Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 391 (2022) 114587
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials

Somdatta Goswami[a], Minglang Yin[b,c], Yue Yu[d], George Em Karniadakis[a,c,*]

[a] *Division of Applied Mathematics, Brown University, Providence, RI, United States of America*
[b] *Center for Biomedical Engineering, Brown University, Providence, RI, United States of America*
[c] *School of Engineering, Brown University, Providence, RI, United States of America*
[d] *Department of Mathematics, Lehigh University, Bethlehem, PA, United States of America*

Received 16 August 2021; received in revised form 19 November 2021; accepted 4 January 2022
Available online 28 January 2022

Abstract

Failure trajectories, probable failure zones, and damage indices are some of the key quantities of relevance in brittle fracture mechanics. High-fidelity numerical solvers that reliably estimate these relevant quantities exist but they are computationally demanding requiring a high resolution of the crack. Moreover, independent simulations need to be carried out even for a small change in domain parameters and/or material properties. Therefore, fast and generalizable surrogate models are needed to alleviate the computational burden but the discontinuous and complex nature of fracture mechanics presents a major challenge to developing such models. We propose a physics-informed variational formulation of DeepONet (V-DeepONet) for brittle fracture analysis. V-DeepONet is trained to map the initial configuration of the defect to the relevant fields of interests (e.g., damage and displacements). Once the network is trained, the entire global solution can be rapidly obtained for any initial crack configuration and loading steps on that domain. While the original DeepONet is solely data-driven, we take a different path to train the V-DeepONet by imposing the governing equations in a variational form with some labeled data. We demonstrate the effectiveness of V-DeepOnet through two benchmarks of brittle fracture and verify its accuracy using results from high-fidelity solvers. Encoding the physical laws to the model with data enhancement in training renders the surrogate model capable of accurately performing both interpolation and extrapolation tasks. Considering that fracture modeling is very sensitive to fluctuations, the proposed V-DeepONet with a hybrid training strategy is able to predict the quantities of interests with good accuracy, which can be easily extended to a wide array of dynamical systems with complex responses.

Keywords: DeepONet; Variational energy; Physics-informed learning; Phase-field; Brittle fracture; Surrogate modeling

1. Introduction

Damage evolution in realistic structures is a complex phenomenon. Accurately representing this behavior relies on complex and computationally expensive high-fidelity models. Traditional approaches in computational science have undergone remarkable growth and progress but they still operate under stringent requirements. More often than

* Correspondence to: Department of Applied Mathematics, 170 Hope Street, Providence-02906, RI, United States of America.
E-mail addresses: somdatta_goswami@brown.edu (S. Goswami), minglang_yin@brown.edu (M. Yin), yuy214@lehigh.edu (Y. Yu), george_karniadakis@brown.edu (G.E. Karniadakis).

https://doi.org/10.1016/j.cma.2022.114587
0045-7825/

not they require precise knowledge of an underlying model that describes conservation. Moreover, most existing numerical methods utilize spatial discretization schemes and thus are prone to the curse of dimensionality. Newer alternatives such as phase-field [1–4] and peridynamics [5–10] models have been able to predict the outcome of carefully controlled experiments; however, accuracy of those models comes at a substantially higher computational cost. Furthermore, operational conditions and material properties in the field can have a significant deviation from those in a controlled laboratory environment, which impairs the reliability of the computed results. In order to assess the likelihood of occurrence of cracks and their possible effects for a range of possible parameters and operating conditions, a large number of high-fidelity simulations is required, which are typically computationally prohibitive.

Surrogate models [11–19] have received a lot of attention because of their ability to quantitatively capture the fundamental attributes of high-fidelity models, while significantly improving the computational efficiency. The intrinsic discontinuous nature of the physical phenomena is the fundamental problem in developing a surrogate model for fracture analysis. The quality of the surrogate models relies on the smoothness in the system response, however fracture mechanics models are extremely sensitive to model property variations [20] (e.g., initial length and location of a pre-notch tip) and hence exhibit erratic behaviors. This problem has prompted a burgeoning literature on reduced-order approaches [21–23], which use existing data sets to create fast emulators often at the expense of accuracy, stability, and generalization. The goal of the present work is to develop a flexible generalized prognosis framework for high-fidelity crack growth models. To this end, we will use deep neural networks (DNNs) to infer the generalized solution of the governing partial differential equations (PDEs) that describe the fracture mechanisms.

Deep learning allows overparametrized neural networks with several processing layers to learn multiple levels of abstraction for representations of the raw input data. These networks are known to be particularly good at supervised learning tasks, which typically necessitate the availability of huge volumes of labeled data. However, data collecting is generally prohibitively expensive in many engineering applications, and the amount of available data is typically minimal. As a result, in this "sparse data" environment, it is critical to use domain knowledge to reduce the demand for labeled training data, or even to train deep learning models using only constraints rather than data. Physics-informed neural networks (PINNs) [24–27] use these constraints to encapsulate the output's specific structure and qualities, which are known to hold due to domain knowledge, such as known physical laws like conservation of momentum, mass, and energy. This approach takes advantage of the expressivity of DNNs to approximate any continuous function. To efficiently approximate the solution of PDEs with discontinuity, variational energy-based PINNs (VE-PINNs) were proposed in [28,29], where the network is trained by minimizing the variational energy of the system (defined using the weak formulation). VE-PINN opened a new paradigm for solving fracture problems with modern neural network architectures, which may be a promising alternative to traditional numerical methods, such as finite-difference and finite-volume methods as it reduces the computational burden of dense discretization. Despite the promise and collection of impressive results for accurately estimating the crack path in brittle fracture problems using a sparse discretization, VE-PINN bears a formidable cost as independent simulations need to be performed for every different domain geometry, input parameters, or initial/boundary conditions (I/BCs). The bottleneck of VE-PINN is quite similar to traditional numerical methods, hence the use of VE-PINN as a surrogate model for approximating the crack path for different initial conditions or for comprehensive uncertainty quantification is practically not feasible.

As the ML revolution continues to sweep the scientific world, a new wave of strategies for expediting the simulation of PDEs [30–39] is being offered. In a general setting, discovering PDEs solely from data without any prior knowledge is challenging. In most practical circumstances, it is necessary to have a surrogate model of the PDE solution operator that can simulate PDE solutions repeatedly for varied I/BCs, rather than discovering the PDE in an explicit form, to handle this difficulty. DeepONet proposed in [40] is one of the possible ways to learn the PDE solution operators from the labeled input–output datasets. The idea of DeepONet is motivated by the universal approximation theorem for operators. This defines a new and relatively under-explored realm for DNN-based approaches that map infinite-dimensional functional spaces rather than finite-dimensional vector spaces (functional regression) [41]. The computational model consists of two DNNs, one encodes the input function at fixed sensor points (branch net), while another for the location of the output function (trunk net). In this work, we propose a variational energy-based framework of DeepONet to parametrize and learn the solution operator that maps multiple initial conditions of the crack in the domain as input function to the branch net to their associated solutions at the locations embedded in the trunk net, thus overcoming the fundamental challenge of VE-PINN.

The proposed deep learning model is trained by minimizing a hybrid loss function constructed using the PDEs defining the variational formulation of phase field approach, its associated I/BCs and relatively small input–output

datasets generated using the in-house high-fidelity solvers. Once the model has been trained on a set of specific conditions (initial configurations, loading steps, etc.), it can be used to build the global PDE solution to predict quantities of interest using a simple iterative approach in which the previous time-step's prediction is utilized as an initial condition for the current time step. We demonstrate that this approach can effectively enable the integration of evolution equations subject to a range of multiple initial conditions with good generalization accuracy, all at a fraction of the computational cost needed by classical numerical solvers. The main advantage of our approach is that we make no assumptions on the regularity of the full model and as we demonstrate herein the ability to generalize and predict outcomes for inputs outside the distribution (extrapolation),

We demonstrate the performance of the surrogate model on two benchmark problems of fracture: crack growth under tensile loading (Mode-I) and shear loading (Mode-II). In this work, we have used the phase field approach to model fracture within the framework of isogeometric analysis (IGA) developed in [42] to generate data. However, the method presented here is not restricted to any specific fracture model or mode of failure or data generated using specific high-fidelity solvers. Even though the surrogate model proposed in this work is applicable to various evolution equations, e.g., time dependent problems and fracture dynamics, the integration of variational formulation makes its a perfect choice for PDEs with discontinuous nature which manifests in both spatial domain and system response for different material properties.

The remainder of the paper is organized as follows. In Section 2, we discuss the problem statement for phase-field modeling of brittle fracture using the variational energy formulation. In Section 3, we provide an overview of the DeepONet framework put forth in [40]. Implementation of the variational formulation within DeepONet and the concept of hybrid loss function is discussed in Section 4. The details of the construction of the proposed surrogate model within V-DeepONet, its implementation and its numerous application are elaborated in Section 5, with data generation procedures described in Section 6. In Section 7 different aspects of the surrogate model illustrating its performance are tested through numerical problems. Each numerical example in the manuscript is accompanied with a detailed discussion about the neural network architecture we employed as well as details about its training process. Finally, Section 8 presents the concluding remarks and possibilities future work. In the Appendices we present more details and additional cases for the interested reader to be able to reproduce our results.

2. Phase field modeling of fracture

In recent times, phase field modeling approaches have been extensively used in science and engineering to model a variety of phenomena. Modeling fracture, using the phase field approach, involves the integration of two fields, namely the vector-valued elastic field and the scalar-valued phase field. While crack nucleation may depend on stress, the propagation of cracks requires an increase in the fracture energy or the surface energy, Ψ_c of a solid [43]. Hence, the energy criteria are used in the study of fracture using the phase field approach [44]. In this section, we introduce the optimization problem written in terms of energy minimization to solve phase field based fracture analysis. However, for detailed conceptualization of the phase field based fracture modeling, readers may refer to [45].

The physical domain, $\Omega \subset \mathbb{R}^d$, is defined with the external boundary, $\partial\Omega \subset \mathbb{R}^{d-1}$, where d denotes the number of spatial dimensions, $d \in \{1, 2, 3\}$. In a quasi-static loading regime, the total energy functional, $\mathcal{E}$, can be written as [46]:

$$
\begin{aligned}
\mathcal{E} &= \Psi_e + \Psi_c - \mathcal{P}_{ext}, \\
\text{where} \quad \Psi_e &= \int_\Omega \psi_e\left(\boldsymbol{\epsilon}\left(\boldsymbol{w}\right), \phi\left(\boldsymbol{x}\right)\right) d\Omega, \\
\Psi_c &= \int_{\Gamma_d} G_c \, d\Gamma \;\; \approx \int_\Omega G_c \Theta\left(\phi(\boldsymbol{x}), l_0\right) d\Omega, \\
\text{and} \quad \mathcal{P}_{ext} &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{w} \, d\Omega + \int_{\partial\Omega_N} \boldsymbol{t}_N \cdot \boldsymbol{w} \, d\Gamma.
\end{aligned}
\tag{1}
$$

In Eq. (1), Ψ_e is the stored elastic strain energy, $d\Gamma$ denotes the integral on the co-dimensional one space (curves for $d = 2$ an surfaces for $d = 3$), Γ_d is the evolving internal discontinuity boundary and ψ_e is the strain energy density functional expressed in terms of the linearized strain tensor, $\boldsymbol{\epsilon}(\boldsymbol{w})$, where $\boldsymbol{w}$ denotes the solution of the elastic field and a continuous scalar parameter $\phi(\boldsymbol{x})$ denoting the phase field used to track the fracture pattern. The cracked

region is represented by $\phi = 1$ while the undamaged portion is given by $\phi = 0$. The fracture energy, Ψ_c, is defined in terms of the critical energy release rate, G_c, integrated over the fracture surface. The phase-field approximation introduces a crack density functional, Θ, that is dependent on a length scale parameter, l_0, the phase-field, ϕ, and derivatives of ϕ such that the approximation stated in Eq. (1) for the fracture energy holds true. A main feature of phase-field modeling is the assumption that the process zone has a finite width, which is controlled by l_0. A sharp crack topology is recovered in the limit as $l_0 \to 0$ [47]. The external potential energy, $\mathcal{P}_{ext}$, is computed using the prescribed boundary force. The traction load, $\boldsymbol{t}_N$ is applied over the Neumann boundary, $\partial\Omega_N$, and a distributed body force, $\boldsymbol{f}$, is applied over the whole domain.

In the phase field approach, the crack path is resolved by minimizing the energy functional, $\mathcal{E}$, defined in Eq. (1). The problem statement can be written as:

$$\text{Minimize:} \quad \mathcal{E} = \Psi_e + \Psi_c, \tag{2}$$

without the application of external load, $\mathcal{P}_{ext}$ and subject to proper boundary conditions. Ψ_e describes a smooth transition from the intact bulk material to the fully cracked state, characterized by a monotonically decreasing stress-degradation function, $g(\phi)$, which reduces the stiffness of the bulk material. Taking into account that the compressive strain energy does not participate in the propagation of the crack, a tension–compression split of $\Psi_e(\epsilon)$ is considered as:

$$\begin{aligned} &\Psi_e(\boldsymbol{\epsilon}) = g(\phi)\,\psi_e^+(\boldsymbol{\epsilon}) + \psi_e^-(\boldsymbol{\epsilon}), \\ &\text{where} \quad g(\phi) = (1-\phi)^2. \end{aligned} \tag{3}$$

ψ_e^+ and ψ_e^- are the tensile and the compressive components of the strain energies densities obtained by the spectral decomposition of the strain tensor. In this work, without loss of generality, we consider the Dirichlet condition on displacement, while the approach can also be extended to other types of conditions. Using the variational approach, the traction-free Neumann boundary conditions are automatically satisfied. In Eq. (2), Ψ_e and Ψ_c are defined as:

$$\begin{aligned} \Psi_e &= \int_\Omega f_e(\boldsymbol{x}) d\Omega, \\ \Psi_c &= \int_\Omega f_c(\boldsymbol{x}) d\Omega, \end{aligned} \tag{4}$$

where

$$f_e(\boldsymbol{x}) = g(\phi)\psi_e^+(\boldsymbol{\epsilon}) + \psi_e^-(\boldsymbol{\epsilon}), \tag{5a}$$

$$f_c(\boldsymbol{x}) = \frac{G_c}{2l_0}\left(\phi^2 + l_0^2|\nabla\phi|^2\right) - g(\phi)H(\boldsymbol{x},t), \tag{5b}$$

where $H(\boldsymbol{x},t)$ is the strain-history functional, introduced in [48] to enforce irreversibility conditions on the crack growth. The strain-history function is defined as:

$$H(\boldsymbol{x},t) = \max_{s\in[0,t]} \psi_e^+(\boldsymbol{\epsilon}(\boldsymbol{x},s)) \quad \forall \ \boldsymbol{x} \in \Omega. \tag{6}$$

The strain history functional could also be used to initialize the crack in the domain. The initial strain-history functional, $H(\boldsymbol{x},0)$ defined as

$$H(\boldsymbol{x},0) = \begin{cases} \frac{BG_c}{2l_0}\left(1 - \frac{2d(\boldsymbol{x},l)}{l_0}\right) & d(\boldsymbol{x},l) \leqslant \frac{l_0}{2} \\ 0 & d(\boldsymbol{x},l) > \frac{l_0}{2} \end{cases}. \tag{7}$$

In the next section, we will provide an overview of the conventional DeepONet framework put forth in [40] before we discuss how the variational formulation is integrated in the DeepONet to develop a surrogate model for fracture analysis.

3. DeepONet

The idea of DeepONet is motivated by the universal approximation theorem for operators [49], which states that a neural network with a single hidden layer can approximate accurately any linear/non-linear continuous function or operator. Before we focus on learning the solution operators of the parametric PDEs, it is important to understand

the difference between a function regression and an operator regression. In the function regression approach, the solution operator is parametrized as a neural network between finite dimensional Euclidean spaces: $\mathcal{F} : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_1}$, where d_1 is the number of discretization points. However, in operator regression, a function is mapped to another function through an operator. In other words, it is the mapping of infinite-dimensional space to another infinite dimensional space. Using operator regression, the operators would be trained to approximate the solution of the input functions by learning the non-linear operator from the data.

The DeepONet architecture consists of two neural networks: one encodes the input function, $\boldsymbol{u}$ at fixed sensor points (branch net), while another represents the output for the location, $\boldsymbol{y}$ of evaluation of the output function (trunk net). The success of deep learning has been largely attributed to the depth of the networks, i.e., the number of successive affine transformations followed by non-linearity, which is shown to be extracting hierarchical features from the data. Hence, in this work, we have used two deep, fully connected feed-forward neural networks (branch net and trunk net) to approximate the solution operator. The goal of the DeepONet algorithm is to learn the operator, $\mathcal{G}$, which takes as an input the function $\boldsymbol{u}$ in the branch net, and then $\mathcal{G}(\boldsymbol{u})$ is the corresponding output function. The output of the branch net is evaluated at $\boldsymbol{y}$ continuous coordinates (input to the trunk net). Although the architecture proposed here can be applied to more general problems, in the following we illustrate the formulation on a 2D problem for simplicity. The output of the DeepONet is a scalar and is expressed as $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u})(\boldsymbol{y})$, where $\boldsymbol{\theta} = (\mathbf{W}, \boldsymbol{\beta})$ includes the trainable parameters (weights, $\mathbf{W}$, and biases, $\boldsymbol{\beta}$) of the DeepONet. The input functions to the branch net may include, the shape of the physical domain, the initial or boundary conditions, constant or variable coefficients, source terms, etc. Even though the branch net takes a function as input, we have to represent the input functions discretely, so that network approximations can be applied. To that end, all the input functions, $\boldsymbol{u}$, are evaluated at finite locations, $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$, referred to as sensors. The location of the m sensors must be the same for all the functions, $\boldsymbol{u}$. We do not enforce any constraints on the output locations, $\boldsymbol{y}$. A schematic representation of DeepONet is shown in Fig. 1(a), where the branch net takes as input n functions represented as $\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \boldsymbol{u}^{(3)}, \ldots, \boldsymbol{u}^{(n)}$, and the solution operator is evaluated at $\boldsymbol{y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_p\} = \{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \ldots, (\hat{x}_p, \hat{y}_p)\}$, which are the inputs to the trunk net. Here $\hat{x}_i$ $\hat{y}_i$ denote the x and y coordinates of point $\boldsymbol{y}_i$, respectively.

