

Finite Element Representation Network (FERN) for Operator Learning with a Localized Trainable Basis

Zecheng Zhang ^{*}, Hao Liu [†], Guosheng Fu[‡], Hayden Schaeffer [§], Guang Lin [¶]

Abstract

We propose a finite-element local basis-based operator learning framework for solving partial differential equations (PDEs). Operator learning aims to approximate mappings from input functions to output functions, where the latter are typically represented using basis functions. While non-learnable bases reduce training costs, learnable bases offer greater flexibility but often require deep network architectures with a large number of trainable parameters. Existing approaches typically rely on deep global bases; however, many PDE solutions exhibit local behaviors such as shocks, sharp gradients, etc., and in parametrized PDE settings, these localized features may appear in different regions of the domain across different training and testing samples. Motivated by the use of local bases in finite element methods (FEM) for function approximation, we develop a shallow neural network architecture that constructs adaptive FEM bases. By adopting suitable activation functions, such as ReLU, the FEM bases can be assembled exactly within the network, introducing no additional approximation error in the basis construction process. This design enables the learning procedure to naturally mimic the adaptive refinement mechanism of FEM, allowing the network to discover basis functions tailored to intrinsic solution features such as shocks. The proposed learnable adaptive bases are then employed to represent the solution (output function) of the PDE. This framework reduces the number of trainable parameters while maintaining high approximation accuracy, effectively combining the adaptivity of FEM with the expressive power of operator learning. To evaluate performance, we validate the proposed method on seven families of PDEs with diverse characteristics, demonstrating its accuracy, efficiency, and robustness.

1 Introduction

Deep neural operators (DNOs) are a class of operator learning (OL) approaches [16, 20, 22, 25, 29, 35, 36, 38–40, 48, 49, 52, 63, 73, 74] that aims to approximate an operator G , which maps an input function u to an output function $v = G(u)$, by utilizing deep neural networks architectures. Since many problems in scientific computing can be formulated as operator approximations, OL has emerged as a powerful alternative technique within scientific machine learning (SciML) [9, 48, 50, 56–58, 58, 70] for solving computational problems. One of their advantages over classical

^{*}Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, USA. (Email: zecheng.zhang.math@gmail.com)

[†]Department of Mathematics, Hong Kong Baptist University, Hong Kong, China. (Email: haoliu@hkbu.edu.hk)

[‡]Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, USA. (Email: gfu@nd.edu)

[§]Department of Mathematics, UCLA, Los Angeles, CA 90095, USA. (Email: hayden@math.ucla.edu)

[¶]Department of Mathematics and Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA. (Email: guanglin@purdue.edu)

techniques is their faster inference time and their ability to adapt to problem-specific tasks. It has been shown theoretically that deep neural networks are adaptive to data structures and model smoothness [11, 23, 41, 53, 61]. Recent advances in multi-operator learning (MOL) [10, 28, 45–47, 54, 59, 60, 64–66, 68, 69], which employ a single network architecture to approximate multiple operators simultaneously, have significantly expanded the scope of OL. This development enables the application of OL to large-scale engineering problems [6, 17, 21, 31, 51, 55, 62, 75], positioning it as one of the few SciML frameworks already adopted in practical applications.

The study of operator learning started in the early 1990s with the works [12, 13], which established a universal approximation theorem for operators and laid the theoretical foundation of the field. In recent years, a variety of DNOs framework have been developed [20, 25, 30, 36–40, 48, 52, 63, 71, 73, 74]. For example, DeepONet is the first modern deep neural operator and was able to show applicability to spatiotemporal systems. However, the vanilla version of DeepONet is not invariant to the discretization of the input function; in particular, the input function cannot be discretized on different meshes. This limitation was partially resolved in [74] under suitable assumptions. Nevertheless, the discretization size was still required to remain the same across different functions. This restriction was improved on both numerically and theoretically in [71, 73], through the introduction of distributed learning strategies. Significant theoretical progress [4, 29, 32–34, 34, 42–44, 72] has been made to analyze properties such as convergence, generalization, and expressivity of neural operators, further solidifying OL as an important topic in SciML.

A central mathematical motivation for the design of DNOs is to represent the output function v as a linear combination of learnable basis functions (represent the basis functions using neural networks) with corresponding learnable coefficients. This idea is commonly used in numerical methods, such as the finite element method and spectral methods, which express functions as expansions over basis functions. Within OL, this idea was first illustrated in the shallow neural network universal approximation results of [12, 13], and was later rigorously established in [44], which provided complete neural scaling laws, including error convergence with respect to network size and generalization error with respect to the number of training samples.

An example of such a structure is the DeepONet [40, 48], which employs a trunk network to learn basis functions and a branch network to learn the coefficients. Another widely used framework is the FNO [36–38, 63] which adopts the iterative structure $u_{n+1} = \mathcal{F}^{-1}(\mathcal{K}_n(\mathcal{F}(u_n))) + W u_n$, where $\mathcal{F}$ denotes the Fourier transform, W is a learnable weight matrix, and $\mathcal{K}_n$ is a learned convolution. In this formulation, although part of the basis is fixed as Fourier and inverse Fourier modes, the term $W u$ still represents a linear combination of learned column basis functions weighted by coefficients determined by u . More recent architectures that leverage attention mechanisms [18, 46, 47, 59] to construct DNOs can also be interpreted within a similar framework, as attention effectively implements a data-dependent basis expansion with learned coefficients.

Both numerical evidence [40, 48] and theoretical analysis [44] demonstrate that, in order to achieve strong approximation performance, the networks used to learn the basis functions are typically deep, which results in globally supported basis functions, i.e., their support spans the entire domain of the input function u . Such approaches can be viewed as spectral methods [52], but with the important distinction that the basis functions are learnable and non-polynomial (depending on the choice of the neural network activation function). While global basis functions offer advantages such as smoothness and high accuracy for problems with globally coherent structures, they are less effective for problems involving localized phenomena, sharp gradients, or discontinuities. This is common in many PDEs with shocks, interfaces, or boundary layers. In such cases, locally supported basis functions, e.g., those used in the finite element or finite volume methods, generally yield better performance by more accurately resolving localized features. Moreover, OL aims to predict the solutions of a parametrized family of PDEs, implying that the

localized features of different samples are often evenly distributed across the solution landscapes. Consequently, local bases are sufficient to capture these features effectively, eliminating the need for global bases.

By contrast, the finite element method (FEM), originally introduced in [15, 19, 26], provides a rigorous framework for constructing locally supported basis functions [8, 14, 27] utilizing in solving PDE. FEM is one of the main approach employed in numerical methods since it is capable of handling a wide range of PDE with complex geometries through the use of unstructured meshes [3, 76]. In the most common setting, the basis elements are piecewise linear Lagrange basis functions, often referred to as hat functions, defined on elements of a triangulation of the computational domain. These locally supported basis functions enable accurate resolution of localized features.

The ReLU activation function [44, 67] is capable of constructing piecewise linear functions, which can be used for interpolation and for defining basis functions with local support. In particular, with only two degrees of freedom (trainable parameters), one can construct a one-dimensional hat function. Motivated by this observation, we propose to 2-parameter ReLU networks to construct local FEM-type basis functions for operator learning. Since each basis is parameterized by trainable variables, the approach can be regarded as an adaptive FEM method. We validate the proposed framework on a broad family of PDEs that are traditionally solved by FEM, and demonstrate that the newly designed neural operators achieve comparable or even improved accuracy while requiring significantly lower computational cost. Since our approach is directly inspired by the finite element representation of functions, we refer to it as the Finite Element Representation Network (FERN).

We summarize our contributions as follows:

1. We propose a novel operator learning framework based on finite-element local bases, where adaptive FEM bases are constructed directly through a shallow neural network. The use of local bases is crucial, as PDE solutions may exhibit localized behaviors such as shocks, sharp gradients, or fast decays. Moreover, in parametrized PDE problems, these local features may appear at different spatial locations across samples, making local bases essential for accurately capturing such variations while maintaining efficiency and interpretability.
2. By adopting suitable activation functions such as ReLU, the FEM bases can be assembled exactly within the network, introducing no approximation error in the basis construction process. The learning procedure naturally mimics the adaptive refinement mechanism of adaptive FEM, enabling the network to discover basis functions tailored to the underlying solution structures.
3. The proposed framework achieves high prediction accuracy with a reduced number of trainable parameters, owing to the low cost of constructing the bases. This provides an effective approach to employing shallow network structures while maintaining strong predictive performance. Extensive experiments on PDEs with distinct characteristics, including solutions whose landscapes exhibit bumps, demonstrate the method’s accuracy, efficiency, and robustness.

The rest of the manuscript is organized as follows. We introduce our methodology in Section 2, and will verify our methods and present the numerical evidence in Section 3. We finally discuss some future work in Section 4.

2 Review and methodology

2.1 Operator learning

Let us denote the operator $G : U \rightarrow V$ as the target operator, where U and V are function spaces defined on domains Ω_u and Ω_v with dimensions d_1 and d_2 , respectively. The objective of operator learning is to design a neural network structure G_θ , with θ representing all trainable parameters, to approximate G . More precisely, we aim to achieve

$$G(u)(x) := v(x) \approx G_\theta(\hat{u})(x),$$

for $u \in U$ and $x \in \Omega_v$, where $v = G(u)$ and $\hat{u}$ is a discretization of u .

A useful construction of G_θ is inspired by the classical numerical analysis, which first approximates $v \in V$ as

$$v(x) = \sum_{k=1}^N c_k \phi_k(x),$$

where c_k are coefficients and ϕ_k are basis functions. In standard numerical methods, the basis functions ϕ_k are chosen independently of the solution $v(x)$ and are fixed *a priori*, for example, as finite element basis functions. As a concrete example, in the low-order continuous finite element method, the function space V is defined as the space of continuous, piecewise linear functions over a fixed computational mesh. Each basis function ϕ_k is a hat function associated with node k of the mesh, taking the value one at node k and zero at all other nodes. This construction represents the function v as a linear combination of locally supported basis functions, yielding a flexible yet structured approximation framework.

In operator learning, the coefficients c_k must encode both u and G . Consequently, one can approximate the operator as

$$G(u)(x) \approx \sum_{k=1}^N c_k(u) \phi_k(x), \quad (1)$$

where $c_k : U \rightarrow \mathbb{R}$ are functionals on U encoding both u and G . To realize this approximation, one designs network structures $\hat{c}_k$ and $\hat{\phi}_k$ as surrogates for c_k and ϕ_k , respectively. In the remainder of this work, we adopt the terminology of [48], referring to the networks $\hat{c}_k$ as branch networks and $\hat{\phi}_k$ as trunk networks.

This framework was first studied mathematically in [13], whose authors analyzed shallow approximations of $\hat{c}_k$ and $\hat{\phi}_k$, although the convergence rates and parameter estimates for encoding u , encoding the operator, and constructing the basis $\hat{\phi}_k$ remained unclear. Subsequently, [48, 49, 52] proposed deep computational extensions of this shallow structure, significantly improving its empirical performance but without providing theoretical guarantees. Finally, the work of [44] rigorously established convergence and generalization results for this operator learning framework, which was the first explicit estimates of the parameter complexity required for u encoding, operator encoding, and basis construction.

Notably, [44] shows that the depth of the network $\hat{\phi}_k$ required to achieve an error of at most $\varepsilon > 0$ in the L^∞ norm scales as $-\log(\varepsilon)$. This implies that a deep architecture is necessary, which can be computationally expensive. This theoretical result is consistent with numerical observations: shallow structures for $\hat{\phi}_k$ often fail to yield satisfactory computational performance (see Section 3 for examples). A central challenge is to reduce the cost of constructing the basis while maintaining high prediction accuracy.

Additionally, the suggested total parameters count [44] for $\hat{c}_k$ is of order $(\sqrt{c_U}\varepsilon^{-(d_2+1)c_U})(c_U^2 \log c_U + c_U^2 \log(\varepsilon^{-1}))$ with $C_u = O(\varepsilon^{-d_1})$. This estimate dominates the parameter count for trunk network $\hat{\phi}_k$, which is of order $-\log(\varepsilon)$. However, in numerical experiments, the parameter count of the branch network typically dominates that of the trunk network to ensure accurate performance. This raises an important question: how can we design a structure that achieves high predictive accuracy while requiring fewer trainable parameters in $\hat{\phi}_k$, so that the network structures $\hat{c}_k$ and $\hat{\phi}_k$ are consistent with theoretical expectations.

2.2 Finite element methods

The finite element method is a powerful numerical technique for solving partial differential equations (PDEs). One of its key strengths lies in its ability to operate on unstructured meshes that can accurately represent complex geometries. In this framework, the computational domain Ω is discretized into a triangulation $\mathcal{T}_h = \{K\}$, where each K denotes an individual mesh element. Based on the continuity requirements of the problem, local finite element spaces are then constructed over this mesh. For example, the classical continuous piecewise linear Lagrange finite element space on a simplicial mesh is defined as

$$V_h(\mathcal{T}_h) = \{v_h \in C^0(\Omega) : v_h|_K \in P_1(K) \text{ for all } K \in \mathcal{T}_h\}, \quad (2)$$

where $P_1(K)$ represents the space of linear polynomials over the simplicial element K .

In particular, adaptive finite element methods (AFEM) dynamically refine the mesh to better resolve local features of the solution according to various adaptivity criteria [1, 2, 7]. For instance, let $\Omega \subset \mathbb{R}^2$ be a polygonal domain. If $u \in H^s(\Omega)$ with $0 < s < 2$, then there exists an admissible triangulation $\mathcal{T}_h$ with $\mathcal{O}(N)$ vertices such that [5]

$$\inf_{v_h \in V_h(\mathcal{T}_h)} \|u - v_h\|_{L^2(\Omega)} \leq C N^{-s/2} |u|_{H^s(\Omega)}, \quad (3)$$

where $V_h(\mathcal{T}_h)$ denotes the space of continuous, piecewise linear finite element functions defined in (2). The constant $C > 0$ depends only on the domain Ω and on the shape-regularity of the mesh family.

The remarkable performance of adaptive FEM motivates the idea of constructing finite element partitions and basis functions directly through neural networks, where the adaptive process of finding optimal basis functions is embedded and optimized within the network training. To this end, we utilize the ReLU activation function, which serves as a fundamental building block in designing such FEM-inspired neural architectures.

