114037`.
URL `https://www.sciencedirect.com/science/article/pii/S0045782521003686`

[48] F. A. Viana, R. G. Nascimento, A. Dourado, Y. A. Yucesan, Estimating model inadequacy in ordinary differential equations with physics-informed neural networks, Computers & Structures 245 (2021) 106458. `doi:https://doi.org/10.1016/j.compstruc.2020.106458`.
URL `https://www.sciencedirect.com/science/article/pii/S0045794920302613`

[49] R. Zhang, Y. Liu, H. Sun, Physics-informed multi-lstm networks for metamodeling of nonlinear structures, Computer Methods in Applied Mechanics and Engineering 369 (2020) 113226. doi:https://doi.org/10.1016/j.cma.2020.113226. URL https://www.sciencedirect.com/science/article/pii/S0045782520304114

[50] Y. A. Yucesan, F. A. Viana, Hybrid physics-informed neural networks for main bearing fatigue prognosis with visual grease inspection, Computers in Industry 125 (2021) 103386. doi:https://doi.org/10.1016/j.compind.2020.103386.

[51] V. Shankar, G. B. Wright, A. L. Fogelson, R. M. Kirby, A radial basis function (rbf)-finite difference method for the simulation of reaction-diffusion equations on stationary platelets within the augmented forcing method, International Journal for Numerical Methods in Fluids 75 (2014) 1–22.

[52] V. Shankar, G. B. Wright, R. M. Kirby, A. L. Fogelson, A radial basis function (rbf)-finite difference (fd) method for diffusion and reaction-diffusion equations on surfaces, Journal of Scientific Computing 63 (2014) 745–768.

[53] R. Caruana, Multitask learning, Machine learning 28 (1997) 41–75.

[54] M. Penwarden, S. Zhe, A. Narayan, R. M. Kirby, Multifidelity modeling for physics-informed neural networks (pinns), Journal of Computational Physics 451 (2022) 110844.

[55] B. Bahmani, W. Sun, Training multi-objective/multi-task collocation physics-informed neural network with student/teachers transfer learnings, arXiv preprint arXiv:2107.11496 (2021).

[56] P. Thanasutives, M. Numao, K.-i. Fukui, Adversarial multi-task learning enhanced physics-informed neural networks for solving partial differential equations, in: 2021 International Joint Conference on Neural Networks (IJCNN), IEEE, 2021, pp. 1–9.

[57] B. Wang, A. Qin, S. Shafiei, H. Dia, A.-S. Mihaita, H. Grzybowska, Training physics-informed neural networks via multi-task optimization for traffic density prediction, in: 2023 International Joint Conference on Neural Networks (IJCNN), IEEE, 2023, pp. 1–7.

[58] Z. Zou, G. E. Karniadakis, L-hydra: multi-head physics-informed neural networks, arXiv preprint arXiv:2301.02152 (2023).

[59] L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nature Machine Intelligence 3 (3) (2021) 218–229.

[60] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Transactions on Neural Networks 6 (4) (1995) 911–917. doi:10.1109/72.392253.

[61] B. A. Kolmogoroff, A. Kolmogoroff, Uber die beste annaherung von funktionen einer gegebenen funktionenklasse, Annals of Mathematics 37 (1936) 107. URL https://api.semanticscholar.org/CorpusID:124467241

[62] A. Pinkus, n-widths in approximation theory, 1985. URL https://api.semanticscholar.org/CorpusID:119692918

[63] J. A. Evans, Y. Bazilevs, I. Babuška, T. J. Hughes, n-widths, sup–infs, and optimality ratios for the k-version of the isogeometric finite element method, Computer Methods in Applied Mechanics and Engineering 198 (21) (2009) 1726–1741, advances in Simulation-Based Engineering Sciences – Honoring J. Tinsley Oden. doi:https://doi.org/10.1016/j.cma.2009.01.021. URL https://www.sciencedirect.com/science/article/pii/S0045782509000280

[64] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[65] D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Mathematical programming 45 (1-3) (1989) 503–528.

[66] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Advances in neural information processing systems 27 (2014).

[67] Z. Xiang, W. Peng, X. Liu, W. Yao, Self-adaptive loss balanced physics-informed neural networks, Neurocomputing 496 (2022) 11–34. doi:https://doi.org/10.1016/j.neucom.2022.05.015. URL https://www.sciencedirect.com/science/article/pii/S092523122200546X

[68] M. Razaviyayn, T. Huang, S. Lu, M. Nouiehed, M. Sanjabi, M. Hong, Nonconvex min-max optimization: Applications, challenges, and recent theoretical advances, IEEE Signal Processing Magazine 37 (5) (2020) 55–66.

[69] F. Schäfer, A. Anandkumar, Competitive gradient descent, Advances in Neural Information Processing Systems 32 (2019).

[70] Q. Zeng, Y. Kothari, S. H. Bryngelson, F. Schäfer, Competitive physics informed networks, arXiv preprint arXiv:2204.11144 (2022).

[71] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366. doi:https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208

[72] K. Shukla, V. Oommen, A. Peyvan, M. Penwarden, N. Plewacki, L. Bravo, A. Ghoshal, R. M. Kirby, G. E. Karniadakis, Deep neural operators as accurate surrogates for shape optimization, Engineering Applications of Artificial Intelligence 129 (2024) 107615.

[73] Q. Cao, S. Goswami, T. Tripura, S. Chakraborty, G. E. Karniadakis, Deep neural operators can predict the real-time response of floating offshore structures under irregular waves, Computers & Structures 291 (2024) 107228.

[74] E. Haghighat, U. b. Waheed, G. Karniadakis, A novel deeponet model for learning moving-solution operators with applications to earthquake hypocenter localization, arXiv preprint arXiv:2306.04096 (2023).

[75] S. Wang, S. Sankaran, H. Wang, P. Perdikaris, An expert's guide to training physics-informed neural networks, arXiv preprint arXiv:2308.08468 (2023).

[76] S. Dong, N. Ni, A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks, Journal of Computational Physics 435 (2021) 110242.