Crack Path Prediction with Operator Learning using Discrete Particle System data Generation

Elham Kiyani[1*], Venkatesh Ananchaperumal[2*], Ahmad Peyvan[1], Mahendaran Uchimali[3], Gang Li[2], and George Em Karniadakis[1]

[1]Division of Applied Mathematics, Brown University, USA
[2]Department of Mechanical Engineering, Clemson University, USA
[3]School of Mechanical Sciences, Indian Institute of Technology Bhubaneswar, India
[*]Authors contributed equally to this work.

Abstract

Accurately modeling crack propagation is critical for predicting failure in engineering materials and structures, where even small cracks can evolve rapidly and lead to catastrophic damage. The interaction of cracks with discontinuities in the surrounding domain, such as holes, plays a significant role in the deflection and arrest of crack paths. The recent developments in the discrete particle system with multibody interactions based on constitutive behavior demonstrate the ability to capture crack nucleation and evolution, without the limitations of continuum assumptions. Data on crack propagation from Constitutively informed Particle Dynamics (CPD) simulations are used to train operator learning models, particularly through Deep Operator Networks (DeepONets), offers a powerful data-driven alternative by learning mappings between function spaces rather than finite-dimensional inputs and outputs. In this work, we investigate two DeepONet variants—vanilla and Fusion DeepONet—for predicting the evolution of crack propagation over time in specimens with varying geometries. Notably, our study addresses not only geometric variability across samples but also the inherent time-dependent nature of fracture evolution. Three representative cases are examined: (i) varying pre-crack notch height without active fracture, and (ii) and (iii) combinations of varying notch heights and hole radii where fracture propagates dynamically on irregular discrete tessellation. The networks are trained on 32 to 45 samples, using geometric parameters as input to the branch network and spatial-temporal coordinates to the trunk network. Our results show that the Fusion DeepONet consistently outperforms the vanilla DeepONet across all scenarios. Predictions are more accurate in non-fracturing cases, while fracture-driven scenarios involving both displacement and crack evolution present greater challenges. These findings highlight the capability of Fusion DeepONet to generalize across complex, geometry-varying, and time-dependent crack propagation phenomena.

1 Introduction

Brittle fracture has long been a critical topic in the study of material failure, with significant implications in structural engineering, materials science, and geomechanics. It refers to a fracture process that occurs with minimal plastic deformation and is often accompanied by rapid crack propagation [1]. Due to its sudden and catastrophic nature, understanding the mechanisms of brittle fracture is essential for designing safer and more reliable materials and structures [2].

Over the decades, experimental investigations have played a pivotal role in characterising the fracture behaviour of brittle materials such as ceramics, glasses, rocks, and certain metals under specific conditions. A wide range of techniques has been employed to determine fracture toughness, analyse crack paths, and explore surface features and underlying microstructural failure

mechanisms. Methods such as Scanning Electron Microscopy (SEM), Digital Image Correlation (DIC), acoustic emission monitoring, and X-ray computed tomography have enabled researchers to probe the initiation and evolution of cracks at various length scales. These techniques provide insights into the influence of microstructure, flaws, and loading conditions on brittle crack propagation.

Parallel to experimental efforts, the computational modelling of brittle fracture has been extensively studied to predict crack initiation, growth, and interaction under various conditions. Several numerical techniques have been developed to describe crack paths based on different failure criteria and material models. Among them, the Extended Finite Element Method (XFEM) [3], Finite Fracture Mechanics (FFM) [4], and the Embedded Finite Element Method (EFEM) [5] have emerged as prominent approaches for simulating brittle crack propagation. XFEM, for instance, allows discontinuities to be represented independently of the mesh by enriching the displacement field with special functions, enabling the simulation of arbitrary crack growth without remeshing. FFM introduces an energy-based criterion for crack advance, taking into account both the energy release rate and fracture resistance. EFEM, on the other hand, modifies the finite element formulation to embed a crack within an element, providing an alternative to mesh refinement around the crack tip. In parallel, the Cohesive Zone Model (CZM) has been widely adopted to simulate fracture by incorporating a traction-separation law that governs damage evolution ahead of the crack tip [6].

Despite their advantages, these computational methods come with significant challenges and limitations. One major difficulty lies in the accurate tracking of the crack path evolution, especially in heterogeneous or complex materials. For methods like XFEM and EFEM, a splitting algorithm must be defined for elements that are either intersected by the crack or contain the crack tip. This becomes computationally demanding and can result in crack topology problems, particularly when quadratic or higher-order displacement interpolation schemes are used [7].

Regularised continuum models such as the phase field model are popularly used to describe the evolution of cracks [8–11]. These regularised models incorporate higher gradients of the field variables, resulting in diffuse interfaces. Notably, the implicit kinetic relations of phase field models do not account for the micro-scale effects. Most importantly, the interface thickness determines the length scale (in the range of nm) of the model, which provides additional limitations. Another continuum model adopted to describe the crack initiation and propagation is peridynamics, where the horizon radius plays a crucial role in the model result.

Meanwhile, mesoscale discrete models are shown to be capable of describing the micro-scale features and their influence on the macroscopic behaviour. The fundamental difference of discrete models from their continuum counterparts lies in treating the body as a discrete or continuum entity. The consequence of this assumption is discussed in many aspects; specifically, the dispersion relation for an elastic material is linear in continuum, whereas the corresponding one for discrete models is sinusoidal. Thus, another limiting feature of the continuum approach is the non-dispersive nature of the model. On the other hand, discrete models, even with the simplest particle interactions, are dispersive. Thus, in discrete models, the lattice-level dispersive effects arise naturally, and these models are naturally suitable for describing cracks or any microscale phenomenon [12, 13].

Recent advances in discrete particle dynamics have enabled the internal forces among particles to be derived directly from a material's continuum free energy using multi-body interactions [14–18]. This Constitutively informed Particle Dynamics (CPD) formulation overcomes the limitations of traditional pairwise lattice-based models by allowing the complete elastic behavior of a material to emerge from its constitutive matrix or strain energy function, rather than relying on predefined network architectures. The initial configuration of the particle system is established through Delaunay triangulation, which serves as a fixed reference lattice for computing internal force resistances. By capturing the interactions and displacements between particles, the CPD method facilitates accurate computation of stress, strain, and failure behavior in heterogeneous materials [14, 15].

While CPD provides a physically grounded framework for simulating complex deformation

and fracture processes, it can be computationally expensive when applied to large-scale systems or across varying geometries. In parallel, machine learning has recently emerged as a powerful suite of tools for modeling complex material behavior, including fracture [19–23]. These approaches have been successfully applied to phase-field fracture modeling [21,22], damage evolution in quasi-brittle materials [24], and fatigue crack growth prediction [25], offering data-driven surrogates that can significantly accelerate predictive simulations.

Machine learning models have increasingly been adopted to simulate particle motion in discrete systems, offering significant computational advantages over traditional Discrete Element Method (DEM) simulations. For example, a Convolutional Neural Network (CNN) has been employed to update particle positions using intermediate velocities and learned correction terms [26]. In powder mixing applications, a Recurrent Neural Network with Stochastic Randomness (RNNSR) captures both Lagrangian and Eulerian components of particle behavior to predict individual trajectories [27]. The NN4DEM framework introduces a physics-informed CNN by embedding contact force models directly into convolutional kernels, enabling structured-grid predictions that retain physical fidelity [28]. In contrast, Kazemi et al. [29] proposed an ML-DEM approach that couples short-run DEM data with a deep network employing continuous convolution operators to forecast particle displacements in rotary drums with high accuracy and reduced cost.

Beyond surrogate modeling, operator learning has emerged as a powerful paradigm in scientific machine learning, particularly for approximating solutions to partial differential equations (PDEs). Unlike conventional regression-based models, operator learning seeks to approximate mappings between infinite-dimensional function spaces, offering generalization across varying initial/boundary conditions and geometries [30–36]. A leading architecture in this domain is the Deep Operator Network (DeepONet) [30], which uses a branch-trunk structure to separately encode input functions and query locations. Physics-informed DeepONets extend this framework by incorporating residuals of governing equations into the loss function [22, 23, 37], and two-step training strategies have further improved accuracy and stability [38, 39].

Further enhancements in training efficiency and model generalization have been achieved through two-step training procedures [22, 38, 39], which decouple the optimization of the branch and trunk networks before recombining them. This strategy was recently applied in the context of brittle fracture modeling using phase-field formulations [22], where various trunk architectures—including standard feedforward networks, Kolmogorov–Arnold Networks (KANs) [40–42], and physics-informed designs—were evaluated for their ability to accurately capture crack nucleation, propagation, and branching behaviors. In parallel, Crack-Net [43] introduces a specialized deep learning framework for predicting crack propagation and stress–strain curves in particulate composite materials. By incorporating an implicit constraint that links the evolution of cracks to mechanical responses within the network architecture, Crack-Net enables data-efficient, long-term predictions of fracture behavior across diverse material morphologies and interface conditions. To address challenges related to complex geometries and irregular computational grids, the Fusion DeepONet framework [36] further extends the classical DeepONet by introducing a multi-scale conditioned neural field. This allows for accurate and geometry-aware predictions across heterogeneous domains, making it particularly suitable for modeling responses in evolving or non-uniform materials where spatial variations and structural features play a critical role.

Building on this foundation, the present work integrates DeepONets with the CPD framework to predict particle-level displacements and fracture evolution. This data-driven methodology combines the physical interpretability of CPD with the expressive representational capacity of neural operators, enabling accurate and efficient prediction of crack propagation across varying material configurations without the need for retraining on each new geometry.

In this study, we evaluate and compare the performance of vanilla DeepONet and Fusion DeepONet architectures for predicting crack evolution over 100 time steps. To assess the models' generalization and robustness, we design three representative scenarios:

1. Varying the pre-crack notch height above a hole with fixed radius ($r = 1$ cm),

2. Repeating the first scenario while explicitly modeling failed elements in the domain, and
3. Varying the hole radius while maintaining a fixed notch position located 1.5 cm above the hole center.

These cases serve to test the models' ability to generalize across diverse geometric and failure configurations inherent to fracture dynamics in complex materials.

The paper is structured as follows: Section 2 provides a comprehensive review of the Constitutively informed Particle Dynamics (CPD) framework and its general formulation. In Section 3, we discuss the modeling of geometric discontinuities and brittle cracks, as well as their influence on crack path evolution. Section 4 introduces the DeepONet and Fusion DeepONet architectures, along with the details of the training process and network design. In Section 5, we present an analysis of the performance of the vanilla and Fusion DeepONet models for data-driven crack propagation prediction, and the paper concludes with a summary in Section 6.