2.3 ReLU functions

The ReLU function, defined as $\sigma(x) = \max\{x, 0\}$, is a widely used activation function for introducing nonlinearity in neural networks. A notable feature of ReLU is its ability to generate piecewise linear functions. Consider the interval $[0, 1]$ partitioned by a mesh $x_0 = 0 < x_1 < \dots < x_n = 1$. Given prescribed slopes k_i on each subinterval $[x_{i-1}, x_i]$ for $i = 1, \dots, n$, the corresponding piecewise linear function can be expressed as

$$f(x) = k_1 \sigma(x) + \sum_{j=1}^{n-1} (k_{j+1} - k_j) \sigma(x - x_j). \quad (4)$$

This representation highlights the flexibility of ReLU in constructing piecewise linear functions and serves as the motivation for employing ReLU to build finite element basis functions. In fact,

this connection was rigorously studied in [24], showing that continuous piecewise linear (CPWL) finite element functions can be represented exactly by deep ReLU neural networks. Moreover, they establish lower bounds on the depth required (e.g. at least two hidden layers for $d \geq 2$ with d denoting the spatial dimension) and discuss how this equivalence underlies the expressive power of ReLU networks in approximating classical finite element spaces.

2.4 FEM basis construction

We construct the network following the structure in Equation (1). Instead of employing a deep architecture (which typically requires a large number of trainable parameters) to build the basis functions $\phi_k(x)$, we consider constructing the basis ϕ_k using FEM hat functions. Specifically, a hat function (illustrated in Figure 1) is defined as

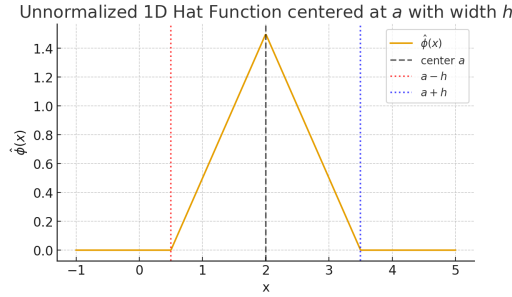


Figure 1: A hat function demonstration.

$$p_{a,h}(x) = \begin{cases} x - (a - h), & a - h < x \leq a, \\ -x + (a + h), & a < x < a + h, \\ 0, & \text{otherwise.} \end{cases}$$

By Equation (4), this hat function $p_{a,h}$ can be represented exactly by the ReLU functions as

$$p_{a,h}(x) = \sigma(x - (a - h)) - 2\sigma(x - a) + \sigma(x - (a + h)).$$

Notably, to represent each hat function, only 2 parameters are used. We then formulate our approximation as

$$G_\theta(u)(x) = \sum_{k=1}^N \hat{c}_{k,\theta_k}(\hat{u}) p_{a_k,h_k}(x) = \sum_{k=1}^N \hat{c}_{k,\theta_k}(\sigma(x - (a_k - h_k)) - 2\sigma(x - a_k) + \sigma(x - (a_k + h_k))),$$

where $\hat{c}_{k,\theta_k}$ denotes the neural network architecture with parameters θ_k , θ denotes the collection of all θ_k , and the trainable parameters a_k, h_k used to construct the hat function basis.

As a result, the FEM basis can be constructed exactly as a combination of ReLU functions, introducing no approximation error in the basis construction process. During network training, the centers a_k and widths h_k are treated as learnable parameters, effectively mimicking the adaptive refinement process in FEM and enabling the network to accurately capture the output functions (e.g., the PDE solutions). Examples illustrating this behavior can be found in Sections 3.4, 3.5, 3.6, and 3.7, where the proposed method successfully captures bumps or sharp peaks (singular features) in the solution landscapes, demonstrating its capability to represent non-smooth behaviors effectively.

3 Numerical examples

In this section, we present a series of numerical examples to demonstrate the performance of the proposed method. We consider seven PDEs that exhibit a wide range of physical and mathematical properties, including gradient-flow dynamics, dispersive behavior, and nonlinear advection–diffusion. Specifically, the following PDEs are studied: the Allen–Cahn (AC) equation, the Cahn–Hilliard (CH) equation, the Fokker–Planck (FP) equation, the Aggregation–Diffusion (AD) equation, the Keller–Segel (KS) equation, the Korteweg–de Vries (KdV) equation, and the viscous Burgers equation. Among these, the Allen–Cahn, Cahn–Hilliard, Fokker–Planck, Aggregation–Diffusion, and Keller–Segel equation are examples of gradient-flow systems that evolve toward equilibrium under an energy-dissipation principle, whereas the KdV equation represents a dispersive system, and the viscous Burgers equation is a dissipative nonlinear advection–diffusion model. To show the adaptivity of the methods and later shows that our methods mimic the adaptive FEM, in FK, AD, KS and KdV example, we created either bumps, shocks, or fast decay in the solution landscape, which is capture by our method. Furthermore, to demonstrate the adaptivity of the proposed method—and to show that it effectively mimics the adaptive FEM, which explains its high numerical accuracy and efficiency—we design experiments in the FK, AD, KS, and KdV examples where the solution landscapes exhibit bumps, shocks, or rapid decays.

To fairly evaluate the proposed method, we compare our method against the standard DeepONet, which can be viewed as using non-polynomial global bases [52]. For a fair comparison, the coefficient branch network $\hat{c}_{k,\theta_k}(\cdot)$ structure is kept identical between our method and the standard DeepONet. However, the DeepONet employs a deep (and, for additional comparison with our shallow FEM network, a shallow) fully connected architecture for its trunk network, while our method uses the proposed shallow structures to construct the FEM local polynomial. The number of basis functions N is also kept the same across all comparison tests. In addition, we include comparisons with the POD-based approach, where the POD modes of the output functions are used as the trunk network in examples where such a method is applicable.

3.1 Summary of Results

Accuracy and efficiency. Owing to the use of hat functions in constructing the basis, the proposed FEM method FERN substantially reduces the number of training parameters. Despite having significantly fewer parameters than DeepONet with an identical structure (i.e., the same number of basis functions N and the same coefficient networks c_{k,θ_k}), we observe comparable accuracy across all examples, and in 10 out of the 13 comparison tests, the FERN achieves improved accuracy with smaller prediction variance across all testing samples. In contrast, when a two-layer fully connected network is used as the DeepONet trunk basis (the smaller DeepONet), the error increases significantly, resulting in worse performance than the proposed method. This highlights the influence of the trunk network’s depth and parameter count. However, our method is able to reduce the number of parameters while simultaneously improving accuracy.

Adaptivity. For all examples, we observed that the learned FEM basis exhibits adaptivity. Particularly, **(1)** for the Fokker–Plank (Section 3.4), aggregation–diffusion (Section 3.5), Keller–Segel (Section 3.6), and KdV (Section 3.7) equation, where we intentionally introduced localized structures (bumps or peaks), the learned FEM basis functions adaptively concentrate around these regions in order to accurately capture the local features. Meanwhile, **(2)** for the Allen–Cahn problem (Section 3.2) and the Cahn–Hilliard equation (Section 3.3), the distinctive features of the solution landscapes, such as shocks, rapid growth, and rapid decay, are more evenly distributed across the domain among different training and testing samples. Conse-

quently, although the hat basis functions h are initialized with relatively large (global) supports, they shrink to smaller supports after training, exhibiting localized behavior; at the same time, their centers remain approximately evenly distributed after learning, enabling the model to capture these features effectively throughout the domain. These results show that our method effectively mimics the adaptive FEM and numerically justify its accuracy and efficiency.

3.2 Allen-Cahn Equation

In this section, we consider the Allen-Cahn equation,

$$u_t - \epsilon^2 u_{xx} + F'(u) = 0, x \in [0, 1], t \in [0, 10], \quad (5)$$

where $F(u) = (u^2 - 1)^2/4$, and $\epsilon = 0.01$. The target operator is the mapping from the initial condition to the solution at the terminal time. We generate the initial condition as $\lambda \sin(2\pi x) + (1 - \lambda) \sin(6\pi(x - 0.5 + \mu))$, where λ and μ are free parameters. We will consider two sets of experiments with different free parameters settings.

3.2.1 One degree of freedom

In this set of experiments, we fix $\mu = 0.5$ and sample λ uniformly from $[0, 1]$. We plot two typical solutions in Figure 2, and present the results in Table 1.

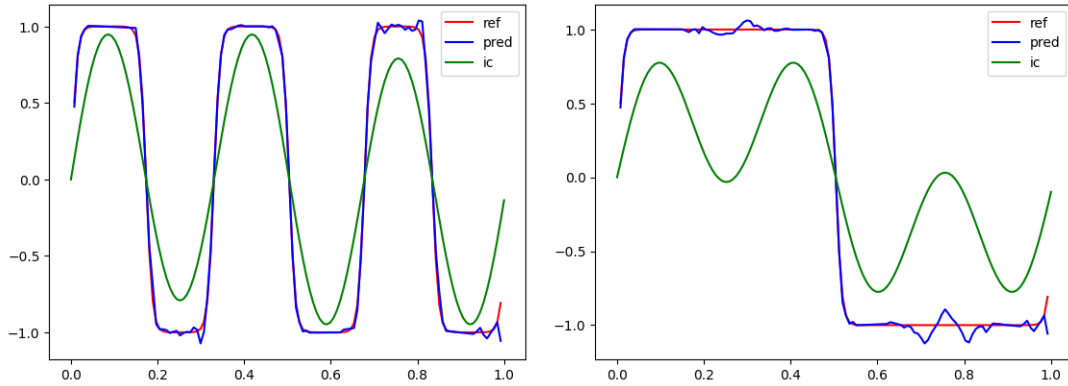


Figure 2: Demonstration of the predictions for the Allen Cahn example (one degree of freedom). The relative prediction error for left and right examples are 3.70% and 4.00% respectively. The average and variance of the relative error for 190 testing samples are $3.61\% \pm 1.3\%$.

	DeepONet-S	DeepONet-L	FERN	POD (same mesh)
Relative L^2 error	$55.76\% \pm 25.77\%$	$3.39\% \pm 1.94\%$	$3.61\% \pm 1.3\%$	$0.15\% \pm 0.17\%$
# Coefficient Parameters	19,200	19,200	19,200	19,200
# Basis Parameters	4,200	54,700	80	0
# Total Parameters	23,400	73,900	19,280	19,200

Table 1: Comparison of different models for the Allen–Cahn example. All models use a shared branch network architecture ($\hat{c}_k$) and employ 40 basis functions. For DeepONet, two configurations are considered: (1) DeepONet-S, which uses 2 fully connected layers to construct the basis functions, and (2) DeepONet-L, which uses 7 layers. The POD basis achieves the best accuracy among all methods. However, its use requires that both training and testing be performed on the same mesh, and that all training output functions be evaluated on this mesh. In contrast, the other models are evaluated on a denser mesh than used during training, demonstrating greater flexibility. Additionally, the performance of the POD method drops significantly in the following examples, where more degrees of freedom are used in data generation, resulting in a higher-rank dataset.

Setting details. We generate a total of 167 input functions (initial conditions, ICs) along with their corresponding solutions for training purposes. For each output function, 100 evaluation points are uniformly sampled from the spatial domain. Each input function is discretized using 22 uniformly spaced sensors, corresponding to a mesh size of 22. The model is trained using a cosine annealing learning rate schedule. All neural network architecture are constructed using 40 basis functions (DeepONet global basis, FEM basis or POD basis). Each coefficient branch network follows a fully connected architecture of size $22 \times 20 \rightarrow 20 \times 1$, with the hyperbolic tangent (Tanh) function as the activation. The centers of the FEM basis functions are initialized uniformly over the interval $[0, 1]$, with a fixed support size $h = 0.05$. A histogram of the learned FEM basis centers and h after training is presented in Figure 3.

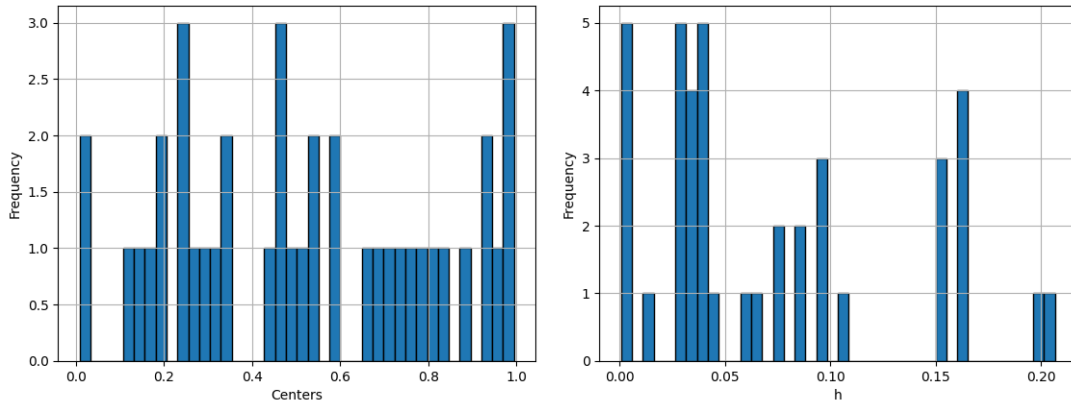


Figure 3: Allen-Cahn equation with one degree of freedom. Left: Histogram of all learned FEM basis centers a_k , which are initially uniformly distributed over $[0, 1]$. Right: Histogram of all learned support sizes h_k , all of which are initialized to 0.05. Since the distinctive features of the solution landscapes, such as shocks, bumps, rapid decay, and rapid growth, are evenly distributed across the domain for different samples (see Figure 2), most learned hat functions shrink to smaller supports after training (right panel), exhibiting localized behavior. Meanwhile, their centers remain evenly distributed to effectively capture these features across various samples (left panel). This behavior is consistently observed for other numbers of basis functions tested; see Figure 7 for an illustration.

Results analysis. As shown in Table 1, the proposed FERN method achieves results comparable to the standard DeepONet while using significantly fewer trainable parameters. This demonstrates the effectiveness of the proposed approach: when a shallow network is designed to construct the FEM basis, a limited number of parameters is sufficient to achieve low relative prediction error.

To further verify this observation, we also decrease the number of layers used in the trunk (basis) network. As shown in Table 1, when the number of fully connected layers is reduced to two, the prediction error increases significantly. Although the POD method yields the smallest error, it relies on stronger assumptions when performing POD on the output functions, for instance, all outputs must be discretized on the same mesh. This restriction makes the POD-based approach a fixed-grid-to-fixed-grid mapping, which effectively corresponds to learning a function rather than an operator. Moreover, when the number of free parameters increases to two, the discretized output functions exhibit higher rank, causing the POD method to produce a larger error (7.76%), which exceeds the 4.44% error obtained using the proposed FEM basis. See the next section and Table 2 for further details.