Let us consider that the branch neural network consists of l_{br} hidden layers, where the l_{br}th layer is the output layer consisting of q neurons. Considering an input function, $\boldsymbol{u}^{(i)}$ in the branch net, the network returns a feature embedding $[b_1, b_2, \ldots, b_q]^T$ as output. The output, $\boldsymbol{Z}_{br}^{l_{br}}$ of the feed-forward branch neural network is expressed as:

$$\begin{aligned} \boldsymbol{Z}_{br}^{l_{br}} &= \left[b_1, b_2, \ldots, b_q\right]^T, \\ &= \sigma_{br}\left(\mathbf{W}^{l_{br}} \boldsymbol{z}^{l_{br}-1} + \boldsymbol{\beta}^{l_{br}}\right), \end{aligned} \tag{8}$$

where $\sigma_{br}(\cdot)$ denotes the non-linear activation function for the branch net and $\boldsymbol{z}^{l_{br}-1} = f_{br}(\boldsymbol{u}^{(i)}(\boldsymbol{x}_1), \boldsymbol{u}^{(i)}(\boldsymbol{x}_2), \ldots, \boldsymbol{u}^{(i)}(\boldsymbol{x}_m))$, where $f_{br}(\cdot)$ denotes a function. Similarly, consider a trunk network with l_{tr} hidden layers, where the l_{tr}th layer is the output layer consisting of q neurons. The trunk net takes the continuous coordinates in $\boldsymbol{y}$ as inputs, and outputs a features embedding $[t_1, t_2, \ldots, t_q]^T$. The output of the trunk net can be represented as:

$$\begin{aligned} \boldsymbol{Z}_{tr}^{l_{tr}} &= \left[t_1, t_2, \ldots, t_q\right]^T, \\ &= \sigma_{tr}\left(\mathbf{W}^{l_{tr}} \boldsymbol{z}^{l_{tr}-1} + \boldsymbol{\beta}^{l_{tr}}\right), \end{aligned} \tag{9}$$

where $\sigma_{tr}(\cdot)$ denotes the non-linear activation function for the trunk net and $\boldsymbol{z}^{l_{tr}-1} = f_{tr}(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_p)$, where $\boldsymbol{y}_j = \{(\hat{x}_j, \hat{y}_j)\}_{j=1}^{p}$. For a single input function, $\boldsymbol{u}^{(i)}$, the DeepONet prediction, $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u})$, evaluated at any coordinate, $\boldsymbol{y}$ can be expressed as:

$$\begin{aligned} \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}^{(i)})(\boldsymbol{y}) &= \sum_{k=1}^{q} \left(\sigma_{br}(\mathbf{W}_k^{l_{br}} \boldsymbol{z}_k^{l_{br}-1} + \boldsymbol{\beta}_k^{l_{br}}) \cdot \sigma_{tr}(\mathbf{W}_k^{l_{tr}} \boldsymbol{z}_k^{l_{tr}-1} + \boldsymbol{\beta}_k^{l_{tr}})\right), \\ &= \sum_{k=1}^{q} b_k(\boldsymbol{u}^{(i)}(x_1), \boldsymbol{u}^{(i)}(x_2), \ldots, \boldsymbol{u}^{(i)}(x_m)) \cdot t_k(\boldsymbol{y}). \end{aligned} \tag{10}$$

In general, a DeepONet training dataset is a triplet of the form, $\left[\{\boldsymbol{u}^{(i)}\}_{i=1}^{n}, \{\boldsymbol{y}_j\}_{j=1}^{p}, \mathcal{G}(\boldsymbol{u})(\boldsymbol{y})\right]$:

$$\left[\begin{bmatrix} \boldsymbol{u}^{(1)}(\boldsymbol{x}_1), \boldsymbol{u}^{(1)}(\boldsymbol{x}_2), \ldots, \boldsymbol{u}^{(1)}(\boldsymbol{x}_m) \\ \boldsymbol{u}^{(1)}(\boldsymbol{x}_1), \boldsymbol{u}^{(1)}(\boldsymbol{x}_2), \ldots, \boldsymbol{u}^{(1)}(\boldsymbol{x}_m) \\ \vdots \\ \boldsymbol{u}^{(1)}(\boldsymbol{x}_1), \boldsymbol{u}^{(1)}(\boldsymbol{x}_2), \ldots, \boldsymbol{u}^{(1)}(\boldsymbol{x}_m) \\ \boldsymbol{u}^{(i)}(\boldsymbol{x}_1), \boldsymbol{u}^{(i)}(\boldsymbol{x}_2), \ldots, \boldsymbol{u}^{(i)}(\boldsymbol{x}_m) \\ \vdots \\ \boldsymbol{u}^{(n)}(\boldsymbol{x}_1), \boldsymbol{u}^{(n)}(\boldsymbol{x}_2), \ldots, \boldsymbol{u}^{(n)}(\boldsymbol{x}_m) \end{bmatrix}, \begin{bmatrix} \boldsymbol{y}_1^{(1)} \\ \boldsymbol{y}_2^{(1)} \\ \vdots \\ \boldsymbol{y}_p^{(1)} \\ \boldsymbol{y}_1^{(i)} \\ \vdots \\ \boldsymbol{y}_p^{(n)} \end{bmatrix}, \begin{bmatrix} \mathcal{G}(\boldsymbol{u}^{(1)})(\boldsymbol{y}_1^{(1)}) \\ \mathcal{G}(\boldsymbol{u}^{(1)})(\boldsymbol{y}_2^{(1)}) \\ \vdots \\ \mathcal{G}(\boldsymbol{u}^{(1)})(\boldsymbol{y}_p^{(1)}) \\ \mathcal{G}(\boldsymbol{u}^{(i)})(\boldsymbol{y}_1^{(i)}) \\ \vdots \\ \mathcal{G}(\boldsymbol{u}^{(n)})(\boldsymbol{y}_p^{(n)}) \end{bmatrix}\right]. \tag{11}$$

In the triplet shown in Eq. (11), each input function, $\boldsymbol{u}^{(i)}$ is repeated p times, where p is the number of points at which the solution operator, $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}^{(i)})$, is evaluated to construct the loss function. The training dataset of a DeepONet consists of three parts; input to the branch net, $\boldsymbol{u}$, input to the trunk net, $\boldsymbol{y}$, and the target values of the solution, $\mathcal{G}(\boldsymbol{u})(\boldsymbol{y})$. Taking a two-dimensional problem with scalar-valued input $\mathbf{u}(\boldsymbol{x})$ and output $\mathcal{G}(\boldsymbol{u})(\boldsymbol{y})$ for illustration, the tensor dimensions for each of the components of the training set are: $\dim(\boldsymbol{u}) := (n \times p, m)$, $\dim(\boldsymbol{y}) := (n \times p, 2)$, and $\dim(\mathcal{G}(\boldsymbol{u})(\boldsymbol{y})) := (n \times p, 1)$. DeepONet requires large annotated data-sets consisting of paired input–output observations, while they provide a simple and intuitive model architecture that is fast to train, allowing for a continuous representation of the target output functions that is independent of resolution. In Eq. (11), $\mathcal{G}(\boldsymbol{u})(\boldsymbol{y})$ represents the ground truth, which could be obtained either from experimental data or from high-fidelity simulations. Conventionally, the trainable parameters of the DeepONet represented by $\boldsymbol{\theta}$ in Eq. (10) are obtained by minimizing a loss function. Common loss functions used in literature include the l_2-loss function and the l_1-loss function [50].

$$\begin{aligned} l_1 &= \sum_{i=1}^{n}\sum_{j=1}^{p}\left|\mathcal{G}(\boldsymbol{u}^{(i)})(\boldsymbol{y}_j^{(i)}) - \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}^{(i)})(\boldsymbol{y}_j^{(i)})\right|, \\ l_2 &= \sum_{i=1}^{n}\sum_{j=1}^{p}\left(\mathcal{G}(\boldsymbol{u}^{(i)})(\boldsymbol{y}_j^{(i)}) - \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}^{(i)})(\boldsymbol{y}_j^{(i)})\right)^2, \end{aligned} \tag{12}$$

where $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}^{(i)})(\boldsymbol{y}_j^{(i)})$ is the predicted value obtained from the DeepOnet, while $\mathcal{G}(\boldsymbol{u}^{(i)})(\boldsymbol{y}_j^{(i)})$ is the target value.

DeepONet has shown remarkable success in diverse fields of applications like electro-convection multiphysics [51], bubble dynamics [52], etc., where the network is trained using large datasets. However, in fracture mechanics, collecting a large amount of data from experiments is improbable as performing experiments for various different crack lengths, locations, and material parameters would render the process very expensive. Moreover, it is infeasible to perform a rigorous and computationally intensive crack-growth simulation within the possible short time span following a discrete-source damage event. Hence, in a small data regime, we will encode the physical laws that govern the growth of fracture to train the DeepONet for predicting damage paths. Since the growth of fracture is energy-driven, we encode the variational form of the governing PDE into the DeepONet, terming it the variational energy-based DeepONet (V-DeepONet). Along with the physics of the problem, we use relatively small input–output datasets to improve the prediction accuracy of the network, thereby proposing a hybrid loss function. In the next section, we present the V-DeepONet algorithm to compute the optimized parameters, $\boldsymbol{\theta}^*$, associated with the two deep neural network architecture.

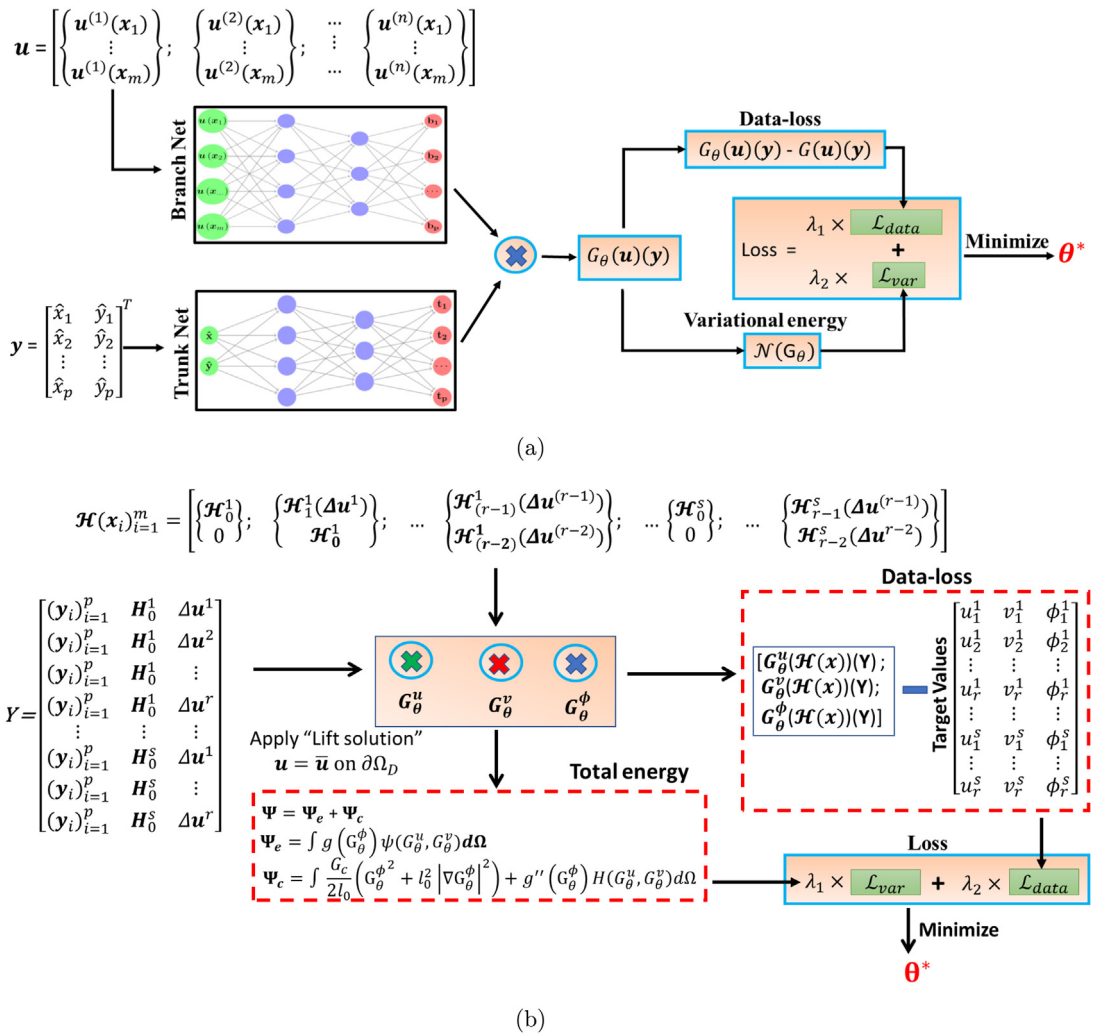
4. Variational energy based DeepONet

In this section, we develop the V-DeepONet, which is inspired by the variational form of the governing PDE. Without the loss of generality, we consider the physics of a problem, defined by a generic time-independent differential equation of the form:

$$\mathcal{K}\left(\boldsymbol{w}, \nabla\boldsymbol{w}, \ldots, \nabla^{\alpha}\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{f}\left(\boldsymbol{x}\right)\right) = 0, \quad \boldsymbol{x} \in \Omega, \tag{13a}$$

$$\boldsymbol{w}\left(\boldsymbol{x}_D\right) = \boldsymbol{w}_D, \quad \boldsymbol{x}_D \in \partial\Omega, \tag{13b}$$

defined over the physical domain, Ω with boundaries, $\partial\Omega$. $\nabla\boldsymbol{w}$ denotes the first order derivatives of $\boldsymbol{w}$ with respect to $\boldsymbol{x}$, and $\nabla^{\alpha}\boldsymbol{w}$ represents all derivatives of $\boldsymbol{w}$ with the form $\frac{\partial^{\alpha}\boldsymbol{w}_i}{\partial\boldsymbol{x}_1^{\alpha_1}\cdots\boldsymbol{x}_d^{\alpha_d}}$ with $\sum_{i=1}^{d}\alpha_i = \alpha$, where $\boldsymbol{w}_i$ is



(a)

(b)

Fig. 1. (a) Schematic representation of DeepONet. The branch net takes the function $\{\boldsymbol{u}^{(i)}\}_{i=1}^{n}$ as input evaluated at m fixed sensor points, denoted by $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$ and returns features embedding $[b_1, b_2, \ldots, b_q]^T \in \mathbb{R}^q$ as output. The trunk net takes the continuous coordinates and parameters, $\boldsymbol{y}_i = \{\hat{x}_i, \hat{y}_i\}_{i=1}^{p} \in \mathbf{Y}$ as inputs, and outputs a features embedding $[t_1, t_2, \ldots, t_q]^T \in \mathbb{R}^q$. The features embedding of the branch and trunk networks are merged via a dot product to output the solution operator, $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u})(\boldsymbol{y})$. The parameter $\boldsymbol{\theta}$ denotes the collection of all trainable weights and bias in the branch and the trunk network. The optimized parameters, $\boldsymbol{\theta}^*$, are obtained by minimizing the hybrid loss function defined as the weighted sum of variational energy, $\mathcal{L}_{var}$, and the data-driven loss, $\mathcal{L}_{data}$, where the weights are denoted by λ_1 and λ_2. (b) The computational framework of the proposed surrogate model is shown schematically. The input to the branch net is the tensile strain energy of previous two displacement steps ($\boldsymbol{\mathcal{H}}_{r-1}, \boldsymbol{\mathcal{H}}_{r-2}$), while the input to the trunk net is the evaluation coordinates, $\boldsymbol{y}$, the initial configuration described using Eq. (7), and the applied displacement. For predicting the crack located in the first displacement step, the input to the branch net is the initial strain energy, $\boldsymbol{\mathcal{H}}_0$, padded with zeros at all sensor locations to indicate no prior history.

the ith component of $\boldsymbol{w}$ and similarly for $\boldsymbol{x}_i$. α denotes the highest order of derivative required to describe the underlying PDE. The forcing function, $\boldsymbol{f}(\boldsymbol{x})$ is a known source term and the operator, $\mathcal{K}$ is usually a differential or integro-differential operator. Eq. (13b) represents the Dirichlet boundary condition, where $\boldsymbol{x}_D$ represents a Dirichlet boundary point. Since the method is based on the energy principle, the homogeneous Neumann boundary conditions are automatically satisfied. Assuming that $\boldsymbol{f}(\boldsymbol{x})$ is known, we aim to learn the solution operator such that:

$$\mathcal{G}_{\boldsymbol{\theta}} : f(\boldsymbol{x}) \rightarrow \boldsymbol{w}(\boldsymbol{x}). \tag{14}$$

Based on the formulation of DeepONet defined in Section 3, the input function of the branch net is the source term, $f(\boldsymbol{x})$, which is evaluated at $\{\boldsymbol{x}_i\}_{i=1}^{m}$ sensor points, where $\boldsymbol{x}_i \in \mathcal{X}$. The DeepONet is approximating the

solution operator, $\mathcal{G}_{\boldsymbol{\theta}}\left(\boldsymbol{f}\left(\boldsymbol{x}\right)\right)$, which is evaluated at a set of points, $\boldsymbol{y}_{j=1}^{p}$, that are randomly sampled in the domain of $G_{\boldsymbol{\theta}}\left(\boldsymbol{f}\left(\boldsymbol{x}\right)\right)$, and are used to approximately enforce a set of given physical constraints, typically described by the PDE in Eq. (13a). Let the variational energy formulation of Eq. (13a) be expressed as:

$$\mathcal{V}_e = \int_{\Omega} \mathcal{F}\left(\boldsymbol{w}, \nabla \boldsymbol{w}, \ldots, \nabla^{\alpha} \boldsymbol{w}, \boldsymbol{x}, \boldsymbol{f}\left(\boldsymbol{x}\right)\right) d\Omega, \tag{15}$$

where $\mathcal{F}$ is a differentiable functional. With this, the solution to Eq. (13a) can be obtained by solving the following optimization problem:

$$\begin{aligned} \boldsymbol{w}^{*} &= \underset{w}{\operatorname{argmin}}\, \mathcal{V}_e\left(\mathcal{G}(\boldsymbol{f}(\boldsymbol{x}))\right) \\ \text{subject to: } & \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{f}(\boldsymbol{x}))\left(\boldsymbol{y} = \boldsymbol{x}_D\right) = \boldsymbol{w}_D. \end{aligned} \tag{16}$$