2 Constitutively informed Particle Dynamics

The CPD model introduces a way for the discrete system to describe a general constitutive behavior through the definition of interparticle forces via continuum free energy [14]. Multiple forms of free energy have been used with CPD, such as non-convex free energy for solid phase transformations [15,16], neo-Hookean free energy for hyperelastic behaviours [17], and combinations of these with isotropic elastic free energy to describe heterogeneous composites [18,44]. This section provides a brief overview of the formulation. Detailed descriptions of the model and its applications for characterizing evolving cracks and microstructure can be found in [14].

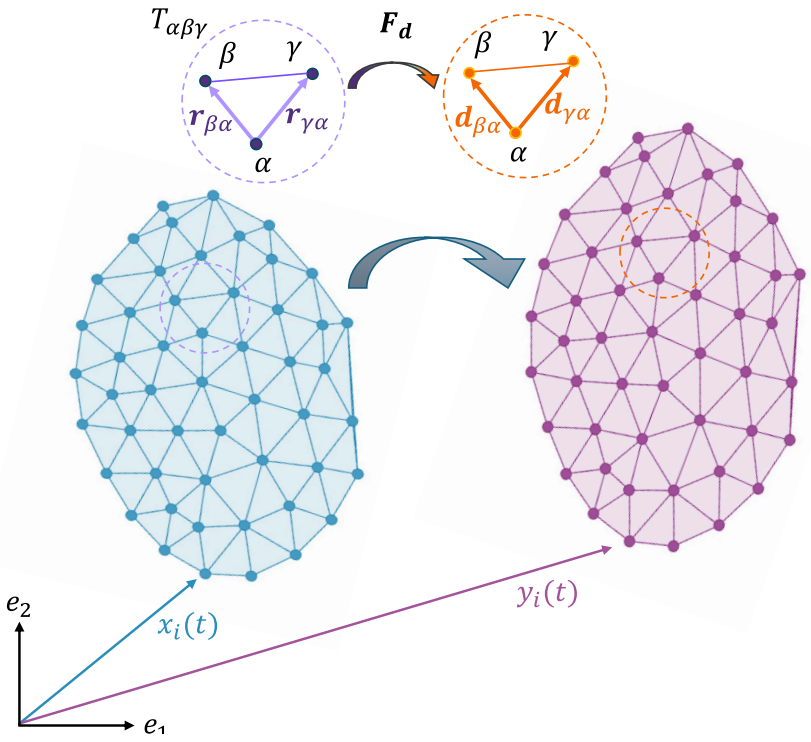


Figure 1: Discretization and mapping of deformation of the domain through particle displacement.

The domain is treated as a collection of irregularly distributed interacting particles. The model describes the behaviour of solid materials through the three particle interactions using the reference configuration $\boldsymbol{x}_i(t)$ and the deformed configuration represented as $\boldsymbol{y}_i(t)$ as shown in Figure 1. By defining the inter-particle forces as a function of $\boldsymbol{x}_i(t)$, $\boldsymbol{y}_i(t)$ and material properties through

continuum free energy, the current trajectory of the particle $\boldsymbol{y}_i(t)$ is obtained by solving Newton's equations of motion for every particle. The governing equations are given,

$$m_i \frac{\mathrm{d}^2 \boldsymbol{y}_i}{\mathrm{d}t^2} + c \frac{\mathrm{d}\boldsymbol{y}_i}{\mathrm{d}t} = \boldsymbol{f}_i, \quad i = 1, 2, \ldots, N, \tag{1}$$

where m_i denotes the mass of particle i, and c represents a damping coefficient. The net force $\boldsymbol{f}_i$ on particle i, results from interacting neighbor particles. The interacting neighbors for a particle are identified through Delaunay triangulation in reference configuration, as shown in Figure 1.

The specific triangle $\mathcal{T}_{\alpha\beta\gamma}$ composed of particles α, β, and γ in reference and deformed configurations is shown as the inset in Figure 1 to demonstrate the methodology to obtain multi-body interaction. The deformation of the triangle $\mathcal{T}_{\alpha\beta\gamma}$ described by relative position vectors in reference and current configurations, respectively, as follows:

$$\begin{aligned} \boldsymbol{r}_{\gamma\alpha} &= \boldsymbol{x}_\gamma - \boldsymbol{x}_\alpha, \quad \boldsymbol{r}_{\beta\alpha} = \boldsymbol{x}_\beta - \boldsymbol{x}_\alpha, \quad \boldsymbol{r}_{\beta\gamma} = \boldsymbol{x}_\beta - \boldsymbol{x}_\gamma, \\ \boldsymbol{d}_{\gamma\alpha} &= \boldsymbol{y}_\gamma - \boldsymbol{y}_\alpha, \quad \boldsymbol{d}_{\beta\alpha} = \boldsymbol{y}_\beta - \boldsymbol{y}_\alpha, \quad \boldsymbol{d}_{\beta\gamma} = \boldsymbol{y}_\beta - \boldsymbol{y}_\gamma. \end{aligned} \tag{2}$$

A linear transformation function $\boldsymbol{F}_d$ is assumed for each Delaunay triangle $\mathcal{T}_{\alpha\beta\gamma}$, that maps the reference and the deformed configurations such that $\boldsymbol{d}_{\beta\alpha} = \boldsymbol{F}_d \boldsymbol{r}_{\beta\alpha}$ and $\boldsymbol{d}_{\gamma\alpha} = \boldsymbol{F}_d \boldsymbol{r}_{\gamma\alpha}$. By solving the pair of equations for each triangle, the discrete deformation tensor $\boldsymbol{F}_d$ expressed as a function of particle positions $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ $\quad i = \alpha, \beta,$ and γ,

$$\boldsymbol{F}_d = \left(\boldsymbol{d}_{\beta\alpha} \otimes \boldsymbol{e}_1 + \boldsymbol{d}_{\gamma\alpha} \otimes \boldsymbol{e}_2\right) \left(\boldsymbol{r}_{\beta\alpha} \otimes \boldsymbol{e}_1 + \boldsymbol{r}_{\gamma\alpha} \otimes \boldsymbol{e}_2\right)^{-1}. \tag{3}$$

The expression $\boldsymbol{a} \otimes \boldsymbol{b}$ denotes the dyadic product of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. The energy of each triangle is expressed as a function of the discrete analogue of the Lagrangian strain tensor ($\boldsymbol{E_d} = \frac{1}{2}(\boldsymbol{F}_d^T \boldsymbol{F}_d - \boldsymbol{I}$) as

$$W_{\alpha\beta\gamma} = V_{\alpha\beta\gamma} \tilde{\psi}\left(\boldsymbol{E}_d\right). \tag{4}$$

The interparticle interactions are defined from the macroscopic constitutive response of the material through the total energy of the body $W = \sum W_{\alpha\beta\gamma}$, where the sum is taken over all Delaunay triangles in the domain. The free energy is given by $\tilde{\psi} = \frac{1}{2}\boldsymbol{E}^T \mathbf{C} \boldsymbol{E}$, where $\boldsymbol{C}$ is the constitutive matrix of the material. The interaction force on particle i with all its neighbors is then

$$\boldsymbol{f}_i = -\partial W / \partial \boldsymbol{y}_i\,, \quad \text{where} \quad W = \sum V_{\alpha\beta\gamma} \tilde{\psi}\left(\boldsymbol{E}_d\right). \tag{5}$$

2.1 Description of brittle crack

Brittle failure is described using the maximum principal stress failure criterion. The stress is evaluated for every triangle in the domain using the corresponding Lagrangian strain $\boldsymbol{E}_d$ and constitutive relation. The maximum principal stress ($\sigma_1 > \sigma_2$) is compared with the critical value,

$$\sigma_1 \leq \sigma_t^U; \quad \sigma_2 \geq -\sigma_c^U \tag{6}$$

where σ_U^t and σ_U^c are the ultimate tensile and compressive strengths, respectively. The maximum principal stress (σ_1) of each triangle is continuously monitored during simulation, and if the triangle stress satisfies the failure criteria $\sigma_1 \geq \sigma_t^U$ or $\sigma_2 \leq -\sigma_c^U$, the interaction forces associated with that triangle are set to zero, and the corresponding particles no longer interact through that particular triangle.

Consider six particles as shown in Figure 2 which are interacting through the respective Delauney triangles. Since particle i is interacting with all other five neighboring particles through triangles $T_1 - T_5$ the interaction force on particle i can be obtained by

$$\boldsymbol{f}_i = \sum_{j=1}^{5} -\frac{\partial W_{T_j}}{\partial \boldsymbol{y}_i}; \quad W_{T_j} = W_{\alpha\beta\gamma}, \tag{7}$$

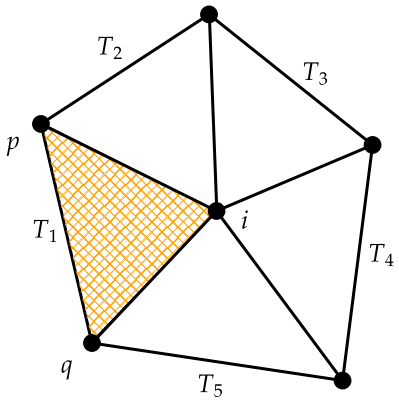


Figure 2: The particle i and its neighbors interacting through Delauney triangles T_i; $i = 1, 2 \ldots 5$. The strain energy density of T_1 at a particular instance shown satisfies the failure criteria $\sigma_1^{T_1} = \sigma_U^t$.

W_{T_j} denote the strain energy of the triangle T_j and it is evaluated using $W_{\alpha\beta\gamma}$ from Eq. 4 where α, β, and γ denote the corresponding particles in the respective triangle T_j. Specifically the particles p, q and i are interacting through triangle T_1. While the system evolves, consider the failure condition $\sigma_1^{T_1} = \sigma_U^t$ is satisfied for a triangle T_1. Then the interaction force from triangle T_1 on particles p, q and i are made zero. The particle interaction between p and q are completely removed but the particles p and i interacting through T_2 and the particles q and i interacting through T_5.

3 Modeling geometric discontinuity and brittle crack

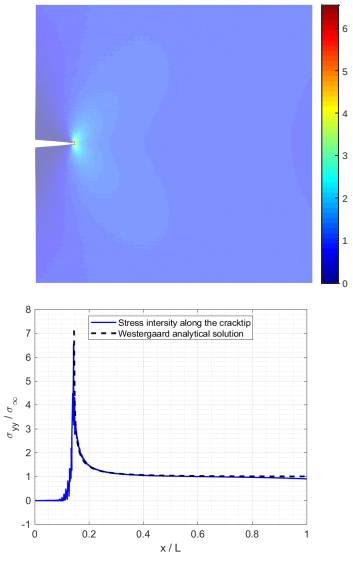
To demonstrate the model's ability to capture stress concentrations around discontinuities, three benchmark cases were considered: a pre-existing sharp crack (Figure 3a), a circular hole (Figure 3b), and a circular hole interacting with a pre-existing crack (Figure 3c). The domain is discretized with approximately 16,000 particles with over 30,000 multi-body interaction triangulations that are identified using Delaunay triangulation as shown in Figure 5b. Displacement-based loading is applied to particles on the top and bottom boundaries, incremented in small steps until a predefined value is reached to ensure a quasi-static loading scenario. Newton's equations of motion are solved for each particle using an isotropic elastic strain energy function, with Young's modulus $E = 210\,\text{GPa}$ and Poisson's ratio $\nu = 0.3$, as presented in [14].