3.2.2 Two degrees of freedom

In this set of experiments, λ and μ are both sampled from $[0, 1]$ uniformly; as a results, the solutions present more variability and make the problem more challenging. We plot four typical solutions in Figure 4, and the results in Table 2.

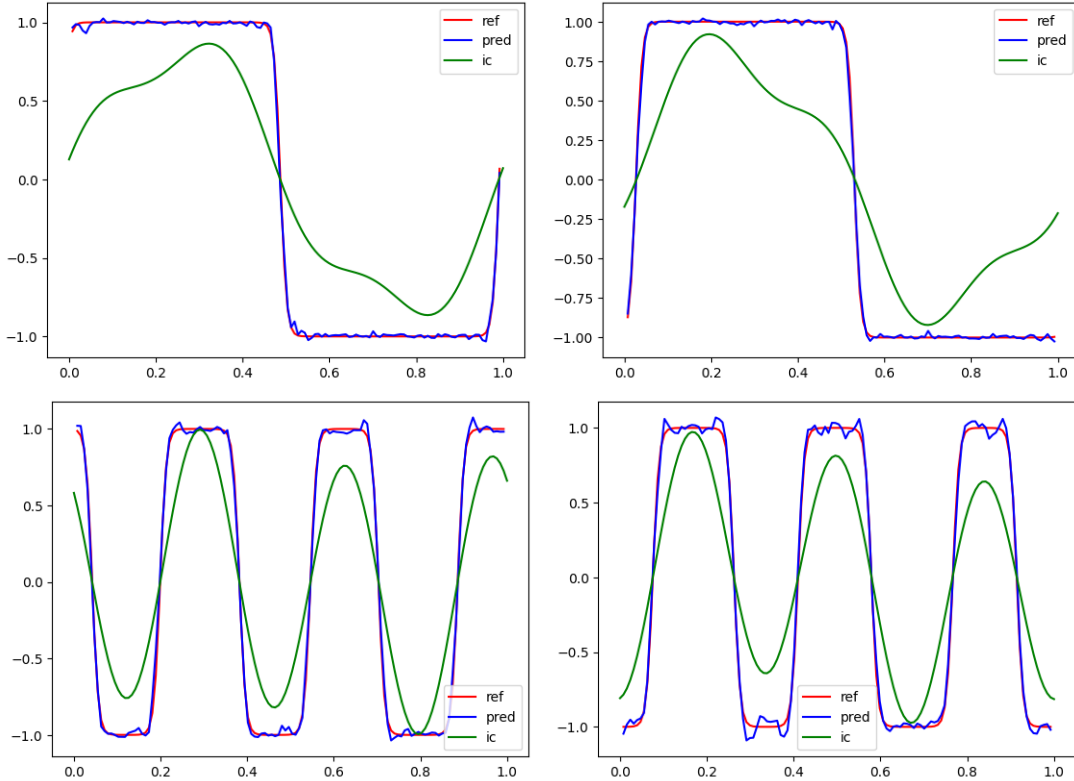


Figure 4: Predicted solutions for the Allen–Cahn equation with two degrees of freedom. The relative prediction errors for the four cases (displayed from left to right, top to bottom) are 2.21%, 2.23%, 3.90%, and 4.76%, respectively.

	DeepONet	FERN	POD (same mesh)
Relative L^2 error	$8.65\% \pm 4.08\%$	$4.44\% \pm 2.2\%$	$7.76\% \pm 5.52\%$
# Coefficient Parameters	38,400	38,400	38,400
# Basis Parameters	58,700	160	0
# Total Parameters	97,100	38,560	38,400

Table 2: Comparison of different models for the Allen–Cahn example with two degrees of freedom. All models employ the same branch networks architecture ($\hat{c}_k$) and approximate the output functions using 80 basis functions. We additionally test DeepONet with a two-layer trunk network; although this reduces the parameter count to 46,600, the relative error stabilizes at 63.25%, which is substantially larger than that obtained with FERN. Additionally, compared to the previous examples where a single degree of freedom was used to generate the training data, the accuracy of the POD method drops as the rank of the dataset increases.

Setting details. We generate a total of 250 input functions (initial conditions, ICs) along with their corresponding solutions for training. Each output function is evaluated at 100 uniformly sampled points from the spatial domain, while each input function is discretized using 22 uniformly spaced sensors, corresponding to a mesh size of 22. All models are trained for 2,000 epochs using a cosine annealing learning rate schedule. The neural network architectures across all models employ 80 basis functions, either the proposed FEM basis, the POD basis, or the standard DeepONet basis. Each coefficient branch network adopts a fully connected architecture with layer sizes $22 \times 20 \rightarrow 20 \times 1$. For DeepONet, we report results using the ReLU activation function, which yielded a better accuracy compared to Tanh activation. For the FEM and POD models, the coefficient networks use the Tanh activation function for better accuracies. Notably, FERN exhibits robustness to the choice of activation: using ReLU, it achieves a relative prediction error of $4.45\% \pm 2.0\%$. The centers of the FEM basis functions are initialized uniformly over the interval $[0, 1]$, with a fixed support size $h = 0.05$. A histogram of the learned FEM basis centers and h after training is presented in Figure 5.

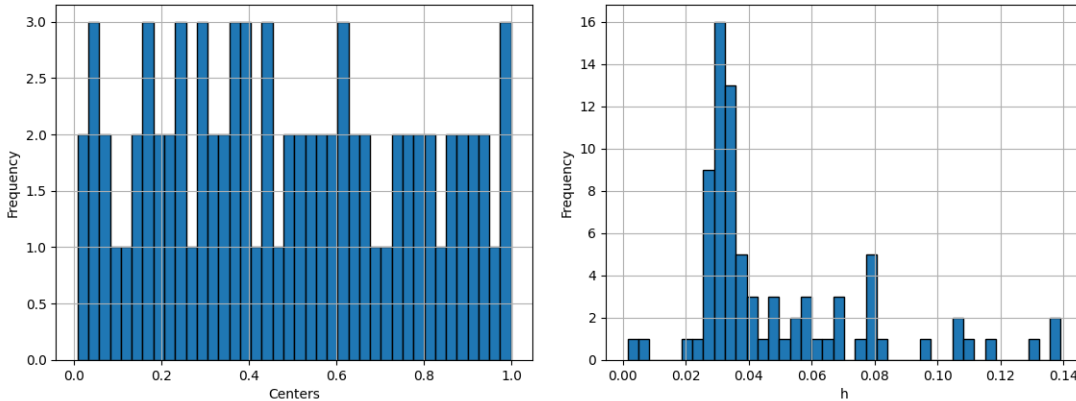


Figure 5: Allen-Cahn equation with two degrees of freedom. Left: Histogram of all learned FEM basis centers a , which are initially uniformly distributed over $[0, 1]$. Right: Histogram of all learned support sizes h , all of which are initialized to 0.05. Since the distinctive features of the solution landscapes, such as shocks, bumps, rapid decay, and rapid growth, are evenly distributed across the domain for different samples (see Figure 4), most learned functions shrink to smaller supports after training (right picture), exhibiting localized behavior. However, their centers remain evenly distributed to effectively capture these features across various samples (left picture). This behavior is consistently observed across different numbers of basis functions N tested (see Figure 7).

Analysis of results. From Table 2, it can be observed that the proposed method achieves the highest prediction accuracy with the smallest variance. Although the POD non-trainable basis uses slightly fewer trainable parameters (38,400 vs. 38,560), it fails to handle output discretizations on different meshes and yields a larger error due to the increased number of degrees of freedom used to generate the input functions. These results further demonstrate the accuracy and efficiency of the proposed method.

A study on the number of basis. We investigate the relationship between prediction error and the number of basis networks (corresponding to the mesh size in the output space) for the proposed FEM-based methods. Specifically, we vary the number of learned FEM basis functions from 20 to 100 and train a separate model for each setting. All models are initialized with uniformly distributed centers in the domain $[0, 1]$, and each basis function is assigned a fixed size of $h = 0.05$. The resulting error decay behavior is presented in Figure 6. Roughly, first order convergence is observed with respect to the number of basis.

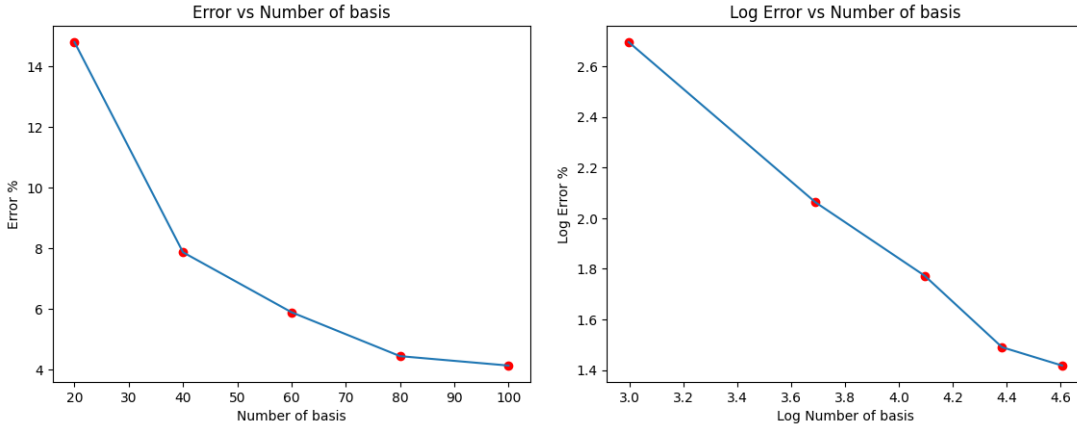


Figure 6: Allen-Cahn equation with two degrees of freedom and various number of basis. Left: Error decay with respect to the number of learned hat function basis. Right: decay in the log scale.

In addition, we present the learned mesh size distributions for models with 20, 40, 60, 80, and 100 basis functions in Figure 7. From the figure, it can be observed that the learned mesh sizes gradually shift toward zero as the number of basis functions decreases. This indicates that, when fewer basis functions are used, the model tends to learn basis functions with larger support.

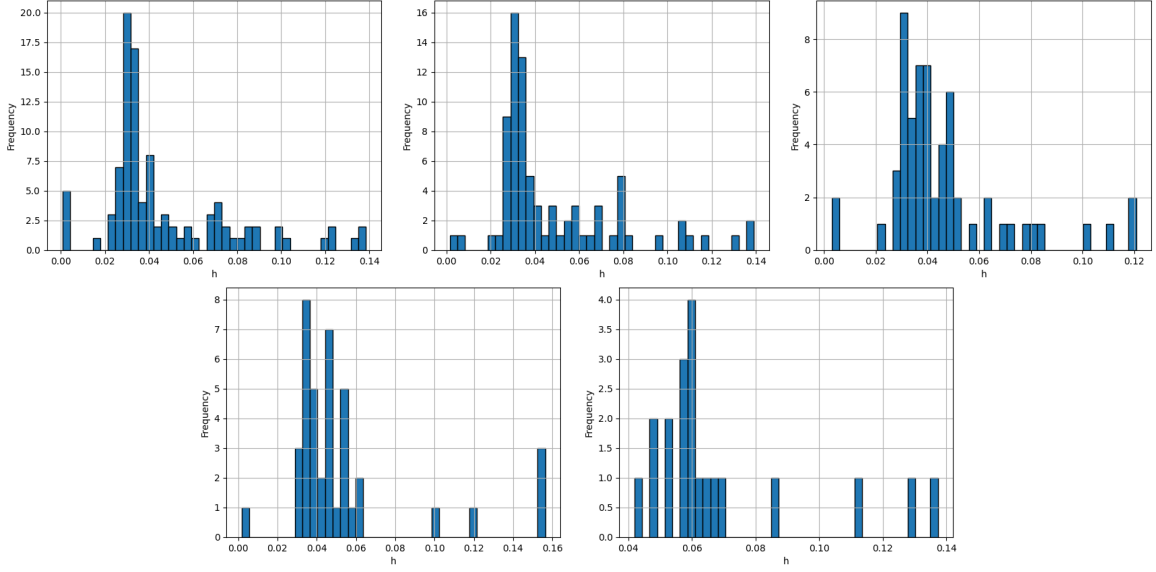


Figure 7: Allen-Cahn equation with two degrees of freedom and various number of basis. Distributions of the learned h for models with different numbers of basis functions (20, 40, 60, 80, and 100). Each experiment is initialized with $h = 0.05$. As shown, the learned h values shift across all cases, demonstrating the adaptivity of the proposed method.

3.3 Canh-Hilliard Equation

In this section, we study the Canh-Hilliard equation,

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left(M \frac{\partial}{\partial x} \left(-\epsilon^2 \frac{\partial^2 u}{\partial x^2} + F(u) \right) \right), x \in [0, 1], t \in [0, 10],$$

where $F(u) = (u^2 - 1)^2/4$, and $M = 0.01$. The target operator is the mapping from the initial condition to the solution at the terminal time. We generate the initial condition by sampling free parameter λ and μ for $\lambda \sin(2\pi x) + (1 - \lambda) \sin(6\pi(x - 0.5 + \mu))$. We will consider two sets of experiments with different free parameters settings.

3.3.1 One degree of freedom

In this set of experiments, we set the $\mu = 0.5$, and sample λ uniformly from $[0, 1]$. We display two typical solutions in Figure 8 and the numerical results in Table 3.

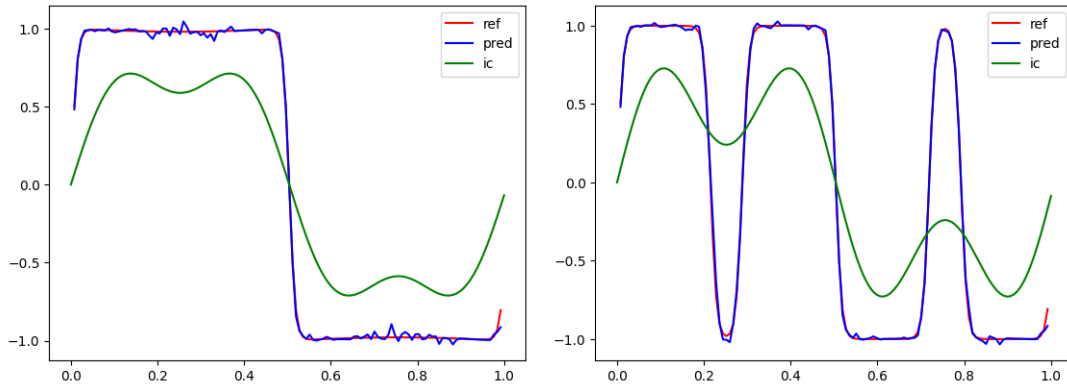


Figure 8: Demonstration of the predictions for the Cahn-Hilliard equation with one degree of freedom. The relative prediction error for left and right examples are 2.62% and 2.19% respectively. The average relative error for 190 testing samples is $2.62\% \pm 0.41\%$.