In V-DeepONet, we utilize the same approach as discussed in Eq. (10) to obtain the solution operator, $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{f}(\boldsymbol{x}))$, with parameters, $\boldsymbol{\theta} = [\mathbf{W}, \boldsymbol{\beta}]$. Next, the DeepONet outputs are modified in such a way so that the solution operator when evaluated at the Dirichlet boundary points, the boundary conditions are exactly satisfied. Therefore, the modified output of the DeepONet is defined as:

$$\mathcal{G}_{\boldsymbol{\theta}}(f(\boldsymbol{x}))\left(\boldsymbol{y}\right) = \tilde{\boldsymbol{w}}_D + B(\boldsymbol{y}) \cdot \hat{\mathcal{G}}_{\boldsymbol{\theta}}(\boldsymbol{f}(\boldsymbol{x}))\left(\boldsymbol{y}\right), \tag{17}$$

where $\hat{\mathcal{G}}_{\boldsymbol{\theta}}$ is the solution obtained from the DeepONet, $\tilde{\boldsymbol{w}}_D$ is a function chosen such that $\tilde{\boldsymbol{w}}_D = \boldsymbol{w}_D$ and $B(\boldsymbol{y}) = 0$ on the Dirichlet boundary [33]. Hence, the boundary conditions are satisfied and we have no boundary-loss term in the loss function. In the next step, we compute the derivatives of the solution operator, $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{f}(\boldsymbol{x}))\left(\boldsymbol{y}\right)$, with respect to the spatial co-ordinates, $\left(\hat{x}, \hat{y}\right)$, defined over the domain using the automatic differentiation technique. These derivatives are components of the variational energy formulation stated in Eq. (15). The computed derivatives are substituted in Eq. (15) along with the solution operator approximating $\boldsymbol{w}(\boldsymbol{x})$ to obtain the total energy. The network parameters can be trained by minimizing the hybrid loss function, $\mathcal{L}\left(\boldsymbol{\theta}\right)$, which is defined as:

$$\begin{aligned} \mathcal{L}\left(\boldsymbol{\theta}\right) &= \lambda_1 \times \mathcal{L}_{data}\left(\boldsymbol{\theta}\right) + \lambda_2 \times \mathcal{L}_{var}\left(\boldsymbol{\theta}\right), \\ \mathcal{L}_{data}\left(\boldsymbol{\theta}\right) &= \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p}\left(\mathcal{G}_{\boldsymbol{\theta}}(f^{(i)}(\boldsymbol{x}_j))(\boldsymbol{y}_k) - \mathcal{G}(f^{(i)}(\boldsymbol{x}_j))(\boldsymbol{y}_k)\right)^2}{n \times p \times m}, \\ \mathcal{L}_{var}\left(\boldsymbol{\theta}\right) &= \frac{\sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{k=1}^{m}\mathcal{F}\left(\boldsymbol{w}, \nabla \boldsymbol{w}, \ldots, \nabla^{\alpha} \boldsymbol{w}, \boldsymbol{x}, f\left(\boldsymbol{x}\right)\right)}{n \times p \times m}, \end{aligned} \tag{18}$$

where $\boldsymbol{w}\left(\boldsymbol{x}\right) \approx \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{f}^{(i)}(\boldsymbol{x}_j))$ and λ_1 and λ_2 are the weights pertaining to each of the component of the loss function. The developed approach seamlessly integrates the data measurements and variational form of the PDE by penalizing the total energy in the hybrid loss function of the V-DeepONet.

In this work, we aim to train the V-DeepONet such that it could be used as a surrogate model to predict the damage path for any given initial defect and for any applied displacement (considering displacement-controlled fracture). To design a surrogate model in the framework of V-DeepONet, we need to choose the input functions, sensor points, parametrized variables judiciously and strike a balance between efficiency and accuracy of the network.

5. Surrogate modeling for fracture analysis

In this section, we provide a detailed description of the proposed surrogate model to infer the probable failure paths. The solution of the coupled problem consists of a d-dimensional vector-valued elastic field and a scalar-valued phase field output. In all the examples presented in this paper, we have considered two-dimensional problems, hence $d = 2$. Therefore, for each material point $\boldsymbol{y}$, the output of the V-DeepONet has three components. Noticing that the output in the vanilla DeepONet Eq. (11) is a scalar, the neural network output architecture needs to be modified. To resolve this issue, we modify the forward pass in Eq. (10) such that the output of the V-DeepONet can be a vector. The tensor dimensions of $\mathcal{G}(\boldsymbol{u})(\boldsymbol{y})$ in Eq. (11) will be $(n \times p, 3)$. The employed V-DeepONet is defined as $\mathcal{G}_{\boldsymbol{\theta}} = \left[\mathcal{G}_{\boldsymbol{\theta}}^{u}, \mathcal{G}_{\boldsymbol{\theta}}^{v}, \mathcal{G}_{\boldsymbol{\theta}}^{\phi}\right]$ to represent the solution map from initial conditions to the associated solutions, where $\mathcal{G}_{\boldsymbol{\theta}}^{u}$, $\mathcal{G}_{\boldsymbol{\theta}}^{v}$ are the solution for the displacement components along x-direction and y-direction, respectively and $\mathcal{G}_{\boldsymbol{\theta}}^{\phi}$ is the solution for the phase field, ϕ.

The goal of the V-DeepONet based surrogate model is to find the possible pathways of failure based on the initial crack configuration. Failure of brittle materials typically starts with the development of a region of high stress around the crack tip, and our method will be employed based on the locations of the regions of high-stress. In this method, we assume the presence of initial defects when the system is not loaded (initial stage). The initial history function defined in Eq. (7) provides a simple, mesh independent mechanism for adding initial defects to a model. Using this approach, the initial crack is identified by a region of very high strain energy, whereas the strain energy is close to zero in other parts of the domain. The proposed method focuses on displacement-controlled failure. As the system is loaded (prescribed displacement is increased incrementally), pre-existing cracks grow leading to the formation of new edges and finally leading to the failure of the system.

To understand how the surrogate model works, we divide the aim of the surrogate model into two sub-parts in order:

- To predict the crack location and the deformed configuration of the domain for any applied displacement, given a fixed initial condition.
- To predict the final crack path for any location of the initial defect, given a fixed applied displacement.

In this section, we will develop independent surrogate models to meet each of the two goals mentioned above. The final surrogate model will then be engineered as an integration of the above two surrogate models. In Section 5.1, we focus on constructing a surrogate model to predict the crack location for any applied displacement, when the domain geometry and the initial condition are fixed. Then in Section 5.2, we will discuss the surrogate model for predicting the crack path for any pre-crack location, with a fixed displacement loading applied on the boundary.

5.1. Surrogate 1: To predict the crack location for any applied displacement

In this section, we present our first contribution for solving displacement-controlled quasi-static fracture problem. At each step, we update the displacement boundary condition with an increment as $\boldsymbol{w}_D \rightarrow \boldsymbol{w}_D + \Delta\boldsymbol{w}$, then solve the nonlinear optimization problem defined in Eq. (16) with this new boundary condition applied. In conventional numerical solvers, small displacement increments $\Delta\boldsymbol{w}$ are essential to capture the brutal nature of crack growth. V-PINNs, which were proposed in [28], successfully implemented the crack growth problem using larger displacement increments and enabled a significant reduction in the computational cost. However, there is a major bottleneck of V-PINNs: the fracture problem is sequentially solved as a short loading-step problem with independent neural network training for each loading step. Even though the concept of transfer learning was implemented in the framework of V-PINNs, the method still remained computationally expensive.

In this work, we train a single V-DeepONet to learn the solution operator of the same PDE for small displacement steps subject to same initial condition. The steps involved in the proposed approach are as follows:

- First, we decide the location of the m sensor points to distinguish two input functions (defined in terms of tensile strain energy) of the branch net.
- Next, for training the V-DeepONet, we consider n steps with corresponding displacement increments, denoted by $\Delta\boldsymbol{w}_1, \Delta\boldsymbol{w}_2, \ldots, \Delta\boldsymbol{w}_n$. Therefore, the input to the branch net will be tensile strain energy computed for n displacements steps evaluated at m sensor locations. The input function corresponding to the displacement step $\Delta\boldsymbol{w}_i$ will be the tensile strain energy due to an applied displacement $\Delta\boldsymbol{w}_{i-1}$, where $i \in \{2, \ldots, n\}$. Here, the tensile strain energy is obtained from the results of high-fidelity IGA simulations. To predict the solution fields for an applied displacement of $\Delta\boldsymbol{w}_1$, the tensile strain energy computed using Eq. (7) at the sensor locations is applied as the input to the branch net. The solution operators are defined such that:
$$\left[\mathcal{G}^u_{\boldsymbol{\theta}}, \mathcal{G}^u_{\boldsymbol{\theta}}, \mathcal{G}^\phi_{\boldsymbol{\theta}}\right] : \boldsymbol{\mathcal{H}}(\boldsymbol{y}) \rightarrow [u(\boldsymbol{y}), v(\boldsymbol{y}), \phi(\boldsymbol{y})]\,, \tag{19}$$
where $\boldsymbol{\mathcal{H}}(\boldsymbol{y})$ is the tensile strain energy, and u, v denote the x- and y-component of the elastic field, respectively.
- In the third step, we prepare the inputs to the trunk net for extracting latent representations of the input coordinates. For each displacement step $\Delta\mathbf{u}_i$, $\{\boldsymbol{y}^{(i)}_j\}^p_{j=1}$ are a set of points sampled in the domain, Ω. The output function is evaluated at $\boldsymbol{y}^{(i)}_j$ for an applied displacement, $\Delta\boldsymbol{w}_i$, where $i \in \{1, \ldots, n\}$.
- Next, we modify the elastic field outputs of the V-DeepONet, $\mathcal{G}^u_{\boldsymbol{\theta}}$ and $\mathcal{G}^v_{\boldsymbol{\theta}}$ as discussed in Eq. (17) so that the Dirichlet boundary conditions are satisfied.

- Finally, we construct the hybrid loss function as defined in Eq. (18) and minimize the loss to obtain the optimized parameters, $\boldsymbol{\theta}^*$.

Once the V-DeepONet is trained, it can be used to predict the elastic field and the phase field for any applied displacement on a fixed initial condition. In order to predict the solution of the V-DeepONet at the kth displacement step, the branch net takes as input the tensile energy at the sensor points obtained from the $(k-1)$th displacement step. To obtain the tensile energy, we compute the displacement gradients, and the eigenvalues of the strain, λ_1^E and λ_2^E using the outputs of the V-DeepONet obtained for the $k-1$th step at the sensor locations. The computed eigenvalues are then used to obtain Ψ^+ and Ψ^-.

$$\Psi^+ = \frac{\nu}{8}\left(\lambda_s + |\lambda_s|\right)^2 + \frac{\mu}{4}\sum_{i=1}^{d}\left(\lambda_i^E + \left|\lambda_i^E\right|\right)^2, \tag{20a}$$

$$\Psi^- = \frac{\nu}{8}\left(\lambda_s - |\lambda_s|\right)^2 + \frac{\mu}{4}\sum_{i=1}^{d}\left(\lambda_i^E - \left|\lambda_i^E\right|\right)^2, \tag{20b}$$

where $\lambda_s = \sum_{i=1}^{d}\lambda_i^E$, and ν and μ are the Lamé constants. The Ψ^+ values at m sensor locations are the input to the branch net for the kth displacement step. To provide an implementation guidance for interested readers, in the following we provide tensor dimensions of the training set, in this surrogate, for each of the components in the triplet discussed in Eq. (11). In the triplet, $\dim(\boldsymbol{u}) := (n \times p, m)$ corresponding to n displacement steps, p sampled points for the evaluation of the output function and m fixed sensor points, $\dim(\boldsymbol{y}) := (n \times p, 3)$, where the first two columns correspond to the spatial location of the evaluation points and the third column corresponds to the applied displacement. In particular, the displacement increment in the principle direction is employed in as the third column of $\boldsymbol{y}$, as will be explained further in Section 7. Lastly, $\dim(\mathcal{G}(\boldsymbol{u})(\boldsymbol{y})) := (n \times p, 3)$, where the three columns are designated for the solution of u, v, and ϕ, respectively.

Having built the surrogate model for predicting the crack location for any applied displacement on a fixed initial condition, we now extend the context to training the surrogate model adequately over a wide range of potential crack starting locations.

5.2. Surrogate 2: To predict the final crack path for any initial crack location

In this section, we train the V-DeepONet adequately over multiple initial conditions (initial location of the defects), to obtain the final crack path at fixed applied displacement, $\Delta\boldsymbol{w}$. This surrogate model defines a mapping between the initial configuration defined using $\boldsymbol{\mathcal{H}}_0(\boldsymbol{y})$ to the solution fields for a fixed applied displacement, $\Delta\boldsymbol{w}$. The steps involved in the proposed approach are as follows:

- First, we identify m sensor locations that could be used for adequate and distinct identification of different initial configuration. As mentioned previously, the location of the sensors must be the same for all input functions in the branch net.
- Next, we consider n initial locations of the crack. For all the initial configurations, we compute the initial strain energy, $\{\boldsymbol{\mathcal{H}}_0^{(1)}, \boldsymbol{\mathcal{H}}_0^{(2)}, \ldots, \boldsymbol{\mathcal{H}}_0^{(n)}\}$ at the fixed sensor locations using Eq. (7). The computed $\boldsymbol{\mathcal{H}}_0^{(i)}$ is the input function to the branch net. The tensor dimension of the input to the branch net will be $(n \times p, m)$. The solution operators are defined such that:

$$\left[\mathcal{G}_{\boldsymbol{\theta}}^u, \mathcal{G}_{\boldsymbol{\theta}}^u, \mathcal{G}_{\boldsymbol{\theta}}^\phi\right] : \boldsymbol{\mathcal{H}}_0(\boldsymbol{y}) \rightarrow [u(\boldsymbol{y}), v(\boldsymbol{y}), \phi(\boldsymbol{y})]\,. \tag{21}$$

- In the third step, we prepare the inputs to the trunk net for encoding the locations of the output function. For each of the initial condition, $\{\boldsymbol{y}_j^{(i)}\}_{j=1}^{p}$ is a set of points sampled in the domain, Ω. The initial strain energy function is computed at the sampled points and is represented by $\boldsymbol{H}_j^{(i)}$. The tensor dimensions of the input to the trunk net will be $(n \times p, 3)$, where the first two columns correspond to the spatial location of $\boldsymbol{y}_j$, and the third column provides the strain energy, $\boldsymbol{H}_j$, at the respective locations. The V-DeepONet is evaluated for n initial conditions at all the sampled points in the trunk net.
- In the next step, the V-DeepONet outputs are modified to eliminate the boundary loss term from the loss function. Finally, the hybrid loss function is obtained and minimized to get the optimized parameters of the V-DeepONet.

Once this surrogate model is trained, it can be used to predict the elastic field and the phase field for any initial crack location at a fixed applied displacement. While testing the surrogate model, the analytically obtained history field (using Eq. (7)) at fixed sensor locations is employed as inputs to the branch net, while the evaluation coordinates and the initial strain energy (corresponding to the evaluation points) provide inputs of the trunk net.

Having discussed the surrogate models for crack locations at various displacement steps (with fixed initial condition) and final crack path for any initial crack location (with fixed applied displacement), we now shift the focus to develop a single surrogate model that can be trained to obtain the crack location at any applied displacement for any initial location of the crack.

5.3. A unified model: To predict the crack location for any initial condition and any applied displacement

Our final aim is to build a surrogate model such that the crack path can be traced from the initial configuration of the defect. This surrogate model would primarily be an integration of the previously discussed surrogate models. However, building such a surrogate model has two major challenges:

- In Section 5.1, to obtain the crack location at any given displacement step, we used the tensile strain energy only from the previous step. This approach is feasible only when a single initial crack configuration is considered. For multiple crack locations, there can be overlap of crack paths, and hence retaining the information of the initial crack location and/or multiple previous steps is essential.
- To obtain the crack path for any initial location at any applied displacement, training the model using just the strain-energy of the previous displacement step (Surrogate 1) or the energy field in the vicinity of the original defect (Surrogate 2) is not sufficient, as the sequence or the original configuration might be lost, thereby leading to erroneous predictions.