The distribution of stress concentration around the crack tip and circular hole is shown in Figure 3a, and Figure 3b, respectively. The stress profiles along the crack tip and the horizontal line passing through the center of the circular hole are compared with analytical solutions given by Westergaard [45] and Kirsch [46], respectively. The numerical results exhibit good agreement with the analytical expressions, given in Eq. 8 for the crack tip and Eq. 9 for the hole. Figure 3c further illustrates the stress distribution and interaction in the domain containing both a crack and a circular hole.

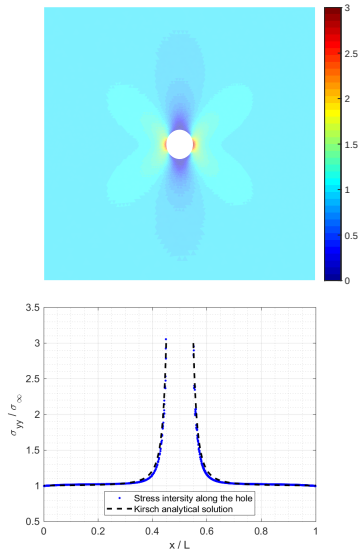
$$\sigma_{yy} = \frac{\sigma_\infty}{\sqrt{1 - \left(\frac{a}{x}\right)^2}} \tag{8}$$

$$\sigma_{yy} = \frac{\sigma_\infty}{2}\left(1 + \left(\frac{r}{x}\right)^2\right) + \frac{\sigma_\infty}{2}\left(1 + 3\left(\frac{r}{x}\right)^4\right) \tag{9}$$

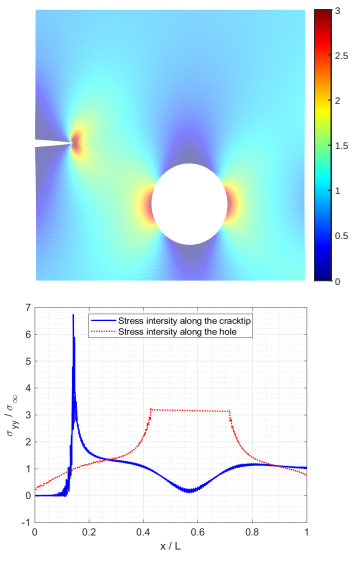
where a is the length of the pre-existing crack, r is the radius of the hole, and x is the horizontal distance along the discontinuity, as shown in Fig. 5a.



(a) Stress concentration near the crack tip.



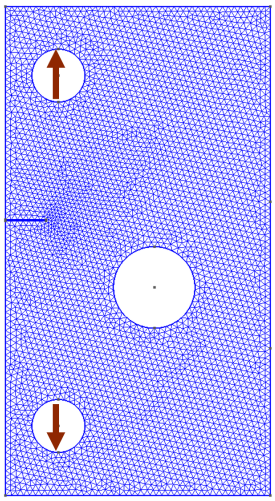
(b) Stress concentration around the hole.



(c) Stress concentration near the cracktip and hole.

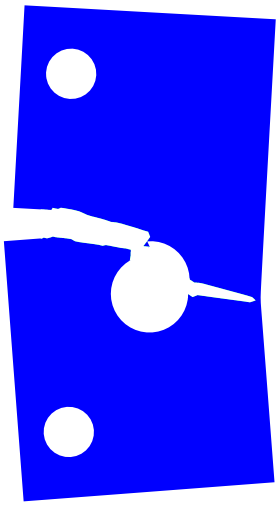
Figure 3: Stress field concentrations around geometric discontinuities in the domain.

Similarly, to demonstrate the model's capability in capturing crack initiation and propagation due to the interaction of stress fields between a pre-existing crack tip and a nearby hole, the fracture mechanism is incorporated as described in Section 2.1. Figure 4 illustrates the model's ability to predict the crack path, which aligns well with experimental observations. Figure 4a shows the domain geometry and applied boundary conditions. The experimentally observed crack trajectory is shown in Figure 4b, reproduced from [47]. The numerically predicted deformation and crack propagation path, just before complete separation, are presented in Figure 4c, demonstrating excellent agreement with the experimental reference.



(a) Domain geometry and boundary conditions used in the simulation.

(b) Experimentally observed crack path reproduced from [47].

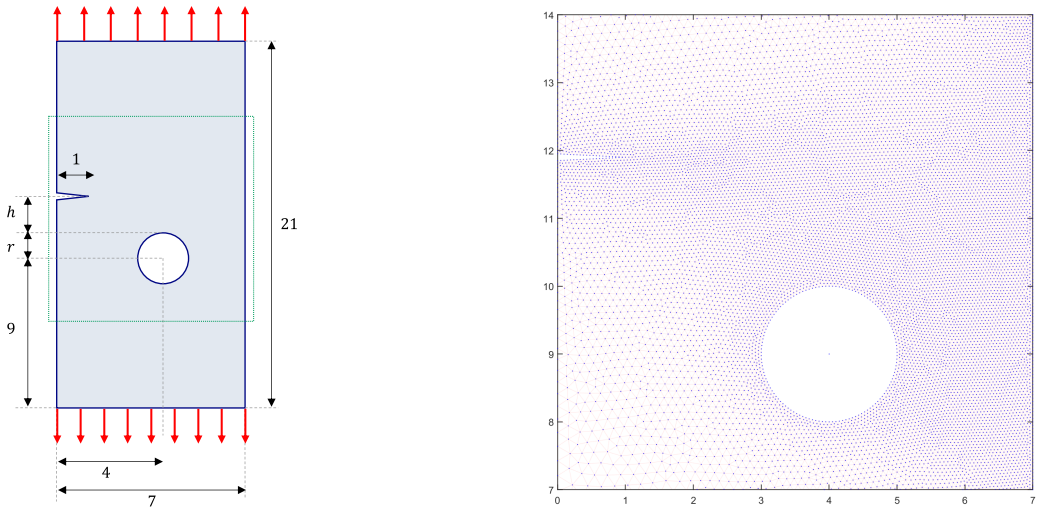


(c) Simulated deformation and crack path prior to complete separation.

Figure 4: Comparison of predicted crack path with experimental results and domain setup.

3.1 Influence of geometric discontinuities on crack path

The rectangular domain with a pre-existing crack and a circular hole, adopted from [48], is considered for the analysis, as shown in Figure 5. The green box in Figure 5a indicates the region of interest where crack propagation is examined, and subsequent figures show a cropped view of this area. The domain is discretized using approximately 16,000 particles, forming over 30,000 multi-body interaction triangulations, as illustrated in Figure 5b.



(a) (b)

Figure 5: (a) Geometry of domain with hole and pre-existing crack along with the loading conditions. The green dotted box highlights the region for crack path analysis. (b) Reference particle configuration and multi-body interaction triangulation for the zoomed-in region.

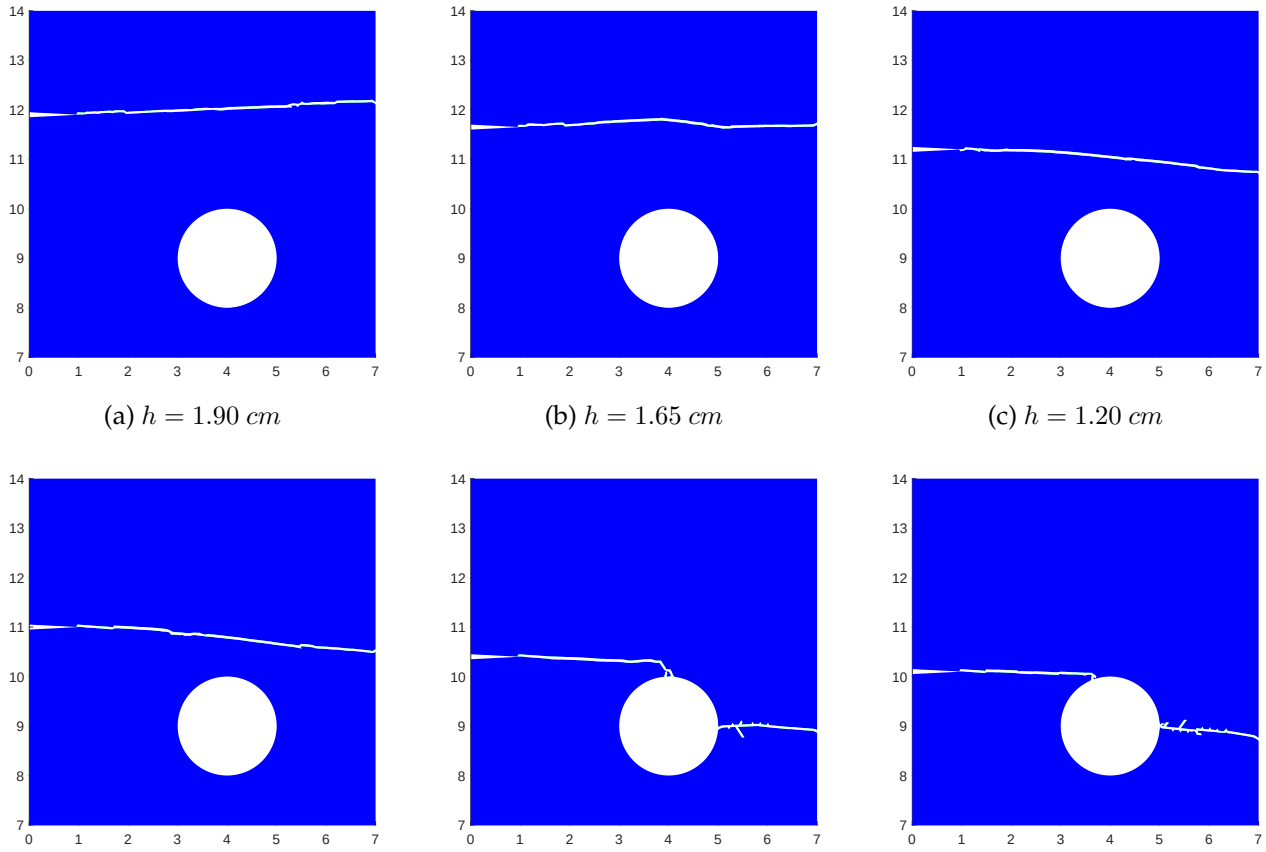
To simulate and generate data for training and testing the operator learning framework on particle displacement and crack evolution, three representative cases are considered. These cases vary in geometric parameters and fracture assumptions, as summarized in Table 1. For each case, the reference particle configuration is recorded at $\tau = 0$, and the deformed configurations are recorded over 100 equally spaced simulation stages, from $\tau = 1$ to $\tau = 100$, for use in training and testing the model. Here, $\tau \in [0, 100]$ denotes the deformation time-step index, representing 100 equally spaced steps from the undeformed configuration ($\tau = 0$) to the fully deformed state ($\tau = 100$).