	DeepONet	FERN	POD
Relative L^2 error	$2.91\% \pm 1.07\%$	$2.62\% \pm 0.41\%$	$0.38\% \pm 0.24\%$
# Coefficient Parameters	28,800	28,800	28,800
# Basis Parameters	56,700	120	0
# Total Parameters	85,500	28920	28,800

Table 3: Comparison of different models for the Cahn-Hilliard equation with one degree of freedom. All models share the same branch network architecture and have 60 basis functions (DeepONet global learnable basis, FEM learnable local basis, and POD basis). We also test a smaller DeepONet structure with a two-layer trunk network while keeping the branch network unchanged. Although this reduces the number of parameters to 35,000, the relative error increases to $28.44\% \pm 11.07\%$.

Setting details. We generate a total of 250 input functions (initial conditions, ICs) along with their corresponding solutions for training. Each output function is evaluated at 100 uniformly sampled points from the spatial domain, while each input function is discretized using 22 uniformly spaced sensors, corresponding to a mesh size of 22. All models are trained for 2,000 epochs using a cosine annealing learning rate schedule. The neural network architectures across all models employ 60 basis functions—either the proposed FEM basis, the POD basis, or the standard DeepONet basis. Each coefficient branch network adopts a fully connected architecture with layer sizes $22 \times 20 \rightarrow 20 \times 1$ and are activated by ReLU. For the DeepONet and FEM models, the basis networks use the Tanh activation function. The centers of the FEM basis functions are initialized uniformly over the interval $[0, 1]$, with a fixed support size $h = 0.05$. A histogram of the learned FEM basis centers and h after training is presented in Figure 9.

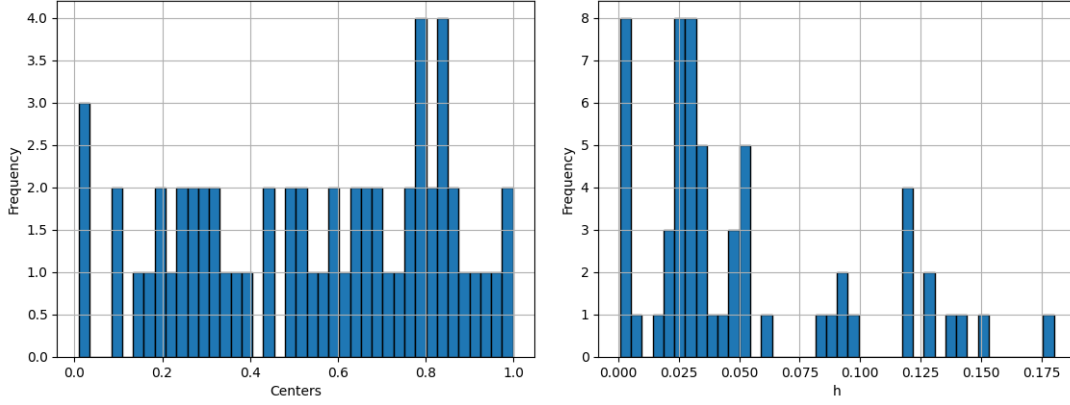


Figure 9: Cahn-Hilliard equation with one degree of freedom. Left: Histogram of all learned FEM basis centers, which are initially uniformly distributed over $[0, 1]$. Right: Histogram of all learned support sizes h , all of which are initialized to 0.05. Since the distinctive features of the solution landscapes, such as shocks, bumps, rapid decay, and rapid growth, are evenly distributed across the domain for different samples (see Figure 8), most learned functions shrink to smaller supports (0.025) after training (left picture), exhibiting localized behavior. However, their centers remain almost evenly distributed to effectively capture these features across various samples (right picture).

Analysis of the results. From Table 3, the proposed method achieves a lower error ($2.62\% < 2.91\%$) compared to the standard DeepONet with a deep network-constructed basis. At the same time, the number of trainable parameters is significantly smaller, as we use only 2×60 parameters (two parameters per basis function) to construct a shallow network for building the FEM basis. Although the POD method outperforms the proposed method in terms of accuracy, it relies on non-trainable POD bases, requiring strong assumptions before application. For instance, all output functions must be discretized on the same mesh, which greatly limits the flexibility of the training data and restricts the POD-based operator learning to mappings from fixed grids to fixed grids, essentially resembling standard function learning rather than general operator learning. Additionally, when we increase the number of degrees of freedom used to generate the training input functions, the output functions exhibit greater variability (i.e., higher rank). Consequently, the accuracy of the POD method drops significantly, becoming worse than that of the proposed method. See the results of the next experiment in Table 4 for details.

Lastly, we observe the basis localization during network training. As shown in Figure 9, all FEM hat functions are initialized with a uniform width of $h_k = 0.05$. However, after training, most functions shrink to narrower supports, exhibiting localized behavior. Since the special features of the output functions—such as sharp decays or mild shocks (see Figure 8 for illustration)—are relatively evenly distributed across the domain, the centers a_k of the learned functions do not shift significantly. In contrast, for examples where shocks are concentrated in specific regions of the domain, we observe clear localization of the functions around those regions. See FK, KS, AD, KdV sections for examples.

3.3.2 Two degrees of freedom

In this set of experiments, we set λ and μ both are sample uniformly from $[0, 1]$. We plot four typical solutions in Figure 10 and present the numerical results in Table 4.

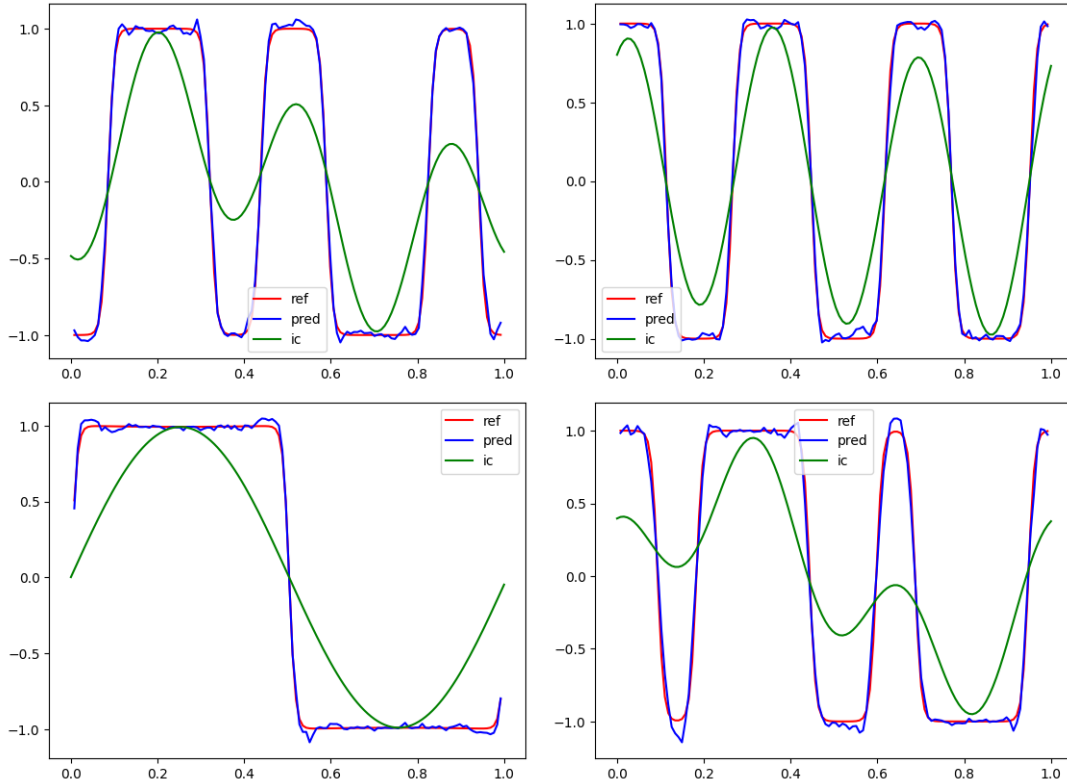


Figure 10: Predicted solutions for the Cahn-Hilliard equation with two degrees of freedom. The relative prediction errors for the four cases (displayed from left to right, top to bottom) are 4.09%, 4.29%, 2.59%, and 6.89%, respectively.

	DeepONet	FERN	POD
Relative L^2 error	$14.68\% \pm 4.19\%$	$5.59\% \pm 2.2\%$	$7.22\% \pm 3.41\%$
# Coefficient Parameters	120,000	120,000	61,440
# Basis Parameters	75,700	500	0
# Total Parameters	195,700	120,500	61,440

Table 4: Comparison of different models for the Cahn-Hilliard equation with two degrees of freedom. All models share the coefficient networks; the DeepONet and FERN network have 250 learnable basis, while the POD network has 128 basis the largest number of basis due to the discretization size.

Setting details. We generate a total of 1,000 input functions (initial conditions, ICs) along with their corresponding solutions for training. Each output function is evaluated at 100 uniformly sampled points from the spatial domain, while each input function is discretized using 22 uniformly spaced sensors, corresponding to a mesh size of 22. All models are trained for 2,000 epochs using a cosine annealing learning rate schedule. The neural network architectures across all models employ 250 basis functions, either the proposed FEM basis, the POD basis, or the standard DeepONet basis. Each coefficient branch network adopts a fully connected architecture with layer sizes $22 \times 20 \rightarrow 20 \times 1$ and are activated by ReLU. For the DeepONet and FEM models, the basis trunk networks use the Tanh activation function. The centers of the FEM basis functions are initialized uniformly over the interval $[0, 1]$, with a fixed support size $h = 0.05$. A histogram of the learned FEM basis centers and h after training is presented

in Figure 11.

Analysis of the results. Similar to our observations for the Allen-Cahn equation, the proposed method achieves the highest prediction accuracy with the smallest variance.

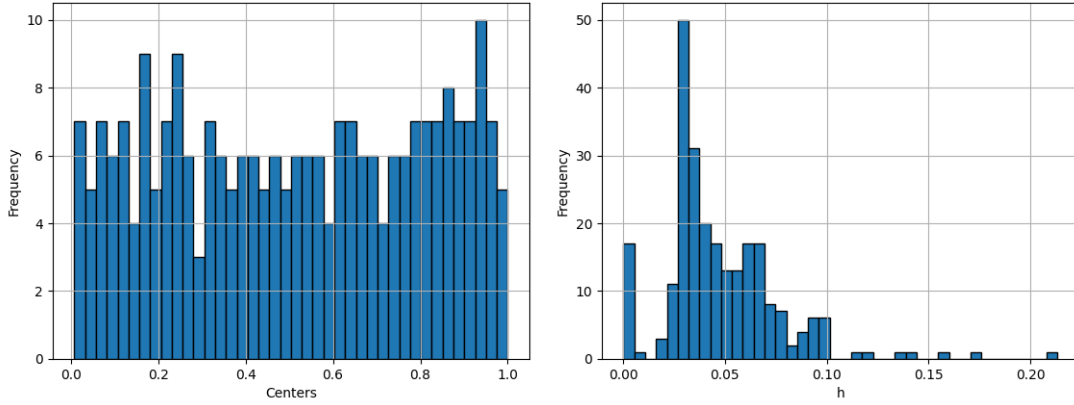


Figure 11: Cahn-Hilliard equation with two degrees of freedom. Left: Histogram of all learned FEM basis centers, which are initially uniformly distributed over $[0, 1]$. Right: Histogram of all learned support sizes h , all of which are initialized to 0.05. Most learned FEM hat functions shrink to smaller supports after training (left picture), exhibiting localized behavior. However, their centers remain evenly distributed to effectively capture these features across various samples (right picture).

3.4 Fokker-Planck Equation

In this section, we consider the Fokker-Planck (FK) equation,

$$u_t - (u(\log(u) + \cos(2\pi x))_x)_x = 0, x \in [0, 1], t \in [0, 0.1]. \quad (6)$$

The target operator is the mapping from the initial condition to the solution at the terminal time. We generate the initial condition by sampling free parameters c_0 and c_1 . Specifically, the initial condition has the form $u_0(x) = c_1 \exp(-100(x - c_0)^2) + 10^{-3}$, where c_0 is uniformly drawn from $[0.3, 0.7]$ and c_1 is from $[1, 10]$. We plot two typical solutions in Figure 12. Notably, the terminal solutions exhibit a bell-shaped profile with a peak centered around 0.5 across different initial conditions. This consistent behavior provides an opportunity to evaluate the adaptivity of the learning process, that is, the model allocates more basis functions to the central region of the domain, where the solution is most concentrated, rather than to the flat tails.

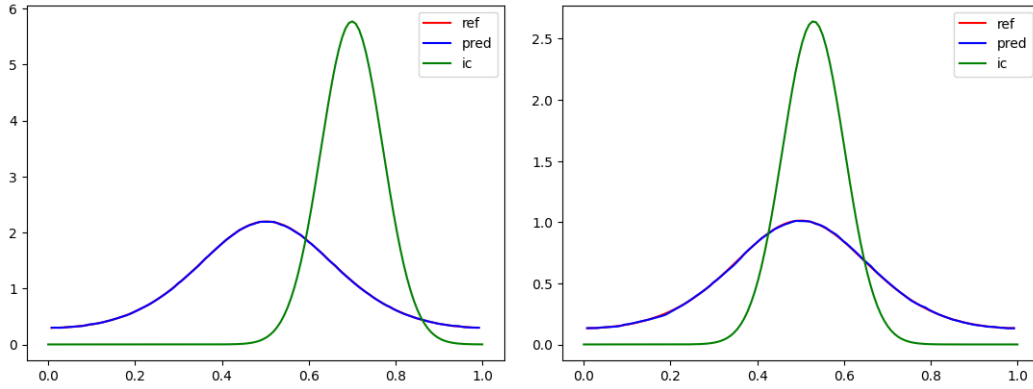


Figure 12: Fokker-Plank equation solution illustrations. The predictions are obtained using the proposed method with 30 basis functions, yielding errors of 0.28% and 0.59% respectively.