To resolve these challenges, we need to construct a V-DeepONet based surrogate model, which is capable of learning order dependencies in a sequence of prediction problems. Both these challenges will be addressed in the surrogate model, as will be discussed in this section. The following steps are considered for constructing the unified surrogate model:

- Choice of the m sensor locations: The location of the sensors should be carefully chosen such that they are representative of various initial configurations as well as crack locations at various applied displacement steps. It is worth emphasizing that the input function space should be large enough to cover as many potential states of the underlying PDE system as possible. Otherwise, the trained model may not generalize very well for out-of-distribution initial conditions, possibly leading to large errors or even erroneous predictions.
- Next, we consider s initial locations of the crack. For each initial condition, we compute the initial strain energy, $\{\boldsymbol{\mathcal{H}}_0^{(1)}, \boldsymbol{\mathcal{H}}_0^{(2)}, \ldots, \boldsymbol{\mathcal{H}}_0^{(s)}\}$, at the sensor locations using Eq. (7). Now, for each initial crack configuration, we consider r displacement steps, $\{\Delta\boldsymbol{w}_1, \Delta\boldsymbol{w}_2, \ldots, \Delta\boldsymbol{w}_r\}$. Corresponding to every initial condition, we obtain the tensile strain energy, $\boldsymbol{\mathcal{H}}_i^{(j)}$, from the results of the IGA simulations, corresponding to an applied displacement $\Delta\boldsymbol{w}_i$, where $i \in \{1, \ldots, r-1\}$ and $j \in \{1, \ldots, s\}$.
- In the third step, we prepare the inputs for the branch net. To capture each sequential crack growth for r applied displacements, we create a window of two steps, such that:

$$\boldsymbol{\mathcal{H}} = \left[\begin{bmatrix}\boldsymbol{\mathcal{H}}_0^{(1)}\\ 0\end{bmatrix}; \quad \begin{bmatrix}\boldsymbol{\mathcal{H}}_1^{(1)}\\ \boldsymbol{\mathcal{H}}_0^{(1)}\end{bmatrix}; \quad \ldots, \quad \begin{bmatrix}\boldsymbol{\mathcal{H}}_{r-1}^{(1)}\\ \boldsymbol{\mathcal{H}}_{r-2}^{(1)}\end{bmatrix}; \quad \begin{bmatrix}\boldsymbol{\mathcal{H}}_0^{(2)}\\ 0\end{bmatrix}; \ldots \quad ; \begin{bmatrix}\boldsymbol{\mathcal{H}}_{r-1}^{(s)}\\ \boldsymbol{\mathcal{H}}_{r-2}^{(s)}\end{bmatrix}\right]. \tag{22}$$

  The solution operators are defined as:

$$\left[\mathcal{G}_{\boldsymbol{\theta}}^{u}, \mathcal{G}_{\boldsymbol{\theta}}^{v}, \mathcal{G}_{\boldsymbol{\theta}}^{\phi}\right] : \boldsymbol{\mathcal{H}}(\boldsymbol{y}) \rightarrow [u(\boldsymbol{y}), v(\boldsymbol{y}), \phi(\boldsymbol{y})], \tag{23}$$

  where $\boldsymbol{\mathcal{H}}(\boldsymbol{y})$ is defined in Eq. (22). The tensor dimensions of the input to the branch net will be $(r \times s \times p, 2\,m)$, where the $r \times s = n$, the total number of input functions.
- In the next step, we prepare the input to the trunk net. To address the issue of retaining the original crack configuration for the associated branch net entries, the initial strain energy is included in the trunk net. For each initial crack location $\{\boldsymbol{\mathcal{H}}_0^{(i)}\}_{i=1}^{s}$, $\{\boldsymbol{y}_j^{(i)}\}_{j=1}^{p}$ is a set of points sampled in the domain, Ω. The initial history function for s initial conditions is computed at the sampled points and denoted as $\boldsymbol{H}_j^{(i)}$. The tensor dimensions

of the input to the trunk net will be $(n \times p, 4)$, where the first two columns correspond to $\boldsymbol{y}_j$, the third column is allocated for the history value, $\boldsymbol{H}_j$, and the fourth column is for the applied displacement. The V-DeepONet is evaluated at the locations and conditions defined in the trunk net.

- In the final step, the V-DeepONet outputs are modified to match the Dirichlet boundary condition. Then, the hybrid loss function is constructed and minimized to obtain the optimized parameters of the networks.

Once this model is trained, it can be used to make predictions for any initial configuration at various applied displacements sequentially. To predict crack paths using this model, we follow similar steps as discussed in Section 5.1, except that in this surrogate we provide the strain energy of the last two displacements steps at the sensor location as the input to the branch net. Fig. 1(b) provides a pictorial description of the framework and implementation details of the surrogate model to obtain the failure paths. For more clarity, an algorithm summarizing the overall framework is presented in Algorithm 1 of Appendix A.

In this work, we focus on brittle fracture problems under quasi-static loading, where crack branching is a rare phenomenon. As natural extensions, more fracture problems can be considered, e.g., dynamic fracture problems, ductile fracture problems, and problems with complex geometries. For problems under dynamic loading, the current V-DeepONet framework can be easily extended to take into account the crack branching phenomenon. However, such cases would require a much larger number of sensors, to take into account the location of sharp crack tips at multiple locations. To extend this framework for ductile fracture, only the governing equations need to be changed, while considering much lesser number of sensor locations, since the crack tip is not as sharp as in brittle fracture. To extend the framework for a complex geometry is also straightforward; the branch net takes the energy field as input to represent complex features as well as multiple initial crack locations. However, a large number of sensors would be required to represent multiple initial crack locations.

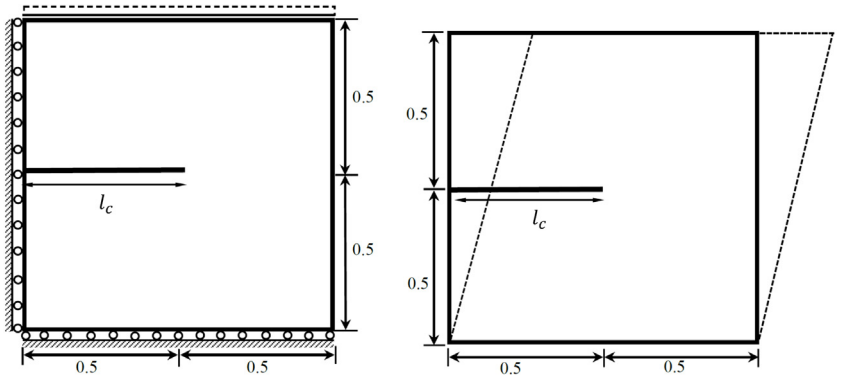
6. Data generation

In Section 5, we presented details about the proposed surrogate model developed within the framework of V-DeepONet. The model is trained using a hybrid loss function, which is the weighted sum of energy loss (using the variational form of the governing PDE) and data-driven loss. The labeled high-fidelity datasets for constructing the data-driven loss are simulated using phase field codes developed within the framework of isogeometric analysis (IGA). In this regard, we have used the codes developed in [42], which are available in https://github.com/somdattagoswami/IGAPack-PhaseField. However, we note that the proposed approach in this article can also be applied to more general training datasets. For instance, one may also consider a V-PINNs algorithm to generate high-fidelity datasets. In our work, we have not considered any experimental data. However, if available, the model can incorporate experimental data to improve the accuracy of the surrogate. To verify the performance of the V-DeepONet based surrogate model, we compare the predicted failure paths to those computed with the high-fidelity fracture mechanics model.

7. Simulation results

In this section, we explore our method using two benchmark problems from fracture mechanics to demonstrate the effectiveness of the developed surrogate model. In the first problem, we have studied the growth of cracks in a plate under tensile loading (Mode-I failure), while the second one is the growth of cracks under shear loading (Mode-II failure). For both the problems we have shown the efficiency of the surrogate model to make out-of-distribution predictions. In all examples, the V-DeepONet is trained using the Adam optimizer [53]. The implementation has been carried out using the `PyTorch` framework [54]. Throughout all examples, we will employ the hyperbolic tangent activation function (`tanh`) and initialize the V-DeepONet parameters using Xavier initialization. Details on the network architecture, such as number of layers, number of neurons in each layer are provided with each example.

Even though the paper focuses on surrogate modeling for predicting brittle failure, we have included a pedagogical problem to show that the proposed V-DeepONet can be used to study general problems with discontinuity. To demonstrate this, we have studied the flow in heterogeneous porous media on a two-dimensional plate with geometric discontinuity. Interested readers may refer to Appendix B.



(a) Tensile loading. (b) Shear loading.

Fig. 2. Geometrical setup and boundary conditions for the single-edge notched plate are subjected to different loading conditions [42]. All the units are in mm. The crack length, l_c and location shown in the setup diagram is representative of the presence of initial defects. For both the problems, we assume plane strain conditions.

7.1. Brittle fracture in a plate loaded in tension

In the first example, we consider a unit square plate with a horizontal crack at the middle height from the left outer edge. The geometrical setup and the boundary conditions of the problem are shown in Fig. 2(a). The material properties considered are: $\nu = 121.15$ kN/mm^2, $\mu = 80.77$ kN/mm^2, as Lamé's first and second parameters, respectively, and $G_c = 2.7 \times 10^{-3}$ kN/mm. For this problem, we have considered the length scale parameter, $l_0 = 0.0625$ mm. The Dirichlet boundary conditions are:

$$u(0, y) = v(x, 0) = 0, \quad v(x, 1) = \Delta v, \tag{24}$$

where u and v are the solutions of the elastic field in x and y-directions, respectively and Δv is the applied tensile displacement. In this example, we will train the V-DeepONet model using $s = 6$ initial conditions (by changing the crack lengths, l_c, in Fig. 2(a)) and consider $r = 7$ displacement steps for obtaining the crack path. The six initial configurations used for training the network are $l_c \in \{0.25, 0.3, 0.35, 0.4, 0.45, 0.55\}$ mm, while the considered displacement steps are $\Delta v \in \{1.4, 2.2, 3.2, 4.4, 5, 5.6, 5.8\} \times 10^{-3}$ mm. To build the surrogate model, $m = 212$ sensors are chosen close to the cracked region, to distinctly represent each of the function in the input space of the branch net. We compute the initial strain energy, $\mathcal{H}_0^{(i)}$, $i \in \{1, \ldots, s\}$ using Eq. (7) at m sensor points. Now, from the high-fidelity data obtained using IGA simulations, we obtain the tensile strain-energy, $\mathcal{H}_j^{(i)}$ for the applied displacements $\Delta \boldsymbol{w}_j$, where $j \in \{1, \ldots, r-1\}$. The input data for the branch net, $\mathcal{H}(\boldsymbol{x})$ is prepared in the same way as discussed in Eq. (22). In this example, we aim to learn the solution operators, $\mathcal{G}_{\boldsymbol{\theta}}^u, \mathcal{G}_{\boldsymbol{\theta}}^v, \mathcal{G}_{\boldsymbol{\theta}}^\phi$ mapping $\mathcal{H}(\boldsymbol{y})$ to the solutions $u(\boldsymbol{y})$, $v(\boldsymbol{y})$, and $\phi(\boldsymbol{y})$. To this end, we represent the operator by a DeepONet, where both the branch net and the trunk net are 5-layer fully-connected neural networks with 50 neurons per hidden layer and equipped with `tanh` activations. For each input function in the branch net, samples are provided on $p = 1372$ points in the domain. Once the solution is evaluated at the sampled points, the V-DeepONet outputs for the elastic field are lifted to exactly match the Dirichlet boundary conditions, following:

$$\begin{aligned} \mathcal{G}_{\boldsymbol{\theta}}^u &= [x(1-x)]\hat{\mathcal{G}}_{\boldsymbol{\theta}}^u, \\ \mathcal{G}_{\boldsymbol{\theta}}^u &= [y(y-1)]\hat{\mathcal{G}}_{\boldsymbol{\theta}}^v + y\Delta v, \end{aligned} \tag{25}$$

where $\hat{\mathcal{G}}_{\boldsymbol{\theta}}^u$ and $\hat{\mathcal{G}}_{\boldsymbol{\theta}}^v$ are obtained from the neural network. The trainable parameters, $\boldsymbol{\theta}$ of the V-DeepONet are obtained by minimizing the hybrid loss function in Eq. (18). Now, we test the model to predict the solutions for $l_c = 0.5$ mm. To do this, we compute the initial history function, $\mathcal{H}_0$, analytically using Eq. (7) at the sensor locations and also at the sampled points where the solution has to be evaluated. The input to the branch net of the trained model is $[\mathcal{H}_0, 0]$, while the input to the trunk net is the sampled points, the initial history field at the points sampled in the domain and Δv_1. The surrogate model predicts the solution of the elastic field, u, v and the phase field,

ϕ corresponding to Δv_1. In addition, the network predicts the tensile strain energy for the applied displacement, which is denoted as $\mathcal{H}_1$. To predict the solution for an applied displacement of Δv_2, the input to the branch net is $[\mathcal{H}_1, \mathcal{H}_0]$, while the input to the trunk net remains the same except for the applied displacement, which is changed to Δv_2. In a similar way the solution for any crack length at any displacement step can be obtained sequentially. Fig. 3 shows the predicted crack path and the displacement component in y-axis for some displacement steps, considering $l_c = 0.5$ mm. In Fig. 4(a), we have presented the predicted displacement component in y-axis against the ground truth at two locations along x-axis for three displacement steps. The averaged error of predicted ϕ is 0.63%.

It is worth investigating that the trained V-DeepONet is capable of yielding accurate predictions for out-of-distribution test data. To illustrate this, we have tested the trained model for predicting the crack path and displacement components for $l_c = 0.65$, which is beyond the crack distribution length range $l_c \in [0.25, 0.55]$ in our training dataset. In Fig. 4(b), we have presented a comparison of the predicted displacement in y-axis against the ground truth at $x = 0.7208$ mm and $x = 0.9542$ mm for 3 displacement steps. For the out-of-distribution prediction, an averaged error of 1.85% on ϕ is obtained. For interested readers, the predicted solutions obtained using the V-DeepONet based surrogate model, at certain displacement steps, are shown in Fig. 13 of Appendix C, where we have compared the predicted results with those simulated using the high fidelity model and the point-wise errors are also plotted. The force–displacement curve obtained using the surrogate model for $l_c = 0.5$ mm is shown in Fig. 5.

7.2. Brittle fracture in a plate loaded in shear

In this example, we investigate the same square plate as stated in Section 7.1 for a pure shear loading mode. The geometrical setup and the boundary conditions of the problem are shown in Fig. 2(b). We consider the same material parameters as used in Section 7.1. The Dirichlet boundary conditions are[1]:

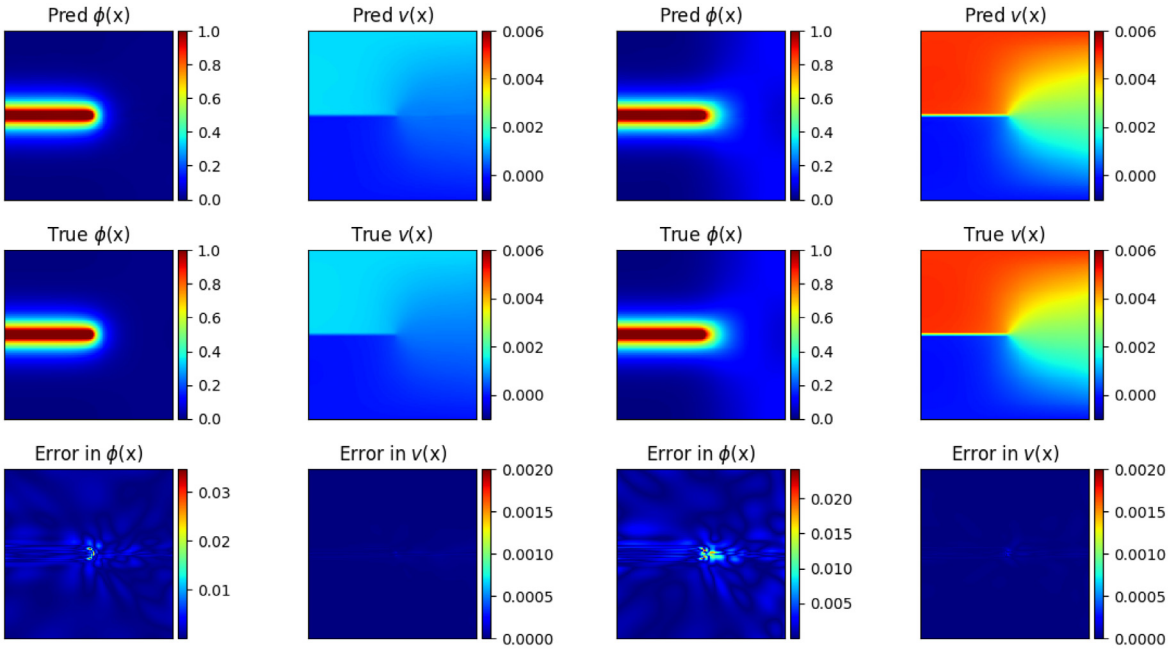
$$u(x, 0) = v(x, 0) = 0, \quad u(x, 1) = \Delta u, \tag{26}$$

where u and v are the solutions of the elastic field in x and y-axis, respectively and Δu is the applied shear displacement on the top edge of the plate. In this example, we have trained the V-DeepONet using $n = 85$ initial crack locations, with aiming to predict the final crack path for any initial crack location in the domain. The training sample consists of crack lengths, $l_c \in [0.3, 0.65]$ and the height of the crack varied between $[0.2, 0.675]$. Fig. 6 presents the predicted solution of the elastic field and the phase field for two samples. The predicted ϕ has a relative error of 0.67%.