Table 1: Geometric and mesh details for three representative case studies.

Case	Parametrization Strategy	Radius (cm)	Height (cm)	Number of Samples	Fracture Accounted
Case 1	Varying pre-crack notch height	$r = 1$	$h = 0$ to 2	40	No
Case 2	Varying pre-crack notch height	$r = 1$	$h = 0$ to 2	50	Yes
Case 3	Varying hole radius	$r = 0.5$ to 1.5	$h = 1.5$	51	Yes

In **Case 1**, a total of forty rectangular specimens are analyzed, each containing a circular hole of fixed radius $r = 1$ cm. The height parameter h, which defines the vertical distance between the pre-existing horizontal crack and the circular hole, is systematically varied from $h = 0$ cm to $h = 2$ cm. These simulations are carried out without assuming any failure criteria, thereby focusing purely on elastic deformation and particle displacement without fracture. In **Case 2**, the same rectangular

domain geometry is retained with a fixed circular hole of radius $r = 1$ cm, but this time fifty different configurations are considered by varying the height h from $h = 0$ cm to $h = 2$ cm. Unlike the first case, these simulations incorporate fracture evolution, enabling the study of crack initiation and propagation in relation to the geometry. **Case 3** explores the influence of the hole radius on crack behavior. Here, the vertical distance h is fixed at $h = 1.5$ cm, and the radius r of the circular hole is varied from $r = 0.5$ cm to $r = 1.5$ cm. Fifty-one unique specimens are simulated in this case, all of which include fracture modeling.

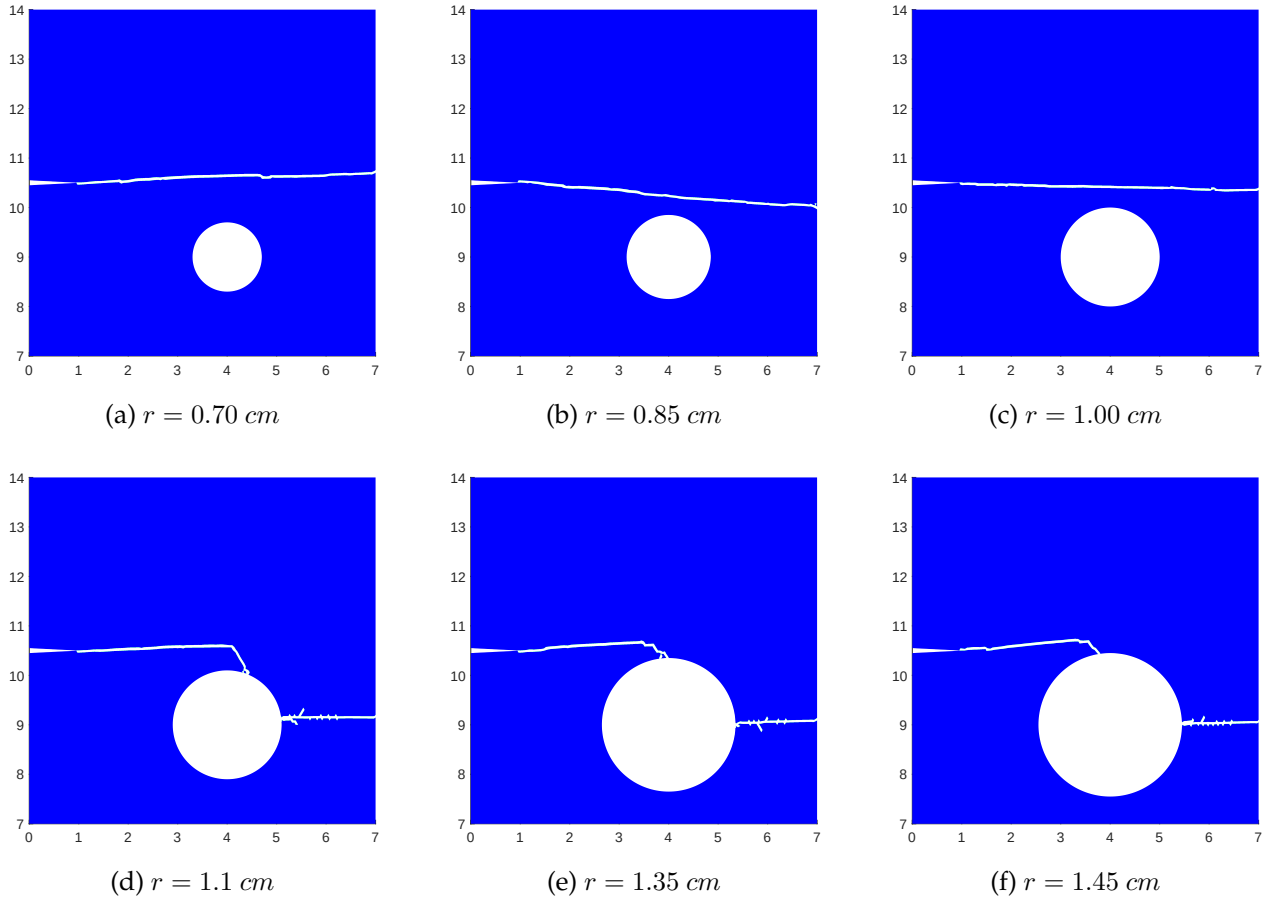


(a) $h = 1.90\ cm$ (b) $h = 1.65\ cm$ (c) $h = 1.20\ cm$

(d) $h = 1.00\ cm$ (e) $h = 0.40\ cm$ (f) $h = 0.10\ cm$

Figure 6: **Case 2** Crack paths simulated by varying the height (h) of pre-existing crack from the hole of constant radius of $r = 1\ cm$.

Simulated crack paths for selected specimens from the second and third cases are shown in Figures 6(a–f) and 7(a–f), respectively, focusing on the zoomed-in region highlighted by the green dotted box in Figure 5. In this analysis, the key geometric parameters influencing crack propagation include the initial crack length a, the radius of the circular hole r, and the vertical separation h between the crack tip and the hole center. For all simulations, the crack length is fixed at $a = 1$ cm. It is observed that as the vertical distance h is reduced, the crack begins to deviate from its original direction, increasingly interacting with the stress field generated by the hole. This interaction can result in the crack deflecting toward, and eventually merging with, the hole. A similar effect is noted as the radius r increases, it promotes crack deflection and eventual coalescence. For a fixed pre-existing crack length $a = 1$ cm, this transition is consistently observed when the dimensionless parameter $(r + h)/r \leq 1.4$, where the geometric configuration favors crack–hole interaction, often resulting in the crack being trapped by the hole and eventual passing through the hole on further loading.



(a) $r = 0.70\ cm$ (b) $r = 0.85\ cm$ (c) $r = 1.00\ cm$

(d) $r = 1.1\ cm$ (e) $r = 1.35\ cm$ (f) $r = 1.45\ cm$

Figure 7: **Case 3** Crack paths simulated by varying the radius (r) of the hole positioned at $h = 1.5\ cm$ below the pre-existing crack.

4 Neural Operator Learning with DeepONet

In conventional machine learning, models are typically trained to learn relationships between finite-dimensional vectors. However, such formulations are often inadequate for scientific and engineering problems that require learning mappings between functions defined over continuous domains—that is, in infinite-dimensional spaces. Operator learning overcomes this limitation by approximating mappings of the form

$$\mathcal{G} : u(\xi) \mapsto v(\xi),$$

where both the input $u(\xi)$ and the output $v(\xi)$ are functions defined on a continuous domain [22, 30, 36]. This framework is particularly well-suited for surrogate modeling of parametric partial differential equations (PDEs), where the objective is to learn a solution operator that maps input conditions—such as initial states, boundary values, or source terms—to the corresponding output fields.

4.1 DeepONet Setup

Let $\mathcal{G} : \mathcal{A} \to \mathcal{B}$ be a (possibly nonlinear) operator between function spaces $\mathcal{A}$ and $\mathcal{B}$, defined over a spatial domain $D \subset \mathbb{R}^d$. Given a set of training examples $\{(a_i, b_i)\}_{i=1}^N$, where $b_i = \mathcal{G}(a_i)$, the goal is to learn an approximation $\hat{\mathcal{G}}$ such that

$$\hat{\mathcal{G}}(a) \approx \mathcal{G}(a), \quad \text{for all } a \in \mathcal{A}.$$

In other words, the learned operator should generalize well to both training and unseen input functions.

DeepONet is a neural network architecture designed to approximate such operators. It consists of two subnetworks:

- **Branch network:** This network encodes the input function f, evaluated at a fixed set of sensor points $\{\xi_1, \xi_2, \ldots, \xi_k\}$, and maps it to a latent feature vector $\boldsymbol{b} \in \mathbb{R}^p$. For multiple input samples, this yields a matrix $B \in \mathbb{R}^{m \times p}$, where each row corresponds to one input function.
- **Trunk network:** This network takes an evaluation point $\xi \in D$ and produces a coordinate-dependent feature vector $\boldsymbol{\phi}(\xi) \in \mathbb{R}^p$, forming a learned basis for reconstructing the output. For a set of evaluation points, it generates a matrix $\Phi \in \mathbb{R}^{n \times p}$.

The final prediction of the operator applied to input function f_i at a point ξ_j is given by the inner product of the branch and trunk outputs:

$$\mathcal{G}(f_i)(\xi_j) \approx \sum_{l=1}^{p} B_{il}(f_i; \theta_B) \cdot \Phi_{jl}(\xi_j; \theta_T), \tag{10}$$

where θ_B and θ_T are the trainable parameters of the branch and trunk networks, respectively.

The displacement field $\boldsymbol{u}(\boldsymbol{x}_i, t)$ is defined as the difference between the deformed configuration $\boldsymbol{y}_i(t)$ and the reference configuration $\boldsymbol{x}_i$ of particle i. Mathematically, this is expressed as

$$\boldsymbol{u}(\boldsymbol{x}_i, t) = \boldsymbol{y}_i(t) - \boldsymbol{x}_i,$$

where $\boldsymbol{x}_i = \boldsymbol{y}_i(0)$ denotes the initial (undeformed) position. This formulation is implemented numerically by computing the displacement at each time step as the difference between the current and initial particle positions.