We present the numerical results in Table 5 and Table 6. Two sampling scenarios are considered: (1) In the first case, the terminal solutions corresponding to each initial condition are uniformly sampled over the entire domain with a mesh size of 64. This setting permits the use of the POD method to construct basis functions for prediction. (2) In the second case, one-third of the terminal solutions are sampled uniformly only over the interval $[0, 0.5]$, another third over $[0.5, 1]$, and the remaining third uniformly over the full interval $[0, 1]$. Under this non-uniform sampling scheme (different meshes for different output functions), POD cannot be applied directly, necessitating the use of learnable basis functions. In both cases, our method achieves comparable or superior prediction accuracy while using significantly fewer trainable parameters. For example, with 30 basis functions, FERN requires only 14,460 trainable parameters, whereas the standard DeepONet uses 68,100.

	DeepONet (30)	2-layer DeepONet (30)	FERN (30)	DeepONet (10)	FERN (10)
Relative L^2 error	$1.16\% \pm 1.36\%$	$4.23\% \pm 3.49\%$	$1.10\% \pm 0.58\%$	$1.28\% \pm 1.59\%$	$1.15\% \pm 0.63\%$
# Coefficient Parameters	14,400	14,400	14,400	4,800	4,800
# Basis Parameters	53,700	3,200	60	51,700	20
# Total Parameters	68,100	17,600	14,460	56,500	4,820

Table 5: Comparison of different models for the Fokker-Plank equation (the numbers 30 and 10 denote the number of basis N used) on uniform grids. The terminal solutions are uniformly sampled over the interval $[0, 1]$ for all samples. Notably, all DeepONet models achieve accurate predictions when using a deep basis network; however, their performance deteriorates when the basis network is reduced to only two layers, see the third column. In contrast, FERN maintain high accuracy while requiring substantially fewer trainable parameters.

	DeepONet (30)	2-layer DeepONet (30)	FERN (30)	DeepONet (10)	FERN (10)
Relative L^2 error	$1.08\% \pm 0.94\%$	$6.93\% \pm 4.34\%$	$1.48\% \pm 0.9\%$	$1.26\% \pm 1.27\%$	$1.35\% \pm 0.83\%$
# Coefficient Parameters	14,400	14,400	14,400	4,800	4,800
# Basis Parameters	53,700	3,200	60	51,700	20
# Total Parameters	68,100	17,600	14,460	56,500	4,820

Table 6: Comparison of different models for the Fokker-Plank equation (the numbers 30 and 10 denote the number of basis N used) on non-uniform grids. In this setting, the terminal solutions are not uniformly sampled from the full domain $[0, 1]$ across all samples. Specifically, one-third of the samples are drawn from the interval $[0, 0.5]$, another one-third from $[0.5, 1]$, and the remaining one-third from the full interval $[0, 1]$. This non-uniform sampling pattern prevents the direct application of POD-based basis functions. Similar to the uniform sampling results shown in Table 5, the performance of the standard DeepONet deteriorates when the number of layers in the basis network is reduced to two, whereas FERN maintains accurate predictions while using significantly fewer trainable parameters.

Detailed settings. We generate a total of 42 input functions (initial conditions, ICs) along with their corresponding solutions for training. Each input function is discretized using 22 uniformly spaced sensors, corresponding to a mesh of size 22. The corresponding output functions are evaluated at 64 spatial points, yielding a total of 42×64 training samples. Two different samples are tested, refer to Table 5 and Table 6 for details. All FEM-based models are trained for 2,000 epochs using a cosine annealing learning rate scheduler, while DeepONet models are trained with more epochs to ensure convergence to a stable error level. For the standard DeepONet, the basis network follows a fully connected feedforward architecture with the structure $1 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times K$, where each hidden layer is followed by a tanh activation, and the final layer is linear. All layers include bias terms except for the output layer. We also evaluate a variant of DeepONet with only two layers in the basis network, where the architecture is $1 \times 100 \rightarrow 100 \times 100$, and the activation function used is ReLU.

Accuracy. On uniform grids, the performance of FERN is similar to that of DeepONet, but with a smaller variance, see Table 5. From Table 6, on nonuniform grids, when 30 basis functions are used, FERN achieves a prediction error of 1.48%, while the standard DeepONet with a deep trunk basis attains an error of 1.08%. Although the proposed method does not yield a smaller average error in this example, it exhibits a smaller prediction variance (0.9% vs 0.94%) and requires significantly fewer trainable parameters (14,460 vs. 68,100), as it employs a two-layer shallow structure to construct the adaptive FEM basis.

To further demonstrate that not all shallow-structured basis networks are effective, we also reduce the number of fully connected layers in the standard DeepONet to two, which decreases the number of trainable parameters to 17,600. However, despite having more parameters than the proposed method (17,600 vs. 14,460), the resulting error is considerably larger (6.93% vs. 1.48%). This highlights the effectiveness of the learned adaptive FEM basis.

Adaptivity. As shown in Figure 12, all terminal solutions, represented in terms of the FEM basis, exhibit a bell-shaped profile with their peaks centered near the middle of the domain.

Consequently, we expect the learned basis functions to reflect this spatial concentration pattern. This behavior is indeed observed in our results. To further validate this hypothesis, we conduct experiments using a larger number of basis functions. In the left panel of Figure 13, we display the distribution of the learned FEM basis centers. The results indicate that a greater number of basis functions are concentrated near the center of the domain. At the same time, the right panel shows the average support width h of the learned hat functions, revealing that basis functions centered near the middle region of the domain tend to have wider supports compared to those located elsewhere. Since hat functions are composed of two linear segments with slopes 1 and -1 , a larger support (width h) corresponds to a basis function of greater magnitude. This property may enhance the network's ability to represent the bell-shaped terminal solutions effectively.

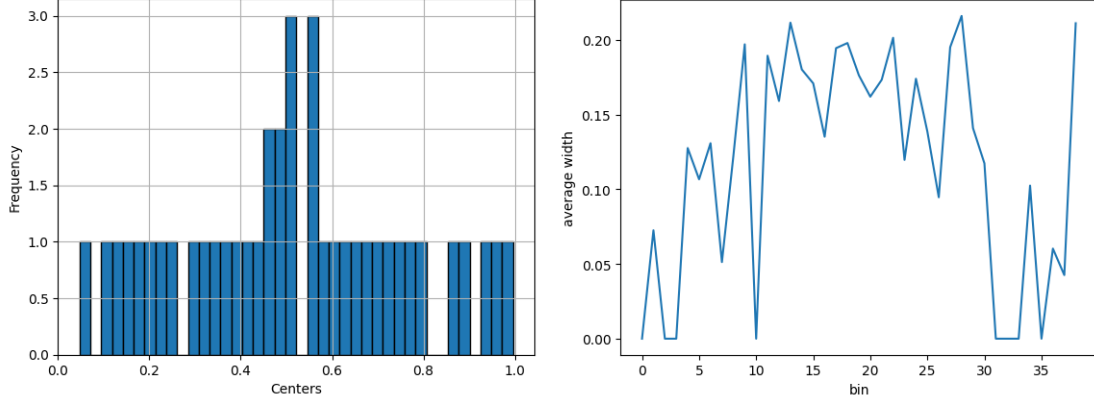


Figure 13: Fokker-Plank equation. The left panel shows the distribution of the learned basis function centers (40) across the domain. The right panel presents the average support width h of the learned hat functions within each subinterval (bin), where the average is computed over all basis functions whose centers fall within the respective bin. We observe that, after training, more basis functions are concentrated near $x = 0.5$, where a bump appears in the solution landscape (see Figure 12), demonstrating the adaptivity of the learning.

We also present the basis functions before and after training for the case with 10 basis functions in Figure 14. The initial centers are uniformly distributed across the domain with a fixed width $h = 0.05$.

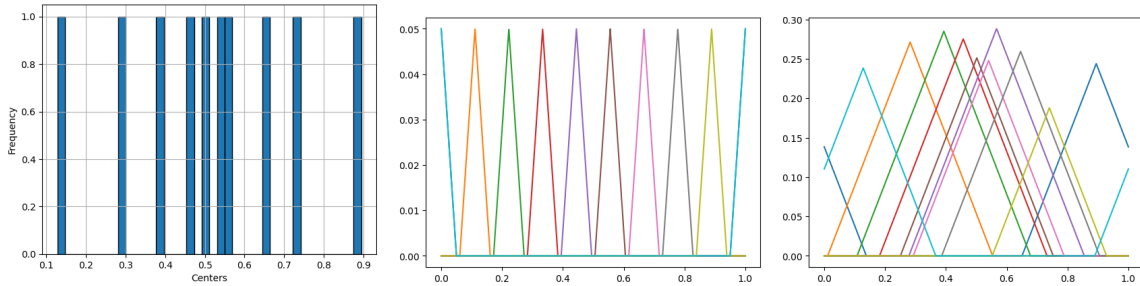


Figure 14: Fokker-Plank equation. Demonstration of the basis functions (10 in total). Left: distribution of basis centers after training. Middle: all basis functions before training. Right: all basis functions after training. We observe that, after training, more basis functions are concentrated near $x = 0.5$, where a bump appears in the solution landscape (see Figure 12), demonstrating the adaptivity of the learned FEM functions.

3.5 Aggregation-diffusion equation

In this section, we study the aggregation-diffusion equation,

$$u_t - (u(Du^{m-1} + W(x) * u)_x)_x = 0, \quad x \in [-6, 6], \quad t \in [0, 200],$$

where $D = 0.4, m = 2$, $W(x) = -\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$ with $\sigma = 1$, and $*$ is the convolution. The target operator is the mapping from the initial condition to the solution at the terminal time. We generate the initial condition by sampling the free parameter c_0 uniformly in $[1, 5]$ from

$$u_0(x) = \frac{c_0}{2\sqrt{2\pi}}(\exp(-(x - x_0)^2/2) + \exp(-(x + x_0)^2/2)).$$

We present two representative solutions of the system in Figure 15. Notably, despite differing initial conditions, the terminal states consistently exhibit a bell-shaped profile centered within the domain, with the solution remaining nearly flat near the boundaries. This example serves as a test of the adaptability of the learned basis functions. The coefficient networks (used in both the standard DeepONet and the FEM-based network) adopt a fully connected architecture with layer dimensions $22 \times 20 \rightarrow 20 \times 1$, and all layers are activated by the hyperbolic tangent (tanh) function.

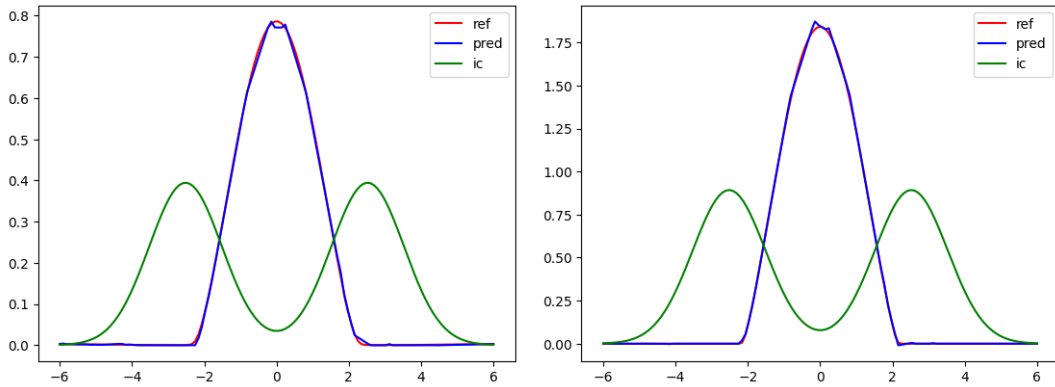


Figure 15: Demonstration of two solutions of the aggregation-diffusion equation. Notably, the terminal solutions corresponding to different initial conditions remain close to zero outside the central region of the domain, indicating strong spatial localization.

We present the numerical results in Table 7. The proposed methods can achieve similar accuracy while using fewer trainable parameters, given the same training and network settings. We give the numerical training details below.

	DeepONet (40)	FERN (40)	DeepONet (20)	FERN (20)
Relative L^2 error	$1.44\% \pm 2.69\%$	$1.08\% \pm 0.54\%$	$1.41\% \pm 2.74\%$	$1.51\% \pm 2.39\%$
# Coefficient Parameters	19,200	19,200	9,600	9,600
# Basis Parameters	54,700	80	52,700	40
# Total Parameters	73,900	19,280	62,300	9,640

Table 7: Comparison of different models for the aggregation-diffusion equation (the numbers 40 and 20 denote the number of basis N used). We also test a smaller DeepONet with a two-layer trunk network while keeping the branch network the same as in all other models. Although this reduces the total number of parameters to 23,400 (still more than the proposed method with 19,280 parameters), the relative error increases significantly to $61.63\% \pm 2.74\%$.

Setting details. We generate a total of 42 input functions (initial conditions, ICs) along with their corresponding solutions for training. Each input function is discretized using 22 uniformly spaced sensors, corresponding to a mesh of size 22. The corresponding output functions are evaluated at 64 uniformly sampled spatial points, yielding a total of 42×64 training samples. All models are trained for 2,000 epochs using a cosine annealing learning rate schedule. We test two different settings for the number of basis functions, as summarized in Table 7. In both settings, the coefficient network architecture (used in both the standard DeepONet and FERN) adopts a fully connected architecture with layer dimensions $22 \times 20 \rightarrow 20 \times 1$, and all layers are activated by the hyperbolic tangent (tanh) function.

The centers of the FEM basis functions are initialized uniformly over the input interval, with a fixed support width of $h = 0.05$. For the standard DeepONet, the basis network follows a fully connected feedforward architecture with the structure $1 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times K$, where each hidden layer is followed by a tanh activation, and the final layer is linear. All layers include bias terms except the output layer.

Accuracy. As shown in Table 7, the proposed FEM-basis method requires significantly fewer trainable parameters across both basis settings. When 20 basis functions are used, FERN attains a slightly higher average error than the standard DeepONet (1.51% vs. 1.41%). However, when the number of basis functions increases to 40, FERN achieves a smaller error (1.08% vs. 1.44%), a trend that is consistently observed in most of our experiments. Furthermore, even in cases where the average prediction error is higher, FERN consistently exhibits smaller variance in prediction errors, demonstrating its robustness and suggesting superior generalization performance in extrapolation scenarios.