Now, we have carried out multiple comprehensive studies to depict the versatility of the proposed surrogate model. In the first experiment, we compare the accuracy and efficiency of V-DeepONet with solely data driven DeepONet [40]. To that end, we consider changing the crack length, l_c, while fixing the height of the crack at the middle of the left outer edge, and predict the final damage path. To train the surrogate model we have used $n = 11$ samples with initial crack lengths, $l_c \in [0.2, 0.7]$ in steps of 0.05. For this experiment, we have used the surrogate model as discussed in Section 5.2 and have considered $m = 934$ sensors, closely placed around the region of high stresses. The initial strain energy, $\mathcal{H}(\boldsymbol{y}_j)$, where $j \in \{1, \ldots, m\}$, is computed using Eq. (7) for all the l_c, which is used as input to the branch net. For the trunk net, $p = 6024$ points are sampled in the domain. The input to the trunk net is a tensor of dimension $(n \times p, 3)$, where the first two columns contain the spatial locations of the sampled points, while the third column corresponds to the initial strain energy computed at the sampled points for each of the n cases. In this experiment, we aim to learn the solution operators, $\mathcal{G}^u_{\boldsymbol{\theta}}, \mathcal{G}^u_{\boldsymbol{\theta}}, \mathcal{G}^{\phi}_{\boldsymbol{\theta}}$ so as to map $\mathcal{H}(\boldsymbol{y})$ to their solutions $u(\boldsymbol{y})$, $v(\boldsymbol{y})$, and $\phi(\boldsymbol{y})$ for predicting the final damage path considering $\Delta u = 1.2 \times 10^{-2}$ mm. To this end, we represent the operators by a V-DeepONet, where both the branch net and the trunk net are 4-layers fully-connected neural networks with $[100, 50, 50, 50]$ neurons, respectively. Once the solution is evaluated at the sampled points, the outputs for the elastic field are modified to exactly satisfy the Dirichlet boundary conditions, as:

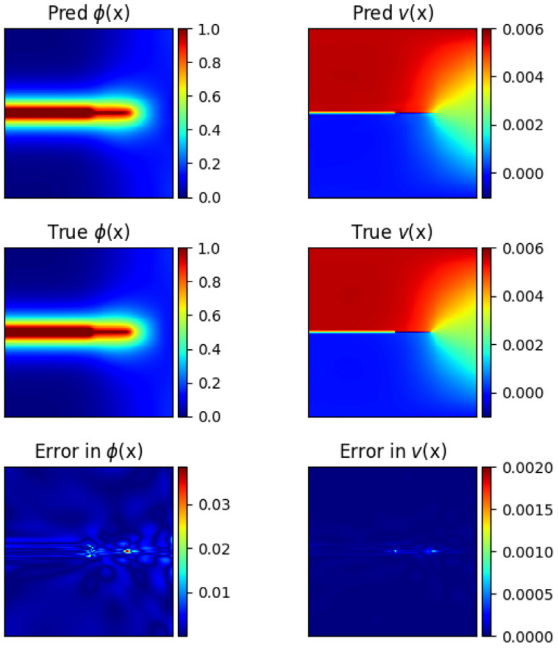
$$\begin{aligned} \mathcal{G}^u_{\boldsymbol{\theta}} &= [y(1 - y)]\hat{\mathcal{G}}^u_{\boldsymbol{\theta}} + y\Delta u, \\ \mathcal{G}^u_{\boldsymbol{\theta}} &= [y(y - 1)] \times [x(x - 1)]\hat{\mathcal{G}}^v_{\boldsymbol{\theta}}, \end{aligned} \tag{27}$$

[1] In some high-fidelity phase-field solvers, additional constraints have been applied over ϕ in the shear problem, such that $\phi(x, 0) = 0$ (see, e.g., [55,56]). For simplicity, in our high-fidelity numerical solver we have not applied this boundary condition but it might be considered in future work.

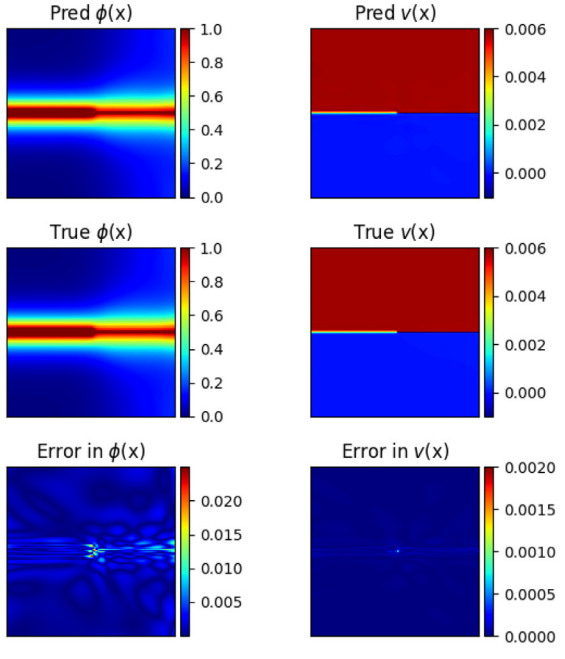


(a) $\Delta v_1 = 1.4 \times 10^{-3}$ mm.

(b) $\Delta v_5 = 5.0 \times 10^{-3}$ mm.

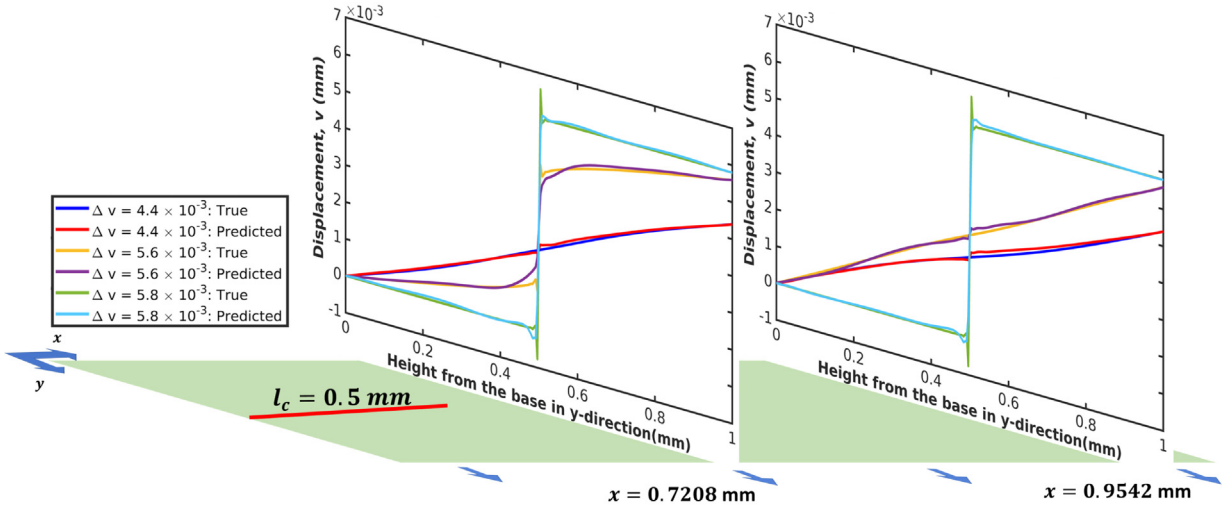


(c) $\Delta v_6 = 5.6 \times 10^{-3}$ mm.

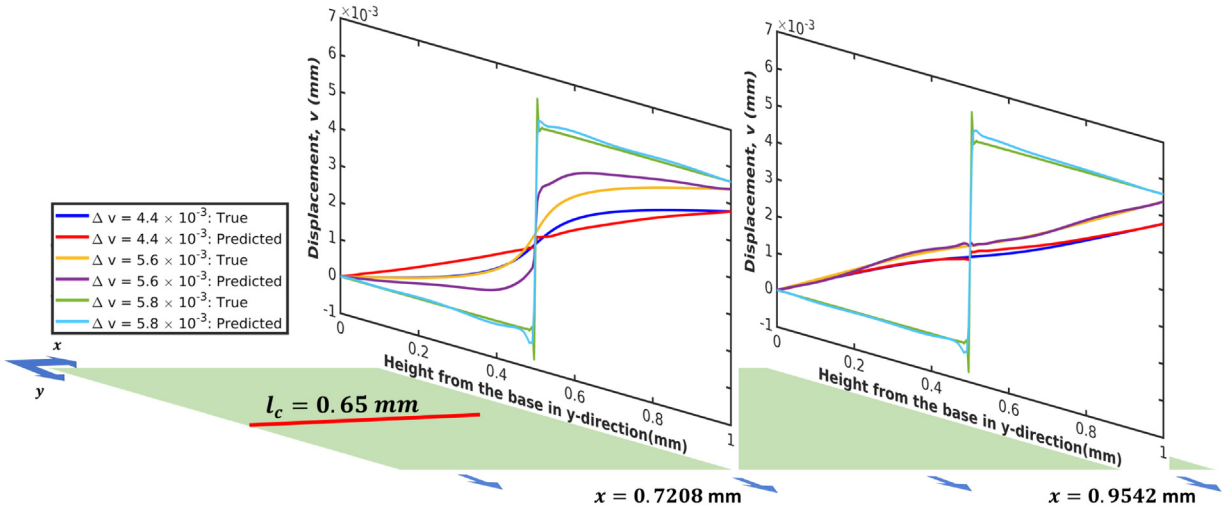


(d) $\Delta v_7 = 5.8 \times 10^{-3}$ mm.

Fig. 3. Tensile failure: The V-DeepONet is trained with 6 values of $l_c \in [0.3, 0.55]$ for 7 displacement steps. The plots for prediction fields with $l_c = 0.5$ mm at certain displacement steps are presented. For each plot, the top row presents the predicted phase field and the displacement along y-axis, respectively. The middle row is depicting the ground truth obtained using IGA simulations, while the last row shows the error between the predicted value and the ground truth.



(a) $l_c = 0.5$ mm.



(b) $l_c = 0.65$ mm

Fig. 4. Tensile failure: The V-DeepONet is trained with 6 values of $l_c \in [0.3, 0.55]$ for 7 displacement steps. The line plots depict the displacement component in y-direction (denoted as $v(x, y)$) at $x_1 = 0.7208$ mm and $x_2 = 0.9542$ mm. Plots in (a) consider $l_c = 0.5$ mm and (b) $l_c = 0.65$ mm (out-of distribution). In each of the plots, $v(x_i, y)$ is plotted as a function of y, for $i = 1, 2$. Different colors of lines represent results from different displacement increment loading.

where $\hat{\mathcal{G}}^u_{\boldsymbol{\theta}}$ and $\hat{\mathcal{G}}^v_{\boldsymbol{\theta}}$ are obtained from the DeepONet. The trainable parameters of the V-DeepONet are obtained by minimizing the hybrid loss function in Eq. (18). The trained surrogate model is used to predict the final crack path and solutions of the elastic field for $l_c = 0.375$ mm and $l_c = 0.685$ mm. The predicted plots are shown in Fig. 7. The training trajectory of the V-DeepONet using 11 training samples is shown in Fig. 14 in Appendix C. In the plot, the training loss depicts the hybrid loss given by Eq. (18). The conventional DeepONet is trained with the same 11 samples, keeping the network architecture of the branch net and the trunk net exactly the same. A prediction error of 26.2% is reported when trained with 11 training samples. To improve the accuracy, the training samples are increased to 22, but the model is unable to capture the mode-II failure. Lastly, the number of training samples is increased to 43 and a relative mean error of 3.12% is reported for ϕ. Fig. 8 presents the plots of the predicted

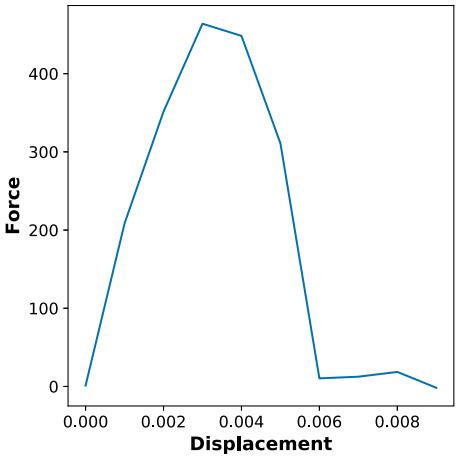


Fig. 5. Tensile failure: The force–displacement curve obtained using the V-DeepONet for $l_c = 0.5$ mm.

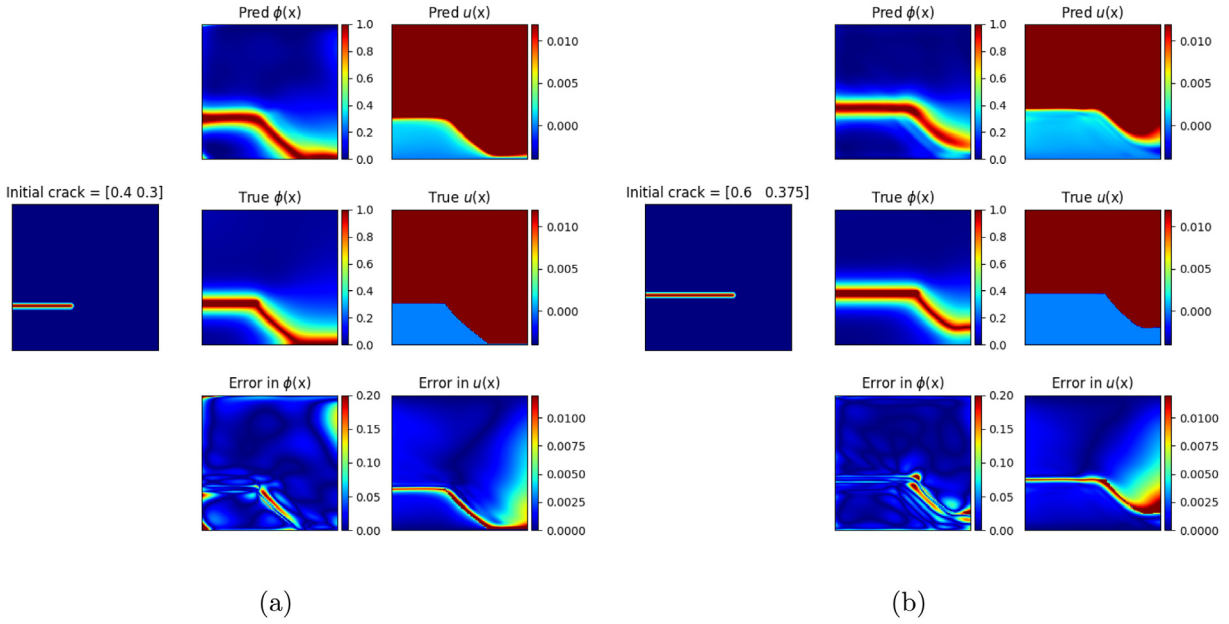


Fig. 6. Shear failure: The V-DeepONet is trained with 85 samples with crack tips located throughout the domain to predict the final damage path. The plots for two test cases are shown where the crack tips are located at: (a) $(0.4, 0.3)$, (b) $(0.6, 0.375)$. For each plot, the initial configuration is shown on the left. The top row presents the predicted phase field, displacement along x-axis, and displacement along y-axis, respectively. The middle row is depicting the ground truth obtained using the simulations in IGA, while the last row shows the error.

solutions for $l_c = 0.475$ mm and $l_c = 0.585$ mm when the conventional DeepONet is trained with 43 samples. The predicted results for the data driven DeepONet depict that it is unable to capture the crack diffusion phenomenon and also it cannot generalize complex fracture phenomenon with limited data-sets.

In the next experiment, we change the location of the crack vertically and at the same time we change the length of the horizontal crack. The aim of this experiment is to find the final damage path for any given location of the crack in the domain and any crack length and also to test the capability of the model to make an out-of-distribution prediction. To start with, $n = 20$ initial configurations have been considered to train the V-DeepONet, with crack lengths varying between $[0.3, 0.65]$, while the height of the crack is between $[0.2, 0.35]$. The trained model is used to predict the crack path for an initial crack tip located beyond the training data range, at $(0.425, 0.4)$. The predicted ϕ has an error 26.44%. To improve the generalization of the V-DeepONet, we add 20 more samples to the training