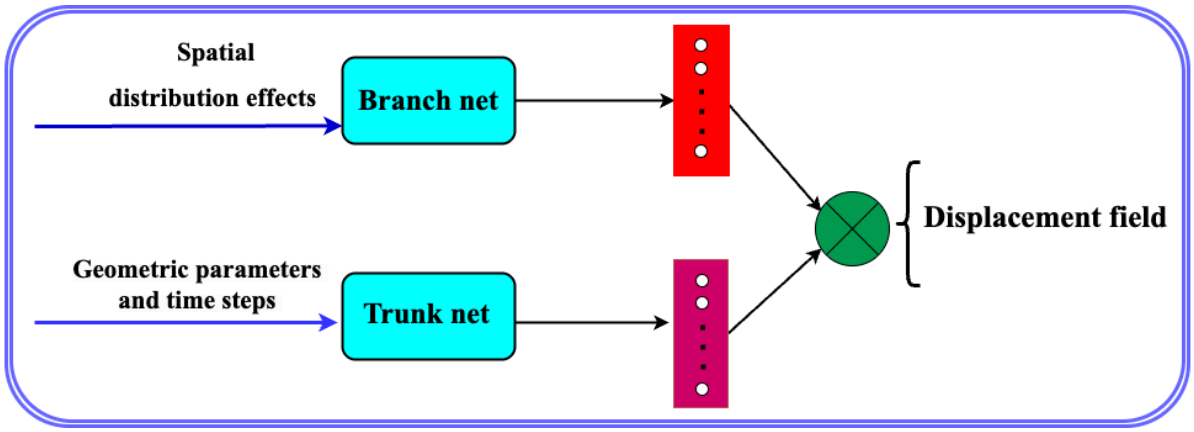


Figure 8: Schematic of the DeepONet architecture for displacement field prediction. The branch network encodes input functions related to spatial variations, such as pre-crack notch height or hole radius. The trunk network processes the spatial coordinate $\boldsymbol{x} \in \mathbb{R}^2$ and time t. The output is the predicted displacement field $\boldsymbol{u}(\boldsymbol{x}, t)$.

Figure 8 illustrates the DeepONet architecture used in this work. The decomposition in (10) decouples the encoding of the input function from the spatial-temporal representation of the output, making DeepONet a powerful and flexible framework for learning nonlinear operators in complex physical systems. In this setup, the input to the branch network consists of the initial geometry parameters, such as the pre-crack notch height h_i and the hole radius r_i. The trunk network takes as input the spatial coordinate $\boldsymbol{x} \in \mathbb{R}^2$ and time t. The final output is the predicted displacement field $\boldsymbol{u}(\boldsymbol{x}, t)$.

4.2 Fusion-DeepONet

Fusion DeepONet is an advanced variant of the standard DeepONet, designed to improve data efficiency and predictive accuracy in geometry-dependent problems defined on arbitrary or irregular grids [36]. Like the original architecture, Fusion DeepONet consists of two subnetworks: a branch network that encodes input functions or parameters (e.g., geometric features, initial or boundary conditions), and a trunk network that processes spatial and/or temporal coordinates.

The key innovation in Fusion DeepONet lies in its multi-scale feature fusion mechanism, which conditions each hidden layer of the trunk network on the corresponding hidden layer of the branch network. Let $\mathbf{S}_B^{(l)} \in \mathbb{R}^w$ denote the cumulative feature vector obtained by summing all hidden features from the first to the $(l-1)$-th layer of the branch network, and let $\mathbf{a}_T^{(l)} \in \mathbb{R}^w$ be the hidden feature vector at the l-th layer of the trunk network. The updated trunk representation at layer l is given by:

$$\mathbf{a}_T^{(l)} = \mathbf{S}_B^{(l)} \odot \sigma\left(\boldsymbol{W}_T^{(l)} \mathbf{a}_T^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{11}$$

where $\boldsymbol{W}_T^{(l)} \in \mathbb{R}^{w\times w}$ is a weight matrix, $\mathbf{b}^{(l)} \in \mathbb{R}^w$ is a bias vector, σ is a nonlinear activation function, and $\odot$ denotes element-wise (Hadamard) multiplication.

This layer-wise fusion allows the trunk network to adaptively incorporate geometry-dependent information at multiple scales, enhancing its expressivity in modeling complex spatio-temporal behavior—particularly in regions with high gradients or discontinuities.

Similar to the vanilla DeepONet, the final output is obtained via a contraction (inner product) between the outputs of the branch and trunk networks:

$$\boldsymbol{u}(\xi) = \sum_{k=1}^{p} \tilde{\boldsymbol{b}}_k(\mu) \cdot \tilde{t}_k(\xi), \tag{12}$$

where $\tilde{\boldsymbol{b}}_k(\mu) \in \mathbb{R}^d$ denotes the k-th component of the vector-valued output from the final layer of the branch network, evaluated at the input parameters μ (e.g., geometric or boundary features), and $\tilde{t}_k(\xi) \in \mathbb{R}$ represents the corresponding scalar basis function from the fused trunk network, evaluated at the spatial-temporal coordinate $\xi \in \mathbb{R}^n$. For additional details on the Fusion DeepONet architecture, see [36].

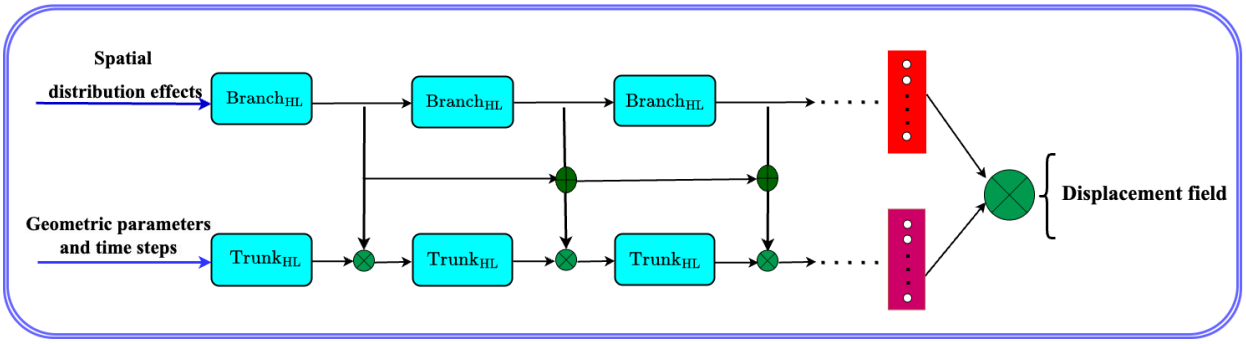


Figure 9: Schematic of the Fusion DeepONet architecture for displacement field prediction. The branch network encodes geometry-dependent input parameters such as the initial crack height h_i or hole radius r_i. The trunk network takes as input the spatial coordinate $\boldsymbol{x} \in \mathbb{R}^2$ and time t . Each hidden layer of the trunk is conditioned on the corresponding hidden layer of the branch network (denoted as $\texttt{Branch}_{\texttt{HL}}$ and $\texttt{Trunk}_{\texttt{HL}}$), while the final layer of the branch provides global conditioning to the output.

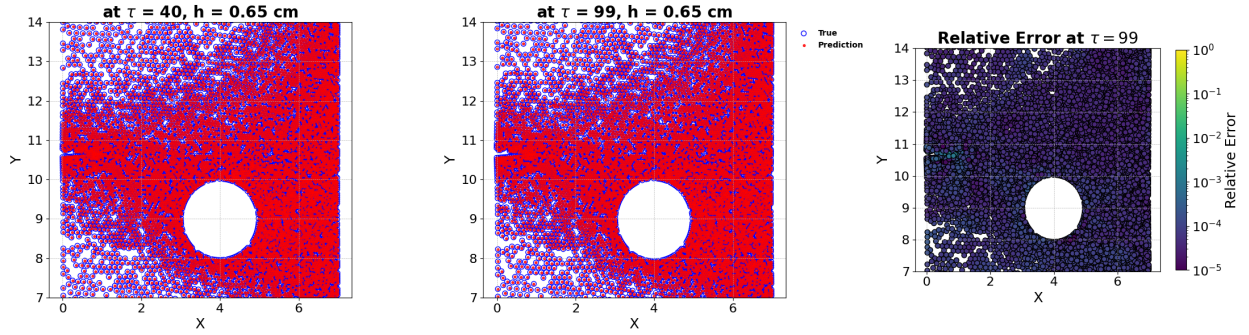
Figure 9 illustrates the Fusion DeepONet architecture applied to displacement field prediction. In this formulation, the branch network captures input-dependent features such as variations in crack height h_i or hole radius r_i, while the trunk network processes the query coordinates $\boldsymbol{x} \in$

$\mathbb{R}^2$ and time t. Unlike the standard DeepONet, the Fusion DeepONet incorporates hierarchical conditioning, where each hidden layer in the trunk is dynamically modulated by the corresponding hidden layer from the branch network. In addition, the final output of the branch network provides global conditioning at the output stage.

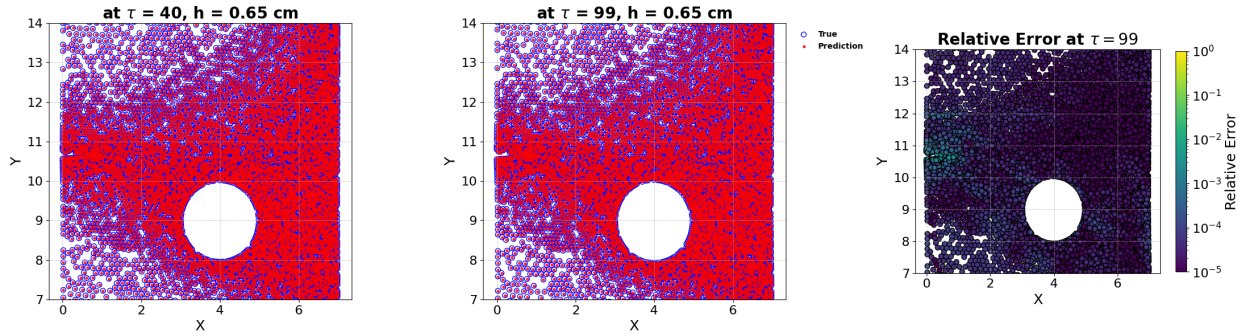
This layered interaction enables multi-scale feature fusion, allowing coarse geometric information to guide the learning of both low- and high-frequency components of the solution. As a result, the architecture is particularly effective at modeling solutions with sharp features or irregular geometries. To maintain consistent modulation across layers, the architecture assumes matching layer depths and widths in both branch and trunk networks, except for the final linear layer in the branch. Although this design enhances learning capacity and generalization, it introduces additional computational overhead compared to the vanilla DeepONet, especially during training.