Adaptivity. As shown in Figure 15, the terminal solutions exhibit a bell-shaped profile and decay rapidly outside the center of the domain. Since each hat basis function consists of two linear segments with slopes 1 and -1, a larger support width h corresponds to a basis function with greater spatial extent. Therefore, we expect the learned hat functions to have larger support when their centers are located near the center of the domain, and smaller support as their centers move toward the boundaries. This behavior is confirmed in our numerical experiments. To validate this hypothesis, we vary the number of basis functions from 10 to 40 and plot the distribution of the learned support widths h across uniformly divided subintervals of the domain. See Figure 16 for the results.

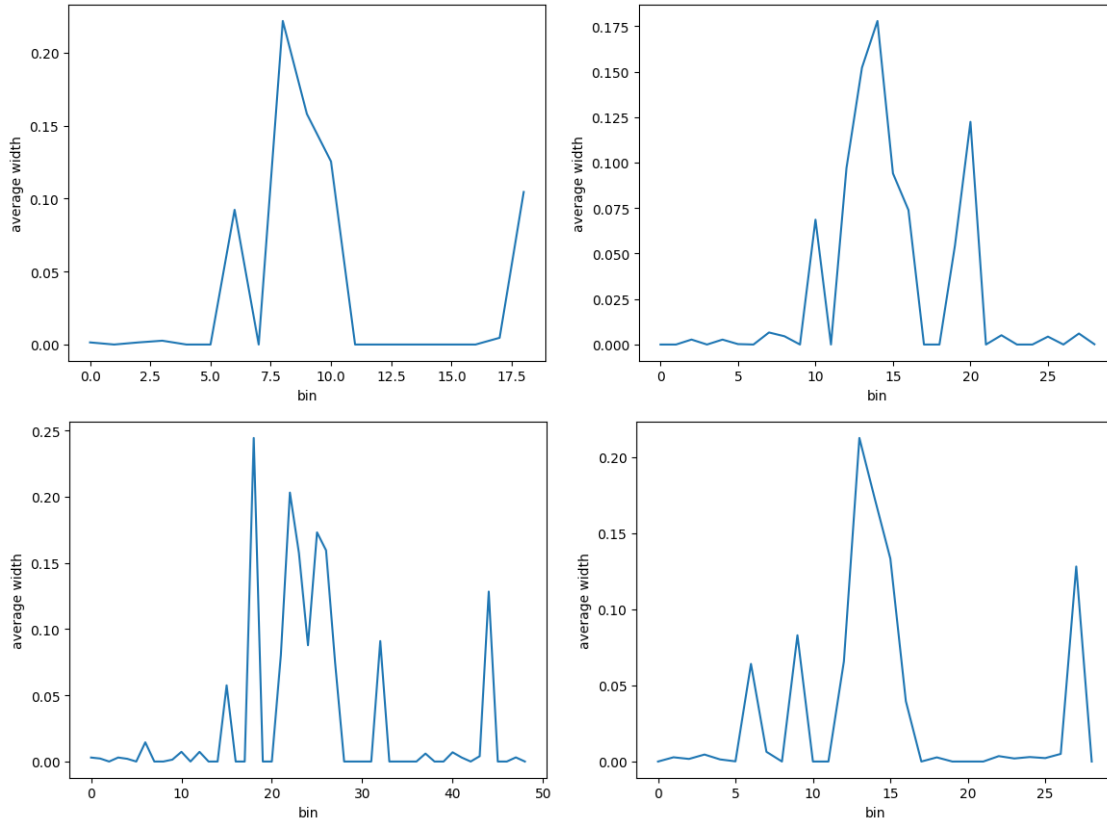


Figure 16: Aggregation-diffusion equation. Distribution of learned FEM basis support across the domain for different numbers of basis functions, ranging from 10 to 40 (ordered from left to right, top to bottom). The PDE domain is partitioned into uniform bins (subintervals), and for each bin, we compute the average support size h of the learned FEM basis functions whose centers a_k fall within that bin. The x-axis represents the bins from the left to the right boundary, while the y-axis indicates the corresponding average support. Note that larger support values result in wider hat functions, since the slopes of the underlying piecewise linear components are fixed at 1 and -1.

3.6 Keller-Segel Equation

In this section, we study the Keller-Segel (KS) equation,

$$\begin{cases} u_t - (u(D \log(u) - \chi v)_x)_x = 0, \\ -v_{xx} + v = u, \end{cases} \quad x \in [0, 1], t \in [0, 1.0]$$

where $D = 0.01, \chi = 5.0$. The target operator is the mapping from the initial condition to the solution at the terminal time. We generate the initial condition by sampling the free parameter c_0 uniformly in $[0.2, 0.8]$ from

$$u_0(x) = 1 + c_0 \sin(2\pi(x - 0.25)).$$

We present two typical solutions of the system in Figure 17. Notably, despite differing initial conditions, the terminal states consistently exhibit a bump profile centered within the domain, with the solution remaining nearly flat near the boundaries. This example serves as a test of the adaptability of the learned basis functions to capture a sharp gradient in the solution landscape.

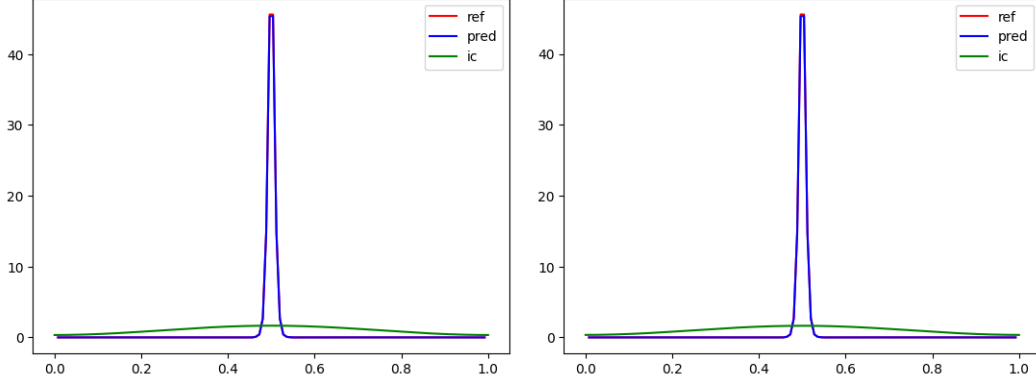


Figure 17: Demonstration of two solutions of the Keller-Segel equation. Notably, the terminal solutions corresponding to different initial conditions exhibit a bump close to $x = 0.5$.

We present the numerical results in Table 8. The proposed methods can achieve similar accuracy while using fewer trainable parameters, given the same training and network settings. We give the numerical training details below.

	DeepONet(40)	2-layer DeepONet	FERN (40)	FERN (60)
Relative L^2 error	$0.73\% \pm 0.05\%$	$5.97\% \pm 0.23\%$	$1.08\% \pm 0.67\%$	$0.78\% \pm 0.29\%$
# Coefficient Parameters	19,200	19,200	19,200	28,800
# Basis Parameters	54,700	4,200	80	120
# Total Parameters	73,900	23,400	19,280	28,920

Table 8: Comparison of different models for the Keller-Segel equation (the numbers 40 and 60 denote the number of basis N used). All proposed FEM-based models employ only a single layer to construct the FEM basis. Notably, when the number of basis layers in the standard DeepONet is reduced to two, its performance degrades. In contrast, FERN maintains high accuracy with a substantially smaller number of trainable parameters.

Setting details. We generate a total of 42 input functions (ICs) and their corresponding solutions for training, where each input function is discretized using 22 uniformly spaced sensors (i.e., a mesh of size 22). Different output functions are evaluated on different non-uniform meshes to introduce challenges for using POD bases: specifically, one-third of the output functions are sampled only on a fine mesh (size 49) in the first third of the domain $[0, 1]$, another third only in the middle third, and the final third only in the last third of the domain. All FEM-based models are trained for 2,000 epochs using a cosine annealing learning rate schedule, while DeepONet models are trained with significantly more epochs to reach stable performance. We explore different configurations for the number of basis functions and the depth of the basis network in DeepONet, as summarized in Table 8.

In all settings, the coefficient network architecture (used in both DeepONet and FERN) follows a fully connected architecture with dimensions $22 \times 20 \rightarrow 20 \times 1$, and use the hyperbolic tangent (tanh) activation for all layers. The centers of the FEM basis functions are uniformly initialized across the domain with a fixed support width of $h = 0.05$. For the standard DeepONet, the basis network adopts a deep fully connected architecture with the structure $1 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times N$, where N is the number of basis, with tanh activations following each hidden layer and a linear output layer; all layers include bias terms except the final one. For the two-layer DeepONet variant, the basis network consists of two layers with dimensions $1 \times 100 \rightarrow 100 \times 1$.

Accuracy. Firstly, due to the use of a non-uniform mesh, as detailed in the previous section, the POD method cannot be applied. As shown in Table 8, consistent with the results observed in other examples, the proposed method achieves comparable or even superior average relative prediction error while using substantially fewer trainable parameters. In particular, when a standard DeepONet is constructed with a two-layer fully connected trunk network, the average error is 5.97%, whereas the proposed method with the learnable FEM basis achieves a significantly lower error of 1.08%. By increasing the depth of the basis network of DeepONet or the number of basis for the propose method, both accuracies can be improved. While the proposed method uses fewer number of parameters.

Adaptivity. As shown in Figure 15, the terminal solutions exhibit a bump profile and decay rapidly outside the center of the domain. Therefore, we expect the learned hat functions may focus more closely to the bump. This behavior is confirmed in our numerical experiments. To validate this hypothesis, we vary the number of basis functions from 40 to 80 and plot the distribution of the learned FEM basis centers. See Figure 18 for the results.

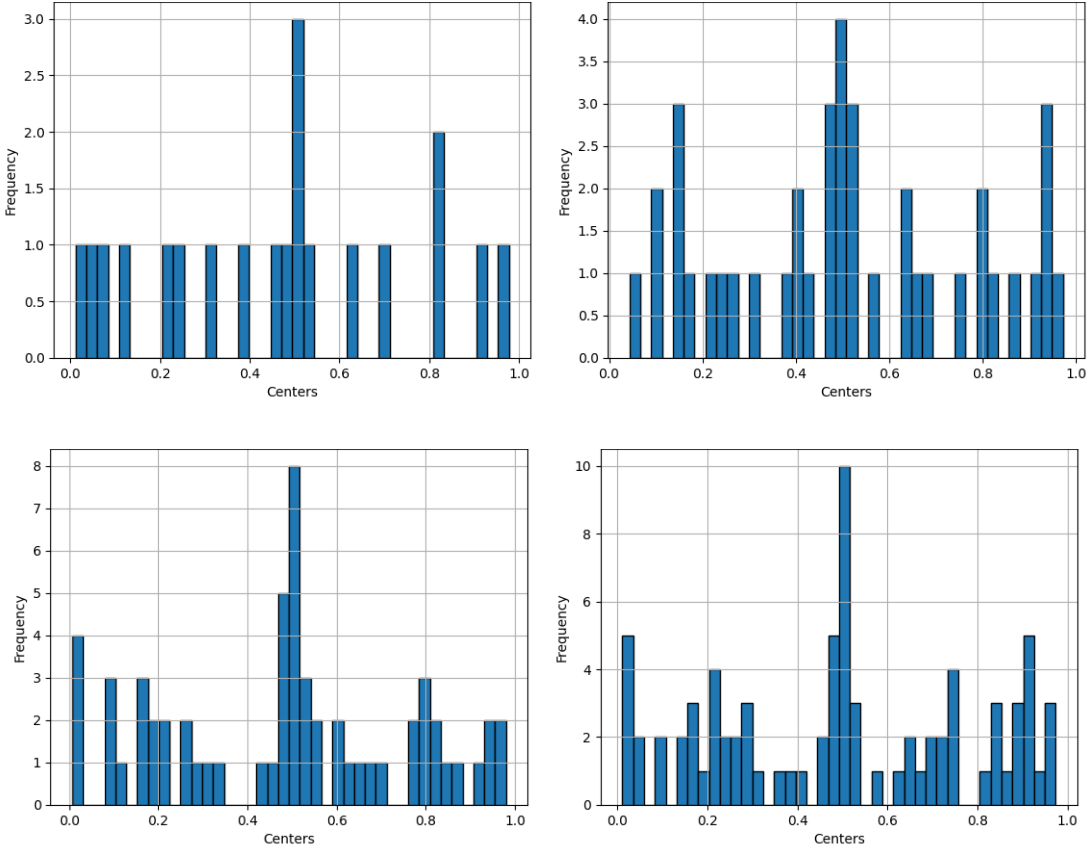


Figure 18: Demonstration of the distributions of the learned basis centers for the Keller-Segel equation. From left to right, top to bottom, we consider models with 20, 40, 60, and 80 basis functions, respectively. The output function exhibits a prominent bump near $x = 0.5$ (see Figure 17), and we observe that more basis functions are concentrated around this region, indicating that the learned basis adapts to the underlying solution structure.

To better illustrate the concept, Figure 19 displays the basis functions (20 in total) both before and after training. Initially, all basis centers are uniformly distributed with a fixed width $h = 0.05$. After training, more basis functions are concentrated near the bump of the terminal solution. The corresponding distribution of basis centers is shown in Figure 18.