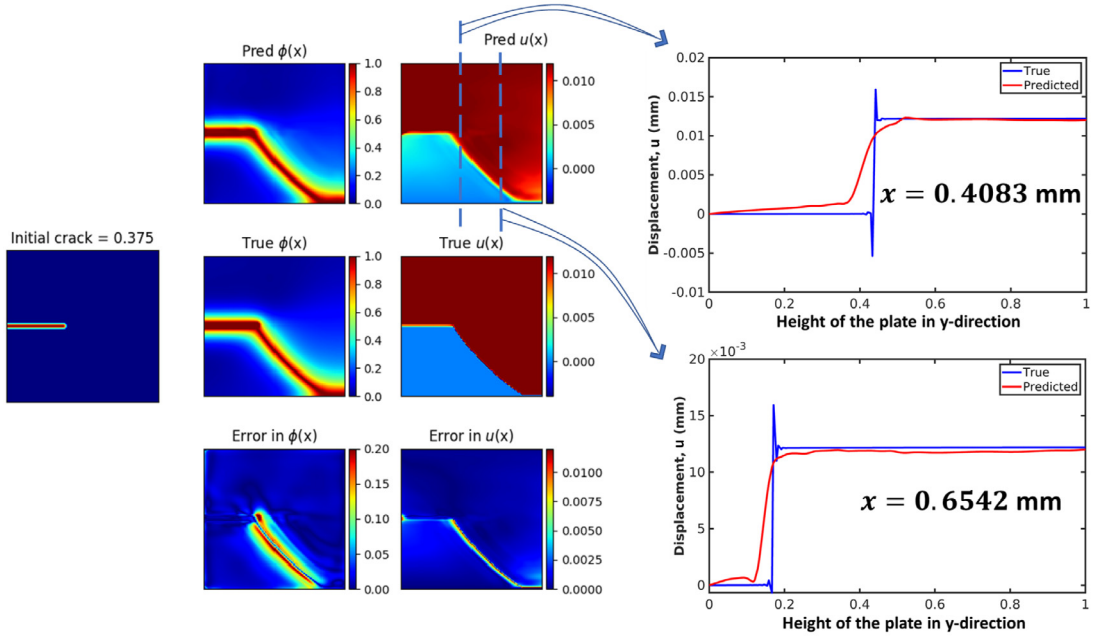
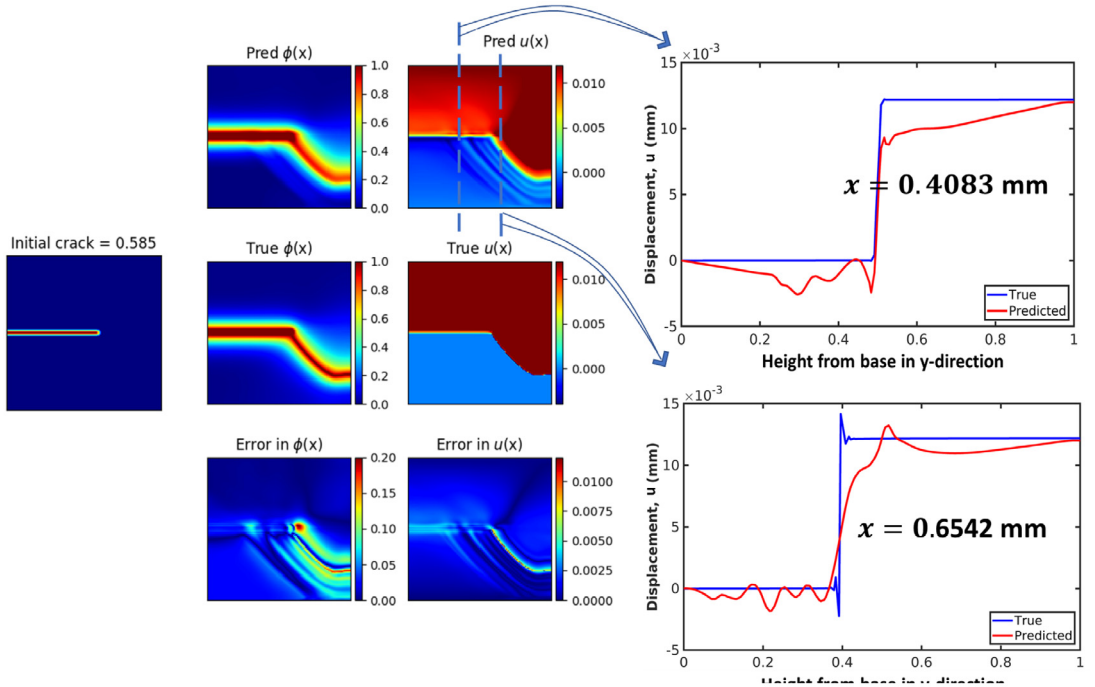
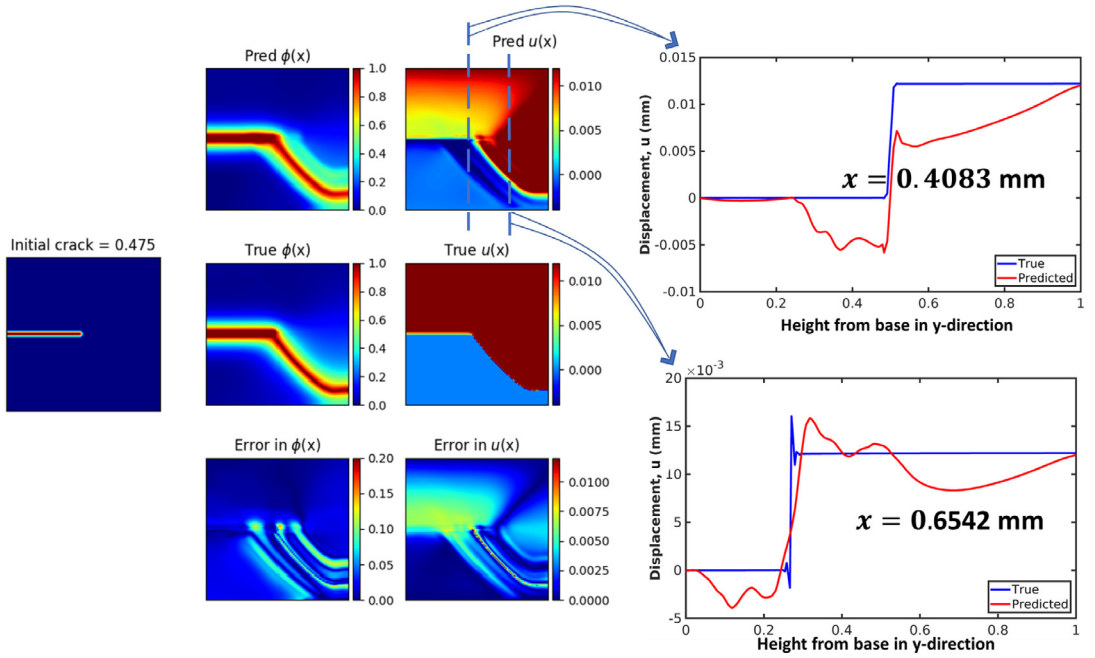


Fig. 7. Shear failure: The V-DeepONet is trained with 11 crack lengths to predict the final damage path for any crack length, when the height of the crack is fixed at the center of the left edge. The plots are for (a) $l_c = 0.375$ mm and (b) $l_c = 0.685$ mm, where $\Delta u = 0.220$ mm. For each plot, the predicted displacement in x-direction is plotted for two locations along the x-axis and is compared with ground truth to show the accuracy of the prediction.



(a)



(b)

Fig. 8. Shear failure: the data-driven DeepONet (original) is trained with 43 crack lengths to predict the final damage path for any crack length, when the height of the crack is fixed at the center of the left edge. The plots are for (a) $l_c = 0.475$ mm and (b) $l_c = 0.585$ mm, where $\Delta u = 0.220$ mm. For each plot, the predicted displacement in x-direction is plotted for two locations along the x-axis and is compared with ground truth to show the accuracy of the prediction.

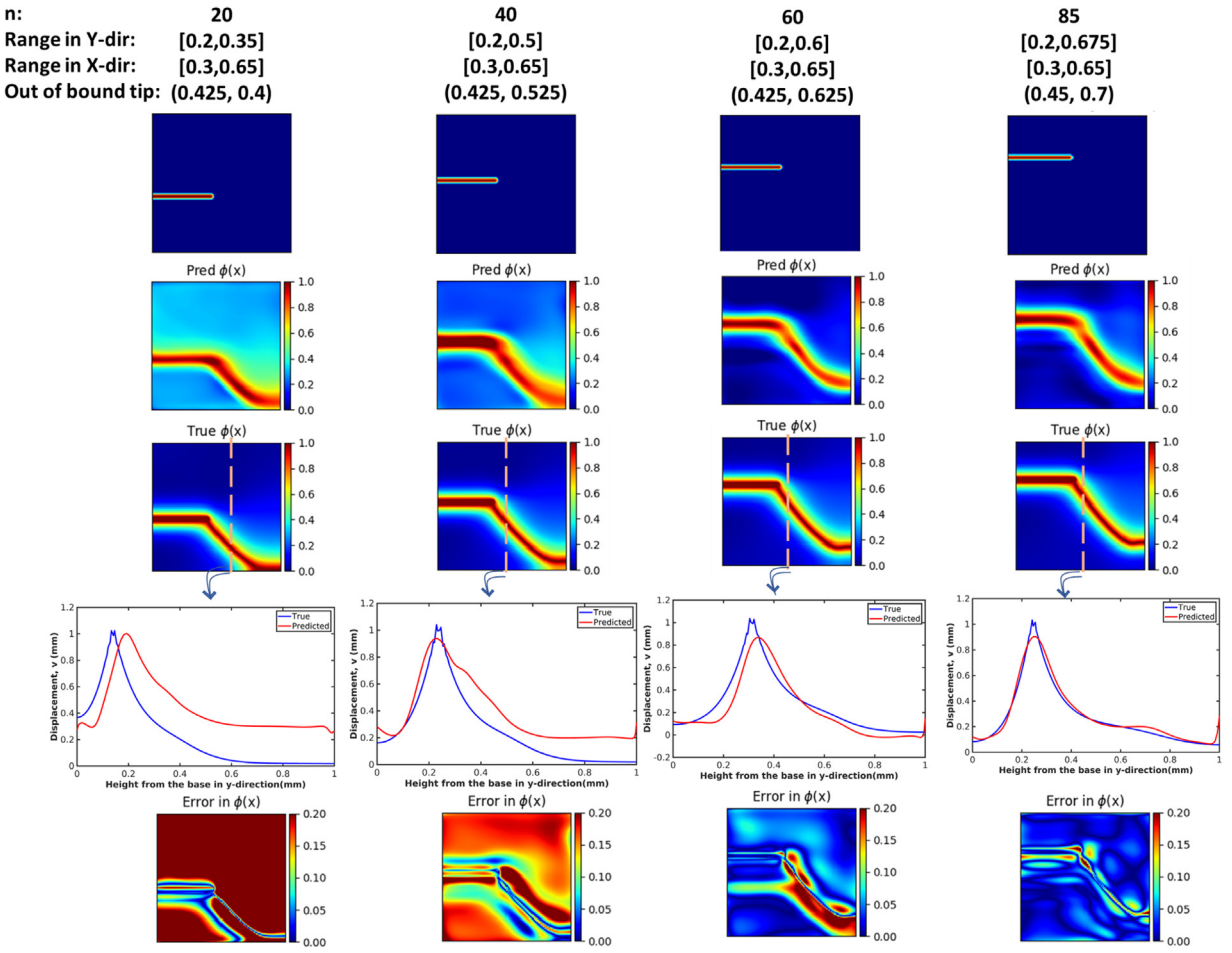


Fig. 9. Shear failure: The V-DeepONet is trained with $n = 20$, 40, 60, 85 samples (from left to right). For each n, a corresponding out-of distribution prediction is made. It is observed that the V-DeepONet can generalize the solution with just 85 training samples. A cross-section taken at $x = 0.6048$ is taken to show the predicted ϕ against the ground truth. The relative mean error is reported as 26.44%, 13.49%, 7.49% and 3.17% (from left to right).

set, hence $n = 40$. The length of crack varies over the same range as the previous case, however the height of crack varies between $[0.2, 0.5]$. The trained V-DeepONet is used to predict the final crack path for a crack tip located at $(0.425, 0.525)$, which is again beyond the range of the training samples. The predicted ϕ is reported to have 13.49% error. To the 40 training samples, another 20 training samples are added, making $n = 60$. In this case the height of the crack varies in the range $[0.2, 0.6]$. The crack path is predicted for a crack tip located at $(0.425, 0.625)$, which is again beyond the training range. The prediction accuracy of the model is improved, and an error of 7.49% is reported in this case. In the last case, 25 training samples are added, hence $n = 85$. The crack height varies between $[0.2, 0.675]$ and the crack length varies between $[0.3, 0.65]$ in the 85 training samples. The V-DeepONet based surrogate model is used to predict the crack path for a crack tipped at $(0.4, 0.7)$, an out-of-bound sample. The predicted ϕ has an error of 3.17%. The prediction plots of ϕ against the ground truth at a cross-section located at $x = 0.6048$ mm and also error plot over the whole domain is presented in Fig. 9. In this experiment, $m = 1547$ sensors are considered, which is pictorially shown in Fig. 10(a). To construct the hybrid loss function, $p = 6024$ points are sampled in the domain. The solution is approximated with a V-DeepONet, where the branch and trunk networks are two separate 4-layer fully-connected neural networks with $[100, 100, 50, 50]$ neurons, respectively. For plots presenting the prediction of the elastic field solutions, readers may refer to Fig. 15 in Appendix C. The out-of-distribution prediction accuracy increases with the number of samples since an over-parametrized neural

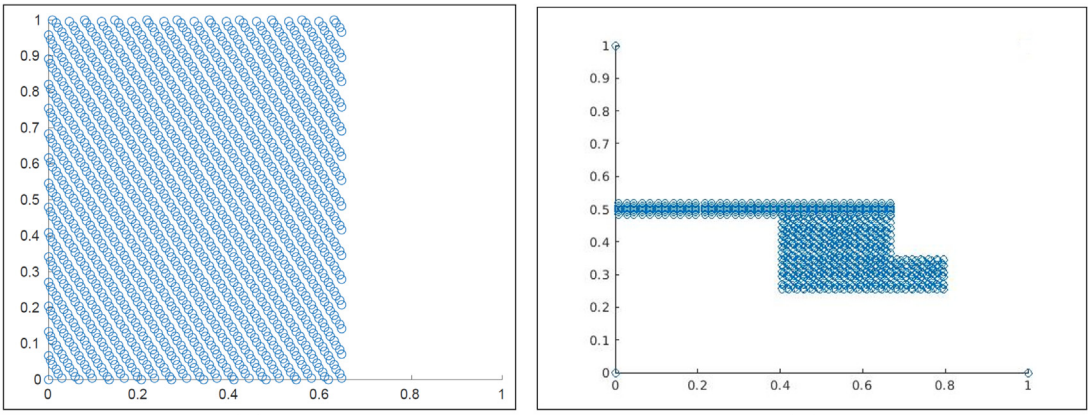


Fig. 10. Shear failure: (a) The $m = 1547$ sensor points are shown using blue circles. The sensors are chosen to represent the input functions discretely, so that network approximations can be applied. In this example, the training samples are chosen such that the initial crack can be placed anywhere between $[0, 1]$ in the y-axis. However, the initial crack lengths are restricted between $[0.2, 0.65]$. So, the sensor locations are between $[0, 0.65]$ along the horizontal axis and between $[0, 1]$ along the vertical axis. (b) The $m = 854$ sensor points chosen to represent the input functions discretely as the crack grows.

network can generalize better. The accuracy of the model in making an out-of-bound prediction with just 85 training samples, is attributed to the hybrid loss function which integrates physics with the data loss. In Fig. 6, we present the predicted final crack path and the solutions for the elastic field for two crack tips located at $(0.4, 0.3)$ and $(0.6, 0.375)$, considering $\Delta u = 0.0120$ mm, when the V-DeepONet model is trained with 85 samples.

In the last experiment we train the V-DeepONet to obtain the crack location at various displacement steps for different initial conditions. For this experiment, we consider a $s = 10$ initial conditions and $r = 3$ displacement steps, $\Delta u = \{1.04, 1.08, 1.14\} \times 10^{-2}$ mm, where we have fixed the height of the crack at the center of the left edge and varied the initial crack length in a range such that $l_c \in [0.4, 0.55]$. The selection of appropriate $m = 824$ sensors points is essential in this experiment. In Fig. 10(b), we show the location of sensor points chosen for this experiment. It is essential that the sensor points are well placed to accurately capture the strain energy for all the displacement steps for all the initial conditions, and must be distinguishable from one another. We compute the initial strain energy, $\boldsymbol{\mathcal{H}}_0^{(i)}$, $i \in \{1, \ldots, r\}$ using Eq. (7) at m sensor points. From the high-fidelity data simulated using IGA, we obtain the tensile strain-energy, $\boldsymbol{\mathcal{H}}_j^{(i)}$ for the applied displacements Δu_j, where $j \in \{1, \ldots, s\}$. The input data for the branch net, $\boldsymbol{\mathcal{H}}(\boldsymbol{y})$ is prepared in the same way as discussed in Eq. (22). In this example, we aim to learn the solution operators, $\mathcal{G}_{\boldsymbol{\theta}}^u, \mathcal{G}_{\boldsymbol{\theta}}^v, \mathcal{G}_{\boldsymbol{\theta}}^\phi$ from $\boldsymbol{\mathcal{H}}(\boldsymbol{y})$ to approximate the solution of $u(\boldsymbol{y})$, $v(\boldsymbol{y})$, and $\phi(\boldsymbol{y})$. To that end, we represent the operators by a V-DeepONet, where both the branch net and the trunk net are 4-layer fully-connected neural networks with 100 neurons per hidden layer. For each input function in the branch net, $p = 6024$ points are sampled in the domain. The ground truth is obtained from the high fidelity solver. Once the solution is evaluated at the sampled points, the V-DeepONet outputs for the elastic field are modified to exactly satisfy the Dirichlet boundary conditions using Eq. (27). The hybrid loss function is then constructed as the sum of the data loss and the total energy. The trainable parameter of the V-DeepONet is then computed by minimizing the hybrid loss using the Adam optimizer [53].