5 Performance of DeepONet and Fusion DeepONet

In this section, we compare the performance of the vanilla DeepONet and the Fusion-DeepONet in predicting crack paths across the three case studies listed in Table 1.



(a) Case 1: Prediction of particle positions using **Fusion DeepONet**.



(b) Case 1: Prediction of particle positions using **vanilla DeepONet**.

Figure 10: Comparison of predicted and true particle positions at two time steps, $\tau = 40$ and $\tau = 99$, for **Case 1**, using (a) Fusion DeepONet and (b) vanilla DeepONet architectures. Red points represent the predicted particle positions, while hollow blue circles denote the true positions. The figure on the right shows the relative error at $\tau = 99$, comparing the predicted and true particle positions.

Figure 10 illustrates the predictive performance of (a) the Fusion DeepONet and (b) the vanilla DeepONet for **Case 1**, comparing predicted and true particle positions at two representative time steps: $\tau = 40$ and $\tau = 99$. The panel on the right shows the relative error at $\tau = 99$ for both architectures. In **Case 1**, the pre-crack notch height is varied while the hole radius is fixed at $r = 1$ cm (see Table 1). The dataset contains 40 samples, split into 35 for training and 5 for testing.

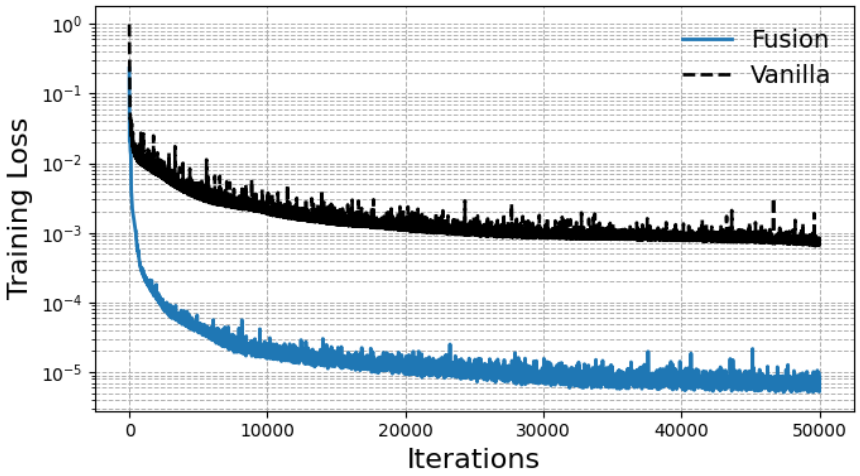


Figure 11: Case 1 — Comparison of training loss between the Fusion DeepONet and the vanilla DeepONet over 50000 iterations.

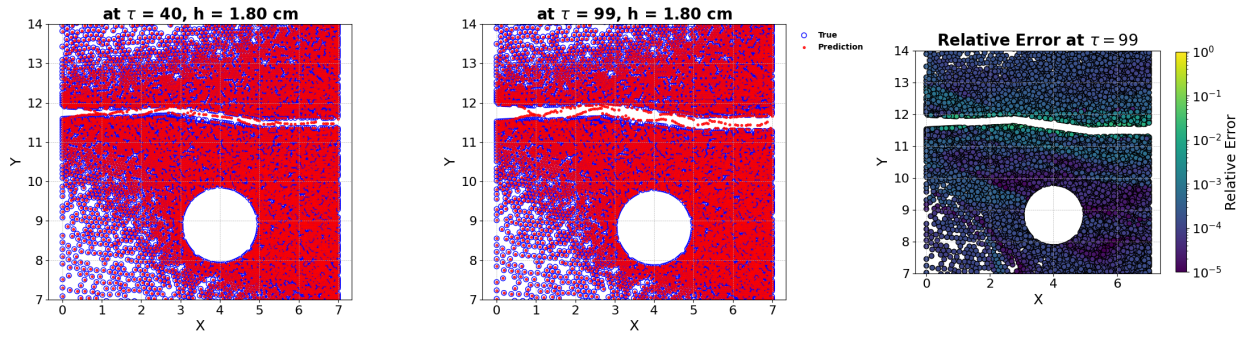
Since no elements fail in this scenario, particles experience only incremental displacements over time without any fracture events.

The vanilla DeepONet architecture employs five hidden layers in both the branch and trunk networks, each with 100 neurons, and a latent dimension of 200 for the final basis projection. The `tanh` activation function is used throughout, and training is performed using the Adam optimizer from the `optax` library with a fixed learning rate of 1×10^{-4}. In contrast, the Fusion DeepONet consists of three hidden layers in both subnetworks, each with 64 neurons, and a final latent layer also of size 64 for trunk subnetwork and 2×64 for branch subnetwork. It utilizes the Rowdy activation function [49], a modified form of `tanh` designed to enhance expressive feature learning. The model is trained using the Adam optimizer with an exponentially decaying learning rate, starting from 1×10^{-3}, with a decay step of 2000 and a decay rate of 0.91.

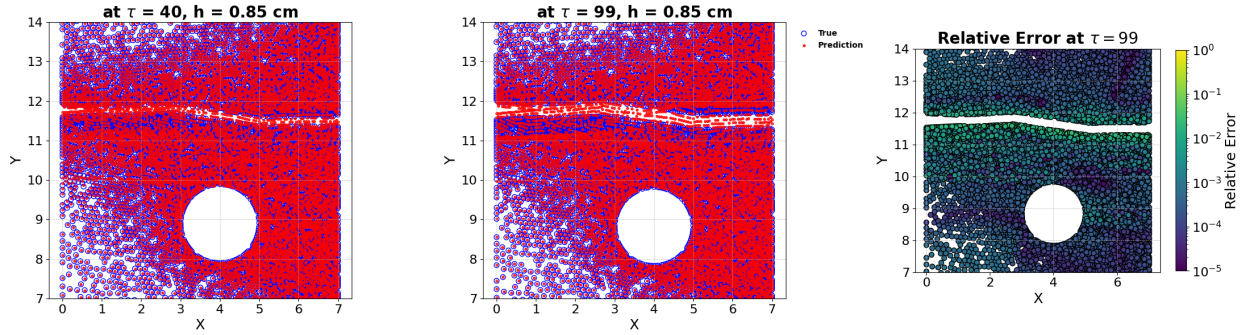
In Figure 10, red dots indicate predicted particle positions, while hollow blue circles denote the corresponding ground truth. Each red-blue pair corresponds to the same particle at a given time step, and their overlap visually reflects the prediction accuracy. The comparison demonstrates that both models can learn the particle dynamics, with Fusion DeepONet showing visibly improved alignment. Figure 11 shows the evolution of training loss, measured as mean squared error (MSE), for both models. The Fusion DeepONet exhibits faster and more stable convergence, achieving an MSE as low as 10^{-5}, compared to the vanilla DeepONet, which plateaus around 10^{-3}.

As summarized in Table 1, **Case 2** involves varying the pre-crack notch height while keeping the hole radius fixed at $r = 1$ cm. The dataset consists of 50 samples with different notch heights, of which 45 are used for training and 5 for testing. In contrast, **Case 3** considers a fixed pre-crack notch height while varying the initial hole radius over the range $r = 0.5$ cm to $r = 1.5$ cm. This dataset includes 51 samples, with 45 used for training and 6 for testing. The same architectures used in **Case 1** are employed here for both the vanilla DeepONet and Fusion DeepONet models. Specifically, the vanilla DeepONet uses five hidden layers in both the branch and trunk networks, each containing 100 neurons, and a latent dimension of 200 for the final basis projection. The Fusion DeepONet consists of three hidden layers in both subnetworks, each with 64 neurons, and a final latent layer also of size 64 for trunk network and 2×64 for branch networks.

Figures 12 and 13 illustrate the performance of both models across 100 time steps for **Case 2** and **Case 3**, respectively. In each plot, predicted and true particle positions are compared at two representative time steps: $\tau = 40$ and $\tau = 99$. Red points represent predicted particle positions, while hollow blue circles denote the corresponding ground truth. Each red-blue pair corresponds to the same particle at a specific time, and the degree of overlap between the markers reflects



(a) Case 2: Comparison between predicted and true crack path using **Fusion DeepONet**.



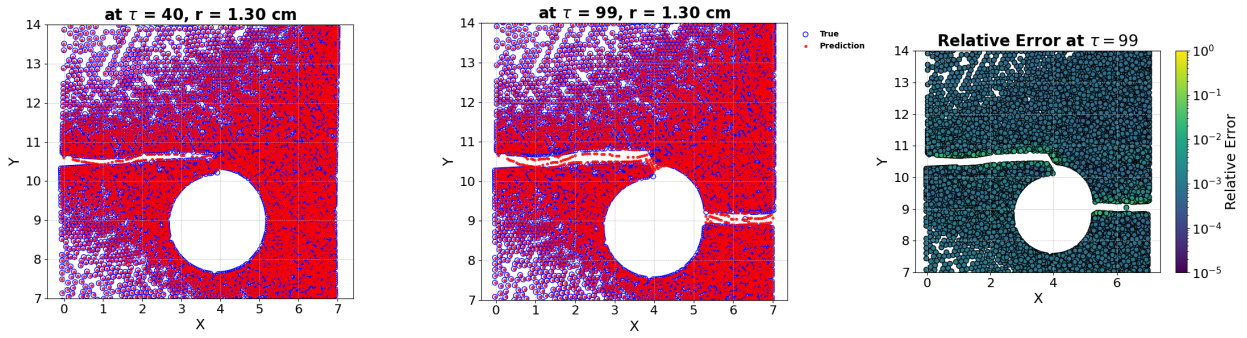
(b) Case 2: Comparison between predicted and true crack path using **vanilla DeepONet**.

Figure 12: Predicted and true crack paths at two time steps, $\tau = 40$ and $\tau = 99$, along with the relative error at $\tau = 99$, comparing particle-level predictions using (a) Fusion DeepONet and (b) vanilla DeepONet architectures for **Case 2**. Red points denote predicted particle positions, while hollow blue circles represent the corresponding ground truth. Closer alignment between red and blue markers indicates higher predictive accuracy.