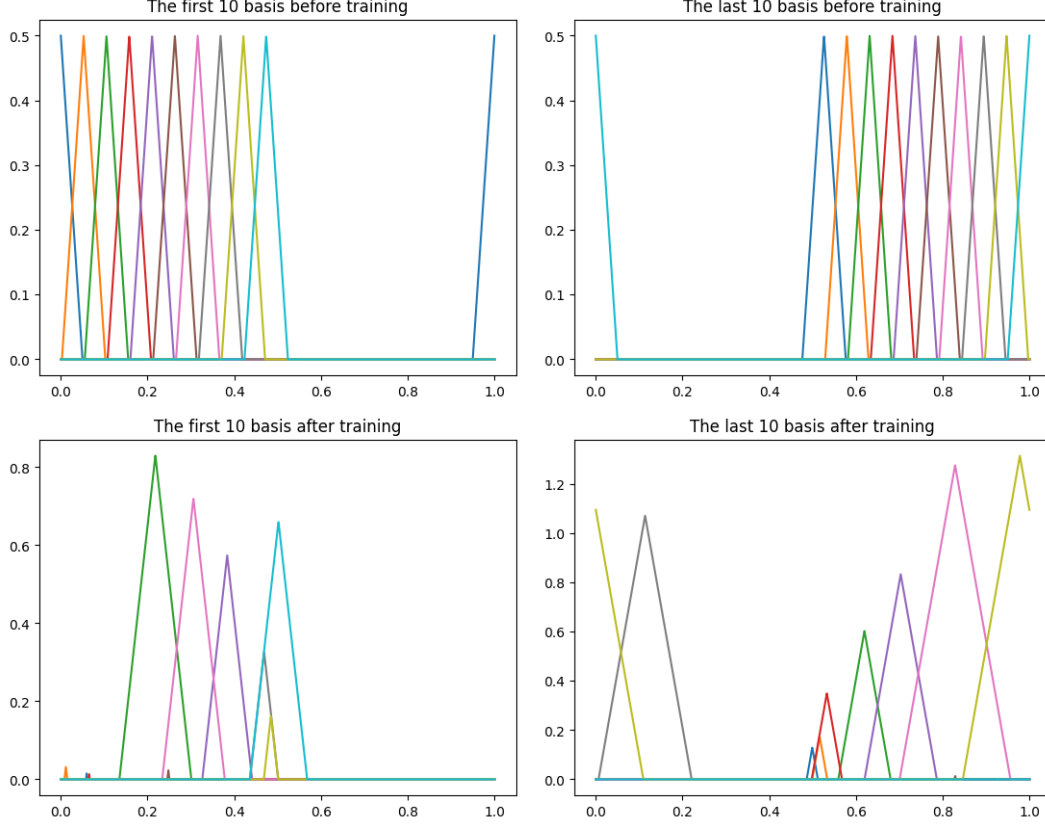


Figure 19: Keller-Segel equation. Demonstration of the FEM basis functions before and after training with 20 basis functions. There is a bump close to $x = 0.5$ (see Figure 17). We observe from this figure that more localized bases with smaller supports are concentrated around $x = 0.5$ to capture the bump, while fewer bases with larger supports are distributed in other regions to represent the flatter tails (left and right) of the solution.

3.7 KdV equation

In this example, we consider the KdV equation,

$$u_t - \epsilon u_{xxx} + (u^2/2)_x = 0, x \in [0, 2], t \in [0, 2].$$

The target operator maps the initial condition to the solution at the terminal time, with the initial conditions generated using

$$u_0(x) = 3c_1 \text{sech}(k_1(x - x_1))^2 + 3c_2 \text{sech}(k_2(x - x_2))^2,$$

with the one free parameter uniformly sampled from $[0, 1]$. We plot two typical initial conditions and solution pairs in Figure 20. Notably, all solutions exhibit a bump within the domain $[0, 1]$, and we use this example to further examine the adaptivity of the proposed method.

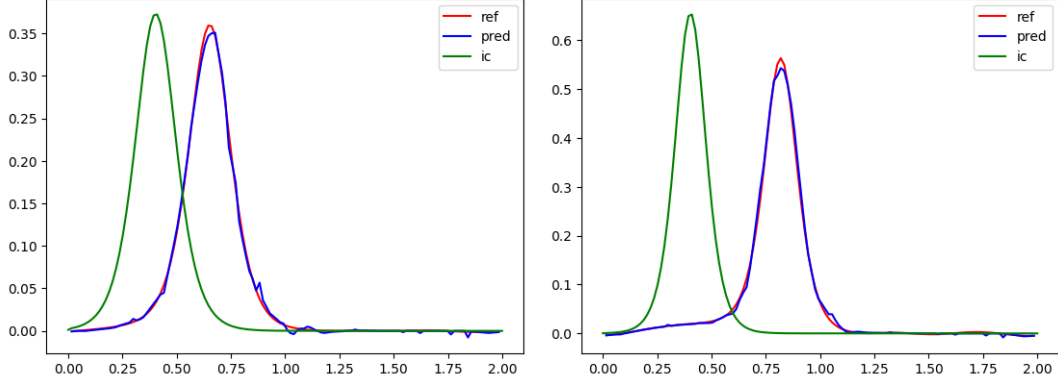


Figure 20: Demonstration of KdV solutions and the prediction using the FEM method with 80 basis functions. The relative errors are 3.86% for the solution shown on the left and 4.23% for the solution on the right.

Setting details. We generate a total of 250 input functions (ICs) and their corresponding solutions for training, where each input function is discretized using 100 sensors (i.e., a mesh of size 100). The output functions are evaluated on a mesh with 64 points for all samples. All FEM-based models are trained for 2,000 epochs using a cosine annealing learning rate schedule, while DeepONet models are trained with significantly more epochs to reach stable performance. We summarize the results in Table 9, and present the distribution of the learned centers in Figure 21.

In all settings, the coefficient network architecture (used in both DeepONet and FERN) follows a fully connected architecture with dimensions $22 \times 20 \rightarrow 20 \times 1$, and use the hyperbolic tangent (tanh) activation for all layers. The centers of the FEM basis functions are uniformly initialized across the domain with a fixed support width of $h = 0.05$. For the standard DeepONet, the basis network adopts a deep fully connected architecture with the structure $1 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times N$ with ReLU activations following each hidden layer and a linear output layer, where N is the number of basis; all layers include bias terms except the final one.

Accuracy. As shown in Table 9, with the same number of basis functions, compared to DeepONet, FERN gives a smaller error with lower variance and fewer parameters. We also test a smaller DeepONet with a two-layer trunk network while keeping the branch network the same as in all other models. Although this reduces the total number of parameters to 113,800, the relative error increases significantly to $13.25\% \pm 4.75\%$.

Adaptivity. As shown in Figure 20, the terminal solutions exhibit a bump in the left half of the domain. We present in Figure 21 the distribution of learned centers. We observe that most centers concentrate on the left half of the domain, demonstrating the adaptivity of the proposed method.

	DeepONet(80)	FERN (80)
Relative L^2 error	$6.0\% \pm 2.81\%$	$3.93\% \pm 1.10\%$
# Coefficient Parameters	105,600	105,600
# Basis Parameters	58,700	160
# Total Parameters	164,300	105,760

Table 9: KdV results with 80 basis. We also test a smaller DeepONet with a two-layer trunk network while keeping the branch network the same as in all other models. Although this reduces the total number of parameters to 113,800, the relative error increases significantly to $13.25\% \pm 4.75\%$.

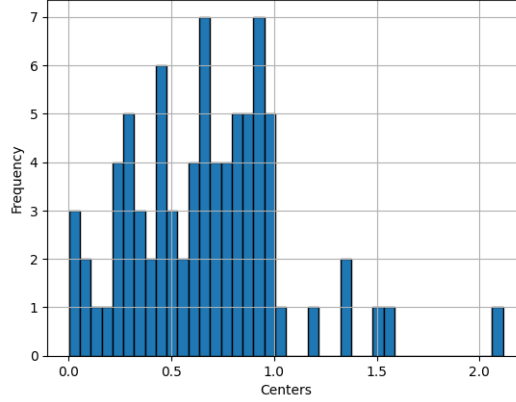


Figure 21: Learned basis center distributions. We create a bump in the solution landscape within the region $[0, 1]$ (see Figure 20 for examples). As shown in this figure, a greater number of basis functions are relocated to this region (though initialized uniformly distributed in the domain), demonstrating the adaptivity of the proposed method.

3.8 Viscous Burgers' equation

Lastly, as a widely tested example in operator learning, we consider the Viscous Burgers' equation,

$$u_t - 0.01u_{xx} + (u^2/2)_x = 0, x \in [0, 1], t \in [0, 1].$$

The target operator maps the initial condition to the solution at the terminal time, with the initial conditions generated using

$$u_0(x) = c_1 \sin(2\pi(x - c_0)) + 0.5,$$

with c_0 uniformly sampled from $[0, 0.5]$ and c_1 uniformly sampled from $[0.5, 1.0]$. We plot two typical initial conditions and solutions pairs in Figure 22.

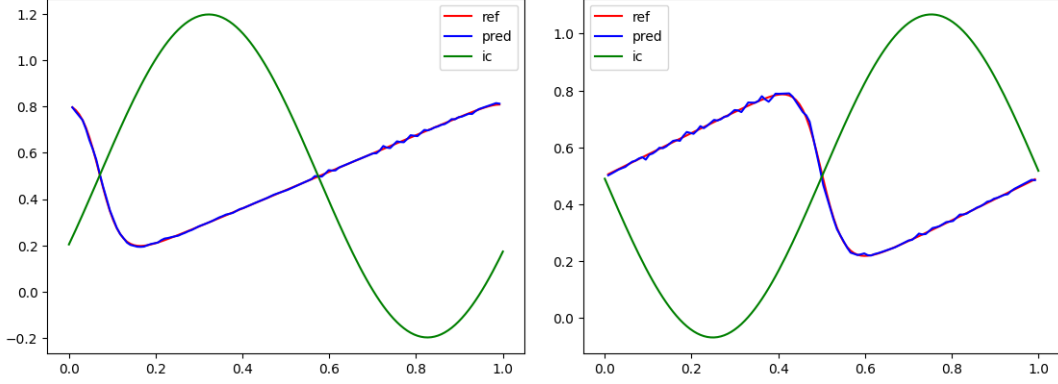


Figure 22: Demonstration of viscous Burgers’ equation solutions and the prediction using the FEM method with 40 basis functions. The relative errors are 0.71% for the solution shown on the left and 1.01% for the solution on the right.

Setting details. We generate a total of 84 input functions (ICs) and their corresponding solutions for training, where each input function is discretized using 22 sensors (i.e., a mesh of size 22). The output functions are evaluated on a mesh with 64 points for all samples. All FEM-based models are trained for 2,000 epochs using a cosine annealing learning rate schedule, while DeepONet models are trained with significantly more epochs to reach stable performance. We summarize the results in Table 10.

In all settings, the coefficient network architecture (used in both DeepONet and FERN) follows a fully connected architecture with dimensions $22 \times 20 \rightarrow 20 \times 1$, and use the hyperbolic tangent (tanh) activation for all layers. The centers of the FEM basis functions are uniformly initialized across the domain with a fixed support width of $h = 0.05$. For the standard DeepONet, the basis network adopts a deep fully connected architecture with the structure $1 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times 100 \rightarrow 100 \times K$, with ReLU activations following each hidden layer and a linear output layer; all layers include bias terms except the final one.

Results analysis. As shown in Table 10, the proposed method gives an accuracy comparable to DeepONet but with fewer parameters.

	DeepONet(40)	FERN (40)
Relative L^2 error	$0.95\% \pm 0.41\%$	$0.93\% \pm 0.31\%$
# Coefficient Parameters	19,200	19,200
# Basis Parameters	54,700	80
# Total Parameters	73,900	19,280

Table 10: Viscous Burgers example results with 40 basis.

4 Discussion

In this paper, we developed a finite-element polynomial basis-based operator learning framework that efficiently solves families of PDEs with a substantially reduced number of trainable parameters. By constructing adaptive FEM bases through a shallow neural architecture, the proposed method retains the flexibility of learnable representations while inheriting the local adaptivity and interpretability of traditional finite element methods. Numerical experiments across seven distinct PDE families confirm that the method achieves competitive or superior accuracy com-

pared to standard DeepONet models, often with orders of magnitude fewer parameters and smaller variance in prediction errors. The proposed framework opens several promising directions for future research. One avenue is to construct higher-order FEM bases by incorporating different polynomial families (e.g., Legendre, Chebyshev, or hierarchical bases) and exploring alternative activation functions to enhance expressiveness. Another important direction is to analyze and improve extrapolation capabilities, as the adaptive nature of the learned FEM bases suggests potential for stronger generalization beyond the training regime. Overall, this study bridges classical finite element analysis and modern operator learning, offering a pathway toward efficient, interpretable, and adaptive neural operator frameworks for solving complex PDE systems.

Acknowledgment

ZZ was supported by the U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program DE-SC0025440. HL was partially supported by HKRGC ECS grant 22302123, HKRGC GRF 12301925, and Guangdong and Hong Kong Universities “1+1+1” Joint Research Collaboration Scheme UICR0800008-24. HS was supported in part by NSF DMS 2427558. GL was supported by the NSF (DMS-2053746, DMS-2134209, ECCS-2328241, CBET-2347401 and OAC-2311848), and U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program DE-SC0023161, the Uncertainty Quantification for Multifidelity Operator Learning (MOLUcQ) project (Project No. 81739), and DOE–Fusion Energy Science, under grant number: DE-SC0024583.

References

- [1] M. AINSWORTH AND J. T. ODEN, *A posteriori error estimation in finite element analysis*, Pure and Applied Mathematics (New York), Wiley-Interscience [John Wiley & Sons], New York, 2000.
- [2] W. BANGERTH AND R. RANNACHER, *Adaptive finite element methods for differential equations*, Lectures in Mathematics ETH Zürich, Birkhäuser Verlag, Basel, 2003.
- [3] K.-J. BATHE, *Finite Element Procedures*, Upper Saddle River, NJ, 2 ed., 1996. International edition, revised; “Prentice Hall International Series in Civil Engineering and Engineering Mechanics”.
- [4] K. BHATTACHARYA, B. HOSSEINI, N. B. KOVACHKI, AND A. M. STUART, *Model reduction and neural networks for parametric pdes*, The SMAI journal of computational mathematics, 7 (2021), pp. 121–157.
- [5] P. BINEV, W. DAHMEN, R. DEVORE, AND P. PETRUSHEV, *Approximation classes for adaptive methods*, vol. 28, 2002, pp. 391–416. Dedicated to the memory of Vassil Popov on the occasion of his 60th birthday.
- [6] C. BODNAR, W. P. BRUINSMA, A. LUCIC, M. STANLEY, A. ALLEN, J. BRANDSTETTER, P. GARVAN, M. RIECHERT, J. A. WEYN, H. DONG, ET AL., *A foundation model for the earth system*, Nature, (2025), pp. 1–8.
- [7] A. BONITO, C. CANUTO, R. H. NOCHETTO, AND A. VEESER, *Adaptive finite element methods*, Acta Numer., 33 (2024), pp. 163–485.