Now, we use the trained model to predict the solutions of phase field and the elastic field at all the displacement steps for $l_c = 0.435$ mm. For predicting the crack position at $\Delta u = 0.0104$ mm, the input to the branch net is the strain energy analytically obtained using Eq. (7) at 854 sensor locations padded with 854 zeros, indicating that there is no history behind. The solution for the field variables are obtained and then strain energy is computed at $\Delta u = 0.0104$ mm. To predict the solution of the elastic field at $\Delta u = 0.0106$ mm (different from the trained displacements), the input to the branch net is the current strain energy and the initial strain energy. The surrogate model predicts the displacements, u, v and the phase field, ϕ corresponding to all the displacement steps sequentially. The predicted plots for $l_c = 0.435$ mm are shown in Fig. 11. The computational time (CPU) for running each high-fidelity simulation is 345 min, while for training the surrogate model (GPU) using the basic version of DeepONet

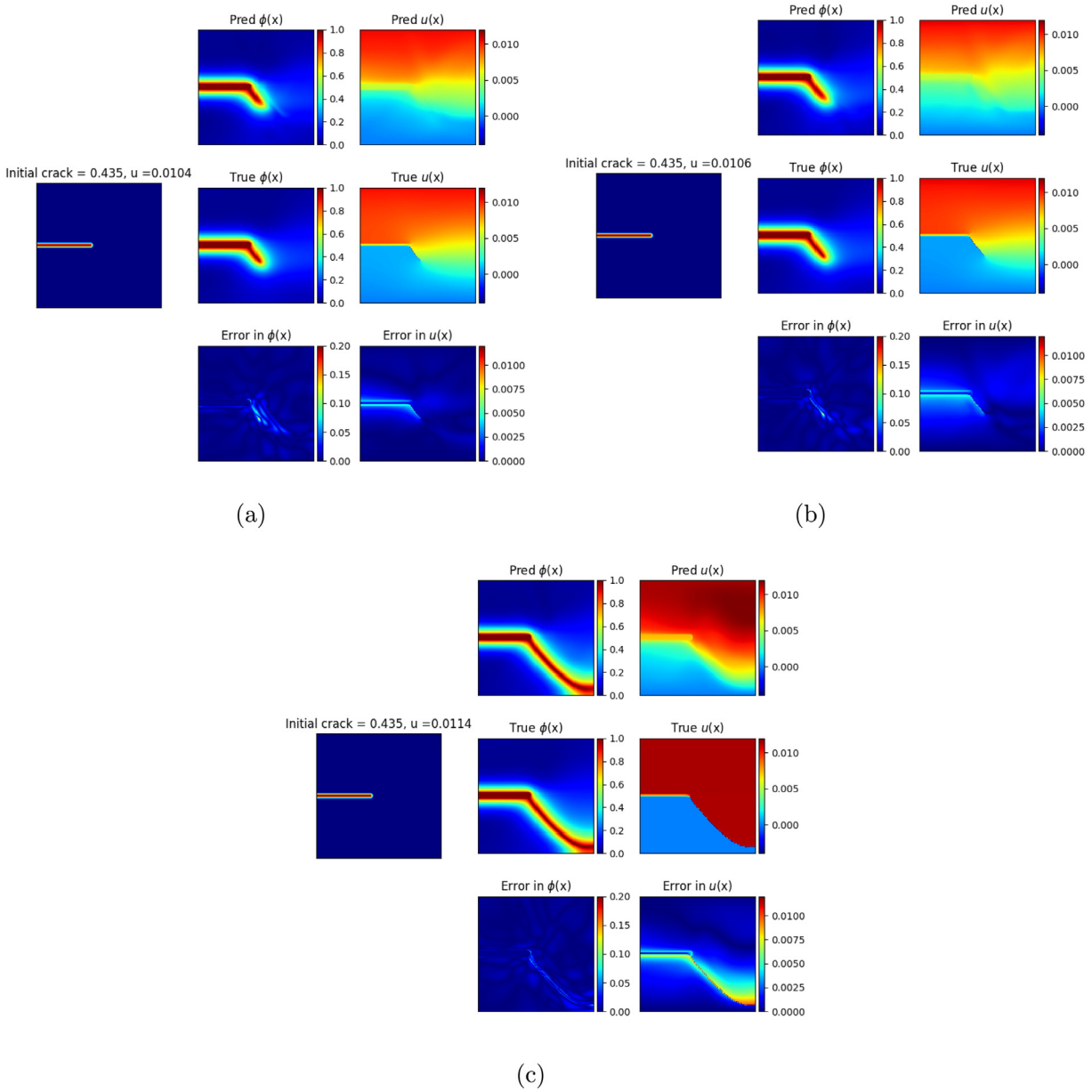


Fig. 11. Shear failure: V-DeepONet is trained with 10 initial crack configurations for 3 displacement steps. The plots for a testing sample with initial crack length $= 0.435$ mm at 3 displacement steps are shown. (a) $\Delta u = 0.0104$ mm, (b) $\Delta u = 0.0106$ mm, (c) $\Delta u = 0.0114$ mm. For each plot, the initial configuration is shown on the left with the applied displacement at that step. The top row presents the predicted phase field and the displacement along x-axis. The middle row is depicting the ground truth obtained using the simulations in IGA, while the last row shows the error.

requires 220 min. A faster DeepONet implementation proposed in [57] could further improve the computational efficiency of the Variational energy based DeepONet.

The examples shown in this work are for proof of concept. This method can be extended to problems with complex geometries like a plate with holes [20]. The main advantage of V-DeepONet for resolving complex crack patterns lies in the fact that it is governed by the laws of physics but it is also data driven to take into account any available experimental observations. Considering material randomness [58,59] as a part of failure analysis is essential. Extending the V-DeepONet architecture for modeling failure in heterogeneous material systems would be considered as part of our future research.

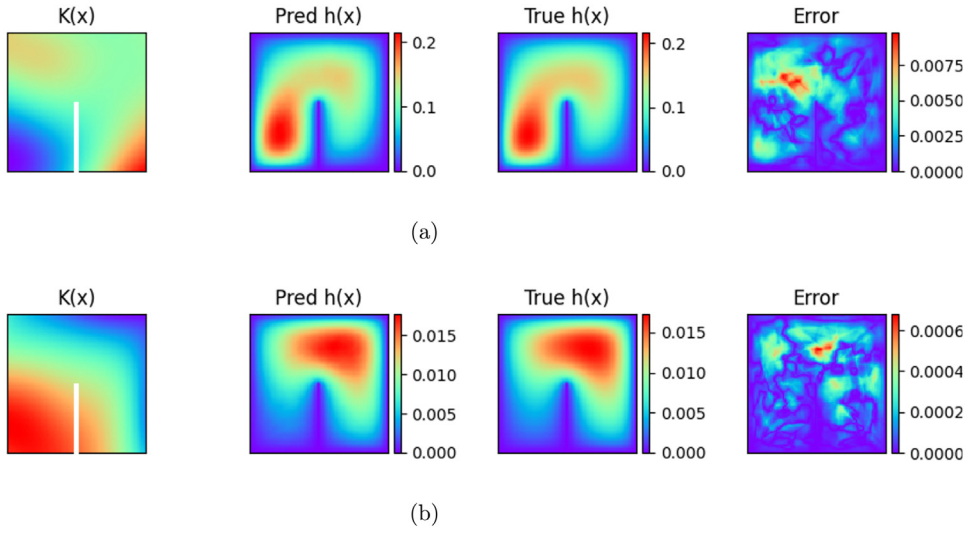


Fig. 12. Flow in heterogeneous porous media: The predicted $h(\boldsymbol{x})$ for a given permeability, $K(\boldsymbol{x})$ (plotted on log scale) are shown for two samples. True $h(\boldsymbol{x})$ represents the ground truth and is the simulated solution using the PDE toolbox. The difference between the predicted $h(\boldsymbol{x})$ and the ground truth is shown in the error plot.

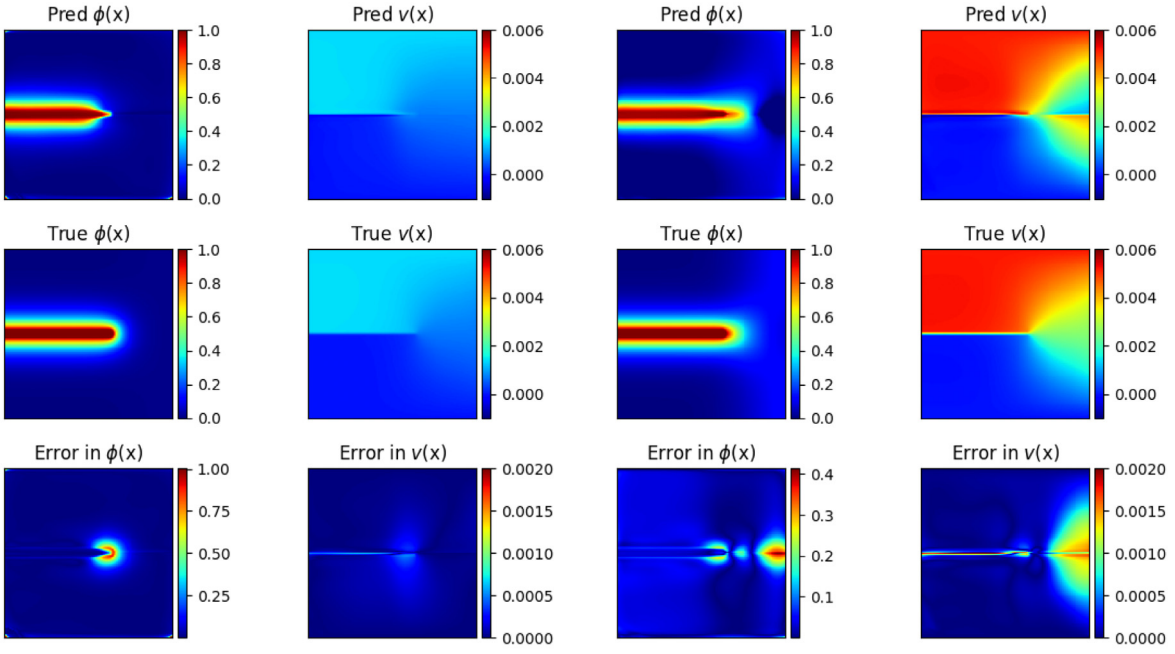
8. Summary and discussion

We have proposed neural networks to address the limitations of numerical methods that can simulate a single set of I/BCs and domain geometry at a time. The model developed in the framework of V-DeepONet provides an efficient surrogate of high-fidelity simulations to estimate key quantities of interest for brittle fracture such as failure paths, failure zones, and damage along failure. Once the V-DeepONet is trained, the model can predict failure paths and the corresponding displacements for any location of the crack tip and at any applied displacement at a fraction of a second. The salient features of the proposed model are:

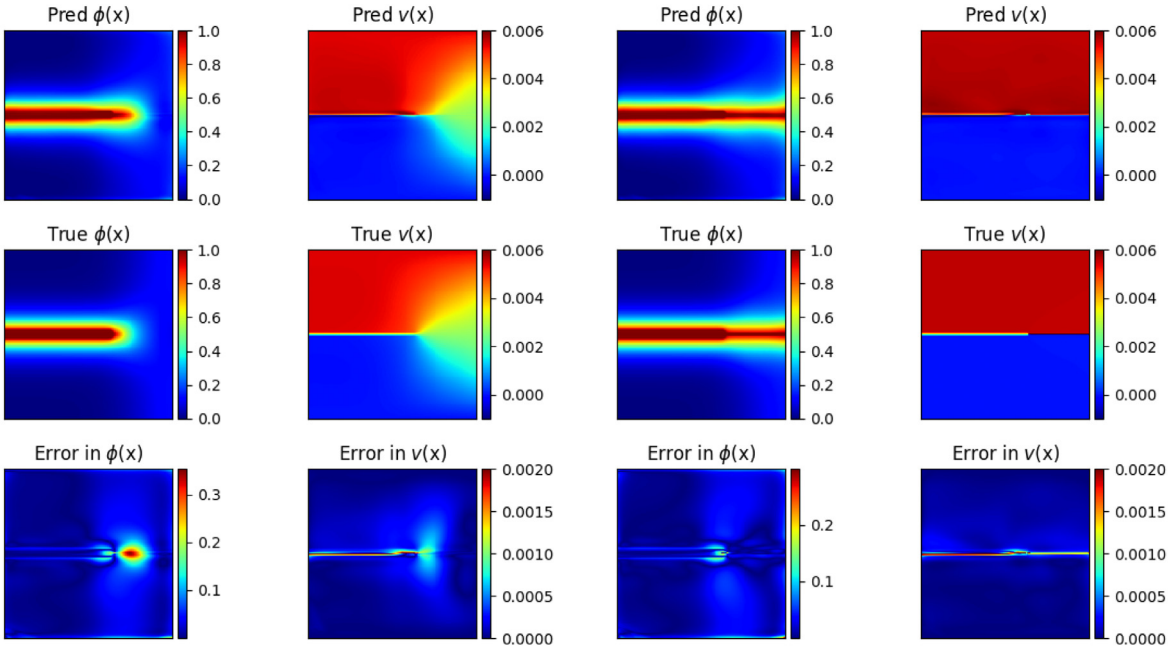
1. The model considers a hybrid loss function for training, which is the weighted sum of physics driven loss and data loss. This makes it ideal for extrapolation tasks.
2. The physics loss is derived from the variational form of the governing PDE, which makes it a very good choice for problems with discontinuous solutions.
3. Scaling to larger and more complex problems is often computationally prohibitive with computational tools such as molecular dynamics. This approach is independent of the dimensionality.
4. An extrapolation accuracy of 3.17% is reported for a Mode-II failure, which is often a challenging problem to solve even for classical numerical methods.

The proposed methodology is ideal for usage in domains such as reliability analysis, uncertainty quantification and design optimization.

Despite the excellent performance of the proposed surrogate model, it is important to note that this approach is handling fracture problems, which is very sensitive to minor fluctuations and hence, has certain limitations. First and foremost is the data-scaling issue. The input to the feed-forward network is energy, which is in the range $[0, 10^4]$ for a single step. Scaling the input tensile energy is essential but it has to be scaled judiciously since the input functions at the sensor locators have to be unique and distinguishable. Secondly, the weights of the loss-terms λ_1 and λ_2 have to be modulated manually to strike a balance between the data-driven loss and the energy loss. Finally, the outputs of the V-DeepONet are the elastic field, which is of the order $O(10^{-3})$, and the phase field, which is of the order $O(1)$. Hence, a scaling of the individual outputs of the V-DeepONet is important before computing the energy and the data-driven loss. In future works, we will address some of these issues and adopt an adaptive approach to choose the weights.



(a) $\Delta v_1 = 1.4 \times 10^{-3}$ mm. (b) $\Delta v_4 = 5.0 \times 10^{-3}$ mm.



(c) $\Delta v_5 = 5.6 \times 10^{-3}$ mm. (d) $\Delta v_6 = 5.8 \times 10^{-3}$ mm.

Fig. 13. Tensile failure: V-DeepONet is trained with 6 $l_c \in [0.3, 0.55]$ for 7 displacement steps. In these plots, we present the predicted solution $l_c = 0.65$ mm (out-of-distribution) at four displacement steps, Δv, which are predicted sequentially. For each plot, the top row presents the predicted phase field and the displacement along y-axis, respectively. The middle row shows the ground truth obtained using the IGA simulations, while the last row shows the error between the predicted value and the ground truth.

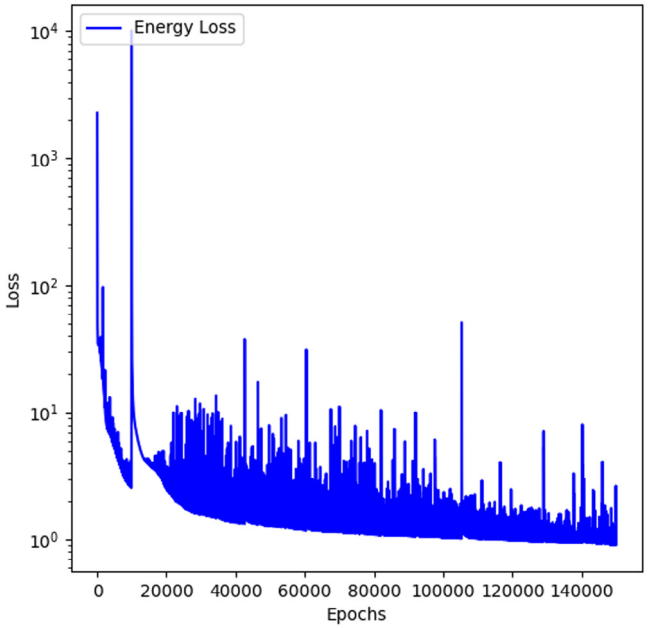


Fig. 14. Shear failure: The training trajectory of V-DeepONet using 11 training samples. The hybrid loss function in Eq. (18) is minimized to obtain the optimized $\boldsymbol{\theta}^*$.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was funded by the DOE PhILMs project (no. DE-SC0019453), OSD/AFOSR MURI grant FA9550-20-1-0358, and National Institutes of Health, USA grant (U01 HL142518). Y. Yu was supported by the National Science Foundation, USA under award DMS 1753031. The authors thank Dr. Xuhui Meng for helpful discussion and his Darcy's problem simulation code. The authors would also like to thank Prof. Blaise Bourdin, McMaster University, Canada, for his valuable comments which helped to improve the manuscript. S. Goswami thanks Dr. Khemraj Shukla for his support on setting on high-performance computing environment.

Appendix A. Algorithm for constructing the proposed unified surrogate model.

Pre-requirement of data from the high-fidelity solver:

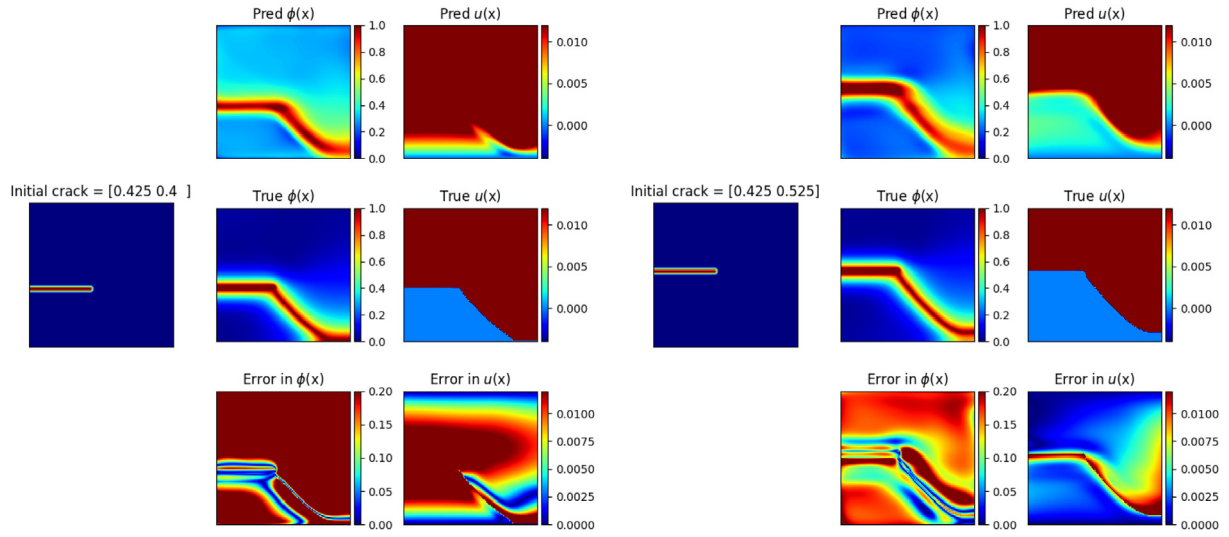
1. Sample the domain for p points, $\{\boldsymbol{y}_i\}_{i=1}^{p}$, where the V-DeepONet will be evaluated.
2. For each of the n initial defect locations, obtain the responses, $u(\boldsymbol{y})$, $v(\boldsymbol{y})$, $\phi(\boldsymbol{y})$ for selected r displacement steps, $\{\Delta \boldsymbol{w}_i\}_{i=1}^{r}$.
3. Compute the tensile strain energy, $\mathcal{H}_i$ for $\{\Delta \boldsymbol{w}_i\}_{i=1}^{(r-1)}$, corresponding to n cases.