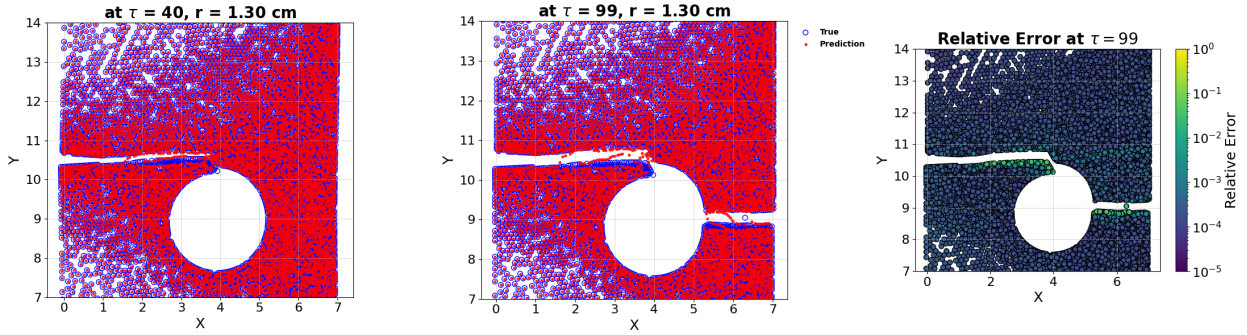
the prediction accuracy. These results demonstrate that both models effectively learn the particle dynamics and accurately predict the crack path. Notably, they capture the interaction between the crack and the hole, particularly in samples where the crack propagates near or intersects the hole boundary. Figure 12 presents the predictions for a sample with $h = 0.85$ cm, where the crack does not interact with the hole. In contrast, Figure 13 shows predictions for a sample with $r = 1.30$ cm, where the crack interacts with the hole before failure occurs. In both cases, the relative error ranges from 10^{-5} to 10^{0}. As expected, the prediction error is notably higher in regions near the crack tip and path, while it remains lower in areas farther from the crack, indicating localized difficulty in capturing the highly nonlinear fracture behavior.

Figure 14(a) presents a comparison of the relative $\mathcal{L}_2$ error over 100 time steps for all three cases, using both the vanilla DeepONet and the Fusion DeepONet. Across all scenarios, the Fusion DeepONet consistently outperforms the vanilla architecture, achieving lower prediction errors in modeling crack path evolution. Among the three cases, **Case 1** proves to be the simplest to predict. This case involves only particle displacements over time, with no element failure, making it a pure displacement prediction problem. In contrast, **Case 2** and **Case 3** involve both displacement and fracture, significantly increasing the complexity of the prediction task due to the presence of discontinuities and topological changes. Additionally, the plot reveals that the relative $\mathcal{L}_2$ error tends to increase over time, particularly in **Case 2** and **Case 3**. The observed spikes in error correspond to the onset of fracture events, while the models initially predict displacements accurately, the emergence of failed elements introduces sudden changes in the system's dynamics, making accurate prediction more difficult in later time steps.

In **Case 1**, the relative $\mathcal{L}_2$ error increases gradually over time and eventually stabilizes, reflecting



(a) Case 3: Comparison between predicted and true crack path using **Fusion DeepONet**.



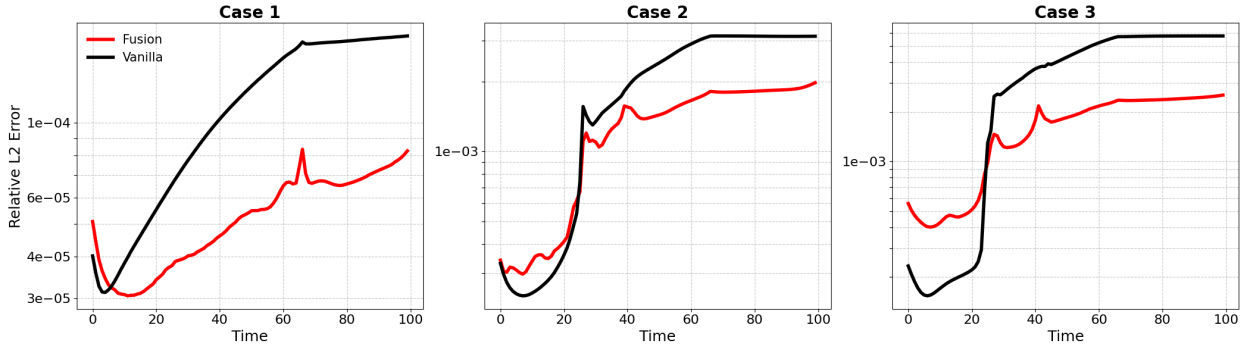
(b) Case 3: Comparison between predicted and true crack path using **vanilla DeepONet**.

Figure 13: Predicted and true crack paths at two time steps, $\tau = 40$ and $\tau = 99$, for **Case 3**, where the initial radius of the hole is varied. Red points indicate the predicted particle positions, while hollow blue circles represent the ground truth. Each red-blue pair corresponds to the same particle at a given time step. This comparison demonstrates the ability of both the Fusion and vanilla DeepONet architectures to generalize to unseen geometries. Notably, the Fusion DeepONet exhibits improved alignment with the true displacements, especially in regions where the crack path is highly sensitive to variations in hole radius.

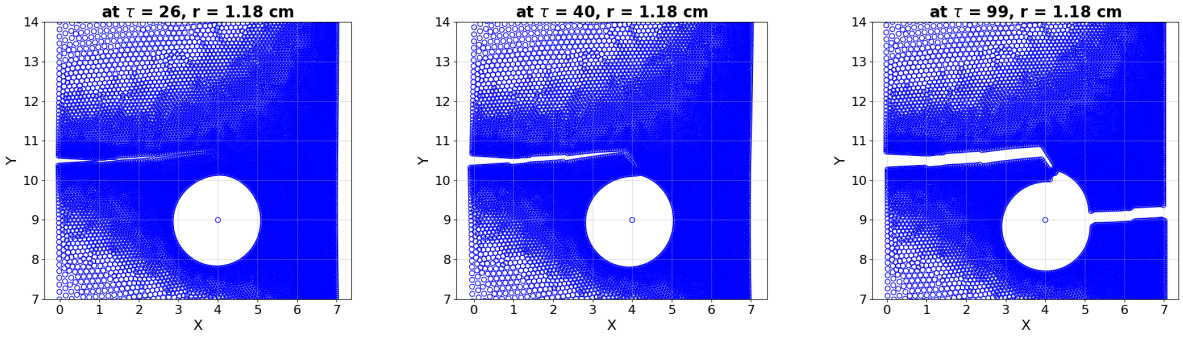
a purely elastic response with no crack propagation. This stable behavior indicates that the model effectively captures the physics of the elastic regime. In contrast, **Case 2** and **Case 3** show a sharp increase in error after approximately $\tau = 20$ time steps. As illustrated in Figure 14(b), this jump corresponds to the onset of crack propagation, where the solution begins to accountsfor material separation during fracture. A subsequent spike in error is observed around $\tau = 40$, coinciding with the moment the crack re-nucleates from the central hole in scenarios involving interaction with the hole. The onset of crack nucleation introduces geometric discontinuities and increases the complexity of the system, making the prediction task more challenging, especially for the vanilla DeepONet. The error continues to rise until approximately $\tau = 66$ after which the system is allowed to equilibriate without loading, after which it stabilizes, indicating that the system is reaching a new equilibrium state by the final time step ($\tau = 100$).

Table 2 presents the training error and training time for displacement prediction in all three cases using the vanilla DeepONet and the Fusion DeepONet. The training error for **Case 1** is lower than that of the other two cases, which is expected given the simpler physical setting—no fracture—and the smaller number of training samples (35 samples for **Case 1**, compared to 45 samples for both **Case 2** and **Case 3**).

The training time for the vanilla DeepONet is higher than that of the Fusion DeepONet. This is primarily because the vanilla DeepONet was trained for 60,000 iterations, whereas the Fusion DeepONet was trained for 50,000 iterations. These iteration counts were chosen empirically based on convergence behavior; training was stopped once the loss plateaued and no significant improve-



(a) Relative $\mathcal{L}_2$ error over 100 time steps for **Case 1**, **Case 2**, and **Case 3**.



(b) Crack propagation for **Case 3** at three selected time steps for $r = 1.18$ cm.

Figure 14: Comparison of (a) the relative $\mathcal{L}_2$ error over time for three case studies reported in Table 1, and (b) crack propagation visualization for **Case 3** at three time steps when $r = 1.18$ cm.

ment was observed.

In addition to differences in iteration counts, the architecture of the vanilla DeepONet is more complex. Both its branch and trunk networks consist of 5 hidden layers with 100 neurons per layer. In contrast, the Fusion DeepONet uses only 3 hidden layers with 64 neurons per layer in both networks. This reduced model size in the Fusion DeepONet results in a smaller computational footprint and faster training, despite achieving better predictive performance.

Table 2 also summarizes the average relative $\mathcal{L}_2$ error for the vanilla and Fusion DeepONet methods across the three case studies. While the average errors for each case are relatively close between the two methods, a more detailed analysis in Figure 14(a) reveals a key distinction. Specifically, the Fusion DeepONet consistently outperforms the vanilla DeepONet after $\tau = 20$, coinciding with the onset of crack propagation. This performance advantage becomes especially apparent as the system transitions from an elastic to a fractured state, where predictive accuracy becomes more challenging. The comparable average errors reported in the table arise because the early elastic regime dominates the error average, despite the superior performance of Fusion in the later fracture-dominated phase.

6 Summary

We investigated the application of two variants of DeepONet—vanilla and Fusion—for modeling crack propagation in specimens with varying geometries. The training data for these models was obtained from Constitutively Informed Particle Dynamics (CPD) simulations, which provide a physics-informed framework for capturing both crack nucleation and propagation. The networks were trained using inputs such as the pre-crack notch height and hole radius, which varied across

Table 2: Training time and relative error for vanilla and Fusion DeepONet.

Method	Case 1	Case 2	Case 3
Vanilla DeepONet (Time)	1752 min	3245 min	2986 min
Fusion DeepONet (Time)	996 min	1294 min	1143 min
Vanilla DeepONet (Error)	6.20e-05	9.86e-04	8.46e-04
Fusion DeepONet (Error)	4.97e-05	5.13e-04	1.01e-03

different samples. In **Case 1**, only the notch height was varied, and no element failure occurred, making it a pure displacement prediction task. In **Cases 2** and **3**, the notch height and the hole radius were varied, respectively, and fracture evolution was included by accounting for element failure. The number of training samples used in each case was 32 (varying notch heights), 45 (varying notch heights), and 45 (varying radii), respectively. In all cases, the trunk network received spatial coordinates and time over 100 time steps as input.

The Fusion DeepONet achieved lower prediction errors in modeling the spatiotemporal evolution of displacements and cracks. Among the three cases, **Case 1** exhibited the lowest overall error, as it involves only elastic displacement without crack propagation. The relative $\mathcal{L}_2$ error in this case increases gradually and eventually plateaus, indicating stable and accurate predictions throughout the elastic regime. In contrast, **Cases 2** and **3** involve fracture dynamics, making them significantly more challenging due to the presence of topological discontinuities and rapidly evolving damage. Both cases exhibit a sharp rise in relative error during the onset of crack initiation. Before this point, the networks accurately capture the displacement fields. However, as elements begin to fail, abrupt transitions caused by material separation introduce significant challenges for prediction. These spikes in error closely coincide with fracture nucleation events.