- [8] S. C. BRENNER AND L. R. SCOTT, *The mathematical theory of finite element methods*, vol. 15 of Texts in Applied Mathematics, Springer, New York, third ed., 2008.
- [9] S. L. BRUNTON, J. L. PROCTOR, AND J. N. KUTZ, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proceedings of the national academy of sciences, 113 (2016), pp. 3932–3937.
- [10] Y. CAO, Y. LIU, L. YANG, R. YU, H. SCHAEFFER, AND S. OSHER, *Vicon: Vision in-context operator networks for multi-physics fluid dynamics prediction*, arXiv preprint arXiv:2411.16063, (2024).
- [11] M. CHEN, H. JIANG, W. LIAO, AND T. ZHAO, *Nonparametric regression on low-dimensional manifolds using deep relu networks: Function approximation and statistical recovery*, Information and Inference: A Journal of the IMA, 11 (2022), pp. 1203–1253.
- [12] T. CHEN AND H. CHEN, *Approximations of continuous functionals by neural networks with application to dynamic systems*, IEEE Transactions on Neural networks, 4 (1993), pp. 910–918.
- [13] ———, *Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems*, IEEE Transactions on Neural Networks, 6 (1995), pp. 911–917.
- [14] P. G. CIARLET, *The finite element method for elliptic problems*, vol. 40 of Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002. Reprint of the 1978 original [North-Holland, Amsterdam; MR0520174 (58 #25001)].
- [15] R. COURANT, *Variational methods for the solution of problems of equilibrium and vibrations*, Lecture notes in pure and applied mathematics, (1994), pp. 1–1.
- [16] M. V. DE HOOP, D. Z. HUANG, E. QIAN, AND A. M. STUART, *The cost-accuracy trade-off in operator learning with neural networks*, arXiv preprint arXiv:2203.13181, (2022).
- [17] C. DENG, S. FENG, H. WANG, X. ZHANG, P. JIN, Y. FENG, Q. ZENG, Y. CHEN, AND Y. LIN, *Openfwi: Large-scale multi-structural benchmark datasets for full waveform inversion*, Advances in Neural Information Processing Systems, 35 (2022), pp. 6007–6020.
- [18] Y. EFENDIEV, W. T. LEUNG, G. LIN, AND Z. ZHANG, *Efficient hybrid explicit-implicit learning for multiscale problems*, Journal of Computational Physics, (2022), p. 111326.
- [19] K. FENG, *Difference schemes based on variational principle (in Chinese)*, Applied Mathematics and Computational Mathematics, 2 (1965), pp. 238–262.
- [20] S. GOSWAMI, M. YIN, Y. YU, AND 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), p. 114587.
- [21] J. GUIBAS, M. MARDANI, Z. LI, A. TAO, A. ANANDKUMAR, AND B. CATANZARO, *Adaptive fourier neural operators: Efficient token mixers for transformers*, arXiv preprint arXiv:2111.13587, (2021).
- [22] W. HAO, X. LIU, AND Y. YANG, *Newton informed neural operator for solving nonlinear partial differential equations*, Advances in neural information processing systems, 37 (2024), pp. 120832–120860.

- [23] A. HAVRILLA AND W. LIAO, *Understanding scaling laws with statistical and approximation theory for transformer neural networks on intrinsically low-dimensional data*, Advances in Neural Information Processing Systems, 37 (2024), pp. 42162–42210.
- [24] J. HE, L. LI, J. XU, AND C. ZHENG, *Relu deep neural networks and linear finite elements*, J. Comput. Math., 38 (2020), pp. 502–527.
- [25] A. A. HOWARD, M. PEREGO, G. E. KARNIADAKIS, AND P. STINIS, *Multifidelity deep operator networks*, arXiv preprint arXiv:2204.09157, (2022).
- [26] A. HRENNIKOFF, *Solution of problems of elasticity by the framework method*, (1941).
- [27] C. JOHNSON, *Numerical solution of partial differential equations by the finite element method*, Dover Publications, Inc., Mineola, NY, 2009. Reprint of the 1987 edition.
- [28] D. JOLLIE, J. SUN, Z. ZHANG, AND H. SCHAEFFER, *Time-series forecasting and refinement within a multimodal pde foundation model*, Journal of Machine Learning for Modeling and Computing, 6 (2025), pp. 77–89.
- [29] N. KOVACHKI, S. LANTHALER, AND S. MISHRA, *On universal approximation and error bounds for fourier neural operators*, Journal of Machine Learning Research, 22 (2021), pp. 1–76.
- [30] V. KUMAR, S. GOSWAMI, K. KONTOLATI, M. D. SHIELDS, AND G. E. KARNIADAKIS, *Synergistic learning with multi-task deeponet for efficient pde problem solving*, Neural Networks, 184 (2025), p. 107113.
- [31] T. KURTH, S. SUBRAMANIAN, P. HARRINGTON, J. PATHAK, M. MARDANI, D. HALL, A. MIELE, K. KASHINATH, AND A. ANANDKUMAR, *Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators*, in Proceedings of the platform for advanced scientific computing conference, 2023, pp. 1–11.
- [32] S. LANTHALER, *Operator learning with pca-net: upper and lower complexity bounds*, Journal of Machine Learning Research, 24 (2023), pp. 1–67.
- [33] S. LANTHALER, S. MISHRA, AND G. E. KARNIADAKIS, *Error estimates for deeponets: A deep learning framework in infinite dimensions*, Transactions of Mathematics and Its Applications, 6 (2022), p. tnac001.
- [34] S. LANTHALER AND A. M. STUART, *The parametric complexity of operator learning*, arXiv preprint arXiv:2306.15924, (2023).
- [35] S. LANTHALER, A. M. STUART, AND M. TRAUTNER, *Discretization error of fourier neural operators*, arXiv preprint arXiv:2405.02221, (2024).
- [36] Z. LI, D. Z. HUANG, B. LIU, AND A. ANANDKUMAR, *Fourier neural operator with learned deformations for pdes on general geometries*, arXiv preprint arXiv:2207.05209, (2022).
- [37] —, *Fourier neural operator with learned deformations for pdes on general geometries*, Journal of Machine Learning Research, 24 (2023), pp. 1–26.
- [38] Z. LI, N. KOVACHKI, K. AZIZZADENESHELI, B. LIU, K. BHATTACHARYA, A. STUART, AND A. ANANDKUMAR, *Fourier neural operator for parametric partial differential equations*, arXiv preprint arXiv:2010.08895, (2020).

- [39] C. LIN, Z. LI, L. LU, S. CAI, M. MAXEY, AND G. E. KARNIADAKIS, *Operator learning for predicting multiscale bubble growth dynamics*, The Journal of Chemical Physics, 154 (2021).
- [40] G. LIN, C. MOYA, AND Z. ZHANG, *B-deeponet: An enhanced bayesian deeponet for solving noisy parametric pdes using accelerated replica exchange sgld*, Journal of Computational Physics, 473 (2023), p. 111713.
- [41] H. LIU, M. CHEN, T. ZHAO, AND W. LIAO, *Besov function approximation and binary classification on low-dimensional manifolds using convolutional residual networks*, in International Conference on Machine Learning, PMLR, 2021, pp. 6770–6780.
- [42] H. LIU, B. DAHAL, R. LAI, AND W. LIAO, *Generalization error guaranteed auto-encoder-based nonlinear model reduction for operator learning*, Applied and Computational Harmonic Analysis, 74 (2025), p. 101717.
- [43] H. LIU, H. YANG, M. CHEN, T. ZHAO, AND W. LIAO, *Deep nonparametric estimation of operators between infinite dimensional spaces*, Journal of Machine Learning Research, 25 (2024), pp. 1–67.
- [44] H. LIU, Z. ZHANG, W. LIAO, AND H. SCHAEFFER, *Neural scaling laws of deep relu and deep operator network: A theoretical study*, arXiv preprint arXiv:2410.00357, (2024).
- [45] Y. LIU, J. SUN, X. HE, G. PINNEY, Z. ZHANG, AND H. SCHAEFFER, *Prose-fd: A multi-modal pde foundation model for learning multiple operators for forecasting fluid dynamics*, arXiv preprint arXiv:2409.09811, (2024).
- [46] Y. LIU, J. SUN, AND H. SCHAEFFER, *BCAT: A block causal transformer for pde foundation models for fluid dynamics*, arXiv preprint arXiv:2501.18972, (2025).
- [47] Y. LIU, Z. ZHANG, AND H. SCHAEFFER, *Prose: Predicting operators and symbolic expressions using multimodal transformers*, Neural Networks, 180 (2024), p. 106707.
- [48] L. LU, P. JIN, G. PANG, Z. Q. ZHANG, AND G. E. KARNIADAKIS, *Learning nonlinear operators via deeponet based on the universal approximation theorem of operators*, Nature Machine Intelligence, 3 (2021), pp. 218–229.
- [49] L. LU, X. MENG, S. CAI, Z. MAO, S. GOSWAMI, Z. Q. ZHANG, AND G. E. KARNIADAKIS, *A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data*, Computer Methods in Applied Mechanics and Engineering, 393 (2022), p. 114778.
- [50] L. LU, X. MENG, Z. MAO, AND G. E. KARNIADAKIS, *Deepxde: A deep learning library for solving differential equations*, SIAM review, 63 (2021), pp. 208–228.
- [51] S. MAO, R. DONG, K. M. YI, L. LU, S. WANG, AND P. PERDIKARIS, *Disk2planet: A robust and automated machine learning tool for parameter inference in disk-planet systems*, arXiv preprint arXiv:2409.17228, (2024).
- [52] B. MEURIS, S. QADEER, AND P. STINIS, *Machine-learning-based spectral methods for partial differential equations*, Scientific Reports, 13 (2023), p. 1739.
- [53] R. NAKADA AND M. IMAIZUMI, *Adaptive approximation and generalization of deep neural network with intrinsic dimensionality*, Journal of Machine Learning Research, 21 (2020), pp. 1–38.

- [54] E. NEGRINI, Y. LIU, L. YANG, S. J. OSHER, AND H. SCHAEFFER, *A multimodal pde foundation model for prediction and scientific text descriptions*, arXiv preprint arXiv:2502.06026, (2025).
- [55] J. PATHAK, S. SUBRAMANIAN, P. HARRINGTON, S. RAJA, A. CHATTOPADHYAY, M. MARDANI, T. KURTH, D. HALL, Z. LI, K. AZIZZADENESHELI, ET AL., *Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators*, arXiv preprint arXiv:2202.11214, (2022).
- [56] M. RAISSI, P. PERDIKARIS, AND 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), pp. 686–707.
- [57] H. SCHAEFFER, R. CAFLISCH, C. D. HAUCK, AND S. OSHER, *Sparse dynamics for partial differential equations*, Proceedings of the National Academy of Sciences, 110 (2013), pp. 6634–6639.
- [58] H. SCHAEFFER AND S. G. MCCALLA, *Sparse model selection via integral terms*, Physical Review E, 96 (2017), p. 023302.
- [59] S. SUBRAMANIAN, P. HARRINGTON, K. KEUTZER, W. BHIMJI, D. MOROZOV, M. W. MAHONEY, AND A. GHOLAMI, *Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior*, Advances in Neural Information Processing Systems, 36 (2024).
- [60] J. SUN, Y. LIU, Z. ZHANG, AND H. SCHAEFFER, *Towards a foundation model for partial differential equations: Multioperator learning and extrapolation*, Physical Review E, 111 (2025), p. 035304.
- [61] T. SUZUKI, *Adaptivity of deep relu network for learning in besov and mixed smooth besov spaces: optimal rate and curse of dimensionality*, arXiv preprint arXiv:1810.08033, (2018).
- [62] J. WANG, Y. MAO, X. LIU, AND W. HAO, *Learning patient-specific spatial biomarker dynamics via operator learning for alzheimer’s disease progression*, arXiv preprint arXiv:2507.16148, (2025).
- [63] G. WEN, Z. LI, K. AZIZZADENESHELI, A. ANANDKUMAR, AND S. M. BENSON, *U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow*, Advances in Water Resources, 163 (2022), p. 104180.
- [64] L. YANG, S. LIU, T. MENG, AND S. J. OSHER, *In-context operator learning with data prompts for differential equation problems*, Proceedings of the National Academy of Sciences, 120 (2023), p. e2310142120.
- [65] L. YANG, S. LIU, AND S. J. OSHER, *Fine-tune language models as multi-modal differential equation solvers*, Neural Networks, (2025), p. 107455.
- [66] L. YANG, T. MENG, S. LIU, AND S. J. OSHER, *Prompting in-context operator learning with sensor data, equations, and natural language*, arXiv preprint arXiv:2308.05061, (2023).
- [67] D. YAROTSKY, *Error bounds for approximations with deep relu networks*, Neural networks, 94 (2017), pp. 103–114.
- [68] Z. YE, X. HUANG, L. CHEN, Z. LIU, B. WU, H. LIU, Z. WANG, AND B. DONG, *Pdeformer-1: A foundation model for one-dimensional partial differential equations*, arXiv preprint arXiv:2407.06664, (2024).

- [69] Y. YU, N. LIU, F. LU, T. GAO, S. JAFARZADEH, AND S. A. SILLING, *Nonlocal attention operator: Materializing hidden knowledge towards interpretable physics discovery*, Advances in Neural Information Processing Systems, 37 (2024), pp. 113797–113822.
- [70] L. ZHANG AND H. SCHAEFFER, *On the convergence of the sindy algorithm*, Multiscale Modeling & Simulation, 17 (2019), pp. 948–972.
- [71] Z. ZHANG, *Modno: Multi-operator learning with distributed neural operators*, Computer Methods in Applied Mechanics and Engineering, 431 (2024), p. 117229.
- [72] Z. ZHANG, H. LIU, W. LIAO, AND G. LIN, *Coefficient-to-basis network: a fine-tunable operator learning framework for inverse problems with adaptive discretizations and theoretical guarantees*, Philosophical Transactions A, 383 (2025), p. 20240054.
- [73] Z. ZHANG, C. MOYA, L. LU, G. LIN, AND H. SCHAEFFER, *D2no: Efficient handling of heterogeneous input function spaces with distributed deep neural operators*, Computer Methods in Applied Mechanics and Engineering, 428 (2024), p. 117084.
- [74] Z. ZHANG, L. WING TAT, AND H. SCHAEFFER, *Belnet: Basis enhanced learning, a mesh-free neural operator*, Proceedings of the Royal Society A, 479 (2023), p. 20230043.
- [75] M. ZHU, S. FENG, Y. LIN, AND L. LU, *Fourier-deeponet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness*, Computer Methods in Applied Mechanics and Engineering, 416 (2023), p. 116300.
- [76] O. C. ZIENKIEWICZ, R. L. TAYLOR, AND J. Z. ZHU, *The finite element method: its basis and fundamentals*, Elsevier/Butterworth Heinemann, Amsterdam, seventh ed., 2013.