We put forth the algorithm for constructing the surrogate model to predict the crack path for brittle fracture in Algorithm 1.

Appendix B. Flow in heterogeneous porous media

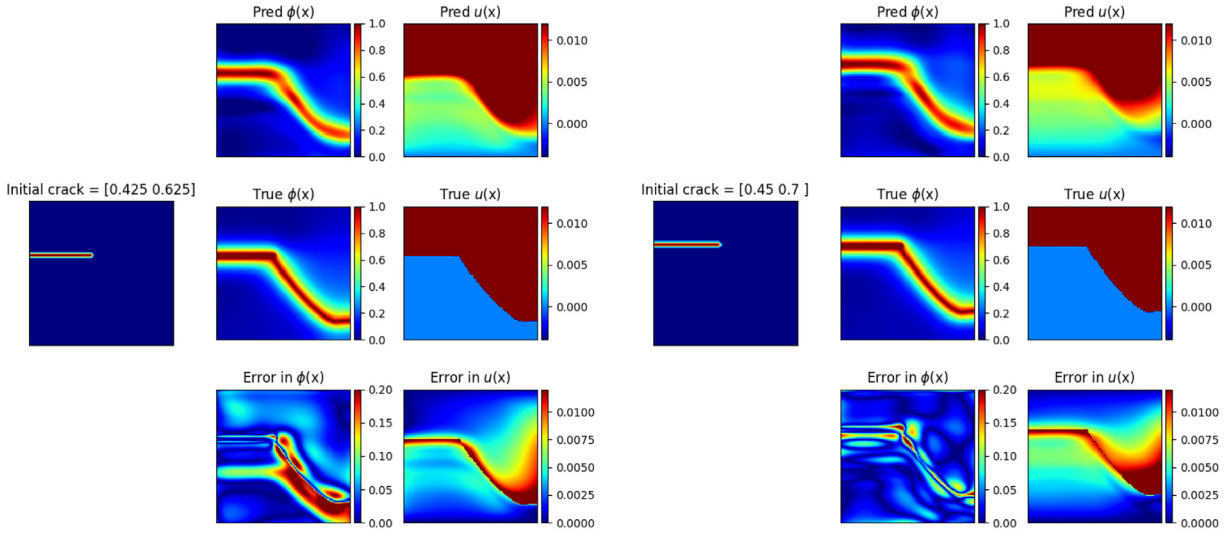
We consider a two-dimensional flow through heterogeneous porous media, which is governed by the following equation:

$$\begin{aligned} &-\nabla \cdot (K(\boldsymbol{x})\nabla h(\boldsymbol{x})) = 1, \quad \boldsymbol{x} = (x, y), \\ &\text{subjected to} \quad h(\boldsymbol{x}) = 0, \quad \forall\ \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{28}$$



(a) $n = 20$

(b) $n = 40$



(c) $n = 60$

(d) $n = 85$

Fig. 15. Shear failure: V-DeepONet is trained with 20,40, 60 and 85 samples with crack tips located throughout the domain. For each of the sample sizes, different crack tips beyond the training range have been considered for prediction. Using the trained model, we predict the final damage path and the displacement in x-axis for an out-of-distribution model for each of the cases.

where $K(\boldsymbol{x})$ is spatially varying hydraulic conductivity, and $h(\boldsymbol{x})$ is the hydraulic head. In this example, we aim to learn the operator such that:

$$\mathcal{G}_{\boldsymbol{\theta}} : K(\boldsymbol{x}) \rightarrow h(\boldsymbol{x}). \tag{29}$$

The setup is of a unit square plate with a discontinuity of 5×10^{-3} mm. For generating multiple permeability field for training the V-DeepONet, we describe the conductivity field, $K(\boldsymbol{x})$, as a stochastic process. In particular, we take $K(\boldsymbol{x}) = \exp(F(\boldsymbol{x}))$, with $F(\boldsymbol{x})$ denoting a truncated Karhunen–Loève (KL) expansion for a certain Gaussian

Algorithm 1: V-DeepONet-based surrogate model for predicting brittle fracture.

Inputs: The location of m sensors, $\mathcal{X}_s = \{\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots, \boldsymbol{x}_m\}$.
Compute the initial history function, $\{H_0^i\}_{i=1}^n$ at $\mathcal{X}_s$ using Eq. (7).
Obtain tensile strain energy, $\{\mathcal{H}_i^j\}_{i,j=1}^{n,r-1}$ at $\mathcal{X}_s$ from the high-fidelity data.
Compute the initial history function, $\{\mathbf{H}_0^j\}_{j=1}^n$ at $\{\boldsymbol{y}_i\}_{i=1}^p$ points.
Prepare the data for trunk net: $\left[\{\boldsymbol{y}_i\}_{i=1}^p, \{\mathbf{H}_0^j\}_{j=1}^n, \{\Delta \boldsymbol{w}_k\}_{k=1}^r\right]$ as shown in Fig. 1(b).
Construct the input data for branch net as defined in Eq. (22). For each case, the tensor dimensions of the window is $(p, 2\ m)$.
Initialize the V-DeepONet and the weights of the network using Xavier initialization technique.
Obtain the solution operators, $\mathcal{G}_{\boldsymbol{\theta}}$ and evaluate it at $\{\boldsymbol{y}_i\}_{i=1}^p$ using Eq. (10).
Construct the hybrid loss-function using Eq. (18).
Minimize the hybrid loss and obtain the optimized parameters, $\boldsymbol{\theta}^*$.
For prediction, choose an initial configuration and obtain $\boldsymbol{\mathcal{H}}_0$ at $\mathcal{X}_s$ and $\mathbf{H}_0$ at q random points sampled in the domain, and t displacement steps.
for $i = 1, \ldots, t$ **do**
    if $i == 1$ **then**
        Input to branch net = $[\boldsymbol{\mathcal{H}}_0(\mathcal{X}_s), \mathbf{0}]$.
        Repeat this vector q times.
    else
        Input to branch net = $[\boldsymbol{\mathcal{H}}_i, \boldsymbol{\mathcal{H}}_{i-1}]$.
        Repeat this vector q times.
    end
    Input to the trunk net: $\left[\{\boldsymbol{y}_j\}_{j=1}^q, \mathbf{H}_0, \Delta \boldsymbol{w}_i\right]$.
    Predict the solutions, $u^*(\boldsymbol{y}_j)$, $v^*(\boldsymbol{y}_j)$ and $\phi^*(\boldsymbol{y}_j)$.
    Using the solutions compute the energy, $\boldsymbol{\mathcal{H}}_i$ required as input to the next step.
end

process, which is a finite-dimensional random variable. In our samples, the leading 100 terms in the KL expansion were kept for the Gaussian process with zero mean and the following kernel [60]:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}((x, y), (x', y')) = \exp\left[\frac{-(x - x')}{2l_1^2} + \frac{-(y - y')^2}{2l_2^2}\right], \tag{30}$$
$$\boldsymbol{x}, \boldsymbol{x}' \in [0, 1]^2, l_1 = l_2 = 0.25.$$

In this example, the V-DeepONet is trained using the variational formulation, without any labeled input–output datasets. The optimization problem can be defined as:

$$\begin{aligned} &\text{Minimize:} \quad \mathcal{E} = \Psi_h, \\ &\text{subject to:} \quad h(\boldsymbol{x}) = 0 \text{ on } \partial\Omega_D, \end{aligned} \tag{31}$$

where

$$\Psi_h = \frac{1}{2}\int_\Omega K(\boldsymbol{x})|\nabla h(\boldsymbol{x})|^2 \ d\Omega - \int_\Omega h(\boldsymbol{x}) \ d\Omega. \tag{32}$$

We approximate the operator by a V-DeepONet architecture, where the branch and trunk networks are two separate 6-layer fully-connected neural networks with 32 neurons per hidden layer. In this example, $n = 200$ samples of permeability matrix are used to train. The solution operator is evaluated at $p = 10000$ randomly sampled points.

The prediction of $h(\boldsymbol{x})$ for two samples of $K(\boldsymbol{x})$, using V-DeepONet is shown in Fig. 12. The accuracy of V-DeepONet is verified by comparing the prediction of the network against the solution of Eq. (28) obtained using the

finite-element-based Partial Differential Equation Toolbox in Matlab using the same $K(\boldsymbol{x})$. It is interesting to note that we have tried to solve the problem by minimizing the residual [61]. However, the residual based DeepONet is not able to approximate the solution of $h(\boldsymbol{x})$ for a given $K(\boldsymbol{x})$.

Appendix C. Additional results

See Figs. 13–15.

References

[1] C. Kuhn, R. Müller, A continuum phase field model for fracture, Eng. Fract. Mech. 77 (18) (2010) 3625–3634.
[2] M.J. Borden, C.V. Verhoosel, M.A. Scott, T.J. Hughes, C.M. Landis, A phase-field description of dynamic brittle fracture, Comput. Methods Appl. Mech. Eng. 217 (2012) 77–95.
[3] T.T. Nguyen, J. Yvonnet, Q.-Z. Zhu, M. Bornert, C. Chateau, A phase field method to simulate crack nucleation and propagation in strongly heterogeneous materials from direct imaging of their microstructure, Eng. Fract. Mech. 139 (2015) 18–39.
[4] M.J. Borden, T.J. Hughes, C.M. Landis, C.V. Verhoosel, A higher-order phase-field model for brittle fracture: Formulation and analysis within the isogeometric analysis framework, Comput. Methods Appl. Mech. Eng. 273 (2014) 100–118.
[5] E. Emmrich, O. Weckner, On the well-posedness of the linear peridynamic model and its convergence towards the Navier equation of linear elasticity, Commun. Math. Sci. 5 (4) (2007) 851–864.
[6] S.A. Silling, Reformulation of elasticity theory for discontinuities and long-range forces, J. Mech. Phys. Solids 48 (1) (2000) 175–209.
[7] Y. Yu, F.F. Bargos, H. You, M.L. Parks, M.L. Bittencourt, G.E. Karniadakis, A partitioned coupling framework for peridynamics and classical theory: analysis and simulations, Comput. Methods Appl. Mech. Eng. 340 (2018) 905–931.
[8] E. Haghighat, A.C. Bekar, E. Madenci, R. Juanes, A nonlocal physics-informed deep learning framework using the peridynamic differential operator, Comput. Methods Appl. Mech. Eng. 385 (2021) 114012.
[9] N. Trask, H. You, Y. Yu, M.L. Parks, An asymptotically compatible meshfree quadrature rule for nonlocal problems with applications to peridynamics, Comput. Methods Appl. Mech. Eng. 343 (2019) 151–165.
[10] Y. Yu, H. You, N. Trask, An asymptotically compatible treatment of traction loading in linearly elastic peridynamic fracture, Comput. Methods Appl. Mech. Eng. 377 (2021) 113691.
[11] D.C. Psichogios, L.H. Ungar, A hybrid neural network-first principles approach to process modeling, AIChE J. 38 (10) (1992) 1499–1511.
[12] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Eng. 379 (2021) 113741.
[13] B. Peherstorfer, B. Kramer, K. Willcox, Combining multiple surrogate models to accelerate failure probability estimation with expensive high-fidelity models, J. Comput. Phys. 341 (2017) 61–75.
[14] Y. Hou, T. Sapanathan, A. Dumon, P. Culière, M. Rachik, A novel development of bi-level reduced surrogate model to predict ductile fracture behaviors, Eng. Fract. Mech. 188 (2018) 232–249.
[15] B.P. van de Weg, L. Greve, M. Andres, T. Eller, B. Rosic, Neural network-based surrogate model for a bifurcating structural fracture response, Eng. Fract. Mech. 241 (2021) 107424.
[16] E.R. Martínez, S. Chakraborty, S. Tesfamariam, Machine learning assisted stochastic-XFEM for stochastic crack propagation and reliability analysis, Theor. Appl. Fract. Mech. 112 (2021) 102882.
[17] H. You, Y. Yu, N. Trask, M. Gulian, M. D'Elia, Data-driven learning of nonlocal physics from high-fidelity synthetic data, Comput. Methods Appl. Mech. Eng. 374 (2021) 113553.
[18] H. You, Y. Yu, S. Silling, M. D'Elia, Data-driven learning of nonlocal models: from high-fidelity simulations to constitutive laws, AAAI Spring Symposium, MLPS, 2021.
[19] H. You, Y. Yu, S. Silling, M. D'Elia, A data-driven peridynamic continuum model for upscaling molecular dynamics, 2021, arXiv preprint arXiv:2108.04883.
[20] T. Bittencourt, P. Wawrzynek, A. Ingraffea, J. Sousa, Quasi-automatic simulation of crack propagation for 2D LEFM problems, Eng. Fract. Mech. 55 (2) (1996) 321–334.
[21] P. Beran, W. Silva, Reduced-order modeling-new approaches for computational physics, in: 39th Aerospace Sciences Meeting And Exhibit, 2001, p. 853.
[22] D. Amsallem, C. Farhat, Interpolation method for adapting reduced-order models and application to aeroelasticity, AIAA J. 46 (7) (2008) 1803–1813.
[23] D. Amsallem, J. Cortial, K. Carlberg, C. Farhat, A method for interpolating on manifolds structural dynamics reduced-order models, Int. J. Numer. Methods Eng. 80 (9) (2009) 1241–1258.
[24] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[25] E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications, Comput. Methods Appl. Mech. Eng. 362 (2020) 112790.
[26] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.

[27] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, 2021, arXiv preprint arXiv:2105.09506.
[28] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, Theor. Appl. Fract. Mech. 106 (2020) 102447.
[29] S. Goswami, C. Anitescu, T. Rabczuk, Adaptive fourth-order phase field analysis using deep energy minimization, Theor. Appl. Fract. Mech. 107 (2020) 102527.
[30] M. Yin, X. Zheng, J.D. Humphrey, G.E. Karniadakis, Non-invasive inference of thrombus material properties with physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 375 (2021) 113603.
[31] A.D. Jagtap, G.E. Karniadakis, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (5) (2020) 2002–2041.
[32] E. Zhang, M. Yin, G.E. Karniadakis, Physics-informed neural networks for nonhomogeneous material identification in elasticity imaging, 2020, arXiv preprint arXiv:2009.04525.
[33] W. E, B. Yu, The deep ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[34] L. Bar, N. Sochen, Unsupervised deep learning algorithm for PDE-based forward and inverse problems, 2019, arXiv preprint arXiv:1 904.05417.
[35] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 481–490.
[36] Y. Zhu, N. Zabaras, BayesIan deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447.
[37] J. Adler, O. Öktem, Solving ill-posed inverse problems using iterative deep neural networks, Inverse Problems 33 (12) (2017) 124007.
[38] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Comput. Mech. 64 (2) (2019) 525–545.
[39] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Multipole graph neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2006.09535.
[40] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[41] M. Yin, E. Ban, B.V. Rego, E. Zhang, C. Cavinato, J.D. Humphrey, G.E. Karniadakis, Simulating progressive intramural damage leading to aortic dissection using an operator-regression neural network, 2021, arXiv preprint arXiv:2108.11985.
[42] S. Goswami, C. Anitescu, T. Rabczuk, Adaptive fourth-order phase field analysis for brittle fracture, Comput. Methods Appl. Mech. Eng. 361 (2020) 112808.
[43] A. Griffith, The phenomena of rupture and flow in solids, Philos. Trans. R. Soc. Lond. 221 (Series A) (1921) 163–198.
[44] G. Francfort, J.-J. Marigo, Revisiting brittle fracture as an energy minimization problem, J. Mech. Phys. Solids 46 (8) (1998) 1319–1342.
[45] S. Goswami, C. Anitescu, T. Rabczuk, Adaptive phase field analysis with dual hierarchical meshes for brittle fracture, Eng. Fract. Mech. 218 (2019) 106608.
[46] J.-Y. Wu, V.P. Nguyen, C.T. Nguyen, D. Sutula, S. Bordas, S. Sinaie, Phase field modeling of fracture, Adv. Appl. Mech. Multi-Scale Theory Comput. 52 (2018).
[47] B. Bourdin, G. Francfort, J.-J. Marigo, Numerical experiments in revisited brittle fracture, J. Mech. Phys. Solids 48 (4) (2000) 797–826.
[48] C. Miehe, M. Hofacker, F. Welschinger, A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits, Comput. Methods Appl. Mech. Eng. 199 (45–48) (2010) 2765–2778.
[49] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw. 6 (4) (1995) 911–917.
[50] R. Rojas, Neural Network: A Systematic Introduction, Springer, 1996.
[51] S. Cai, Z. Wang, L. Lu, T.A. Zaki, G.E. Karniadakis, DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks, J. Comput. Phys. 436 (2021) 110296.
[52] C. Lin, Z. Li, L. Lu, S. Cai, M. Maxey, G.E. Karniadakis, Operator learning for predicting multiscale bubble growth dynamics, J. Chem. Phys. 154 (10) (2021) 104118.
[53] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[54] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019) 8026–8037.
[55] B. Bourdin, G.A. Francfort, J.-J. Marigo, Numerical experiments in revisited brittle fracture, J. Mech. Phys. Solids 48 (4) (2000) 797–826.
[56] T. Gerasimov, L. De Lorenzis, Second-order phase-field formulations for anisotropic brittle fracture, 2021, arXiv preprint arXiv:2107. 13280.
[57] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, G.E. Karniadakis, A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data, 2021.
[58] N.-H. Nguyen, V.P. Nguyen, J.-Y. Wu, T.-H.-H. Le, Y. Ding, Mesh-based and meshfree reduced order phase-field models for brittle fracture: One dimensional problems, Materials 12 (11) (2019).
[59] S. Torquato, H. Haslach Jr., Random heterogeneous materials: microstructure and macroscopic properties, Appl. Mech. Rev. 55 (4) (2002) B62–B63.
[60] X. Meng, L. Yang, Z. Mao, J.d.A. Ferrandis, G.E. Karniadakis, Learning functional priors and posteriors from data and physics, 2021, arXiv preprint arXiv:2106.05863.
[61] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets, 2021, arXiv preprint arXiv:2103.10974.