This study underscores the potential of integrating the CPD simulation framework with DeepONet-based operator learning to model complex fracture behavior in discontinuous domains. CPD's ability to capture crack nucleation and evolution—without the limitations of continuum assumptions—provides high-fidelity training data, while the Fusion DeepONet architecture enhances prediction accuracy and generalization. This hybrid approach offers a powerful and efficient surrogate modeling strategy for accelerating fracture simulations across a broad range of geometries and loading conditions.

Acknowledgments

This research was primarily supported as part of the AIM for Composites, an Energy Frontier Research Center funded by the U.S. Department of Energy (DOE), Office of Science, Basic Energy Sciences (BES), under Award #DE-SC0023389 (computational studies, data analysis). Computing facilities were provided by the Center for Computation and Visualization, Brown University.

References

[1] Eran Bouchbinder, Tamar Goldman, and Jay Fineberg. The dynamics of rapid fracture: instabilities, nonlinearities and length scales. *Reports on Progress in Physics*, 77(4):046501, 2014.

[2] M. M. A. Wahab. Review of crack arrest theory, techniques and applications on fracture control in pressure vessels. *Engineering Failure Analysis*, 145:107164, 2023.

[3] Ted Belytschko and Tom Black. Elastic crack growth in finite elements with minimal remeshing. *International journal for numerical methods in engineering*, 45(5):601–620, 1999.

[4] David Taylor, Pietro Cornetti, and Nicola Pugno. The fracture mechanics of finite crack extension. *Engineering Fracture Mechanics*, 72(7):1021–1038, 2005.

[5] Michael Ortiz, Yves Leroy, and Alan Needleman. A finite element method for localized failure analysis. *Computer methods in applied mechanics and engineering*, 61(2):189–214, 1987.

[6] Donald S Dugdale. Yielding of steel sheets containing slits. *Journal of the Mechanics and Physics of Solids*, 8(2):100–104, 1960.

[7] Patrick Diehl, Robert Lipton, Thomas Wick, and Mayank Tyagi. A comparative review of peridynamics and phase-field models for engineering fracture mechanics. *Computational Mechanics*, 69(6):1259–1293, 2022.

[8] Chenyi Luo, Lorenzo Sanavia, and Laura De Lorenzis. Phase-field modeling of drying-induced cracks: Choice of coupling and study of homogeneous and localized damage. *Computer Methods in Applied Mechanics and Engineering*, 410:115962, 2023.

[9] Laura De Lorenzis and Tymofiy Gerasimov. Numerical implementation of phase-field models of brittle fracture. In *Modeling in Engineering Using Innovative Numerical Methods for Solids and Fluids*, pages 75–101. Springer, 2020.

[10] Blaise Bourdin, Gilles A Francfort, and Jean-Jacques Marigo. Numerical experiments in revisited brittle fracture. *Journal of the Mechanics and Physics of Solids*, pages 797–826, 2000.

[11] Blaise Bourdin, Gilles A. Francfort, and Jean-Jacques Marigo. The variational approach to fracture. *Journal of Elasticity*, pages 5–148, 2008.

[12] Rohan Abeyaratne and Srikanth Vedantam. A lattice-based model of the kinetics of twin boundary motion. *Journal of the Mechanics and Physics of Solids*, 51(9):1675–1700, 2003.

[13] Rajiv K Kalia, Aiichiro Nakano, Priya Vashishta, Cindy L Rountree, Laurent Van Brutzel, and Shuji Ogata. Multiresolution atomistic simulations of dynamic fracture in nanostructured ceramics and glasses. *International Journal of Fracture*, 121:71–79, 2003.

[14] Mahendaran Uchimali, Balkrishna C Rao, and Srikanth Vedantam. Constitutively informed multi-body interactions for lattice particle models. *Computer Methods in Applied Mechanics and Engineering*, 366:113052, 2020.

[15] Mahendaran Uchimali, Balkrishna C Rao, and Srikanth Vedantam. Modeling size and orientation effects on the morphology of microstructure formed in martensitic phase transformations using a novel discrete particle model. *Acta Materialia*, 205:116528, 2021.

[16] Venkatesh Ananchaperumal, Srikanth Vedantam, and Mahendaran Uchimali. A discrete particle model study of the effect of temperature and geometry on the pseudoelastic response of shape memory alloys. *International Journal of Mechanical Sciences*, 230:107527, 2022.

[17] Phanindra Paravastu and Srikanth Vedantam. Modeling failure of hyperelastic solids interacting with fluids. *Computational Particle Mechanics*, 12(1):153–164, 2025.

[18] Venkatesh Ananchaperumal and Srikanth Vedantam. Modeling the role of phase boundaries on the pullout response of shape memory wire reinforced composites. *Mechanics of Advanced Materials and Structures*, 30(6):1128–1137, 2023.

[19] J.N. Fuhg, G.A. Padmanabha, N. Bouklas, W.C. Sun, N.N. Vlassis, M. Flaschel, P. Carrara, and L. De Lorenzis. A review on data-driven constitutive laws for solids. *Archives of Computational Methods in Engineering*, 2024.

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

[21] Manav Manav, Roberto Molinaro, Siddhartha Mishra, and Laura De Lorenzis. Phase-field modeling of fracture with physics-informed deep learning. *Computer Methods in Applied Mechanics and Engineering*, page 117104, 2024.

[22] Elham Kiyani, Manav Manav, Nikhil Kadivar, Laura De Lorenzis, and George Em Karniadakis. Predicting crack nucleation and propagation in brittle materials using deep operator networks with diverse trunk architectures. *arXiv preprint arXiv:2501.00016*, 2024.

[23] Somdatta Goswami, Cosmin Anitescu, Souvik Chakraborty, and Timon Rabczuk. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106:102447, 2020.

[24] Bin Zheng, Tongchun Li, Huijun Qi, Lingang Gao, Xiaoqing Liu, and Li Yuan. Physics-informed machine learning model for computational fracture of quasi-brittle materials without labelled data. *International Journal of Mechanical Sciences*, page 107282, 2022.

[25] Zhiying Chen, Yanwei Dai, and Yinghua Liu. Crack propagation simulation and overload fatigue life prediction via enhanced physics-informed neural networks. *International Journal of Fatigue*, 186:108382, 2024.

[26] Liqiang Lu, Xi Gao, Jean-François Dietiker, Mehrdad Shahnam, and William A Rogers. Machine learning accelerated discrete element modeling of granular flows. *Chemical Engineering Science*, 245:116832, 2021.

[27] C Li, N Zobeiry, K Keil, S Chatterjee, and A Poursartip. Advances in the characterization of residual stress in composite structures. In *Int. SAMPE Tech. Conf*, 2014.

[28] Sadjad Naderi, Boyang Chen, Tongan Yang, Jiansheng Xiang, Claire E Heaney, John-Paul Latham, Yanghua Wang, and Christopher C Pain. A discrete element solution method embedded within a neural network. *Powder Technology*, 448:120258, 2024.

[29] Saman Kazemi, Reza Zarghami, Navid Mostoufi, Rahmat Sotudeh-Gharebagh, and Riyadh I Al-Raoush. A novel ml-dem algorithm for predicting particle motion in rotary drums. *Engineering Analysis with Boundary Elements*, 177:106258, 2025.

[30] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, pages 218–229, 2021.

[31] Minglei Lu, Chensen Lin, Martin Maxey, George Em Karniadakis, and Zhen Li. Bridging scales in multiscale bubble growth dynamics with correlated fluctuations using neural operator learning. *International Journal of Multiphase Flow*, 180:104959, 2024.

[32] Benjamin Shih, Ahmad Peyvan, Zhongqiang Zhang, and George E. Karniadakis. Transformers as neural operators for solutions of differential equations with finite regularity. *arXiv preprint arXiv:2405.19166*, 2024.

[33] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, pages 1–97, 2023.

[34] Lizuo Liu, Kamaljyoti Nath, and Wei Cai. A causality-deeponet for causal responses of linear dynamical systems. *Communications in Computational Physics*, pages 1194–1228, 2024.

[35] Siavash Khodakarami, Vivek Oommen, Aniruddha Bora, and George Em Karniadakis. Mitigating spectral bias in neural operators via high-frequency scaling for physical systems. *arXiv preprint arXiv:2503.13695*, 2025.

[36] Ahmad Peyvan and Varun Kumar. Fusion deeponet: A data-efficient neural operator for geometry-dependent hypersonic flows on arbitrary grids. *arXiv preprint arXiv:2501.01934*, 2025.

[37] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, page eabi8605, 2021.

[38] Sanghyun Lee and Yeonjong Shin. On the training and generalization of deep operator networks. *SIAM Journal on Scientific Computing*, pages C273–C296, 2024.

[39] Ahmad Peyvan, Vivek Oommen, Ameya D. Jagtap, and George E. Karniadakis. Riemannonets: Interpretable neural operators for riemann problems. *Computer Methods in Applied Mechanics and Engineering*, page 116996, 2024.

[40] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-Arnold Networks. *arXiv preprint arXiv:2404.19756*, 2024.

[41] David A. Sprecher and Sorin Draghici. Space-filling curves and Kolmogorov superposition-based neural networks. *Neural Networks*, pages 57–67, 2002.

[42] Mario Köppen. On the training of a kolmogorov network. In *Artificial Neural Networks—ICANN 2002: International Conference, Madrid, Spain, August 28–30, 2002*, pages 474–479.

[43] Hao Xu, Wei Fan, Lecheng Ruan, Rundong Shi, Ambrose C Taylor, and Dongxiao Zhang. Crack-net: A deep learning approach to predict crack propagation and stress–strain curves in particulate composites. *Engineering*, 2025.

[44] Venkatesh Ananchaperumal, Srikanth Vedantam, and Mahendaran Uchimali. Modelling delamination of elastic layers from shape memory alloy substrates. *Shape Memory and Superelasticity*, pages 1–10, 2025.

[45] H. M. W. Westergaard. Bearing pressures and cracks. *Journal of Applied Mechanics*, 6:A49–A53, 1939.

[46] E. G. Kirsch. Die theorie der elastizität und die bedürfnisse der festigkeitslehre. *Zeitschrift des Vereines deutscher Ingenieure*, 42:797–807, 1898.

[47] E Giner, N Sukumar, JE Tarancón, and FJ Fuenmayor. An abaqus implementation of the extended finite element method. *Engineering fracture mechanics*, 76(3):347–368, 2009.

[48] Asher A Rubinstein. Mechanics of the crack path formation. *International Journal of Fracture*, 47:291–305, 1991.

[49] Ameya D Jagtap, Yeonjong Shin, Kenji Kawaguchi, and George Em Karniadakis. Deep Kronecker neural networks: A general framework for neural networks with adaptive activation functions. *Neurocomputing*, 468:165–180, 2022.